КОМПЬЮТЕРЛІК ЖЕЛІЛЕР
2

COMPUTER NETWORKS

FIFTH EDITION

ANDREW S. TANENBAUM
Vrije Universiteit
Amsterdam, The Netherlands

DAVID J. WETHERALL
University of Washington
Seattle, WA

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Эндрю С. Таненбаум
Дэвид Дж. Уэзеролл

КОМПЬЮТЕРЛІК ЖЕЛІЛЕР
2

Оқулық

Алматы, 2014

ӘОЖ 004.7 (075.8)
КБЖ 32.973 я 73
Т 18

Қазақстан Республикасы Білім және ғылым министрлігінің «Оқулық» республикалық ғылыми-практикалық орталығы бекіткен

Қазақ тіліне аударған:
Махметова Анар Мусаевна

Т 18 **Таненбаум Э., Уэзеролл Д.**
Компьютерлік желілер. Оқулық / Ауд. А. М. Махметова. – Алматы, 2014. – 532 бет.

ISBN 978-601-217-485-4
1-бөлім. – 532 б.
ISBN 978-601-217-486-1

Алдарыңызда қазіргі заманғы желілік технологиялар бойынша танымал эксперт Эндрю Таненбаумның Вашингтон университетінің профессоры Дэвид Уэзеролмен бірлесіп жазған беделді оқулығының бесінші басылымы. Қазіргі кезде классикаға айналған бұл оқулықтың алғашқы басылымы 1980 жылы жарық көрген болатын.

Оқулықта компьютерлік желілердің даму беталысы және қазіргі қалып-күйін анықтайтын негізгі тұжырымдамалар берілген. Авторлар программалық және аппараттық жабдықтамалардың жұмыс қағидаларын, құрылымын, желілер ұйымдастырылуының барлық тұстарын және деңгейлерін – физикалық деңгейден бастап қолданбалы деңгейге дейін егжей-тегжейлі түсіндірген. Теориялық қағидалардың баяндалуы Интернет және әртүрлі типтегі компьютерлік желілердің жұмысындағы айқын, тәжірибелік мысалдармен толықтырылған. Бесінші басылым желілік технологиялар саласындағы соңғы өзгерістерді, нақтырақ 802.12 және 802.16 сымсыз желі стандарттарын, 3G желілерін, RFID технологиясын, CDN контентін жеткізу инфрақұрылымын, пирингтік желілер, ағындық таратылымдар, интернет-телефония және тағы да басқа көптеген жаңалықтарды есепке алып қайта өңделген.

ӘОЖ 004.7 (075.8)
КБЖ 32.973 я 73

ISBN 978-601-217-486-1
ISBN 978-601-217-485-4 – (1 бөлім)

Сюзанаға, Барбараға, Даниэльге, Аронға, Марвинге, Матильдаға және Брэм мен Кішкентайдың естелігіне арналады (А.С.Т.)

Катринге, Люсиге және Пепперге (Д.Дж.У.)

МАЗМҰНЫ

6-ТАРАУ
ТРАНСПОРТТЫҚ ДЕҢГЕЙ

Транспорттық деңгей желілік деңгеймен бірлесіп бүкіл хаттамалар иерархиясының өзегін қалайды. Желілік деңгей дейтаграммалар мен виртуалды арна көмегімен дестелерді толассыз жеткізуді қамтамасыз етеді. Желілік деңгейге негізделген транспорттық деңгей, қолданылатын желінің физикалық сипаттамаларына байланыссыз қажет сенімділікті қамтамасыз ете отырып, деректерді ақпарат көзі машинасынан адресат машинасындағы үдеріске жеткізуге жауап береді. Ол қосымшалар желіде жұмыс істеуге керекті абстракцияларды құрастырады. Транспорттық деңгейсіз көпсатылы хаттамалар концепциясының маңызы жоғалады. Бұл тарауда біз транспорттық деңгейді оның көрсететін қызметтері, сенімділік, қосылу және асыра жүктелу мәселелерін шешетін қолайлы API схемасы, сонымен бірге хаттамалары (TCP және UDP тәрізді) және өнімділігін қоса толығымен қарастырамыз.

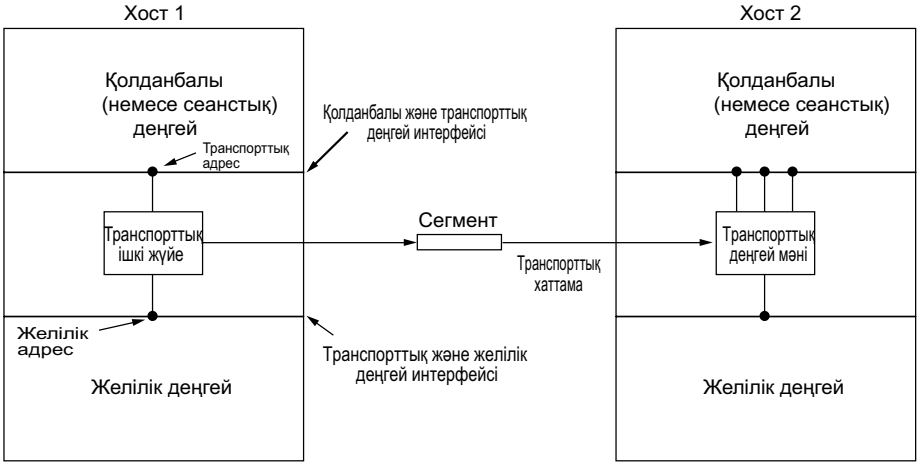
6.1. ТРАНСПОРТТЫҚ ҚЫЗМЕТ

Келесі бөлімдерде біз транспорттық қызметпен танысамыз. Біз қолданбалы деңгейге ұсынылатын қызмет түрлерін қарастырамыз. Біздің әңгіметіміз тым абстракты болмас үшін біз транспорттық деңгей базалық операцияларының екі жиынтығын талдаймыз. Алдымен негізгі ойларды көрсету үшін қарапайым (іс жүзінде қолданылмайтын) жиынтықты, ал сонан кейін интернетте нақты қолданылатын интерфейстерді қарастырамыз.

6.1.1. Жоғары деңгейлер ұсынатын қызметтер

Транспорттық деңгейдің соңғы мақсаты – өз тұтынушыларына, әдетте, ол қолданбалы деңгей үдерістері, деректерді тиімді, сенімді және үнемді тасымалдау қызметін көрсету. Мақсатқа жету үшін транспорттық деңгей

желілік деңгей ұсынған қызметтерге жүгінеді. Транспорттық деңгейдің жұмысын орындаушы программа және/немесе аппаратура **транспорттық ішкі жүйе** немесе **транспорттық объект (transport entity)** деп аталады. Транспорттық ішкі жүйе операциялық жүйенің ядросында, желілік қосымша жүктеген кітапхана модулінде, тұтынушының жеке үдерісінде немесе тіпті желілік интерфейс платасында орналасуы мүмкін. Алғашқы екі жағдай Интернет желісінде жиі кездеседі. Желілік, транспорттық және қолданбалы деңгейлер арасындағы логикалық байланыс *6.1-суретінде* бейнеленген.



6.1-сурет. Желілік, транспорттық және қолданбалы деңгейлер

Транспорттық деңгей қызметтері, желілік деңгей қызметтеріндей, байланыс орнатумен және байланыссыз көрсетілетін қызметтерге бөлінеді. Байланыс орнатумен көрсетілетін транспорттық қызмет сәйкес желілік қызметке көп жағынан ұқсас. Екі жағдайда да байланыс: байланыс орнату, деректеді тасымалдау және байланысты үзу сияқты үш сатыда жүреді. Адрестеу және әртүрлі деңгейде ағынды басқару да ұқсас. Одан бетер, әртүрлі деңгейдегі байланыссыз қызмет көрсету де ұқсас. Алайда, байланыссыз транспорттық қызмет және байланысы бар желілік қызметтің аралас түрін іске асыру өте қиын, себебі ол үшін дәл осы мезетте орнатылған байланысты десте жөнелтілісімен үзу керек болады.

Заңды сұрақ туындайды: егер транспорттық деңгей қызметі желілік деңгей қызметіне соншалықты ұқсас болса, онда екі деңгейдің керегі не? Бір деңгей не себепті жеткіліксіз? Бұл өте маңызды сұрақ. Транспорттық деңгей программалық жабдықтамасы толығымен тұтынушы машинасында іске қосылады, ал желілік деңгей көбіне байланыс операторы басқаратын (қалай болғанда да ауқымды желіде) маршруттауыштарда іске қосылады. Егер желілік деңгей байланысы бар қызмет ұсынса, бірақ бұл қызмет сенімсіз болса не болады? Егер дестелер жиі жоғалатын болса не болады? Егер маршруттауыштар дүркін-дүркін істен шығатын болса не болады?

Бұл жағдайда тұтынушылар үлкен мәселемен қақтығысады. Олар желілік деңгейді бақылай алмайды, сондықтан олар жақсы маршруттауышты пайдаланып немесе арналық деңгей қателігін өңдеуді жетілдіру арқылы (маршруттауыштар оларға тиесілі емес болғандықтан) нашар қызмет мәселесін шеше алмайды. Жалғыз мүмкіндік – қызмет сапасын жақсарту үшін желілік деңгей үстінде орналасқан тағы бір деңгейді пайдалану. Егер байланыссыз желіде дестелер жиі жоғалса немесе бұрмаланып тасымалданса, транспорттық ішкі жүйе келеңсіздікті анықтап, қайта тасымалдауды жүзеге асырады. Егер байланысы бар желіде транспорттық ішкі жүйе желілік байланыстың кенеттен үзілгенін білсе, ол осы уақытта жөнелтілген деректермен не болғаны жайлы мәлімет болмаса да, қашықтықтағы транспорттық ішкі жүйемен жаңа байланыс орната алады. Ол жаңа желілік байланыс көмегімен тең рангілі объектіге сұраныс жіберіп, қандай деректердің адресатқа жеткенін, қандай деректердің жетпегенін және үзілістің қай жерде орын алғанын анықтап, деректер тасымалдауды жалғастырады.

Шын мәнінде, транспортық деңгейдің болғандығының арқасында транспорттық қызмет төменгі желілік қызметке қарағанда әлдеқайда сенімді болуы мүмкін. Одан бетер, транспорттық қызметтің базалық операциялары желі түріне байланысты (мысалы, байланыссыз Ethernet қызметі WiMAX қызметінен әлдеқайда өзгеше болуы мүмкін) өзгеріп отыратын, желілік қызметтің базалық операцияларына тәуелсіз болатындай етіп құрастыруға болады. Егер желілік қызметті транспорттық қызмет базалық операцияларының артына жасырып қойсақ, онда желілік қызметті өзгерту үшін, дәл сол қызметті төменгі деңгейдің басқа қызметтері көмегімен орындайтын, бір кітапханалық процедуралар жиынын екіншісімен алмастыру жеткілікті болады.

Транспорттық деңгейдің бар болғандығының арқасында қолданбалы программалар стандартты базалық операциялар жиынын пайдаланып, өз жұмыс қабілеттілігін түрлі желілерде сақтай алады. Оларға түрлі желі интерфейстерін және сенімділік деңгейін есепке алудың қажеті жоқ. Егер барлық нақты желілер мінсіз жұмыс істесе және барлық желілерде кепілді түрде өзгермейтін базалық операциялар жиынтығы болса, онда мүмкін транспорттық деңгей қажет болмас еді. Алайда, нақты әлемде ол жоғары деңгейді технология, құрылғылар және жетілмеген желі егжей-тегжейінен оқшауландырады. Міне, осы себептен біріншіден төртінші деңгейлер және төртіншіден жоғары деңгейлер арасында шектеулер қойылады. Төменгі төрт деңгейді **транспорттық қызметті ұсынушылар (transport service provider)**, ал жоғары деңгейлерді **транспорттық қызметті тұтынушылар (transport service user)** ретінде қарастыруға болады. Ұсынушыға және тұтынушыға бөлу деңгей құрылғыларына елеулі әсер етеді және транспорттық деңгейді маңызды орынға орналастырады, себебі ол ұсынушы және тұтынушы арасындағы деректерді сенімді тасымалдау қызметін көрсететін негізгі шекараны құрайды. Қосымшаларға көрінетін нақты осы деңгей.

6.1.2. Транспорттық қызметтің базалық операциялары

Тұтынушылар транспорттық қызметке қол жеткізе алу үшін, транспорттық деңгей қолданбалы программаларға транспорттық қызмет интерфейсін ұсынуы тиіс. Барлық транспорттық қызметтің өз интерфейсі бар. Негізгі қағидалар мен түсініктерді білу үшін, біз бұл бөлімнің басында транспорттық қызметтің бір қарапайым (жорамалды түрде) мысалын және оның интерфейсін қарастырамыз. Келесі бөлім нақты мысалға арналады.

Транспорттық қызмет желілік қызметке ұқсас, бірақ кейбір ерекшеліктері бар. Негізгі айырмашылық – желілік қызмет нақты желілер көрсететін қызметтерді, олардың барлық ерекшеліктерімен қоса модельдеуге арналған. Нақты желілер дестелерді жоғалтуы мүмкін, сондықтан жалпы жағдайда желілік қызмет сенімсіз.

Байланыс орнату арқылы жүргізілетін транспорттық қызмет керісінше сенімді болып келеді. Әрине, нақты желілерде қателіктер бар, сондықтан дәл осы транспорттық деңгей сенімсіз желінің үстінде сенімді қызметті қамтамасыз етуі тиіс.

Мысал ретінде UNIX (немесе үдерістердің әрекеттесуін қамтамасыз ететін кез келген басқа құрал көмегімен) жүйесінде арнамен байланысқан, бір машинада жұмыс істейтін екі үдерісті қарастырайық. Бұл үдерістер олардың арасындағы байланыс мінсіз деп ойлайды. Олардың дестенің бүлінуі, жоғалуы, асыра жүктелу және т.б. жайлы білгілері келмейді. Оларға тек жүз пайыздық сенімді байланыс керек. А үдерісі деректерді арнаның бір басына орналастырады, ал екінші басында В үдерісі орын алады. Міне, нақты осы жағдай үшін байланыс орнатуы бар транспорттық қызмет қажет – желілік қызметтің кемшіліктерін жасыру – тұтынушы үдерісі, үдерістер әртүрлі машинада орындалса да, қателіксіз биттер ағыны бар деп есептеу үшін.

Сонымен бірге транспорттық деңгей сенімсіз қызмет түрін (дейтаграммалық) ұсынады, бірақ ол жайлы айтарлықтай еш нәрсе жоқ («бұл дейтаграмма» дегеннен басқа), сондықтан біз бұл тарауда байланыс орнатуы бар транспорттық қызмет жайлы айтамыз. Дегенмен, байланыссыз транспорттық қызметке негізделген қосымшалар бар, мысалы, клиент-серверлік есептеу жүйелері және ағындық мультимедиа, сондықтан төменде біз ол жайлы еске саламыз.

Желілік және транспорттық қызметтер арасындағы екінші айырмашылық – олардың кімге арналғандығы. Желілік қызметті тек транспорттық ішкі жүйелер қолданады. Өзінің жеке транспорттық ішкі жүйесін жазатындар өте аз, сондықтан тек желілік қызметтен тұратын тұтынушылар және программалар кездеспейді. Транспорттық қызметтің базалық операциялары, керісінше, көптеген программалар қолданады, демек, программалаушылар да қолданады. Сол себепті транспорттық қызмет қолдануға ыңғайлы және қолайлы болуы керек.

Транспорттық қызмет жайлы түсінік алу үшін *6.1-кестеде* атап көрсетілген негізгі бес базалық операцияны қарастырайық. Бұл транспорттық интерфейс қатты оңтайландырылған, бірақ байланыс орнатуы бар транспорттық қызметтің тағайындалуы жайлы түсінік береді. Ол қолданбалы програмаларға байланысты орнатуға, пайдалануға және босатуға мүмкіндік береді. Көптеген қосымшалар үшін бұл жеткілікті.

Бұл операцияларды қалай қолдану керек екенін түсіну үшін, серверден және қашықтықтағы бірнеше клиенттерден тұратын үдерісті қарастырайық. Бастапқыда сервер `LISTEN` операциясын орындайды, әдетте, бұл үшін жүйемен байланысатын кітапханалық процедура шақырылады. Нәтижесінде сервер клиент өзімен байланысқанша оқшауланады. Клиент сервермен сөйлескісі келгенде `CONNECT` операциясын орындайды. Транспорттық ішкі жүйе өзімен байланысқан клиентті оқшаулап, серверге десте жөнелту арқылы осы операцияны орындайды. Дестенің деректер өрісінде сервердің транспорттық ішкі жүйесіне арналған транспорттық деңгейдің мәлімдемесі орналасқан.

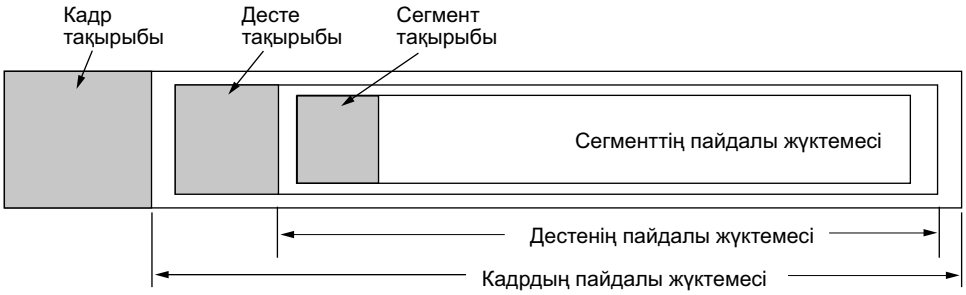
Терминология жайлы біраз сөз айтқан дұрыс болар деп санаймыз. Жақсы термин болмағандықтан, бір транспорттық ішкі жүйенің екіншісіне жөнелтетін мәлімдемесі үшін **сегмент (segment)** терминін қолданамыз. Бұл белгі TCP, UDP және басқа да интернет-хаттамаларда қолданылады. Ескі хаттамаларды біршама икемсіз **TPDU (Transport Protocol Data Unit – транспорттық хаттаманың деректер модулі)** атауы пайдаланылады. Қазір ол мүлдем қолданылмайды, алайда, сіз оны ескі мақалалардан және кітаптардан кездестіресіз.

6.1-кесте. Қарапайым транспорттық қызметтің базалық операциялары

Базалық операция	**Жөнелтілген сегмент**	**Мәні**
LISTEN (КҮТУ)	(жоқ)	Қандай да бір үдеріс байланысуға әрекет жасамағанша серверді оқшаулау
CONNECT (БАЙЛАНЫСТЫРУ)	CONNECTION REQUEST	Екпінді түрде байланыс орнатуға талпыну
SEND (ЖӨНЕЛТУ)	ДЕРЕКТЕР	Ақпарат жөнелту
RECEIVE (ҚАБЫЛДАУ)	(жоқ)	Деректер жеткенше серверді оқшаулау
DISCONNECT (АЖЫРАТУ)	DISCONNECTION REQUEST	Байланысты үзу

Транспорттық деңгей алмасатын сегмент дестеге (желілік деңгей алмасатын) енгізіледі. Бұл дестелер өз кезегінде арналық деңгей алмасатын кадрларда орналасады. Кадрды алғаннан кейін арналық деңгей үдерісі кадр тақырыбын өңдейді, егер тағайындалған адрес келген орынмен сәйкес келсе, кадрдың пайдалы жүктеме өрісінің мәнін жоғары, желілік ішкі жүйеге жібе-

реді. Дәл осылайша желілік ішкі жүйе десте тақырыбын өңдеп, дестенің пайдалы жүктеме өрісінің мәнін жоғары транспорттық ішкі жүйеге жібереді. Осындай кіріктірілгендік *6.2-суретінде* көрсетілген.



6.2-сурет. Сегменттер, дестелер және кадрлар кіріктірілуі

Сонымен, біздің клиент және сервер әрекеттесуі жайлы мысалымызға қайтып оралайық. Клиенттің `CONNECT` сұранысының нәтижесінде серверге `CONNECTION REQUEST` (байланысуға сұраныс) сегменті жөнелтіледі. Сегмент кезде транспорттық ішкі жүйе сервердің `LISTEN` операциясымен оқшауланғанын (сервер сұранысты өңдей алатындығын) тексереді. Сонан кейін ол сервердің оқшаулану қалпынан шығарып, клиентке `CONNECTION ACCEPTED` (байланыс қабылданды) сегментін жібереді. Клиент сегментті қабылдағаннан кейін оқшаулану қалпынан шығады, осы әрекеттен кейін байланыс орнатылды деп саналады.

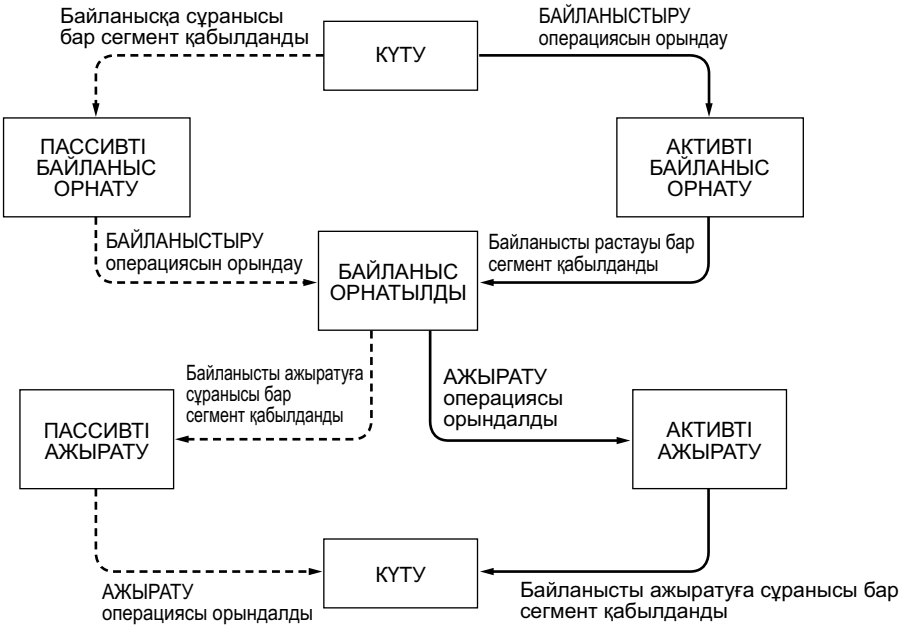
Енді клиент пен сервер `SEND` және `RECEIVE` базалық операцияларының көмегімен деректермен алмаса алады. Қарапайым жағдайда екі жақтың кез келгені `RECEIVE` оқшауландыру операциясының көмегімен екінші жақтан `SEND` операциясының көмегімен келетін сегментті күту режиміне ауыса алады. Сегмент келген кезде қабылдаушы оқшаулану режимінен шығады. Сонан кейін ол қабылданған сегментті өңдеп, жауап қайтарады. Бұл схема екі жақ кімнің қабылдау, кімнің жөнелту кезегі екенін естен шығармағанша жақсы жұмыс істейді.

Транспорттық деңгейде қарапайым бір бағыттық деректерді тасымалдау операцияның өзі желілік деңгейдегіден күрделі жұмыс істейтініне назар аударыңыз. Әрбір жөнелтілген деректер дестесі ақыры расталады. Басқарушы сегменті бар дестелерде айқын немесе жасырын түрде расталады. Бұл растаулар транспорттық ішкі жүйемен, желілік деңгей хаттамасының көмегімен басқарылады және транспорттық деңгей тұтынушыларына көрінбейді. Транспорттық ішкі жүйе сәйкесінше таймерді және қайта тасымалдауды қадағалауы тиіс. Бұл механизмдердің барлығы байланыс сенімі биттер арнасы болып сезілетін транспорттық деңгей тұтынушыларына беймәлім. Бір тұтынушы арнаға биттерді орналастырады, ал ол ғажайып түрде арнаның екінші басында сол жөнелтілген ретпен пайда болады. Осындай күрделі механизмді тұтынушыдан жасыра алу, көпдеңгейлі хаттамалардың қуатты құрал екенін дәлелдейді.

Байланыс қажет болмаған кезде оны үзіп, екі транспорттық ішкі жүйелер кестесінен басқаларға орын босату керек. Байланысты үзудің екі нұсқасы бар: симметриялы және асимметриялы. Асимметриялы нұсқада кез келген транспорттық қызмет тұтынушысы қарапайым `DISCONNECT` операциясын шақыра алады, нәтижесінде қашықтықтағы транспорттық объектіге `DISCONNECTION REQUEST` (байланысты үзуге сұраныс) басқарушы сегментті жөнелтіледі. Сегмент қабылданғаннан кейін қашықтықтағы транспорттық ішкі жүйемен байланыс үзіледі.

Симметриялы нұсқада әр бағыт бір-біріне байланыссыз жеке жабылады. Бір бетте `DISCONNECT` операциясы орындалып жатса, бұл оның басқа жөнелтетін деректері жоқ, бірақ ол әлі де өз әріптесінен деректер қабылдауға дайын дегенді білдіреді. Бұл схемада байланыс екі жақта `DISCONNECT` операциясын орындаған кезде үзіледі.

Байланысты орнату және үзу қалып-күй диаграммасында *6.3-суретте* көрсетілген. Әр алмасу транспорттық қызметтің жергілікті тұтынушысымен немесе кіріс дестенің орындаған қандай да бір оқиға-операциямен туындайды. Түсінікті болу үшін біз әр сегмент жеке расталады деп есептейміз. Сонымен бірге, біз бірінші қадамды клиент жасайтын байланысты симметриялы үзу моделі қолданылады деп болжаймыз. Кейіннен TCP жайлы айтқан кезде, нақты модельдерді қарастыратын боламыз.



6.3-сурет. Байланысты басқару қарапайым схемасының қалып-күй диаграммасы. Курсивпен белгіленген алмасулар дестенің келуімен туындайды. Тұтас сызықпен клиент қалып-күй тізбегі көрсетілген. Пунктирлі сызықпен сервер қалып-күй тізбегі көрсетілген.

6.1.3. Беркли сокеттері

Біз енді транспорттық деңгейдің басқа базалық операцияларын қарастырамыз – **TCP (Transmission Control Protocol – тасымалдауды басқару хаттамасы)** хаттамалары үшін қолданылатын сокеттердің базалық операциялары (кейде ұялар деп атайды). Сокеттер алғашқы рет 1983 жылы Berkely UNIX 4.2BSD операциялық жүйесінде қолданыла бастады. Ол тез арада танымал болып келеді, ал қазір ол интернет-программалауда көптеген операциялық жүйелерде, әсіресе, UNIX-де кеңінен пайдаланылуда. Сонымен бірге, Windows – "winsock" жүйесінде сокеттерді программалауға арналған арнайы API бар.

Бұл базалық операциялар *6.2-кестеде* келтірілген. Сокеттер моделі жоғарыда келтірілген моделге өте ұқсас, бірақ әлдеқайда икемді және ұсынатын мүмкіндіктері мол. Осы модельге сәйкес сегменттерді осы тараудың соңына қарай қарастырамыз.

Тізімдегі алғашқы төрт базалық операциялар дәл осы реті бойынша сервермен орындалады. `SOKET` операциясы жаңа сокет құрастырып, ол үшін транспорттық ішкі жүйе кестесінде орын тағайындайды. Шақыру параметрлері қолданылатын адрестер форматын, қажет қызмет түрін (мысалы, сенімді биттер ағыны) және хаттаманы көрсетеді. Сәттілік жағдайда `SOKET` операциясы, `OPEN` процедурасы файл үшін жұмыс істегендей, келесі операцияны шақыру кезінде пайдаланылатын әдеттегі файл сипаттамасын қайтарады.

6.2-кесте. TCP-ге арналған сокеттердің базалық операциялары

Базалық операция	Мәні
SOKET (СОКЕТ)	Жаңа сокет құрастыру (байланыс ұясы)
BIND (БАЙЛАУ)	Жергілікті адресті сокетке байлау
LISTEN (КҮТУ)	Байланысты қабылдау ықыласы жайлы хабарлау; кезек мөлшерін көрсету
ACCEPT (ҚАБЫЛДАУ)	Кіріс байланысты пассивті орнату
CONNECT (БАЙЛАНЫСТЫРУ)	Активті түрде байланыс орнатуға талпыну
SEND (ЖӨНЕЛТУ)	Байланыс деректерін жөнелту
RECEIVE (ҚАБЫЛДАУ)	Байланыстан деректер қабылдау
CLOSE (ЖАБУ)	Байланысты үзу

Жаңа құрастырылған сокеттің желілік адресі жоқ. Адрес `BIND` операциясының көмегімен тағайындалады. Сервер сокетке адрес тағайындағаннан кейін онымен қашықтықтағы клиенттер байланыса алады. `SOKET` шақыруы адресті тікелей құрастырмайды, себебі кейбір үдерістер адреске үлкен мағына береді (мысалы, олар бір адресті жылдар бойы пайдаланды және ол адрес бәріне белгілі), ал кейбіріне ол маңызды емес.

Артынша LISTEN шақыруы жүреді. Ол бірнеше клиент бірмезгілде байланысуға талпынған жағдайда кіріс байланыстарына кезектен орын береді. Біздің алғашқы мысалдағы LISTEN операциясынан айырмашылығы, сокет моделіндегі LISTEN операциясы оқшаулаушы шақыру болып саналмайды.

Кіріс байланыстардың күтуін оқшаулау үшін сервер ACCEPT операциясын орындайды. Байланысқа сұранысы бар сегмент алғаннан кейін транспорттық ішкі жүйе қасиеттері бастапқы сокеттікіндей жаңа сокет құрастырады да, ол жайлы сипаттама файлын қайтарады. Осы кезде сервер жаңа сокет үшін, байланысты өңдеу үшін үдеріс немесе ағынды тармақтап, бірегей сокет үшін келесі байланысты күтуге көшеді.

Енді осы үдерісті клиент тұрғысынан қарастырайық. Бұл жағдайда да алдымен SOKET операциясының көмегімен сокет құрастырылуы керек, бірақ бұнда BIND операциясы қажет емес, себебі қолданылатын адрестің сервер үшін еш мағынасы жоқ. CONNECT операциясы шақырушыны оқшаулап, активті байланысу үдерісін іске қосады. Бұл үдеріс аяқталғаннан кейін (демек, сервер жөнелткен сәйкес сегмент қабылданғаннан кейін) клиент үдерісі оқшаулаудан шығарылады, байланыс орнатылды деп саналады. Бұдан кейін екі жақта жартылай дуплексті байланыс арқылы деректерді жөнелтіп, қабылдау үшін SEND және RECEIVE операцияларын пайдалана алады. Сонымен бірге, егер SEND және RECIEVE арнайы қасиеттерін пайдалану қажет болмаса, стандартты UNIX-шақырулары READ және WRITE қолдануға болады.

Сокет моделдерінде байланысты симметриялы үзу қолданылады. Байланыс екі жақта CLOSE операциясын орындаған кезде үзіледі.

Сокеттер кеңінен таралып, іс жүзінде транспорттық қызметтің қосымшалар үшін абстракциялану стандартына айналды. **Сенімді байттар ағыны** (іс жүзінде бұл біз жоғарыда айтқан сенімді биттер арнасы) деп аталатын байланысты орнату қызметтерін ұсыну үшін сокет-API жиі TCP-хаттамаларымен бірге қолданылады. Бұл API басқа да хаттамалармен үйлесе алады, алайда, барлық жағдайда транспорттық қызмет тұтынушылары үшін нәтиже бірдей болу керек.

Сокет-API-дің артықшылығы – қосымшалар оны басқа да транспорттық қызметтер үшін пайдалана алады. Мысалы, сокеттер көмегімен байланыссыз транспорттық қызметті жүзеге асыруға болады. Бұл жағдайда CONNECT операциясы қашықтықтағы түйіннің адресін, ал SEND және RECIEVE дейтаграммаларды жөнелтіп, қабылдайды (Кейде шақырудың кеңейтілген жиынтығы қолданылады, мысалы, қосымшаға бір транспорттық түйінмен шектелуге мүмкіндік беретін SEND және RECIEVEFROM операциялары.). Сокеттер кейде байттық ағын орнына асыра жүктелуді басқаруды қосатын немесе қоспайтын мәлімдемелер ағыны қолданылатын транспорттық хаттамалармен бірге пайдаланылады. Мысалы, UDP нұсқасы болып келетін **DCCP (Datagram Congestion Control Protocol – асыра жүктелуді басқаруы бар дейтаграммалық хаттама)** асыра жүктелуді басқаруды қосады (Rohler және басқалар, 2006). Қандай қызметті таңдау керек екендігін алдымен транспорттық қызмет тұтынушылары шешеді.

Дегенмен де транспорттық интерфейс сауалындағы соңғы нүктені сокеттер қоя алмайды. Көп жағдайда қосымшаларға байланысқан ағындар тобымен жұмыс істеуге тура келеді, мәселен, браузер бірмезгілде серверден бірнеше объектіні сұрауы мүмкін. Бұндай жағдайда сокеттерді қолдану, әр объект үшін бір ағын пайдалану керек және нәтижесінде асыра жүктелуді басқару әр ағын үшін жеке (бүкіл топ үшін емес) орындалады дегенді білдіреді. Әрине, бұл тиімді нұсқа емес, себебі ағындар жиынын басқару қосымшаға жүктеледі. Байланысқан ағындар тобын тиімді өңдеп, бұл үдерістегі қосымша рөлін азайту үшін жаңа хаттамалар мен интерфейстер құрастырылды. Мысал ретінде RFC 4960 сипатталған **SCTP (Stream Control Transmission Protocol – ағындарды басқаруы бар тасымалдау хаттамасы)** және **SST (Structured Stream Transport – деректерді иерархиялық ағындық транспорттау)** келтіруге болады (Ford, 2007). Бұл хаттамалар ағындар тобымен жұмыс жасау ыңғайлы болу үшін сокет-API-ді сәл өзгертеді және аралас трафикпен жұмыс істеу (байланыс орнатуы және байланыссыз) тәрізді жаңа мүмкіндіктерді қамтамасыз етеді. Бұның қаншалықты сәтті болғанын уақыт өте білетін боламыз.

6.1.4. Сокетті программалау мысалы: Интернетке арналған файл-сервер

Нағыз шақырулар қалай орындалатынын білу үшін, *6.1-листингінде* көрсетілген клиент және сервер жұмысын бейнелейтін программаны қарастырайық. Интернетте жұмыс жасайтын қарапайым сервер және оны қолданатын клиент бар. Программада көптеген шектеулер бар (олар жайлы біз әлі айтамыз), бірақ бұл кодты Интернетке қосылған кез келген UNIX-жүйеде компеляциядан өткізіп, орындауға болады. Клиентті сипаттайтын кодты белгілі бір параметрлермен орындауға болады. Бұл оған сервер қол жеткізе алатын кез келген файлды алуына мүмкіндік береді. Файл стандартты шығару құрылғысында бейнеленеді, дискіге немесе қандай да бір үдеріске қайта бағыттауға да болады.

6.1-листинг. Клиент және сервер үшін сокетті пайдалану программасы

```
/* Бұл парақта келесі парақта орналасқан серверлік программадан файл сұрайтын
клиенттік программа берілген. Сервер сұранысқа жауап ретінде файл жөнелтеді. */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345     /* Клиент және сервер арасындағы келісім
                                 бойынша */
```

```
#define BUF_SIZE 4096              /* Жөнелтілетін блок мөлшері */
int main(int argc, char *argv)
{
int c, s, bytes;
char buf [BUF_SIZE];               /* кіріс файлына арналған буфер */
struct hostent *h;                 /* сервер жайлы ақпарат */
struct sockaddr_in channel;        /* IP-адресті сақтау */

if (argc != 3) fatal («Іске қосу үшін: client сервер_аты файл_аты енгізіңіз”);
h = gethostbyname(argv[1]);        /* хост IP-адресін іздеу */
if (!h) fatal(«gethostbyname орындау қателігі”);

s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (s <0) fatal(“Сокет”);
memset(&channel, 0, sizeof(channel));
channel.sin_family = AF_INET;
memcpy(&channel.sin_addr.s_addr, h>h_addr, h>h_length);
channel.sin_port = htons(SERVER_PORT);

c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
if (c < 0) fatal(«Байланысу қателігі”);

/* Байланыс орнатылды. Соңында нөлдік байты бар файл аты жөнелтіледі.
*/
write(s, argv[2], strlen(argv[2])+1);

/* Файлды қабылдап, стандартты шығару құрылғысына жазу. */
while (1) {
      bytes = read(s, buf, BUF_SIZE);              /* Сокеттен оқу */
if (bytes <= 0) exit(0);           /* Файл соңын тексеру */
write(1, buf, bytes);              /* Стандартты шығару құрылғысына жазу */
   }
}
fatal(char *string)
{
   printf(“%s\n”, string);
   exit(1);
}

/* Сервер программасының коды */
#include <sys/types.h>
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define SERVER_PORT 12345          /* Сервер және клиент арасындағы
                                      келісім бойынша */
```

```
#define BUF_SIZE 4096                    /* Жөнелтілетін блоктар мөлшері */
#define QUEUE_SIZE 10
int main(int argc, char *argv[])
{
  int s, b, l, fd, sa, bytes, on = 1;
  char buf[BUF_SIZE];                    /* шығыс файлына арналған буфер */
  struct sockaddr_in channel;            /* IP-адрес жазылған */

/* Сокетке байлау үшін адрес құрылымын жасау */
memset(&channel, 0, sizeof(channel));              /* channel нөлді жазу */
channel.sin_family = AF_INET;
channel.sin_addr.s_addr = htonl(INADDR_ANY);
channel.sin_port = htons(SERVER_PORT);

/* Пассивті режим. Байланысты күту. */
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);     /* Сокет жасау */
if (s < 0) fatal(«Сокет қателігі”);
setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
if (b < 0) fatal(“Байлау қателігі”);

l = listen(s, QUEUE_SIZE);                 /* Кезек мөлшерін анықтау */
if (l < 0) fatal(«Күту қателігі”);

/* Енде сокет құрастырылып, байланды. Күту және байланысты өңдеу. */
while (1) {
   sa = accept(s, 0, 0);             /* Байланысқа сұраныстыкүтуді оқшаулау */
   if (sa < 0) fatal(«Қол жеткізу қателігі”);

   read(sa, buf, BUF_SIZE);                /* Сокеттен файл атын оқу */

   /* Файлды қабылдап, қайтару. */
   fd = open(buf, O_RDONLY);               /* Файлды жөнелту үшін ашу */
   if (fd < 0) fatal(«Файлды ашу қателігі”);

   while (1) {
        bytes = read(fd, buf, BUF_SIZE); /* Файлдан оқу */
        if (bytes <= 0) break;                    /* Файл соңын тексеру */
        write(sa, buf, bytes);                    /* Байттарды сокетке жазу */
       }
   close(fd);                                     /* Файлды жабу */
   close(sa);                                     /* Байланысты үзу */
  }
}
```

Алдымен программаның серверді сипаттайтын бөлігін қарастырайық. Ол кейбір стандартты тақырыптарды қосудан басталады. Соңғы үшеуі Интернетпен байланысты негізгі құрылымдар мен анықтамалардан тұрады. Сонан кейін SERVER_PORT 12345 түрінде анықталады. Бұл мән кездейсоқ алынған. 1024-тен 65535 арасындағы кез келген сан жарайды, тек егер ол қандай да басқа үдеріспен қолданылмаса, нөмірі 1023-тен төмен порттар артықшылығы бар тұтынушылар үшін қорда сақталады.

Келесі екі жолда серверге қажет екі тұрақты анықталады. Оның біріншісі файлдарды жөнелтуге қажет деректер блогының мөлшерін (байтпен) анықтайды, екіншісі, аяқталмаған байланыстардың ең үлкен санын анықтайды. Бұл байланыстар орнатылғаннан кейін жаңа байланыстар қабылданбайды.

Жергілікті айнымалылар хабарланғаннан кейін сервер программасының өзі басталады. Бастапқыда ол сервер IP-адресінде жазылатын деректер құрылымын инициализациялайды. Бұл құрылым сервер сокетімен байланыста болады. Memset шақыруы деректер құрылымын толығымен нөлге айналдырады. Келесі үш меншіктеу осы құрылымның үш өрісін толтырады. Соңғысында сервер порты жазылған. Htonl және htons функциялары мәндерді стандартты форматқа түрлендірумен айналысады. Бұл программаға сандық разрядтарды little-endian (мысалы, Intel x86) түрінде ұсынатын машинада да, big-endian (мысалы, SPARC) түрінде ұсынатын машинада да қалыпты орындауға мүмкіндік береді. Олардың семантикаларының егжей-тегжейі маңызды емес.

Бұдан кейін сервер сокетті құрастырып, қателіктерге (s<0 арқылы тексеріледі) тексереді. Программаның соңғы версиясында қателік жайлы мәлімдеме сәл түсініктілеу болуы мүмкін. Setsockopt шақыруы портты бірнеше рет, ал серверді сұраныстарды бірінен соң бірін өңдеп шексіз қолдану үшін қажет. Енді IP-адрес сокетке байланады да bind шақыруының табысты аяқталуы тексеріледі. Инициялизациялаудың соңғы сатысы listen шақыруы болып саналады. Бұл шақыру, сервердің кіріс шақырулар қабылдауға дайын және жүйеге сервер ағымдағы шақыруды өңдеп жатқан кезде, шақыруларды QUEUE_SIZE-ге дейін кезекке қою керек екендігі жайлы мәлімдейді. Кезек толған кезде жаңа келген сұраныстар қабылданбайды.

Осы тұстан программаның ешуақытта тоқтамайтын негізгі цикл басталады. Оны тек сырттан тоқтатуға болады. Accept шақыруы серверді тек клиент байланыс орнатуға талпынатын уақытқа ғана оқшаулайды. Егер шақыру сәтті аяқталса, accept сокет дискрипторын (сипаттамасын) қайтарады. Бұл дискрипторды файл дискрипторларын арнаға жазу және оқу үшін пайдаланғандай, жазу және оқу үшін қолдануға болады. Алайда, сокеттің бір бағытталған арнадан айырмашылығы ол екі бағытта жұмыс істейді, сондықтан байланыстан деректерді оқу (немесе жазу) үшін sa (жағымды сокетті) пайдалануға болады. Арнаның файлдық дискрипторлары оқу немесе жазу үшін пайдаланылуы мүмкін, бірақ бір мезгілде емес.

Байланыс орнатылғаннан кейін сервер файл атын оқиды. Егер ол қолжетімсіз болса, сервер оқшауланып оны күтеді. Файл аты алынғаннан кейін

сервер оны ашып, файлдан деректер блогын оқып, оны сокетке жазатын циклге енеді. Бұл барлық сұралған деректер жазылып біткенше жалғасады. Сонан кейін файл жабылып, байланыс үзіледі және жаңа шақыруды күту басталады. Бұл цикл шексіз қайталанады.

Енді клиентті сипаттайтын программа кодын қарастырайық. Программаның қалай жұмыс жасайтынын түсіну үшін, алдымен, оның қалай іске қосылатынын білу керек. Егер ол client деп аталса, онда оның әдеттегі шақырылуы төмендегідей:

```
Client flits.csvu.nl   /usr/tom/filename >f
```

Бұл шақырылу тек сервер *files.cs.vu.nl* адресі бойынша орналасса, /usr/tom/filename файлы бар және оған сервер оқу үшін қолжеткізе алса ғана жүзеге асырылады. Егер шақыру сәтті жүргізілсе, файл Интернет арқылы жөнелтіліп, f орнына жазылады. Бұдан кейін клиент программасы өз жұмысын аяқтайды. Серверлік программа өз жұмысын жалғастыратын болғандықтан клиент программасы файл алуға жаңа сұраныспен қайта іске қосылуы мүмкін.

Клиент программасы файлдарды іске қосу және хабарламалардан басталады. Жұмыс аргументтар санының дұрыстығын тексеруден басталады (arc=3 іске қосу жолында программа аты және екі аргумент болды дегенді білдіреді). Назар аударыңыз, argv[1]-де сервер аты жазылған (мысалы, *flits.cs.vu.nl*) және ол gethostbyname көмегімен IP-адреске түрлендіріледі. Функция сервер атын іздеу үшін DNS пайдаланады. Біз DNS технологиясын *7-тарауда* қарастырамыз.

Бұдан кейін сокет құрастырылып, инициализацияланады. Клиент connect арқылы сервермен TCP-байланыс орнатуға талпынады. Егер сервер іске қосылып, SERVER_PORT-пен байланысқан және көрсетілген машинада жұмыс істеп тұрса немесе бос тұрса немесе listen (күту кезегі) кезегінде бос орын болса, онда клиентпен байланыс ерте ме, кеш пе орнатылады. Бұл байланыс арқылы клиент файл атын, сокетке жазу арқылы жөнелтеді. Жөнелтілген байттар саны файл атын жөнелтуге бөлінген саннан бірге артық, себебі нөлдік шектеуіш байт қажет. Бұл байт көмегімен сервер файл атының қай жерде аяқталатынын анықтайды.

Енді клиент программасы циклге енеді. Файлды сокеттен блок блогымен оқып стандартты шығару құрылғысына көшіреді. Үдеріс аяқталған кезде программа да тоқтайды.

fatal процедурасы қателік жайлы мәлімдеме шығарады да аяқталады. Бұл процедура серверге де қажет, ол тек орынды үнемдеу тұрғысынан ғана қалдырылып кеткен. Клиент және сервер пограммалары жеке компиляцияланатын және әдеттегі жағдайда әртүрлі машинада іске қосылатын болғандықтан fatal процедурасының коды бөлінетін болмауы керек.

Бұл екі программаны (кітаппен байланысты басқа материалдар тәрізді) кітаптың

http://www.prenhall.com/tanenbaum

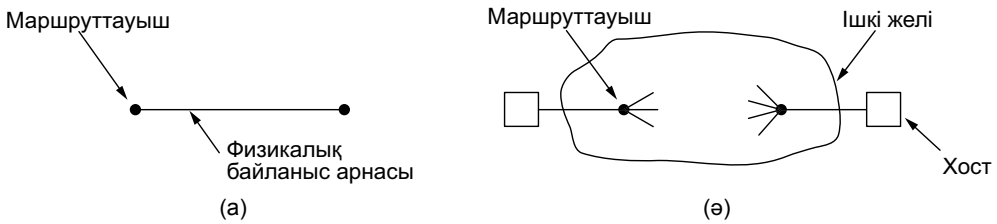
адресі бойынша орналасқан веб-сайттан табуға болады.

Бір айта кететіні, мұндай сервер техникадағы соңғы жаңалықтар негізінде тұрғызылмаған. Тексерілетін қателіктер өте аз, ал қателіктер жайлы мәлімдеме тым қарапайым жасалған. Жүйенің өнімділігі төмен болады, себебі барлық сұраныстар тек кезек кезегімен (бір сұраныстар ағыны пайдаланылады) өңделеді. Ақпараттарды қоғау тіпті қарастырылмаған, ал UNIX жүйелік шақыруларын пайдалану – бұл платформаға тәуелсіз болуға қол жеткізудің ең жақсы шешімі емес. Техникалық тұрғыдан біршама дөрекі болжамдар жасалады. Мысалы, файл аты әрқашан буферге сыяды және қателіксіз жеткізіледі. Осындай кемшіліктерге қарамастан бұл программа көмегімен Интернетке арналған толыққанды жұмыс істейтін файл-сервер ұйымдастыруға болады. Толық ақпаратты Donahoo және Calvert (2008, 2009) басылымынан қарауға болады.

6.2. ТРАНСПОРТТЫҚ ХАТТАМАЛАР ЭЛЕМЕНТІ

Транспорттық қызметтер екі транспорттық ішкі жүйе арасында қолданылатын **транспорттық хаттамалармен** жүзеге асырылады. Кейбір қатынастарда транспорттық хаттамалар *3-тарауда* егжей-тегжейлі қарастырылған арналық деңгей хаттамаларын еске салады. Екі хаттамада басқа сұрақтармен бірге қателіктерді өңдеумен, кезек және ағынды басқарумен айналысады.

Алайда, олардың көптеген елеулі айырмашылықтары да бар. Бұл айырмашылықтар *6.4-суретте* көрсетілгендей хаттамалардың әртүрлі жағдайда жұмыс істейтініне байланысты. Арналық деңгейде екі маршруттауыш бірбірімен тікелей физикалық арна (сымды немесе сымсыз) арқылы әрекеттеседі. Ал транспорттық деңгейде физикалық арна үлкен бір желімен алмастырылған. Бұл айырмашылық хаттамаларға маңызды әсерін тигізеді.



6.4-сурет. а – арналық деңгейдің; ә – транспорттық деңгейдің қоршауы

Біріншіден, сым немесе оптыталшық торабы арқылы екі нүктелі байланысу кезінде маршруттауышқа өзінің қандай маршруттауышпен сөйлескісі келетінін көрсету қажет емес – әр шығыс торабы белгілі бір маршруттауышқа апарады. Транспорттық деңгейде қабылдаушы адресін нақты түрде көрсету қажет.

Екіншіден, сым бойынша байланыс орнату үдерісі (*6.4 а-суреті*) қарапайым: қарсы бетте әр кез құрылғы бар (егер ол істен шықпаса). Кез келген

жағдайда жұмыс көп емес. Сымсыз байланыстыру кезінде жағдай мүлдем басқаша. Барлық тағайындалған адрестер бойынша жай ғана мәлімдемені жөнелту жеткілікті. Егер мәлімдеменің қабылданғаны жайлы растау келмесе (қателік салдарынан), мәлімдеме қайта жөнелтілуі мүмкін. Транспорттық деңгейде бастапқы байланысты орнату төменде келтірілгендей өте күрделі жүргізіледі.

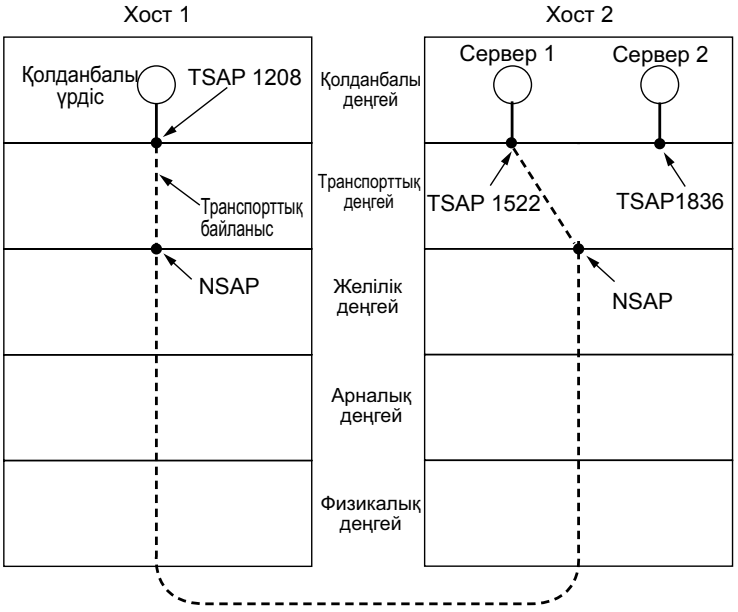
Арналық және транспорттық деңгей арасындағы тағы бір өкінішті айырмашылық – желінің ақпаратты сақтауға әулетті мүмкіндігі бар. Маршруттауыш дестені байланыс арнасы бойынша жөнелткен кезде ол жетуі немесе жолда жоғалып кетуі мүмкін. Бірақ ол біраз уақыт Жер шарын шарлап, сонан соң кездейсоқ пайда болып, тағайындалған адреске өзінен кейін жөнелтілген дестелерден кейін келуі мүмкін емес. Егер желі өз ішінде тәуелсіз маршрутталатын дейтаграммалар пайдаланатын болса, онда дестенің қандай да бір ғажайып жолмен жүріп, тағайындалған адреске белгіленген уақыттан кейін және дұрыс емес ретпен келу ықтималдығы бар. Сонымен бірге дестенің дубликаттары да алынуы мүмкін. Желінің дестелерді кідіртіп, оның көшірмесін жасау мүмкіндігінің салдары кейде апатты болуы мүмкін және деректердің дұрыс тасымалдануын қамтамасыз ететін арнайы хаттамаларды пайдалануды қажет етеді.

Арналық және транспорттық деңгейлер арасындағы соңғы айырмашылық сапалық түрден сандық түрге жақын. Ағындарды буферлеу және басқару екі деңгейде де қажет, бірақ транспорттық деңгейде өткізгіштік қабілеттілігі байланыс бәсекелестігі салдарынан өзгеріп отыратын байланыстардың көп болуы, арналық деңгейде қолданылатын амалдардан басқа амалды қажет етпеуі мүмкін. *3-тарауда* айтылған кейбір хаттамалар әр торап үшін буфердің тұрақты санын бөледі, сондықтан кадр келген кезде үнемі бос буфер бар. Транспорттық деңгейде басқарылатын байланыстар көп және өткізгіштік қабілеттілік өзгермелі болғандықтан, әр байланысқа бірнеше буфер бөлу тиімсіз. Келесі бөлімде біз осы және басқа да сұрақтарды қарастырамыз.

6.2.1. Адрестеу

Бір қолданбалы үдеріс басқа қолданбалы үдеріспен байланыс орнатуға талпынған кезде нақты кіммен байланысқысы келетінін көрсетуі керек. (Байланысты талап ететін транспорттық қызметтің мәселесі «әр мәлімдемені кімге жіберу керек?» дегенмен бірдей). Әдетте, қолданылатын әдіс, үдерістер байланыс орнатуға сұранысты жөнелтетін транспорттық адресті анықтау болып табылады. Интернетте бұндай соңғы нүктелер **порттар** деп аталады. Біз транспорттық деңгейдегі соңғы нүктені анықтау үшін **TSAP (Transport Service Access Point – транспорттық деңгей қызметтеріне қол жеткізу нүктесі)** терминін пайдаланамыз. Желілік деңгейдің сәйкес нүктелері (желілік деңгей адрестері) **NSAP (Network Service Access Point – желілік қызметтерге қолжеткізу нүктесі**) деп аталады. IP-адрестер NSAP мысалы болып келеді.

NSAP, TSAP және транспорттық байланыс арасындағы өзара әрекеттестік *6.5-суретте* бейнеленген. Клиенттің және де сервердің де қолданбалы үдерістері жергілікті TSAP-мен қашықтықтағы TSAP-мен байланыс орнату үшін байланыса алады. Мұндай байланыстар *6.5-суретте* көрсетілгендей әр хосттағы NSAP арқылы өтеді. Әр компьютердің жеке NSAP-сы бар желіде TSAP бірге пайдаланылатын NSAP соңғы нүктесін анықтау үшін қажет.



6.5-сурет. Транспорттық және желілік қызметтерге және транспорттық байланысқа қол жеткізу нүктелері

Транспорттық байланыс үшін мүмкін деген сценарий келесідей болады:

1. Пошта серверінің үдерісі 2-хосттағы TSAP 1522 қол жеткізу нүктесімен байланысып, кіріс шақыруды күтеді. Үдерістің TSAP-мен қалай байланысатындығы желілік модельге қатыссыз және толығымен операциялық жүйеге байланысты. Мысалы, LISTEN тәрізді қарапайым операцияны шақыруға болады.
2. 1-хосттың қолданбалы үдерісі пошталық мәлімет жөнелткісі келеді, сондықтан ол TSAP 1208 байланысады. Сонан кейін ол желіге жөнелтуші адресі ретінде 1-хосттағы TSAP 1208, ал қабылдаушы адресі ретінде 2-хосттағы TSAP 1522-ні көрсетіп, CONNECT сұранысын жөнелтеді. Бұл әрекеттің нәтижесінде қолданбалы үдеріс және сервер арасында транспорттық байланыс орнатылады.
3. Қолданбалы үдеріс пошталық мәлімдеме жөнелтеді.

4. Пошталық сервер мәлімдеменің жеткізілетінін хабарлайды.
5. Транспорттық байланыс үзіледі.

2-хостта өз TSAP-мен байланысып, сол NSAP-тан келетін байланысқа кіріс сұранысты күтіп отырған басқа да серверлер орналасқан.

Жоғарыда көрсетілген сурет барлық жағынан жақсы, тек бір кішкене сұрақ сыртта қалып тұр: 1-хосттағы тұтынушы үдерісі пошталық сервердің TSAP 1522-мен байланысқанын қалай біледі? Мүмкін пошталық сервер TSAP 1522-ге ұзақ жылдар бойы байланысатын шығар және тұтынушылар бұл жайлы біртіндеп біле бастайды. Бұл жағдайда қызметтердің файлда жазылып, белгілі бір орында сақталатын тұрақты TSAP-адрестері бар. Мысалы, UNIX-жүйелердің `/etc/services` адресі бойынша серверлер тізімі және оларға бекітілген порттар көрсетілген, дәлірек онда пошталық сервердің TCP 25 портын пайдаланатындығы көрсетілген.

Тұрақты TSAP-адрестер ешуақытта өзгермейтін шағын қызметтер (мысалы, веб-сервер) үшін жақсы. Жалпы жағдайда тұтынушы үдерістері TSAP-адрестері алдын ала белгісіз немесе қысқа уақыт аралығында ғана жұмыс істейтін басқа тұтынушы үдерістерімен қатынасқысы келеді.

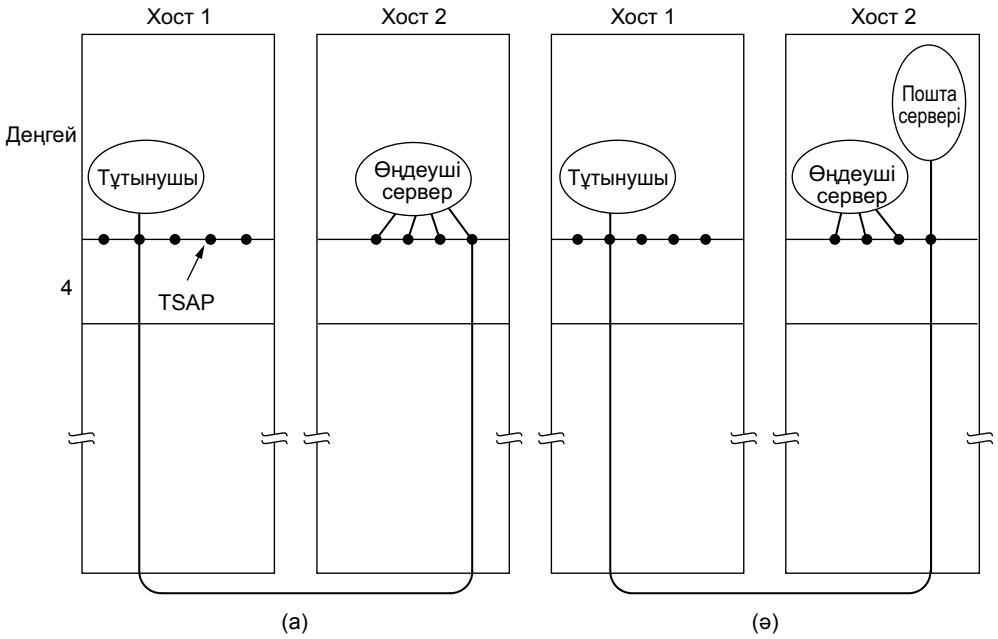
Бұл мәселені шешу үшін басқа амалды қолдануға да болады. Бұл модельде **порттарды сәйкестендіру (portmapper)** деп аталатын арнайы үдеріс қолданылады. Қызмет түріне сәйкес TSAP-адресті табу үшін, мысалы, “Bit Torrent” – тұтынушы порттарды сәйкестендірушімен (TSAP-адресі бәріне белгілі) байланысады. Сонан кейін тұтынушы өзіне қажет қызмет түрінің атын көрсетіп мәлімдеме жөнелтеді, ал порттарды сәйкестендіруші оған осы қызметтің TSAP-адресін хабарлайды. Бұдан кейін тұтынушы порттарды сәйкестендірушімен байланысты үзіп, өзіне қажет қызметпен жаңа байланыс орнатады.

Бұл модельде, жаңа қызмет құрастырылған кезде ол порттарды сәйкестендірушіде тіркеліп, оған қызмет түрінің атын (әдетте, ол ASSCII жолы) және TSAP-адресін хабарлауы тиіс. Порттарды сәйкестендіруші болашақтағы сұраныстарға жауап беру үшін алынған ақпаратты өзінің деректер қорында сақтайды.

Порттарды сәйкестендіруші функциясы телефон анықтама қызметі жұмысына ұқсас – аттарды нөмірлерге түрлендіреді. Телефон жүйесіндегідей порттарды сәйкестендірушінің (немесе бастапқы байланыс хаттамасындағы өңдеуші сервердің) TSAP-адресі шын мәнінде, белгілі болуы керек. Егер сіз телефон анықтама қызметінің нөмірін білмесеңіз операторға қоңырау шала алмайсыз. Егер сіз анықтама нөмірі айқын деп болжасаңыз, басқа елде жүріп оны табуға тырысасыз.

Нақты машинада орналасқан серверлік үдерістердің көбі сирек пайдаланылатын болғандықтан олардың барлығын тұрақты TSAP-адресімен активті режимде қолдап отыру тым бейберекет іс. Балама нұсқа *6.6-суретте* көрсетілген. Ол **бастапқы байланыс хаттамасы (initial connection protocol)**

деп аталады. Барлық серверлерге белгілі TSAP-адрес тағайындай бергенше, қашықтықтағы тұтынушыларға қызмет көрсеткісі келген әр машина арнайы **өңдеуші сервер (process server)** иеленеді. Аса активті емес сервер үшін ол прокси тәрізді қызмет етеді. UINX-жүйелерінде мұндай сервер *inetd* деп аталады. Байланысқа сұранысты күтіп, ол бір мезгілде бірнеше порттарды тыңдайды. Тұтынушылар алдымен TSAP-адресін және қызмет түрін көрсетіп CONNECT сұранысын жөнелтеді. Егер оларды ешбір сервер күтпесе, онда олар *6.6 а-суретінде* көрсетілгендей өңдеуші сервермен байланысады.



6.6-сурет. 1-хосттың тұтынушы үдерісі 2-хост пошталық серверімен өңдеуші сервер арқылы байланыс орнатады

Сұранысты алғаннан кейін өңдеуші сервер сұранған сервердің ішкі үдерісін туындатып, оған тұтынушымен орнатылған байланысты мұрағаттауға мүмкіндік береді. Жаңа сервер қажет жұмысты орындайды, бұл кезде өңдеуші сервер *6.6 ә-суретінде* көрсетілгендей басқа сұраныстарды күту режиміне ауысады. Бұл тәсіл тек серверлер талап бойынша құрастырылатын жағдайда жұмыс істейді.

6.2.2. Байланыс орнату

Байланыс орнату жеңіл естілгенімен іс жүзінде мүлдем оңай емес болып келеді. Алғаш қарағанда транспорттық ішкі жүйеге адресатқа CONNECTION REQUEST байланысқа сұраныс сегментін жөнелтіп, жауап ретінде

CONNECTION ACCEPTED (байланыс қабылданды) есту жеткілікті болып көрінеді. Өкінішке орай, желі деселерді жоғалтуы, кідіртуі немесе қайталауы мүмкін.

Желі асыра жүктелген және растау еш уақытта дер кезінде келмейтін, әр десте кешігіп, екі-үш рет қайталап жөнелтілетін жағдайды елестетіңіз. Айталық желі дейтаграммаларға негізделген және әр десте өз маршрутымен жүреді делік. Кейбір дестелер кептелісте кідіріп, кешігіп жөнелтуші оны жоғалда деп есептеген кезде келеді.

Ең нашар сценарий төмендегідей болып келеді. Тұтынушы банкпен байланыс орнатып, ірі көлемдегі қаражатты сенімсіз бір адамның шотына аудару жайлы мәлімдеме жібереді. Өкінішке орай десте желінің түкпір-түкпірін аралап серуендеуді шешеді. Бұл уақыт ішінде жөнелтуші дестені қайталап жөнелтуге мәжбүр болады. Бұл жолы ол ең қысқа жолмен жүріп уақытылы жетеді де, тұтынушы байланысты үзеді.

Кезекті сәтсіздік орын алады: дестелердің алғашқысы серуендеп болып қажет ретімен келіп жетіп, банкке жаңа байланыс орнатып (қайталап) ақша аударуды талап етеді. Банктің сұраныстың қайталанып отырғанын анықтауға мүмкіндігі жоқ. Ол бұның екінші тәуелсіз транзакция екен деп шешеді де тағы да ақша аударады.

Мұндай сценарий мүмкін емес, тіпті, шындыққа жатпайтын болып көрінгенімен негізгі ой мынада: хаттамалар кез келген жағдайда дұрыс жұмыс істеуі тиіс. Жиі кездесетін жағдайлар желі өнімді жұмыс істеу үшін ең үлкен тиімділікпен жүзеге асырылуы тиіс. Алайда, хаттамалар сонымен бірге сирек кездесетін жағдайларда да істен шықпай дұрыс жұмыс істеуі керек. Кері жағдайда желі сенімсіз және кездейсоқ істен шығып, ол жайлы хабар бермеуі де мүмкін.

Бұл бөлімнің қалған бөлігінде кідіріп қалған дубликаттар мәселесін қарастырамыз. Осының ішінде сенімсіз байланыс орнататын алгоритмдерге аса көңіл бөлеміз. Негізгі мәселе кідіріп қалған дубликаттардың жаңа десте ретінде танылатындығында. Дестелердің дубликаттарынан және кідірісінен құтыла алмаймыз. Алайда, егер бұл жағдай орын алатын болса, онда десте дубликаттары шектетіліп, жаңа десте ретінде өңделмеуі керек.

Мәселені бірнеше амалмен шешіп көруге болады және оның бірі шын мәнінде қанағаттанарлық болып табылады. Мысалы бір реттік транспорттық адрестерді қолдану керек. Бұл жағдайда әр кез транспорттық адрес керек болғанда жаңа адрес генерацияланады. Байланыс үзілген кезде адрес жойылады. Нәтижесінде кідірген десте дубликаттары транспорттық адресті таба алмайды, демек қауіп төндірмейді. Алайда, бұл жағдайда үдеріспен байланыс орнату қиындайды.

Басқа бір амал, әр байланысқа бірегей идентификатор (әр орнатылған байланыс үшін бірге өсіп отыратын тізбектік нөмір) беріледі. Идентификаторды байланыс орнатуға талпынған жақ таңдап, байланысқа сұраныс орналасқан сегменттен бастап, әр сегментке орналастырады. Әр байланыс үзілгеннен

кейін әрбір транспорттық ішкі жүйе ескірген байланыстар жұбымен (біррангілі транспорттық ішкі жүйе және байланыс идентификаторы) жазылған кестені жаңарта алады. Әрбір келген сұраныс үшін байланыс идентификаторы кестеде жазылған ба, жоқ па тексеріледі (ол кестеде ертеде үзілген байланыстан қалуы мүмкін).

Өкінішке орай, схеманың кемшілігі бар: әр транспорттық ішкі жүйе анықталмаған уақыт аралығында байланыстар тарихы жайлы ақпаратты сақтауы тиіс, сонымен бірге бұл ақпаратты жөнелтуші де, қабылдаушы да сақтауы керек. Кері жағдайда, егер машина істен шықса және өз жадысын жоғалтса, ол қандай байланыстың қолданылғанын, қайсының қолданылмағанын анықтай алмайды.

Бұның орнына мәселені жеңілдететін басқа амал қолдануға болады. Ескірген және адасқан дестелерді жойып, олардың желіде шексіз өмір сүруіне жол бермейтін механизм құрастыруға болады. Осындай шектеу арқасында мәселені басқаруға мүмкіндік туады.

Дестенің өмір уақыты ең үлкен белгілі бір мәнге дейін:

1. Желіні шектеулермен жобалау;
2. Әр дестеге транзитті аумақтар санауышын орнату;
3. Әр дестеге уақытша штамп орналастыру сияқты тәсілдердің бірімен шектеледі.

Бірінші тәсілге мүмкін деген ең ұзақ жолдағы асыра жүктелуді қоса алғанда, кідірісті шектеп, дестенің қайталануына жол бермейтін кез келген әдіс жатады. Көп ұлтты желінің бір қаладан бастап бүкіл әлемді қамтитынын ескерсек, бұл ойды іске асыру өте қиын. Екінші әдіс – санауышқа қандайда бір бастапқы мәнін енгізіп, оны әр маршруттауышта бірге кемітіп отыру. Деректерді тасымалдаудың желілік хаттамасы санауышының мәні нөлге жеткен барлық дестелерді елемейді. Үшінші әдіс – әр дестеге оның құрастырылған уақыты орналастырылады, ал маршруттауыштар белгілі бір уақыттан үлкен дестелерді елемеуге келіседі. Соңғы әдіс үшін маршруттауыштар сағатын синхрондау қажет. Бұл аса қиын есеп емес. Іс жүзінде дестенің жасын транзиттік аумақтар санауышының көмегін дәл есептеуге болады.

Іс жүзінде дестенің өлгендімен бірге оның барлық растауларының өлгендігіне кепілдік беру керек. Сол себепті дестенің ең үлкен өмір уақытынан бірнеше есе үлкен T оралымы енгізіледі. Әр желі үшін дестенің ең үлкен өмір уақыты – бұл артығымен алынған тұрақты. Интернет үшін ол 120 с құрайды. Дестенің ең үлкен өмір уақытының нешеге көбейтілетіндігі хаттамаға байланысты және ол тек T уақыт оралымының ұзақтығына әсер етеді. Егер дестe жөнелтілген уақыттан T уақыт оралымы аралығында күткеннен кейін оның барлық ізінің жойылғандығына және оның ашық аспандағы найзағайдай бір жерден шыға келмейтіндігіне сенімді болуға болады.

Дестелердің өмір уақыты шектеулі болған кезде кідіріп қайталанған сегменттерді елемеудің сенімді және қолайлы тәсілін құрастыруға болады. Төменде сипатталған тәсілдің авторы – Томлинсон (Tomlinson, 1975). Кейіннен бұл тәсілді Саншайном (Sunshine) және Далалом (Dalal) жақсартты. Тәсілдің нұсқалары іс жүзінде кеңінен қолданылады және оның бір мысалы TCP.

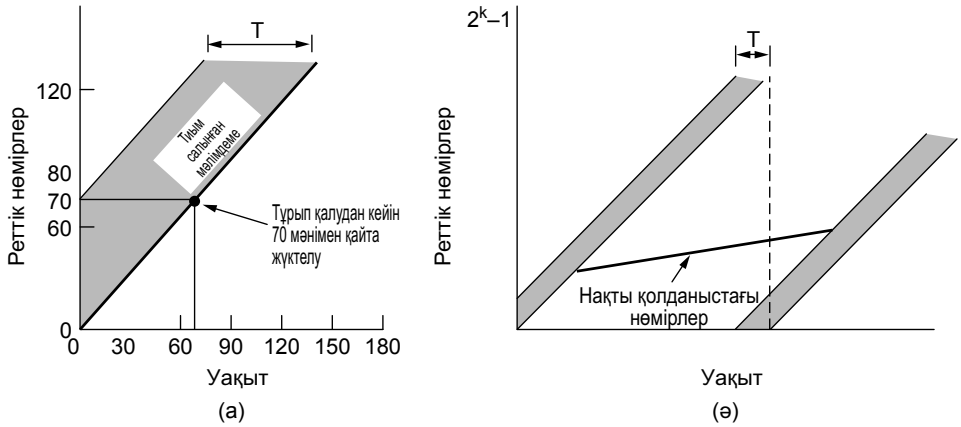
Тәсілдің негізгі идеясы – жөнелтуші сегменттерге келесі T секунд ішінде қайталап қолдануға болатын тізбектік нөмір тағайындайды. Мұндай нөмір мөлшері осы T оралымнан және дестелердің бір секундтағы жылдамдығынан құралады. Сөйтіп, кез келген уақыт сәтінде осындай нөмірі бар тек бір дестe жөнелтіледі. Бұл дестенің көшірмесі пайда болуы мүмкін, бірақ қабылдаушы оны жойып жіберуі тиіс. Алайда, енді келесідей жағдай туындауы мүмкін: дестенің кідірген дубликатын қабылдаушы дәл сондай нөмірі бар жаңа дестe ретінде қабылдауы мүмкін.

Машина жадысындағы алдыңғы қалып-күй мәліметтерінің жоғалу (істен шыққан кезде) мәселесін айналып өту үшін, қайта қалыпқа келгеннен кейін транспорттық ішкі жүйені алғашқы T секунд ішінде екпінсіз күйде қалдыру керек. Бұл жағдайда барлық ескі сегменттер жоғалып, жөнелтуші кез келген тізбектік нөмірмен үдерісті қайта бастай алады. Бұл ойдың кемшілігі ірі интержелілерде T уақыт оралымы тым үлкен болуы мүмкін.

Бұның орнына Томлинсон әр хостты сағатпен қамтамасыздандыруды ұсынды. Әртүрлі хосттардың сағаттарын синхрондау міндетті емес. Сағат уақыттың тең оралымынан кейін ұлғайып отыратын екілік санауыш болады деп болжалды. Бұдан басқа, санауыштың разрядттар саны тізбектік нөмірдегі биттер санына тең (немесе одан көп) болуы тиіс. Соңғы және ең маңызды болжам, хост істен шыққан жағдайда да сағат жүріп тұруы тиіс.

Байланыс орнатылған кезде сағаттың кіші k биттері алғашқы k-биттік реттік нөмір ретінде қолданылады. Сөйтіп, *3-тарауда* сипатталған хаттамалардан ерекшелік – әр байланыс өз сегменттерінің нөмірін әртүрлі саннан бастайды. Бұл нөмірлер диапозоны реттік нөмірлер толық айналым жасап, осындай нөмірлері бар ескі сегменттер жойылғанша жеткілікті болатындай үлкен болуы керек. Реттік нөмірлердің уақытқа сызықтық тәуелділігі *6.7-суретте* көрсетілген. Сегменттердің белгілі бір реттік нөмірлерінің қай кезде жарамсыз болғандығын тыйым салынған аймақтан көруге болады. Осы аумаққа кіретін реттік нөмірі бар сегментті жөнелтпек болған кезде ол кідіріп, тура осындай нөмірі бар, кейін жөнелтілетін басқа дестенің рөлін атқаруы мүмкін. Мысалы, егер хост істен шығып, 70 с кейін жұмысын қайта жалғастырса, ол сағат көрсетілімінің негізінде бастапқы реттік нөмірлерді пайдаланады. Хост есептеуді тыйым салынған аймақтағы ең кіші реттік нөмірден бастайды.

Екі транспорттық ішкі жүйе бастапқы реттік нөмірлер жайлы келіскеннен кейін деректер ағынын басқару үшін сырғымалы терезенің кез келген хаттамасы қолданылады. Мұндай хаттама дестелердің көшірмесін қабылданғаннан кейін дұрыс анықтап, жояды. Сағат көрсетілімі дискретті өзгеретін болғандықтан, іс жүзінде реттік нөмірлер кестесі (қалың сызықпен көрсетілген) түзу емес сатылы. Қарапайымдылық үшін біз бұны есепке алмаймыз.



6.7-сурет. Сегменттер тыйым салынған аймаққа кіре алмайды (а); ресинхронизациялау мәселесі (ә)

Дестелердің реттік нөмірі тыйым салынған аймаққа түспес үшін келесі жағдайға назар аудару керек. Хаттамада мынадай екі себеп бойынша қолайсыз жағдай туындауы мүмкін. Егер хост тым жылдам және тым көп деректер жөнелтетін болса, іс жүзінде қолданылатын реттік нөмірлер қыйсығы бастапқы нөмірлердің уақытқа тәуелді сызығынан әлдеқайда тіке болады. Нәтижесінде реттік нөмір тыйым салынған аймаққа түседі. Бұл жағдайды болдырмас үшін, әрбір ашық байланыста деректер тасымалдау жылдамдығы бір уақыт бірлігінде бір сегментпен шектелуі тиіс. Сонымен бірге бұл транспорттық ішкі жүйе қайта қалыпқа келгеннен кейін жаңа байланысты ашпас бұрын, бір нөмір екі рет қолданылмас үшін, сағаттың қалып-күйін өзгерткенін күтуі тиіс дегенді білдіреді. Демек, сағат қалып-күйінің өзгеру аралығы қысқа болуы керек (1 мкс немесе одан да аз). Алайда, сағат тым жылдам да (реттік нөмірлерге қатысты) жүрмеу керек. Егер жылдамдық C-ға тең, ал реттік нөмірлер кеңістігінің мөлшері *S* болса, реттік нөмірлер толық айналымды тым тез жасамас үшін $S/C > T$ шарты міндетті.

Тыйым салынған аймаққа деректерді тым жылдам жөнелтіп тек төменнен ғана кірмейді. *6.7 ә-суретінде* көрсетілгендей сағат жылдамдығынан кіші деректер тасымалдаудың кез келген жылдамдығында, іс жүзінде қолданылатын реттік нөмірлер қыйсығы тыйым салынған аймаққа сол жақтан, реттік нөмірлер толық айналым жасағаннан кейін кіреді. Осы қисықтың көлбеуі неғұрлым тіке болса, бұл оқиғаны соғұрлым ұзақ күтуге болады. Жағдайды болдырмас үшін реттік нөмірлердің байланыс үшін қозғалысын (немесе байланыстың өмір уақытын) шектеуге болады.

Сағат көрсетілімін пайдаланатын тәсіл сегменттердің кідірген дубликаттарын және жаңа сегменттерді ажырату мәселесін шешеді. Алайда, бұл жағдайда байланыс орнатылған кезде қиындық туындауы мүмкін. Әдетте, қабылдаушы түрлі байланыстардағы реттік нөмірін есте сақтамайтын бол-

ғандықтан, ол қандай да бір бастапқы реттік нөмірі бар CONNECTION REQUEST сегменті алдыңғы байланыстардың бірінің көшірмесі екенін ажырата алмайды. Ағымдағы байланыс үшін мұндай мәселе туындамайды, себебі сырғымалы терезе хаттамасы ағымдағы реттік нөмірді біледі.

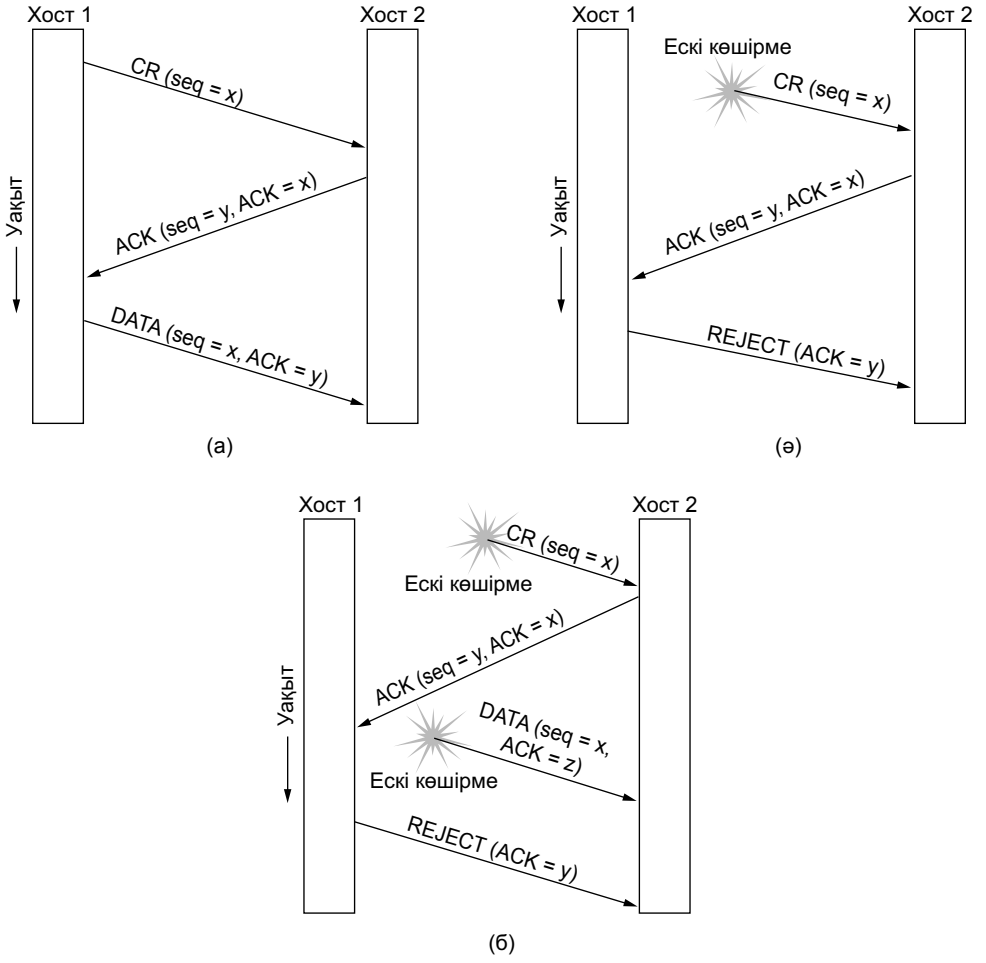
Осындай айрықша мәселені шешу үшін Томлинсон (1975) **үштік қол алысуды (three-way handshake)** ұсынды. Бұл байланыс орнату хаттамасы үш жақтың бірі байланыстың әлі де қолданыста екенін тексереді деп болжайды. Байланыс орнатудың қалыпты процедурасы *6.8 а-суретінде* көрсетілген. Байланыс орнатуға талпынған 1-хост х реттік нөмірін таңдап, осы реттік нөмір жазылған CONNECTION REQUEST сегментін 2-хостқа жөнелтеді. 2-хост *х*-ті растап, өзінің *у* бастапқы реттік нөмірін хабарлап, АСК сегментімен жауап қайтарады. Соңында 1-хост алғашқы жөнелтілетін ақпараттық сегментте 2-хосттың таңдап алған бастапқы реттік нөмірін растайды.

Енді «үштік қол алысудың» басқарушы сегменттің кідірген көшірмесі бар болған кездегі жұмысын қарастырып көрейік. *6.8 ә-суретіндегі* бірінші сегменті – ескі байланыстан қалған CONNECTION REQUEST сегментінің кешіккен көшірмесі. Бұл сегмент 2-хостқа 1-хосттан жасырын келеді. 2-хост бұл сегментке жауап ретінде 1-хостқа АСК сегментімен жөнелтеді, сөйтіп, 1-хосттың шынымен жаңа байланыс орнатқысы келгенін растауын сұрайды. 1-хост расстау жіберуден бас тартқан кезде 2-хост өзінің кідірген дубликатпен алданғанын сезіп байланысты үзеді. Осылайша кідірген дубликат зиян тигізбейді.

Ең нашар жағдайда екі сегментте – CONNECTION REQUEST және АСК ішкі желіде адасып жүреді. Бұл жағдай *6.8 б-суретінде* көрсетілген. Осы тұста 2-хосттың осы мезетте желіде у реттік нөмірі бар сегменттің немесе олардың растауы жоқ екенін біле отырып, 2-хосттан 1-хостқа жөнелтілетін трафиктің бастапқы реттік нөмірі ретінде у-ті пайдалануды ұсынғанын айта кету керек. 2-хост екінші кідірген сегментті алған кезде ол оның дубликат екенін біледі, себебі бұл модульде у емес *z* расталады. Мұнда, хаттаманы қателестіріп, кездейсоқ ешкімге керек емес байланыс орнатқызатын сегменттер комбинациясының жоқ екенін түсінген дұрыс.

TCP байланыс орнату үшін «үштік қол алысуды» пайдаланады. Байланыс ішінде 32-биттік реттік нөмірге уақыт белгісі қосылады. Бұл реттік нөмірдің дестенің ең ұзақ өмір уақыты ішінде, тіпті байланыс жылдамдығы секундына бірнеше гигабит болса да оны қайталанып қолданбау үшін қарастырылған. Бұл механизм TCP-ға шапшаң тораптарды пайдаланғанда орын алатын мәселелерді шешу үшін қосылған. Ол RFC 1323 құжатында сипатталған және **PAWS (Protection Against Wrapped Sequence numbers – реттік нөмірлерді қайталап қолдануды детектрлеу)** деп аталады. PAWS пайда болғанша бірнеше байланыспен жұмыс істеу үшін TCP хаттамасы сағат көрсетілімін қолданатын тәсілді (жоғарыдан қара) пайдаланған. Алайда, бұл әдіс қауіпсіздік тұрғысынан тиімсіз болып шықты. Сағатты пайдалану арқасында зиянкестер келесі реттік нөмірді жеңіл анықтап, «үштік қол алысу» схемасын алдап, жалған

байланысты ынталандырып отырды. Сондықтан іс жүзінде жалған кездейсоқ реттік нөмірлер қолданылады. Мұндай реттік нөмірлер сырт көзге абсолютті кездейсоқ болып көрінгенімен, олардың белгілі бір уақыт аралығында қайталанбағаны дұрыс. Әйтпесе, кідірген дубликаттар желі жұмысында үлкен қиындықтар туындатады.



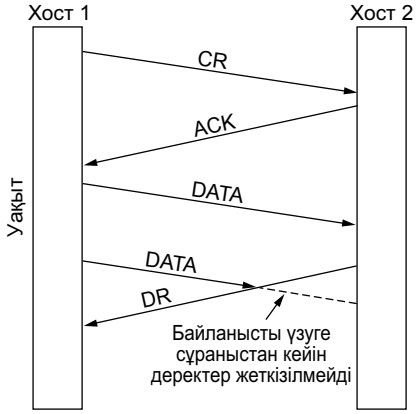
6.8-сурет. «Үштік қол алысу» көмегімен байланыс орнататын үш сценарий (CR – CONNECTION REQUEST): (а) қалыпты жұмыс; (ә) CR ескі көшірмесінің пайда болуы; (б) CR сегментінің және ACK көшірмесі

6.2.3. Байланысты үзу

Байланысты үзу оны орнатудан жеңіл. Дегенмен, мұнда да күтпеген кедергілер бар. Жоғарыда айтылғандай байланысты үзудің: ассимметриялық және симметриялық сияқты екі стилі бар. Байланысты ассимметриялық үзу телефон

жүйесінің жұмыс принципіне сәйкес келеді: сөйлесіп отырған екі жақтың бірі трубканы ілген кезде байланыс үзіледі. Симметриялы үзу кезінде байланыс екі жеке бірбағыттық байланыс ретінде қарастырылады және әр байланысты жеке үзу керек.

Байланысты ассимметриялық үзу кенеттен болады және деректердің жоғалуына әкелуі мүмкін. *6.9-суретте* көрсетілген сценарийді қарастырайық. Байланыс орнатылғаннан кейін 1-хост сегмент жөнелтіп, ол екінші хостқа сәтті жетеді. Бұдан кейін 1-хост басқа сегмент жөнелтеді. Өкінішке орай 2-хост екінші сегмент келіп жеткенше DISCONNECTION REQUST (үзуге сұраныс – DR) мәлімдемесін жібереді. Нәтижесінде байланыс үзіліп, деректер жоғалады.



6.9-сурет. Кенеттен үзіліп, деректер жоғалу

Демек, деректердің жоғалуына жол бермейтін күрделі хаттама қажет. Байланысты деректерді жоғалтпай үзудің бір жолы – симметриялы үзу. Бұнда әр байланыс бір-біріне тәуелсіз үзіледі. Бұл жағдайда хост өзі байланысты үзуге сұраныс жібергеннен кейін де деректерді қабылдауды жалғастыра береді.

Симметриялы үзу, әр жақтың бекітілген көлемдегі тасымалданатын деректері бар және оның қай кезде бітетінін әркім өзі нақты білетін жағдайда өте қолайлы. Басқа жағдайда жұмыстың аяқталғанын және байланысты үзуге болатынын анықтау оңай емес. 1-хост «Мен аяқтадым. Сіз аяқтадыңыз ба?» деп сұрайтын хаттаманы елестетіп көріңіз. Егер 2-хост «Мен де аяқтадым. Сау болыңыз», – деп жауап қайтарса, ешбір күмәнсіз байланысты үзуге болады.

Өкінішке орай, бұл хаттама барлық жағдайда жұмыс істемейді. Екі армия мәселесі деп аталатын танымал мәселе бар. Ақтар армиясы *6.10-суретте* көрсетілгендей алқапта орналасты делік. Алқаптың екі жағындағы жотада көктердің екі армиясы орналасқан. Ақтар армиясы кез келген көк армиядан артық, бірақ көктердің жалпы саны ақтардан көп. Егер көк армияның бірі ақтарға жеке шабуыл жасаса ол жеңіліп қалады, алайда, көктер ақтарға бір мезгілде шабуыл жасаса, олардың жеңіп шығуы мүмкін.

6.10-сурет. Екі армия мәселесі

Көк армия өз қимылдарын синхрондағысы келеді. Алайда, байланысудың жалғыз жолы – алқап арқылы жаяу жүргіншімен мәлімдеме жөнелту. Жол бойында ол ұсталып, мәлімдеме жоғалуы мүмкін (демек, сенімсіз арнаны пайдалануға тура келеді). Мынадай сұрақ туындайды: көктер армиясының жеңіп шығуына көмектесетін хаттама бар ма?

Айталық, көктердің бірінші армиясының командирі: «29 наурыз күні таңсәріде шабуыл жасайық. Өз пікіріңізді айтыңыз», - деген мәлімдеме жөнелтті делік. Енді мәлімдеме сәтті жетті және көктердің 2-армиясының командирі келісті, ал оның жауабы көктердің бірінші армиясына ойдағыдай жеткізілді деп болжайық. Шабуыл болады ма? Шабуылдың болмауы ықтимал, себебі көктердің 2-армия командирі жауаптың сәтті жеткеніне сенімсіз. Егер жауап жетпесе, демек 1-армия шабуылға шықпайды, сондықтан жалғыз шабуылға шығу ақмақтық болар еді.

Енді хаттаманы «үштік қол алысу» арқылы жетілдірейік. Бірегей ұсыныс жасаушы жауапқа растау жөнелту керек. Бірақ бұл жағдай да соңғы мәлімдеменің жеткендігі белгісіз болып қалады. Бұл жағдайда төрттік қол алысу да көмектеспейді.

Іс жүзінде бұл мәселені шешетін хаттаманың жоқ екендігін дәлелдеуге болады. Айталық, бұндай хаттама шын мәнінде, бар делік. Бұл жағдайда хаттаманың соңғы мәлімдемесі маңызды ма, жоқ па? Егер ол маңызды болмаса, онда оны (сонымен бірге басқа да маңызсыз мәлімдемелерді) барлық мәлімдемелері маңызды хаттама қалғанша жояйық. Егер соңғы мәлімдеме адресатқа жетпесе не болады? Бір қазір ғана мәлімдеменің маңызды екенін айттық, сондықтан ол жоғалса, шабуыл болмайды. Соңғы мәлімдемені жөнелтуші ешуақытта оның сәтті жеткеніне сенімді болмайтындықтан ол тәуекелге бармайды. Көктердің басқа армиясы да бұны біледі, сондықтан олар да шабуылдан бас тартады.

Екі армия мәселесінің байланысты үзуге қандай қатысы бар екенін білу үшін «шабуыл» сөзін «үзу» сөзімен алмастырыңыз. Егер екі жақтың біріншісі

екіншісінің байланысты үзуге дайын екеніне сенімді болғанша байланысты үзбесе, онда байланыс ешуақытта үзілмейді.

Іс жүзінде бұндай жағдайды болдырмас үшін келісуден бас тартып, барлық жауапкершілікті транспорттық қызметті тұтынушыларға жүктеу керек. Операцияның қашан орындалатындығы жайлы әрқайсысы тәуелсіз шешім қабылдауы тиіс. Бұл мәселені шешу оңай. *6.11-суретте* «үштік қол алысуды» пайдаланып байланысты үзудің төрт сценарийі көрсетілген. Бұл хаттама қателіксіз болмаса да, әдетте сәтті жұмыс істейді.

6.7 а-суретінде байланысты үзуді инициализациялау үшін бір тұтынушы екіншісіне DR (DISCONNECTION REQUST) сұранысын жөнелтетін әдеттегі жағдай көрсетілген. Сұраныс жеткен кезде қабылдаушы да байланысты үзу DR сұранысын жөнелтіп, сұраныс жоғалып кеткен жағдай үшін таймерді іске қосады. Сұраныс жеткен кезде бірінші жөнелтуші оған жауап ретінде АСК растауы бар сегмент жөнелтіп, байланысты үзеді. Соңында, АСК жеткен кезде қабылдаушы да байланысты үзеді. Байланыстың үзілуі транспорттық ішкі жүйе өзінің ашық байланыстар кестесінен осы байланыс жайлы ақпаратты жойып, байланыс иесіне (транспорттық қызмет тұтынушысы) байланыстың үзілуі жайлы хабарлайды дегенді білдіреді. Бұл процедура тұтынушы DISCONNECT операциясын қолданатын жағдайдан ерекше.

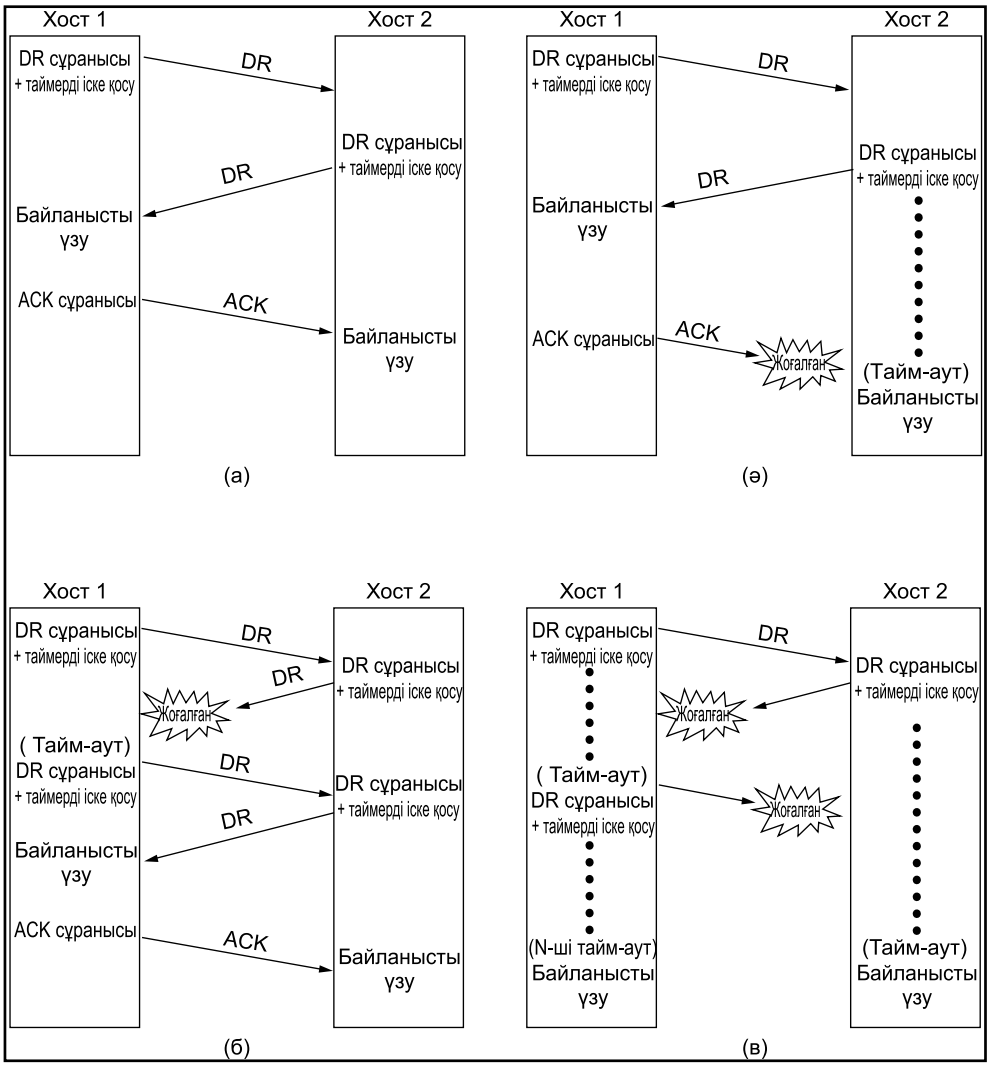
Егер растауы бар соңғы сегмент жоғалса (*6.11 ә-сурет*), жағдайды таймер құтқарады. Белгіленген уақыт өткеннен кейін байланыс бәрі бір үзіледі.

Енді байланысты үзудің екінші DR сұранысы жоғалған жағдайды қарастырайық. Байланысты үзуге сұраныс жіберген тұтынушы күткен жауапты алмайды. Оның күту уақыты аяқталып, ол бәрін қайта бастайды. Бұл жағдайдың, кейіннен жіберілген басқа сұраныстар және растаулар адресатқа сәтті жеткен кезде қалай болатыны *6.11 б-суретінде* көрсетілген.

Соңғы сценарий (*6.11 в-суреті*) алдыңғы жағдайға ұқсас, тек бұл жағдайда қайталап жіберілген байланысты үзу DR сұраныстары сәтсіздікке ұшырайды, себебі сегменттердің барлығы жоғалады. *N* қайта талпыныстан кейін жөнелтуші байланысты үзеді. Бұл уақыт ішінде қабылдаушының да уақыты аяқталып, ол да байланысты үзеді.

Осындай хаттама жеткілікті болғанымен, теорияда егер бастапқы байланысты үзу DR сұранысы және *N* қайта талпыныстар жоғалса, ол қателесуі мүмкін. Жөнелтуші байланысты үзеді, ал қабылдаушы екінші жақтық байланысты үзуге талпынғанын білмегендіктен, байланысты сақтап, екпінді күйінде қалады. Нәтижесінде жартылай ашық байланыс орын алады.

Егер жөнелтушіге *N* қайта талпыныстан кейін шегінбей, жауап алынғанша талпынысын қайталауға мәжбүрлесек, бұл жағдай орын алмас еді. Алайда, егер таймер бойынша байланысты үзуге рұқсат берілмесе, онда жөнелтуші талпынысын шексіз қайталайтын болады, себебі ол ешуақытта жауап алмайды. Егер қабылдаушы жаққа да байланысты таймер бойынша үзуге рұқсат берілмесе, онда хаттама *6.11 в-суретінде* бейнеленген жағдайда үнемі қалып қояды.



6.11-сурет. Байланысты үзудің төрт сценарийі: а – «үштік қол алысудың» қалыпты жағдайы; ә – соңғы растау жоғалған; б – жауап жоғалған; в – жауап және кейінгі сұраныстар жоғалған

Жартылай ашық байланыстарды жою үшін «егер байланыс бойымен белгіленген уақыт аралығында ешбір сегмент тасымалданбаса, онда байланыс автоматты түрде үзіледі», – деген ережені пайдалануға болады. Сөйтіп, егер бір жақ байланысты үзсе, онда екінші жақ екпінсіздікті байқап байланысты үзеді. Бұл ереже де байланыс бір жақтың бастауымен емес, байланысқа қатысушы желі хосттарының арасында дестелер тасымалдау мүмкін емес болған жағдайда орын алады. Ережені жүзеге асыру үшін екі жақта әр сегмент жөнелтілгеннен кейін қайталап іске қосылатын таймерді басқаруы қажет. Егер бұл

таймер жарамсыз болса, онда екінші жақ трубканы іліп қоймас үшін бос сегмент жөнелтіледі. Екінші жағынан, егер байланысты автоматты үзу ережесін қолдансақ, онда байланыс алдымен бір жақтан автоматты түрде үзіледі, онан кейін екінші жақтан және көптеген сегменттер қатарынан жоғалады.

Біз осымен бұл сұрақты талқылауды аяқтаймыз. Енді байланысты деректерді жоғалтпай үзу, алғашқыдай көрінгендей оңай еместігі түсінікті болған шығар деп ойлаймыз. Жоғарыда айтылғанның барлығынан мынадай түйін жасауға болады: транспорттық қызметті тұтынушы байланысты үзу мәселесін шешуге қатысуы тиіс, транспорттық ішкі жүйе бұл мәселені жеке өзі шеше алмайды. Назар аударыңыздар, әдетте TCP байланысты симметриялы үзуді пайдаланса да (бұл жағдайда әр жақ деректер тасымалдауы аяқталғаннан кейін FIN дестесін жөнелтіп, өз байланысын тәуелсіз үзеді), веб-серверлер өз клиенттеріне байланыстың лезде үзілетінін хабарлайтын арнайы RST дестесін жөнелтеді – бұл байланысты ассимметриялы үзуге ұқсайды. Мүмкін бұл веб-серверлер деректер алмасу процедурасымен таныс болғандықтан шығар. Алдымен ол клиенттен сұраныс алады (клиенттің жалғыз жөнелтетіні), ал сонан кейін клиентке жауап қайтарады. Сөйтіп, жауап қайтарылғаннан кейін деректер алмасу аяқталады: әр жақ екіншісіне қажет нәрсені жөнелтті. Сол себепті сервер клиентке ескерту жіберіп байланысты лезде үзеді. Ескертуді алғаннан кейін клиент те бірден байланысты үзеді. Егер ескерту клиентке қандай да бір себептермен жетпесе, онда ол белгілі бір уақыт өткеннен кейін сервердің онымен байланыспайтынын сезіп, байланысты үзеді. Кез келген жағдайда деректер сәтті тасымалданады.

6.2.4. Қателіктерді бақылау және деректер ағынын басқару

Байланысты орнату және үзу үдерістерін оқып білгеннен кейін енді байланысты қолдану барысында оны басқару қалай жүретінін қарастырамыз. Басты мәселелердің бірі – қателікті бақылау және деректер ағынын басқару. Қателіктерді бақылау деректердің қажетті сенімділік дәрежесінде тасымалдануына жауап береді. Әдетте, бұл деректер қатесіз жеткізілуі тиіс дегенді білдіреді. Деректер ағынын басқару – жөнелтуші және қабылдаушы жылдамдықтарын келістіру.

Бұл екі сұрақты да біз арналық деңгей жайлы айтқан кезде қарастырғанбыз. Транспорттық деңгейде де сол *3-тарауда* айтылған механизмдер қолданылады. Оларды қысқаша айта кетейік:

1. Фреймде қателікті анықтау коды жазылған (мысалы, CRC-код немесе бақылау қосындысы). Осы код көмегімен ақпараттың дұрыс жеткізілгендігі тексеріледі.
2. Фреймде сәйкестендіруші реттік нөмір жазылған, оны жөнелтуші ақпараттық сәтті жеткізілгендігі жайлы растау алғанша жөнелтіп отырады. Бұл **ARQ (Automatic Repeat reQuest – қайта тасымалдауға автоматты сұраныс)** деп аталады.

3. Кез келген уақыт сәтінде жөнелтуші тасымалдайтын кадрлар саны шектеулі: егер растау дер кезінде келмесе тасымалдау тоқтатылады. Егер бұл ең үлкен сан бір дестеге тең болса, онда хаттама **тоқталыс және растауды күтуі бар хаттама (stop-and-wait)** деп аталады. Үлкен көлемдегі терезе конвейерлік өңдеуді пайдалануға, сонымен бірге ұзын және шапшаң тораптармен жұмыс істегенде өнімділікті жақсартуға мүмкіндік береді.
4. **Сырғымалы терезе (sliding window)** хаттамасы өзіне осы мүмкіндіктердің барлығын жинақтаған, сонымен бірге екі бағытта да деректер тасымалын қолдайды.

Егер бұл механизмдер арналық деңгейде кадрларға қолданылатын болса, онда орынды сұрақ туындайды: бұны транспорттық деңгейдің сегменттеріне қалай қолдануға болады? Іс жүзінде арналық және транспорттық деңгей көп жерде бірін бірі қайталайтын болып шықты. Оларда бірдей механизмдер қолданылғанымен, олардың қызметі және сапасында айырмашылық бар.

Қызметтеріндегі айырмашылықтарды көрсету үшін қателікті анықтауды қарастырайық. Арналық деңгейдің бақылау қосындысы кадр бір арна бойымен тасымалданғанша, транспорттық деңгей бақылау қосындысы сегмент бүкіл жол бойында тасымалданғанша қорғайды. Бұл әр арнаны тексеруден ерекше – толассыз тексеру. Дестелердің маршруттауыш ішінде де бұзылғандығының мысалдары бар (Saltzer және басқалар, 1984), арналық деңгей бақылау қосындысы дестелерді маршруттауыш ішінде болғанда емес, арна бойымен қозғалғанда ғана қорғайды. Нәтижесінде, әр каналда қателікті анықтауға тексеріс жүргізілсе де, қателікпен жеткізу орын алады.

Осы және басқа да мысалдар ғалымдарға (Saltzer және басқалар) **«толассыздық» принципін (end-to-end argument)** құрастыруға мүмкіндік берді. Осы принципке сәйкес транспорттық деңгейде орындалатын толассыз тексеру деректерді дұрыс тасымалдауға қажет болып табылады. Арналық деңгейде орындалатын тексеру міндетті емес, бірақ өнімділікті жақсартуға мүмкіндік береді (себебі бұзылған десті бәрі бір бүкіл жолды жүріп өтеді, ал бұл желінің артық жүктелуіне әкеледі).

Сападағы айырмашылықты көрсету үшін деректеді қайталап тасымалдау және сырғымалы терезе хаттамасын қарастырайық. Сымсыз арналардың көбі спутниктерге қарағанда бір уақыт бірлігінде тек бір кадр жөнелтуге мүмкіндік береді. Бұл өткізгіштік қабілеттіліктің кідіріс уақытына көбейтіндісі осы арна үшін өте аз және арна ішінде бүтін кадр әзер орналасатынын білдіреді. Сондықтан бұнда өнімділікті арттыру үшін кіші көлемдегі терезені қолдану керек. Мысалы, 802.11 стандартында тоқталысы және растауды күтуі бар хаттама қолданылады. Хаттама бір кадрды тасымалдауды және қайта тасымалдауды оның алынғандығы жайлы растау келгенше орындайды және тек растау алынғаннан кейін ғана басқа кадрға көшеді. Егер терезе көлемі бір кадрдан үлкен болса, бұл

өнімділікті жақсарту орнына тек тасымалдау үдерісін қиындатар еді. Ethernet және интернет-провайдерлер магистралі (коммутациялық) тәрізді сымды және оптыталшықты байланыс тораптарында қателіктің пайда болу жиілігі өте аз, бұл арналық деңгейдегі қайта тасымалдаудан бас тартуға мүмкіндік береді, себебі аз ғана жоғалған кадрлар санын толассыз қайта тасымалдау арқылы қалпына келтіруге болады.

Екінші жағынан, көптеген TCP-байланыстар үшін өткізгіштік қабілеттіліктің кідіріс уақытына көбейтіндісі бір сегменттен әлдеқайда көп. Деректерді бүкіл АҚШ территориясында 1Мбит/с жылдамдықпен тасымалдайтын байланысты қарастырайық, ал дестенің қабылдаушыға дейін жетіп кері қайтатын уақыты 100 мс құрайды. Тіпті осындай баяу байланыстың өзінде 200 Кбит ақпарат қабылдаушыда сегментті жөнелтіп, оған растау алатын уақытқа қажет уақыт аралығында сақталады. Бұндай жағдайда үлкен көлемдегі сырғымалы терезе қолданған дұрыс. Тоқталысы және растауды күтуі бар хаттама өнімділікке үлкен нұқсан келтіреді. Біздің мысалымызда ол желінің нақты жылдамдығына қарамастан бір сегментті тасымалдауды 200 мс-пен (немесе секундына 5 сегмент) шектер еді.

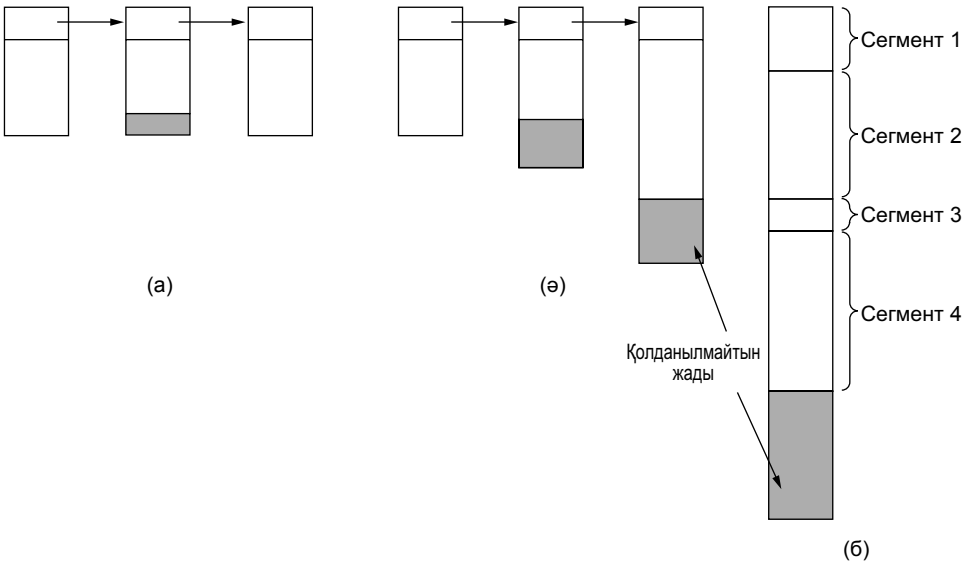
Транспорттық хаттамалар әдетте, үлкен көлемдегі сырғымалы терезені қолданатын болғандықтан, біз буферлеу үдерісін егжей-тегжейлі қарастырамыз. Хосттың әрқайсы жеке өңдейтін бірнеше байланысы болуы мүмкін болғандықтан, сырғымалы терезе үшін оған буферлік кеңістіктің үлкен саны қажет болар еді. Буферді жөнелтуші де, қабылдаушы да пайдалануы тиіс. Жөнелтушіге буфер келіп жеткендігі жайлы растауы келмеген барлық жөнелтілген сегменттерді сақтау үшін қажет. Егер олар жоғалған болса, онда оларды қайталап жөнелту керек болады.

Жөнелтушінің буферлеуді орындайтынын біле отырып, қабылдаушы өз буферін өз қалауынша пайдалануы мүмкін. Мысалы, қабылдаушы барлық байланыстар пайдаланатын бір буферлік кеңістікті ұстап отыруы мүмкін. Сегмент келген кезде оған динамикалық түрде буфер бөлуге талпыныс жасалады. Егер бұл талпыныс жүзеге асса, онда сегмент қабылданады, кері жағдайда ол қабылданбайды.

Жөнелтуші жоғалған сегменттерді қайта жөнелтуге дайын болғандықтан, қабылдаушының сегменттерді қабылдамай тастауы айтарлықтай нұқсан келтірмейді, дегенмен, біраз ресурстар шығындалады. Жөнелтуші растау алғанша тасымалдауды қайталайды.

Жөнелтушіде және қабылдаушыда буферлеуді таңдаудың ымыралы шешімі байланыс трафигінің типіне байланысты. Егер трафик импульсті, кіші қуатты болса, мысалы, интерактивті терминал трафигі тәрізді, онда ешқандай буфер бөлмей, жөнелтуші буферлеуіне сеніп (сегмент кездейсоқ жойылып кетсе), оларды екі жақтан да динамикалық түрде қабылдаған жөн. Екіншіден, файлды тасымалдағанда деректерді ең үлкен жылдамдықпен жөнелту үшін қабылдаушының бүтіндей буфер терезесін бөлгені дұрыс болар еді. Бұндай стратегияны TCP қолданады.

Дегенмен, буферлер жиынтығы жайлы сұрақ ашық қалып тұр. Егер көптеген сегменттердің көлемі бірдей болса, онда буферді *6.12 а-суретінде* көрсетілгендей, әрқайсына бір сегмент сиятын мөлшері біркелкі буферлер массиві ретінде ұйымдастырған жөн. Алайда, егер сегменттердің көлемдері әртүрлі болса, веб-парақты жүктеуге сұраныстан, біррангалық файлдар тасымалдау кезіндегі ірі дестелерге дейін, онда бекітілген көлемдегі буферлер массиві қолайсыз болады. Егер буфер көлемін мүмкін деген ең үлкен сегмент көлеміне сәйкес алсақ, онда кішігірім сегменттерді сақтау кезінде жады тиімсіз шығындалады. Ал, егер буфер көлемін кіші етіп алсақ, онда үлкен сегментті сақтау үшін ілеспе қиындықтарымен бірнеше буфер қажет болады.



6.12-сурет. Буферлер жиынтығының ұйымдастырылуы: а – бекітілген көлемдегі буферлер тізбегі; ә – айнымалы көлемдегі буферлер тізбегі; б – бір байланыс үшін бір үлкен айналымды буфер

Көрсетілген мәселені шешудің басқа бір әдісі *6.12 ә-суретінде* көрсетілгендей айнымалы көлемдегі буферді пайдалану. Бұл әдістің артықшылығы жадыны тиімді пайдалану, бірақ оның есесіне буферді басқару қиындайды. Үшінші әдіс, байланысқа *6.12 б-суретінде* көрсетілгендей біртұтас үлкен айналымды буфер бөлу. Бұл қарапайым схема сегмент көлеміне тәуелсіз жұмыс жасайды, бірақ ол тек барлық байланыстар қатты жүктелген жағдайда жадыны жақсы пайдаланады.

Байланысты ашу және жабу кезінде, сонымен бірге трафик формасы өзгергенде жөнелтуші және қабылдаушы буферді бөлуді динамикалық түрде өзгертіп отыруы тиіс. Демек, транспорттық хаттама жөнелтушіге екінші бетте буфер бөлуге сұраныс жіберуге мүмкіндік беруі керек. Буферлер әр байланысқа жеке

немесе екі хост арасындағы байланыстарға бөлінуі мүмкін. Буферге сұранысқа жауап ретінде қабылдаушы өз буферлерінің қалып-күйін біле отырып, бірақ қандай трафик келетінін білместен, жөнелтушіге ол үшін X буфер қорда сақтап қойғанын хабарлай алады. Ашық байланыстар саны ұлғайған кезде бөлінген буферлер санын немесе көлемін азайту қажет болады. Хаттама осындай мүмкіндіктерді ұсынуы тиіс.

3-тарауда сипатталған сырғымалы терезе хаттамасына қарағанда буферлерді динамикалық бөлу үшін буферлеуді растаудан ажырату керек. Буферлерді динамикалық бөлу іс жүзінде айнымалы көлемдегі терезені пайдалану дегенді білдіреді. Жөнелтуші алдымен өз қажеттілігіне сүйене отырып буфердің белгілі бір санына сұраныс береді. Қабылдаушы өз мүмкіндігінше буфер бөледі. Әр сегментті жөнелткен кезде жөнелтуші буферлер санын бірге кемітіп отыруы тиіс, ал сан нөлге жеткен кезде тоқтауы тиіс. Қабылдаушы бағыттас сегментпен жеке растау және өзінде бар бос буферлер жайлы ақпарат жөнелтеді. Бұл схема TCP-де қолданылады, мұнда буферлер жайлы ақпарат *Window size* өрісінің тақырыбында жазылған.

Төрт биттік реттік нөмірі бар дейтаграммалық желіде динамикалық терезені басқару мысалы *6.13-суретте* көрсетілген. Бұл мысалда деректер сегмент түрінде А хостынан В хостына беріледі, ал растаулар және буфер бөлуге сұраныстар кері бағытта жүреді (олар да сегмент түрінде). Бастапқыда А хосты 8 буфер сұрайды, бірақ оған тек 4 буфер бөлінеді. Сонан кейін ол үш сегмент жөнелтеді, оның соңғысы жоғалады. Алтыншы қадамда А хосты жөнелткен 0 және 1 сегменттерінің жеткеніне растау алады. Нәтижесінде А хосты буферлерді босатып, тағы үш сегмент жөнелтеді (реттік нөмірлері 2, 3 және 4). А хосты 2-нөмірлі сегментті жібергенін біледі, сондықтан 3 және 4 жіберуге болады деп, осы сегменттерді жөнелтеді. Осы қадамда ол оқшауланады, себебі оның буферлер санауышы нөлге жетті. Ол енді өзіне жаңа буфердің бөлінгенін күтеді. Тоғызыншы қадамда А хостының тайм-ауты басталады, себебі ол 2-сегментке растау алған жоқ. Бұл сегмент қайталап жөнелтіледі. 10-жолда В хосты 4-сегментті қоса барлық сегменттердің алынғандығын растайды, бірақ А хостына буфер бөлуден бас тартады. *3-тарауда* сипатталған бекітілген көлемдегі терезе хаттамасында бұндай жағдай мүмкін емес. В хосты жөнелткен келесі сегмент А хостына тағы бір сегмент жөнелтуге рұқсат береді. Бұл В хостында бос буферлік кеңістік пайда болғанда орын алады – себеп, транспорттық қызмет тұтынушысы көп деректер қабылдаған болуы мүмкін.

Буферді бөлудің бұндай схемасында мәселе дейтаграммалық желілерде басқарушы сегмент жоғалған кезде туындайды – бұл іс жүзінде нақты орын алуы мүмкін. 16-жолға назар аударыңыз. В хосты А хостына қосымша буфер бөлді, алайда, бұл жайлы мәлімдеме жоғалып кетті. Күтпеген жағдай! Басқарушы сегменттердің алынғандығы расталмайтын болғандықтан олар тайм-аут бойынша қайталап жөнелтілмейді, сондықтан А хосты шындап және ұзақ уақытқа оқшауланады. Мұндай тұйыққа тіреліс болмас үшін әр хост белгілі бір кезең сайын растауы және әр байланыстың буферлерінің қалып-

күйі жайлы басқарушы сегмент жөнелтіп отыруы тиіс. Бұл түбінде тұйықтан шығуға мүмкіндік береді.

	A	Мәлімдеме	B	Түсініктеме
1	→	< request 8 buffers>	→	А-ға 8 буфер керек
2	←	<ack = 15, buf = 4>	←	В тек 0-3 мәлімдемелерді жөнелтуге мүмкіндік береді
3	→	<seq = 0, data = m0>	→	А-да тек 3 буфер қалды
4	→	<seq = 1, data = m1>	→	А-да енді 2 буфер қалды
5	→	<seq = 2, data = m2>	• • •	Мәлімдеме жоғалды, бірақ А 1 буфер бар деп ойлайды
6	←	<ack = 1, buf = 3>	←	В 0 және 1 сегменттерін алғанын растайды, 2-4 жөнелтуге рұқсат береді
7	→	<seq = 3, data = m3>	→	А-да 1 буфер қалды
8	→	<seq = 4, data = m4>	→	А-да буфер қалмады, ол тоқтауы керек
9	→	<seq = 2, data = m2>	→	А-ның күту уақыты аяқталды, ол тағы да жөнелтеді
10	←	<ack = 4, buf = 0>	←	Сегменттердің барлығы расталды, ол тоқтау керек
11	←	<ack = 4, buf = 1>	←	А енді 5-ші сегментті жөнелте алады
12	←	<ack = 4, buf = 2>	←	В жаңа буфер тапты
13	→	<seq = 5, data = m5>	→	А-ның 1 буфері қалды
14	→	<seq = 6, data = m6>	→	А тағы да оқшауланды
15	←	<ack = 6, buf = 0>	←	А әлі оқшауланған
16	• • •	<ack = 6, buf = 4>	←	Тұйыққа тірелу

6.13-сурет. Буферлерді динамикалық бөлу. Бағыттауыш сызықпен тасымалдау бағыты көрсетілген. Көп нүкте (...) сегменттің жоғалғандығын білдіреді

Біз осы уақытқа дейін үнсіз келісім бойынша деректер тасымалдау жылдамдығына қойылатын жалғыз шектеу қабылдаушыдағы бос буферлер кеңістігінің саны деп есептеп келдік. Алайда, бұл олай емес. Жады микросхемалар мен винчестерлер бағаларының төмендеуіне байланысты (кезінде тым үлкен болған) қазір хосттарды үлкен көлемдегі жадымен қамтамасыз етуге болады, сондықтан буферлердің жетіспеушілігі, тіпті, байланыс ірі территорияны қамтығанның өзінде өте сирек орын алады. Әрине, таңдап алынған буфер көлемінің жеткілікті түрде үлкен болғаны қажет, TCP (Zhang және басқалар, 2002) жағдайында бұл талап әрқашан орындалатын.

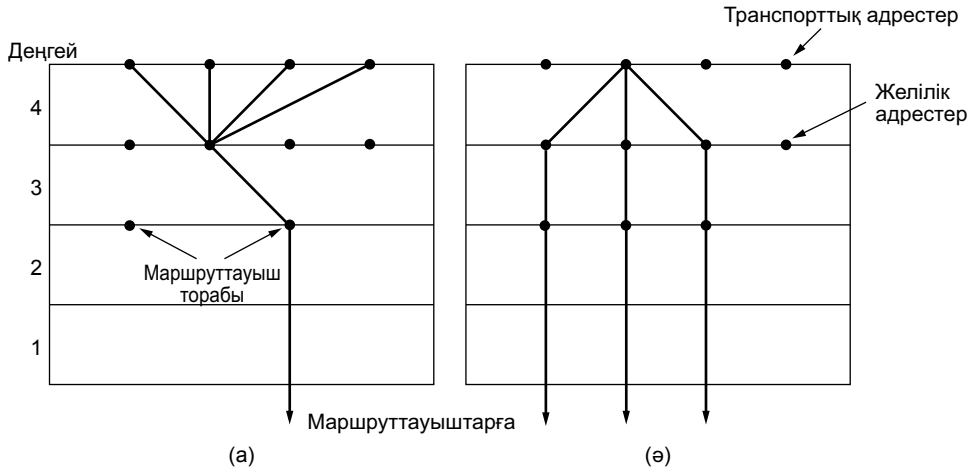
Егер ең үлкен ағын буферлер санына шектеу қоймаса, онда басқа жіңішке орын: желінің өткізгіштік қабілеті туындайды. Егер көршілес маршруттауыштар арасындағы кадрлар алмасудың ең үлкен жылдамдығы секундына x кадр болса және екі хост арасында k қиылыспайтын жолдар бар болса, онда екі хостағы буферлер санына қарамастан олар бір-біріне секундына kx сегменттен артық жөнелте алмайды. Егер жөнелтуші жоғары жылдамдықпен жөнелтетін болса, онда желі асыра жүктелген болады.

Жөнелтушіден келетін деректер тасымалын қабылдаушы буферінің сыйымдылығына емес, желінің өткізгіштік қабілетіне негіздей отырып шектейтін механизм қажет. 1975 жылы Белснес (Belsnes) деректер ағынын басқару үшін сырғымалы терезе схемасын пайдалануды ұсынды. Схемада жөнелтуші терезе мөлшерін желі өткізгіштік қабілеттілігіне динамикалық түрде сәйкестендіріп отырады. Сөйтіп, сырғымалы терезе бір мезетте ағынды басқаруды және асыра

жүктелуді бақылауды жүзеге асыруға мүмкіндік береді. Егер желі секундына с сегмент өңдей алатын болса, ал уақыт айналымы (тасымалдауды, таратуды, кезекте күту, қабылдаушының өңдеу уақытын және растаудың қайтып оралуын қоса алғанда) *r* болса, онда жөнелтушінің терезе көлемі *cr*-ге тең болуы керек. Осындай терезе көлемінде жөнелтуші арнаны бар мүмкіндігінше пайдаланып жұмыс істейді. Желі өткізгіштік қабілеті уақыт өте өзгеретін болғандықтан, терезе мөлшері өзгерісті ескеріп жиі бапталуы тиіс. TCP-де ұқсас схема қолданылады, біз оны төменде қарастырамыз.

6.2.5. Мультиплекстеу

Виртуалды арнада және бір физикалық торапта бір байланыста бірнеше әңгімені біріктіру желілік құрылымның бірнеше деңгейінде маңызды рөл атқарады. Мұндай тығыздау қажеттілігі кейбір жағдайда транспорттық деңгейде де туындайды. Мысалы, егер хосттың тек бір ғана желілік адресі бар болса, онда ол транспорттық деңгейдің барлық байланыстарымен қолданылады. Бұл жағдайда кіріс сегментін қандай үдеріске беру керек екендігін ажырата алатын әдіс керек. Бұндай жағдай **мультиплекстеу** деп аталады, ол *6.14 а-суретінде* көрсетілген. Суретте транспорттық деңгейдің әртүрлі төрт байланысы қашықтықтағы хостпен бір желілік байланысты қолданады (мысалы, бір IP-адрес).



6.14-сурет. Мультиплекстеу: а – тікелей; ә – кері мультиплекстеу

Тығыздау транспорттық деңгейде басқа себептермен де маңызды рөл атқаруы мүмкін. Мәселен, хост бірнеше желілік жолдарды пайдалана алады делік. Егер тұтынушыға бір желілік жол ұсынғаннан да үлкен өткізгіштік қабілеттілік немесе сенімділік қажет болса, онда бұл мәселені *6.14 ә-суретінде* көрсетілгендей трафикті бірнеше жол арасында бөліп тасымалдайтын байланысты қолданып, оларды кезекпен пайдалану арқылы шешуге болады. Бұл

тәсіл **кері мультиплекстеу** деп аталады. Егер *k* ашық желілік байланыс болса, онда тиімді өткізгіштік қабілеттілік *k* есе өсуі мүмкін. Кері мультиплекстеу мысалы ретінде көптеген желілік интерфейстермен байланыс орнатуға мүмкіндік беретін **SCTP (Stream Control Transmission Protocol – ағынды басқарып тасымалдау хаттамасы)** қарастыруға болады. TCP керісінше, жеке сокетті пайдаланады. Кері мультеплекстеу арналық деңгейде де қолданылады. Бұл жағдайда байланыстың бірнеше баяу арналары әлдеқайда жылдам жұмыс істейтін бір арнаға біріктіріледі.

6.2.6. Шалыстан кейін қайта қалыпқа келтіру

Егер хосттар және маршруттауыштар шалысқа бейім болса немесе байланыстар ұзаққа созылатын болса (мысалы, мультимедианы немесе программалық бағдарламаны жүктеген кезде), онда шалыстан кейін қайта қалыпқа келтіру өте өзекті мәселе болып келеді. Егер транспорттық ішкі жүйе толығымен хостта орналасса, желі және маршруттауыштар істен шыққаннан кейін қайта қалпына келтіру қыйындық туындатпайды. Транспорттық ішкі жүйелер сегменттің жоғалуын үнемі күтіп отырады және онымен қалай күресу керек екенін біледі: ол үшін қайта жөнелту орындалады.

Хост істен шыққаннан кейін қайта қалпына келтіру әлдеқайда күрделі мәселе болып келеді. Соның ішінде клиенттер үшін істен шықаннан кейін серверді қайта жүктеп, жұмысты қайта жалғастыру мүмкін болу керек. Бұл жерде қандай қыиыншылық бар екенін түсіндіру үшін, айталық бір хост – клиент – ұзын файлды екінші басқа хостқа – файлдық серверге – қарапайым тоқталысы және растауды күтуі бар хаттама арқылы жөнелтті делік. Сервердің транспорттық деңгейі келген сегменттерді бірінен кейін бірін транспорттық деңгей тұтынушысына жөнелтеді. Файлдардың жартысын алғаннан кейін сервер істен шығып қайта жүктелді делік, бұдан кейін оның барлық кестелері қайта инициализацияланады, сондықтан ол осыған дейін не болғанын білмейді.

Алдыңғы қалып-күйді қалпына келтірмек болып, сервер барлық хосттарға кең таратылымды сегмент жөнелтіп, өзінің қайта жүктелгенін ескертіп, оған ашық байланыстардың қалып-күйін хабарлауды сұрауы мүмкін. Әр клиент екі қалып-күйдің бірінде болуы мүмкін: бір расталмаған сегмент (*S1* қалып-күйі) немесе бірде бір расталмаған сегмент жоқ (*S0* қалып-күйі). Бұл ақпарат клиентке соңғы сегментті қайта жөнелту, жөнелтпеу жайлы шешім қабылдауға жеткілікті.

Бір қарағанда бәрі де түсінікті болып көрінеді: сервердің қайта жүктелгендігін біліп, клиент соңғы расталмаған сегментті қайта жөнелтуі тиіс. Демек, қайта жөнелту тек клиент *S1* қалып-күйінде болғанда мүмкін. Алайда, егер егжей-тегжейлі қарағанда барлығы біз болжағандай қарапайым емес. Мысалы, сервердің транспорттық ішкі жүйесі қолданбалы үдеріске алдымен растауды, сонан кейін дестені жөнелткен жағдайды қарастырайық. Сегментті шығыс ағынға жазу және растауды жөнелту – бір мезгілде орындалатын

әртүрлі ажырамас оқиға. Егер шалыс растау жөнелтілгеннен кейін, бірақ жазу орындалғанша орын алса, онда клиент растауды алып, сервердің қайта жүктелгендігі жайлы хабарлама келген кезде *S0* қалып-күйінде болады. Сөйтіп, клиент сегментті қайта жөнелтпейді, ол сегмент қабылданды деп есептейді, нәтижесінде сегмент жоқ болып шығады.

Осы тұста сіз «Егер сервердің транспорттық ішкі жүйесі орындайтын іс-әрекет ретін өзгертсе не болар еді. Алдымен жазуды орындап, сонан кейін растау жөнелтсе?» – деп ойлайсыз. Айталық жазу орындалды, ал шалыс растау жөнелтілгенше орын алды делік. Бұл жағдайда клиент *S1* қалып-күйінде болады, сондықтан сегментті қайта жөнелтеді, сөйтіп біз шығыс ағында сегмент көшірмесін аламыз.

Сонымен, клиент және сервердің қалай программаланғанына қарамастан хаттама дұрыстап қайта қалпына келмейтін жағдай болады. Серверді екі жолмен программалауға болады: не ол алдымен растау жөнелтеді немесе алдымен сегментті жазады. Клиент: соңғы сегментті үнемі қайта жөнелту, ешуақытта қайта жөнелтпеу, тек *S0* қалып-күйінде қайта жөнелту және тек *S1* қалып-күйінде қайта жөнелту сияқты төрт жолдың бірімен программалануы мүмкін. Сөйтіп, сегіз комбинация аламыз, алайда, әр комбинация үшін хаттаманы жаңылысқа әкелетін оқиғалар жиынтығы бар.

Қабылдаушы хост пайдаланатын стратегия

Жөнелтуші хост пайдаланатын стратегия	Алдымен ACK, сонан кейін жазу			Алдымен жазу, сонан кейін ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Тасымалдауды үнемі қайталау	OK	DUP	OK	OK	DUP	DUP
Тасымалдауды ешуақытта қайталамау	LOST	OK	LOST	LOST	OK	OK
Тасымалдауды S0 қайталау	OK	DUP	LOST	LOST	DUP	OK
Тасымалдауды S1 қайталау	LOST	OK	OK	OK	OK	DUP

OK = Хаттама дұрыс жұмыс істейді
DUP = Хаттама мәлімдеме көшірмесін құрастырады
LOST = Хаттама мәлімдемені жоғалтады

6.15-сурет. Сервер және клиент стратегиясының түрлі комбинациялары

Серверде үш оқиға: растауды жөнелту *(A)*, сегментті шығыс үдеріске жазу (*W*) және шалыс *(C)* орын алуы мүмкін. Олар: *AC(W), AWC, C(AW), C(WA), WAC* және *WC(A)* сияқты алты реттілікпен орын алуы мүмкін, мұнда жақшалар *C* оқиғасынан кейін *A* немесе *W* оқиғалары орын алмауы да мүмкін (демек, мүлдем істен шықты). Сервер және клиент стратегиясының барлық сегіз комбинациясы әрқайсысы өз оқиғалар реттілігімен *6.15-суретінде* көрсетілген. Әр комбинация үшін хаттама қателігіне әкелетін оқиғалар реті бар екеніне назар

аударыңыздар. Мысалы, егер расталмаған сегментті үнемі қайталап жөнелтіп отырса, онда *AWC* оқиғасы анықталмаған дубликаттың пайда болуына әкеледі. Алайда, басқа оқиғалардың басқа тізбегінде хаттама дұрыс жұмыс істейді.

Хаттаманы күрделендіру көмектеспейді. Тіпті, сервер алынған дестені жазбас бұрын клиентпен бірнеше сегменттермен алмасып үлгерсе де, клиент серверде не болып жатқанын нақты білмейді, сондықтан ол серверде шалыстың қай кезде: жазғанға дейін немесе жазғаннан кейін орын алғанын білмейді. Міне, осыдан сөзсіз орын алатын нәтиже: бір мезгілде орын алатын оқиғаларға қатаң тыйым салынған кезде жеке оқиғалар бір мезгілде емес, бірінен соң бірі ретпен орын алады – хосттың істен шығуын және қайта қалыпқа келуін жоғары деңгейлер үшін айқын етуге болады.

Жалпы түрде бұны келесідей тұжырымдауға болады: *N* деңгейінің шалыстан кейін қайта қалпына келуі тек *N+1* деңгейімен және тек жоғары деңгейде алдыңғы қалып-күйді қалпына келтіруге қажетті ақпарат сақталған жағдай да ғана іске асырылуы мүмкін. Бұл жоғарыда келтірілген, егер әр деңгей өзінің ағымдағы қалып-күйін қадағалап отырса, транспорттық деңгей желілік деңгейдегі шалысты қалпына келтіре алады деген тұжырыммен сәйкес келеді.

Бұл мәселе бізді толассыз растау сұрағының мәніне әкеледі. Іс жүзінде транспорттық хаттама төменгі деңгейлер тәрізді тізбекті емес, толассыз болып келеді. Енді тұтынушының қашықтықтағы деректер базасына жүгінген жағдайын қарастырамыз. Қашықтықтағы транспорттық ішкі жүйе алдымен жоғары деңгейге сегменттерді жөнелтіп, сонан кейін растау жөнелтетін етіп программаланған делік. Тіпті, бұл жағдайдың өзінде тұтынушы машинасының растау алғаны қашықтықтағы хост деректер базасын жаңартып үлгерді дегенді білдірмейді. Келуі жұмыстың орындалғанын, ал келмеуі сәйкесінше жұмыстың орындалмағанын білдіретін нағыз толассыз растау мүмкін емес. Бұл сұрақ «Saltzer және басқалар» (1984) басылымында егжей-тегжейлі талқыланады.

6.3. АСЫРА ЖҮКТЕЛУДІ БАСҚАРУ

Егер бірнеше машинаның транспорттық ішкі жүйесі желіге тым көп дестелерді жоғары жылдамдықпен жөнелтсе, желі асыра жүктеліп, өнімділік күрт төмендейді, бұл дестелердің кідірісіне және жоғалуына әкеледі. Осындай жағдайлармен күресуге бағытталған асыра жүктелуді басқару, желілік және транспорттық деңгейлердің бірлесіп жұмыс істеуін қажет етеді. Асыра жүктелу маршруттауышта орын алатын болғандықтан, оны анықтаумен желілік деңгей айналысады. Алайда, асыра жүктелудің себебі желілік деңгейге транспорттық деңгей жөнелткен трафик болып келеді. Сондықтан асыра жүктелуді бақылаудың жалғыз тиімді әдісі – транспорттық хаттамалардың дестелерді баяу тасымалдауы.

Біз *5-тарауда* желілік деңгейдегі асыра жүктелуді бақылау механизмдері жайлы айтқанбыз. Бұл бөлімде біз осы үдерістің басқа бөлігі, транспорттық деңгейдегі асыра жүктелу жайлы айтамыз. Асыра жүктеуді бақылаудың негізгі

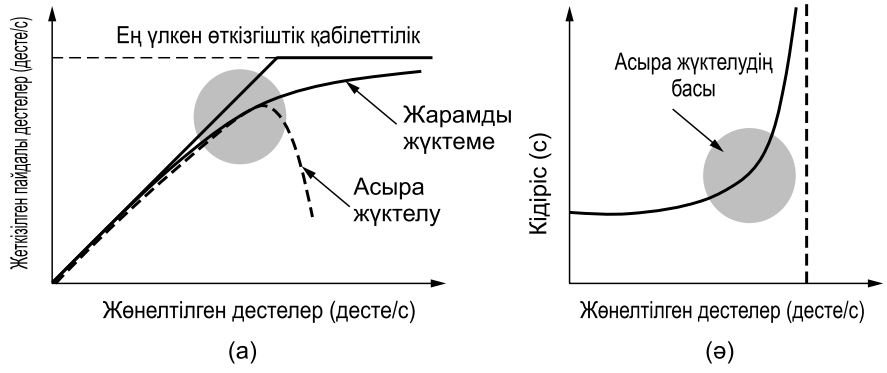
сұрақтарын талқылағаннан кейін біз хосттардың желіге дестелерді жөнелту жылдамдығын реттеуге көмектесетін тәсілдерді сипаттауға кірісеміз. Интернет желісіндегі асыра жүктеуді бақылау көбіне, транспорттық деңгейге негізделген, ол үшін TCP және басқа да хаттамаларға арнайы алгоритмдер кіріктірілген.

6.3.1. Талап етілген өткізгіштік қабілеттілікті бөлу

Трафикті реттеу тәсілдерін сипаттауға көшпес бұрын, біздің асыра жүктелуді бақылау алгоритмінен нені күтетінімізді түсінуіміз керек. Демек, осындай алгоритм желінің қандай қалып-күйін қолдауы керектігін анықтауымыз қажет. Оның мақсаты тек асыра жүктелуді болдырмау емес: ол өткізгіштік қабілеттілікті желіде жұмыс істейтін транспорттық жүйелер арасында дұрыс таратуы керек. Өткізгіштік қабілеттіліктің дұрыс таратылуы желінің жақсы өнімділігін (себебі желі асыра жүктелусіз мүмкін деген барлық қуаттылықты пайдаланып жұмыс істеуі тиіс), бәсекелес траспорттық ішкі жүйелер қағиадасын және ресурстарды бөлуге сұраныстар өзгерісін жылдам қадағалап отыруды қамтамасыз етеді. Біз осы белгілердің әрқайсын жеке қарастырамыз.

Тиімділік және қуат

Транспорттық ішкі жүйелер арасында өткізгіштік қабілеттілікті тиімді тарату үшін желінің барлық мүмкіндігін пайдалану керек. Алайда, бұл жылдамдығы 100 Мбит/с арна жағдайында әр транспорттық ішкі жүйе 20 Мбит/с алуы тиіс дегенді білдіреді. Жақсы өнімділік үшін оларға төмен қуаттылық бөлінуі керек. Себебі трафик көбіне, біркелкі емес. Біз *5.3-бөлімде* **тиімді өткізгіштік қабілеттілікті (goodput – пайдалы дестелер қабылдаушыға келіп жететін жылдамдық)** желіге түсетін жүктеме функциясы ретінде анықтаған болатынбыз. Осы қисық және оған сәйкес кідіріс қисығы (жүктеме функциясы ретінде) *6.16-суретте* көрсетілген.



6.16-сурет. Тиімді өткізгіштік қабілеттілік (а); кідіріс жүктеме функциясы ретінде (ә)

Алғашқыда желідегі жүктеме өскен сәтте тиімді өткізгіштік қабілеттілік тұрақты жылдамдықпен өседі (*6.16 а-сурет*), бірақ жүктеме мәні өткізгіштік қабілеттілік мәніне жақындаған кезде тиімді өткізгіштік қабілеттілік мәнінің өсуі баяулайды. Бұл төмендеу желі буферлерінің толып кетуін және трафик күрт өзгергенде деректердің жоғалуын болдырмау үшін қажет. Егер транспорттық хаттама кідірген бірақ жоғалмаған дестелердің қайта жөнелтуді орындаса, желінің асыра жүктелу салдарынан істен шығуы мүмкін. Бұл жағдайда жөнелтушілер дестелерді көптеп жөнелтуді жалғастыра береді, оның пайдасы азая түседі.

Дестелердің кідіріс графигі *6.16 ә-суретінде* көрсетілген. Кідіріс бастапқыда тұрақты және желідегі таралу кідірісіне сәйкес болып келеді. Жүктеме мәні өткізгіштік қабілеттілік мәніне жақындаған сайын кідіріс өседі, ол алдымен баяу, ал сонан кейін өте жылдам өседі. Бұл да жоғары жүктемеде күрт өсетін трафикке байланысты орын алады. Іс жүзінде кідіріс шексіздікке кетпейді, тек модель шексіз буферлерді қолдануды болжамаса. Оның орнына буферлер толған кезде дестелер жоғалатын болады.

Жақсы тиімді өткізгіштік қабілеттілікке және кідіріске қол жеткізу үшін өнімділік асыра жүктелу басталған сәтте төмендей бастауы керек. Желіде жақсы өнімділікке қол жеткізу үшін өткізгіштік қабілеттілікті кідіріс күрт жоғарылағанша бөлуге болады. Бұл нүкте желі өткізгіштік қабілеттілігінен төмен орналасқан. Оны анықтау үшін 1979 жылы Кляйнрок (Kleinrock) **қуат (power)** өлшемін енгізуді ұсынды. Ол:

$$Қуат = жүктеме/кідіріс$$

формула бойынша есептеледі.

Кідіріс аз және тұрақты болып тұрғанда, қуат желі жүктемесімен бірдей өседі. Кідіріс күрт өскен кезде ол ең үлкен мәнге жетіп, күрт төмендейді. Қуат ең үлкен мәнге жететін жүктеме транспорттық ішкі жүйе желіге бере алатын ең тиімді жүктеме болып саналады.

Ең үлкен өлшем бойынша тең қолжетімділік

Біз өткізгіштік қабілеттілікті бірнеше жөнелтушілер арасында қалай бөлу керек екендігі жайлы әлі айтқан жоқпыз. Жауап өте қарапайым тәрізді: өткізгіштік қабілеттілікті олардың арасында тең бөлуге болады. Алайда, іс жүзінде бұл сұрақ біраз нақтылауды қажет етеді.

Біріншіден, мынадай сұрақ қояйық: бұл мәселенің жүктемені бақылауға қандай қатысы бар? Егер желі жөнелтушіге қандай да бір өткізгіштік қабілеттілік санын ұсынатын болса, онда ол бар нәрсені пайдалануы тиіс. Алайда, іс жүзінде желілер ағын немесе байланыс үшін өткізгіштік қабілеттіліктің қатаң шектелген бір бөлігін қорда сақтамайды. Әрине, ол кейбір ағындар үшін бұны жасай алды (егер қызмет көрсету сапасын қолдау бар болса), бірақ әдетте, көптеген байланыстар қолжетімді өткізгіштік қабілеттілікті пайдаланып қалу-

ға тырысады. Сонымен бірге, желі ресурстарды бір топ байланысқа бірге пайдалануға бөлетін нұсқа да бар. Мысалы, IETF дифференциалды қызмет көрсетуі трафикті екі классқа бөледі. Бір IP-маршруттауыш байланыстары жиі ортақ өткізгіштік қабілеттілікті пайдаланады. Мұндай жағдайда бәсекелес байланыстар арасында өткізгіштік қабілеттілікті бөлу – жүктеуді бақылау механизмі арқылы жүзеге асырылуы тиіс.

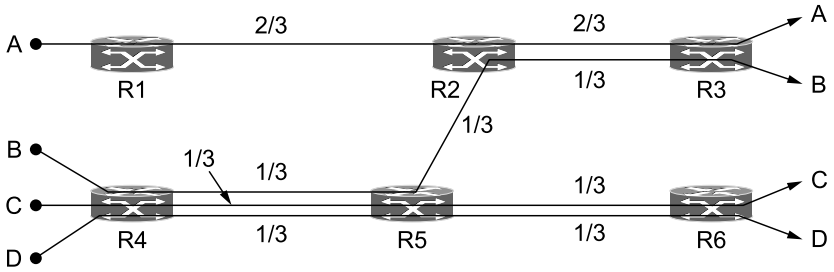
Екіншіден, ағындар және желі теңқұқықтығы дегеніміз не, онша түсінікті емес. Егер *N* ағын бір арнаны пайдаланса, онда шешім қарапайым: олардың әрқайсысына *1/N* өткізгіштік қабілеттілік бөлінеді (импульстік трафик жағдайында бұл шама тиімділік тұрғысынан біршама кіші болады). Ал егер ағындар әртүрлі, бірақ қиылысатын жолдарды пайдаланса не болады? Мысалы, егер бір ағын үш арна, ал қалғандары бір арна арқылы өтетін болса? Демек, үш арна арқылы өтетін ағын желілік ресурстарды көбірек пайдаланады. Сондықтан оған өткізгіштік қабілеттілікті басқаларға қарағанда (бір арна арқылы өтетін) азырақ бөлген дұрыс болар еді. Бұдан басқа, үш арналық ағын өткізгіштік қабілеттілігін азайту арқылы жаңа бір арналық ағындарды қосу мүмкіндігін қамтамасыз еткен дұрыс болар еді. Айтылған мысал көрсеткендей, теңқұқылық және тиімділік арасында қайшылық бар.

Алайда, біз теңқұқықтылықтың желілік жолға тәуелсіз анықтамасын қолданатын боламыз. Тіпті осындай қарапайым модельде байланыстар арасында өткізгіштік қабілеттілікті тең бөлу айтарлықтай күрделі мәселе болып саналады. Себебі әртүрлі байланыстар әртүрлі жолды пайдаланады, ал бұл жолдардың өткізгіштік қабілеттіктері де әртүрлі. Бұл қандай да бір ағынның төмендемелі арнасында жіңішке орын пайда болып, осындай ағын жоғарыламалы арнаның басқаларға қарағанда аз бөлігін алуы мүмкін. Бұл жағдайда басқа ағындарға бөлінетін өткізгіштік жағдайдың төмендеуі, «тұрып» қалған ағындар жұмысын түзетпейді тек баяулатады.

Көп жағдайда желіде **ең үлкен өлшем бойынша тең қолжетімдік** деп аталатын теңқұқықтық формасын пайдаланған ыңғайлы. Бұл бір ағынның өткізгіштік қабілеттілігін үлкейту, басқа бір өткізгіштік қабілеттілігі аз немесе тең ағынның өткізгіштік қабілеттілігін кемітпей мүмкін емес дегенді білдіреді. Басқа сөзбен айтқанда, ағынның өткізгіштік қабілеттілігін үлкейткеннен аз «қамтамасыз» етілген ағындар азап шегеді.

Мысал қарастырайық. *6.17-суретте* өткізгіштік қабілеттілікті ең үлкен өлшем бойынша, тең қолжетімдік бойынша *A, B, C* және *D* төрт ағыны бар желіде тарату көрсетілген. Арналарды байланыстыратын барлық маршруттауыштар 1 деп алынған бірдей өткізгіштік қабілеттілікке ие (әдетте, іс жүзінде бұлай болмайды). Төменде, солжақта (*R4* және *R5* маршруттауыштарының ортасында) орналасқан маршруттауыштың өткізгіштік қабілеттілігіне үш ағын таласады. Сондықтан олардың әрқайсысы өткізгіштік қабілеттіліктің 1/3-ін алады. Қалған А ағыны *R2*-ден *R3* аралығындағы аумақта В-мен бәсекелеседі. В-ға арна өткізгіштік қабілеттіліктің 1/3-ін бөлінгендіктен, А қалған 2/3 бөлікке ие болады. Суретте көрсетілгендей басқа арналарда өткізгіштік қабілеттілік қолданылмайды. Бірақ бұл ресурстарды қандай да бір ағынға, басқа әлдеқайда

баяу ағынның өткізгіштік қабілеттігін төмендетпей бере салуға болмайды. Мысалы, егер *R2*-ден *R3* аралығындағы аумақта *B* ағынына көбірек өткізгіштік қабілеттілік бөлсек, онда *A* ағыны бөлінген өткізгіштік қабілеттілік азаяды. Бұның қисыны бар, себебі дәл осы сәтте *A* ағынының өткізгіштік қабілеттілігі жоғары. Бірақ *B* ағынының өткізгіштік қабілеттілігін жоғарылату үшін *C* немесе *D* (не екеуінің де) ағындарының өткізгіштік қабілеттілігін төмендету керек, сонда оның (немесе олардың) өткізгіштік қабілеті *B*-дан төмен болады. Сонымен, бұл тарату ең үлкен өлшемге сәйкес келеді.



6.17-сурет. Төрт ағын үшін өткізгіштік қабілеттілікті ең үлкен өлшем бойынша тарату

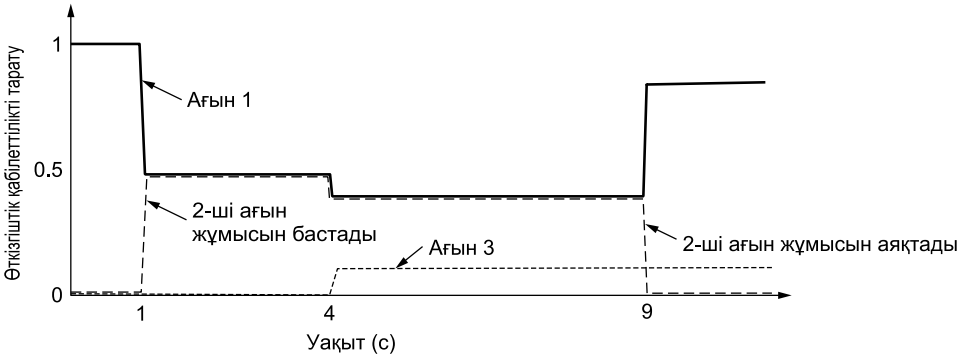
Өткізгіштік қабілеттілікті ең үлкен өлшем бойынша таратуды бүкіл желі деректерінің негізінде есептеп шығуға болады. Бұны көрнекті түрде елестету үшін барлық ағындардың жылдамдығы 0-ден бастап баяу өседі делік. Қандай да бір ағында жіңішке орын пайда болған кезде, оның жылдамдығы өсуді тоқтатады. Әрі қарай қалған ағындардың жылдамдығы өзінде жіңішке орын пайда болғанша өсе береді (өткішгіштік қабілеттік олардың арасында тең бөлініп отырады).

Үшіншіден, теңқұқықтықтың қай деңгейде қарастырылатыны түсініксіз. Желі екі хост арасындағы немесе бір хосттың барлық байланыстары деңгейінде ағындар теңқұқықтығын қамтамасыз ете алады. Біз бұл мәселені салмақты әділ қызмет көрсету жайлы (*5.4-бөлімді қараңыз*) айтқан кезде талқылағанбыз және әр нұсқаға байланысты белгілі бір қиындықтардың бар екенін анықтаған болатынбыз. Мысалы, егер барлық хосттарды теңқұқықты деп санасақ, онда белсенді жұмыс істейтін сервер қарапайым мобильді телефоннан артық ресурс ала алмайды. Егер барлық байланыстарды теңқұқықты деп санасақ, онда хосттарға қосымша жаңа байланыстар ашқан тиімді болады. Бұл сұраққа бірегей дұрыс жауап жоқ болғандықтан теңқұқықтық көбіне байланысқа қатысты қарастырылады. Алайда, бұндай теңқұқықтылықтың дәл болуы міндетті емес. Бірінші кезекте ешбір байланыстың өткізгіштік қабілеттіліксіз қалмауы міндетті. Іс жүзінде, TCP қатаң бәсекелестік тудыратын көптік байланыстар ашуға мүмкіндік береді. Бұл тактика қосымшаларға, әсіресе, желі мүмкіндігіне талаптары жоғары қосымшаларға қолайлы. Мысалы, оны BitTorrent файлдарды біррангілі бірлесіп пайдалану үшін қолданады.

Жинақтылық

Соңғы өлшем, жүктемені бақылау алгоритмі өткізгіштік қабілеттілікті таратуға әділ және тиімді жинақталуы тиіс. Жұмыс режимін таңдаған кезде біз желілік орта статикалық деп есептедік. Алайда, іс жүзіноде байланыс бірде пайда болып, бірде жоғалып кетеді, ал әрбір жеке байланысқа қажет өткізгіштік қабілеттілік уақытпен өзгереді – мысалы, тұтынушы веб-парақты ұзақ уақыт қарап отырып, кенеттен үлкен бейне файлды жүктей бастауы мүмкін.

Желіге деген талаптардың мұндай тұрақсыздығы, мінсіз жұмыс режимінің үнемі өзгеріп отыруына әкеледі. Жүктемені бақылаудың жақсы алгоритмі мінсіз жұмыс режимін жылдам жинақтауы тиіс және уақыт өте оның өзгерістерін қадағалап отыруы керек. Егер алгоритм тым баяу жинақталатын болса, онда ол жұмыс режимінің өзгерістеріне үлгере алмайды. Егер алгоритм тұрақсыз болса, онда кейбір жағдайларда ол қажет нүктеде жинақталмайды немесе оның төңірегінде ұзақ толғанады.

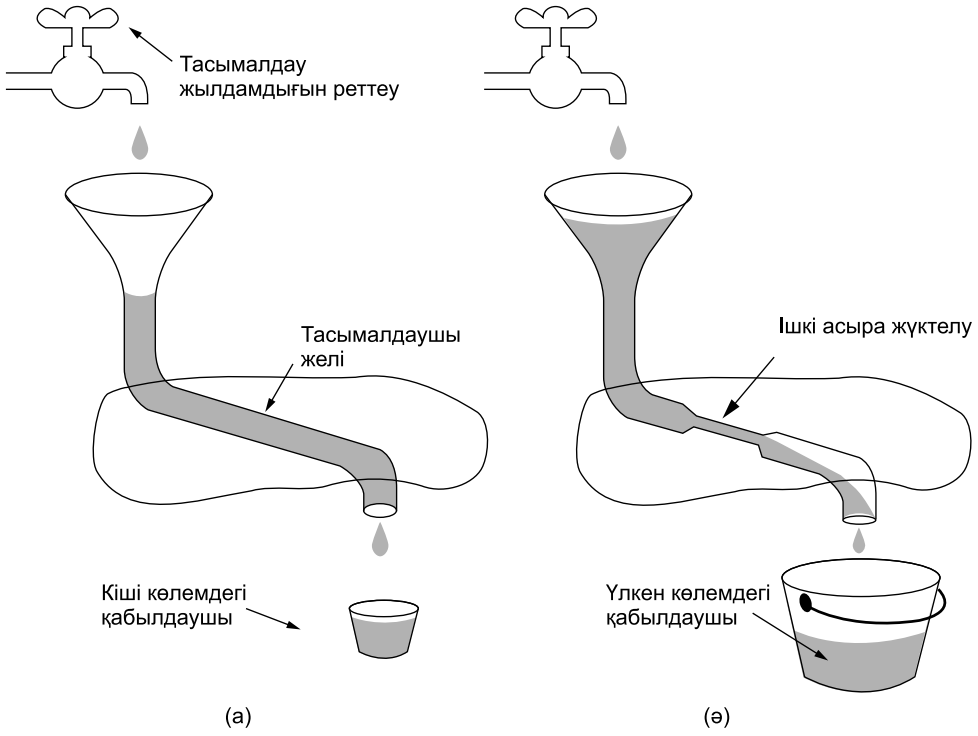


6.18-сурет. Уақыт өте өткізгіштік қабілеттілікті таратудың өзгеруі

6.18-суретте уақыт өте өзгеріп және тез жинақталып отыратын өткізгіштік қабілеттілікті тарату мысалы көрсетілген. Бастапқыда 1 ағын бүкіл өткізгіштік қабілеттілікті алады. Бір секундтан кейін 2 ағын жұмыс істей бастайды. Оғанда өткізгіштік қабілеттік керек. Сондықтан екі ағында ½ өткізгіштік қабілеттілікті алатындай етіп, тарату тез өзгереді. Тағы үш секундтан кейін үшінші ағын пайда болады. Ол өткізгіштік қабілеттіліктің тек 20%-ын пайдаланады. Бұл теңқұқықтық тұрғысынан алғанда (1/3) оған тиесілі өткізгіштік қабілеттіліктен аз. Бірінші және екінші ағындар жағдайдың өзгеруіне тез әрекет етіп, әрқайсысы 40% өткізгіштік қабілеттілікті алады. Уақыт сәтінің 9 с-да екінші ағын өшеді, ал үшінші ағын өзгеріссіз жұмыс істей береді. Бірінші ағын тез арада 80% өткізгіштік қабілеттілікті ала қояды. Уақыттың әр сәтінде өткізгіштік қабілеттілік қосындысы шамамен 100%-ға тең. Демек, желі мүмкіндігі толық пайдаланылады және бәсекелес ағындар тең жағдайда болып келеді (бірақ өткізгіштік қабілеттілікті өзіне қажетінен артық пайдаланбайды).

6.3.2. Жөнелту жылдамдығын реттеу

Енді негізгі сұраққа көшетін уақыт келді: қажет өткізгіштік қабілеттілікті алу үшін жөнелту жылдамдығын қалай реттеуге болады? Жөнелту жылдамдығы екі факторға тәуелді болуы мүмкін. Біріншісі – деректер ағынын басқару, бұл қабылдаушы буферінің сыйымдылығы жеткіліксіз болғанда қажет. Екінші фактор – асыра жүктелу, желі өткізгіштік қабілеттілігі жеткіліксіз болған кезде оны есепке алып отыру қажет. Бұл мәселе *6.19-суретте* су құбыры мысалында бейнеленген. *6.19 а-суретінде* біз сыйымдылығы аз қабылдаушыға апаратын жуан құбырды көреміз. Бұл – шектеуші фактор, деректер ағынын басқару болып саналатын жағдай. Жөнелтуші суды шелекке сиятын мөлшерден көптеп жібермегенше су асып төгілмейді. *6.19 ә-суретінде* шектеуші фактор шелек сыйымдылығы емес, желінің өтпкізгіштік қабілеті. Егер су краннан шұңғымаға тым жылдам құйылатын болса, онда шұңғымадағы су деңгейі көтеріледі, ақырында судың бір бөлігі шұңғыма сыртына ағып кетуі мүмкін.



6.19 сурет. Жылдам желі және сыйымдылығы аз қабылдаушы (а); баяу желі және сыйымдылығы үлкен қабылдаушы (ә)

Жөнелтушіге бұл мысалдар ұқсас болып көрінуі мүмкін, себебі екі жағдайда да деректерді жылдам тасымалдау нәтижесінде дестелер жоғалады. Алайда,

бұл жағдайлар әртүрлі себептер нәтижесінде орын алатын болғандықтан олар әртүрлі шешімді қажет етеді. Мүмкін деген шешімдердің бірі – деректер ағынын көлемі өзгермелі терезе арқылы басқару жайлы біз айтқан болатынбыз. Енді біз асыра жүктелуді басқарумен байланысты басқа шешімді қарастырамыз. Іс жүзінде осы мәселенің кез келгені орын алуы мүмкін болғандықтан, транспорттық хаттама екі шешімді де жүзеге асыра алуы керек.

Транспорттық хаттаманың жөнелту жылдамдығын қалай реттеу керек екендігі желіде қолданылатын кері байланыс формасына тәуелді. Әртүрлі желілік деңгейлер әртүрлі типтегі: анық және анық емес, дәл және дәл емес кері байланысты пайдалануы мүмкін.

Анық дәл кері байланыс мысалы – маршруттауыштар ақпарат көзіне өзінің қолжетімді жөнелту жылдамдығын хабарлайтын жағдай. Мысалы, XCP (eXplicit Congestion Protocol) дәл осылай жұмыс істейді (Katabi және басқалар, 2002). Анық және дәл емес схема – TCP-да ECN-ді (Explicit Congestion Notification – қанығу жайлы мәлімдеу) пайдалану. Бұл жағдайда маршруттауыштар асыра жүктелуден зардап шегетін дестелерді арнайы маркерлейді, сөйтіп жөнелтушіге деректер тасымалдау жылдамдығын төмендету керек екенін хабарлайды, алайда, жылдамдықтың қаншалықты төмендету керек екендігі көрсетілмейді.

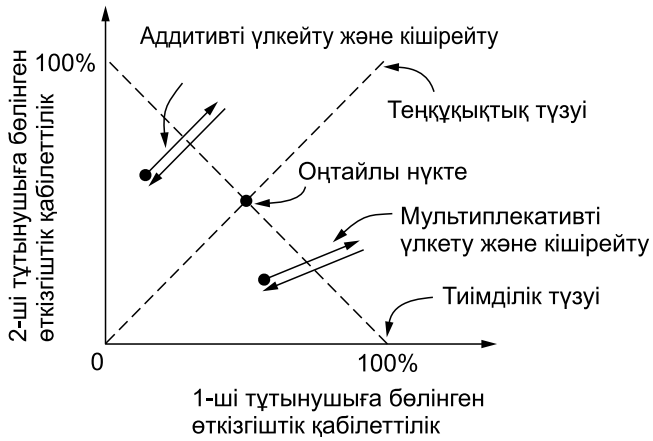
Басқа схемаларда анық сигнал жоқ. FAST TCP айнала кідірісті өлшейді және оны асыра жүктелуді бақылау параметрі ретінде пайдаланады (Wei және т.б., 2006). Соңында, қазіргі заманғы Интернетте кеңінен таралған асыра жүктелуді бақылау тәсілін TCP және кезек соңын лақтырып тастайтын маршруттауыштар немесе асыра жүктелулі кездейсоқ ерте анықтауы бар маршруттауыштар қолданады. Мұнда жоғалған дестелер саны желінің асыра жүктелу көрсеткіші болып саналады. TCP-дің бұл түрінің бірнеше нұсқасы бар – жеке алғанда, Linux жүйесінде қолданылатын CUBIC TCP (На және басқалар, 2008). Сонымен бірнеше тәсілдердің комбинациясы қолданылуы мүмкін. Мысалы, Windowsқа Compound TCP (құрама TCP) енгізілген, мұнда айнала кідіріс те, жоғалған дестелер саны да қанығу көрсеткіші болып саналады (Tan және басқалар, 2006). Бұл схемалар *6.3-кестеде* көрсетілген.

6.3-кесте. Кейбір қанығуды бақылау хаттамаларының сигналдары

Хаттама	**Сигнал**	**Анық?**	**Дәл?**
XCP	Қолдануға ұсынылатын жылдамдық	Иә	Иә
TCP ECN-мен	Асыра жүктелу жайлы ескерту	Иә	Жоқ
FAST TCP	Толассыз кідіріс	Жоқ	Иә
Compound TCP	Дестелердің жоғалуы және толассыз кідіріс	Жоқ	Иә
CUBIC TCP	Дестелердің жоғалуы	Жоқ	Жоқ
TCP	Дестелердің жоғалуы	Жоқ	Жоқ

Анық және дәл сигнал алынғаннан кейін транспорттық ішкі жүйе дестелерді жөнелту жылдамдығын жаңа жұмыс режиміне байланысты орната алады. Мысалы, егер ХСР жөнелтушілерге ұсынылатын жылдамдықты мәлімдейтін болса, онда олар осы жылдамдықты қолдана алулары мүмкін. Бірақ басқа жағдайларда транспорттық ішкі жүйелер жорамалдап әрекет етуге мәжбүр. Қанығу жайлы сигнал жоқ болған кезде жөнелтушілер тасымалдау жылдамдығын жоғарылатуы, ал мұндай сигнал бар кезде – төмендетуі тиіс. Жөнелту жылдамдығын жоғарылату және төмендетуге **басқару заңдылығы (control law)** жауап береді. Мұндай заңдылықты таңдау өнімділікке шешуші әсер етеді.

Қанығудың бинарлық белгісін қарастыра отырып, Chiu және Jain (1989) **AIMD (Additive Increase Multiplicative Decrease – аддитивті жоғарылату мультипликативті төмендету)** басқару заңы тең қолжетімдік идеясын есепке ала отырып, жұмыс режимін тиімді есептеуге мүмкіндік береді деген шешімге келді. Бұны көрсету үшін олар екі байланыс ортақ арнаның өткізгіштік қабілеттілігі үшін бәсекеге түсетін қарапайым көрнекті мысалды келтірді. *6.20-суреттегі* 1-тұтынушыға бөлінген өткізгіштік қабілеттілік X осінде, 2-тұтынушыға бөлінетін өткізгіштік қабілеттілік – Y осінде орналасқан. Әділ тарату жағдайында екі тұтынушы да тең өткізгіштік қабілеттілік санына ие болады. Мұндай таратуға үзік сызықпен көрсетілген теңқұқықтық түзуі сәйкес келеді. Барлық тұтынушыларға бөлінген өткізгіштік қабілеттілік қосындысы 100%-ға (демек, арнаның өткізгіштік қабілеттілігіне тең) болған кезде тарату тиімді болып саналады. Тиімді тарату тиімділік түзуінде орналасқан. Өткізгіштік қабілеттілік қосындысы осы түзумен қиылысқан кезде екі тұтынушы да қанығу жайлы сигнал алады. Осы түзулердің қиылысу нүктесі тұтынушылар бірдей өткізгіштік қабілеттілік санына ие болатын және желінің бүкіл өткізгіштік қабілеттілігі пайдаланылатын оңтайлы жұмыс режиміне сәйкес келеді.

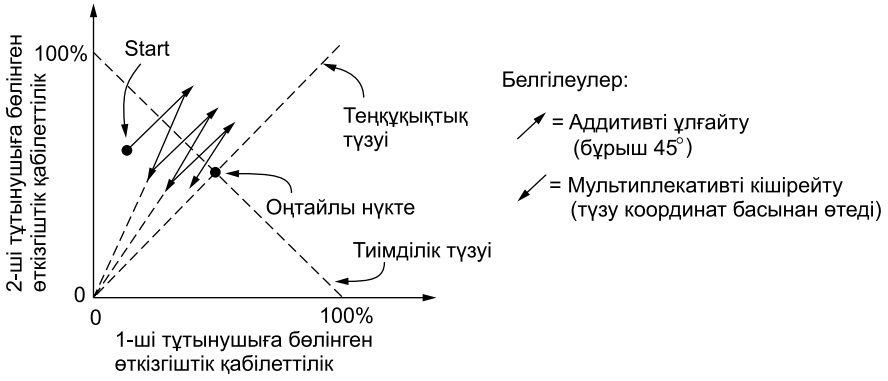


6.20-сурет. Өткізгіштік қабілеттілікті аддитивті және мультипликативті реттеу

Бастапқыда 1-тұтынушыға және 2-тұтынушыға қандай да бір өткізгіштік қабілеттілік бөлінген делік. Екі тұтынушы да өз өткізгіштік қабілеттілігін аддитивті жоғарылата бастаған кезде не болатынын қарастырайық. Мысалы, олар дестелерді жөнелту жылдамдығын әр секунд сайын 1 Мбит/с жоғарылата алады. Нәтижесінде жұмысшы режим тиімділік түзуін қиып өтеді және екі тұтынушы да желідегі қанығу жайлы сигнал алады. Осы сәтте оларға қолданыстағы өткізгіштік қабілеттілікті төмендетуге тура келеді. Алайда, аддитивті төмендету аддитивті түзу бойындағы тербеліске әкеледі (*6.20-суретті қараңыз*). Нәтижесінде жұмысшы режим тиімдірек, бірақ оның әділетті болуы міндетті емес.

Осыған ұқсас, екі тұтынушы да өз өткізгіштік қабілеттілігін желі қанықкандығы жайлы сигнал түскенше, мультиплекативті жоғарылататын жағдайды қарастырайық. Мәселен, олар әр секунд сайын дестелерді жөнелту жылдамдығын 10%-ға өсіреді делік. Егер қанығу жайлы сигнал алынғаннан кейін тұтынушылар өткізгіштік қабілеттілікті мультиплекативті төмендететін болса, онда жұмыс режимі мультиплекативті түзу бойында тербелетін болады (*6.20-суретті қараңыз*). Мультиплекативті түзу көлбеуі аддитивті түзу көлбеуінен ерекше. (Мультиплекативті түзу координаталар басы арқылы өтеді, ал аддитивті түзудің 45^0 көлбеуі бар.) Алайда, бұл бізге ештеңе бермейді. Екі жағдайда да тасымалдаудың оңтайлы жылдамдығы қол жетімсіз.

Енді басқа жағдайды қарастырайық. Айталық, тұтынушылар өткізгіштік қабілеттілікті аддитивті жоғарылатты делік. Қанығу жайлы сигнал алғаннан кейін олар өткізгіштік қабілеттілікті мультипликативті төмендете бастайды. Мұндай тәсіл AIMD басқару заңдылығы (*6.21-сурет*) деп аталады. Осындай түрлендірулер нәтижесінде жұмыс режимі өткізгіштік қабілеттілікті тарату бір мезетте әділ және тиімді болып келетін оңтайлы нүктеде қосылады және бұл бастапқы нүктеге тәуелсіз болып келеді.



6.21-сурет. AIMD басқару заңдылығы

TCP хаттамасы AIMD басқару заңдылығын тек бұл себептен ғана емес, сонымен бірге тұрақтылық тұрғысынан (себебі, асыра жүктелу қалып-күйіне

кіру одан шығуға қарағанда әлдеқайда жеңіл) пайдаланады. Алайда, өткізгіштік қабілеттілікті тарату нәтижесі тіпті, біз ойлағандай әділ емес. Себебі терезе көлемі әр байланыс үшін жеке болып саналатын айналма кідіріске байланысты тағайындалады. Сондықтан жақын арада орналасқан хосттар алыс орналасқан хосттарға қарағанда үлкен өткізгіштік қабілеттілікке ие болады (қалған тең шартты жағдайда).

AIMD басқару заңдылығы көмегімен TCP-да дестелерді жөнелту жылдамдығының асыра жүктелуді бақылаудың қалай реттелетіндігін біз *6.5-бөлімде* егжей-тегжейлі қарастырамыз. Бұл мәселе бір қарағаннан әлдеқайда күрделірек. Мәселе жылдамдықтың белгілі бір уақыт кезеңінен кейін өлшенетіндігінен туындайды. Ал трафик әдетте, мультиплексті болып келеді. Іс жүзінде дестелерді жөнелту жылдамдығының орнына жиірек сырғымалы терезе мөлшері беріледі. Мұндай стратегия TCP-да да қолданылады. Егер терезе өлшемі *W*-ға тең болса, ал айналма кідіріс – *RTT*, онда эквивалентті жылдамдық *W/RTT*-ға тең. Бұл өте ыңғайлы, себебі деректер ағынын басқару да терезе өлшемін реттеуге негізделген. Бұдан басқа, мұндай тәсілдің маңызды артықшылығы бар: егер дестелердің жеткізілгендігі жайлы растаулар келмей қалса, онда *RTT* уақыттан кейін жөнелту жылдамдығы төмендетіледі.

Соңында, кейде бір желіде әртүрлі транспорттық хаттамалар қолданылады. Егер осы хаттамалар бірмезгілде асыра жүктелуді әртүрлі басқару заңдылықтары арқылы бақыламақ болған кезде не болады? Жауап айқын: өткізгіштік қабілеттілікті тарату әділ болмайды. Интернет желісінде асыра жүктелуді бақылаудың кеңінен таралған формасы TCP болғандықтан, жаңа транспорттық хаттамаларды TCP-мен бірлесіп жұмыс істегенде ресурстарды әділ тарату шарты сақталатындай етіп құрастыруға қатаң талап қойылады. Ертеректе қолданылған ағындық тарату хаттамалары бұл талапқа сәйкес келмеді және TCP өткізгіштік қабілеттілігін едәуір төмендетті. Нәтижесінде, TCP транспорттық хаттамалары мен TCP емес хаттамалар бірлесіп, бір-біріне кедергі жасамай, асыра жүктелуді **TCP-мен достық** жағдайында бақылау идеясы дами бастады (Floyd және басқалар, 2000).

6.3.3. Сымсыз байланысу мәселелері

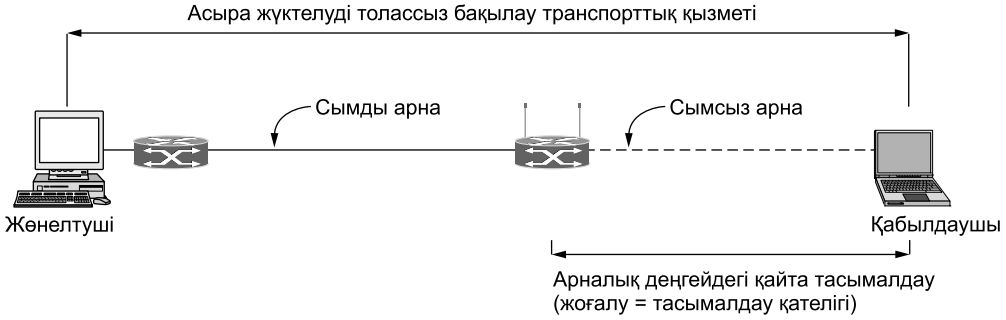
Асыра жүктелуді бақылауы бар транспорттық хаттамалар (TCP тәрізді) қолданыстағы желіге және арналық деңгей құрылғыларына тәуелсіз болуы керек. Теорияда бұл жақсы айтылғанымен, іс жүзінде сымсыз желілер жағдайында белгілі бір қиындықтар туындайды. Олардың ең негізгісі, көптеген хаттамаларда дестелердің жоғалуы (жеке алғанда TCP-де) асыра жүктелу сигналы болып келеді. Бірақ сымсыз желілерде тасымалдау қателігіне байланысты дестелердің жоғалуы жиі орын алады.

Егер хаттама AIMD басқару заңдылығын пайдаланатын болса, үлкен өткізгіштік қабілеттілік дестелер жоғалуының аз болуын қажет етеді. Padhye және басқалардың (1998) зерттеулері өткізгіштік қабілеттіліктің дестелер

жоғалу жылдамдығының квадрат түбіріне кері пропорционалды өсетіндігін көрсетті. Іс жүзінде бұл жылдам TCP-байланыстар үшін дестелер жоғалу жылдамдығы өте аз дегенді білдіреді – орта есеппен бұл көрсеткіш 1%-ды құрайды, ал көрсеткіш 10% болғанда байланыс жұмыс істеуін тоқтатады. Алайда, 802.11 жергілікті желісі тәрізді сымсыз желілер үшін кадрлардың жоғалу жылдамдығы 10% және одан да жоғары болуы әдеттегі жағдай. Егер бұны есепке алмасақ, қанығу сигналы ретінде дестелердің жоғалуын пайдаланатын асыра жүктелуді бақылау схемалары сымсыз байланыстардың жылдамдығын аса қатаң шектеп, «тұншықтырып» тастайды.

Асыра жүктелуді бақылаудың мұндай алгоритмі дұрыс жұмыс істеу үшін ол тасымалдау қателігінен емес, тек өткізгіштік қабілеттіліктің жеткіліксіздігінен жоғалған дестелерді есепке алуы керек. Мүмкін деген шешімдердің бірі – деректердің жоғалуын сымсыз арна арқылы қайта жөнелтумен жасыру. Мысалы, 802.11-де әр кадрды жеткізу үшін тоқтап, растауды күтуі бар хаттама қолданылады: кадрды жөнелту бірнеше рет қайталанылады, тек сонан кейін ғана дестенің жоғалуы жайлы ақпарат жоғары деңгейге жетеді. Көп жағдайда дестелер тағайындалған адреске жөнелту қателігіне қарамастан жеткізіледі (бұл қателіктер жайлы жоғарғы деңгей ештеңе білмейді).

Осы стратегия қолданылатын, сымсыз және сымды желі арқылы өтетін жол *6.22-суретте* бейнеленген. Мұнда екі мәселеге назар аудару қажет. Біріншіден, жөнелтуші жолда сымсыз арна бар екендігін білмейді, себебі оның көретіні – өзі тікелей жалғанған сымды арна. Интернет желісінде жолдар гетерогенді, алайда, жөнелтушіге жолдың қандай арналардан тұратынын анықтайтын әмбебап амалдар да бар. Бұл асыра жүктелуді бақылау мәселесін қиындатады, себебі сымсыз және сымды арналар үшін әртүрлі хаттама қолдану оңай мәселе емес.



6.22-сурет. Сымсыз арнасы бар жолдағы асыра жүктелуді бақылау

Екінші аспект – өзінше бір жұмбақ. Сурет дестелердің жоғалғандығы жайлы деректерге негізделген екі механизмді: арналық деңгейдегі кадрлардың қайта тасымалдануы және транспорттық деңгейдегі асыра жүктелуді бақылауды бейнелейді. Жұмбақ – бұл екі механизмнің қалайша қатар әрекет ететіндігінде. Дестенің жоғалуы тек бір механизмді іске қосуы керек, бұл не тасымалдау қателігі, не асыра жүктелу жайлы сигнал, бірақ екеуі бір мезгілде

болмауы керек. Егер екі механизм де жұмыс істей бастаса (демек, кадрды қайта тасымалдау да, тасымалдау жылдамдығын төмендету де орын алып жатыр), біз бастапқы мәселеге қайта ораламыз: схема сымсыз арналар үшін тым баяу жұмыс жасайды. Осы сұрақты сараптап, түсініп, оған жауап бере аласыз ба?

Жұмбақтың жауабы мынада – екі механизм әртүрлі уақыт масштабында жұмыс істейді. 802.11 тәрізді сымсыз желілерде арналық деңгейдегі қайта тасымалдау бірнеше микросекунд немесе миллисекунд аралығынан кейін орын алады. Дестелер жоғалуын қадағалайтын транспорттық хаттамалар таймері бірнеше миллисекунд немесе секундтан кейін іске қосылады. Үш реттік айырмашылыққа байланысты сымсыз арналар кадрлардың жоғалуын анықтап, қайта тасымалдау арқылы оны транспорттық ішкі жүйе анықтағанша шешіп қояды.

Бұндай стратегия көптеген транспорттық хаттамалардың көптеген сымсыз арналармен қалыпты жұмысы істеуі үшін қажет. Алайда, бұл шешім жарамайтын жағдайларда бар. Кейбір сымсыз арналардың, мысалы, спутниктік арнаның, айналма кідірісі үлкен. Бұндай жағдайда дестелер жоғалуын жасыра алатын **FEC (Forward Error Correction – қателіктен сақтандыра түзету)** тәрізді басқа тәсілдер қажет немесе транспорттық хаттама асыра жүктелуді бақылау үшін жоғалуларсыз сигналдарды пайдалануы тиіс.

Сымсыз арналардағы асыра жүктелуді бақылауға қатысты екінші мәселе – айнымалы өткізгіштік қабілеттілік. Бұл сымсыз арнаның өткізгіштік қабілеттілігі уақыт өте өзгеріп отырады, тіпті, кейде түйіндер жылжып, орын ауыстырған кезде және сигнал/шу қатынасы өзгергенде секірмелі өзгереді дегенді білдіреді. Ал сымды арналарда өткізгіштік қабілеттілік тұрақты болып келеді. Нәтижесінде транспорттық хаттама сымсыз арнаның өткізгіштік қабілеттілік өзгерісіне бейімделуі тиіс. Кері жағдайда желінің асыра жүктелуі орын алады немесе арна қуаты толығымен қолданылмайды.

Мүмкін деген шешімдердің бірі – бұл жайлы ойламау. Асыра жүктелуді бақылау алгоритмдері жаңа тұтынушылар қосылғанда немесе дестелерді жөнелту жылдамдығы өзгерген кезде онсыз да жақсы жұмыс істеуі керек. Сымды арнаның өткізгіштік қабілеттілігі бекітілген болғандығына қарамастан, әрбір жеке тұтынушы үшін басқа тұтынушылардың өзгермелі қылығы қолжетімді өткізгіштік қабілеттілік тербелісі ретінде қабылданады. Сондықтан құрамында 802.11 сымсыз арнасы бар жол үшін TCP хаттамасын қолданып, жақсы өнімділікке қол жеткізуге болады.

Алайда, сымсыз арнадағы жоғары тұрақсыздық салдарынан сымды байланыстар үшін құрастырылған хаттамалар өзгерістерді қадағалап үлгермеуі мүмкін, нәтижесінде өнімділік едәуір төмендейді. Мұндай жағдайда сымсыз желілер үшін құрастырылған арнайы транспорттық хаттама қажет. Әсіресе, көптеген сымсыз арналармен және мобильді маршруттауыштармен қиылысатын, сонымен бірге дестелердің жоғалу саны жоғары сымсыз ұялы желілерге арналып құрастырылған хаттамалар ерекше қызығушылық тудырады. Бұл бағыттағы зерттеулер жалғасуда. Сымсыз арналарға арналған транспорттық хаттамалар мысалдарын Li және басқалар (2009) кітабынан табуға болады.

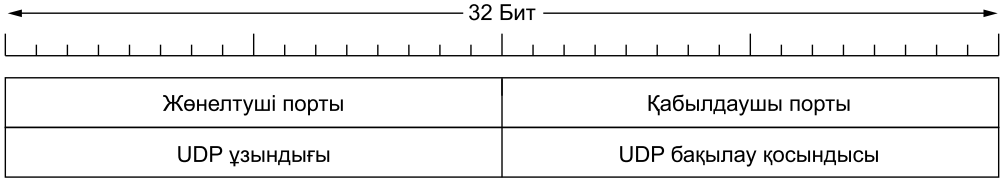
6.4. ИНТЕРНЕТ ТРАНСПОРТТЫҚ ХАТТАМАЛАРЫ: UDP

Интернетте транспорттық деңгейдің екі негізгі хаттамасы қолданылады. Олардың бірі байланыс орнатуды талап етеді, екіншісі – талап етпейді. Бұл хаттамалар бірін-бірі толықтырады. Байланысты талап етпейтін хаттама – UDP. Ол тек қосымшалар арасында дестелер жөнелтіп, оларға өздерінің жекеменшік хаттамаларын баптауға мүмкіндік береді. TCP, керісінше, байланысты талап ететін хаттама. Ол іс жүзінде бәрін орындайды – байланыс орнатады, деректерді қайта жөнелте отырып, желі сенімділігін қамтамасыз етеді, сонымен бірге деректер ағынын басқарып, асыра жүктелуді қадағалайды – және бұның барлығы хаттаманы пайдаланатын қосымшалар атынан жүргізіледі.

Келесі бөлімдерде біз UDP және TCP қарастырамыз. UDP қарапайым болғандықтан біз оны бірінші қарастырамыз. Транспорттық деңгейдің UDP хаттамасы әдетте, операциялық жүйеде іске қосылады, ал UDP пайдаланатын хаттамалар тұтынушылар кеңістігінде іске қосылады, сондықтан бұл қолданысты қосымша деп есептеуге болады. Алайда, оларда қолданылатын тәсілдер басқа да көптеген қосымшаларға өте пайдалы, сондықтан оларды транспорттық қызметтің бір бөлігі деп қарастыруға болады. Сондықтан біз олар жайлы да әңгіме қозғаймыз.

6.4.1. UDP негіздері

Интернет хаттамалары жиынтығының ішінде байланыс орнатусыз жұмыс істейтін **UDP (User Datagram Protocol – тұтынушы дейтаграммаларын тасымалдау хаттамасы)** транспорттық хаттамасы бар. UDP қосымшаларға инкапсуляцияланған IP-дейтаграммаларды байланыс орнатусыз жөнелтуге мүмкіндік береді. UDP RFC768 құжатында сипатталған.



6.23-сурет. UDP тақырыбы

UDP хаттамасының көмегімен 8-байттық тақырыптан және одан кейін пайдалы жүктемеден тұратын **сегменттер** жөнелтіледі. Тақырып *6.23-суретте* көрсетілген. Екі **порттың** нөмірі жөнелтуші және қабылдаушы машиналар ішінде сокеттерді сәйкестендіру үшін қызмет етеді. UDP дестесі келген кезде оның пайдалы жүктеме өрісіндегі ақпарат тағайындалған портпен байланысқан үдеріске беріледі. Бұл байланыстыру `BIND` типті базалық операцияны орындағанда жүзеге асырылады. Бұл *6.1-листингінде* TCP-ға қатысты көр-

сетілген болатын (UDP-да байланыстыру үдерісі дәл солай жүреді). Өзіңізге порттарды қосымшалар жалға алатын пошта жәшігі ретінде елестетіңіз. Біз оны TCP талқылағанда егжей-тегжейлі қарастыратын боламыз, ол да портты қолданады. Іс жүзінде әдеттегі IP орнына UDP қолданудың мәні – жөнелтуші және қабылдаушы порттарын көрсету. Бұл екі өріссіз транспорттық деңгейде әр кіріс дестемен орындалатын іс-әрекетті анықтау мүмкін емес. Порт өрістеріне сәйкес кіріктірілген сегменттерді сәйкес қосымшаларға жеткізу жүзеге асырылады.

Ақпарат көзі порты жайлы ақпарат алдымен жөнелтушіге жіберілетін жауап құрастырылған кезде қажет. Кіріс сегменттегі *Ақпарат көзі порты* өрісінің мәні шығыс сегменттегі *Тағайындалған порт* өрісіне көшіріліп, жөнелтілетін жауаптың қарсы бетте нақты кімге арналғандығын көрсетеді.

UDP ұзындығы өрісі тақырыптан және деректердоен тұрады. Ең кіші ұзындық тақырып ұзындығына тең, демек, 8 байт. Ең үлкен ұзындық 65 515 байтқа тең – бұл IP-дестe өлшеміне қойылатын шектеуге байланысты 16 бит көмегімен жазылатын ең үлкен саннан кіші.

Міндетті емес *Бақылау қосындысы* өрісі сенімділікті жоғарылату үшін қажет. Бұл өрісте тақырып, деректер және жалған тақырып бақылау қосындысы жазылған. Есептеулер орындалған кезде *Бақылау қосындысы* өрісі мәні нөлге теңестіріледі, ал деректер өрісінің ұзындығы тақ сан болса, ол нөлдік байтпен толықтырылады. Бақылау қосындысын есептейтін алгоритм барлық 16-разрядтық сөздерді қосымша кодта қосып шығады, сонан кейін барлық қосынды үшін қосымшаны есептейді. Соңында қабылдаушы бүкіл сегменттің бақылау қосындысын, *Бақылау қосындысы* өрісімен қоса есептеген кезде нәтиже нөлге тең болуы керек. Егер ол есептелмесе, оның мәні 0-ге тең (нағыз нөлдік бақылау қосындысы бірліктермен кодталады). Алайда, бақылау қосындысын есептеу функциясын өшіріп тастау, тек жоғары дәрежедегі өнімділік қажет болған жағдайдан (мысалы, сандық форматтағы дауысты жөнелткенде) басқа кезде қателік болады.

6.24-сурет. UDP бақылау қосындысына кіріктірілген жалған тақырып

IPv4 жағдайында жалған тақырып *6.24-суретте* көрсетілгендей болады. Ол жөнелтуші және қабылдаушының 32-разрядты IP-адресінен, UDP (17) ар-

налған хаттама нөмірінен, UDP-сегментке арналған байттар санауышынан (тақырыпты қоса) тұрады. UDP бақылау қосындысына жалған тақырыпты қосу хаттамалар иерархиясын бұзғанымен, себебі ондағы IP-адрестер UDP-деңгейіне емес IP-деңгейге жатады, ол дестелердің дұрыс емес жеткізілгендігін анықтауға мүмкіндік береді. TCP-де бақылау қосындысы үшін дәл осындай жалған тақырып қолданылады.

UDP нені орындамайтынын тікелей айтқан дұрыс болатын шығар. Сонымен, UDP деректер ағынын басқарумен, асыра жүктелуді қадағалаумен, бұзылған сегментті қайта жөнелтумен *айналыспайды.* Бұның барлығы тұтынушы үдерістеріне жүктеледі. UDP нені орындайды? UDP порттар және міндетті емес толассыз қателіктерді анықтауды пайдаланып, бірнеше үдерістерді қайта мультиплекстеу арқылы IP-ға интерфейс ұсынады. UDP орындайтын барлық әрекеттері міне, осы.

Ағынды басқарып, қателіктер және уақыт аралығын басқарғысы келетін үдерістер үшін UDP – дәрігердің жазып берген емі. Бұның өте пайдалы болатын аумақтарының бірі – клиент-серверлік қосымшалар. Клиент серверге көбіне қысқа сұраныс жөнелтеді және қысқа жауап алуға сенеді. Егер сұраныс немесе жауап жоғалса, клиент белгілі бір уақыт аралығы өткеннен кейін сұранысты қайталауға талпыныс жасайды. Бұл тек кодты қарапайымдатып қоймай, сонымен бірге бастапқы баптауды қажет ететін хаттамаларға қарағанда (TCP тәрізді), қажетті мәлімдемелер санын азайтуға мүмкіндік береді.

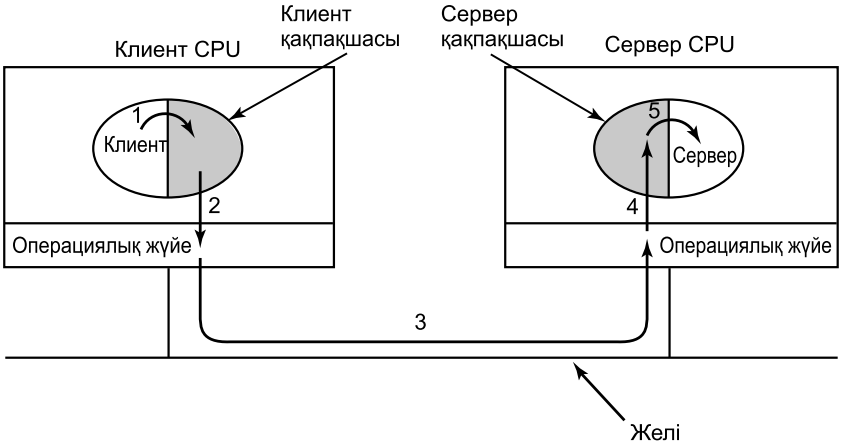
DNS (Domain Name System – домен аттары қызметі) – бұл UDP пайдаланатын қосымша, жоғарыда дәл осылай сипатталған. Біз оны *7-тарауда* қарастырамыз. Қысқаша айта кететін болсақ, егер программаға IP-адресті хосттың аты бойынша табу керек болса, мысалы, *www.cs.berkeley.edu,* ол DNS серверге осы адрес көрсетілген UDP-десте жөнелтеді. Сервер сұранысқа жауап ретінде хосттың IP-адресі көрсетілген UDP-десте жібереді. Ешқандай алдын ала баптау, сонымен бірге жұмыс аяқталғаннан кейін байланысты үзу қажет емес. Тек желі арқылы екі мәлімдеме жөнелтіледі.

6.4.2. Қашықтықтағы процедураны шақыру

Белгілі бір мағынада қашықтықтағы хостқа мәлімдеме жөнелту және жауап алу, программалау тіліндегі функцияны шақыруға ұқсас. Екі жағдайда да бір немесе бірнеше параметрді жөнелтіп, қажет жауапты аласыз. Бұл ой құрастырушыларды желі арқылы процедураны шақыру формасында орындалатын сұраныс-жауап іс-әрекетін ұйымдастыруға әкелді. Бұндай шешім желілік қосымшалар ұйымдастыруды қарапайым және үйреншікті іске айналдырады. Мысалы, DNS серверге UDP-десте жөнелтіп, жауап күтіп, белгілі бір уақыт өткеннен кейін (егер екі жақтың бірі баяу жұмыс істейтін болса) сұранысты қайта жөнелтумен айналысатын *get_IP_address(хост_аты)* процедурасын елестетіңіз. Сөйтіп, желілік технологиялармен байланысты ерекшеліктердің барлығы программистен жасырулы болады.

Бұл саладағы негізгі жұмысты 1984 жылы Биррел (Birrell) және Нельсон (Nelson) жазған болатын. Іс жүзінде программаларға қашықтықтағы хосттарда орналасқан процедураларды шақыруға рұқсат беру ұсынылды. Бірінші машинада орналасқан үдеріс екінші машинадағы үдерісті шақырған кезде, бірінші машинаның шақырушы үдерісі оқшауланады да екінші машинада шақырылған үдеріс орындалады. Ақпарат шақырушы үдерістен параметр түрінде берілуі мүмкін және процедура нәтижесі жауап ретінде кері жөнелтіледі. Желі бойында мәлімет тасымалдау қосымша программасынан жасырын болады. Мұндай технология **RPC (Remote Procedure Call – процедураны қашықтықтан шақыру)** деген атпен танымал және көптеген желілік қосымшалардың негізі болып отыр. Дәстүр бойынша шақырушы үдеріс клиент, ал шақырылушы – сервер деп аталады. Біз бұнда да осылай атайтын боламыз.

RPC идеясы – қаршықтықтағы процедураны шақыруды мүмкіндігінше жергілікті шақыруға ұқсату. Қарапайым жағдайда қашықтықтағы процедураны шақыру үшін, клиент программасы **клиенттік қақпақша (client stub)** деп аталатын кішігірім кітапханалық процедурамен байланысқан болуы тиіс. Клиенттік қақпақша клиенттер адресі кеңістігінде серверлік процедураны бейнелейді. Сәйкесінше сервер **серверлік қақпақша (server stub)** деп аталатын процедурамен байланысқан болуы тиіс. Бұл процедуралар клиенттің серверлік процедураны шақыруы жергілікті орындалмайтынын жасырады.



6.25-сурет. Процедураны қашықтықтан шақыруды орындау қадамдары. Қақпақшалар көлеңкеленген

Қашықтықтағы процедураны шақыру кезінде орындалатын нақты қадамдар *6.25-суретте* көрсетілген. Бірінші қадамда клиенттік қақпақшасы шақырылады. Бұл процедураны жергілікті шақыру, параметрлерді стекке қарапайым түрде орналастырылады. Екінші қадамда клиенттік қақпақша параметрлері мәлімдемеге орналастырылып, осы мәлімдемені жөнелту үшін жүйелік шақыру жүзеге асырылады. Параметрлерді мәлімдемеге орналастыру **маршалинг (marshaling)** деп аталады. Үшінші қадамда операциялық жүйе мәлімдемені

клиент машинасынан серверге жөнелтеді. Төртінші қадам – операциялық жүйе кіріс дестені серверлік қақпақшаға береді. Соңғы бесінші қадамда серверлік процедура ашылған параметрлерімен шақырылады. Жауап беру кезінде де осы қадамдар орындалады, тек тасымалдау кері бағытта жүргізіледі.

Мұндағы ең маңыздысы, аты серверлік процедурамен сәйкес келетін тұтынушы жазған клиенттік процедура, әдеттегі (демек жергілікті) клиенттік қақпақшаны шақыруды орындайды. Клиенттік процедура мен клиенттік қақпақша бір адрестік кеңістікте болғандықтан, параметрлер қарапайым тасымалданады. Сәйкесінше серверлік процедура күтілулі параметрлермен, өзі орналасқан адрестік кеңістіктегі процердурамен шақырылады. Серверлік процедура тұрғысынан қарағанда, ешқандай төтенше жағдай орын алмайды. Сонымен, сокеттер көмегімен енгізу/шығаруды орындаудың орнына, желілік коммуникация процедураны қарапайым шақырумен жүзеге асырылады.

RPC тұжырымдамасының әдемілігіне қарамастан, оның өзіндік суасты тастары бар. Әңгіме ең алдымен, көрсеткіштердің параметр ретінде қолданылуы жайлы. Әдеттегі жағдайда көрсеткішті процедураға параметр ретінде берудің ешқандай қиындығы жоқ. Шақырылатын процедура көрсеткішті шақырушы процедура тәрізді қолдана алады, себебі екі процедура да бір виртуалды адрестер кеңістігінде орналасқан. Процедураны қашықтықтан шақыру кезінде көрсеткішті тасымалдау мүмкін емес, себебі клиент пен сервердің адрестік кеңістігі екі түрлі.

Кейде қандай да бір айланың көмегімен көрсеткіштерді тасымалауға болады. Айталық, бірінші параметр бүтін k санына көрсеткіш делік. Клиенттік қақпақша k маршалингін орындап, оны серверге жөнелте алады. Серверлік қақпақша алынған k айнымалысына көрсеткіш құрастырып, оны серверлік процедураға береді. Серверлік процедураның күткені де осы болатын. Серверлік процедура басқаруды серверлік қақпақшаға берген кезде, соңғысы k-ны кері клиентке жөнелтеді. Ол жерде айнымалының мәні жаңартылып, ескінің орнына жазылады (егер оны сервер өзгерткен болса). Іс жүзінде параметрлерді сілтеме арқылы тасымалдау кезінде орындалатын стандартты іс-әрекеттер тізбегі параметрлер көшірмесін тікелей және кері тасымалдаумен алмастырылды. Алайда, әрқашан бұл айланы орындау мүмкін бола бермейді – нақтырақ, көрсеткіш графқа немесе басқа күрделі деректер құрылымына сілтеме жасайтын болғанда. Осының салдарынан, біз қашықтықтан шақырылатын процедуралар параметріне белгілі бір шектеу қойылуы қажет екенін көреміз.

Екінші мәселе – деректер нашар типтелген программалау тілдерінде (мысалы, C тілі) екі вектордың (массивтердің) скаляр көбейтіндісін есептейтін процедураны, олардың өлшемін көрсетпей-ақ заңды түрде жазуға болады. Бұл құрылымдардың әрқайсының тек шақырушы және шақырылатын процедураларға белгілі қандайда бір шектеуші мәні бар. Бұндай жағдайда клиенттік қақпақша параметрлерді орналастыра алмайды: олардың өлшемін анықтау мүмкін емес.

Үшінші мәселе, әрдайым спецификация немесе код бойынша параметрлер типін анықтау мүмкін бола бермейді. Мысал ретінде, параметрлерінің саны кез келген (бірден кем емес) болуы мүмкін *printf* процедурасын келтіруге болады және параметрлер бүтін сандар (int, short, long), симвмолдық, жолдық, әртүрлі ұзындықтағы жылжымалы нүктелі нақты сан және басқа типтер болуы мүмкін. С тілінің осындай толеранттылығының салдарынан *printf* процедурасын қашықтықтан шақыру мәселесін іс жүзінде орындау мүмкін емес. Дегенмен, процедураны қашықтықтан шақырту тек С (С++) тілі қолданылмаған жағдайда ғана мүмкін деген ереже жоқ – бұл программистер арасында RPC тәсілінің беделін түсірер еді.

Төртінші мәселе ауқымды айнымалыларды қолданумен байланысты. Қалыпты жағдайда шақырушы және шақырылушы процедура ауқымды айнымалылар (параметрлерден басқа) көмегімен өзара әрекеттесе алады. Бірақ, егер шақырылатын процедура қашықтықтағы машинада болатын болса, ауқымды айнымалыларды қолданатын программа жұмыс істей алмайды, себебі ауқымды айнымалылар бөлуші ресурс ретінде қызмет ете алмайды.

Бұл мәселелер процедураны қашықтықтан шақыру тәсілінен үміт үзу керек дегені емес. Іс жүзінде ол кеңінен қолданылады, тек қалыпты жұмыс істеу үшін кейбір шектеулер қажет.

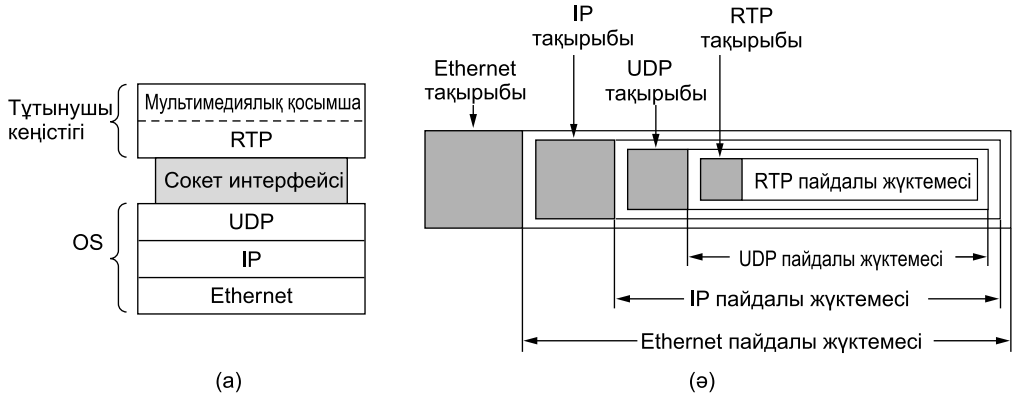
Транспорттық хаттамалар тұрғысынан UDP RPC-ді жүзеге асыруға жақсы негіз болып саналады. Қарапайым жағдайда сұраныстар және жауаптарды бір UDP-дестесімен жөнелтуге, ал өңдеуді жылдам орындауға болады. Алайда, бұл ойды жүзеге асыру үшін басқа механизмдер қажет. Сұраныс немесе жауап жоғалған жағдайда клиентке дестені қайта жөнелтуге дейінгі уақытты санап отыратын таймер қажет. Жауаптың сұранысқа нақты емес растау қызметін атқаратынына назар аударыңыздар. Сондықтан сұраныс жеке растауды қажет етпейді. Кейде параметрлер немесе нәтижелер UDP-дестесінің ең үлкен мөлшерінен үлкен болуы мүмкін, сондықтан үлкен көлемдегі мәлімдемені жөнелту үшін арнайы хаттама қажет. Егер көптік сұраныстар және жауаптар қилысатын болса (параллельді программалау жағдайындай), жауаптың сұранысқа сәйкестігі арнайы белгі көмегімен көрсетілуі тиіс.

Жоғары деңгей мәселесі – операция идемпотентті (демек, істен шығу қаупінсіз қайталанбауы тиіс) болуы мүмкін. Қарапайым жағдайда біз DNS-сұраныс және жауап тәрізді идемпотентті операциялармен жұмыс істейміз. Клиент бұндай дестелерді қатерсіз, жауап келгенше қайталап жөнелте алады. Не себеп болғаны маңызды емес: тек деста серверге жетпесе немесе жауап жоғалса болды. Кез келген жағдай, келген жауап біреу болады (әрине, осы аралықта DNS деректер базасы жаңарып кеттпесе). Алайда, операциялардың барлығы идемпотентті емес, мысалы, олар санауышты инкременттеу тәрізді жанама әрекеттерді іске қосуы мүмкін болғандықтан. Бұндай операциялар үшін RPC күрделі семантиканы талап етеді: процедураны шақыру кезінде ол бірнеше рет орындалмауы тиіс. Бұндай жағдайда TCP-байланыс орнату және сұранысты UDP емес, TCP арқылы жөнелту қажет болады.

6.4.3. Нақты уақыт масштабындағы транспорттық хаттамалар

Клиент-серверлік процедураны қашықтықтан шақыру – бұл UDP кеңінен қолданылатын аумақ. Тағы бір осындай аумақ – нақты уақыттық мультимедиалық қосымшалар. Жеке алғанда, сұраныс бойынша музыка және бейне, интернет-телефония, интернет-радио, бейнеконференциялардың және басқа да мультимедиялық қосымшалардың танымал бола бастауымен бірге, олардың барлығының азды-көпті ұқсас нақты уақыттық транспорттық хаттамаларды қолдана отырып, қайтадан велосипед құрастыра бастағаны түсінікті болды. Біраз уақыт өткеннен кейін, мультимедиялық қосымшаларға арналған ортақ транспорттық хаттаманың бар болғандығы жақсы болар еді деген ой келді.

Сөйтіп, **RTP (Real-Time Transport Protocol – нақты уақыт масштабындағы транспорттық хаттама)** хаттамасы пайда болды. Ол RFC 3550 құжатында сипатталған және қазір мультимедиалық қосымшалар үшін кеңінен қолданылады. Біз деректерді нақты уақытта тасымалдаудың екі тұсын қарастырамыз. Біріншісі – аудио және бейнені дестелік тасымалдау үшін қолданылатын RTP хаттамасы. Екінші тұсы – қалпына келтіруге арналған деректерді өңдеу (көбіне қабылдаушы жақта). Бұл функциялар *6.26-суретте* бейнеленген хаттамалар стеніне кіреді.



6.26-сурет. RTP: а – хаттамалар стегіндегі RTP орны; ә – дестелердің кіріктірілуі

RTP әдетте UDP-ның үстінде жұмыс істейді (операциялық жүйелерде). Бұл былайша орындалады. Мультимедиялық қосымша бірнеше аудио-, бейне-, мәтіндік және басқа да ағындардан тұруы мүмкін. Олар RTP кітапханасында жазылады. Кітапхана да, қосымша тәрізді тұтынушы кеңістігінде орналасады. Кітапхана ағындарды тығыздап, сокетке жөнелтілетін RTP дестелеріне орналастырылады. Сокеттің екінші басында (операциялық жүйеде) UDP дестесі генерацияланып, RTP-дестеге орналастырылады. Олар арна арқылы жөнелтілу

үшін (мысалы, Ethernet) IP хаттамаға беріледі. Қабылдаушы хостта кері үдеріс орын алады. Соңында мультимедиялық қосымша деректерді RTP-кітапханадан алады. Ол мультимедианы қалпына келтіруге жауап береді. Сипатталған жағдай хаттамалар стегі *6.26 а-суретінде* көрсетілген. Кіріктірілген дестелер жүйесі *6.26 ә-суретінде* көрсетілген.

Нәтижесінде, тек RTP-дің қандай деңгейге тиесілі екенін анықтау қалады. Хаттама тұтынушы кеңістігінде жұмыс істейтін және қосымша программамен байланысты болғандықтан, оның қосымша хаттама тәрізді екені сөзсіз. Екінші жағынан ол ортақ, транспорттық қызметтер ұсынатын қосымшаға тәуелсіз хаттама. Осы тұрғыдан ол транспорттық хаттама болып көрінуі мүмкін. Ең дұрысы оны мынадай етіп анықтау: RTP – бұл кездейсоқ қолданбалы деңгейде орналасқан транспорттық хаттама, сондықтан біз оны осы тарауда қарастырып отырмыз.

RTP – нақты уақыт масштабындағы транспорттық хаттама

RTP-дің негізгі функциясы – нақты уақыт масштабындағы бірнеше ағындарды бірыңғай UDP дестелер ағынына тығыздау. UDP ағынын бір бағытта немесе бірден бірнеше адресатқа бағыттауға болады. RTP әдеттегі UDP пайдаланатын болғандықтан оның дестелерін маршруттауыштар өңдейді, егер оған IP деңгейіндегі қызмет көрсету сапасы қасиеті қосылмаса. Жеке алғанда, жеткізуге ешбір кепілдік жоқ және дестелер жоғалуы, кідіруі, бұзылуы және т.б. мүмкін.

RTP форматының қабылдаушыға мультимедиялық деректерді өңдеуге көмектесетін бірнеше ерекшеліктері бар. RTP ағыны жөнелтетін әр дестенің алдыңғы дестеден бірге артық болып келетін нөмірі бар. Мұндай амал қабылдаушыға дестенің жоғалғанын анықтауға мүмкіндік береді. Егер қандай да бір дестенің жоғалғаны анықталса, онда қабылдаушыға оңтайлы шешімді таңдау қосымшаға жүктеледі. Егер дестелер бейнедеректерден тұратын болса, онда жоғалған кадрларды қалдыруға болады. Аудиодеректер жағдайында жоғалған мәндерді интерполяция көмегімен жуықтап келтіруге болады. Бұл жағдайда қайталап жөнелту жақсы шешім емес, себебі бұл көп уақыт алады және қайталап жөнелтілген деста ешкімге керек болмай қалады. Сондықтан RTP хаттамасында ешқандай растау және қайталап тасымалдауға сұраныс механизмі қарастырылмаған.

RTP пайдалы жүктеме өрісінде, форматы жөнелтуші қосымша форматына сәйкес келетін деректердің бірнеше элементі орналасуы мүмкін. RTP хаттамасындағы желіаралық әрекет әрқайсысына бірнеше кодтау форматтарын сәйкестендіруге болатын, бірнеше профильдерді (мысалы, жеке аудио ағындар) анықтау арқылы жүзеге асырылады. Айталық, аудио ағын PCM көмегімен (8-биттік символдары бар 8 кГц жолақ) дельта кодтау, болжамалы кодтау, GSM-кодтау, MP3-кодтау және т.б. кодталуы мүмкін. RTP-де ақпарат көзі кодтау тәсілін көрсете алатын арнайы тақырып өрісі бар, алайда, әрі қарай ақпарат көзі кодтау үдерісіне ықпал ете алмайды.

Нақты уақыт қосымшаларына қажет тағы бір функция – уақыт белгісі. Мұндағы ой – ақпарат көзіне әр дестенің бірінші элементімен уақыт белгісін байланыстыруға мүмкіндік беру. Уақыт белгісі ағынды жөнелтудің бастау сәтіне байланысты қойылады, сондықтан тек *белгілер арасындағы* кезеңдер ғана маңызды. Абсолюттік мән іс жүзінде ешқандай рөл атқармайды. Жақын арада біз осындай механизмнің қабылдаушыға деректердің біраз бөлігін буферлеп, әр бөлігін ағын басынан дұрыс миллисекундтар саны өткеннен кейін, нақты бөлік жазылған дестенің қай кезде келгеніне тәуелсіз ойнатуға мүмкіндік беретінін көреміз.

Бұның салдары желілік кідірістің толқу әсерін төмендетіп қоймай, сонымен бірге бірнеше ағындарды өзара синхрондауға мүмкіндік береді. Мысалы, сандық теледидарда бейне ағын және екі аудио ағын болуы мүмкін. Екінші аудио ағын әдетте стерео немесе фильмнің шет тілде қайталанған дыбыстық жолының үнін үшін қажет. Әр ағынның өз физикалық көзі бар, алайда, бірегей таймер арқылы генерацияланатын уақыттық белгілер көмегімен, тіпті, тасымалдау немесе қабылдау кезінде қателіктер кетсе де ағындарды синхрондауға болады.

6.27-сурет. RTP тақырыбы

RTP тақырыбы *6.27-суретте* көрсетілген. Ол үш 32-разрядтты сөзден және мүмкін деген кеңейтілімдерден тұрады. Бірінші сөзде *Версия* өрісі бар, қазіргі кезде оның мәні – 2. Ағымдағы версия соңғы немесе соңғының алдындағы болады деп үміттенеміз, себебі оны сәйкестендіретін екі биттік өрісте тек бір жаңа нөмірге орын қалды (дегенмен, 3 коды версияның нақты нөмірі кеңейтілім өрісінде орналасқан дегенді білдіруі мүмкін).

P биті десте мөлшері толтыру байттарының есебінен 4 байтқа еселі етіп жасалған дегенді білдіреді. X биті кеңейтілген тақырыптың бар екенін білдіреді. Кеңейтілген тақырыптың форматы және қызметі анықталмайды. Ол үшін тек кеңейтілімнің бірінші сөзінде кеңейтілімнің жалпы ұзындығы болуы міндетті. Бұл – келешектегі әртүрлі, болжауға келмейтін талаптар үшін қордағы мүмкіндік.

CC өрісі ағынның неше қызметтес ақпарат көздерін біріктіретінін хабарлайды. Олардың саны 0-ден 15-ке дейін болуы мүмкін (төменде қара). *M* биті – нақты қосымшамен байланысқан маркер. Ол бейне кадрдың басын, аудио арнадағы сөздің басын немесе тағы да басқа қосымшаға түсінікті маңызды нәрсені белгілеу үшін пайдаланылады. *Деректер типі* өрісінде қолданыстағы кодтау алгоритмі жайлы ақпарат жазылады (мысалы, тығыздалмаған 8-биттік аудио, MP3 және т.б.). Мұндай өріс әр дестеде бар болғандықтан, кодтау тәсілі тікелей ағынды тасымалдау кезінде өзгеруі мүмкін. *Реттік нөмір* – бұл әр RTP дестесінде инкременттелетін қарапайым санауыш. Ол жоғалған дестелерді анықтау үшін пайдаланылады.

Уақыт белгісі ағын көзімен генерацияланады және дестенің бірінші сөзі құрастырылған сәтті жазу үшін қызмет етеді. Уақыт белгілері уақыт толқынысының әсерін немесе қабылдауыштағы джиттерді, қалыпқа келтіру сәтін дестенің келу уақытына тәуелсіз етуді төмендетуге мүмкіндік береді. *Синхрондау көзін сәйкестендіру* – дестенің қай ағынға тиесілі екенін анықтауға мүмкіндік береді. Бұл – бірегей UDP-дестелер ағыны түрінде қозғалатын бірнеше деректер ағынын мультиплекстеу және кері мультеплекстеу үшін қолданылатын тәсіл. Соңғы өріс, *қызметтес көздерді сәйкестендіру* бар болған жағдайда, соңғы ағын бірнеше ақпарат көздерінен құрастырылған кезде қолданылады. Бұл жағдайда араластырушы құрылғы синхрондау көзі болып табылады, ал көздерді сәйкестендіру өрісінде араласқан ағындар тізімі көрсетіледі.

RTCP – нақты уақытық басқарушы транспорттық хаттама

RTP хаттамасының **RTCP (Real-Time Transport Control Protocol – нақты уақыттық басқарушы транспорттық хаттама)** атты кішігірім туыстас хаттамасы бар. Бұл хаттама RFC 3550 құжатында RTP-мен бірге анықталған. Ол кері байланысты қолдаумен, синхрондаумен, тұтынушы интерфейсін қамтамасыз етумен айналысады, бірақ ешқандай медиадеректерді тасымалдамайды.

Оның бірінші функциясы кідірістер, уақыт кідірісі толқыны немесе джиттер, өткізгіштік қабілеттілік, асыра жүктелу және де ақпарат көзі хабарлайтын басқа желі қасиеттері бойынша кері байланыс үшін пайдаланылады. Алынған ақпаратты кодтаушы, желі қалып-күйі мүмкіндік бергенде деректерді жөнелту жылдамдығын өсіру немесе желіде қандайда да бір қиындық туындағанда жылдамдықты төмендету үшін есепке алады (бұл сапаны жақсартуға әкеледі). Тұрақты кері байланыс, кодтау алгоритмдерін ағымдағы жағдайда ең жақсы сапаны қамтамасыз ету үшін динамикалық баптауды қамтамасыз етеді. Мысалы, ағынды тасымалдау кезінде өткізгіштік қабілеттілік жоғарылауы да, төмендеуі де мүмкін, осыған байланысты кодтау тәсілі де өзгереді – айталық, MP3 8-биттік PCM-мен немесе дельта-кодтаумен алмастырылуы мүмкін. *Деректер типі* өрісі қабылдаушыға осы десте үшін қандай кодттау алгоритмі пайдаланылғанын хабарлайды. Бұл талап бойынша ағынды тасымалдау кезінде кодттау алгоритмін өзгертуге мүмкіндік береді.

Кері байланыс мәселесі RTCP мәлімдемесінің барлық қатысушыларға бірдей таратылатындықта. Бұл жағдайда, егер топ үлкен болса, онда RTCP пайдаланатын өткізгіштік қабілеттілік күрт өседі. Бұны болдырмас үшін RTCP жөнелтушілері, барлығын қоса алғанда олар медиадеректер тасымалдау үшін қолданылатын өткізгіштік қабілеттіліктің 5%-ын алатындай етіп, өздерінің мәлімдеме жөнелту жылдамдығын төмендетеді. Ал әр қатысушы осы өткізгіштік қабілеттілікті (бұл ақпаратты ол жөнелтушіден ала алады) және қатысушылар санын (ол оны RTCP мәлімдеме негізінде есептейді) білуі тиіс.

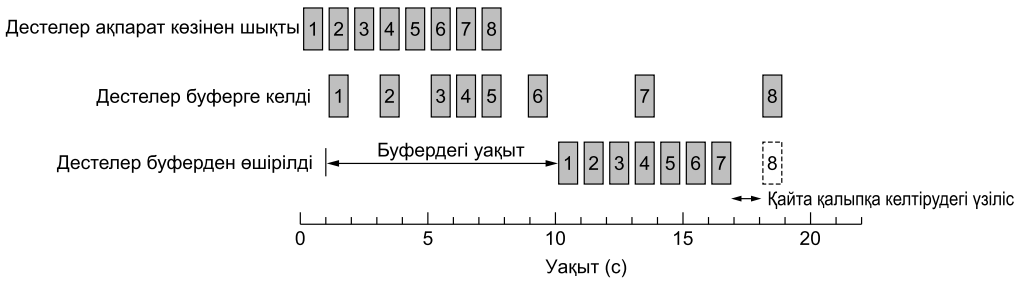
Сонымен бірге RTCP ағын аралық синхрондауы орындайды. Бұл жердегі қиындық – әр ағын әртүрлі мүмкіндігі және жылдамдық қалқуы бар таймерді пайдаланады. RTCP осы мәселені шешуге және параметрлері әртүрлі ағындарды синхрондауға көмектеседі.

Соңында, RTCP түрлі ақпарат көздеріне ат беруге (мысалы, әдеттегі ASCII-мәтін көмегімен) мүмкіндік береді. Бұл ақпарат қабылдаушыда бейнеленіп, ағымдағы ағын көзін анықтауға көмектеседі.

RTP жайлы егжей-тегжейлі ақпаратты Perkins (2002) басылымынан табуға болады.

Қалпына келтіру кезінде буферлеу және джиттерді басқару

Ақпарат қабылдаушыға келген кезде, дұрыс уақыт мезетінде қалпына келтіруді бастаған жөн. Жалпы жағдайда бұл уақыт сәті RTP-дестесінің келу уақытымен сәйкес келмейді, себебі әр дестенің желі арқылы өту уақыты әртүрлі. Тіпті жөнелтуші оларды желі арқылы бірдей уақыт кезеңінен кейін жөнелтетін болса да, олар қабылдаушыға әртүрлі уақыт кезеңінен кейін келетін болады. Уақыт кідірісінің осылайша ауытқуы **джиттер (jitter)** деп аталады. Егер медиадеректер келген бетте бірден қалпына келтірілетін болса, онда тіпті кішкене джиттер күрделі кедергі келтіруі: сурет үзік-үзік, ал дыбыс түсініксіз болуы мүмкін.



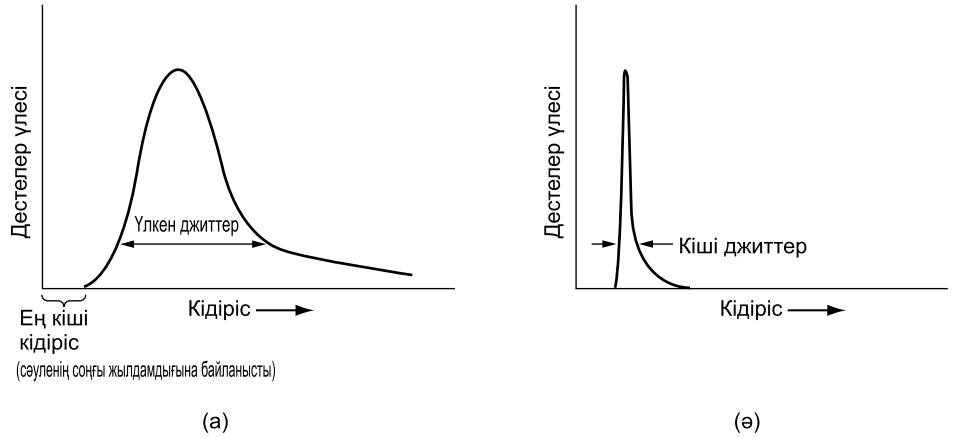
6.28-сурет. Дестелерді буферлеу арқылы шығыс ағынын тегістеу

Бұл мәселені шешу жолы – дестелерді қабылдауышта қалпына келтірер алдында **буферлеу.** *6.28-суретте* көрсетілген мысалда біз тағайындалған орын-

ға үлкен джиттермен келіп жатқан дестелер ағынын көреміз. Бірінші десте серверден *t=0c* уақытында жөнелтілген болатын, ал клиентке *t=1c* уақытында келіп жетті. Екінші десте кідіріп, 2 с кейін келеді. Келгеннен кейін дестелер клиент машинасындағы буферге орналастырылады.

t=10 с уақыт сәтінде қалпына келтіру басталады. Бұл кезде буферде 1-ден 6-ға дейінгі дестелер бар, сондықтан оларды буферден тең уақыт аралығында алып, біркелкі қалпына келтіруді қамтамасыз етуге болады. Жалпы жағдайда тең уақыт аралығын пайдалану міндетті емес, себебі RTP уақыт белгісінде қалпына келтіруді қашан бастау керек екендігі көрсетілген.

Өкінішке орай 8-дестенің кідіргені соншалық, ол қалпына келтіру уақытына үлгермей қалады. Бұл жағдайда екі нұсқа бар. Ойнатқыш сегізінші дестені тастап кетеді де, келесі дестелерді қалпына келтіруге кіріседі. Немесе ол сегізінші десте келгенше қалпына келтіруді тоқтатып, музыкада немесе фильмде жағымсыз үзіліс тудырады. Егер қосымша нақты уақыт режимінде жұмыс істейтін болса (мысалы, Интернет арқылы қоңырау шалу), онда ең дұрысы дестені тастап кету. Мұндай қосымшалар күту режимінде нашар жұмыс істейді. Ағындық мультимедиамен жұмыс істейтін қосымшаларда, ойнатқыш қалыпқа келтіруді уақытша тоқтата алады. Жағдайды жақсарту үшін қалпына келтіруді кейінірек бастап, көлемі үлкен буферді қолдануға болады. Ағындық аудио және бейне ойнатқышы барлық дестелер (жоғалғандарынан басқасы) дәл уақытында жету үшін, көбіне, көлемі 10 с шамасындағы буферді қолданады. Нақты уақытта жұмыс істейтін (мысалы, бейне конференция) және шапшаң жауапты талап ететін қосымшалар кішкене буферді пайдаланады.



6.29-сурет. Джиттер: а – үлкен; ә – кіші

Біркелкі қалпына келтіру барысында **қалпына келтіру кідірісі (playback point)** – қабылдаушы қалпына кетіруді бастамас бұрын, медиадеректерді күту уақыты басты рөл атқарады. Бұл параметр джиттерге тәуелді. Үлкен және кіші джиттері бар байланыс арасындағы айырмашылық *6.29-суретте* көрсетілген.

Орташа кідірістің қатты ерекшелігі жоқ, алайда, үлкен джиттер кезінде дестелердің 99%-ын қамтитын қалпына келтіру кідірісі қажет болуы мүмкін – бұл кіші джиттер жағдайынан әлдеқайда көп.

Қалпына келтіру кідірісін дұрыс таңдау үшін, қосымша RTP уақыт белгісі мен келу уақыты арасындағы айырманы есептей отырып, джиттерді өлшей алады. Бұл айырма (және қосымша қандай да бір тұрақты) әрбір жеке дестенің кідірісін құрайды. Алайда, бәсекелес трафик және маршруттың өзгеруі тәрізді кейбір факторлар уақыт кідірісінің өзгеруіне себеп болады. Қосымшалар бұны назарға алып, қажет болған кезде қалпына келтіру кідірісін өзгертуі керек. Бірақ дұрыс жүзеге асырмау салдарынан бұндай өзгерістер қатты кедергі тудырады. Аудио үшін мүмкін деген шешімдердің бірі – **дауыс сегменттері (talkspurts)** арасындағы қалпына келтіру кідірісін, демек, үзіліс уақытын түзету. Қысқа және сәл ұзын үзіліс арасындағы айырмашылық байқалмайды. Бұл мүмкін болу үшін, RTP қосымшаларға жаңа дауыс сегментінің басын *M* маркері көмегімен белгілеуге мүмкіндік береді.

Медиадеректер ұзақ ойнатылмайтын жағдайда да нақты уақыт режимінде жұмыс істейтін қосымшалар үшін күрделі қиындық тудырады. Егер онсыз да ең қысқа жол қолданылатын болса, тарату кідірісін азайтатын амал жоқ. Қалпына келтіру кідірісін ойнату басына кешіккен дестелер еншісін көбейту арқылы төмендетуге болады. Егер бұл мүмкін болмаса, онда соңғы нұсқа қалады: жоғары қызмет сапасын сұрау арқылы джиттерді, мысалы, басымдылық жеткізуі бар дифференциалды қызметті төмендету. Басқа сөзбен айтқанда, бұл үшін параметрлері жақсы желі қажет.

6.5. ИНТЕРНЕТ ТРАНСПОРТТЫҚ ХАТТАМАЛАРЫ: TCP

UDP қарапайым хаттама және өте маңызды аумақта қолданылады. Ең бірінші бұл клиент-серверлік әрекеттесулер және мультимедиа. Дегенмен, көптеген интернет-қосымшаларға сенімді, тізбекті тасымалдау қажет. UDP бұл талаптарды қанағаттандыра алмайды, сондықтан басқа хаттама керек. Осындай хаттама TCP деп аталады және ол Интернеттің жегін аты болып саналады. Төменде біз оны егжей-тегжейлі қарастырамыз.

6.5.1. TCP негізі

TCP (Transmission Control Protocol – тасымалдауды басқару хаттамасы) хаттамасы сенімсіз интержелі арқылы толассыз сенімді байттар ағынын қамтамасыз ету үшін арнайы құрастырылған болатын. Біріктірілген желінің жеке желіден айырмашылығы – оның әртүрлі аумақтарының топологиясы, өткізгіштік қабілеттілігі, кідіріс уақытының мәні, дестелер мөлшері және басқа да параметрлер бойынша қатты ерекшеленуі мүмкін. TCP құрастырылған

кезде негізгі назар хаттаманың біріктірілген желі қасиеттеріне бейімделуі және әртүрлі мәселелер туындаған кезде істен шығуға тұрақты болуына аударылған болатын.

TCP хаттамасы 1981 жылдың қыркүйек айында RFC 793 құжатында сипатталған болатын. Уақыт өте оның қателіктері және дәлсіздіктері түзетіліп, жетілдірілді. Қазіргі кезде RFC 793-тің толықтырмасы болып саналатын көптеген (осы хаттаманың танымалдығы жайлы жорамал беретін) басқа RFC: нақтылаулар мен түзетулер RFC 1122 сипатталған, жоғары өнімділік кеңейтілімі – RFC 1323, таңдаулы растаулар – RFC 2018, асыра жүктеуді басқару – RFC 2581, қызмет көрсету сапасы үшін тақырып өрістерін пайдалану – RFC 2873, қайта тасымалдауды жетілдірілген таймері – RFC 2988, асыра жүктелу жайлы анық хабарламалар – RFC 3168 бар. Бұл толық тізім емес, сондықтан барлық RFC жұмыс істеу ыңғайлы болу үшін арнайы көрсеткіш құрастырылды (әрине, тағы бір RFC-құжат түрінде) – RFC 4614.

TCP хаттамасын қолдайтын әр машинада TCP транспорттық ішкі жүйесі бар. TCP транспорттық ішкі жүйесі не кітапханалық процедура, не тұтынушы үдерісі, не (жиірек) жүйе ядросының бөлігі. Кез келген жағдайда транспорттық ішкі жүйе TCP-ағындарды және IP-деңгейі бар интерфейсін басқарады. TCP-ішкі жүйесі жергілікті үдерістерден тұтынушы деректер ағынын қабылдап, оларды 64 Кбайттан аспайтын (іс жүзінде бұл сан 1460 байт дерекке тең, бұл оны IP және TCP тақырыптары бар бір Ethernet кадрына орналастыруға мүмкіндік береді) бөліктерге бөліп, жеке IP-дейтаграмма түрінде жөнелтеді. TCP-деректері бар IP-дейтаграмма машинаға келген кезде олар бастапқы байттық ағынды қалпына келтіретін TCP-ішкі жүйеге беріледі. Қарапайымдылық үшін TCP транспорттық ішкі жүйесін (программалық жабдықтаманың бөлігі) немесе TCP хаттамасын (ережелер жиынтығы) біз кейде «TCP» дейтін боламыз. Мысалы, «Тұтынушы деректерді TCP-ға береді» деген, әрине, TCP транспорттық ішкі жүйесі дегенді білдіреді.

IP деңгейі дейтаграмманың дұрыс жеткізілуіне кепілдік бермейді және олардың жөнелтілу жылдамдығына шектеу қоймайды. Дұрыс жөнелту жылдамдығын (өткізгіштік қабілеттілікті тиімді пайдалану және асыра жүктелуді болдырмау үшін) таңдауды, мерзімі өткен күту кезеңдерін бақылауды және қажет болған жағдайда тағайындалған орынға жетпей қалған дейтаграмманы қайта жөнелтумен айналысуды TCP-дің орындауына тура келеді. Дейтаграммалар дұрыс ретпен келмейтін жағдайлар да болады. Осындай дейтаграммалардан мәлімдемені қалпына келтірумен де TCP айналысады. Сонымен, TCP хаттамасы көптеген қосымшалар армандайтын және IP хаттамасы ұсынбайтын, жақсы өнімділікті және сенімділікті қамтамасыз етуге бағытталған.

6.5.2. TCP қызмет көрсету моделі

TCP қызметтерінің негізінде жөнелтуші де, қабылдаушы да құрастыратын **сокеттер (ұя немесе соңғы нүкте)** жатыр. Олар *6.1.3-бөлімде* талқыланған

болатын. Әр сокеттің хост IP-адресінен және хостқа қатысты жергілікті болып саналатын **порттың** 16-биттік нөмірінен тұратын нөмірі (адресі) бар. TCP-де порт деп TSAP-адресті айтады. TCP қызметіне қол жеткізу үшін бір машина сокеті мен келесі машина сокетінің арасында анық байланыс орнатылуы тиіс. Сокеттердің базалық операциялары *6.2-кестеде* келтірілген.

Бір мезетте бір сокетті бірнеше байланыс үшін пайдалануға болады. Басқа сөзбен айтқанда, екі немесе одан да көп байланыстар бір сокетпен аяқталуы мүмкін. Байланыстар екі жақтағы сокеттер идентификаторымен ерекшеленеді (*socket1, socket2*). Виртуалды арналар нөмірі немесе идентификаторлар қолданылмайды.

Мәні 1024 төмен порттар нөмірі стандарт қызметтер үшін қорға алынған және тек артықшылығы бар тұтынушыларға қолжетімді (мысалы, root UNIX жүйеде). Олар **белгісіз порттар (well-known ports)** деп аталады. Мысалы, хост поштасын қашықтықтан жүктегісі келетін кез келген үдеріс хост-адресаттың 143 портымен байланысады, сөйтіп оның IMAP-демонына сүйене алады. Танымал порттар тізімі *www.iana.org* сайтында келтірілген. Қазіргі кезде мұндай порттардың саны 700-ден асады. Олардың кейбіреулері *6.4-кестеде* келтірілген.

6.4-кесте. Кебір қордағы порттар

Порт	**Хаттама**	**Қолданылуы**
20, 21	FTP	Файлдарды тасымалдау
22	SSH	Жүйеге қашықтықтан кіру, Telnet орнына
25	SMTP	Электронды пошта
80	HTTP	Жаһандық өрмек (World Wide Web)
110	POP-3	Электронды поштаға қашықтықтан қолжеткізу
143	IMAP	Электронды поштаға қашықтықтан қолжеткізу
443	HTTPS	Қатерден қорғаныс (SSL/TLS арқылы HTPP)
543	RTSP	Мультимедианы қалпына келтіру
631	IPP	Принтерді бірлесіп қолдану

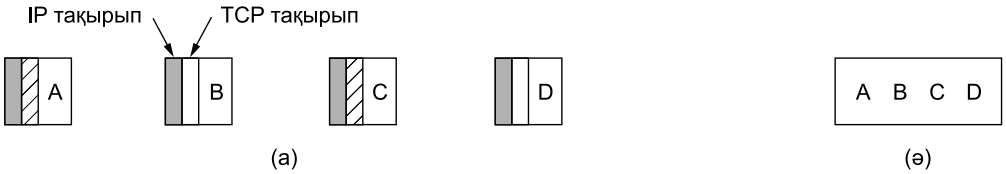
Нөмірлері 1024-тен 49151-ге дейінгі порттарды артықшылығы жоқ тұтынушылар үшін IANA арқылы тіркеуге болады, алайда, қосымшалар өз жеке порттарын таңдауы мүмкін (әдетте олар осыла жасайды да). Мысалы, BittTorrent қосымшасы файлдарға біррангілі бірлесіп қолжеткізу үшін (бейресми) 6881-6887 порттарын пайдаланады, сонымен бірге басқа порттарды да пайдалануы мүмкін.

Әрине, FTP-демонын 21-портпен жүктеу кезінде байланыстырып қоюға болар еді және сол кезде SSH-демонын 22-портпен байланыстыруға және т.с.с. Алайда, егер біз осылай істейтін болсақ, біз жадыны уақыттың көп бөлігінде босқа тұратын демондар жайлы ақпаратпен толтырар едік. Оның орнына UNIX-те әдетте, **inetd (Internet daemon)** деп аталатын бір демон қызметіне жүгінеді.

Ол бірнеше порттармен байланысып, бірінші кіріс байланысын күтеді. Ол байланыс орын алған кезде *inetd* жаңа үдеріс құрастырып, сұранысты өңдейтін ол үшін қолайлы демонды шақырады. Сөйтіп, тек бір *inetd* үнемі екпінді, қалғандары олар үшін жұмыс бар болған кезде шақырылады. *Inetd*-тің арнайы конфигурациялық файлы бар, одан порттың қызметі жайлы білуге болады. Бұл жүйелік әкімшілік жүйені ең жиі қолданылатын порттармен (мысалы, 80) тұрақты демондар баланыста болатындай етіп баптай алады дегенді білдіреді, ал қалғандарымен – *inetd*.

Барлық TCP-байланыстар толық дуплексті және екі нүктелі болып келеді. Толық дуплекс – трафик бір мезетте қарсы екі бетке де жүруі мүмкін дегенді білдіреді. Екі нүктелі байланыс – оның екі соңғы нүктесі бар дегенді білдіреді. TCP хаттамасы кең таратуды және көп адрестілікті қолдамайды.

TCP-байланыс – мәліметтер ағыны емес, байттық ағын. Мәліметтер арасында шекара сақталмайды. Мысалы, егер жөнелтуші үдеріс TCP-ағынға төрт 512-байттық деректер бөлігін жазса, бұл деректер қабылдаушы үдеріске төрт 512-байттық бөлік, екі 1024-байттық бөлік, бір 2048-байттық бөлік ретінде (*6.30-сурет*) немесе тағы басқаша жеткізілуі мүмкін. Қабылдаушы деректердің қалай жазылғандығын анықтайтын әдіс жоқ.



6.30-сурет. Жеке IP-дейтаграммалар тәрізді жөнелтілген төрт 512-байттық сегменттер (а); READ процедурасын бір рет шақыру арқылы қосымшаға жеткізілген 2048 байт дерек (ә)

UNIX жүйесінде файлдардың да осындай қасиеттері бар. Файлды оқитын программа оның: блокпен, байтпен немесе бірден тұтас жазылғандығын анықтай алмайды. UNIX жүйесіндегі файл жағдайындағыдай TCP-программа байттардың қызметі жайлы ешнәрсе білмейді де және қызықпайды да. Олар үшін байт – жай байт.

Қосымшадан деректерді алып, TCP хаттама оларды бірден жөнелтуі немесе өз қалауынша үлкен деректер бөлігін жөнелту үшін буферге орналастыруы мүмкін. Бірақ кейде қосымшаға деректердің бірден жөнелтілгендігі қажет. Айталық, интерактивті ойын тұтынушысы жаңартулар ағынын жөнелткісі келеді делік. Жаңартулардың басқа жаңартулар пайда болғанша буферде сақталмай бірден жөнелтілгендігі маңызды. Деректерді жөнелтуге мәжбүрлеу үшін TCP-де PUSH (итеру) жалаушасы қарастырылған. Бастапқыда бұл жалауша көмегімен қосымшалар TCP-ға дестені жөнелтуді кейінге шегере тұру керектігін хабарлайды деп ойластырылған болатын. Алайда, қосымшалар

деректер жөнелткенде мұндай жалауша орната алмайды. Оның орнына әртүрлі операциялық жүйелерде деректер тасымалын жылдамдататын арнайы параметрлер бар (мысалы, `TCP_NONDELAY` Windows және Linux-те).

Интернеттің ертедегі тәсілдері қызықтыратындар үшін біз TCP қызметінің қызықты ерекшелігі жайлы әңгімелейміз, ол хаттама құрамына әлі де кіреді, бірақ сирек қолданылады. Әңгіме **жедел деректер (urgent data)** жайлы болады. Деректердің бір бөлігінің басымдылығы жоғары болған жағдайда, демек ол бірден өңделуі тиіс. Мысалы, егер программамен интерактивті режимде әрекеттесіп отырған тұтынушы басталған қашықтықтағы үдерісті үзу үшін Ctrl-C пернесін басатын болса, жөнелтуші үдеріс деректердің шығыс ағынына басқарушы ақпарат орналастырып, оны TCP-қызметке `URGENT` (жедел) жалаушасымен бірге жөнелтуі мүмкін. Бұл жалауша TCP-ішкі жүйесін деректер жинақтауды тоқтатып, осы байланыс үшін бар деректерді желіге кідіріссіз жөнелтуге мәжбүрлейді. Жедел деректер тағайындалған орынға жеткен кезде, қабылдаушы қосымша тоқтап (UNIX терминологиясында «сигнал алып»), кіріс ағыннан деректер оқып, оның ішінен жедел деректі табуы тиіс. Қосымша жедел деректің қай жерде аяқталатынын білу үшін, оның соңы маркерленеді. Жедел деректердің басы маркерленбейді. Қосымша өзі анықтауы тиіс.

Бұл схема – басқаның барлығын қосымшаға қалдыратын дөрекі сигнал механизімі. Теорияда жедел деректерді қолдану орынды болып көрінгенімен, схема алғашқы пайда болған кезде жақсы жүзеге асырылды және сол себепті қолданысқа енді. Қазір ол жүзеге асыру қиындықтарынан қолдануға ұсынылмайды, сондықтан қосымшалардың өз сигналдар жүйесіне сүйенуіне тура келеді. Мүмкін келесі транспорттық хаттамаларда бұл ой жақсы жүзеге асырылар.

6.5.3. TCP хаттамасы

Бұл бөлімде TCP хаттамасы жалпылама түрде қарастырылады. Келесі бөлімде хаттама тақырыбының әр өрісін талқылаймыз.

Хаттаманың барлық құрылымын анықтайтын TCP қасиеті – TCP байланыстағы әр байттың өз 32-разрядты нөмірі бар. Интернет пайда болған алғашқы жылдары маршруттауыштар арасындағы бөлінген торап бойынша деректер тасымалдаудың базалық жылдамдығы 56 Кбит/с болатын. Ең үлкен жылдамдықпен үнемі деректер жіберіп отыратын хостқа реттік нөмірлер толық айналым жасау үшін аптадан көп уақыт қажет болар еді. Қазіргі жылдамдықта реттік нөмірлер тез аяқталады, бұл жайлы төменде айтылатын болады. Жеке 32-разрядты реттік нөмірлер бір бағытта сырғымалы терезе орнын және кері бағытта растауды көрсету үшін пайдаланылады. Бұл жайлы да төменде айтылатын болады.

Жөнелтуші және қабылдаушы TCP-ішкі жүйесі сегменттер түрінде деректермен алмасады. **TCP сегменті** бекітілген 20 байтты тақырыптан (және міндетті емес бөлік) тұрады, одан кейін деректер байты болуы мүмкін. Сегмент

мөлшері TCP программалық жабдықтамасымен анықталады. Ол бір сегментке бірнеше жазу операциясының нәтижесінде алынған деректерді біріктіруі немесе керісінше, бір жазу операциясының нәтижесін бірнеше сегментке таратуы мүмкін. Сегмент мөлшері екі шекпен шектелген. Біріншіден, әр сегмент TCP-тақырыпты қоса алғанда IP-дестенің 65 515-байттық пайдалы өрісіне сыюы тиіс. Екіншіден, әр арнада тасымалданушы блоктың ең үлкен мөлшері **(MTU, Maximum Transfer Unit)** бар. Жөнелтушіде және қабылдаушыда әр сегмент фрагменттерге бөлінбей, жеке дестеде жөнелтілуі және қабылдануы үшін ол MTU-ге сыюы тиіс. Іс жүзінде тасымалданушы блоктың ең үлкен мөлшері әдетте 1500 байтты құрайды (бұл Ethernet пайдалы жүктеме өрісі мөлшеріне сәйкес), сөйтіп сегмент мөлшерінің жоғарғы шегі анықталады.

Дегенмен, егер TCP-сегменті бар IP-деста MTU өте төмен жол арқылы өтетін болса фрагменттелуі мүмкін. Бұл жағдайда өнімділік төмендейді және басқа да мәселелер пайда болады (Kent және Mogul, 1987). Бұның орнына TCP жүзеге асырылуында RFC 1191 құжатында сипатталған тәсіл көмегімен **MTU маршрутты анықтау** орындалады, біз ол жайлы *5.5.5.-бөлімде* айтқан болатынбыз. Бұл тәсіл ICMP қателер жайлы мәлімдемесін пайдалана отырып, жолдың барлық арналары бойынша MTU-нің ең кіші мәнін есптейді. Осы мәннің негізінде TCP фрагменттеуді болдырмайтындай етіп сегмент мөлшерін таңдайды.

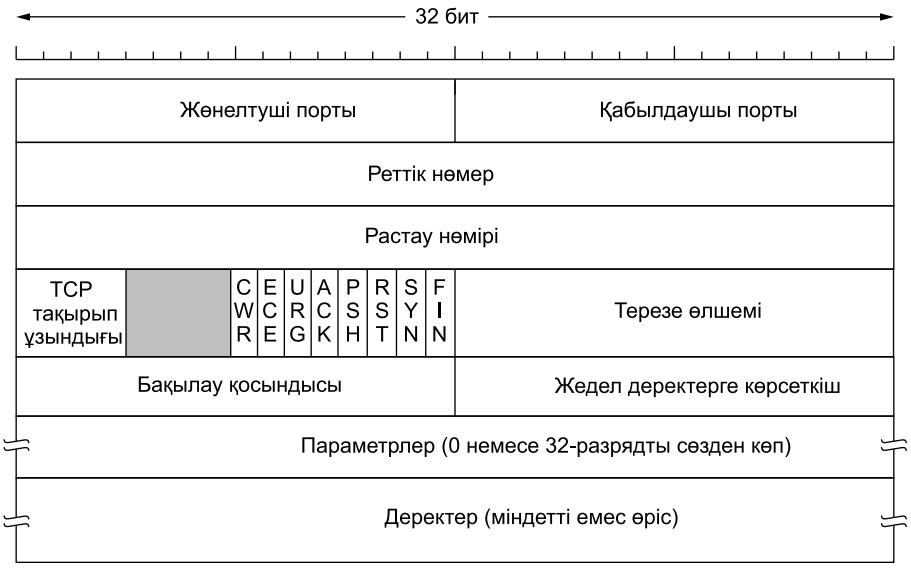
TCP-ішкі жүйелерінің пайдаланатын негізгі хаттамасы – терезе мөлшері динамикалық өзгеретін сырғымалы терезе хаттамасы. Сегментті жөнелту кезінде жөнелтуші таймерді іске қосады. Сегмент тағайындалған орынға келіп жеткен кезде қабылдаушы TCP-ішкі жүйесі сегментті келесі күтілулі сегмент реттік нөміріне тең растау нөмірімен (егер жөнелтетін деректер бар болса деректермен, жоқ болса дерексіз) және терезенің жаңа мөлшерімен кері жөнелтеді. Егер растауды күту уақыты аяқталса, жөнелтуші сегментті тағы бір рет жөнелтеді.

Бұл хаттама қарапайым болып көрінгенімен, оның егжей-тегжейлі қарастыратын бірнеше элементтері бар. Сегменттер ретсіз келуі мүмкін. Мысалы, 3072-ден 4095-ке дейінгі байттар келіп, бірақ оларға растау жөнелтілмеген, себебі 2048-ден 3071-ге дейінгі байттар әлі келмеген жағдай болуы мүмкін. Сонымен бірге сегменттер желіде ұзақ кідіріп, жөнелтушінің оларға растау күту уақыты аяқталып, ол сегментті қайта жөнелтуі де мүмкін. Қайта жөнелтілген сегментте басқа фрагменттердің басқа диапазоны болуы мүмкін, сондықтан дұрыс қабылданған байттар нөмірін анықтау үшін өте тиянақты әкімшілік керек. Дегенмен, әр байт ағында өз ығысуы бойынша бірегей анықталатын болғандықтан, бұл есепті жүзеге асыруға болады.

TCP бұл мәселелерді шеше алуы және тиімді шешуі тиіс. TCP-ағындардың өнімділін оңтайландыру үшін көп күш жұмсалды. Келесі бөлімде біз TCP хаттамасының әртүрлі жүзеге асырылуларында қолданылатын бірнеше алгоритмдерді қарастырамыз.

6.5.4. TCP-сегмент тақырыбы

TCP-сегмент тақырыбының құрылымы *6.31-суретте* көрсетілген. Әр сегмент бекітілген форматтағы 20-байттық тақырыптан басталады. Одан кейін қосымша өрістер (параметрлер) орналасады. Қосымша өрістерден кейін 65535-20-20=65495 байт деректер орналасады, мұндағы, алғашқы 20 байт – IP-тақырып, екінші 20 байт – TCP-тақырып. Сегменттерде деректер болмауы да мүмкін. Бұндай сегменттер растауды жөнелту үшін және басқарушы мәліметтер үшін жиі пайдаланылады.



6.31-сурет. TCP тақырыбы

TCP-тақырыптың әр өрісін қарастырайық. *Қабылдаушы порты* және *Жөнелтуші порты* өрістері жергілікті соңғы байланыс нүктелерінің идентификаторы болып келеді.TCP-порт IP-адреспен бірге соңғы нүктенің бірегей 48-битті идентификаторын құрайды. Жөнелтушіге және қабылдаушыға қатысты осындай екі идентификатор бірегей байланысты анықтайды. Байланыстың бұл идентификаторы бес ақпараттық құрама бөліктен тұратын болғандықтан **бес бөліктен тұратын кортеж (5 tuple)**: хаттама (TCP), жөнелтуші IP-адресі, жөнелтуші порты, қабылдаушы IP-адресі, қабылдаушы порты деп аталады.

Реттік нөмір және *Растау нөмірі* өрістері өздерінің әдеттегі функцияларын орындайды. Назар аударыңыздар: Растау нөмірі соңғы алынғанға емес, реті бойынша келесі күтілулі байтқа қатысты. Бұл **жинақтаушы растау (cumulative acknowledgement)**, себебі бір нөмір өзінде барлық қабылданған деректер жайлы ақпаратты біріктіреді. Оның пайдаланылу аймағы жоғалған деректер шегінен шықпайды. Екі өріс те 32-разрядты, себебі TCP-ағында деректің әр байты нөмірленеді.

TCP-тақырып ұзындығы өрісінде 32-байттық сөзбен бейнеленген TCP-тақырып мөлшері жазылады. Бұл ақпарат қажет, себебі *Параметрлер* өрісі, ал онымен бірге бүкіл тақырып айнымалы ұзындықты болуы мүмкін. Бұл тақырып ұзындығымен бірдей.

Бұдан кейін қолданылмайтын 4-биттік өріс орналасқан. Бұл биттердің 30 жыл бойында қолданылмау (бастапқыда 6-биттік өрістің тек 2-биті қолданылатын болды) TCP дизайнының қаншалықты жақсы құрастырылғанының куәсі. Бұлай болмағанда хаттама бұл биттерді TCP кемшіліктерін жою үшін қолданар еді.

Әрі қарай сегіз 1-биттік жалаушалар орналасқан. Қанығу жайлы анық ескерту (RFC 3168-ді қараңыз) қолданылатын жағдайда *CWR* және *ECE* желідегі асыра жүктелу жайлы хабтарлайды. TCP-қабылдағыш желінің асыра жүктелгендігі жайлы білген кезде ол *ECE* жалаушасының көмегімен TCP-жөнелтушіге жөнелту жылдамдығын төмендетуді сұрап, *ECN-эхо* сигналын жөнелтеді. TCP-жөнелтуші жөнелту жылдамдығын төмендеткеннен кейін ол бұны TCP-қабылдаушыға *CWR* жалаушасының көмегімен *Қанығу терезесі төмендетілді* сигналын жібереді. Бұдан кейін TCP-қабылдағыш *ECN-эхо* сигналын жөнелтуді тоқтатады. TCP асыра жүктелуін бақылаудағы асыра жүктелу жайлы анық ескеру рөлі жайлы біз *6.5.10-бөлімінде* қарастырамыз.

Ағымдағы реттік нөмірден *жедел деректер* орналасқан орынға дейінгі ығысу жазылған *Жедел деректерге көрсеткіш* өрісі қолданылған кезде *URG* битіне 1 жазылады. Сонымен, TCP хаттамасында үзіліс мәлімдемелері жүзеге асырылады. Жоғарыда айтқандай бұл тәсіл жөнелтушінің TCP-ді араластырмай, қабылдаушыға сигнал жөнелтуіне мүмкіндік береді, бірақ ол сирек қолданылады.

Егер *ACK* битінде 1 жазылса, онда *Растау нөмірі* өрісінде мағыналы деректер бар дегенді білдіреді. Көптеген дестелер үшін бұл осылай. Кері жағдайда бұл сегментте растау жоқ және *Растау нөмірі* өрісі есепке алынбайды.

PSH биті іс жүзінде PUSH-жалауша болып саналады. Бұл жалаушаның көмегімен жөнелтуші қабылдаушыны деста алына салысымен деректерді сақтамай бірден қосымша жөнелтуді сұрайды (Қабылдаушы үлкен тиімділікке қол жеткізу үшін буферлеумен айналысуы мүмкін).

RST биті хосттың істен шығу немесе басқа да бір жағдайлармен тұйыққа тірелген жағдайда байланыс қалып-күйін кенеттен үзу үшін қолданылады. Сонымен бірге ол дұрыс емес сегменттен немесе байланыс құрастырудан бас тарту үшін қолданылады. Егер *RST* биті орнатылған сегмент алсаңыз, бұл қандай да бір мәселенің туындағанын білдіреді.

SYN биті байланыс орнату үшін пайдаланылады. Байланысқа сұраныстың *SYN=1*, ал *ACK=0* болса, бұл растау өрісінің қолданылмайтынын білдіреді. Бірақ сұранысқа жауапты растау болуы мүмкін, сондықтан ондағы бұл биттердің мәндері бірдей: *SYN=1, ACK=1*. Сөйтіп, *SYN* биті `CONNECTION REQUEST` сегментінде, `CONNECTION ACCEPTED` сегментінде белгілеу үшін, ал *ACK* биті оларды бір-бірінен ажырату үшін қолданылады.

FIN биті байланысты үзу үшін пайдаланылады. Ол жөнелтушіде басқа *жөнелтілетін* деректердің жоқ екенін көрсетеді. Алайда, байланыс жабылғанымен үдеріс анықталмаған уақыт аралығында деректер *қабылдай* беруі мүмкін. *FIN* және *SYN* биттері бар сегменттерінің дұрыс ретпен орындалуына кепілдік беретін реттік нөмірлері бар.

TCP хаттамасында ағынды басқару айнымалы көлемдегі сырғымалы терезе көмегімен жүзеге асырылады. *Терезе көлемі* өрісі растау алынған байттан кейін неше байт жөнелтуге болатындығын көрсетеді. *Терезе көлемі* өрісінің мәні нөлге тең болуы мүмкін, бұл *Растау нөмірі – 1*-ге дейінгі байттардың қабылданғанын, бірақ қабылдаушы оны өңдеп үлгермегендіктен басқа байттарды әзірше қабылдай алмайтындығын білдіреді. Әрі қарай жөнелтуге рұқсатты *Растау нөмірі* өрісінің мәні дәл осындай және *Терезе көлемі* өрісінің мәні нөл емес сегмент жөнелту арқылы алуға болады.

3-тарауда біз кадр қабылданғанын растау, тасымалдауды жалғастыруға рұқсатпен байланысты хаттамаларды қарастырған болатынбыз. Бұл байланыс хаттамалардағы сырғымалы терезе көлемінің қатаң бекітілгендігінің салдары болатын. TCP-де растау деректер тасымалдау рұқсатынан бөлектенген. Іс жүзінде қабылдаушы: «Мен *k*-ға дейінгі байттарды қабылдадым, алайда, мен қазір қабылдауды жалғастыра беремін», – деуі мүмкін. Мұндай бөлу (*айнымалы көлемдегі* сырғымалы терезеде айтылатын) хаттамаға қосымша иілгіштік береді. Төменде біз бұны егжей-тегжейлі талқылаймыз.

Бақылау қосындысы өрісі сенімділікті жоғарылату үшін қызмет етеді. UDP-дағыдай мұнда тақырып, деректер және жалған тақырып бақылау қосындысы жазылады. Бірақ UDP-дан айырмашылығы жалған тақырып өрісінде TCP хаттамасының нөмірі (6) жазылады, ал бақылау қосындысы міндетті болып саналады. Толық ақпарат алу үшін *6.4.1-бөлімін* қараңыз.

Параметрлер өрісі стандартты тақырып қамтымайтын қосымша мүмкіндіктерді ұсынады. Параметрлер өте көп және олардың кейбіреулері кеңінен қолданылады. Олардың ұзындықтары әртүрлі, бірақ барлығы 32 битке еселі (артық орындар нөлмен толтырылады) және TCP тақырыбының ең үлкен мөлшері болып саналатын, 40 байт деген белгіге дейін жетуі мүмкін. Байланыс орнатылған кезде қосымша өрістер қарсы бетпен келісу үшін немесе жай байланыс сипаттамаларын мәлімдеу үшін пайдаланылуы мүмкін. Байланыстың барлық өмір уақытында сақталатын өрістер бар. Барлық қосымша өрістердің форматы: Тип–Ұзындық–Мән.

Осындай өрістердің бірінің көмегімен әр хост өзі қабылдай алатын **сегменттің ең үлкен ұзындығын (MSS, Maximum Segment Size)** көрсете алады. Пайдаланылатын сегменттердің ұзындығы неғұрлым үлкен болса, соғұрлым тиімділік жоғары, себебі бұл жағдайда 20-байттық тақырып түріндегі үстеме шығындардың салыстырмалы салмағы төмендейді, алайда, барлық хосттардың үлкен сегмент қабылдауға мүмкіндігі жоқ. Хосттар байланыс орнатқан кезде сегменттің ең үлкен көлемі жайлы бір-біріне хабарлай алады. Үнсіз келісім бойынша көлем 536 байтқа тең. Барлық хосттар 536+20=556 бай

көлеміндегі TCP-сегменті қабылдауға міндетті. Сегменттің ең үлкен көлемі әр бағыт үшін тағайындалуы мүмкін.

Тасымалдау жылдамдығы жоғары және/немесе терезе кідірісі үлкен торап үшін 16-биттік өріске сәйкес 64 Кбайт көлемдегі терезе тым кіші болып келеді. Осылайша, OC-12 торабы (шамамен, 600 Мбит/с) толық терезе торапқа 1 мс кем уақытта берілуі мүмкін. Егер сигналдың екі бағытта таралу уақытының мәні 50 мс құраса (бұл трансконтиненталдық оптыталшықты кабельге сәйкес), жөнелтуші уақыттың 98%-ын растауды күтуге жұмсайды. Үлкен көлемдегі терезе тиімділікті арттырар еді. **Терезе масштабы** параметрі екі хостқа байланыс орнату кезінде терезе масштабы жайлы келісуге мүмкіндік береді. Бұл сан екі жаққа да *Терезе көлемі* өрісін 14 разрядқа дейін солға жылжытып, терезе көлемін 230 байтқа (1 Гбайт) кеңейтуге мүмкіндік береді. TCP хаттамасының көптеген іске асырылуы осы мүмкіндікті қолдайды.

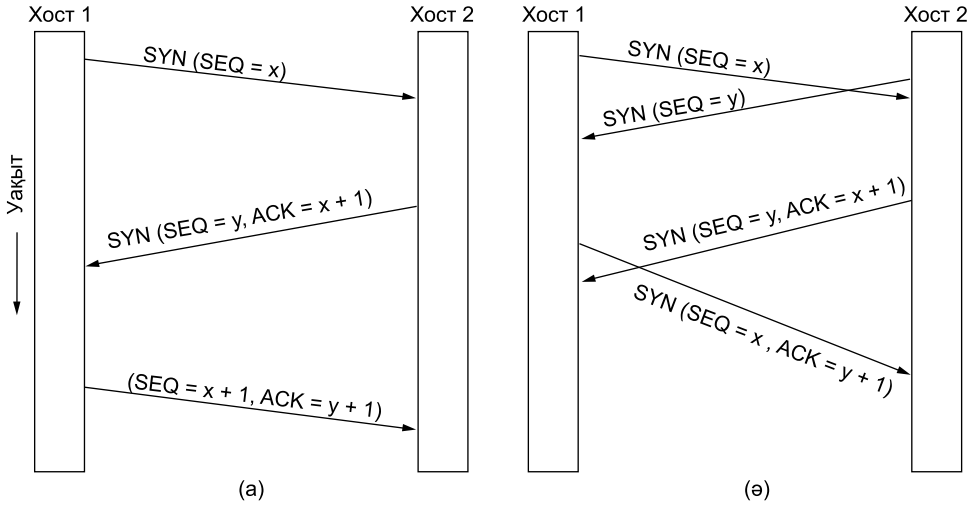
Жөнелтушіден қабылдаушыға және кері тасымалданатын **уақыт белгісі** үшін аттас параметр бар. Егер байланысты баптау кезінде бұл параметрді пайдалану қабылданса, онда ол әр дестеге қосылады. Ол айнала кідіріс жайлы деректер алуға көмектеседі, ал бұл өз кезегінде дестелердің жоғалғандығын анықтауға көмектеседі. Сонымен бірге ол 32-биттік реттік нөмірдің логикалық кеңейтілімі ретінде пайдаланылады. Жылдам байланыс кезінде реттік нөмірлер толық айналымды жылдам өтуі мүмкін, нәтижесінде жаңа және ескі деректерді ажырату мүмкін болмайды. **Реттік нөмірлердің қайта қолданылуын детектрлеу (PAWS, Protection Against Wrapped Sequence numbers)** бұл мәселені болдырмас үшін уақыт белгісі ескі сегменттерді жояды. Соңында, **іріктеп растау (SACK, Selective ACKnowledgement)** көмегімен қабылдаушы жөнелтушіге жеткізілген дестелердің реттік нөмірлер диапазоны жайлы хабарлай алады. Бұл параметр *Растау нөмірі* өрісіне қосымша болып саналады және деста жоғалғаннан кейін деректер жеткізілген (мүмкін көшірме ретінде) жағдайда пайдаланылады. Жаңа деректер *Растау нөмірі* өрісінің тақырып өрісінде бейнеленбеген, себебі онда тек реті бойынша келесі күтілетін байт орналасады. Егер іріктеп растау қолданылатын болса, онда жөнелтуші қабылдаушыда қандай деректердің бар екенін үнемі білетін болады және қайта жөнелтуді тек қажет болған жағдайда ғана орындайды. Іріктеп растау RFC 2108 және RFC 2883 құжаттарында сипатталған. Соңғы кезде бұл схема жиі пайдаланылады. Оның асыра жүктелуді бақылау кезінде қолданылуы жайлы біз *6.5.10-бөлімде* айтамыз.

6.5.5. TCP-байланысты орнату

Хаттамада TCP байланыс «үштік қол алысу» көмегімен орнатылады. Байланыс орнату үшін бір жақ (мысалы, сервер) `LISTEN` және `ACCEPT` базалық операцияларын орындап немесе нақты ақпарат көзін көрсетіп немесе көрсетпей кіріс байланысты пассивті түрде күтеді.

Келесі жақ (мысалы, клиент) байланыс орнатқысы келетін IP-адрес және портты, TCP-сегменттің ең үлкен көлемін және қалауы бойынша тұтынушының кейбір параметрлерін (мысалы, пароль) көрсетіп, `CONNECT` операциясын орындайды. `CONNECT` операциясы орнатылған *SYN* биті, алынып тасталған ACK биті бар TCP-сегментті жөнелтіп, жауап күтеді.

Осы сегмент тағайындалған орынға келген кезде TCP-ішкі жүйесі қандай да бір үдерістің параметр ретінде *Қабылдаушы порты* өрісінде жазылған портты көрсетіп, `LISTEN` операциясын орындағандығын тексереді. Егер бұндай үдеріс жоқ болса, онда ол *RST* биті орнатылған сегментті жөнелтіп, байланыстан бас тартады.



6.32-сурет. Қалыпты жағдайда TCP-байланыс орнату (а);
бір мезгілде екі жақпен де байланыс орнату (ә)

Егер қандайда бір үдеріс көрсетілген портты тыңдап отырса, онда кіріс TCP-сегменті сол үдеріске беріледі. Соңғысы оны қабылдау немесе бас тартуы мүмкін. Егер үдеріс байланысты қабылдаса, онда ол жауап ретінде растау жөнелтеді. Қалыпты жағдайда жөнелтілетін TCP-сегменттер тізбегі *6.32 а-суретінде* көрсетілген. *SYN* биті орнатылған сегменттің реттік нөмірлер кеңістігінің 1 байт кеңістігін алатынына назар аударыңыздар. Бұл олардың растауындағы бір мағыналықты болдырмайды.

Егер екі хост бірмезгілде бір-бірімен байланыс орнатқысы келсе, онда бұл жағдайдағы оқиғалар тізбегі *6.32 ә-суретіне* сәйкес болады. Нәтижесінде екі байланыс емес, тек бір байланыс орнатылады, себебі соңғы екі нүктені байланыс анықтайды. Демек, егер екі байланыс та өздерін (x, y) жұбының көмегімен сәйкестендірмек болса, (x, y) үшін бір ғана кестелік жазба жазылады.

Естеріңізде болса, әр жеке хост таңдап алған реттік нөмірлердің бастапқы мәні тұрақты болып қалмай (мысалы, нөл) баяу өзгеруі тиіс. Біз *6.2.2-бөлімінде*

айтқандай бұл ереже дестелердің кідіріп қалған көшірмелерінен қорғайды. Бастапқыда бұл схема өз қалып-күйін әр 4 мс сайын өзгертіп отыратын таймер көмегімен жүзеге асырылған болатын.

Алайда, «үштік қол алысу» схемасының жүзеге асырылу мәселесі – тыңдаушы үдеріс өз реттік нөмірін өзінің жеке *SYN*-сегментін жөнелткенше есте сақтауы тиіс. Бұл қаскөй жөнелтуші *SYN*-сегменттер ағынын жөнелтіп және байланысты үзбей хост ресурстарын оқшаулауы мүмкін. Мұндай шабуылдар **SYN-сегменттермен басып тастау (SYN flood)** деп аталады. XX ғасырдың 1990 жылдары осының салдарынан көптеген веб-серверлер сал болып қалған.

Мұндай шабуылдан қорғанудың бір түрі **SYN coolkies** деп аталады. Реттік нөмірді есте сақтаудың орнына хост нөмірдің криптографиялық мәнін генерациялап, оны шығыс сегментке жазады да ұмытады. Егер «үштік қол алысу» аяқталса, бұл реттік нөмір (1-ге өскен) хостқа қайтып келеді. Хост, кіріс деректер белгілі болған жағдайда (бұл IP-адрес және басқа хост порты, сонымен бірге қандайда бір құпия мән болуы мүмкін) сол криптографиялық функцияның мәнін есептеп, дұрыс реттік нөмірді қайта генерациялауы мүмкін. Осы процедура көмегімен хост расталған реттік нөмірдің дұрыстығын ол нөмірді жеке есте сақтамай-ақ тексере алады. Бұл тәсілдің бір ерекшелігі, ол TCP қосымша өрістермен (параметрлермен) жұмыс істейді. Сондықтан SYN coolkies тек *SYN*-сегменттермен басылып қалған жағдайда ғана қолдануға болады. Жалпы бұл өте қызық әдіс. Ол жайлы толық ақпаратты RFC 4987 және Lemon (2002) оқуға болады.

6.5.6. TCP байланысты үзу

TCP-байланыс толық дуплексті болғанымен, байланыс үзілуінің қалай жүзеге асырылатынын білу үшін оны симплексті байланыстар пары деп есептеген дұрыс. Әр симплексті байланыс өз серіктесіне тәуелсіз үзіледі. Байланысты үзу үшін екі жақтың кез келгені *FIN* биті орнатылған TCP-сегмент жөнелте алады, бұл оның басқа жөнелтетін деректері жоқ екенін білдіреді. Бұл TCP-сегмент растау алған кезде тасымалдаудың осы бағыты жабылады. Дегенмен, деректер қарсы бағытта белгісіз уақыт аралығында жөнелтіле береді. Байланыс екі бағытта жабылған кезде үзіледі. Әдетте, байланысты үзу үшін: әр бағытта бір-бірден *FIN* битімен және *SYN* битімен төрт TCP-сегмент қажет. Бірінші АСК биті және екінші *FIN* биті бір TCP-сегментте орналасуы мүмкін. Бұл сегменттер санын үшке дейін қысқартады.

Телефонмен әңгімелесудегідей, екі сөйлесуші де бірмезгілде қоштасып трубканы ілгендей, TCP-байланыстың екі жағыда *FIN* битін бір мезгілде жөнелтуі мүмкін. Олардың екеуі де әдеттегі растауды алады және байланыс жабылады. Іс жүзінде бір мезгілде және тізбекпен үзілудің айырмашылығы жоқ.

Екі армия мәселесін болдырмас үшін (*6.2.3-бөлімді қараңыз*) таймер қолданылады. Егер соңғы *FIN* битіне жауап десте өмір уақытының екі ең үлкен

кезеңі ішінде келмесе, *FIN*-сегментті жөнелтуші байланысты үзеді. Екінші жақ оған ешкімнің жауап бермегенін сезіп, ол да байланысты үзеді. Бұл шешім мінсіз болмағанымен, бірақ мінсіздіктің мүмкін емес екенін есепке ала отырып, барды пайдалануға тура келеді. Іс жүзінде мәселе сирек туындайды.

6.5.7. TCP-байланыстарды басқару моделі

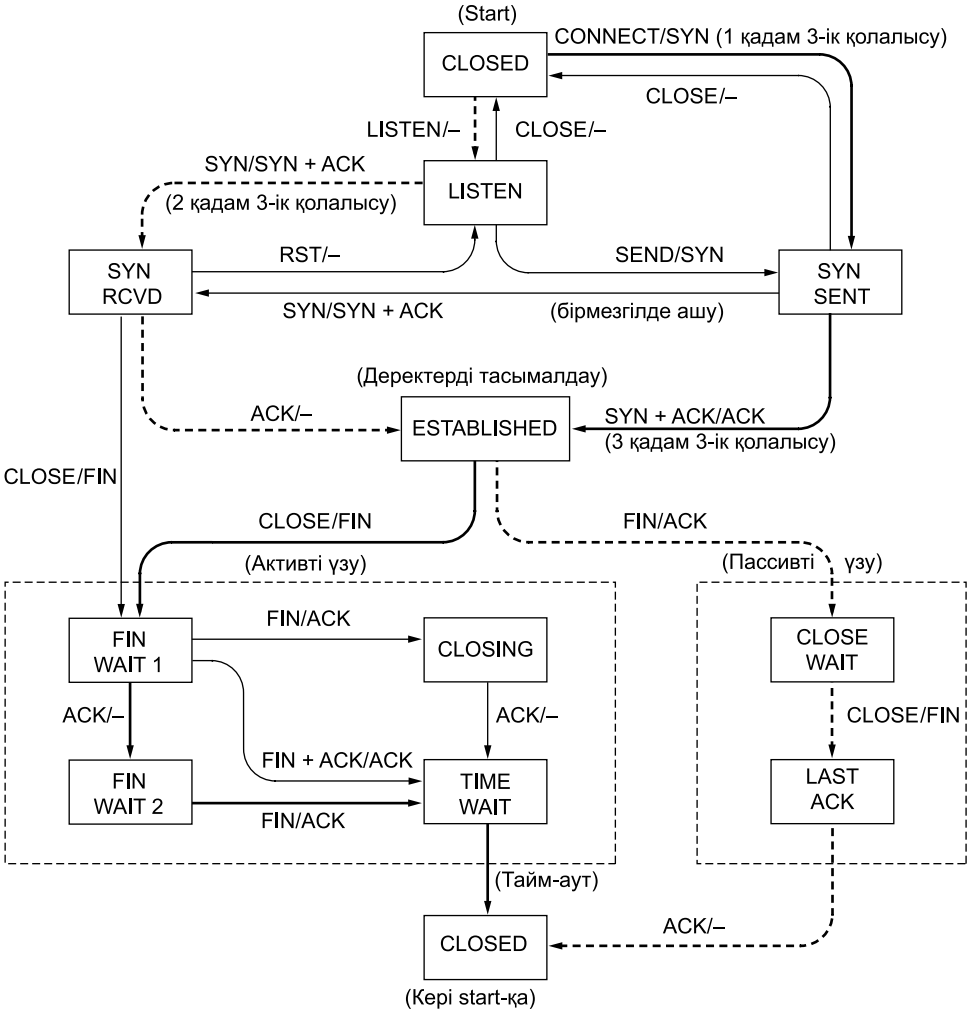
Байланысты орнату және үзуге қажет сатылары, 11 қалып-күйі 6.5-кестеде келтірілген соңғы автомат түрінде бейнелеуге болады. Бұл қалып-күйлердің әрқайсында оқиғаға рұқсат беру, не шек қою орын алуы мүмкін. Қандайда бір рұқсатқа жауап қандайда бір іс-әрекет орындалады. Шектеу қойылатын оқиға орын алған кезде қателік жайлы мәлімдеме беріледі.

6.5-кесте. TCP-байланысты басқаратын соңғы автомат қалып-күйі

Қалып-күй	Сипаттамасы
CLOSED	Жабық. Байланыс активті және орнату үдерісінде емес
LISTEN	Күту. Сервер кіріс сұранысты күтуде
SYN RCVD	Байланыс сұранысы келді. Растауды күту
SYN SENT	Байланыс сұранысы жөнелтілді. Қосымша байланысты аша бастады
ESTABLISHED	Орнатылды. Деректер тасымалдаудың қалыпты жағдайы
FIN WAIT 1	Қосымша жөнелтетін деректер жоқ екенін хабарлады
FIN WAIT 2	Екінші жақ байланысты үзуге дайын
TIME WAIT	Күту, желіде барлық дестелер жоғалғанша
CLOSING	Екі жақ бір мезгілде байланысты жабуға талпынды
CLOSE WAIT	Екінші жақ байланысты үзуді бастады
LAST ACK	Күту, желіде барлық дестелер жоғалғанша

Әр байланыс *CLOSED* қалып-күйінде басталады. Ол бұл қалып-күйден байланыс ашудың активті (`CONNECT`) немесе пассивті (`LISTEN`) талпынысын орындап шыға алады. Егер қарсы жақ кері әрекет орындаса, байланыс орнатылады және *ESTABLISHED* қалып-күйінде көшеді. Байланысты үзуді екі жақтың кез келгені бастауы мүмкін. Үзу үдерісі аяқталғаннан кейін байланыс *CLOSED* қалып-күйінде қайтып оралады. Соңғы автомат *6.33-суретінде* бейнеленген. Пассивті сервермен байланыспақ болған клиенттің әдеттегі жағдайы қалың сызықпен – біркелкі сызық клиент үшін және үзік сызық сервер үшін көрсетілген. Жіңішке сызық оқиғалардың әдеттегіден өзгеше тізбегін білдіреді. *6.33-суреттегі* әр сызық оқиға/әрекет жұбымен белгіленген. Оқиға – тұтынушының жүйелік процедураға (`CONNECT, LISTEN, SEND` немесе `CLOSE`) немесе сегменттің (*SYN, FIN, ACK* немесе *RST*) келуі немесе десте өмір уақытының екі кезеңіне тең күту уақытының аяқталуы болуы мүмкін. Әрекет –

басқарушы сегментті (*SYN, FIN* немесе *RST*) жөнелту болуы мүмкін. Ешқандай әрекетті орындамау сызықпен көрсетілген. Жақша ішінде түсініктеме берілген.



6.33-сурет. TCP-байланыс соңғы автоматы. Қалың біркелкі сызық клиенттің қалыпты жолын көрсетеді. Қалың үзік сызықпен сервердің қалыпты жолы көрсетілген. Жіңішке сызық әдеттегіден өзгеше жағдайды көрсетеді

Диаграмманы бастапқыда клиент жолымен (біркелкі қалың сызық) жүрген, ал сонан кейін сервер жолымен (қалың үзік сызық) жағдайда түсіну жеңіл. Клиент машинасындағы қосымша `CONNECT` операциясын шақырған кезде жергілікті TCP-ішкі жүйесі байланыс жазбасын құрастырып, оның қалып-күйін *SYN SENT* деп белгілеп, *SYN*-сегментін жөнелтеді. Бірнеше қосымшалар бір мезгілде бірнеше байланыс ашуы мүмкін, сондықтан әр жеке байланыстың байланыс жазбасында сақталатын өз қалып-күйі бар. *SYN+ACK*

сегменттері келген кезде TCP-ішкі жүйесі «үштік қол алысудың» соңғы ACK-сегментін жөнелтеді де *ESTABLISHED* қалып-күйіне ауысады. Бұл қалып-күйде деректерді қайта жөнелтуге және тасымалдауға болады. Деректер тасымалдау аяқталғаннан кейін клиент `CLOSE` операциясын орындайды. Нәтижесінде серверге *FIN*-сегменті (үзік сызықты тікбұрыш, пассивті үзу түрінде белгіленген) келеді. Енді сервер `CLOSE` операциясын орындайды, ал *FIN*-сегменті клиентке жөнелтіледі. Клиенттен растау келген кезде сервер байланысты үзіп, ол жайлы жазбаны өшіреді.

6.5.8. TCP сырғымалы терезесі

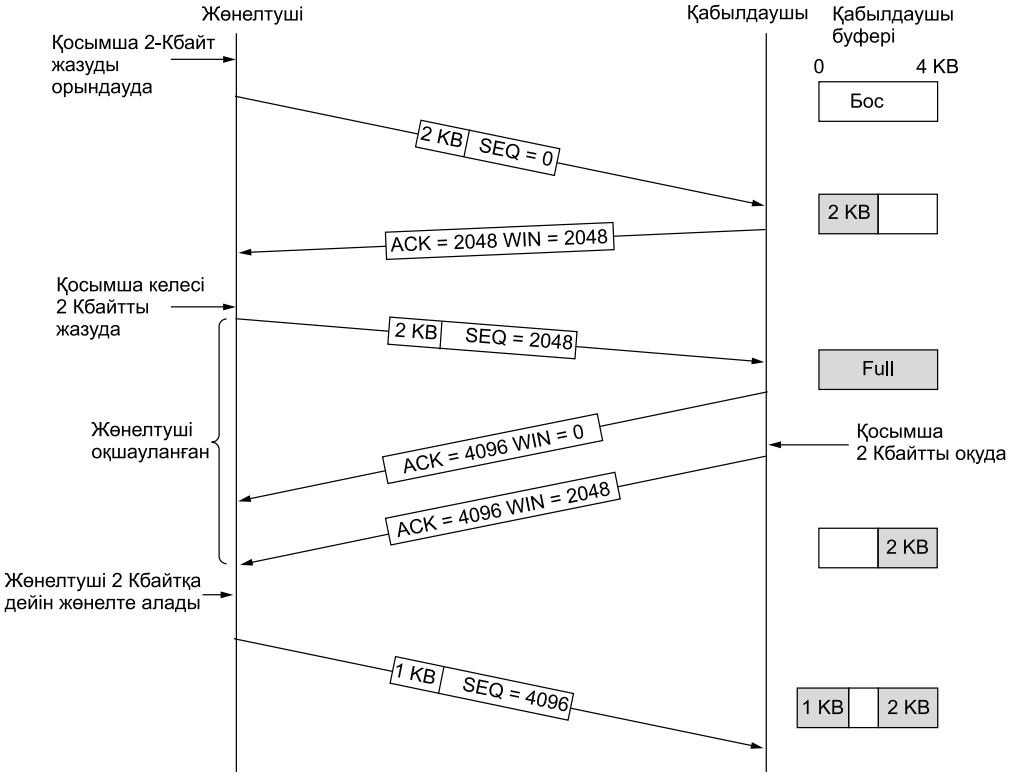
Жоғарыда айтылғандай, терезені басқару TCP-де сегменттердің дұрыс жеткізілуін растау және қабылдауышта буфер бөлу мәселесін шешеді. Мысалы, қабылдауышта *6.34-суретте* көрсетілгендей 4096-байттық буфер бар делік. Егер жөнелтуші 2048-байттық сегмент жөнелтіп, ол қабылдауышқа сәтті жеткізілсе, қабылдауыш оның қабылдағандығын растайды. Бұл жағдайда қабылдауышта бар жоғы 2048-байт бос буферлік кеңістік қалады (қосымша буферден деректердің қандайда бір бөлігін алғанша), ол бұл жайлы жөнелтушіге терезенің көлемін (2048) және келесі күтілетін байт нөмірін көрсетіп хабарлайды.

Барлық «1К» белгілеуін «1 Кбайт», «2К» белгілеуін «2 Кбайт», «4К» белгілеуін «4 Кбайт», ал «3К» белгілеуін «2 Кбайт» деп түзету керек (түп нұсқаға сәйкес болу үшін).

Бұдан кейін жөнелтуші тағы 2048 байт жөнелтеді. Олардың да қабылданғаны расталады, бірақ терезе көлемі 0-ге тең деп жарияланады. Жөнелтуші тасымалдауды қабылдаушы хост буферде орын босатып, терезе көлемін үлкейткенше тоқтату керек.

Терезе көлемі нөл болған жағдайда – жөнелтуші тек екі жағдайда ғана сегменттер жөнелте алады. Біріншіден, тек жедел деректерді жөнелтуге рұқсат етіледі, мысалы тұтынушы қашықтықтағы машинада орындалып жатқан үдерісті жою үшін. Екіншіден, жөнелтуші қабылдаушыдан терезе көлемі және келесі күтілетін байт жайлы ақпаратты қайталауын сұрап 1-байттық сегмент жөнелте алады. Мұндай десте **байқау сегменті (window probe)** деп аталады. TCP стандарты терезе көлемі жайлы хабарлама жоғалған кезде тұйыққа тірелу жағдайын болдырмас үшін, бұл мүмкіндікті анық түрде қарастырады.

Жөнелтушілер қосымшадан келген деректерді бірден жөнелтуге міндетті емес. Сонымен бірге ешкім қабылдаушыдан растауды лезде жөнелтуді талап етпейді. Мысалы, *6.34-суретте* TCP-ішкі жүйесі қосымшадан алғашқы 2 Кбайт деректерді алғаннан кейін, терезе көлемі 4 Кбайт екенін біле отырып, бірден 4 Кбайт пайдалы жүктемесі бар сегмент жөнелту үшін, қабылданған деректерді келесі 2 Кбайт деректер келгенше буферде сақтап қойса дұрыс болар еді. Бұл әрекет ету еркіндігін өнімділікті жақсарту үшін пайдалануға болады.



6.34-сурет. TCP-дегі терезені басқару

Перненің әр басылғанына әрекет ететін қашықтықтағы терминалмен байланысты (мысалы, telnet немесе SSH) қарастырайық. Ең нашар жағдайда символ жөнелтуші TCP-ішкі жүйесіне келген кезде ол 21-байттық TCP-сегмент құрастырады да, оны IP-деңгейге жібереді. IP-деңгей өз кезегінде 41-байттық IP-дейтаграмма жөнелтеді. Қабылдаушы бетте TCP-ішкі жүйесі 40-байттық растаумен (20 байт TCP-тақырып және 20 байт IP-тақырып) жауап қайтарады. Сонан кейін, қашықтықтағы терминал бұл байтты буферден оқыған кезде, TCP-ішкі жүйесі терезені 1 байтқа оңға жылжытып, буфер көлемі жайлы жаңартылған ақпаратты жөнелтеді. Бұл дестенің көлемі де 40 байтты құрайды. Соңында, қашықтықтағы терминал бұл символды өңдегеннен кейін 41-байттық дестемен тасымалданатын жаңғырықты кері жөнелтеді. Сөйтіп, пернетақтадан енгізілген әр символ үшін жалпы көлемі 162 байтты құрайтын төрт деста жөнелтіледі. Торап өткізгіштік қабілеттілік тапшылығы жағдайында бұндай жұмыс тәсілі жарамсыз.

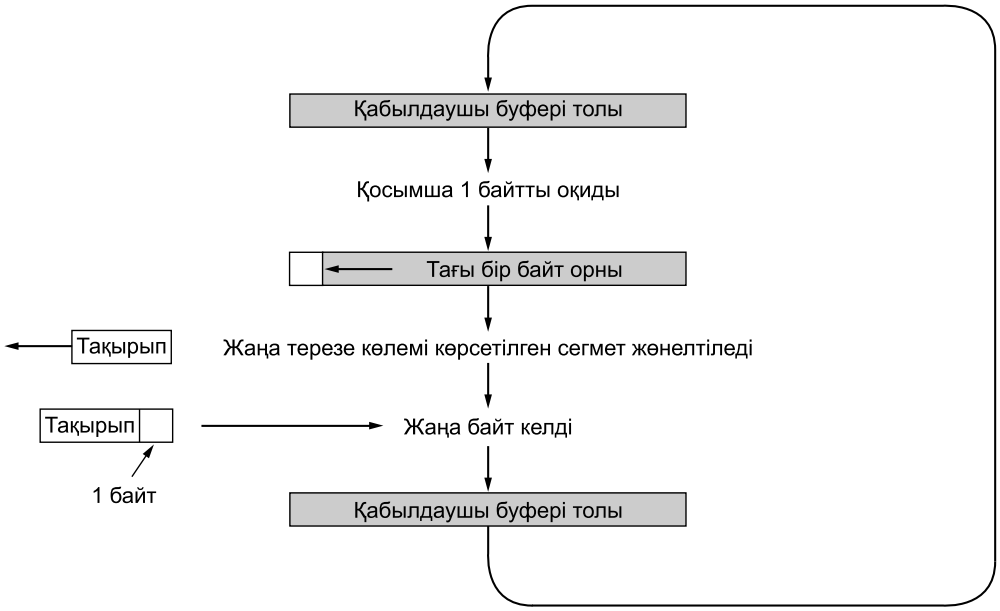
Жағдайды жақсарту үшін TCP-дің көптеген іске асырылуы **кейінге қалдырылған растауды (delayed acknowledgements)** пайдаланады. Бұл тәсілдің мағынасы – растау және терезе көлемі жаңартуларын бір дестеге қосып жіберуге болатындай қосымша деректер алуға үміттеніп, 500мс уақытқа дейін кідірту.

Егер терминал жаңғырықты 500 мс ішінде жіберіп үлгерсе, қашықтықтағы бетке тек бір 41-байттық деста жөнелту керек болады, сөйтіп, желі жүктемесі екі есе төмендейді.

Кейінге қалдырылған растау желі жүктемесін төмендеткенімен, көптеген кіші дестелер (мысалы, 1 байт нақты деректері бар 41-байттық дестелер) тасымалдайтын жөнелтушінің желіні пайдалану тиімділігі төмен болып қала береді. Тиімділікті жоғарылатуға мүмкіндік беретін тәсіл **Нагль алгоритмі (Nagle's algorithm)** деген атпен белгілі (Nagle, 1984). Нагль ұсынысы өте қарапайым: егер деректер жөнелтушіге кішкене бөліктермен келетін болса, жөнелтуші бірінші фрагменті жөнелтіп, қалғандарын бірінші фрагментке растау келгенше буферде сақтайды. Бұдан кейін буфердегі барлық деректерді бір TCP-сегмент түрінде жөнелтіп, қабылданғандық жайлы растау келгенше буферлеуді жалғастыруға болады. Сонымен, уақыттың әр сәтінде тек бір кішкене деста жөнелтуге болады. Егер деста байланыстың екі басына жетіп үлгеретін уақыт аралығында қосымша деректердің көптеген кіші бөліктерін жөнелтсе, Нагль алгоритмі осындай бірнеше бөліктерді бір сегментке біріктіреді, сөйтіп желі жүктемесі айтарлықтай төмендейді. Бұдан басқа, осы алгоритмге сәйкес, егер буфердегі деректер көлемі сегменттің ең үлкен көлемінен көп болса, жаңа деста бірден жөнелтілуі тиіс.

Нагль алгоритмі TCP хаттамаларының әртүрлі жүзеге асыруларында кеңінен қолданылады, алайда, кейде алгоритмді өшіріп тастау жақсы болатын жағдайлар да кездеседі. Нақты айтқанда, Интернет арқылы жүретін интерактивті ойындарға жаңартулары бар кішкене дестелердің жылдам ағыны қажет. Егер бұл деректерді дестелік тасымалдау үшін буферлейтін болсақ, ойын дұрыс жүрмейді және тұтынушының көңілі толмайды. Мәселенің тағы бір жіңішке жері, растауды кідірту үшін Нагль алгоритмін пайдалану уақытша тұйыққа тірелуге әкелуі мүмкін: тұтынушы растауды тіркеуге болатын деректерді күтеді, ал жөнелтуші растауды күтеді, онсыз жаңа деректерді жөнелтуге болмайды. Нақты айтқанда, осының салдарынан веб-парақтардың жүктелуі кідіруі мүмкін. Бұндай жағдайда Нагль алгоритмін өшіру (*TCP_NODELAY* параметрі) қарастырылған. Осы және басқа да шешімдер жайлы Mogul және Minshall (2001) басылымынан оқуға болады.

TCP хаттамасының өнімділігін төмендететін тағы бір мәселе, **ақымақ терезе синдромы (silly window syndrome)** деген атпен белгілі (Clark, 1982). Мәселе мығынасы мынада, TCP-ішкі жүйесі деректерді ірі бөліктермен жөнелтеді, ал қабылдаушы беттегі интерактивті қосымша оны символдап оқиды. Жағдай түсінікті болу үшін *6.35-суретті* қарастырайық. Бастапқы қалып-күй мынадай: қабылдаушы беттегі TCP-буфер толық (демек, терезе көлемі 0-ге тең) және бұл жағдайды жөнелтуші біледі. Сонан кейін интерактивті қосымша TCP-ағыннан бір символды оқиды. Қабылдаушы TCP-ішкі жүйесі жөнелтушіге терезе көлемінің үлкейгендігі және енді 1 байт жөнелтуге болатындығы жайлы хабарлайды. Жөнелтуші 1 байт жөнелтеді. Буфер қайта толады. Қабылдаушы 1 байттық сегмент үшін растау жөнелтеді және нөлдік терезе көлемі жайлы хабарлайды. Осылай шексіз жалғасуы мүмкін.



6.35-сурет. Ақымақ терезе синдромы

Дэвид Кларк (David Clark) қабылдаушы бетке бір байттық терезе көлемі жайлы хабарлауға тыйым салуды ұсынды. Оның орнына қабылдаушы буферде бос орынның едәуір жиналғанын күтуі тиіс. Нақтырақ, қабылдаушы байланыс орнатқан кезде хабарлаған ең үлкен көлемдегі сегментті қабылдайтындай жағдай туғанша немесе буфер кем дегенде жартылай босағанша терезе көлемі жайлы мәлімет жөнелтпеуі керек. Бұдан басқа, жөнелтуші тым кіші сегметтерді жөнелтуден бас тартып, жөнелту тиімділігін арттыруға ықпал ете алады. Оның орнына ол толық немесе кем дегенде қабылдаушының жарты буфер көлеміне тең сегмент жөнелтуге болатындай, терезе көлемінің жеткілікті түрде үлкен болғанын күтуі қажет.

Ақымақ терезе синдромынан арылу мәселесінде Нагль алгоритмі және Кларк шешімі бір-бірін толықтырады. Нагль деректерді TCP-ішкі жүйесіне символдап беру мәселесін шешуге тырысты. Кларк TCP-дан деректерді символдап қабылдайтын қосымша мәселесін шешуге тырысты. Екі шешімде жақсы және бірлесіп жұмыс істей алады. Олардың түп мағынасы: деректерді тым кіші бөліктермен жөнелтпеу және тым кіші бөліктермен жөнелтуді сұрамауда.

Қабылдаушы TCP-ішкі жүйесі терезенің ең үлкен көлемі жайлы ақпаратты жаңарта отырып, өнімділікті арттыру мәселесінде бұдан әрі бара алмайды. Жөнелтуші TCP-ішкі жүйесі тәрізді ол да, үлкен көлемдегі деректер жинақталғанша оларды буферлеп, қосымшалардан түскен `READ` оқуға сұранысты оқшаулай алады. Сонымен, TCP-ішкі жүйесі үндеу азаяды (онымен бірге үстеме шығындарда). Әрине мұндай әдіс жауапты күту уақытын үлкейтеді,

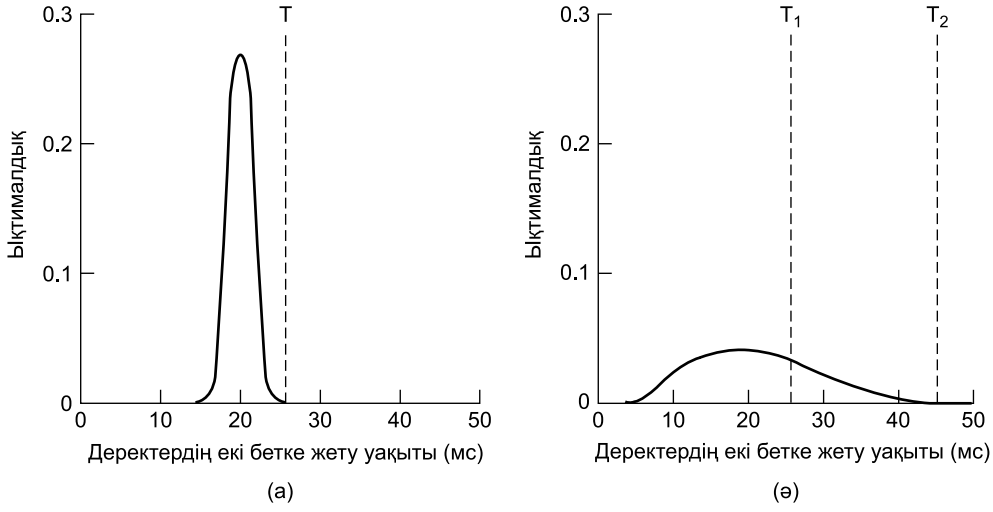
бірақ интерактивті қосымшалар үшін, мысалы, файлдарды тасымалдау кезінде бүкіл операцияға кеткен уақыттың төмендеуі жеке сұранысқа жауапты күту уақытының үлкеюінен әлдеқайда маңызды.

Қабылдаушының тағы бір мәселесі, сегменттер ретсіз келуі мүмкін. Олар дұрыс ретпен қосымша беруге болатын буферде сақталуы тиіс. Іс жүзінде ретсіз келген дестелерді қабылдамауға болады, бірақ жөнелтуші оларды бәрібір қайта жөнелтеді. Алайда, мұндай схема тиімсіз жұмыс істейді.

Растауды тек деректер растау байтына дейін қабылданған кезде ғана жөнелтуге болады. Бұл **жинақтаушы растау** деп аталады. Егер қабылдаушыға 0, 1, 2, 4, 5, 6 және 7-сегменттер келсе, онда ол тек 2-сегменттің соңғы байттына дейін растай алады. Жөнелтушінің күту уақыты аяқталған кезде ол 3-сегментті қайта жөнелтеді. Егер 3-сегмент келгенше 4-тен 7-ге дейінгі сегменттер буферде сақталса, ол барлық барлық байттардың, тіпті, 7-сегменттің соңғы байтына дейін растау жөнелте алады.

6.5.9. TCP-де таймерлерді басқару

TCP хаттамасында көптеген түрлі таймерлер қолданылады (концепция солай). Олардың ішіндегі ең маңыздысы **қайталап тасымалдау таймері (RTO, Retransmission TimeOut).** Сегмент жөнелтілген кезде қайталап тасымалдау таймері іске қосылады. Егер сегменттің қабылданғандығы жайлы растау таймер уақыты кезеңі біткенше келсе, таймер тоқтатылады. Егер керісінше, уақыт кезеңі растау келгенше бітіп кетсе, сегмент тағы да жөнелтіледі (ал таймер қайта іске қосылады). Сәйкесінше сұрақ туындайды: күту уақыт аралығы қандай болу керек?



6.36-сурет. Растаудың келу уақыты ықтималдығының тығыздығы: а – арналық деңгейде; ә – TCP үшін

Бұл мәселе арналық деңгейдің 802.11 тәрізді хаттамаларына қарағанда транспорттық деңгейде әлдеқайда күрделілеу. 802.11 хаттамасында күту кідірісі микросекундпен есептеледі және оны болжау жеңіл (шашырау аралығы үлкен емес), сондықтан таймерді растаудың болжамалы келу уақытынан сәл кейін қоюға болады (*6.36 а-суретін қараңыз*). Арналық деңгейде растау көп уақытқа сирек кідіретін болғандықтан (кептеліс жоқ болғандықтан), белгіленген уақытта растаудың келмеуі кадрдың немесе растаудың жоғалғандығын білдіреді.

TCP хаттамасы мүлдем басқа жағдайда жұмыс істеуге мәжбүр. Растаудың келу уақыты тығыздығының ықтималдығы функциясының таралуы бұл деңгейде жатықтау (*6.36 ә-сурет*), ал вариациялығы үлкен. Сондықтан деректердің жөнелтушіден қабылдаушыға дейін неше уақыт жүретінін болжау оңай емес. Егер күру уақыт кезеңінің мәнін тым кіші етіп алатын болсақ (мысалы, T_1 *6.36 ә-суретте*), Интернетті пайдасыз дестелермен толтырып тастайтын артық қайта тасымалдаулар пайда болады. Егер бұл кезең мәнін тым үлкен алсақ (T_2), онда күту уақытының ұлғаюының салдарынан деста жоғалған кезде өнімділік төмендейді. Одан бетер, орташа мән және растаудың келу уақыты дисперциясының шамасы асыра жүктелу пайда болғанда және ол қалпына келгенде бірнеше секунд аралығында өзгеруі мүмкін.

Шешім – желі өнімділігі өлшемдеріне сүйене отырып, үнемі өзгеріп отыратын күту кезеңі мәнінің динамикалық алгоритмін пайдалану. TCP-де қолданылатын алгоритмді 1988 жылы Джекобсон (Jacobson) құрастырған. Алгоритм жұмысы мынадай: әр байланыс үшін TCP хаттамасында ***SRTT*** **(Smoothed Round-Trip Time – орташа айналмалы кідіріс)** айнымалысы қарастырылған. Айнымалыда осы байланыс үшін растауды алуды күтудің ағымдағы ең жақсы уақыты сақталуы тиіс. Сегмент жөнелтілген кезде растауды алуға қажет уақытты өлшейтін және растау уақытылы келмеген кезде қайта тасымалдауды бастайтын таймер іске қосылады. Егер растау күту кезеңі аяқталғанша келіп үлгерсе, TCP-ішкі жүйесі оның келгеніне кеткен уақытты (R) өлшейді. Сонан кейін *SRTT* айнымалысының мәні

$$SRTT=\alpha SRTT+(1-\alpha)R$$

формула бойынша жаңартылады. Мұндағы α – ескі мәндердің қаншалықты жылдам ұмытылатындығын анықтайтын салмақтық коэффициент. Әдетте, ол 17/8-ге тең. Бұл формула – **салмақталған сырғымалы орташа шама (EWMA, Exponentially Weighed Moving Average)** немесе төменгі жиілік сүзгісі, оның көмегімен шуды жоюға болады.

Тіпті, *SRTT* айнымалысының мәні белгілі болған кезде де растауды күту кезеңін анықтау оңай шаруа емес. TCP-дің алғашқы нұсқаларында бұл мән *2xRTT* түрінде есптелген, алайда, тұрақты көбейткіштің иілгіш емес және растау келетін уақыт мәні шашыраңқы өсетін жағдайды есепке алмайтыны тәжірибеде көрсетілген болатын. Жеке алғанда, кездейсоқ (пуассондық) трафиктер кезегі жүктеме өткізгіштік қабілеттілікке жақындаған кезде кідірісінің және оның шашыраңқылығының өсетіндігін көрсетеді. Нәтижесінде қайталап тасымалдау

таймері іске қосылуы мүмкін, одан кейін дестенің түпнұсқасы желіде болса да оның көшірмесі жөнелтіледі. Әдетте, бұндай жағдайлар жүктеме жоғары болған кезде орын алады және бұл желіге артық десте жөнелтудің ең жақсы уақыты.

Бұл мәселені шешу үшін Джекобсон күту уақыты кезеңін айналмалы кідірске және айналмалы кідірістің орташа шамасының ауытқуына сезімтал етуді ұсынды. Ол үшін тағы бір тегістелген айнымалы қажет болды – **RTTVAR (Round-Trip Time Variation – айналма кідіріс аутқуы)**. Айнымалы:

$$RTTVAR=\beta RTTVAR+(1-\beta)|SRTT-R|$$

формуласы бойынша есептеледі.

Алдыңғы жағдайдағыдай бұл – салмақталған сырғымалы орташа шама. Әдетте β=3/4. Күту кезеңінің мәні, *RTO*:

$$RTO=SRTT+4\times RTTVAR.$$

формуласы бойынша анықталады.

Көбейткішті 4 деп таңдау еркін болып табылады, алайда, бүтін санды 4-ке көбейтуді бір жылжыту командасының көмегімен орындауға болады. Ал күту уақытының мұндай мәнінде қайта тасымалдау саны бір пайыздан артық болмайды. RTTVAR айнымалысының стандартты емес, орташа ауытқу екеніне назар аударыңыз, бірақ іс жүзінде олардың мәндері өте жақын. Джекобсон жұмысында күту кезеңі мәнін тек бүтін қосу, азайту және жылжыту көмегімен есептеудің көптеген жолдары келтірілген. Қазіргі хосттар үшін мұндай үнемділік қызық болмаса да, оны TCP-ді жаппай қолдану идеясы қажет етеді: ол суперкомпьютерде де, кішігірім құрылғыларда да жұмыс істеуі керек. RFID-чиптер үшін оны әлі ешкім жүзеге асырған жоқ. Кім біледі? Ол да мүмкін.

Күту кезеңі, сонымен бірге айнымалылардың бастапқы мәндері қалай есептелетіні жайлы толық мәліметтерді RFC 2988 құжатынан алуға болады. Қайталап тасымалдау таймері үшін, алдын ала бағалауға қарамастан ең кіші мән 1 с-ке тең етіп орнатылады. Бұл мән Allman және Paxson (1999) өлшеулеріне негізделген, артық қайта тасымалдауды болдырмас үшін қажет.

Айналма кідіріс R-ді есептеу үшін деректерді алғанда, сегмент қайта жөнелтілген кезде не істеу керек деген сұрақ туындайды. Қандай да бір сегмент үшін растау келген кезде – оның дестенің бірінші жөнелтілуіне, не соңғы жөнелтілуіне қатысты екені түсініксіз. Қате болжау күту кезеңі мәнін қатты төмендетуі мүмкін. Бұл мәселені радиоэуесқой Фил Карн (Phil Karn) анықтаған болатын. Оны TCP/IP-дестелерді өзінің сенімсіздігімен танымал қысқатолқынды әуесқой радиобайланыс арқылы тасымалдау қызықтырған. Фил Карнның ұсынысы өте қарапайым: қайта жөнелтілген дестелер үшін бағалауды жаңартпау керек. Бұдан басқа, әрбір қайта жөнелту кезінде күту уақытын сегмент бірінші талпыныстан жеткенше еселеу керек. Бұл түзетулер **Карн алгоритмі (Karn's algorithm)** (Karn және Partridge, 1987) деген атқа ие болды. Ол TCP-дің көптеген іске асыруларында пайдаланылады.

TCP хаттамасында тек қайталап тасымалдау таймері қолданылмайды, сонымен бірге **табандылық таймері (persistence timer)** деп аталатын тағы бір

таймер бар. Ол келесі тұйыққа тірелуді болдырмауға негізделген. Қабылдаушы нөлдік терезе көлемі көрсетілген растау жөнелтіп, жөнелтушіге күте тұру керек екенін білдіреді. Біраз уақыттан кейін қабылдаушы терезенің жаңа көлемін көрсетіп деста жөнелтеді, алайда, бұл деста жоғалып кетеді. Енді екі жақта өзіне қарсы беттің іс-әрекетін күтеді. Табандылық таймері іске қосылған кезде жөнелтуші қабылдаушыға: «Ағымдағы қалып-күй өзгерді ме?» – деген сауалы бар деста жөнелтеді. Қабылдаушы жауап ретінде ағымдағы терезе көлемін жөнелтеді. Егер әлі де нөлге тең болса, табандылық таймері қайта іске қосылады да, цикл қайталанады. Егер терезе көлемі ұлғайса, жөнелтуші деректер жөнелте алады.

Хаттаманың кейбір жүзеге асыруларында **белсенділікті тексеру таймері (keepalive timer)** деп аталатын үшінші таймер қолданылады. Байланыс ұзақ уақыт аралығында пайдаланылмайтын болса, екі жақтың бірін байланыстың қарсы бетінде тірі жан бар екенін тексеруге мәжбүрлейтін белсенділікті тексеру таймері іске қосылады. Егер тексеруші жақ жауап алмаса, байланыс үзіледі. Хаттаманың бұл ерекшелігінің қарама-қайшылығы бар, себебі ол қосымша үстеме шығындарға әкеледі және өмірге бейім байланыстарды байланысты қысқа уақыттық жоғалту салдарынан үзіп жібереді.

Әр TCP-байланыста қолданылатын соңғы таймер – байланысты жабар кезде соңғы автоматтың *TIME WAIT* қалып-күйінде іске қосылатын таймер. Ол байланыс жабылғаннан кейін желіде өзі құрастырған дестелер қалмайтынына кепілдік беру үшін дестенің екі еселенген өмір уақытын санайды.

6.5.10. TCP-дегі асыра жүктелуді бақылау

TCP-дің ең маңызды функцияларының бірі – асыра жүктелуді бақылау. Қандайда бір желіге оның өңдейтін қабілеттілігінен әлдеқайда көп деректер келіп түскен кезде, желіде кептелістер пайда болады. Бұл жағдайдан Интернентте қашып құтыла алмайды. Желілік деңгей маршруттауыштарда ұзын-сонар кезектердің пайда болғандығы жайлы білген кезде, ол жағдайды дестелерді жою арқылы түзеуге тырысады. Желілік деңгеймен кері байланыс транспорттық деңгейдің міндеті, бұл оған асыра жүктелуді бақылап, қажет болған кезде дестелерді жөнелту жылдамдығын төмендетуге мүмкіндік береді. Интернеттегі асыра жүктелуді бақылауда және деректерді транспорттауда TCP таптырмас құрал.

Асыра жүктелуді бақылаудың жалпы сұрақтарын біз *6.3-бөлімде* қарастырған болатынбыз. Негізгі ой мынада: **AIMD (Additive Increase Multiplicative Decrease – аддитивті жоғарылату мильтиплекативті төмендету)** басқару заңдылығын пайдаланатын транспорттық хаттама желінің асыра жүктелгендігі жайлы екілік сигналды алған кезде, өткізгіштік қабілеттілікті әділ таратумен қосылады. TCP асыра жүктелуді бақылаудың бұл амалын терезелер көмегімен жүзеге асырады, сигнал ретінде дестенің жоғалуы пайдаланылады. Уақыттың әр сәтінде желіде байтар саны әр жөнелтушінің бекітілген санынан артық

болуы мүмкін емес. Осы байттар саны **асыра жүктелу терезесін** құрайды. Сәйкесінше жөнелту жылдамдығы байланыстың айналма кідірісіне бөлінген терезе көлеміне тең. Терезе көлемі AIMD ережелеріне сәйкес беріледі.

Естеріңізде болса, асыра жүктелу терезесінен басқа, қабылдаушының буферге орналастыра алатын байттар санын анықтайтын ағынды басқару терезесі де бар. Бұл екі параметр де маңызды рөл атқарады: жөнелтуші желіге жібере алатын байттар саны осы терезелердің ең кішісінің мәніне тең. Сонымен тиімді терезе – жөнелтушіні және қабылдаушыны қанағаттандыратын көлемнің ең кішісі. Мұнда екі жақтың да қатысы қажет. Егер екі терезенің бірі уақытша толған болса, TCP деректер тасымалдауды тоқтатады. Егер қабылдаушы: «64 Кбайт жөнелтіңіз» – десе, алайда, жөнелтуші 32 Кбайттан көп жөнелтсе желіде кептеліс болатынын біледі, сондықтан ол 32 Кбайт жөнелтеді. Егер жөнелтуші желінің үлкен көлемдегі деректер санын өткізе алатынын білсе, мысалы, 128 Кбайт, онда ол қабылдаушы сұраған көлемдегі (64 Кбайт) деректерді жөнелтеді. Ағынды басқару терезесі ертеде сипатталған болатын, сондықтан алдағы уақытта тек асыра жүктелу терезесі жайлы айтатын боламыз.

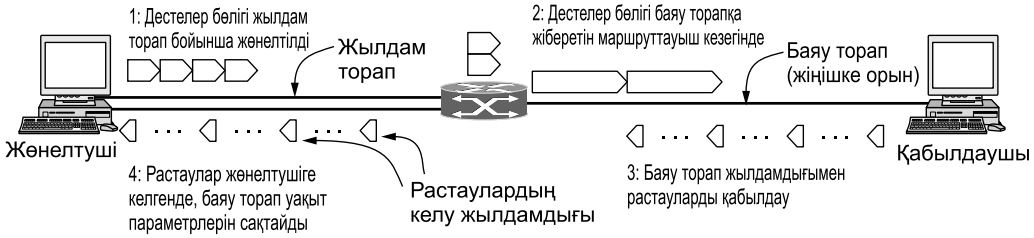
Қазіргі заманғы асыра жүктелуді бақылау схемасы TCP-де Ван Джекобсон (Van Jacobson, 1988) іске асырған болатын. Бұл шын мәнінде, қызықты оқиға. 1986 жылдан бастап Интернет танымалдылығының өсуі, кейіннен **асыра жүктелу салдарынан желінің тұрып қалуы (congestion collapse)** деп аталған жағдайға алып келді. Бұндай кезде тиімді өткізгіштік қабілеттілік желінің асыра жүктелу салдарынан ұзақ уақытқа күрт төмендеп (100 еседен аса) кететін. Джекобсон (және басқалар) жайдайды сараптап, қандай да бір тиімді шешімін табуға бел буды.

Нәтижесінде Джекобсонға жоғары деңгей шешімін – асыра жүктелу терезесін таңдау үшін AIMD тәсілін пайдалануды іске асыруға мүмкіндік туды. Әсіресе, TCP-де асыра жүктелуді бақылаудың күрделілігіне қарамастан, ол оны қолданыстағы хаттамаға, ешбір мәлімдеме форматын өзгертпей енгізе алғандығы қызық. Осының нәтижесінде жаңа шешімді бірден іске енгізу мүмкін болды. Джекобсон алдымен дестенің жоғалуына, ақпараттық сәл кешігіп келуіне (асыра жүктелу басталғаннан кейін) қарамастан, асыра жүктелудің сенімді сигналы екенін байқады. Соңында, асыра жүктелу кезінде дестелерді жоймайтын маршруттауышты елестету мүмкін емес. Әрі қарай бұның өзгеруі екіталай. Тіпті буферлік жады терабайтпен есептелгенде де, желі жылдамдығы да секундына бірнеше терабайтқа өседі.

Мұнда бір жіңішкелік бар. Дестенің жоғалуын асыра жүктелу сигналы ретінде пайдалану, тасымалдау қателігі салыстырмалы түрде сирек орын алады деп болжайды. Сымсыз желілер жағдайында (мысалы, 802.11) бұл олай емес, сондықтан ол жерде қайта тасымалдаудың арналық деңгейдегі жеке өз механизмі қолданылады. Қайта тасымалдау ерекшелігіне байланысты сымсыз желілерді желілік деңгейде тасымалдау қателігі салдарынан болған дестелердің жоғалуы есепке алынбайды. Сымды желілерді және оптыталшықты тораптарда биттер бойынша қателіктер жиілігі әдетте, өте төмен.

Интернетке арналған TCP алгоритмдерінің барлығы дестелер асыра жүктелу салдарынан жоғалады деген болжамға негізделген. Сондықтан олар өз шымшығын бақылаған шахтерлердей тайм-ауттарды мұқият қадағалап, мәселенің кез келген белгісін анықтауға тырысады. TCP-дегі қайталап тасымалдау таймері орташа мәнді және айналма кідіріс ауытқуын есепке алатынын айтқан болатынбыз. Осындай таймерлерді ауытқуды есепке ала отырып, жақсарту Джекобсон жұмысындағы маңызды қадам болды. Егер қайта тасымалдауды күту уақыты дұрыс таңдалған болса, TCP-жөнелтуші желіні жүктейтін шығыс байттар санын қадағалай алады. Ол үшін оған жөнелтілген және расталған дестелер реттік нөмірін салыстыру жеткілікті.

Енді біздің мәселеміз айтарлықтай қарапайым болып көрінеді. Бізге керегі – асыра жүктелу терезесін бақылау (реттік және растау нөмірлер көмегімен) және оны AIMD ережесі бойынша өзгерту. Алайда, өздеріңіз байқағандай іс жүзінде бәрі әлдеқайда күрделі. Бірінші қиындық – дестелерді желіге жөнелту тәсілі (тіпті қысқа уақыт аралығында) желілік жолға байланысты болуы керек. Кері жағдайда кептеліс туындайды. Мысал ретінде қанығу терезесі 64 Кбайт, жылдамдығы 1 Гбит/с коммутациялық Ethernet желісіне қосылған хостты қарастырайық. Егер хост бір дегеннен толық терезе жөнелтсе, баяу ADSL-торап (1 Мбит/с) арқылы трафик шашу орын алуы мүмкін. Гигабиттік торап арқылы жарты миллисекундта өткен бұл трафик VoIP тәрізді хаттамаларды оқшаулап, баяу тораптың жұмысын жарты секунд уақытқа қатырып тастайды. Тек асыра жүктелумен күресуге бағытталған емес, керісінше, оны тудыруға бағытталған хаттама ғана осылай жұмыс істейді.



6.37-сурет. Жөнелтуші жіберген дестелер бөлігі және растаудың келу жылдамдығы

Алайда, дестенің кішігірім бөліктерін жөнелту пайдалы болуы мүмкін. Жылдам торапқа (1 Гбит/с) қосылған хост-жөнелтуші дестелердің кішігірім бөлігін (4) жолдың жиһішке бөлігі немесе ең баяу бөлігі болып саналатын баяу желіде (1 Мбит/с) орналасқан қабылдаушыға жөнелткен кезде не болатыны *6.37-суретте* көрсетілген. Алдымен бұл 4 десте желі бойымен жөнелтілген жылдамдықпен жылжиды. Сонан кейін маршруттауыш оларды кезекке орналастырады, себебі олар жылдам торап арқылы жөнелтілетін баяу желіге қарағанда жылдам келеді. Бірден жөнелтілген дестелер саны көп

болмағандықтан, бұл кезек аса ұзын емес. Суретте баяу желі арқылы өтетін дестелер ұзындау болып көрінетіне назар аударыңыздар, себебі олардың жөнелтілуі жылдам желіге қарағанда көп уақытты қажет етеді.

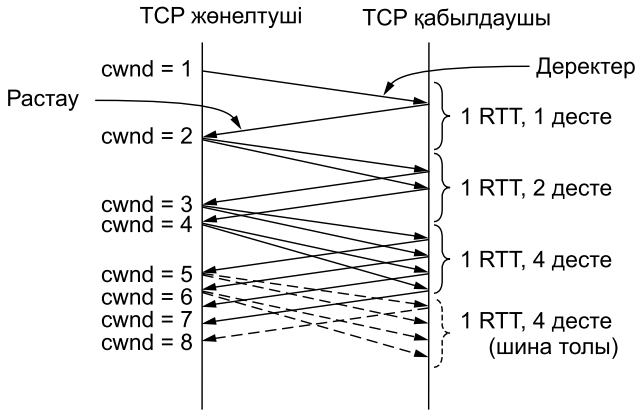
Соңында, дестелер қабылдаушыға келіп жетеді де олардың қабылданғандығы расталады. Растаудың жөнелтілу уақыты дестенің баяу арна арқылы келу уақытына байланысты. Сондықтан кері жолда дестелер арасындағы арақашықтық бастапқыдағы дестелердің жылдам арна арқылы қозғалғанынан ұзын болады. Ол растаудың желі арқылы және кері қозғалғанында өзгермейді.

Мұндай мына жағдай өте маңызды: растау жөнелтушіге дестелер жолдың ең баяу арнасы арқылы қозғалатын жылдамдықпен келеді. Жөнелтушіге де осы жылдамдық қажет. Егер ол дестелерді желіге осы жылдамдықпен жөнелтетін болса, олар ең баяу арна мүмкіндік беретіндей жылдамдықпен жылдам қозғалатын болады және маршруттауыштарда кезекте тұрып қалмайды. Мұндай жылдамдық **растаудың келу жылдамдығы (ack clock)** деп аталады және TCP-дің ажырамас бөлігі болып саналады. Бұл параметр TCP-ге трафикті тегістеп, маршруттауыштардағы қажет емес кезектерді болдырмауға көмектеседі.

Екінші қиындық – AIMD ережесі бойынша жылдам желілерде жақсы жұмыс режиміне жету, бастапқыда кіші асыра жүктелу терезесі таңдалған болса, тым көп уақытты талап етеді. Трафикті 10 Мбит/с жылдамдықпен және 100 мс айналма кідіріспен тасымалдауға мүмкіндік беретін орташа желілік жолды қарастырайық. Бұл жағдайда өткізгіштік қабілеттілік және кідіріс уақыты көбейтіндісіне тең, демек 1 Мбит немесе 1250 байттан 100 дестe, асыра жүктелу терезесін пайдаланған қолайлы. Егер бастапқыда терезе көлемін бір дестеге тең етіп алып, кейіннен айналма кідіріс уақытына тең уақыт кезеңінде оны бір дестеге ұлғайтса, байланыс тек 100 айналма кідіріс уақытынан кейін ғана, демек 10 с, қалыпты жұмыс істей бастайды. Бұл тым ұзақ. Теорияда біз үлкен терезеден бастай алар едік, айталық, 50 дестe көлеміндегі. Онда бірден 50 дестe жөнелткенде кептеліс туындайды – бұл сценарий жайлы біз жоғарыда айтқан болатынбыз.

Джекобсон ұсынған шешім – сызықтық және мультиплекстік ұлғайтуды біріктіреді. Байланыс орнатқан кезде жөнелтуші көлемі төрт сегменттен аспайтын терезе орнатады (Бастапқыда терезенің бастапқы көлемі бір сегменттен аспайтын, кейіннен бұл мән тәжірибе негізінде төртке дейін ұлғайтылды). Бұл жайлы RFC 3390 құжатында егжей-тегжейлі айтылған. Сонан кейін жөнелтуші желіге бастапқы терезені жөнелтеді. Дестелердің қабылданғандығы айналма кідіріске тең уақыттан кейін расталады. Сегменттің қабылданғандығы жайлы растау қайталап тасымалдау таймері іске қосылғанша келген әр жағдайда, жөнелтуші асыра жүктелу терезесін бір сегмент ұзындығына (байтпен) өсіреді. Бұдан басқа, егер сегмент қабылданса, демек, желіде бір сегмент аз болды. Сондықтан нәтижесінде әрбір расталған сегмент тағы екі сегмент жөнелтуге мүмкіндік береді. Сонымен, қанығу терезесі көлемі айналма кідірісіне тең уақыт кезеңінде екі есе ұлғаяды.

Бұл алгоритм **баяу старт (slow start)** деп аталады, алайда, іс жүзінде ол баяу емес – ол экспоненциалды өседі – әсіресе, бірден ағынды басқарудың толық терезесін жөнелтуге мүмкіндік беретін алдыңғы алгоритммен салыстырғанда. Баяу старт *6.38-суретінде* көрсетілген. Бірінші айналма кідіріс уақытында (RTT) жөнелтуші желіге бір десте жібереді (және қабылдаушы бір десте алады). Екінші айналма кідіріс уақытында екі, үшінші уақытта – төрт десте жөнелтіледі.



6.38-сурет. Бастапқы асыра жүктелу терезесі бір сегмент баяу старт

Баяу старт айналма кідіріс және жылдамдық мәнінің кең диапазоны үшін жақсы жұмыс істейді. Желі жолына байланысты жөнелту жылдамдығын реттеу үшін ол растаудың келу жылдамдығын пайдаланады. Растаудың жөнелтушіден қабылдаушыға қалай оралатынын қарастырайық (*6.38-сурет*). Жөнелтуші растау алғанда қанығу терезесін бірден 1-бірлікке өсіріп, желіге екі десте жөнелтеді. (Бір десте терезенің бір бірлікке өскеніне сәйкес келеді, ал екіншісі тағайындалған орынға жетіп, желіден шыққан жеткен десте орнына жөнелтіледі. Расталмаған дестелер саны асыра жүктелу терезесімен уақыттың әр сәтінде анықталады.) Алайда, бұл екі дестенің қабылдаушы хостқа жөнелтілген уақыт аралығында келуі міндетті емес. Айталық, жөнелтуші қуаттылығы 100 Мбит/с Ethernet желісіне қосылған делік. Әр 1250-байттық дестенің жөнелтілуіне 100 мкс кетеді. Сондықтан дестелер арасындағы кезең кіші болуы мүмкін, 100 мкс басталады. Егер дестелердің жолы қуаттылығы 1Мбит/с ADSL-торап арқылы өтетін болса, жағдай өзгереді. Енді әр дестені жөнелту үшін 10 мс қажет болады. Сөйтіп, дестелер арасындағы кезең кем дегенде, 100 есе өседі. Егер дестелер қандай да бір уақыт сәтінде, бір буферде жөнелтуді бірге күтіп қалмаса, бұл кезең сол үлкен күйінде қалады.

Егер дестелердің қабылдаушыға келу кезеңіне қарасақ, *6.38-суретте* сипатталған жағдайды көруге болады. Бұл кезең растауды жөнелту кезінде сақталады, демек, қабылдаушы оларды қабылдаған кезде де. Егер желілік жол баяу болса, растау баяу келеді (айналма кідіріске тең уақытта). Егер желі жолы

жылдам болса, растау да жылдам келеді (тағы да айналма кідіріске тең уақытта). Жөнелтуші жаңа дестені жөнелту кезінде тек растаудың келу жылдамдығын ескеруі тиіс – баяу старт алгоритмі осында орындайды.

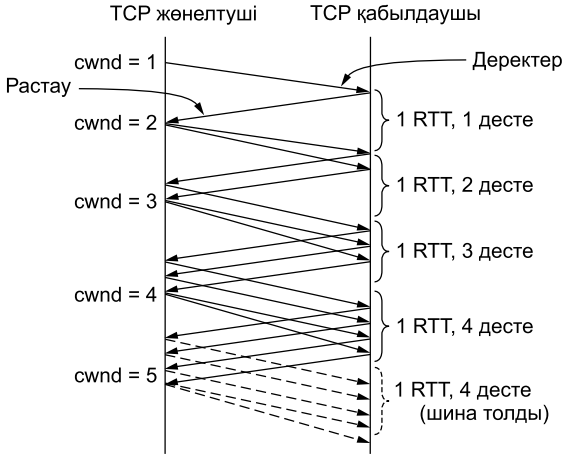
Баяу старт алгоритмі экспоненциалдық өсуге әкелетін болғандықтан қандай да бір сәтте (кейінірек болғаннан гөрі ертерек) желіге тез арада тым көп дестелер жөнелтіледі. Нәтижесінде маршруттауыштарда кезектер пайда болады. Кезек асыра толғанда дестелер жоғала бастайды. Бұл жағдайда, дестенің қабылданғандығы жайлы растау жөнелтуші хостқа келмеген кезде сәйкес таймер іске қосылады. Қанығу терезесінің қай сәтте тым үлкен болғанын *6.38-суреттен* көруге болады. Үш айналма кідіріс уақытынан кейін желіде төрт десте жүреді. Бұл осы байланыс үшін асыра жүктелу терезесінің көлемі төрт десте болғаны қолайлы дегенді білдіреді. Алайда, бұл дестелердің қабылданғандығы расталатын болғандықтан, баяу старт алгоритмі асыра жүктелу терезесін өсіруді жалғастыра береді және нәтижесінде бір айналма кідіріс уақытынан кейін жөнелтуші желіге сегіз десте жөнелтеді. Неше дестенің жөнелтілгендігіне қарамастан тек төрт десте бір айналма кідіріс уақытында тағайындалған орынға жетіп үлгереді. Бұл желілік шинаның толғандығын білдіреді. Қосымша дестелер желіге түскеннен кейін маршруттауыштардағы кезектерде тұрып қалады, себебі желі оларды қабылдаушыға мүмкіндігінше жылдам жеткізе алмайды. Жақын арада кептеліс және дестелердің жоғалуы орын алады.

Баяу стартты бақылау үшін жөнелтуші әр байланыс үшін шекті мәнді жадыда сақтауы тиіс. Бұл **баяу старт шегі (slow start threshold)** деп аталады. Бастапқыда байланыстың мүмкіндіктерін шектемеу үшін, ағынды басқару терезесінің көлемінен аспайтын еркін жоғары мән орнатылады. TCP баяу старт алгоритмін пайдалана отырып, асыра жүктелу терезесін ұлғайтуды тайм-аут немесе асыра жүктелу терезесінің көлемі шекті мәннен асқанша (немесе қабылдаушы терезесі толғанша) жалғастырады.

Десте жоғалуы анықталғанда (мысалы, тайм-аут жағдайында) баяу старт шегі асыра жүктелу терезесінің жартысына тең етіп орнатылады және бүкіл үдеріс басынан басталады. Мұндағы ой, егер асыра жүктелудің ағымдағы терезесі тым үлкен болса, ол біраз кідіріспен тұрақтандыруға болатын асыра жүктелуге әкеледі. Асыра жүктелу болмаған, екі есе кіші терезе көлемі, асыра жүктелу терезесіне қолайлы: жолдың өткізгіштік қабілеттілігі тиімді қолданылады және деректер жоғалмайды. Біздің мысалымызда, *6.38-суретте,* терезенің сегіз дестеге дейін өсуі деректердің жоғалуына әкелуі мүмкін, ал көлемі төрт десте терезе ыңғайлы болар еді. Бұдан кейін асыра жүктелу терезесінің көлемі бастапқы мәнге тең етіп орнатылады және баяу старт алгоритмі басынан бастап орындалады.

Баяу старт шегінен асып кеткен кезде TCP баяу старттан аддитивті ұлғайтуға көшеді. Бұл режимде асыра жүктелу терезесі айналма кідіріске тең уақыт кезеңінен кейін бір сегментке өседі. Баяу старт жағдайындағыдай, терезе көлемінің өсуі, әр айналым сайын бір сегментке емес, жеткізілгендік жайлы растау алынған кезде жүзеге асырылады. Айталық, *cwnd* – асыра жүктелу

терезесі, ал *MSS* – сегменттің ең үлкен өлшемі. Әдетте, терезенің ұлғаюы, жеткендігі расталуы мүмкін әр *cwnd/MSS* дестесі үшін *(MSS×MSS)/cwnd* коэффициентімен жүргізіледі. Бұл ұлғаюдың жылдам болуы міндетті емес. Жалпы ой, TCP-байланыс – өткізгіштік қабілеттілік төмен болатындай тым кіші де емес және кептеліс туындайтындай тым үлкен де емес, оңтайлы терезе көлемімен мүмкіндігінше ұзақ жұмыс істеу керек.



6.39-сурет. Терезенің бастапқы көлемі бір сегмент болғандағы аддитивті ұлғайту

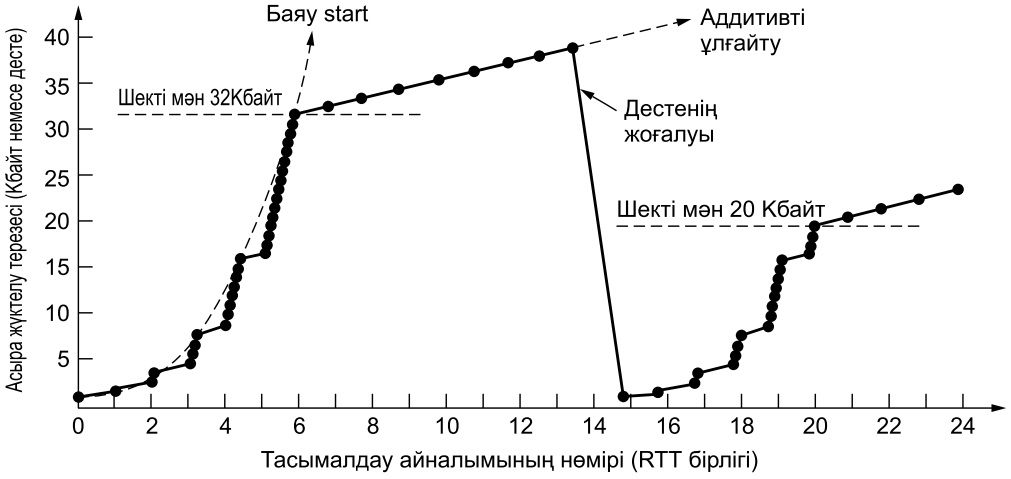
Аддитивті ұлғайту *6.39-суретте* көрсетілген. Жағдай баяу стар кезіндегідей. Әр айналым соңында жөнелтушінің асыра жүктелу терезесі желіге қосымша бір дестe жіберуге болатындай етіп өсіріледі. Баяу стартпен салыстырғанда өсудің тізбектік жылдамдығы өте төмен болып келеді. Кішкене терезелер үшін айырмашылық айтарлықтай емес, бірақ егер сізге терезені, мысалы, 100 сегментке өсіру керек болса, сезінерліктей болады.

Мұндай біраз нәрсені жақсартып, өнімділікті өсіруге болады. Бұл схеманың кемшілігі – тайм-аутты күту. Әдетте, растауды күту уақыты үлкен болады, себебі оның бағасы төмендетілген болу керек. Егер дестe жоғалса, қабылдаушы сәйкес растауды жөнелтпейді, яғни растау нөмірі өзгермейді. Жөнелтуші желіге жаңа дестелерді жібере алмайды, себебі оның терезесі толық. Хост мұндай қалып-күйде, таймер іске қосылып, дестені қайталап жөнелту орын алғанша, ұзақ уақыт болуы мүмкін. Осы сәтте баяу старт қайта басталады.

Жөнелтуші дестелердің бірінің жоғалғандығын жылдам біле алатын бір амал бар. Жоғалған дестеден кейінгі дестелер қабылдаушыға келе бастаған кезде, олар жөнелтушіге келетін растаудың жөнелтілуін іске қосады. Олардың барлығының растау нөмірі бірдей және ол **растау көшірмелері (duplicate acknowledgements)** деп аталады. Жөнелтуші әрбір көшірмені алған сайын, жоғалған дестенің келмей, басқа дестенің келу ықтималдығы бар.

Дестелер әртүрлі жолдармен жүруі мүмкін болғандықтан, олар ретсіз келеді. Бұл жағдайда растау көшірмелері дестe жоғалғандығының белгісі емес.

Алайда, Интернетте мұндай жағдай сирек кездеседі. Егер дестелердің түрлі жолдармен жүру нәтижесінде олардың реті бұзылса да, ол елеуліктей қатты бұзылмайды. Сондықтан TCP-де үш растау көшірмесі шартты түрде дестенің жоғалғандық белгісі болып саналады. Сонымен бірге растау нөмірі бойынша нақты қай дестенің жоғалғандығын анықтауға болады. Бұл реті бойынша келесі деста. Оны қайталап жөнелтуді таймердің іске қосылғанын күтпей, бірден орындауға болады. Бұл эвристикалық тәсіл **жылдам қайталу (fast retransmisson)** деген атқа ие болды. Жылдам қайталау орын алған кезде баяу старт шегін тайм-аут жағдайындағыдай ағымдағы терезенің жарты көлеміне тең етіп орнатылады. Баяу стартты терезе көлемін бір сегмент етіп алып бастауға болады. Жаңа деста қайталап жөнелтілген дестеге, сонымен бірге дестенің жоғалғандығы анықталғанға дейін жөнелтілген қалған басқа деректерге растау келіп жететіндей уақыттан кейін жөнелтіледі.



6.40-сурет. TCP Tahoe-дағы баяу старт және келесі аддитивті өсу

Қазіргі сәтте бізде жұмысы *6.40-суретте* бейнеленген, асыра жүктелуді бақылау алгоритмі бар. Бұл версия, 1988 жылы шығарылған және осы ойлардың барлығы жүзеге асырылған 4.2BSD Tahoe құрметіне TCP Tahoe деп аталады. Бұл мысалдағы сегменттің ең үлкен көлемі 1 Кбайтқа тең. Бастапқыда асыра жүктелу терезесінің көлемі 64 Кбайтқа тең етіп орнатылған, сонан кейін тайм-аут орын алды және порт 32 Кбайтқа тең болды, ал асыра жүктелу терезесі – 1 Кбайт (тасымалдау 0). Сонан кейін асыра жүктелу терезесі шекке жеткенше (32 Кбайт) әр қадамда екі еселенеді. Терезе көлемі әрбір растау келген сайын өсіп отырады, демек, үздіксіз, сондықтан біздің графигіміз дискретті сатылы. Әр айналымда терезе көлемі бір сегментке өседі.

Тасымалдаудың он үшінші айналымы сәтсіз болып шығады (солай болуы тиіс) және олардың бірі дестенің жоғалуымен аяқталады. Жөнелтуші оны үш растау көшірмесін алғаннан кейін біледі. Бұдан кейін жоғалған деста қайта

жөнелтіледі, ал шекті мән ағымдағы терезенің жарты көлеміне тең етіп орнатылады (40 Кбайттың жартысы 20 Кбайт) және баяу старт қайта басталады. Терезе көлемі бір сегмент жаңа баяу старттың басталуына дейін бір айналым уақыт өтеді. Осы аралықта ертеде тасымалданған деректер, жоғалған десте көшірмесімен қоса желіден кетіп үлгереді. Асыра жүктелу терезесі баяу старт алгоритміне сәйкес 20 Кбайттық шекті мәнге жеткенше қайта ұлғаяды. Осыдан кейін терезе көлемінің өсуі қайтадан сызықты болады. Растау көшірмелерінің көмегімен немесе тайм-аут орын алғаннан (немесе қабылдаушы терезесі толғанша) кейін анықталатын десте жоғалғандығы осылай белгілі болады.

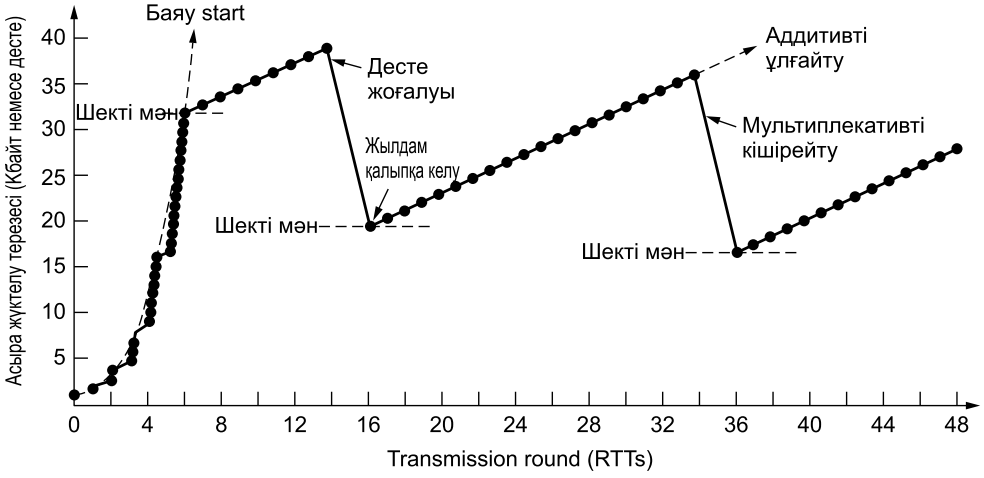
TCP Tahoe версиясы (мұнда қайталап тасымалдаудың жақсы таймерлері қолданылады) желінің асыра жүктелу салдарынан тұрып қалу мәселесін шешетін, асыра жүктелудің жұмысшы алгоритмдерін іске асырады. Джекобсон оны қалай жақсарту керек екенін ойлап тапты. Жедел қайта тасымалдау кезінде байланыс тым үлкен көлемдегі тереземен жұмыс істейді, бірақ растаудың келуі есепке алына береді. Әр растау көшірмесі келген сайын, тағы бір дестенің желіден шығу ықтималдығы жоғары келеді. Сөйтіп, желідегі дестілердің жалпы санын есептеп, әрбір қосымша растау көшірмесін алған сайын жаңа десте жөнелтуге болады.

Бұл ойды жүзеге асыратын эвристикалық тәсіл **жылдам қалыпқа келу (fast recovery)** деген атқа ие болды. Бұл – баяу старттың шегі ретінде ағымдағы терезе көлемі немесе оның жартысы (жылдам қайта тасымалдау кезінде) алынған кезде растаулардың келу жылдамдығын төмендетпеуге мүмкіндік беретін уақытша режим. Бұл үшін растау көшірмелері (жылдам тасымалдауды іске қосқан үшеуін қоса) желідегі дестелер саны жаңа шекті мәнге дейін төмендегенше саналады. Бұған айналма кідірістің шамамен жартысы кетеді. Осы сәттен бастап жөнелтуші әрбір қабылданған растау көшірмесі үшін желіге жаңа десте жөнелте алады. Жылдам қайта тасымалдаудан кейінгі бір айналымда жоғалған дестенің қабылданғандығы расталады. Осы уақыттан бастап растау көшірмелерінің толассыз ағынмен келуі тоқталады және алгоритм жылдам қалыпқа келу режимінен шығады. Асыра жүктелу терезесі баяу старттың жаңа шекті мәніне тең болады және сызықты түрде ұлғая бастайды.

Нәтижесінде бұл тәсіл жаңа байланыс орнату және тайм-аут орын алған кезден басқа көп жағдайда баяу старттан бас тартуға мүмкіндік береді. Тайм-аут бірден көп десте жоғалған кезде орын алады, ал жылдам қайталау көмектеспейді. Қайта-қайта баяу стартты бастағанның орнына, активті байланыс асыра жүктелу терезесі түрі араға ұқсас аддитивті ұлғаю сызығы (бір айналымда бір сегмент) және мультиплекативті кему (бір айналымда екі рет) бойымен жылжыйды. Бұл – біз бастапқыда жүзеге асырмақ болғнан AIMD ережесі.

Осындай ара тәрізді қозғалыс *6.41-суретте* көрсетілген. Бұл тәсіл TCP Reno қолданылады. Ол 1990 жылы шыққан 4.3BSD Reno құрметіне TCP Reno деп аталған. Іс жүзінде TCP Reno – жылдам қалыпқа келуі бар TCP Tahoe. Бастапқы баяу старттан кейін қанығу терезесі жөнелтуші растаудың қажетті

көшірмелерін алып, дестелердің жоғалғандығын анықтағанша сызықты түрде ұлғаяды. Жоғалған десте қайта жөнелтіледі де, алгоритм әрі қарай жылдам қалыпқа келу режимінде жұмыс істейді. Бұл режим растаулардың келу жылдамдығын есепке алуды қайта жөнелтілген дестеге растау келгенше тоқтатпауға мүмкіндік береді. Бұдан кейін асыра жүктелу терезесінің мәні 1 емес, жаңа баяу старт шегінің мәніне тең болады. Бұл белгісіз ұзақ уақытқа созылуы мүмкін. Асыра жүктелу терезесі үнемі өткізгіштік қабілеттілік және кідіріс уақыты көбейтіндісінің оңтайлы мәніне жақын болады.



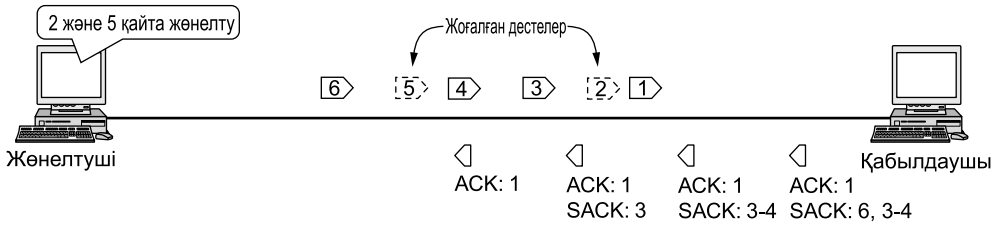
6.41-сурет. Жылдам қалыпқа келу немесе TCP Reno үшін ара тәріздес график

TCP Reno қолданылатын терезе көлемін таңдау механизмі екі онжылдықтан аса уақыт TCP-дегі асыра жүктелуді бақылаудың негізі болып келеді. Осы жылдар ішінде механизм көптеген өзгерістерге ұшырады, жеке алғанда – бастапқы терезені таңдаудың жаңа әдістері пайда болды, бір мағыналы емес жағдайлар жойылды. Жақсартулар екі және одан да көп дестелер жоғалғаннан кейін қалыпқа келтіру механизмдерін де қамтыды. Мысалы, TCP RenoNew версиясы бір жоғалған дестеден кейін қабылданған ішінара растаулар нөмірлерін басқа жоғалған дестені қалпына келтіру үшін пайдаланады (Hoe, 1996) (RFC 3782 құжатын қараңыз). 1990 жылдың ортасынан бастап жоғарыда сипатталған алгоритмдердің басқа басқару заңдылықтарына сүйенген нұсқалары пайда бола бастады. Мысалы, Linux жүйесінде CUBIC TCP (На және басқалар, 2008), ал Windows жүйесіне Compound TCP нұсқасы қосылған (Tan және басқалар, 2006).

Күрделі екі жаңартулар TCP жүзеге асырылуына қатысты. Біріншіден, бұл алгоритмнің күрделілігі – растаулар көшірмесі бойынша қандай дестелердің жоғалғандығын, қандай дестелердің жоғалмағандығын анықтау керек. Жинақтау растауының нөмірінде мұндай ақпарат жоқ. Ең қарапайым шешім – сәтті қабылданған байттардың үшке дейінгі диапазоны бар **іріктеп растауды**

(SACK, Selective ACKnowledgement) пайдалану болып табылды. Мұндай ақпарат жөнелтушіге қандай дестелерді қайталап жөнелту керек екенін нақтырақ анықтауға және әлі жеткізілмеген дестелерді бақылауға мүмкіндік береді.

Байланыс орнату кезінде жөнелтуші және қабылдаушы бір-біріне іріктелген растаулармен жұмыс істей алатындықтарын көрсетіп, *SACK premitted* параметрін жөнелтеді. Іріктеп растау қосылған кезде деректер алмасу *6.42-суретте* көрсетілгендей жүреді. Қабылдаушы *Растау нөмірі* өрісін әдеттегідей пайдаланады – соңғы қабылданған байттың ретік нөмірі бойынша жинақтап растау. Үшінші деста ретсіз келген кезде (себебі 2 деста жоғалған) ол 1-деста үшін жинақталған растаумен бірге (көшірме) қабылданған деректер үшін *SACK option* жөнелтіледі. *SACK option*-да растау нөмірінен кейін қабылданған байттар диапазоны жайлы ақпарат орналасады. Бірінші диапазон – растау көшірмесі тиесілі деста. Келесі диапазон, егер ол бар болса, келесі блокқа қатысты. Әдетте, үш диапазоннан артық қолданылмайды. Алтыншы деста келген кезде іріктеп растау екі диапазонды пайдаланады: біріншісі – 3- және 4-дестелердің алынғандығын, ал екіншісі – 6-дестенің алынғандығын (1-дестеге дейін алынғандарға қоса) көрсетеді. Барлық қабылданған *SACK option* сүйене отырып, жөнелтуші қандай дестелерді қайта жөнелту керек екенін шешеді. Біздің жағдайда 2- және 5-дестені алған дұрыс.



6.42-сурет. Іріктеп растау

Іріктеп растауда ұсынбалы ақпарат жазылған. Іс жүзінде растау көшірмелері бойынша жоғалған дестелерді анықтау және асыра жүктелу терезесін өзгерту жоғарыда сипатталғандай жүреді. Дегенмен, іріктеп растау бірнеше деста бірден жоғалған кезде жағдайды қалпына келтіруде көмектеседі, себебі олардың көмегімен жөнелту қандай дестелердің адресатқа жетпегенін анықтайды. Іріктеп растау барлық жерде бірдей қолданылмайды. Ол жайлы RFC 2883 құжатында, ал іріктеп растау көмегімен TCP-де басқаруды бақылау RFC 3517 құжатында сипатталған.

Екінші жақсарту – асыра жүктелудің қосымша сигналы ретінде асыра жүктелу жайлы анық ескертуді (ECN, Explicit congestion Notification) пайдалану. Асыра жүктелу жайлы анық ескерту хосттарға асыра жүктелу жайлы хабарлауға мүмкіндік беретін IP-деңгейінің механизмі (*5.3.4-бөлімді қараңыз*). Оның көмегімен TCP-қабылдаушы IP-дан қанығу жайлы сигнал алады.

Егер байланыс орнату кезінде жөнелтуші және қабылдаушы бір-біріне *ECE* және *CWR* биттерінің көмегімен асыра жүктелу жайлы анық ескерту ала алатынын хабарласа, онда ол TCP-байланысқа қосылады. Бұл жағдайда әр десте тақырыбында TCP-сегментпен бірге десте асыра жүктелу жайлы анық ескерту бере алатындығы жазылады. Сондықтан асыра жүктелу қаупі туған кезде, осындай ескерпені қолдайтын маршруттауыштар асыра жүктелу нақты орын алған кезде дестелерді жойғанның орнына, сәйкес жалаушалары бар дестелерге асыра жүктелу жайлы сигналды орналастырады.

Егер қабылданған қандайда бір дестеде асыра жүктелу жайлы анық ескертпе бар болса, TCP-қабылдаушы оны біліп, *ECE* (ECN-жаңғырық) жалаушасының көмегімен жөнелтушіге асыра жүктелу жайлы хабарлайды. Жөнелтуші бұл сигналдың қабылданғандығын *CWR* жалаушасының (Асыра жүктелу терезесі төмендетілді) көмегімен растайды.

Бұндай сигналдарға және дестенің жоғалуына жөнелтуші дәл әрекет етеді. Енді нәтиже жақсарады: бірде-бір десте жоғалмаса да асыра жүктелу анықталды. Асыра жүктелу жайлы анық ескертпелер RFC 3168 құжатында сипатталған. Оларға хосттардың да, маршруттауыштардың да қолдауы қажет болғандықтан Интернетте олар кеңінен қолданылмайды.

TCP-дегі асыра жүктелуді бақылаудың толық механизмдері және олардың жұмысы RFC 5681 құжатында сипатталған.

6.5.11. TCP болашағы

Интернеттің «жегін аты» TCP-ді көптеген қосымшалар қолданады. Уақыт өте құрастырушылар әртүрлі желілерде жоғары өнімділікке қол жеткізу үшін бұл хаттаманың түрін өзгертіп, қосымшалар енгізеді. Қазіргі кезде хаттаманың көптеген версиялары бар, олардың әрқайсысы осы жерде сипатталған классикалық алгоритмге жаңа бір затты қосады: бұл әсіресе, асыра жүктелуді бақылауға және желілік шабуылдарға төтеп беруге қатысты. TCP-дің Интернетпен параллель даму ықтималдығы жоғары. Бұнда біз екі жеке сұрақты қарастырамыз.

Біріншіден, TCP хаттамасы көптеген қосымшалар үшін өзекті транспорттық семантика сұрақтарын шешпейді. Мысалы, кейбір қосымшалар өздері жөнелткен мәлімдеме немесе жазбалар шекарасының сақталғанын қалайды. Басқа қосымшалар өзара байланысты мәлімдемелермен жұмыс істегісі келеді – мысалы, бір серверден бірнеше объектіні жүктейтін веб-браузер тәрізді. Кейбір қосымшаларға желілік жолдарды басқарудың қосымша мүмкіндіктері қажет. TCP өз сокеттерінің стандартты интерфейсімен бұл талаптарды орындауға мүмкіндігі жоқ. Нәтижесінде TCP шеше алмайтын мәселені әр қосымша өз бетінше шешуге мәжбүр. Бұл интерфейсі сәл өзгерген жаңа хаттамалардың пайда болуына әкелді. Олардың ішінде RFC 4960 сипатталған **SCTP (Stream Control Transmission Protocol – ағынды басқарып тасымалдау хаттамасы)** және **SST (Structured stream Transport – деректерді иерархиялық**

ағындық транспорттау) (Ford, 2007). Әдеттегідей, егер кімде-кім ғасырлар бойы тексеріліп келген механизмді өзгертуге ұсыныс жасаса: «Тұтынушылар жаңа мүмкіндіктерге қол жеткізгісі келеді» және «Егер ешнәрсе сынбаса, онда ешнәрсені жөндеудің де қажеті жоқ» дейтін қарсылас екі лагерь пайда болады.

Екінші сұрақ асыра жүктелуді бақылауға қатысты. Асыра жүктелуді бақылаудың түрлі механизмдерін және оларды жақсарту жолдарын егжей-тегжейлі қарастырғаннан кейін бұл сұрақ шешілген болып көрінуі мүмкін. Бұл олай емес. Жоғарыда айтылған асыра жүктелуді бақылау формасы өте танымал, асыра жүктелу сигналы ретінде дестенің жоғалуын пайдаланады. Өткізгіштік қабілеттілікті ара тәріздес схема көмегімен модельдеу кезінде Padhye және оның әріптестері (1998) жылдамдықты өсірген кезде дестелердің жоғалу жиілігі күрт төмендейтінін анықтаған. Айналма кідіріс 100 мс және дестe көлемі 1500 байт болған кезде 1 Гбит/с өткізгіштік қабілеттілікке қол жеткізу үшін әр 10 минут сайын бір дестe жоғалуы мүмкін. Бұл дестелердің 2×10-8 жоғалу жиілігіне сәйкес, ал бұл өте аз. Дестелердің жоғалуы өте сирек орын алады, сондықтан бұл көрсеткіш асыра жүктелудің жақсы сигналы бола алады, Басқа себептермен жоғалған дестелер саны (мысалы, тасымалдау қателігі 10-7 жиілігімен орын алады) бұл саннан оңай асып кетуі мүмкін, ал бұл өткізгіштік қабілеттіліктің төмендеуіне әкеледі.

Осы уақытқа дейін бұл өзекті емес болып көрінген, алайда, желілердің жылдам бола бастауына байланысты құрастырушылар асыра жүктелу мәселесіне қайта оралуда. Мүмкін деген нұсқалардың бірі – дестенің жоғалуы мүлдем есепке алынбайтын жаңа алгоритмді пайдалану. Бірнеше мысал *6.2-бөлімде* келтірілген болатын. Асыра жүктелу сигналы ретінде, мысалы, FAST TCP-дегідей асыра жүктеме кезінде өсетін айналма кідірісті алуға болады (Wi және басқалар, 2006). Басқа амалдар да болуы мүмкін, қайсының жақсы екенін уақыт көрсетеді.

6.6. ӨНІМДІЛІК СҰРАҚТАРЫ

Өнімділік сұрақтары компьютерлік желілерде маңызды рөл атқарады. Жүздеген немесе мыңдаған компьютерлер өзара байланысқан болған кезде, олардың өзара әрекеттесуі өте күрделі болады және болжалмайтын салдарларға әкелуі мүмкін. Бұл қиындық салдары – төмен өнімділік және көптеген жағдайда оның себебін анықтау мүмкін емес. Келесі бөлімдерде біз желі өнімділігімен байланысты көптеген сұрақтарды қарастырып, мәселелер шеңберін анықтап, оларды шешу тәсілдерін талқылаймыз.

Өкінішке орай, желі өнімділігі түсінігі – бұл ғылымнан гөрі өнер. Қандай да бір іс жүзіндегі қолданыс мүмкін деген теориялық база өте аз. Біздің қолымыздан келетіні – ұзақ тәжірибе негізінде алынған бірнеше тәсілдерді ұсыну, сонымен бірге іс жүзіндегі бірнеше нақты мысалдар келтіру. Біз бұл пікірталасты, кейжерде TCP-ден мысалдар келтіріп бейнелеу үшін, әдейі TCP желілеріндегі транспорттық деңгей оқылатын сәтке қалдырдық.

Келесі бөлімдерде біз желі өнімділігінің:

1. Өнімділіктің төмендеу себептері;
2. Желі өнімділігін өлшеу;
3. Жылдам желілер үшін хосттарды жобалау;
4. Сегменттерді жылдам өңдеу;
5. Тақырыптарды тығыздау;
6. Өткізгіштік қабілеттілігі жоғары созылыңқы желілер хаттамасы (long fat networks) сияқты негізгі аспектілерін қарастырамыз.

Бұл аспектілер өнімділікті хосттарда және деректер желі ішінде қозғалғанда орындалатын операциялар, сонымен бірге желі мөлшері және жылдамдығы тұрғысынан қарастыруға мүмкіндік береді.

6.6.1. Компьютерлік желілерде өнімділіктің төмендеу себептері

Өнімділіктің төмендеуінің кейбір түрлері бос ресурстардың уақытша болмауы салдарынан болады. Егер маршруттауыштарға трафик ол өңдей алатын деңгейден көп келсе, кептеліс туындап, өнімділік күрт түседі. Асыра жүктелу мәселесі осы және алдыңғы тарауларда егжей-тегжейлі қарастырылды.

Өнімділік – ресурстардың құрылымдық үйлесімсіздігі орын алған кезде де төмендейді. Мысалы, егер гигабайттық байланыс торабы өнімділігі төмен компьютерге қосылған болса, онда орталық процессор келген дестелерді жылдам өңдей алмайды, бұл кейбір дестелердің жоғалуына әкеледі. Ерте ме кеш пе бұл дестелер қайта жөнелтіледі, бұл кідірістің ұлғаюына, өткізгіштік қабілеттілікті өнімсіз пайдалануға және жалпы өнімділіктің төмендеуіне әкеледі.

Асыра жүктелу синхронды орын алуы мүмкін. Мысалы, егер сегментте қате параметр көрсетілсе (мысалы, тағайындалған порт нөмірі), көп жайдайда қабылдаушы қателік жайлы мәлімдемені кері жөнелтеді. Енді қате сегмент кең тарату тәсілімен 1000 машинаға жөнелтілген жағдайда не болатынын көрейік. Әр машина қателік жайлы мәлімдемені кері жөнелтуі мүмкін. Нәтижесінде орын алған **кең таралымды дауыл (broadcast storm)** желінің қалыпты жұмысын ұзаққа тоқтатуы мүмкін. UDP хаттамасы, ICMP хаттамасы хосттар UDP кеңтаралымды сегменттеріне қателік жайлы мәлімдеме жөнелтуден өзөздерін тоқтататындай етіп өзгертілгенше, осындай мәселеден зардап шеккен болатын. Себебі олардың өткізгіштік қабілеттілігі шектеулі.

Екінші мысал, синхронды асыра жүктелу электр энергиясының өшуімен байланысты болуы мүмкін. Қоректену қайта іске қосылған кезде барлық машиналар қайта жүктеле бастайды. Жүктелудің әдеттегі тізбегі, өзінің нақты адресін білу үшін қандайда бір DHCP-серверге (хосттың динамикалық конфигурация сервері), ал сонан кейін операциялық жүйенің көшірмесін алу үшін

файлдық серверге жүгінуін талап етуі мүмкін. Егер жүздеген машиналар серверге бір сәтте жүгінетін болса ол барлығына бірдей қызмет көрсете алмайды.

Тіпті, синхронды асыра жүктелу жоқ және ресурстар жеткілікті болған кезде де өнімділік жүйелік баптаулар қателігінен төмендеуі мүмкін. Мысалы, машинаның процессоры қуатты және жадысы жеткілікті, бірақ буферге жады жеткіліксіз бөлінуі мүмкін. Бұл жағдайда деректер ағынын басқару сегменттерді қабылдауды баяулатып, өнімділікті шектейді. Мұндай мәселе Интернет жылдам бола бастағылы, ал ағынды басқару терезесі бекітілген (64 Кбайт) болғалы көптеген TCP-байланыстар үшін орын алды.

Сонымен бірге өнімділікке күту таймері мәнінің дұрыс орнатылмауы да әсер етеді. Сегмент жөнелтілген кезде, әдетте, модуль жоғалып кететін жағдай үшін таймер іске қосылады. Растауды күту аралығын тым кіші етіп таңдау, сегменттерді артық қайта тасымалдауға әкеледі. Егер күту кезеңін үлкен етіп таңдаса, онда бұл сегмент жоғалған жағдайда кідірістің ұлғаюына әкеледі. Бапталмалы параметрлерге, сонымен бірге растауды жөнелтуге бағыттас деректер модулін күту кезеңі және растау жоқ болған жағдайда қайта тасымалдау талпыныстар саны да жатады.

Нақты уақыт режимінде жұмыс істейтін (мысалы, аудио және бейнені қалыпқа келтіретін) қосымшаларға қатысты тағы бір мәселе – джиттер. Жақсы өнімділік үшін оларға жақсы орташа өткізгіштік қабілеттілік жеткіліксіз. Мұндай қосымшаларға төмен кідіріс қажет. Ал ол үшін жүктеме желіге тиімді таратылуы, сонымен бірге арналық және желілік деңгейде қызмет көрсету сапасын қолдау керек.

6.6.2. Желі өнімділігін өлшеу

Желі жұмысының сапасы нашар болған кезде, оның тұтынушылары желілік операторлардан жақсартуларды талап етіп арызданады. Желі өнімділігін жақсарту үшін, операторлар алдымен мәселенің мәнін дәл анықтаулары керек. Желінің ағымдағы қалып-күйін анықтау үшін операторлар өлшеу жүргізулері тиіс. Бұл бөлімде біз желі өнімділігін өлшеу сұрақтарын қарастырамыз. Төменде келтірілетін талқылаулар Могол (Mogul, 1993) жұмыстарына негізделген.

Өлшеуді түрлі тәсілдермен және көптеген орындарды (физикалық түрде де, хаттамалар стегінде де) жүргізуге болады. Өлшеудің кең таралған түрі – қандай да бір екпінділік бар кезде таймерді іске қосу және осы екпінділік ұзақтығын өлшеу. Мысалы, негізгі өлшеулердің бірі, жөнелтушінің жіберген сегментіне растау алуға кететін уақыт ұзақтығын өлшеу. Басқа өлшеулер, қандайда бір оқиға жиілігі есепке алынатын (мысалы, жоғалған сегменттер саны) санауыш көмегімен жүргізіледі. Соңында, белгілі бір уақыт кезеңінде өңделген байттар саны тәрізді сандық көрсеткіштер өлшенеді.

Желі өнімділігін және басқа параметрлерді өлшеу процедурасының көптеген күтпеген кедергілері бар. Төменде біз оларды атап өтеміз. Желі өнімділігін өлшеудің кез келген талпынысында бұл қателіктерді жіте айналып өтуге тырысу қажет.

Іріктеудің жеткілікті түрде үлкен екеніне көз жеткізіңіз

Қандайда бір параметрді бір рет өлшеумен шектелмеңіз – мысалы, бір сегментті жөнелтуге қажет уақытты. Айталық, өлшеуді миллион рет жүргізіп, орта мәнін табыңыз. Бастапқы әсер, мысалы, 802.16 NIC немесе кабельдік модемнің жұмыссыз тұрып қалғаннан кейін қордағы өткізгіштік қабілеттілікті алады, бұл бірінші сегментті жөнелтуді баяулатады, ал оны кезекке қою мәндер шашыраңқылығын өсіреді. Іріктеу неғұрлым көп болса, орта мән және оның орта квадраттық ауытқу дәлдігі соғұрлым жоғары болады. Қателікті статистиканың стандартты формулаларымен есептеуге болады.

Іріктеудің қайталап таныстырмалы екеніне көз жеткізіңіз

Мінсіз жағдайда, параметрлердің миллион өлшеулер тізбегін тәуліктің әртүрлі уақытында және аптаның әртүрлі күндерінде қайталаған дұрыс. Бұл жүйенің әртүрлі жүктелуінің өлшенетін параметрге әсерін анықтауға мүмкіндік береді. Асыра жүктелуді, асыра жүктелу жоқ кезде өлшеудің еш пайдасы жоқ. Кейде өлшеулер нәтижесі бір қарағанда оғаш болып көрінуі мүмкін, мысалы, желіде 10, 11, 13 және 14-сағаттарда күрделі кептелістердің болуы, ал шаңқай түсте болмауы (тұтынушылар түстенетін кезде).

Сымсыз желілер жағдайында сигналдың таралуына орналасу орны маңызды рөл атқарады. Тіпті, өлшеу жүргізілетін түйін сымсыз клиент жанында орна-ласса да, пайдаланылатын антенналардың әртүрлігінен ол клиентке қолжетімді кейбір дестелерді көрмеуі мүмкін. Өлшеуді сымсыз клиент хостында жүргізген дұрыс. Егер бұл мүмкін болмаса, бірнеше бақылау пункттерін таңдап алған дұрыс, сөйтіп неғұрлым толық көрініс алуға болады (Mahajan және басқалар, 2006).

Кэштеу сіздің өлшеулеріңізді қатты бұрмалауы мүмкін

Егер хаттамалар кэштеу механизмдерін қолданатын болса, өлшеулерді қайталау күтпеген нәтиже беруі мүмкін. Мысалы, веб-параққа жолданым немесе DNS атты қарау (IP-адресті табу үшін) алғашқыда желі арқылы орындалуы, ал кейіннен дестелер тасымалы жүргізілмейтін кэшке жолданым арқылы орындалуы мүмкін. Мұндай өлшеулер нәтижесінің еш пайдасы жоқ (тек сіз кэш өнімділігін өлшегіңіз келмесе).

Дестелерді буферлеу де дәл осындай әсер беруі мүмкін. Өнімділікті өлшейтін бір танымал TCP/IP-программа, осы физикалық торап үшін UDP хаттамасы ең үлкен ұйғарымды өнімділіктен әлдеқайда жоғары өнімділікке қол жеткізе алады деп мәлімдеумен атағы шыққан болатын. Бұл қалай болды? UDP-ға жүгіну әдетте, жүйе ядросы мәлімдемені қабылдағаннан кейін басқаруды бірден қайтарады және тасымалдауға кезекке қойылады. Буфер көлемі жет-

кілікті болған жағдайда UDP-ға 1000 жолданымды орындау уақыты барлық деректердің осы уақыт ішінде торапқа жөнелтілгендігін білдірмейді. Олардың көп бөлігі әлі де ядро буферінде.

Деректердің кэштелуі және буферленуінің қалай жүргізілетінін түсінетініңізге нық сенімді болуыңыз керек.

Сіздің тестілеу уақытыңызда ешқандай күтпеген жағдайдың орын алмағанына көз жеткізіңіз

Егер сіз желіні тестілеп жатқан кезде қандайда бір зерттеуші желі арқылы бейнеконференция өткізсе, онда нәтиже әдеттегі күндері жүргізілген өлшеулерден ерекше болуы мүмкін. Барлық жүктемені өзіңіз жасап, тесті бос жүйеде жүргізген дұрыс. Дегенмен, бұл жағдайда қателесу ғажап емес.Сіз түнгі 3 кезінде желіні ешкім пайдаланбайды деп шешуіңіз мүмкін, бірақ нақты осы кезде қорға автоматты көшірме жасау программасы барлық қатты дискілерді магниттік лентаға тығыздау жұмысын бастауы мүмкін. Бұдан басқа, дәл осы уақытта жердің басқа бөлігінде, басқа уақыттық аймақтағы тұтынушы, сіздің тамаша веб-парақтарыңызды қарап, өте күшті трафик құрастыруы мүмкін.

Сымсыз желілерде ерекше қиындықтар туындайды, себебі бұл жағдайда түрлі кедергілерді сигналдан ажырату қиын. Тіпті, жақын арада екпінді сымсыз желі жоқ болса да, кімде кімнің қысқа толқынды пеште попкорн дайындауынан кедергі туындап, 802.11 өнімділігі төмендеп кетуі мүмкін. Сондықтан болжалмаған жағдайдың туындағанын білу үшін, желінің жалпы активтілігін қадағалау керек.

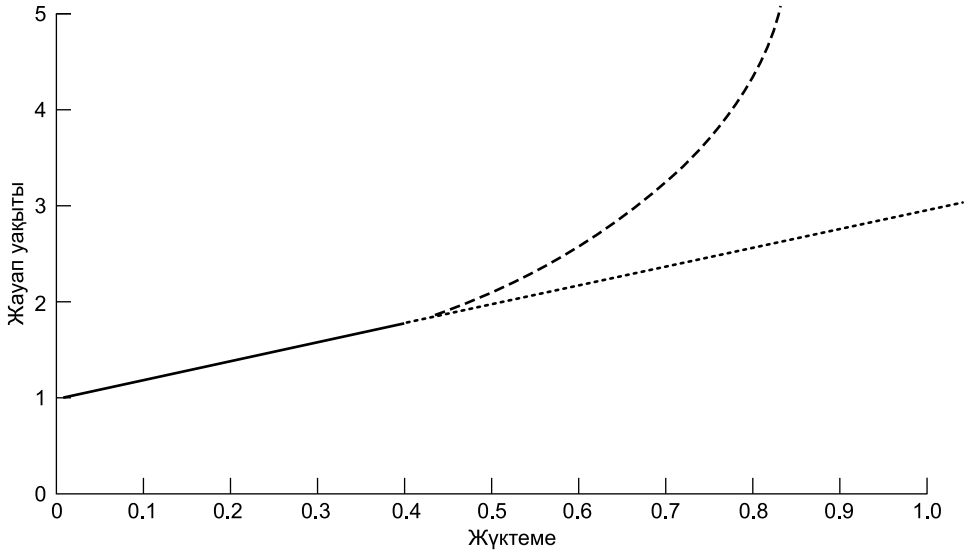
Сирек шкаласы бар сағатты пайдаланғанда ұқыпты болыңыз

Компьютер сағаты қандайда бір санауышқа тең уақыт кезеңі арасында бірлікті қосып отырып жұмыс істейді. Мысалы, миллисекундтық таймер санауыш мәнін 1мс сайын 1-ге өсіреді. Ұзақтығы 1 мс уақыттқа созылатын оқиға үшін мұндай таймерді пайдалану мүмкін, бірақ құнттылықты талап етеді. Әрине, кейбір компьютерлер әлдеқайда дәл сағатты қолданады, бірақ таймер қадамы қандай болса да, одан да аз уақытта орындалатын оқиға табылады. Сонымен бірге сағат дәлдігі қайтаратын мән дәлдігімен әркез сәйкес келе бермейтінін есте сақтау керек.

Мысалы, бір сегментті жөнелтуге қажет уақытты өлшеу үшін, транспорттық деңгей программасынан кірер және шығар кездегі жүйелік сағат көрсетілімін (айталық, миллисекундпен) алу керек. Егер бір сегментті жөнелтуге қажет уақыт 300 мкс тең болса, өлшенетін шама не нөлге, не 1 мс болады, ал бұл дұрыс емес. Дегенмен де, егер өлшеуді миллион рет қайталап, орта мәнін табатын болсаңыз, ол мәннің нақты іс жүзіндегі мәннен кем дегенде, 1 мкс айырмашылығы болады.

Нәтижені таратудан сақ болыңыз

Айталық, сіз желідегі жүктемені 0-ден (қарапайым) 0,4-ке дейін (өткізгіштік қабілеттіліктің 40%) бір затты өлшейсіз (мысалы, қашықтықтағы хосттың жауап уақыты) делік. 802.11 желісі бойынша VoIP дестені жөнелткенде жауап уақыты *6.43-суретте* қалың сызықпен көрсетілгендей болсын. Алынған қисықты сызықты түрде (үзік сызық) тарату қызығарлықтай болып көрінуі мүмкін. Алайда, іс жүзінде жалпылама қызмет көрсету теориясында көптеген параметрлер көбейткіш ретінде қосылады, мұндағы, ρ – жүктеме, сондықтан шын мәніндегі қисық сызықшалы сызықпен көрсетілген гиперболаға ұқсас болады, әсіресе, жүктеме үлкен болғанда. Сонымен, үлкен жүктеме кезінде қыза түсетін бәсекелестік әсеріне назар аудару керек.



6.43-сурет. Жауап уақытының жүктемеге тәуелділігі

6.6.3. Жылдам желілер үшін хосттарды жобалау

Өлшеулер және баптаулар жиі желінің өнімділігін едәуір жақсартуға көмектеседі. Алайда, олар ешуақытта жақсы құрастырылған жобаны алмастыра алмайды. Нашар жобаланған желі тек белгілі бір деңгейге дейін ғана жақсартылуы мүмкін. Өнімділікті әрі қарай жақсарту үшін басынан бастап қайта жасауға тура келеді.

Бұл бөлімде біз хосттардағы желілік хаттамаларды программалық жүзеге асырудың бірнеше эмпирикалық ережелерін қарастырамыз. Тәжірибе бұның әдетте желі өздігінен жылдам жұмыс істейтін жағдайда өнімділіктің шектеуші фактор болып келетінін көрсетеді. Бұл екі себеппен орын алады. Біріншіден,

желілік карталар (NIC) және маршруттауыштар желі жылдамдығымен жұмыс істей алатындай етіп құрастырылады. Демек, дестелерді келіп түскен жылдамдығымен өңдей алады. Екіншіден, бізді қосымшаларға қолжетімді өнімділік қызықтырады. Ал ол арна қуаттылығына байланысты емес, желілік және транспорттық деңгейлер жұмысының нәтижесі болып саналатын, өткізгіштік қабілеттілік және кідіріске байланысты есептеледі.

Программалық жабдықтамалар себебінен үстеме шығындарды төмендету, өнімділікті, өткізгіштік қабілеттілікті жоғарылату және кідірісті төмендету арқылы жақсартады. Ол сонымен қоса, мобильді компьютерлер үшін өзекті, желі бойымен деректерді тасымалдауға қажет электр энергия шығындарын төмендетуге мүмкіндік береді. Бұл олардың көбі желі құрастырушыларға көптен бері белгілі. Алғаш рет оны анық түрде Могол жазған болатын (Mogul, 1993). Біздің әңгіметіміз осы кітаппен тығыз байланысты. Осы тақырып бойынша басқа ақпарат көзі Metcalf (1993) болып саналады.

Орталық процессор жылдамдығы желі жылдамдығынан маңызды

Ұзақ уақыттық тәжірибелер, барлық желілерді операциялық жүйе және хаттама үстеме шығыны желілік операцияның негізгі кідіріс уақытын құрайтынын көрсетті. Мысалы, қашықтықтағы процедураны шақырудың теориялық ең кіші уақыты (RPC, Remote Procedure Call) қуаттылығы 1 Гбит/с Ethernet желісінде 1 мкс құрайды, бұл ең кіші жауап (512 байт) келетін ең кіші сұранысқа (512 байт) сәйкес келеді. Іс жүзінде, қашықтықтағы процедураны шақырғанда программалық жабдықтама салдарынан туындайтын үстеме шығынды қандай да бір төмендету үлкен жетістік болып саналады. Бұл өте сирек орын алады.

Сәйкесінше, гигабиттік тораппен жұмыс істегенде негізгі мәселе – биттерді тұтынушы буферінен торапқа жылдам тасымалдау, сонымен бірге қабылдаушы оны қандай жылдамдықпен келсе, сол жылдамдықпен өңдей алуы. Процессор өнімділігін және жады жылдамдығын екі еселеу жиі өткізгіштік қабілеттілікті екі еселеуге әкеледі. Ал, жіңішке орын хост болған жағдайда тораптың өткізгіштік қабілеттілігін екі еселеу ешқандай әсер бермейді.

Үстеме шығындарды азайту үшін дестелер санын азайтыңыз

Әр сегменттің белгілі бір мөлшердегі үстеме шығыны (мысалы, тақырып) және деректері (мысалы, пайдалы жүктеме) бар. Бұл екі компонентте өткізгіштік қабілеттілікті қажет етеді. Сонымен бірге олардың әрқайсысын өңдеу қажет (мысалы, тақырыпты өңдеу және бақылау қосындысын есептеу). Бір миллион байт жөнелткен кезде процессордың өңдеуге жұмсалатын байттық уақыт шығыны сегмент көлеміне байланысты емес. Алайда, 128-байттық сегментті пайдаланған кезде тақырыпты өңдеу шығыны, 4 байттық сегментке қарағанда 32-рет жоғары болады. Бұл шығындар жылдам өсіп отырады.

Төменгі деңгейлер үстеме шығыны бұл әсерді күшейте түседі. Хост активті режимде жұмыс істеп тұрса, онда әрбір дестенің келуі әркез жаңа үзілісті шақырады. Қазіргі заманғы конвейрлік процессорларда әр үзіліс процессорлық конвейрдің жұмысын бұзады, кэш жұмысының тиімділігін төмендетеді, жадыны басқару мәнмәтінін өзгертуді талап етеді, көшулерді болжау кестесін жояды және стекте процессор регистрларының біршамасын сақтауды қажет етеді. Сөйтіп, жөнелтілетін сегменттерді *n* санына азайту, жалпы алғанда үзілістер санын және үстеме шығындарды *n* рет төмендетуге мүмкіндік береді.

Компьютер де адам тәрізді көп есептілікті нашар игереді деуге болар еді. Осы себеп – негізінен фрагменттеуді қажет етпейтін, ең үлкен көлемдегі MTU-дестелерді жөнелтуге итермелейді. Нагаль алгоритмі және Кларк тәсілі тәрізді механизмдері кіші дестелерді жөнелту талпынысынан қашу болып келеді.

Деректермен орындалатын операциялар санын азайтыңыздар

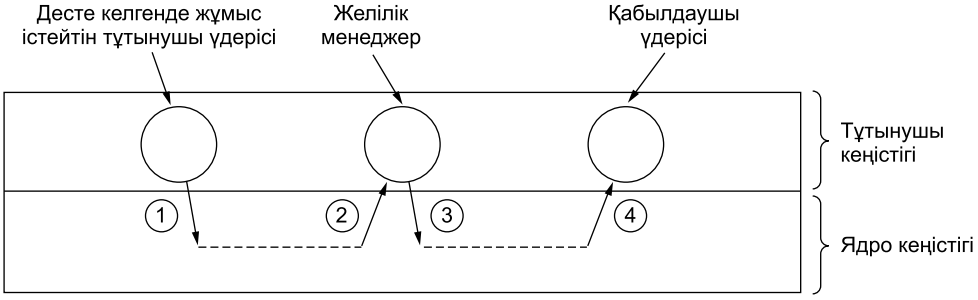
Көпдеңгейлі хаттамалар стегін жүзеге асырудың еең қарапайым тәсілі – әр деңгей үшін бір модульді пайдалану. Өкінішке орай, бұл деректерді көптеген көшірулерге әкеледі (немесе, кем дегенде, оларға қол жеткізуге). Мысалы, дестені желілік карта қабылдағаннан кейін ол әдетте, ядроның жүйелік буферіне көшіріледі. Ол жерден желілік деңгей өңдеу үшін желілік деңгейдің буферіне көшіріледі, сонан кейін – транспорттық деңгей өңдеу үшін транспорттық деңгей буферіне, соңында қабылдаушы қосымшаға жеткізіледі. Қабылданған деста тағайындалған орынға жеткенше ондағы сегмент үш немесе төрт рет көшіріледі.

Мұндай көшіру өнімділікті күрт төмендетуі мүмкін, себебі жадымен орындалатын операция тек ішкі регистрлермен орындалатын операцияға қарағанда ондаған есе баяу болып келеді. Мысалы, егер нұсқаулардың 20%-ы іс жүзінде жадымен байланысты (яғни, деректер кэште жоқ) болса, ал бұлай болуы ықтимал және болжамалы сан, онда кіріс дестелерді өңдеу кезінде нұсқауды өңдеудің орташа уақыты 2,8 есе (0,8×1+0,2×10) төмендейді. Аппараттық жақсартулар бұл жерде көмектеспейді. Мәселе операциялық жүйе орындайтын көшіру операциясының тым көптігінде.

Дұрыс жұмыс істейтін операциялық жүйе, әртүрлі деңгейлерде орындалатын өңдеу үдерісін біріктіріп, көшіру операцияларының санын азайтуға тырысады. Мысалы, TCP және IP әдетте, бірге жұмыс істейді («TCP/IP»), сондықтан өңдеу желілік деңгейден транспорттық деңгейге өткен кезде дестенің пайдалы жүктемесін көшірудің қажеті жоқ. Тағы бір танымал айла – бір айналымда бірден бірнеше операцияны орындау. Мысалы, бақылау қосындысын есептеу жиі деректерді көшіру кезінде орындалады (іс жүзінде қажет болғанда) және жаңа мән соңына қосылады.

Мәнмәтіннің алмасу санын азайтыңыз

Мәнмәтінін алмастырудың (мысалы, ядро режимінен тұтынушы режиміне) біршама жағымсыз қасиеттері бар, мұнда ол үзіліске ұқсас. Ең жағымсызы – кэштің ішіндегісі жоғалады. Мәнмәтінді алмастыруды, деректерді қажет саны жинақталғанша ішкі буферге жөнелтетін кітапханалық процедуралар көмегімен азайтуға болады. Сәйкесінше қабылдаушы бетте кішігірім сегменттер жинақталып, тұтынушыға бірден жөнелтіледі, бұл мәнмәтіннің алмасу санын төмендетеді.



6.44-сурет. Желілік менеджер тұтынушы кеңістігінде болған жағдайда бір дестені өңдеу үшін мәнмәтіннің төрт алмасуы

Ең жақсы жағдайда, келген десте бір алмасуды қажет етеді – ағымдағы тұтынушы үдерісінен ядро режиміне, ал сонан кейін бір алмасу – қабылдаушы үдеріске басқаруды ұсынып, оған келген деректерді беру. Өкінішке орай, көптеген операциялық жүйелерде мәнмәтіннің тағы бір алмасуы орын алады. Мысалы, егер желілік менеджер тұтынушы кеңістігінде жеке үдеріс ретінде жұмыс істейтін болса, онда дестенің келуі басқаруды тұтынушы үдерісінен ядроға, ал сонан кейін ядродан желілік менеджерге, одан қайта ядроға және соңында, ядродан қабылдаушы үдеріске алмастыруды қажет етеді. Мәнмәтіннің бұл алмасулары *6.44-суретте* көрсетілген. Әр десте қабылданған сайын орындалатын барлық алмасулар орталық процессордың уақытын қатты шығындайды және желі өнімділігін төмендетеді.

Орын алған асыра жүктелумен күрескенше, оны болдырмауға тырысыңыз

«Ауырып емделгеннен, ауырудың алдын алған дұрыс» дейтін ескі мақал, желідегі асыра жүктелумен күресу жағдайында да әділетті. Желіде кептеліс орын алған кезде дестелер жоғалады, өткізгіштік қабілеттілік бекерге шығындалады, кідіріс ұлғаяды және т.с.с. Бұл шығынның барлығы керексіз. Асыра жүктелуден кейін қалыпқа келу үдерісі шыдамдылық пен уақытты қажет етеді. Асыра жүктелуді болдырмаудың әлдеқайда тиімді стратегиясы аурудың

алдын алатын егу тәрізді – бұл біршама жағымсыз, бірақ мүмкін деген үлкен келеңсіздіктерден қорғайды.

Тайм-ауттардан аулақ болыңыз

Таймерлер желіде қажет, алайда, оларды орынды пайдаланып, тайм-ауттар санын азайтуға тырысу керек. Таймер іске қосылған кезде әдетте, қандай да бір іс-әрекет қайталанады. Егер іс әрекетті қайталау қажет болса, оны орындау керек, алайда, қайталау аса қажет болсама, онда ол бейберекеттік.

Артық жұмыстан аулақ болу үшін, күту кезеңін сәл артығымен алу керек. Сегмент жоғалған жағдайда тым кеш іске қосылатын таймер кідірісті біршама ұлғайтады (ықтималдығы төмен). Уақытынан ерте іске қосылатын таймер – процессор уақытын, өткізгіштік қабілеттілікті бекерге шығындайды және ондағы маршруттауыштарға түсетін жүктемені бекерге ұлғайтуы мүмкін.

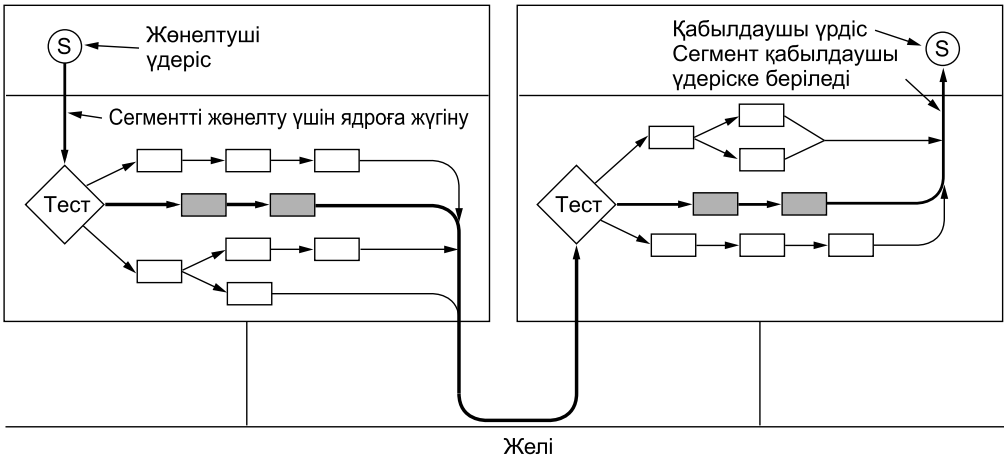
6.6.4. Сегменттерді жылдам өңдеу

Енді, жалпы ережелер жайлы айтқаннан кейін, сегменттерді өңдеуді жылдамдататын бірнеше тәсілдерін қарастырайық. Бұл тақырып бойынша қосымша ақпаратты Clark және басқалар (1989), Chase және басқалар (2001) басылымдарынан қараңыз.

Сегменттерді өңдеудегі үстеме шығындар екі компоненттен: тақырыпты өңдеуге жұмсалатын шығындардан және әр байтты өңдеуге жұмсалатын шығындардан тұрады. Шабуылды екі бағытта бірдей жүргізген дұрыс. Сегментті жылдам өңдеу кілті қалыпты жағдайды ерекшелеуге (деректерді бір бағытта тасымалдау) және бұл жағдайды жеке өңдеуге байланысты. Көптеген хаттамалар оқыс жағдай (мысалы, дестенің жоғалуы) болғанда не істеу керектігіне көп көңіл бөледі. Алайда, хаттамалар жылдам жұмыс істеу үшін, құрастырушы қалыпты жағдайдағы дестені өңдеу уақытын төмендетуі тиіс. Қателік орын алған кезде дестені өңдеу уақытын төмендету – бұл қосалқы мәселе.

ESTABLISHED қалып-күйіне көшу үшін арнайы сегменттер тізбегін жөнелту қажет болса да, қалып-күйге көшкеннен кейін екі жақтың бірі байланысты жапқанша сегменттерді өңдеудің қиындығы жоқ. Деректер жөнелту керек болған кезде *ESTABLISHED* қалып-күйіндегі болған жөнелтуші жақты қарастырайық. Қарапайымдылық үшін транспорттық ішкі жүйе ядрода орналасқан деп болжайық. Алайда, бұл ойларды транспорттық ішкі жүйе тұтынушы кеңістігінде орналасқан жағдайда да немесе кітапханалық процедуралар жөнелтуші үдерісте болған кезде де қолдануға болады. *6.45-суретте* жөнелтуші үдеріс, `SEND` базалық операциясын орындап, үзіліс үдерісін эмуляциялайды да басқаруды ядроға береді. Транспорттық ішкі жүйе алдымен оның қалыпты жағдайда екенін, яғни *ESTABLISHED* қалып-күйі орнатылғанын және екі жақтың да стандартты толық сегмент жөнелтіп

(*URGENT* жалаушасынсыз) байланысты жабуға талпыныс жасамағанын және қабылдаушыда терезе көлемі жеткілікті екенін тексереді. Егер осы шарттардың барлығы орындалса, ешқандай қосымша тексеру қажет емес және транспорттық ішкі жүйе алгоритмі жылдам жолды таңдай алады. Көп жағдайда осылай орындалады да.



6.45-сурет. Жөнелтушіден қабылдаушыға дейінгі жылдам жол қалың сызықпен көрсетілген. Осы жол бойындағы өңдеу қадамдары боялған тікбұрышпен көрсетілген

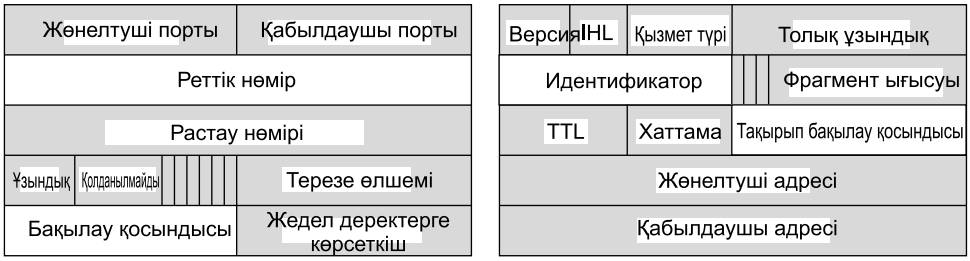
Қалыпты жағдайда көршілес сегменттердің тақырыптары көбіне, бірдей болып келеді. Осы факты пайдалану үшін транспорттық ішкі жүйе өз буферінде тақырып түптұлғасын сақтайды. Жылдам жолдың басында ол жеке сөздермен жаңа тақырып буферіне мүмкіндігінше жылдам көшіріледі. Сонан кейін, оның үстіне айырмашылығы бар өрістер жазылады. Жиірек бұл өрістер қалып-күй айнымалысынан шығарылады – мысалы, сегменттің келесі реттік нөмірі. Сонан кейін сегменттің толтырылған тақырыбына көрсеткіш және тұтынушы деректеріне көрсеткіш желілік деңгейге беріледі. Бұл жерде де сол стратегияны пайдалануға болады (*6.45-суретте* ол көрсетілмеген). Соңында, өңдеу нәтижесінде алынған дестені, желілік деңгей арналық деңгейге жөнелтуге береді.

Іс жүзінде бұл принциптің қалай жұмыс істейтінін көру үшін TCP/IP жағдайын қарастырайық. *6.46 а-суретінде* TCP-тақырып көрсетілген. Бірбағытталған ағындағы тақырыптың келесі сегменттер үшін бірдей өрістері қарайтылған. Тасымалдаушы транспорттық ішкі жүйенің орындайтыны, бұл түптұлға-тақырыптың бес сөзін кіріс буферге көшіру, реттік нөмір өрісін толтыру (бір сөзді көшіріп), бақылау қосындысын есептеп, ағымдағы реттік нөмір жазылған айнымалы мәнін бірге өсіру. Сонан кейін ол тақырып және деректерді стандартты ең үлкен сегментті жөнелтетін, арнайы IP-процедураға бере алады. Сонан кейін, IP-процедура бес сөзден тұратын (*6.46 ә-сурет*)

өз түптұлға-тақырыбын буферге көшіріп, *Идентификатор* өрісін толтырады және тақырыптың бақылау қосындысын есептейді. Десте енді жөнелтуге дайын.

Енді, *6.45-суреттегі* қабылдаушы беттің алынған дестені жылдам өңдеу жолын қарастырамыз. Бірінші қадам – келген сегмент үшін байланыс жазбасын анықтау. TCP хаттамасында байланыс хаттамасы хэш-кестеде сақталуы мүмкін. Оның кілті екі IP-адрестің және екі порттың қандай да бір қарапайым функциясы болуы мүмкін. Байланыс жазбасы анықталғаннан кейін, дұрыс жазба табылғанына сенімді болу үшін, порт нөмірі мен адресін тексеру керек.

Егер соңғы пайдаланылған жазбаға көрсеткіш орнатылса, онда алдымен соны тексеріп, қажет жазбаны іздеу үдерісін қосымша жылдамдатуға болады. Кларк 1989 жылы шыққан кітап авторларымен бірге осы сұрақты зерттеп, бұл жағдайда сәтті жолданым үлесі 90%-дан асады деген қорытындыға келді.



6.46-сурет. TCP-тақырып (а); IP-тақырып (ә); Екі жағдайда түптұлғадан өзгеріссіз алынған өрістер қарайтылған

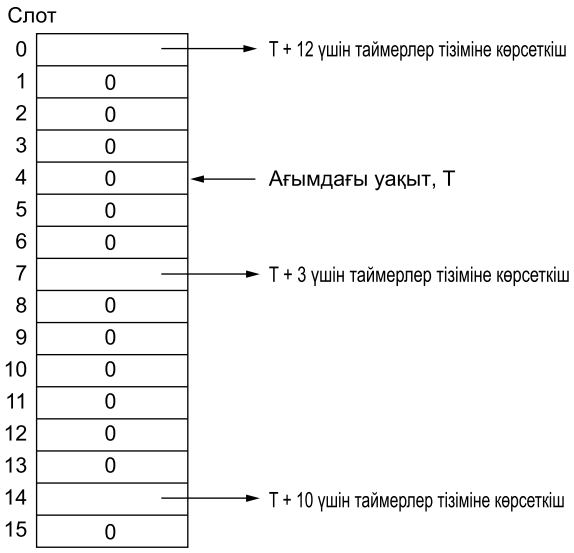
Сонан кейін сегмент адекваттылыққа тексеріледі: байланыс *ESTABLISHED* қалып-күйінде, екі жақта оны үзуге талпыныс жасамайды, сегмент толық, арнайы жалаушалар орнатылған және реттік нөмір күтілген нөмірге сәйкес келеді. Бұл тексеруді орындайтын программа біршама нұсқаулардан тұрады. Егер бұл шарттардың барлығы орындалатын болса, TCP-деңгейдің арнайы жылдам жол процедурасы шақырылады.

Жылдам жол процедурасы байланыс жазбасын жаңартады және деректерді тұтынушыға көшіреді. Көшіру кезінде ол бақылау қосындысын да есептейді, бұл деректерді өңдеу циклі санын азайтады. Егер бақылау қосындысы дұрыс болса, байланыс жазбасы жаңартылып, растау жөнелтіледі. Жеке процедура ретінде жүзеге асырылған, алдымен тақырыпты тексеріп, оның нақты күтілген тақырып екенін анықтайтын тәсіл **тақырыпты болжау (header prediction)** деп аталады. Ол TCP жүзеге асыруларының көбінде қолданылады. Бұл оңтайландыру тәсілін осы бөлімде сипатталған басқа тәсілдермен бірге пайдалану, TCP хаттамасына, желінің өзі жылдам болған жағдайда жергілікті жадыдан жадыға көшіру жылдамдығының 90%-на қолжеткізуге мүмкіндік береді.

Өнімділікті айтарлықтай жақсатуға болатын тағы екі аудан – бұл буферлерді басқару және таймерлерді басқару. Жоғарыда айтылғандай, буферлерді басқарған кезде артық көшірулерден аулақ болған дұрыс. Таймерлерді басқарған кезде олардың өте сирек іске қосылатындығын ескеру керек. Олар, сегменттің жоғалуы тәрізді оқыс жағдайды өңдеуге арналған, алайда, сегменттердің көбі және олардың растаулары сәтті жетеді, сондықтан күту уақытының өтіп кетуі сирек оқиға.

Әдетте, таймерлер программада іске қосылу уақыты бойынша сұрыпталған таймерлердің байланысқан тізімі ретінде жүзеге асырылады. Тізімнің бастапқы элементінде, күту уақытының аяқталуына дейінгі ең кіші уақыт кезеңі (тиктер – ticks) сақталған санауыш орналасқан. Тізімнің әрбір келесі элементінде, алдыңғы элементіне байланысты неше тиктен кейін нақты осы таймердің күту уақытының аяқталатынын көрсететін санауыш орналасқан. Мысалы, егер таймер 3, 10 және 12 тиктен кейін іске қосылатын болса, тізім санауыштарының мәні сәйкесінші 3, 7 және 2 болады.

Сағаттың әр тикі сайын бастапқы элемент санауышы бірге кемиді. Санауыштың мәні нөлге жеткен кезде онымен байланысты оқиға таймері өңделеді, ал келесі таймер тізімнің бастапқы элементі болады. Мұдай схемада таймерді қосу немесе алып тастау ресурстар шығынын қажет ететін операция, ал операциялардың орындалуы тізім ұзындығына пропорционал келеді.



6.47-сурет. Қайталану кестесі

Егер таймердің ең үлкен уақыты шектеулі және алдын ала белгілі болса, онда әлдеқайда тиімді тәсіл пайдалануға болады. Мұнда **қайталану кестесі (timing wheel)** деп аталатын массивті пайдалануға болады (*6.47-суретті* қараңыз). Әр ұя бір тикке сәйкес келеді. Суретте көрсетілген ағымдағы уақыт

– Т–4. Таймерлер 3, 10 және 12 тиктен кейін іске қосылуы керек. Егер жаңа таймер 7 тиктен кейін іске қосылу керек болса, онда тек 11-ұядағы көрсеткіш мәні жаңа таймерлер тізімін көрсетіледі. Сәйкесінше, егер Т+10 уақытына орнатылған таймерді өшіру қажет болса, 14-ұядағы көрсеткішті нөлге қою жеткілікті. Назар аударыңыздар, *6.47-суреттегі* массив Т+15-тен тысқары таймерлерді қолдай алмайды.

Ағымдағы уақыт көрсеткіші сағаттың әр тикі сайын қайталану кестесі бойымен алдыға қарай бір ұяға қозғалады. Егер ол көрсетіп тұрған ұя нөлдік болмаса, ұя көрсетіп тұрған тізімдегі таймерлердің барлығы өңделеді. Негізгі ойдың түрлі нұсқалары Varghese және Lauck (1989) басылымында талқыланады.

6.6.5. Тақырыптарды тығыздау

Біз жылдам желілерді ұзақ әңгімеледік. Қарастырылатын басқа да тақырыптар бар. Өткізгіштік қабілеттік шектелген сымсыз желілер және басқа типтегі желілер өнімділігі мәселесіне ауысайық. Егер программалық жабдықтамалар есебінен болатын шығындарды азайтсақ, мобильді компьютерлер тиімді жұмыс істейтін болады, бірақ бұл желілік арна жіңішке орын болып келетін жағдайда өнімділікті жақсартуға көмектеспейді.

Өткізгіштік қабілеттілікті тиімді пайдалану үшін хаттама тақырыбы және пайдалы жүктеме мүмкіндігінше аз битті алуы керек. Пайдалы жүктеме үшін бұны ақпаратты шағын кодтау арқылы қол жеткізуге болады – мысалы, суретті расторлық немесе тығыздалған құжаттық PDF тәрізді форматтан гөрі, JPEG форматында тасымалдаған дұрыс. Бұл үшін сонымен бірге, қолданбалы деңгейдегі кэштеу механизмі (веб-кэш тәрізді) және тасымалдау санын азайту қажет.

Хаттамалар тақырыбы жайлы не белгілі? Сымсыз желілерде арналық деңгейде тақырып әдетте аз орын алады, себебі олар о бастан төмен өткізгіштік қабілеттілікке есептелген. Мәселен, 802.16 ұзын адрестер орнына байланыстың қысқа идентификаторы пайдаланылады. Алайда, жоғары деңгейлер хаттамасы (IP, TCP, UDP тәрізді) барлық желілер үшін әмбебап, сондықтан оларда шағын тақырыптар қолданылмайды. Іс жүзінде программалық жабдықтама үстеме шығынын азайтатын ағындық өңдеу, тақырып ұзындығының одан бетер ұзаруына әкеледі (мысалы, IPv6-да IPv4 қарағанда сиретілген тақырып пайдаланылады).

Жоғары деңгейлер тақырыбы өнімділікке қатты әсер етеді. Мысалы, VoIP деректердің IP, UDP және RTP арқылы тасымалдану үдерісін қарастырайық. Бұл хаттамаларға ұзындығы 40 байт тақырып қажет (20 байт IPv4 үшін, 8 – UDP, 12 – RTP үшін). IPv6 жағдайында тіпті нашар: 60 байт, 40-байттық IPv6 тақырыбымен қоса. Бұл тақырыптар өткізгіштік қабілеттіліктің жартысынан көбін пайдаланып, бұдан да бетер ұзара беруі мүмкін.

Жоғары деңгейлер тақырыбы пайдаланатын өткізгіштік қабілеттілікті төмендету үшін **тақырыпты тығыздау (header compression)** қолданылады. Бұл жағдайда әмбебап тәсілдер емес, арнайы құрастырылған схема қолданылады. Мұның себебі, тақырыптар жеке-жеке өте нашар тығыздалады (себебі олар онсыз да қысқа), ал декомпрессия үшін бастапқы деректердің тағайындалған орынға келуі қажет. Ал бұл, дестелер жоғалуы салдарынан орындалмауы мүмкін.

Егер хаттама форматын ескеретін болсақ, тақырыпты тығыздауды тиімді жүзеге асыруға болады. Алғашқы схемалардың бірін Ван Джекобсон (1990) құрастырған болатын. Ол баяу тізбекті арна бойымен тасымалдау кезінде TCP/IP-тақырыптарды тығыздау үшін пайдаланылған. Ол әдеттегі 40-байттық TCP/IP-тақырыпты орташа алғанда 3 байтқа дейін тығыздауға мүмкіндік береді. Егер *6.46-суретке* қарайтын болса, бұл тәсілдің неге негізделгенін көруге болады. Мәселен, деста өмірі уақытының бірдей мәнін немесе TCP-порттың бірдей нөмірін әр дестемен қайталап беру міндетті емес. Бұл өрістерді тасымалдау кезінде қалдырып кетіп, деста қабылданғаннан кейін толтыруға болады.

Қалған өрістер болжалмалы сценарий бойынша өзгереді. Мысалы, егер дестелердің жоғалуы орын алмаса, TCP реттік нөмірі деректер жөнелтілген сайын өсіп отырады. Демек, қабылдаушы ықтимал мәнді болжай алады. Нөмірді тек, егер ол күтілген мәннен ерекше болған жағдайда ғана көрсету керек. Тіпті, бұл жағдайдың өзінде де, жаңа деректер келген кезде кері бағытта өсетін растаулар нөмірі тәрізді, тек осы нөмір мен алдыңғы нөмір арасындағы айырманы жөнелтуге болады.

Тақырыптарды тығыздау арқылы, жоғары деңгейлер хаттамасы тақырыбының қарапайым, ал дестені өткізгіштік қабілеттілігі төмен арна арқылы тасымалдану кезінде шағын кодтауды пайдалануға қол жеткізуге болады. Тақырыптың сенімді тығыздалуы **(ROHC, Robust Header Compression) – тақырыпты тығыздаудың қазіргі заманғы нұсқасы**, ол RFC 5795 құжатында фреймовик ретінде анықталған. Құрастырушылар ойы бойынша ол сымсыз арналардағы дестенің жоғалуына жауап қайтармайды. Әр хаттама жиыны үшін – мысалы, IP/UDP/ RTP – жеке профиль бар. Тығыздалған тақырып мәнмәтінге, яғни байланысқа сілтеме арқылы тасымалданады. Осы байланыс үшін тақырып өрістерінің мәнін жеңіл анықтауға болады, бірақ басқа байланыс дестелері үшін емес. ROHC қалыпты жұмыс істеген кезде IP/UDP/ RTP үшін тақырып көлемі 40 байттан 1-3 байтқа дейін төмендейді.

Тақырыпты тығыздаудың негізгі мақсаты – қолданылатын өткізгіштік қабілеттілікті төмендету болса да, бұл механизм көмегімен кідірісті де төмендетуге болады. Кідіріс нақты желілік жол үшін бекітілген тарату кідірісінен және өткізгіштік қабілеттілік пен тасымалданатын деректер көлеміне тәуелді тасымалдау кідірісінен тұрады. Мысалы, қаттылығы 1 Мбит/с арна 1 мкс-та 1 бит жөнелтеді. Егер біз сымсыз желі арқылы медиа деректер

жөнелтуді қарастырсақ, онда мұндай арна баяу болып саналады, сондықтан тасымалдау кідірісі жалпы кідірістің үлкен бөлігін құрайды және қызмет сапасы үшін тұрақты кідіріс өте маңызды.

Егер тақырыпты тығыздауды пайдаланасақ, тасымалданатын деректер көлемі азаяды және онымен бірге тасымалдау кідірісі төмендейді. Кіші көлемдегі дестелерді жөнелту арқылы дәл осы әсерге қол жеткізуге болады. Іс жүзінде біз программалық жабдықтамалар салдарынан туындайтын үстеме шығындардың жақсы көрсеткішін тасымалдау кідірісінің жақсы көрсеткішіне алмастырамыз. Кідірістің тағы бір көзі, сымсыз арнаға қол жеткізу кезегіндегі күту уақыты екенін айта кеткен жөн. Сымсыз желілер әдетте, жиі асыра жүктеулі болатындықтан бұл фактор өте маңызды болуы мүмкін. Бұл жағдайда сымсыз арна нақты уақыт режиміндегі дестелерге төмен кідірісті қамтамасыз ететін қызмет көрсету сапасы механизмдерін іске қосуы керек. Тек тақырыпты тығыздау жеткіліксіз.

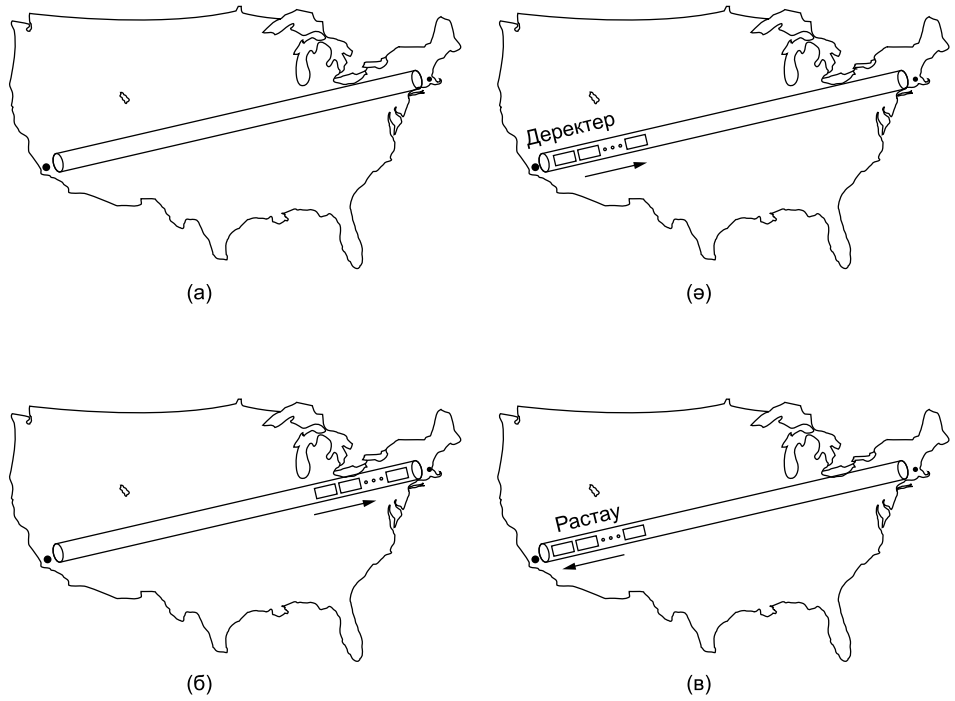
6.6.6. Өткізгіштік қабілеттілігі жоғары созылыңқы желілерге арналған хаттамалар

Деректерді үлкен арақашықтыққа тасымалдайтын гигабиттік желілердің пайда болуы 90-жылдардың басына келеді. Оларды сонымен бірге **өткізгіштік қабілеттілігі жоғары созылыңқы желілер (long fat networks)** деп атайды. Алдымен, оларға ескі хаттамаларды пайдаланбақ болды, алайда, бірден көптеген мәселелер туындады. Бұл бөлімде желілік хаттамалардың әлдеқайда жоғары жылдамдығы және кідіріс мәселелерінің бірнешеуіне тоқталамыз.

Бірінші мәселе, көптеген хаттамалар 32-разрядты реттік нөмірлерді пайдаланады. Өткен жылдарда, маршруттауыштар арасындағы бөлінген торап жылдамдығы 56 Кбит/с тең болған кезде, желіге толық жылдамдықпен үнемі деректер жөнелтетін хосттың реттік нөмірі бітуі үшін бір аптадан көп уақыт керек болар еді. TCP 232 құрастырушылары бұны шексіздікке жақындаудың жақсы көрсеткіші деп санайтын, себебі бір апта бойы дестені желі бойымен адасып, серуендеу ықтималдығы іс жүзінде нөлге тең болатын. Жылдамдығы 10 Мбит/с Ethernet үшін бұл уақыт бір аптадан 57 минутқа дейін қысқарды. Әрине, бұл әлдеқайда аз, дегенмен, тіпті, бұндай кезеңмен де жұмыс істеуге болады. Ethernet Интернетке 1 Гбит/с жылдамдықпен деректер жөнелтетін жағдайда, реттік нөмірлер шамамен, 34 с-та бітіп қалады. Ал бұл өте күрделі жағдай, себебі дестенің Интернеттегі өмір уақыты 120 с тең. Кенеттен 232-нің шексіздікке жақындалған мән ретінде жарамайтыны белгілі болды, себебі көп десте тасымалдайтын жөнелтуші реттік нөмірді, ескі дестелер әлі де желіде жүрген кезде қайталауға тура келеді.

Мәселе, көптеген құрастырушылардың реттік нөмірлер кеңістігі десте өмірі уақытының ең үлкен мәнінен бірнеше рет асып кетеді деп санауында. Сәйкесінше, бұл хаттамаларда реттік нөмірлер толық айналым жасап біткенде, ескі дестелер әлі де желіде жүруі мүмкін деген жағдай қарастырылмаған да.

Гигабиттік желілерді бұл болжау дұрыс емес болып шықты. Қуанықша орай, әр дестенің TCP-тақырыбына қосымша параметр ретінде қосылатын уақыт белгісін үлкен биттер ретінде алып, тиімді реттік нөмірді кеңейту мүмкін болды. Бұл механизм **реттік нөмірлерді қайта пайдалануды детектрлеу (PAWS, Protection Against Wrapped Sequence numbers)** деп аталады. Ол RFC 1323 құжатында сипатталған.



6.48-сурет. Жарты мегабитті Сан-Диегодан Бостонға тасымалдау: а - t=0 уақыт сәтінде; ә – 500 мкс кейін; б – 20 мс кейін; в – 40 мс кейін

Екінші мәселе – ағынды басқару терезесін едәуір ұлғайту қажеттілігі. Мысалы, қабылдаушы буфері 64 Кбайт болған кездегі, 64 Кбайттық деректерді Сан-Диегодан Бостанға жөнелту үдерісін қарастырайық. Тораптағы деректер тасымалдау жылдамдығы 1 Гбит/с делік, ал оптыталшық әйнегіндегі сәуле жылдамдығымен шектелген сигналдың бір бағытта өту жылдамдығы 20 мс тең. Бастапқыда, (t=0) *6.48 а-суретінде* көрсетілгендей арна бос. Тек 500 мкс өткеннен кейін сегменттердің барлығы арнаға келіп түседі (*6.48 ә-суреті*). Бірінші сегмент Борли аймағында, ал қалғандара әлі Оңтүстік Калифорнияда. Осылай болса тұрса да, тасымалдауыш жауап ретінде терезе жайлы жаңа ақпарат алғанша тоқтауы керек.

Жиырма мкс-тан кейін *6.48 б-суретінде* көрсетілгендей бірінші сегмент Бостонға жетеді де жауап ретінде растау жөнелтіледі. Соңында, операция басталғаннан 40 мс өткеннен кейін бірінші растау жөнелтушіге қайтып оралады,

бұдан кейін деректердің келесі бөлігі жөнелтіледі. Тасымалдау торабы 40 мс-тың небәрі 0,5 мс пайдаланылғандықтан пайдалану тиімділігі 1,25 % құрайды. Бұл ескі хаттамалардың гигабиттік тораптармен жұыс істеуінің әдеттегі жағдайы.

Желі өнімділігін сараптаған кезде **өткізгіштік қабілеттілік пен кідіріс уақытының көбейтіндісіне (bandwidth-delay product)** назар аударған дұрыс. Арнаның өткізгіштік қабілеттілігі (секундына битпен алынған) сигналдың тораптың екі басына дейін өту уақыты (секундпен) көбейтіледі. Нәтижесінде арнаның битпен санатын көлемі алынады.

6.48-суреттегі мысалда өткізгіштік қабілеттіліктің кідіріс уақытына көбейтіндісі 40 млн битке тең. Басқа сөзбен айтқанда, жөнелтуші жауап келетін уақытқа дейін 40 млн бит жөнелте алады, егер ең үлкен жылдамдықпен тасымалдайтын болса. Арнаны екі бағытта да толтыру үшін осынша бит қажет. Сонымен, жарты миллион деректер арна көлемінің небәрі 1,25 %-ын құрайды және бұл арнаны пайдалану тиімділігінің 1,25 %-ы.

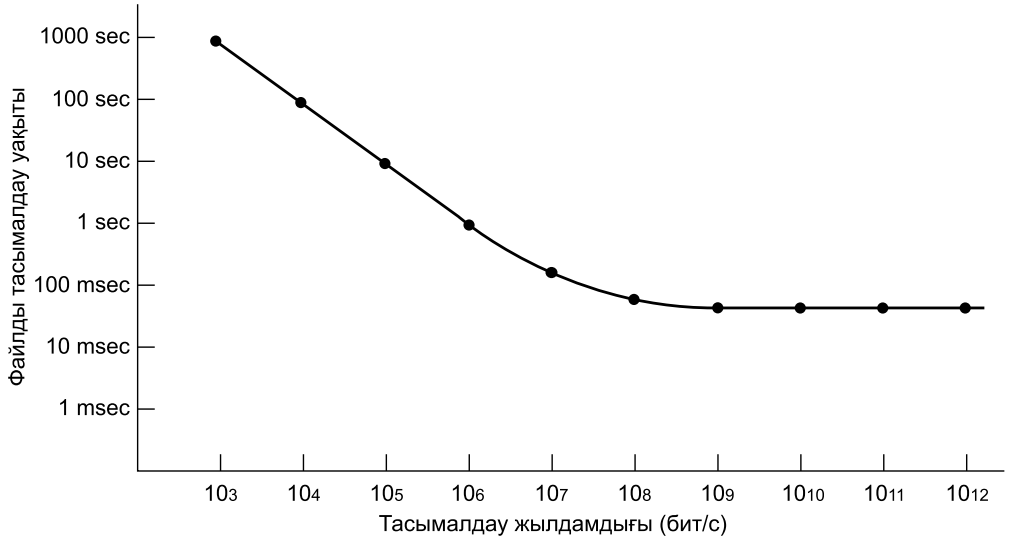
Яғни арнаны тиімді пайдалану үшін қабылдаушы терезесі кем дегенде өткізгіштік қабілеттіліктің кідіріс уақына көбейтіндісіне тең болу керек, ал ең жақсысы одан асып кеткені дұрыс, себебі қабылдаушы бірден жауап бермеуі де мүмкін. Трансконтинентальдық гигабиттік торап үшін әр байланысқа кем дегенде 5 Мбайттық терезе қажет болады.

Үшінші мәселе, алдыңғы мәселемен байланысты – қайта тасымалдаудың N блокқа қайта оралу тәрізді қарапайым схемасы, өткізгіштік қабілеттілік және кідіріс уақыты көбейтіндісі үлкен тораптарда нашар жұмыс істейді. Жылдамдығы 1 Гбит/с трансконтинентальдық торапты қарастырайық. Сигналдың тораптың екі басына жүру уақыты 40 мс. Бұл уақыт ішінде жөнелтуші 5 Мбайт тасымалдап үлгереді. Егер қателік орын алса, ол жайлы жөнелтушіге хабарлау үшін 40 мс қажет болады. *N* блокқа қайта оралу хаттамасын пайдаланған жағдайда жөнелтушіге расталмаған дестені ғана емес, сонымен бірге бұзылған дестеден кейінгі 5 Мбайт дестеге дейін қайталау керек болады. Демек, бұл хаттама ресурстарды тиімсіз пайдаланады. Сондықтан іріктеп қайталау тәсілі тәрізді әлдеқайда күрделі механизмдер қажет.

Төртінші мәселенің мәні – гигабиттік тораптар мегабиттік тораптардан ерекше. Ұзын гигабиттік тораптарда ең негізгі шектеуші фактор өткізгіштік қабілеттілік емес, кідіріс. Көлемі 1 Мбит файлды ұзындығы 4000 км торап арқылы тасымалдау уақытының тасымалдау жылдамдығына тәуелділігі *6.49-суретте* бейнеленген. 1 Мбит жылдамдыққа дейін тасымалдау уақыты негізінен деректерді тасымалдау жылдамдығына тәуелді. Жылдамдық 1 Гбит/с болғанда 40 мс кідіріс 1 мс-тен едәуір көп (оптыталшықты кабель бойымен биттерді тасымалдауға қажет уақыт). Жылдамдықтың одан әрі ұлғаюы деректерді тасымалдау уақытына ешқандай әсер етпейді.

6.49-суретте бейнеленген тәуелділік желілік хаттамалардың шектелгендігін көрсетеді. Растауды күтуі бар, **қашықтықтағы процедураны шақыру (RPC – Remote Procedure Call)** тәрізді хаттамалардың өнімділікке шектеуі

бар екенін көрсетеді. Бұл шектеу сәуле жылдамдығымен байланысты. Бұл жерде оптика аумағындағы ешбір технологиялық жаңалықтар (физиканың жаңа заңдылықтары көмектесер ме еді) көмектеспейді. Егер хост жауап күтіп отырған уақыт кезеңінде гигабиттік торап бекер тұратын болса, онда оның қарапайым мегабиттік тораптан еш артықшылығы жоқ, тек қымбат болғаны.



6.49-сурет. Тасымалдау уақыты және ұзындығы 4000 км тораптағы көлемі 1 Мбит файлды растау

Бесінші мәселе – деректерді тасымалдау жылдамдығы оларды өңдеу жылдамдығынан әлдеқайда тез өсіп отырады (Компьютер құрастырушыларға үндеу: байланыс құралдарын құрастырушыларды соғыңыздар! Біз сіздерге сенеміз). 70-жылдары біріктірілген ARPANET желісі 56 Кбит/с жылдамдықпен жұмыс істеді және өнімділігі шамамен, 1 MIPS (секундына 1 млн нұсқау) компьютерлерден тұрды. Осы цифрларды қазіргі заманғы 1000 MIPS өнімділігі бар және гигабиттік торап арқылы дестелер мен алмасатын компьютерлер цифрымен салыстырыңыз. Бір байтқа келетін нұсқаулар саны он еседен де азайды. Нақты цифрлар уақыттан және нақты жағдайға байланысты, бірақ нәтижесінде хаттамалардың дестелерді өңдеуге жұмсайтын уақыты азайды, сондықтан олар қарапайым болуы тиіс деуге болады.

Енді осы мәселелер жиынтығын шешуге арналған тәсілдерді қарастыруға көшеміз. Жоғары жылдамдықты желі құрастырушыларының жатқа білуі тиіс негізгі қағидасы мынадай:

Жобалай отырып, өткізгіштік қабілеттілікті оңтайландыруға емес, жылдамдықты өсіруге тырысыңыз.

Ескі хаттамаларды жобалау кезінде торапқа берілетін биттер санын азайту мәселесі алға қойылатын, көбіне кіші көлемді пайдалану және бірнеше өрістерді бірге бір байтқа немесе бір сөзге біріктіру есебінен. Қазіргі кезде өткізгіштік қабілеттілікті үнемдеу маңызды емес, ол жеткілікті. Мәселе хаттамалардың дестені өңдеуінде, сондықтан жаңа хаттамалар құрастыру кезінде нақты өңдеу уақытын азайту керек. Қуанышқа орай, IPv6 құрастырушылары мұны жақсы түсінеді.

Әрине, аппараттық деңгейде жылдам желілік интерфейс құрастырып, желі жұмысын жылдамдату қызығарлықтай. Бұл амалдың қиындығы, егер тек хаттама өте қарапайым болмаса, аппараттық интерфейс өзінің жеке процессоры және програмаасы бар қосымша микросхема болып келеді. Желілік сопроцессор, орталық процессор тәрізді соншалықты қымбат болмас үшін, ол көбіне, баяу микросхема болып келеді. Мұндай көзқарас нәтижесінде жылдам процессор баяу сопроцессор өз жұмысын орындағанша күтеді. Орталық процессор күту уақыты кезінде басқа жұмысты орындайды деп болжау қате болар еді. Одан бетер, жалпы қызмет көрсететін екі процессордың әрекеттесуі үшін, олардың дұрыс синхрондалуын қамтамасыз ететін, жан-жақты ойластырылған хаттама қажет. Әдеттегідей, жұмыстың барлығын орталық процессор орындау үшін, ең жақсы амал – қарапайым хаттама құрастыру.

Гигабиттік желілерде десте форматы өте маңызды. Тақырыптағы өрістер саны мүмкіндігінше аз болуы керек – бұл өңдеу уақытын азайтуға мүмкіндік береді. Өрістердің өзі, үлкен көлемдегі пайдалы (қызметтік) ақпарат сиятындай мүмкіндігінше үлкен болуы керек. Бұдан басқа, олар сөз шекерасынан аспауы тиіс, сол кезде оларды өңдеу жеңіл болады. Мұнда «мүмкіндігінше үлкен» дегеніміз ескі дестелер желіде жүрген кезде реттік нөмірлердің қайталану мәселесін болдырмау, терезе көлемі қызметтік өрісінің тым кіші болуы салдарынан қабылдаушының нақты терезе көлемін хабарлау мүмкіндігі жоқ екенін есепке алу керек дегенді білдіреді және т.с.с.

Программалық жабдықтамалар салдарынан туындайтын үстеме шығындарды азайту және желінің тиімді жұмысын қамтамасыз ету үшін, деректер өрісінің ең үлкен мөлшері мүмкіндігінше үлкен болуы тиіс. Гигабиттік желілер үшін 1500 байт десте көлемі өте аз, сондықтан Ethernet гигабиттік желісі көлемі өте үлкен кадрларды 9 байтқа дейін тасымалдауға мүмкіндік береді, ал IPv6 көлемі 64 Кбайттан жоғары джамбограммаларды қолдайды.

Енді жоғары жылдамдықты хаттамалардағы кері байланыс мәселесін қарастырайық. Салыстырмалы түрде ұзын кідіріс айналымы салдарынан кері байланыстан бас тартқан дұрыс: жөнелтушіні хабардар етуге тым көп уақыт жұмсалады. Кері байланыстың бір мысалы – тасымалдау жылдамдығын сырғымалы терезе хаттамасы көмегімен басқару. Қабылдаушының терезе қалып-күйі жайлы ақпаратын жөнелтумен байланысты ұзақ кідірісті болдырмау үшін, жөнелтушінің жылдамдыққа негізделген хаттаманы пайдаланғаны дұрыс. Мұндай хаттамада жөнелтуші қабылдаушымен алдын ала келісілген қандай да бір жылдамдықтан жылдам жөнелте алмайды.

Кері байланыстың екінші мысалы – Джекобсон құрастырған баяу старт алгоритмі. Бұл алгоритм желінің өткізгіштік қабілеттілігін анықтау үшін көптеген сынақ жүргізеді. Жоғары жылдамдықты желіде, желінің жауап рекциясын анықтау үшін бес-алты сынақ жүргізуге үлкен көлемдегі желілік ресурстар жұмсалады. Әлдеқайда тиімді схема – байланыс орнату кезінде желінің, қабылдаушының және жөнелтушінің ресурстарды алдын-ала қорда сақтауы. Сонымен бірге, ресурстарды алдын-ала қорда сақтау, жеткізу біркелкіліксіз (джиттер) әсерін төмендетуге мүмкіндік береді. Қысқаша айтқанда, желілерде тасымалдау жылдамдығының өсуі құрастырушыларды желінің байланысқа бағытталған (немесе соған жақын) жұмыс схемасын таңдауға мәжбүрлейді.

Хаттаманың тағы бір пайдалы қасиеті – байланысқа сұраныспен бірге біршама деректер тасымалдауға мүмкіндік беретіндігі. Бұл жағдайда бір сұраныс және жауап уақытын үнемдеуге болады.

6.7. КІДІРІСТЕРГЕ ТҰРАҚТЫ ЖЕЛІЛЕР

Соңғы бөлімді біз транспорттың жаңа түріне арнаймыз, болашақта ол Интернеттің ең маңызды бөлігі болуы мүмкін. TCP және көптеген басқа транспорттық хаттамалар жөнелтуші және қабылдаушы арасында жұмысшы жол бар деген болжамды ұстанады; кері жағдайда хаттама істен шығады, дестелер жеткізілмейді. Кейбір желілерде жиі толассыз жол жоқ. Мысал ретінде ғарыштық жолды келтіруге болады. Мұнда спутник **LEO (Low-Earth Orbit – төмен жержанындағы орбита)** алма-кезек жер станцияларының қолжетімді аймағында және бұл аймақтан тыс болып отырады. Нақты әр спутник осы жер станциясымен тек белгілі бір уақытта ғана байланыс орната алады, ал екі спутник ешуақытта бір-бірімен тіпті, жер станциясы арқылы да байланыс орната алмайды, себебі олардың бірі үнемі бұл станцияның қолжетімді аймағынан тыс болады. Басқа мысалдар, қозғалыс немесе төтенше шарттар салдарынан байланыс тұрақты емес болып келетін суасты қайықтарға, сондайақ автобустарға, мобильді телефондарға және басқа да компьютермен жабдықталған құрылғыларға қатысты.

Осылай бола тұрса да, деректерді тұрақты байланыс жоқ желілерде де тасымалдауға болады: бұл деректер жұмысшы байланыс пайда болғанша түйіндерде кідіріп қалады. Мұндай тәсіл **мәлімдемелерді коммутациялау** деп аталады. Осындай қағида бойынша құрастырылған желілер **кідірістерге тұрақты (DTN – Delay-Tolerant Network)** немесе **ыдырауға төзімді желілер (Disruption-Tolerant Network, DTN)** деп аталады.

DTN-ды құрастыру жұмысы 2002 жылы, IETF арнайы зерттеу тобын құрастырған кезде басталды. Кідірістерге тұрақты желілерді құрастыру қажеттілігі күтпеген жерден пайда болды: ғарышқа дестелер жөнелтуге талпынған кезде. Ғарыш желілері тұрақсыз байланыспен және ұзақ кідірістермен жұмыс істеуге мәжбүр. Кевин Фолл осындай аспан денелері арасындағы интержелілер, жер

бетіндегі тұрақсыз байланыс әдеттегі жағдай болып саналатын желілерге де жарамды екенін байқады (Fall, 2003). Бұл байланыс кезінде кідіріс және деректердің уақытша сақталуы мүмкін, Интернет моделінің жалпыламасы болып саналады. Деректерді тасымалдау, маршруттауыштардағы дестелер коммутациясына қарағанда, пошта жүйесінің немесе электронды пошта тасымалына ұқсас.

2002 жылдан бастап DTN құрылымы аяқталып, оның қолданылу аймағы кеңейтілді. Мәселен, тәжірибе нәтижесінде ЖАҚ немесе веб-қызмет арқылы алынған, бүкіл Жер шарында орналасқан деректерді жинақтау орталықтарына жөнелтуді қажет ететін, терабайттық деректер массивін елестетіп көріңіз. Операторларға мұндай трафикті қарбалас уақыттан тыс кезде жөнелткен дұрыс. Олар өздері ақша төлеген өткізгіштік қабілеттілікті дұрыс пайдалану үшін белгілі бір кідірістерге дайын. Бұл басқа қосымшалар желіні екпінді түрде пайдаланбайтын, түн уақытында орындалатын қорға көшіруді еске салады. Егер біз ауқымды қызмет көрсетумен жұмыс істейтін болсақ, онда жердің әртүрлі аймағында қарбалас кезең де әртүрлі. Егер айталық, деректер жинақтау орталығы Перт және Бостонда орналасқан болса, онда қарбалас емес кезең мүлдем қиылыспауы мүмкін (бұл қалалардың бірінде күндізгі уақыт болғанда екіншісінде түн уақыт болады).

Алайда, DTN модельдерінде деректерді сақтау және үлкен кідіріс мүмкіндігі қарастырылған. Осындай модель көмегімен деректер массивін қарбалас емес кезде алдымен Бостоннан Амстердамға (себебі бұл қалалардың уақыт айырмашылығы бар-жоғы 6 сағат). Әрі қарай, деректер қарбалас емес өткізгіштік қабілеттілікті пайдаланып Пертке жөнелту мүмкіндігі туындағанша Амстердамда сақталады. Laoutaris және басқалар (2009) жұмысында мұндай модельдің, аз ғана шығындармен жақсы өткізгіштік қабілеттілікті қамтамасыз ететіндігі айтылған және бұл өткізгіштік қабілеттілік әдеттегі толассыз модель өткізгіштік қабілеттілігінен екі есе тасып түседі.

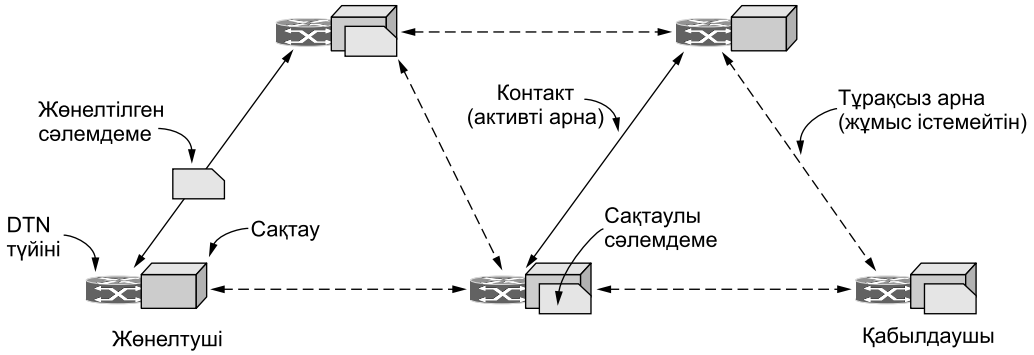
Әрі қарай біз IETF құрастырған DTN хаттамаларын қарастырамыз.

6.7.1. DTN құрылымы

DTN моделі, қазіргі заманғы Интернет негізделген болжамдардың бірінен бас тартуды ұсынады. Ол болжам мынадай: бүкіл байланыс кезінде жөнелтуші және қабылдаушы арасында толассыз жол бар. Ал бұл бұлай болмаған кезде әдеттегі Интернет хаттамалары жұмыс істемейді. Кідірістерге тұрақты желілер толассыз жолдың жоқтығын, мәлімдемелерді коммутациялауға негізделген құрылым көмегімен айналып өтеді (*6.50-сурет*). Сонымен бірге, олар деректерді сенімділігі төмен және кідірісі үлкен арналар арқылы тасымалдауға бейімделген. Бұл құрылым RFC 4838 құжатында сипатталған.

DTN терминологиясында мәлімдеме **сәлемдемелер** деп аталады. DTN түйіндері есте сақтау құрылғыларымен жабдықталған – әдетте, тұрақты жадымен (дисктер, флэш-жадылар және т.б.). Оларда сәлемдемелер қажет арна

іске қосылғанша сақталады. Сонан кейін сәлемдемелерді жөнелту басталады. Арна үзіліспен жұмыс істейді. *6.50-суретте* қазіргі сәтте жұмыс істемейтін бес тұрақсыз арна және екі активті арна бейнеленген. Активті арна контакт деп аталады. *6.50-суретте*, сонымен бірге DTN түйіндерінде, қажет контакты күтіп, сақталған екі сәлемдеме бейнеленген. Осындай схема бойынша дестелер ақпарат көзінен тағайындалған орынға жеткізіледі.

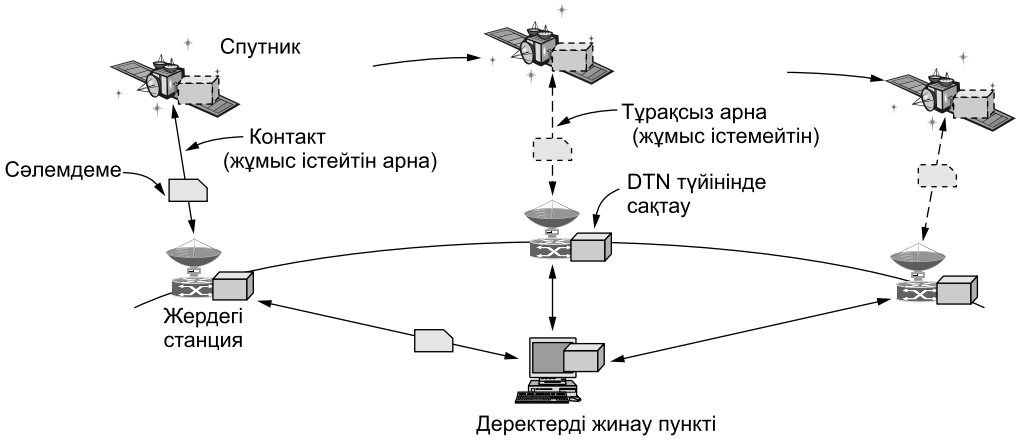


6.50-сурет. DTN құрылымы

Бұл схема маршруттауыштарда дестелермен орын алатын оқиғаға өте ұқсас. Алайда, мұнда сапалы ерекшелік бар. Интернет маршруттауыштарындағы кезектегі күту бірнеше миллисекундқа созылады, нашар жағдайда – секунд. DTN түйіндерінде сәлемдемелер сағаттап сақталуы мүмкін – қалаға баратын автобус келгенше, ұшақ қонбайды, сенсорлық желі түйіні оның жұмысына қажет энергияны жинақтай алмайды, ұйқыдағы компьютер оянбайды және т.с.с. Бұл мысалдар екінші ерекшелікті де бейнелейді: түйіндер өздерінде сақтаулы сәлемдемелермен бірге орын ауыстыра алады (автобус немесе ұшақпен бірге) және бұл деректер жеткізуде маңызды рөл атқаруы мүмкін, ал Интернет маршруттауыштары қозғала алмайды. Сәлемдемелер қозғалысын толығымен сипаттау үшін кейде «қабылдау – орын ауыстыру – жөнелту» («store – carry – forward») деген терминдер пайдаланылады.

Мысал үшін, *6.51-суретте* бейнеленген жағдайды қарастырайық. DTN хаттамалары ғарышта алғаш рет осылай қолданылған болатын (Wood және басқалар, 2008). Сәлемдеме көзі – Жер суреттерін жасайтын, LEO, Халықаралық табиғи апаттарды талдау орталығы спутниктерінің бірі. Сурет деректер жинау орталықтарының біріне келуі титіс. Бірақ, спутниктің Жер бетіндегі үш станциямен тұрақсыз байланысы бар. Орбита бойымен жылжи отырып, ол станциялармен алма-кезек байланысады. Сөйтіп, спутник, жер беті станциялары және деректер жинау орталығы DTN түйіндері болып саналады. Әр байланыс арқылы сәлемдеме (немесе сәлемдеме бөлігі) жер беті станциясына беріледі. Сонан кейін, сәлемдемелер жер беті транзиттік желісі арқылы деректер жинау орталығына жеткізіледі. Осымен тасымалдау аяқталады.

Бұл мысалдағы DTN құрылымының негізгі артықшылығы, сурет алынған сәтте байланыс жоқ болса, оны спутник жадыда сақтау керек жағдай үшін өте қолайлылығында. Бұдан басқа, DTN-ның тағы бір артықшылығы бар. Біріншіден, барлық суреттерді жөнелту үшін ол байланыста тым аз уақыт болуы мүмкін. Бұл мәселе оңай шешіледі: деректерді үш жер беті станцияларымен байланыстар арасында бөлуге болады. Екіншіден, спутник және жер беті станциясы арасындағы арна, станцияны және жер бетіндегі желіні байланыстыратын арнадан тәуелсіз жұмыс істейді. Бұл станция және спутник арасындағы деректер тасымалдау жылдамдығы, жердегі желіде баяу арналар болсада шектелмейді дегенді білдіреді. Деректер ең үлкен жылдамдықпен тасымалданады. Сәлемдеме станцияда деректер жинақтау орталығына жөнелтуге мүмкіндік туғанша сақталады.



6.51-сурет. DTN-ны ғарышта пайдалану

DTN құрылымын сипаттауда маңызды сұрақ қарастырылмайды: DTN түйіндері арқылы жақсы маршрутты қалай табуға болады. Жақсы маршрут – деректерді қашан және қандай бағытта (контактімен) жөнелту керек екені сипатталған құрылымға байланысты. Кейбір контакт жайлы алдын-ала білуге болады. DTN ғарышта пайдалану тәжрибесінде байланыс уақыты және әр жер беті станциясымен байланыстың 5-тен 14 минутқа дейін созылатыны, сонымен бірге төмендемелі торап өткізгіштік қабілеттілігінің 8,134 Мбит/с құрайтыны да алдын ала белгілі болды. Осы мәліметтер көмегімен суреттері бар сәлемдемелерді жөнелтуді жобалауға болады.

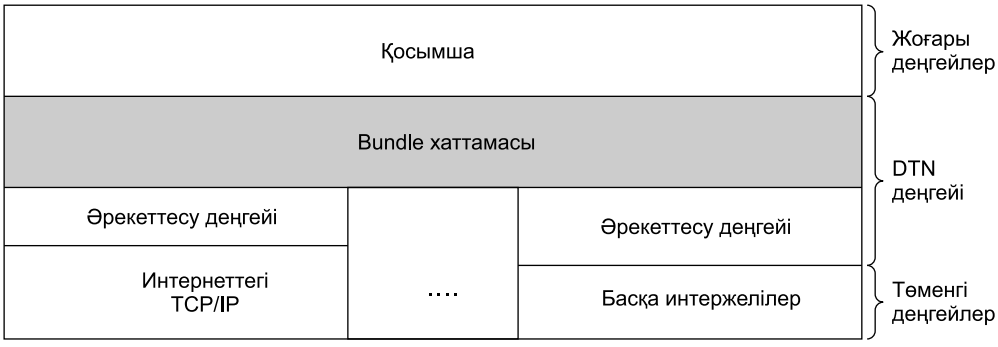
Басқа жағдайда, байланысты өте төмен ықтималдықпен болжауға болады. Мұндай жағдай мысалы – бір-бірімен ұдайы байланысып отыратын (кесте бойынша) автобустар, сонда да кейбір ауытқулар болады, сонымен бірге провайдерлер желісі, мұнда қарбалас емес кезде және жеткілікті өткізгіштік қабілеттілікті ертеде алынған деректер негізінде болжауға болады. Келесі бір шекті жағдай – байланыстардың еркін сәтте және эпизодты болып

келуі. Мысалы, тұтынушыдан тұтынушыға мобильді телефон көмегімен деректер тасымалдау кезінде, қандай тұтынушылардың қай кезде бір-бірімен байланысқа шығатыны белгісіз. Байланыстарды болжау мүмкін емес жағдайда пайдаланылатын стратегияның бірі – өмір уақыты аяқталғанша көшірмелердің бірі тағайындалған орынға жетеді деген үмітпен, сәлемдемелер көшірмесін әртүрлі жолмен жөнелту.

6.7.2. Bundle хаттамасы

DTN-нің қалай жұмыс істейтінін жақсырақ түсіну үшін IETF хаттамаларын қарастырайық. DTN – дамып келе жатқан желі түрлерінің бірі. DTN тәжірибелі жүзеге асыруларының бірінде әртүрлі хаттамалар қолданылады. Бұл үшін IETF хаттамаларын пайдалану міндетті емес. Бірақ бізге оның мысалында негізінде ең маңызды сұрақтарды қарастырған ыңғайлы.

DTN хаттамалар стегі *6.52-суретте* көрсетілген. Негізгі хаттама – бұл **Bundle хаттамасы,** ол RFC 5050 құжатында сипатталған. Ол қосымшалардан мәлімдемелер қабылдап, оны бір немесе бірнеше сәлемдемелер түрінде қабылдау – орын ауыстыру – жөнелту операциясы көмегімен қабылдаушы DTN түйініне жөнелтеді. Ол *6.52-суретте* көрсетілгендей TCP/IP деңгейінен жоғарыда жұмыс істейді. Басқа сөзбен айтқанда, түйіндер арасында сәлемдемелер тасымалдау үшін TCP/IP әр байланыста қолданыла алады. Яғни, Bundle хаттамасы қай деңгейге жатады, транспорттық, әлде қолданбалы деген сұрақ туындайды. RTP жағдайындағыдай, иерархиядағы жоғары деңгейіне қарамастан Bundle хаттамасы көптеген қосымшаларға транспорттық қызмет көрсетеді, сондықтан біз DTN-ды осы тарауда қарастырамыз.



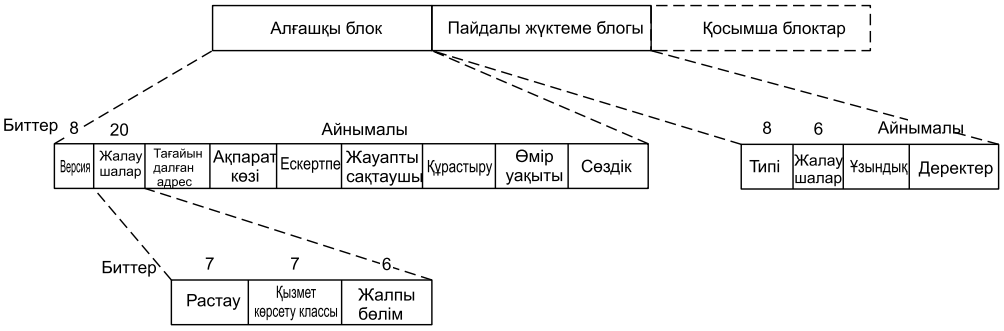
6.52-сурет. DTN хаттамалар стегі

6.52-суретінде сонымен бірге, Bundle хаттамасының басқа да хаттамалар үстінде жұмыс істей алатыны көрсетілген, мысалы, UDP немесе басқа да интержелілер. Жер және Марс арасындағы айналма кідіріс 20 минутты құрайды (олардың өзара орналасуына байланысты). Осындай арнада растаулар мен

қайталап тасымалдаулардың қаншалықты жақсы жұмыс істейтінін елестетіп көріңіз, әсіресе, қысқа мәлімдемелер үшін. Тіпті, жақсы емес. Бұндай жағдайда, қателіктерді түзету коды бар, мүлдем басқа хаттама қажет. Ресурстары тым шектеулі сенсорлық желілерде TCP орнына әлдеқайда жеңіл хаттама қолданылуы мүмкін.

Bundle хаттамасы тұрақты болғандықтан, түрлі басқа транспорттармен үйлесімділік оның есептер шеңберіне кірмейді, хаттамалардың жұмыс аумағы арасындағы саңылау кішікене болуы керек. Бұл ой, *6.52-суретте* көрсетілгендей, қосымша әрекеттесу деңгейін (convergence layer) қосуға әкелді. Іс жүзінде бұл хаттамалардың бірлесіп жұмыс істеуін қамтамасыз ететін, жәй байланыстырушы деңгей. Әрбір төменгі деңгей транспорттың анықтауы бойынша жеке әрекеттесу деңгейі болуы керек. Жаңа және қолданыстағы хаттамаларды іске қосуға мүмкіндік беретін әрекеттесу деңгейлерін, әдетте, стандарттардан табуға болады.

Bundle хаттамасының мәлімдеме форматы *6.53-суретте* келтірілген. Өріс аттары бойынша бұл хаттаманың немен айналысатынын анықтауға болады.



6.53-сурет. Bundle хаттамасының мәлімдеме форматы

Әр мәлімдеме, тақырып ретінде қабылдауға болатын, бастапқы блоктан, пайдалы жүктеме блогынан (деректер үшін) және факультативтік блоктан (мысалы, қауіпсіздік параметрлері үшін) тұрады. Бастапқы блок *Версия* өрісінен (қазіргі кезде оның мәні 6) басталады, онан кейін *Жалаушалар* өрісі орналасқан. Жалаушалар көмегімен басқалардан өзге қызмет көрсету класын (ақпарат көзі сәлемдемені жоғары басымдылықты немесе төмен басымдылықты деп белгілеу үшін) және басқа да өңдеуші сұраныстарды (мысалы, жөнелтуші жеткізілгендікті растауы міндетті ме?) көрсетуге болады.

Әрі қарай адрестер орналасқан. Мұнда біраз қызықты жағдайлар бар. Мұнда *Тағайындалған адрес* және *Ақпарат көзі* идентификаторлар өрісінен басқа *Жауапты сақтаушы* идентификаторын көруге болады. Жауапты сақтаушы – бұл дестелердің жеткізілгендігін қадағалауға міндетті бет. Интернетте бұл міндетте әдетте ақпарат көзіне жүктелген, себебі деректер тағайындалған

орынға жеткізілмеген жағдайда, жөнелтуші қайталап тасымалдауды орындайды. Бірақ, DTN түйін-жөнелтуші үнемі байланыста бола бермейді, демек, ол деректердің жеткізілген, жеткізілмегенін біле бермейді. Бұл мәселені шешу үшін DTN қабылдаушыға жақын орналасқан түйін деректердің жеткізілуіне жауапкершілікті өзіне алатын, **тапсыру-қабылдау процедурасын (custody transfer)** пайдаланады. Мысалы, егер сәлемдеме ұшақта уақытша сақталатын және кейіннен басқа жерде жөнелтілетін болса, ұшақ осы сәлемдеме үшін жауапты сақтаушы болады.

Екінші қызықты сәт, идентификаторлар IP-адрес емес. Bundle хаттамасы әртүрлі транспорттар және интержелілермен жұмыс істейтін болғандықтан, олар өздерінің жеке идентификаторларын пайдаланады. Олар, төменгі деңгей адрестерінен (IP) гөрі, веб-парақтың URL адресі тәрізді жоғарғы деңгейлер атына көбірек ұқсайды. Мұндай идентификаторлар DTN желілеріне, мысалы, электронды поштаны жеткізу немесе программалық жабдықтаманың жаңартылуларын жеткізу тәрізді, қолданбалы деңгейді маршруттауға мүмкіндік береді.

Үшінші қызықты аспект – бұл идентификаторлардың кодталуы, сонымен бірге диагностикалық мәлімдемелерге арналған *Ескерту* өрісінің идентификаторларының кодталуы. Бұл идентификаторлар ұзындығы айнымалы *Сөздік* өрісіне сілтеме арқылы кодталады. Бұл жауапты сақтаушы түйіні немесе диагностикалық мәлімдемелер ақпарат көзімен немесе тағайындалған адреспен сәйкес келгенде тығыздауды пайдалануға мүмкіндік береді. Іс жүзінде мәлімдеме форматын құрастырушылар ұзындығы айнымалы өрістерді шағын түрде қолданып, тиімділікке де, өріс ұзындығын өзгерту мүмкіндігіне де қол жеткізуге ұмтылды. Сымсыз желілерді, сонымен бірге ресурстары шектеулі, сенсорлық желі тәрізді желілерде соңғы көрсеткіш маңызды рөл атқарады.

Келесі орналасқан *Құрастыру* өрісі. Мұнда сәлемдеменің құрастырылған уақыты және жөнелтушінің реттік нөмірі жазылады. Онан кейін, сәлемдеменің қай кезде қажетсіз болып қалатындығы көрсетілген, *Өмір уақыты* өрісі орналасқан. Бұл өрістер қажет, себебі сәлемдемелер DTN түйіндерінде ұзақ сақталуы мүмкін, сондықтан желіде ескірген деректерді жою механизмі болуы керек. Бұл жерде сағат Интернетке қарағанда босаңқы синхрондалуы тиіс.

Бастапқы блок *Сөздік* өрісімен аяқталады. Әрі қарай пайдалы жүктеме өрісі орналасқан. Ол пайдалы жүктеме екенін білдіретін, қысқа *Тип* өрісінен басталады. Онан кейін, өңдеу параметрлері берілетін, *Жалаушалар* орналасады. Онан кейін, алдында *Ұзындық* өрісі бар, *Деректер* өрісі орналасқан. Соңында, олардан кейін факультативті блоктар – жеке алғанда, қауіпсіздік параметрлері көрсетілген блок орналасуы мүмкін.

Кідіріске тұрақты желілердің көптеген тұстары ғылыми бірлестіктерде талқылануын жалғастыруда. Жоғарыда айтып кеткендей, маршруттаудың жақсы стратегиясы байланыстар табиғатына тәуелді. Желі ішінде деректерді сақтау идеясы жаңа мәселенің туындауына әкеледі. Асыра жүктелуді бақылау,

DTN түйіндеріндегі жадыны басқа ресурстар типіне жатқызуы тиіс және мұндай ресурстар да бітіп қалуы мүмкін. Толассыз байланыстың жоқтығы қауіпсіздік мәселесін қиындатады. Түйін сәлемдеменің жеткізілетіндігіне жауапкершілікті өзіне алмас бұрын жөнелтушінің желіде ресми тіркелгендігін және қабылдаушыға бұл сәлемдеменің керек екенін тексеруі мүмкін. Бұл мәселелердің шешімі DTN типіне байланысты, себебі ғарыштық желілердің сенсорлық желілерден айырмашылығы айтарлықтай!

6.8. ТҮЙІНДЕМЕ

Транспорттық деңгей – бұл көпсатылы хаттамаларды түсінудің кілті. Ол әртүрлі қызметтер ұсынады, оның ішіндегі ең маңыздысы толассыз, сенімді, байланысқа бағытталған жөнелтушіден қабылдаушыға дейінгі байттар ағыны. Ол қол жеткізу, байланысты орнатуға, пайдалануға және үзуге мүмкіндік беретін қызметтік примитивтер көмегімен жүзеге асырылады. Транспорттық деңгейдің жалпыға бірдей интерфейсі Беркли сокеттерімен қамтамасыз етіледі.

Транспорттық хаттамалар сенімсіз желілерде байланысты басқару қабілетіне ие болуы тиіс. Байланысты орнату, ең қолайсыз сәтте пайда болатын, деста көшірмелерінің болуымен қиындай түседі. Мұндай көшірмелермен күресу үшін байланыс орнату кезінде «үштік қол алысу» алгоритмі қолданылады. Байланысты үзу орнатудан жеңіл болғанымен, екі армия мәселесінің орын алуына байланысты біз ойлағандай қарапайым емес.

Желілік деңгей абсолютті сенімді болғанның өзінде де, транспорттық деңгейде жұмысы жетерліктей. Ол барлық қызметтік примитивтерді өңдеуі, байланыстар және таймерлерді басқаруы, өткізгіштік қабілеттілікті таратуы және асыра жүктелуді басқаруы тиіс, сонымен бірге деректер ағынын басқару үшін ұзындығы айнымалы сырғымалы терезені пайдалануы керек.

Асыра жүктелуді басқару өткізгіштік қабілеттілікті бәсекелес ағындар арасында әділ таратуы және желіні пайдалау өзгерістерін қадағалауы тиіс. AIMD басқару заңы тиімді және әділ таратуға қол жеткізуге мүмкіндік береді.

TCP және UDP Интернеттің негізгі транспорттық хаттамалары болып саналады. UDP – бұл IP-дестелермен жұмыс істеп, ортақ IP-адресті пайдаланып бірнеше үдерістерді мультиплекстеу және демультиплекстеуді қамтамасыз ететін, байланыс орнатуды қажет етпейтін хаттама. UDP клиент-серверлік әрекеттесуде, мәселен, процедураны қашықтықтан шақыру (RPC) кезінде қолданылады. Сонымен бірге, оның негізінде RTP тәрізді нақты уақыт хаттамаларын құрастыруға болады.

Интернеттің кеңінен таралған хаттамасы – TCP. Ол асыра жүктелуді бақылай отырып, сенімді дуплексті байттар ағынын қамтамасыз етеді. Ол барлық сегменттер үшін 20-байттық тақырыпты пайдаланады. TCP хаттамасының өнімділікті оңтайландыруына көп назар бөлінген. Ол үшін онда Нагаль (Nagle), Кларк (Clark), Джекобсон (Jacobson), Карн (Karn) және басқа да алгоритмдер қолданылады.

Желі өнімділігі әдетте хаттама және сегменттерді өңдеудегі үстеме шығындар негізінде анықталады және де деректер тасымалдау жылдамдығының өсуімен бұл жағдай нашарлай түседі. Хаттамалар сегменттер санын азайтатындай және тасымалдау кідірісінің үлкен мәндерінде жұмысқа қабілетті болып қалатындай етіп құрастырылуы тиіс. Гигабайттық желілерде қарапайым хаттама және ағындық өңдеу қажет.

Кідіріске тұрақты желілерді құрастыру түйіндер байланысы тұрақсыз немесе арналар кідірісі үлкен желілерде дестелерді жеткізуге мүмкіндік береді. Аралық түйіндер деректер сәлемдемесін сақтайды, тасымалдайды және жөнелтеді, сонымен бірге жөнелтуші және қабылдаушы арасында қандай да бір уақыт сәтінде жұмысшы жол жоқ болғанның өзінде де олардың жеткізілуіне кепілдік береді.

СҰРАҚТАР

1. Біздің *6.1-кестеде* келтірілген транспорттық примитивтер мысалымызда, `LISTEN` оқшаулаушы шақыру болып саналады. Бұл міндетті ме? Олай болмаған жағдайда, оқшауламайтын базалық операцияны қалай пайдалану керек екенін түсіндіріңіз. Бұның мәтінде сипатталған схемамен салыстырғандағы артықшылығы неде?

2. Транспорттық қызметтің базалық операциялары, байланыс орнату кезінде соңғы екі нүктесі симметриялы әрекет етпейді деп болжайды: бір басы (сервер) `LISTEN` операциясын, ал екіншісі (клиент) – `CONNECT` операциясын орындайды. Бірақ біррангілі қосымшаларда (мысалы, BitTorrent тәрізді файлдарға бірлесіп қол жеткізу қосымшаларында) барлық соңғы нүктелер бірдей деп есептеледі. Олардың арасында серверлер және клиенттер жоқ. Осындай қосымшалар өз жұмысында транспорттық қызметтің базалық операциясын қалай пайдалана алады?

3. Қалып-күй диаграммасының негізінде жатқан модельде (*6.3-суретті* қараңыз) дестелер желілік деңгейде жоғалуы мүмкін, сондықтан жеке расталуы тиіс деп болжалады. Айталық, желілік деңгей жеткізудің 100% сенімділігін қамтамасыз етеді және ешуақытта дестені жоғалтпайды. *6.3-суретте* көрсетілген қалып-күй диаграммасына қандай да бір өзгерістер енгізу қажет пе? Қажет болса, қандай?

4. *6.1-листингтің* екі бөлігінде де `SERVER_PORT` мәні клиентте де, серверде де бірдей болуы керек. Неліктен бұл маңызды?

5. *6.1-листингінде* келтірілген файлдық сервер мысалын қараңыз. Клиенттің `connect()` жүйелік шақыруы, сервердің күту кезегінің асыра толуынан басқа себеппен сәтсіз аяқталуы мүмкін бе? Желі мінсіз деп санаңыз.

6. Серверді үнемі активті қалып-күйде ұстау керек пе, әлде өңдеуші сервер көмегімен талап бойынша іске қосқан дұрыс па екенін шешу үшін, серверді пайдалану жиілігі критерийін қолдануға бола ма? Қандай да бір басқа критерий ойлап табуға бола ма?

7. Айталық, сағатпен басқарылып, 15-разрядты сағат санауышы бар бастапқы реттік нөмірлерді генерациялайтын схема қолданылады. Сағат әр 100 мс сайын бір тик қосады, ал дестенің ең ұзақ өмір уақыты 60 с-ке тең. Ресинхрондау қаншалықты жиі жүргізілуі тиіс:

(а) Ең нашар жағдайда?
(ә) Деректер минутына 240 реттік пайлаланған кезде?

8. Не себептен дестенің Т ең ұзақ өмір уақыты мүмкіндігінше үлкен болуы керек? Дестe ғана емес, оның растауы да жоғалғандығына кепілдік беру үшін бе?

9. Байланыс орнату үшін «үштік қол алысу» орнына «екілік қол алысу» (яғни үшінші мәлімдеме қажет болмады) пайдаланылды деп болжаңыз. Бұл жағдайда тұйыққа тірелу мүмкін бе? Мысал келтіріңіз немесе тұйыққа тірелудің болмайтынын дәлелдеңіз.

10. Өзіңізге кез келген екі армияның келісімі жеткілікті деп шешілген, *n* армияның жинақталған мәселесін елестетіңіз. Көктердің жеңіске жетуіне мүмкіндік беретін хаттама бар ма?

11. Хосттарды іскен шыққаннан кейін қалыпқа келтіру мәселесін қарастырайық (*6.15-сурет*). Егір растаудың жазылу және жөнелтілу арасындағы (немесе керісінше) кезеңді салыстырмалы түрде кіші етуге мүмкін болса, хаттаманың қателесу ықтималдығын азайтатын, жөнелтуші мен қабылдаушының ең жақсы стратегиясы қандай болар еді?

12. *6.17-суретте* бейнеленген желіге *R1, R2* және *R6* маршруттауыштары арқылы өтетін жолды пайдаланатын, жаңа *E* ағыны қосылды деп болжаңыз. Ең үлкен критерий бойынша бес ағын үшін өткізгіштік қабілеттілікті тарату қалай өзгереді?

13. Кредиттік хаттаманың (қабылдаушы жөнелтушіге ақпараттың неше блогын қабылдай алатынын ескертеді) сырғымалы терезе хаттамасымен салыстырғандағы артықшылықтары мен кемшіліктерін талқылаңыз.

14. Асыра жүктелуді бақылау кезінде теңқұқықтылықты қамтамасыз ететін басқа саясатшылар бар ма: аддитивті үлкейту аддитивті азайту (AIAD), мультиплекативті үлкейту аддитивті азайту (MIAD), мультиплекативті үлкейту мультиплекативті азайту (MIMD). Олардың жинақтылығы және тұрақтылығы жайлы не айтасыз?

15. UDP хаттамасы не үшін қажет? Тұтынушы үдерістеріне өңделмеген IP-дестелерді жай жөнелте салуға рұқсат беру жеткілікті болмас па еді?

16. Жалпыға белгілі адрес бойынша орналасқан, қашықтықтағы сервердегі файлға сұраныс жасауға мүмкіндік беретін, UDP негізінде құрастырылған қолданбалы деңгейдің қарапайым хаттамасын қарастырыңыз. Бастапқыда клиент файл атын көрсетіп сұраныс жасайды, ал сервер сұралған файлдың түрлі бөліктерінен тұратын ақпараттық дестелер тізбегімен жауап қайтарады. Сенімділікті және бөліктердің дұрыс ретпен жеткізілуін қамтамасыз ету үшін, клиент те, сервер де күтуі бар хаттаманы қолданады. Мұндай хаттамамен айқын өнімділік мәселесінен басқа қандай мәселе туындауы мүмкін? Үдерістердің істен шығу ықтималдығына назар аударыңыз.

17. Клиент өзінен 100 км қашықтықта орналасқан серверге, жылдамдығы 1 Гбит/с оптыталшық арқылы 128-байттық сұраныс жөнелтеді. Қашықтықтағы процедураны шақыру кезіндегі тораптың тиімділігі қандай?

18. Алдыңғы сұрақтағы жағдайды тағы да қарастырыңыз. Берілген, жылдамдығы 1 Гбит/с және жылдамдығы 1 Мбит/с торап үшін, мүмкін деген ең кіші жауап уақытын есептеңіз. Алынған мәндерге сүйеніп, қандай қорытынды шығаруға болады?

19. UDP-да да TCP-де де порттар нөмірі, мәлімдеме жеткізілгенде қабылдаушы ішкі жүйені сәйкестендіру үшін қолданылады. Бұл хаттамалар үшін жаңа абстракты идентификаторлардың (порт нөмірі) не үшін құрастырылғандығының екі себебін атаңыз және бұл хаттамалар пайда болған сәтке дейін қолданыста болған үдерістер идентификаторы не себептен пайдаланылмады?

20. RPC-дің кейбір жүзеге асырулары клиентке UDP көмегімен және TCP көмегімен жүзеге асырулар арасында таңдау жасауға мүмкіндік береді. Қандай жағдайда клиент бірінші нұсқаны, қандай жағдайда – екіншісін таңдайды?

21. *A* ақпарат көзінен *D* қабылдаушыға дестелер тасымалдау кезіндегі орташа кідірісі бірдей *N1* және *N2* екі желісін қарастырайық. *N1* желісінде кідіріс 10 с ең үлкен мәнімен біркелкі таратылған, ал *N2* желісінде дестелердің 99% кідірісі 1 секундтан кем, ал ең үлкен кідіріс үлкен болуы мүмкін. Егер сіз аудио/бейнені нақты уақыт режимінде тасымалдауды жоспарлап отырсаңыз, RTP қалай пайдалануға болатынын ойластырыңыз.

22. TCP хаттамасының ең кіші MTU-ның TCP және IP үстеме шығындары қоса алғанда, бірақ арналық деңгей үстеме шығындарын есептемегендегі қосынды өлшемі қандай?

23. Фрагменттеу және қайта фрагменттеу операцияларын IP хаттамасы орындайды және ол TCP хаттамасына көрінбейді. Бұл TCP хаттамасы ретсіз келетін деректер жайлы ойламауы керек дегенді білдіре ме?

24. RTP сапасы компакт-дискілерге сәйкес келетін дыбыс жазбаларын тасымалдау үшін пайдаланылады. Сонымен бірге, әр стереоарнаның бір саналымын жөнелту үшін, секундына 44 100 рет тасымалданатын екі 16-биттік сөз пайдаланылады. RTP секундына неше десте жөнелте алуы керек?

25. RTP кодын UDP-мен бірге операциялық жүйе ядросына орналастыруға мүмкіндік бар ма? Жауабыңызды түсіндіріңіз.

26. Бірінші хост үдерісіне p порты, ал 2-хост үдерісіне – *q* порты тағайындалған. Осы екі порт арасында бір мезгілде бірнеше байланыс болуы мүмкін бе?

27. Біз *6.31-суретте* 32-разрядты *Растау нөмірі* өрісіне қосымша төртінші сөзде АСК битінің бар екендігін көрдік. Оның қандай да бір пайдасы бар ма? Жауабыңызды түсіндіріңіз.

28. TCP-сегменттің ең үлкен пайдалы жүктемесінің мөлшері 65 495 байтқа тең болуы мүмкін. Неліктен бұл сан таңдалып алынды?

29. *6.33-суреттегі SYN RCVD* қалып-күйіне көшудің екі жолын сипаттаңыз.

30. Сигналдың, асыра жүктелусіз торап бойымен байланыстың екі басына өту уақыты 10 мс болғанда баяу старт алгоритмін пайдалану әсерін қарастырыңыз. Қабылдаушы терезесінің көлемі 24 Кбайт, ал сегменттің ең үлкен мөлшері 2 Кбайт. Неше уақыттан кейін толық терезені жөнелтуге болады?

31. Айталық, TCP хаттамасының асыра жүктелу терезесі 18 Кбайт етіп орнатылған, тайм-аут қай кезде орын алады. Егер келесі төрт тасымалдау сәтті болса, терезе көлемі қандай болады? Ең үлкен көлем 1 Кбайт деп болжаңыз.

32. TCP хаттамасында сигналдың байланыстың екі басына да жету *RTT* уақытының ағымдағы мәні 30 мс, ал келесі растау 26, 32 және 24 мс кейін келеді. *RTT* жаңа мәні қандай болады? α=0,9 деп есептеңіз.

33. Сигналдың байланыстың бір басына жету уақыты 10 мс құрайтын гигибиттік арна бойымен TCP-машина 65 535 көлеміндегі терезе жөнелтеді. Арнаның ең үлкен қолжетімді өткізгіштік қабілеттілігі қандай? Торапты пайдалану тиімділігі нешеге тең?

34. Хосттың, 1500 байт пайдалы жүктемеден тұратын TCP-дестені торапқа жөнелтетін ең үлкен жылдамдығы қандай? Дестенің желідегі ең үлкен өмір уақыты 120 с тең. Реттік нөмірлер қайталанбау керек. Есептеу кезінде TCP, IP және Ethernet үстеме шығындарын есепке алыңыз. Ethernet кадрлары үздіксіз жөнелтіледі деп болжаңыз.

35. IPv4 адресіндегі шектеулерді түзету үшін IETF едәуір күш жұмсап, IPv6 құрастырды, алайда, бұл версия әлі де баяу ендірілуде. Ал TCP адрестер шектеулерін түзетуге ешкім мұндай күш жұмсаған жоқ. Неліктен бұлай екенін түсіндіріңіз.

36. Сегменттің ең үлкен көлемі 128 байт болғанда, деректер тасымалдаудың ең үлкен жылдамдығы қандай болады? Сегменттің ең ұзақ өмір уақыты 30 с және ол 8-разрядты реттік нөмірлерді пайдаланылады.

37. Айталық, сіз сегментті қабілдауға қажет уақытты өлшеп отырсыз. Үзіліс орын алған кезде сіз жүйелік сағат көрсеткішін миллисекундпен санайсыз. Сегмент толық өңделгеннен кейін сіз сағат көрсеткішін тағы санайсыз. Миллион өлшеу нәтижесінен кейін сіз 270 00 рет 0 мс мәнін және 730 000 рет 1 мс алдыңыз. Алынған нәтижелер негізінде қандай қорытынды жасауға болады?

38. Орталық процессор секундына 1000 нұсқау орындайды (1000 MIPS). Деректер 64-разрядты сөздер ретінде көшіріле алады. Әр сөзді көшіруге 10 нұсқау қажет. Егер дестелерді төрт рет көшіру керек болса, осындай жүйе гигабайттық торапты басқара ала ма? Қарапайымдық үшін барлық нұсқаулар, тіпті, жадымен жұмыс істеу ең үлкен жылдамдықпен, 1000 MIPS орындалады деп есептейміз.

39. Ескі дестелер әліде айналымда болған кезде, олардың ескі реттік нөмірлерді қайта пайдалану мәселесін шешу үшін, 64-разрядты реттік нөмірлерді пайдалануға болады. Алайда, теорияда оптыталшықты кабель 75 Тбит/с өткізгіштік қабілеттіке ие бола алады. Торап жылдамдығы 75Тбит/с және реттік нөмірлер 64-разрядты болған кезде, болашақ желіде реттік нөмірлері бірдей дестенің болмайтындығына кепілдік беру үшін дестенің ең ұзақ өмір уақытын қандай етіп таңдау керек? Реттік нөмірлер TCP-дағыдай әр байтқа беріледі деп санаймыз.

40. Есептеулер нәтижесінде гигабиттік торап хостқа секундына 80 000 дестe жөнелтетіндігі және оларды өңдеуге тек 6250 нұсқау қалдыратындығы белгілі болды. Процессордың өнімділігінің жартысы қосымшаларға беріледі. Сонымен бірге дестелердің мөлшері 1500 байт болады деп болжанады. Есептеуді ARPANET (128 байт) желісі үшін қайта орындаңыз. Екі жағдайда да дестe көлеміне барлық үстеме шығындар қосылған деп есептейміз.

41. Ұзындығы 4000 км-ден үлкен гигабиттік торапта шектеуші фактор өткізгіштік қабілеттілік және кідіріс болып саналады. Жөнелтуші және қабылдаушы арасындағы орташа арақашық шамамен, 20 км болатын аймақтық желіні қарастырайық. Деректер тасымалдаудың қандай жылдамдығы кезінде сигналды екі жаққа да өту уақыты, көлемі 1 Кбайт дестенің бір дестенің тасымалдану жылдамдығына тең болады?

42. Төменде көрсетілген желілердегі өткізгіштік қабілеттілік және кідіріс көбейтіндісін есептеңіз:

(1) T1 (1,5 мбит/с);
(2) Ethernet (10 Мбит/);
(3) T3 (45 Мбит/с);
(4) STS-3 (155 Мбит/с).

RTT=100 мс деп болжаңыз. TCP тақырыбында терезе көлеміне 16-разрядтық өріс бөлінетіндігін ұмытпаңыз. Бұл ескерту есептеулер нәтижесіне қалай әсер етеді?

43. Өткізгіштік қабілеттілігі 50 Мбит/с геостацинарлық спутниктік байланыс арнасы үшін өткізгіштік қабілеттіліктің кідіріске көбейтіндісі нешеге тең? Егер барлық дестелердің көлемі 1500 байт болса (үстеме шығындарды қоса алғанда), дестелердегі терезе қандай болуы керек?

44. *6.1-листиг* моделдейтін файлдық сервер мінсіз емес. Кейбір жақсартулар енгізуге болады. Келесі өзгерістерді орындаңыз:

(а) Клиентте, байттық диапазонға нұсқайтын үшінше аргумент пайда болсын.

(ә) Клиент программасына файлдық серверге жауға мүмкіндік беретін -w жалаушасын қосыңыз.

45. Желілік хаттамалардың барлығы мәлімдемелермен жұмыс істей білуі тиіс. Егер естеріңізде болса, хаттама мәлімдемені тақырыпқа бөлімше қосу (немесе бөлу) арқылы жөнелтеді. Кейбір хаттамалар мәлімдемені бірнеше фрагментке бөледі, ал сонан кейін оны фрагменттерден қалпына келтіреді. Жаңа мәлімдеме құрастыру, тақырып қосу/бөлу, бір мәлімдемені екіге бөлу, екі мәлімдемені біріктіру және мәлімдеме көшірмесін сақтауды жүзеге асырып, мәлімдемелердің басқару кітапханасын құрастырып көріңіз. Деректерді бір буферден екіншісіне көшіруді мүмкіндігінше азайтыңыз. Бұл операциялардың, мәлімдемедегі деректерге тиіспей, тек көрсеткіштермен орындалғандығы өте маңызды.

46. Бірнеше тұтынушылар тобына есептелген желілік хабарласу (чат) жүйесін құрастырып, жүзеге асырыңыз. Чат координаторы жалпыға белгілі адрес бойынша орналасуы, клиенттермен байланыс үшін UDP хаттамасын пайдалануы, әрбір хабарласу алдында чат-серверді баптауы және чат-сессия каталогын қолдауы тиіс. Әр сессияға бір қызмет көрсетуші сервер бөлінуі керек. Клиентпен байланысу үшін сервер TCP-ді пайдалануы тиіс. Клиенттік программа тұтынушыларға әңгімені бастауға, жүріп жатқан пікірталасқа қосылуға және сессиядан шығып кетуге мүмкіндік беруі қажет. Координатор, сервер және клиент кодын құрастырып, жүзеге асырыңыз.

7-ТАРАУ
ҚОЛДАНБАЛЫ ДЕҢГЕЙ

Компьютерлік желілер жайлы негізгі мәліметтерді оқуды аяқтап, біз барлық қосымшалар орналасқан деңгейге көшеміз. Қолданбалы деңгейден төмен орналасқан деңгейлердің барлығы деректерді сенімді жеткізу үшін қызмет етеді, бірақ тұтынушы үшін ешқандай пайдалы әрекет орындамайды. Біз бұл тарауда кейбір нақты желілік қосымшаларды қарастырамыз.

Әрине, тіпті, қолданбалы деңгейдің өзі де қосымшалардың жұмыс істеуіне қажетті қызмет көрсетуші хаттамаларды қажет етеді. Сәйкес қосымшаларды талқыламас бұрын, біз осындай хаттамалардың бірін қарастырамыз. Әңгіме, Интернетте ат беруді қамтамасыз ететін DNS, домендер қызметі жайлы болады. Сонан кейін біз, іс жүзінде қызмет ететін: электронды пошта, Дүние-жүзілік өрмек және мультимедиа сияқты үш қосымшаны қарастырамыз. Біз бұл тарауды контенті жеткізу жайлы әңгімемен аяқтаймыз, сонымен бірге теңрангілі (пирингілік – peer-to-peer немесе қысқаша P2P) желілер жайлы айта кетеміз.

7.1. DNS ДОМЕНДІК АТТАР ҚЫЗМЕТІ

Теориялық тұрғыдан программалар ақпарат орналасқан веб-парақтарға, пошта жәшіктеріне және басқа да ресурстарға компьютердің желілік адресі (мысалы, IP) қол жеткізе алатын болғанымен, оларға мұндай адрестерді есте сақтау қиын. Бұдан басқа, компания веб-парағының *128.111.24.41* адресі бойынша орналасуы компания сервері жаңа машинаға орын ауыстырған кезде жаңа IP-ді барлығына хабарлау керек болады дегенді білдіреді. Машина аты бойынша оның адресін ажырату үшін, жоғары деңгейдегі түсінікті аттарды пайдалану қабылданған. Сондықтан компания веб-серверіне *www.washington.edu* адресі бойынша қол жеткізуге болады. Дегенмен де, желі өздігінен тек

сандық адрестерді түсінетін болғандықтан, аттарды желілік адрестерге түрлендіру механизмі қажет. Келесі бөлімдерде біз осы түрлендірулердің Интернетте қалай жүргізілетіндігін қарастырамыз.

Ертеде, ARPANET желісі кезінде мәтіндік және сандық адрестер арасындағы сәйкестік *hosts.txt* файлына жазылады, мұнда барлық компьютерлердің аттары және IP-адрестерінің тізімі бар болатын. Әр түн сайын барлық хосттар бұл файлды сол файл сақталатын сайттан алатын. Уақытты бөлу жүйесінің басқаруымен жұмыс істейтін, жүздеген үлкен машиналардан тұратын желіде бұл амал өзін-өзі ақтайтын. Алайда, желіге миллион компьютер қосылудан бұрын, барлығына бұл тәсілдің мәңгілік жұмыс істей алмайтыны белгілі болды. Біріншіден, ерте ме кешпе файл көлемі тым үлкен болып кетеді. Алайда, одан да маңыздысы, егер хосттар аттарын басқару бір орталықтан жүргізілмесе, аттар шиеленісі орын алады. Сонымен бірге, ауқымды халықаралық желінің барлық хосттар аттарын бір орталықтан басқаруды елестету өте күрделі. Осы мәселелердің барлығын шешу үшін 1983 жылы **домен аттары қызметі (DNS, Domain Name System)** құрастырылды. Содан бері ол Интернеттің маңызды бір бөлігі болып саналады.

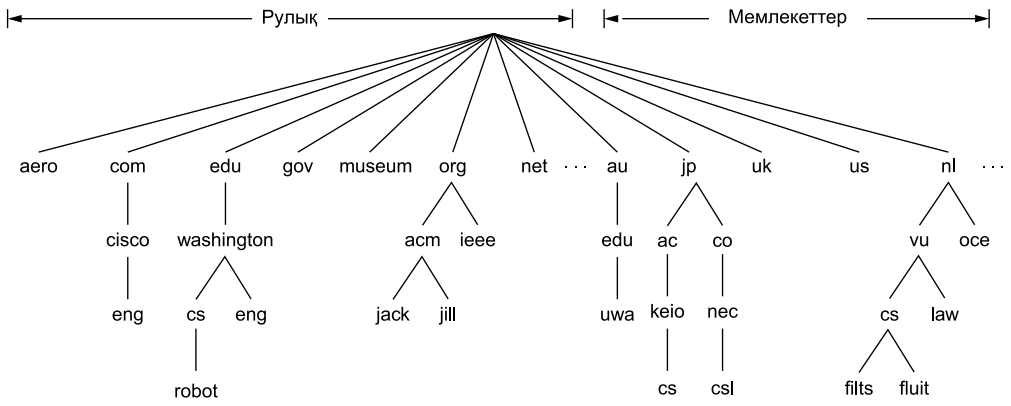
DNS жүйесінің мағынасы домендер мен таратылған деректер базасына негізделген және осы аттар схемасын жүзеге асыратын аттардың иерархиялық схемасында. Бұл жүйе алдымен хосттар атын IP-адрестерге түрлендіру үшін қолданылады, сонымен бірге оны басқа мақсаттарда да пайдалануға болады. DNS жүйесінің анықтамасы RFC 1034, 1035, 2181 құжаттарында берілген және ары қарай басқа құжаттарда да қарастырылған.

Жалпы алғанда DNS жүйесі келесідей қолданылады: аттарды IP-адрестерге түрлендіру үшін қолданбалы **программа танушы (resolver)** деп аталатын кітапханалық процедураға жүгініп, оған атты параметр ретінде береді. Біз танушы мысалын, *gethostbyname 6.1-листингінде* көрген болатынбыз. Танушы ат жазылған сұранысты DNS-серверге жөнелтеді. Сервер атты іздеп, оған сәйкес IP-адресті танушыға қайтарады. Танушы өз кезегінде ол адресті өзін шақырған қолданбалы программаға береді. Сұраныс және жауап дестелері UDP-дестесі ретінде ұсынылады. IP-адресті алғаннан кейін программа адресатпен TCP-байланыс орнатады немесе оған UDP-деста жөнелтеді.

7.1.1. DNS аттар кеңістігі

Үлкен және үнемі өзгеріп отыратын аттар жиынын басқару оңай мәселе емес. Пошта жүйесіндегі хаттарда (нақты немесе жанама түрде) елді, штатты немесе ауданды, қаланы, көшені, үй нөмірін, пәтер және қабылдаушы тегін көрсету қажет. Осындай иерархиялық схеманы қолданғандықтан Нью-Йорк штатының Уайт-Плейнс қаласындағы Мейн көшесінде тұратын және Техас штаты Остин қаласының Мейн көшесінде тұратын Марвин Андерсон арасында шатастырулар туындайды. DNS жүйесіде пошта жұмысына сәйкес жұмыс істейді.

Интернет үшін аттар беру иерархия негізін **ICANN (Internet Corporation for Assigned Names and Numbers – аттар және адрестер тағайындау бойынша интернет-корпорация)** деп аталатын ұйым құрастырған. Интернет дүниежүзілік экономикалық концернге дейін дамығаннан соң, ICANN осы мақсатта 1998 жылы құрастырылған болатын. Интернет 250-ден аса тұжырымдамалық **жоғарғы деңгей домендеріне (top-level domains)** бөлінген. Интернетте домен деп бір логикалық топқа біріктірілген хосттар жиынын атайды. Жоғарғы деңгейдің әр домені ішкі домендерге (subdomains) бөлінеді, ішкі домендер өз кезегінде басқа домендерден тұруы мүмкін және т.с.с. Осы домендердің барлығы *7.1-суретте* бейнеленген ағаш түрінде қарастыруға болады. Ағаш жапырағы ішкі домендерге бөлінбейтін домен болып саналады (хосттар кірмейтін соңғы домен). Мұндай домен бір хосттан немесе үлкен бір мекеменің өкілі болып, қарамағындағы мыңдаған хосттардан тұруы мүмкін.



7.1-сурет. Интернет аттары домендік кеңістігінің бір бөлігі

Жоғарғы деңгей домендері: рулық домендер және мемлекеттік домендер болып екі топқа бөлінеді. Рулық домендерге *7.1-кестеде* келтірілген домендер және 1980 жылдары құрастырылған біртума домендер, сонымен бірге ICANN енгізген жаңа домендер жатады. Алдағы уақытта жоғарғы деңгейдің жаңа базалық деңгейлері қосылатын болады.

ISO 3166 – халықаралық стандартқа сәйкес әр мемлекеттің жеке домені бар. Елдердің, латын әліпбиінен өзгеше әліпби қолданылған интернационалдандырылған домендік аттары 2010 жылы енгізілді. Бұл домендер араб, орыс, қытай және басқа да жазбаларды пайдаланып, хостарға ат беруге мүмкіндік жасайды.

Екінші деңгейдегі *компания_аты.com* тәрізді домендерді қорда қалдыру жеңіл. Жоғары деңгей домендері ICANN тағайындаған **тіркеуіштермен (registrars)** басқарылады. Ат алу үшін сәйкес тіркеуішке (бұл жағдайда *com*) жүгініп, қажет аттың қолжетімді екенін және оның басқа біреудің сауда маркасы еместігін тексеру жеткілікті. Егер барлығы ойдағыдай болса, сұраныс беруші тіркеліп, кішігірім әр жылдық абоненттік ақы үшін екінші деңгейдегі домендік атқа ие болады.

7.1-кесте. Жоғары деңгейдің рулық домендері

Домен	Қолданылуы	Орнаған жылы	Шектелген?
com	Коммерциялық мақсат	1985	Жоқ
edu	Білім мекемесі	1985	Иә
gov	Мемлекет	1985	Иә
int	Халықаралық ұйымдар	1985	Иә
mil	Әскери ұйым	1985	Иә
net	Желілік провайдерлер	1985	Жоқ
org	Коммерциялық емес ұйымдар	1985	Жоқ
aero	Әуе көлігі	2001	Иә
biz	Бизнес	2001	Жоқ
coop	Кооперативтер	2001	Иә
info	Ақпарат	2002	Жоқ
museum	Музей	2002	Иә
name	Адамдар	2002	Жоқ
pro	Мамандар	2002	Иә
cat	Каталония	2005	Иә
jobs	Жұмыс	2005	Иә
mobi	Мобильді құрылғылар	2005	Иә
tel	Байланыс ақпараты	2005	Иә
travel	Саяхат индустриясы	2005	Иә
xxx	Секс-индустрия	2010	Жоқ

Алайда, Интернеттің коммерциялану және интернационалануына байланысты көптеген даулы мәселе туындай бастады, әсіресе домендерге ат беруде. Бұл мәселе ICANN-ның өзіне де байланысты. Мысалы, xxx доменін құрастыру бірнеше жылға созылды және сот отырыстарына әкелді. Порно-контентті жеке доменде орналастыру – жақсы ма, әлде жаман ба? (Кейбіреулер Интернетте порнографиялық сайттың болмағанын қалады, ал басқалары оны жеке доменге орналастырып, ата-ана бақылай алатын программа көмегімен оқшаулау мүмкіндігінің болғанын қалады.) Кейбір домендер өздігінен ұйымдастырылған, ал басқаларында шектеулер бар және кез келген атты ала алмайды (*7.1-кесте* көрсетілгендей). Қандай шектеулер орынды? Мәселен, *pro* доменін алайық. Ол білікті мамандарға арналған. Бірақ, кім маман, кім маман емес? Әрине, дәрігер және адвокат – маман, мұнда талас жоқ. Ал еркін фотографтарды, музыка мұғалімін, су тартқыштарды, қоқыс жинаушыларды, татуировка жасаушыларды, жалшы мен жезөкшелерді не істейміз? Осы және басқа да мамандықтардың білікті мамандары *pro* доменін алуға құқығы бар ма? Кім бұны анықтау керек?

Сонымен бірге, аттарды кіріс көзіне айналдыруға болады. Мысалы, Тувалу елі өз доменін *tv*, теледидар сайттарын жарнамалауға өте ыңғайлы болғандықтан оны $50 млн жалға берген. Іс жүзінде барлық жалпы қолданыстағы ағылшын сөздері, жиі жаңсақ басылулармен, *com* ішкі домендерінің аты ретінде пайдаланылады. Үй тұрмысы, өсімдік, дене бөлігі және т.б. аттардың кез келгенін теріп көріңіз. Тіркеу тәжірибесінде домендік атты сату үшін тіркейтіндерге берген аттары да бар – **киберсвоттинг (cybersquatting)**. Көптеген компаниялар өздерінің шырақсыздығының салдарынан, Интернет дәуірі басталғаннан кейін тіркелмек болған кезде, ең айқын домендік аттардың тіркеліп қойылғанын анықтады. Жалпы алғанда, егер тауарлық ат құқығы бұзылмаса және айлакерлік әрекеттер орын алмаса, онда «бірінші сұрады – бірінші болып алды» ережесі жұмыс істейді. Дегенмен, аттарға байланысты дауды шешу саясаты әлі де тұрақталған жоқ.

Әр домен аты, файлдық жүйедегі файлдың толық аты тәрізді, осы доменнен басталған соңғы нүктеге (атсыз) дейінгі жолдан тұрады. Жол бөліктері нүктемен ажыратылады. Мәселен, Cisco корпорациясы техникалық бөлімінің домені UNIX стилінде *(/com/cisco/eng*) қабылданғандай емес, *eng.cisco.com* болады. Ат берудің осындай иерархиялық жүйесіне байланысты *eng.cisco.com* доменіндегі *eng* аты, Вашингтон университетінің ағылшын тілі факультетін білдіретін *eng.washington.edu* доменімен шатастырылмайды.

Домен аттары абсолютті және салыстырмалы болуы мүмкін. Абсолютті домен аттары үнемі нүктемен аяқталады (мысалы, *eng.cisco.com*), ал салыстырмалы нүктемен аяқталмайды. Салыстырмалы аттардың нақты мағынасын жалғыз рет анықтау үшін, ол қандай да бір мәнмәтінде түсіндірілуі тиіс. Кез келген жағдайда ат берілген домен ағаштың белгілі бір түйінін білдіреді және онан кейінгі түйіндер оның астында орналасады (қарамағында).

Домен аттары символ регистрінің өзгеруіне сезімтал емес. Мысалы, *edu, Edu* және *EDU* бәрінің мағынасы бір. Ат бөліктерінің ұзындығы 63-ке дейін жетуі мүмкін, ал толық жол ұзындығы 255 символдан аспауы керек.

Іс жүзінде домендерді алғашқы рулық домендер арқылы да, ел домендері арқылы да қосуға болады. Мысалы, *cs.washington.edu* еш қиындықсыз *us* доменіне *cs.washington.wa.us* деген атпен орналастыруға болады. АҚШ ұйымдарының барлығы іс жүзінде рулық домендер астында, ал АҚШ-тан тыс орналасқан ұйымдар мемлекет домендерінің астында орналасқан. Бірнеше жоғарғы деңгей домендерінің астында тіркелуге шектеу қоятын ешқандай ереже жоқ. Көптеген компаниялар осылай істейді де (мысалы, *sony.com, sony.net* және *sony.nl*).

Әр домен өз астында орналасқан домендерді таратуды басқарады. Мысалы, Жапонияда *ac.jp* және *co.jp* домендері американдық *edu* және *com* домендеріне сәйкес келеді. Голландияда мұндай ерекшеліктер жоқ және барлық ұйымдар домені бірден *nl* доменінің астында орналасады. Мысал ретінде үш университеттің есептеу техникасы факультеттерінің домендер атын келтірейік («компьютерлік ғылымдар» – computer science):

1. *cs.wasington.edu* (Вашингтон университеті, АҚШ);
2. *cs.vu.nl* (Врийе университеті, Нидерландия);
3. *cs.keio.ac.jp* (Кейо университеті, Жапония).

Жаңа домен құрастыру үшін, ол орналасатын домен рұқсаты қажет. Мысалы, егер Вашингтон университетінде *ulsi.cs.washington.edu* доменін тіркегісі келетін VLSI тобы пайда болса, онда оған *cs.washington.edu* доменін басқарушының рұқсаты қажет. Сәйкесінше, егер жаңа университет құрастырылатын болса, мысалы, Солтүстік-Оңтүстік Дакота университеті, онда ол *edu* доменінің менеджерінен өз доменіне *unsd.edu* атын тағайындауын (егер ол таратылып кетпесе) сұрауы керек. Осылайша, аттар шиеленісін орын алдырмауға болады, ал әр домен өз ішкі домендерінің қалып-күйін қадағалайды. Домен құрастырылып, тіркелгеннен кейін онда ішкі домендер құрастырылуы мүмкін, мысалы, *cs.unsd.edu* домені, ол үшін жоғарыда тұрған доменнен рұқсат алу керек.

Домендер құрылымы желінің физикалық құрылымын емес, ұйымдар және олардың ішкі бөлімдері арасындағы логикалық бөлінуін бейнелейді. Мысалы, егер есептеу техникасы және электротехника факультеттері бір ғимратта орналасып, бір жергілікті желіні пайдаланатын болса да олардың домендері әртүрлі болуы мүмкін. Керісінше, айталық, есептеу техникасы факультеті жергілікті желілері әртүрлі университеттің екі ғимаратында орналасса да, екі ғимараттың хосттары да логикалық түрде бір доменге жатады.

7.1.2. Домен ресурстарының жазбалары

Әр доменнің оның жалғыз хост немесе жоғары деңгей домені екеніне қарамастан онымен байланысқан **ресурстар жазбасы (resource records)** болуы мүмкін. Бұл жазбалар DNS деректер базасы болып саналады. Жалғыз хост үшін ресурстар жазбасы жиі оның жай ғана IP-адресі болып келеді, бірақ көптеген басқа ресурстар жазбасы да бар. Анықтауды қажет ететін домен атын DNS-серверге жіберген кезде, ол жауап ретінде осы домен атымен байланысты ресурстар жазбасын алады. Сөйтіп, DNS жүйесінің нақты қызметі – домен аттарын ресурстар жазбасына түрлендіру.

Ресурс жазбасы бес бөліктен тұрады. Тиімділік үшін олар жиі екілік кодқа түрлендірілсе де, көптеген ресурс жазбалары сипаттамасы ASCII-мәтіні ретінде берілген – әр ресурс жазбасына бір жол беріледі. Біз:

```
Domain_name    Time_to_live    Class   Type   Value
```

форматты пайдаланатын боламыз.

Domain_name (домен аты) өрісі ағымдағы жазба доменін білдіреді. Әдетте, әр домен үшін бірнеше ресурстар жазбасы болады және деректер базасының әр көшірмесінде бірнеше домен ақпараттары сақталады. Домен аты өрісі,

сұранысты орындау кезінде пайдаланылатын іздеудің алғашқы кілті болып саналады. Деректер базасындағы жабалар реті маңызды емес. Домен жайлы сұранысқа оны қанағаттандыратын қажет класс жазбаларының барлығы қайтарылады.

Time_to_live (өмір уақыты) өрісі жазба қалып-күйі қаншалықты тұрақты екенін білдіреді. Сирек өзгеретін деректерге осы өрістің жоғары мәні беріледі, мысалы, 86 400 (тәуліктегі секундттар саны). Тұрақсыз ақпарат төмен мәнмен белгіленеді, мысалы, 60 (1 минут). Біз бұл сұраққа кэштеуді қарастырған кезде қайтып ораламыз.

Әр жазбаның үшінші өрісі *Class (класс)* өрісі. Интернеттегі ақпарат үшін бұл өрістің мәні үнемі *IN*. Басқа ақпараттар үшін басқа кодтар пайдаланылады, алайда, іс жүзінде олар сирек кездеседі.

Type (тип) өрісі DNS-жазба типін білдіреді. Олар өте көп. Маңызды жазбалар типі *7.2-кестеде* келтірілген.

7.2-кесте. DNS ресурстар жазбасының негізгі типтері

Типі	Мағынасы	Мәні
SOA	Бөлімнің бастапқы жазбасы	Осы бөлім параметрлері
A	Хостың IPv4-адресі	Бүтін сан, 32 екілік разряд
AAAA	Хостың IPv6-адресі	Бүтін сан, 128 екілік разряд
MX	Поштамен алмасу	Доменнің электронды пошта қабылдау басымдылығы
NS	Аттар сервері	Осы домен үшін сервер аты
CNAME	Каноникалық ат	Домен аты
PTR	Көрсеткіш	IP-адрес бүркеме аты
SPF	Пошта жіберу ережесі	Мәтіндік түрде кодталған пошта жіберу ережесі
SRV	Қызмет	Осы қызметті көрсетуші хост
TXT	Мәтін	Түсіндірілмейтін ASCII-мәтін

SOA (Start Of Authority – өкілдіктің бастапқы нүктесі) жазбасы аттары сервері бөліміндегі (төменде сипатталған) бастапқы ақпарат көзін хабарлайды, оның әкімшілік электронды поштасы, бірегей реттік нөмірі, түрлі жалаушалар және тайм-ауттар көрсетілген.

Ең маңыздысы ***A* (Address – адрес)** жазбасы. Онда хост үшін 32-разрядты IPv4-адрес интерфейсі көрсетіледі. Сәйкесінше ***AAAA*** («quad A» – «төрт А») жазбасында 128-разрядты IPv6-адрес бар. Басқа машиналар хостпен байланысу үшін, Интернетте әр хосттың кем дегенде, бір IP-адресі болу керек. Кейбір хосттарда бір мезгілде бірнеше желілік байланыстар орнатылуы мүмкін. Бұл жағдайда оларға екі немесе оданда көп *A* немесе *AAAA* жазбасы қажет. Сәйкесінше, DNS бір атқа бірнеше адрес бере алады.

MX жазбасы стандартты болып саналады. Мұнда көрсетілген домен үшін поштаны қабылдауға дайын хост аты көрсетіледі. Себебі кез келген машина поштаны қабылдаумен айналыса алмайды. Егер кімде-кім, мысалы, *bill@ microsoft.com* адресіне хат жазғысы келсе, онда жөнелтуші хост алдымен *microsoft.com* пошталық серверін табуы керек. MX жазбасы оны іздеуде көмектеседі.

Тағы бір маңызды жазба типі – *NS*. *NS* жазбасында домен немесе ішкі домен үшін аттар сервер жайлы мәлімет жазылған. Бұл – домен үшін деректер базасының көшірмесі сақталған хост. Ол аттарды іздеу барысында қолданылады, сондықтан біз бұл үдерісті қысқаша сипаттап кетеміз.

CNAME жазбасы бүркеме ат жасауға мүмкіндік береді. Айталық, сырттан таныс адам, Интернетте аттар құрастырылғаннан кейін, Массачусетск технологиялық институтының (M.I.T.) есептеу техникасы бөліміндегі *paul* тұтынушысына мәлімдеме жөнелткісі келді делік. Ол өзіне қажет адресті таппақ болып, *paul@cs.mit.edu* жолын құрастыруы мүмкін. Алайда, бұл адрес жұмыс істемейді, себебі Массачусетск технологиялық институты, есептеу техникасы бөлімінің домені *csail.mit.edu* деп аталады. Сонымен, M.I.T. білмейтіндерге ыңғайлы болу үшін, екі ат бойынша да қажет доменге жүгінуге мүмкіндік беретін *CNAME* жазбасы құрастыруға болады. Мұндай жазбаның түрі:

```
cs.mit.edu   86400   IN  CNAME   csail.mit.edu
```

болады.

CNAME жазбасы тәрізді *PTR* де басқа атқа нұсқайды. Алайда, іс жүзінде макроанықтауыш (яғни бір жолды екіншісімен алмастыру механизмі) болып келетін *CNAME* айырмашылығы – *PTR,* әдеттегі DNS деректер типі, оның бірігуі мәнмәтінге байланысты. Іс жүзінде *PTR* жазбасы үнемі IP-адреспен байланыстыру үшін қолданылады, бұл IP-адрес бойынша сәйкес машинаның атын анықтауға мүмкіндік береді. Бұл **кері іздеу** (**reverse lookups)** деп аталады.

SRV жазбасы – домендегі қажет қызметті анықтайтын жаңа тип. Мысалы, *cs.washington.edu* үшін веб-сервер *cockatoo.cs.washington.edu* түрінде анықталуы мүмкін. Бұл жазба MX жазбасының кеңейтілімі болып саналады және пошталық қызмет шеңберінде сол қызметі қайталайды.

SPF – бұл да жаңа жазба типі. Ол доменге өзі арқылы қандай машиналар Интернеттің басқа бөлігіне хат жөнелтетінін кодтауға рұқсат береді. Бұл қабылдаушы машиналарға келген поштаның ұйғарымды екенін тексеруге мүмкіндік береді. Егер пошта *dodgy* деп аталатын машинадан келсе, ал домендік жазбада пошта тек *smtp* деп аталатын машинадан жіберіледі деп жазылса, онда бұл мәлімдеменің спам болу ықтималдығы жоғары.

Тізімдегі соңғы тип TXT-жазбасы. Алғашқыда ол домендер өздерін еркін түрде сәйкестендіру үшін қарастырылған болатын. Қазір оның көмегімен машина оқуға арналған ақпарат кодталады, әдетте, бұл – *SPF*-ақпарат.

Ресурстар жазбасының ең соңғы өрісі – *Value (мән)* өрісі сан, домен аты немесе мәтіндік ASCII-жолы болуы мүмкін. Өрістің мағынасы жазба типіне

байланысты. *Value* өрісінің қысқаша сипаттамасы әр негізгі тип үшін *7.2-кестеде* келтірілген.

Доменнің DNS деректер базасында сақталатын ақпарат мысалы *7.1-листингте* келтірілген. Онда, *7.1-суретте* домендер ағашының түйіні ретінде бейнеленген, *cs.vu.nl* доменінің деректер базасының (болжамалы) бір бөлігі көрсетілген. Деректер базасында ресурстар жазбасының жеті типі берілген.

7.1-листинг. *cs.vu.nl* доменінің мүмкін деректер базасының бөлігі

```
; cs.vu.nl үшін ресми ақпарат
cs.vu.nl.       86400   IN      SOA     star boss
                                        (9527,7200,7200,241920,86400)
cs.vu.nl.       86400   IN      MX      1 zephyr
cs.vu.nl.       86400   IN      MX      2 top
cs.vu.nl.       86400   IN      NS      star

star            86400   IN      A       130.37.56.205
zephyr          86400   IN      A       130.37.20.10
top             86400   IN      A       130.37.20.11
www             86400   IN      CNAME   star.cs.vu.nl
ftp             86400   IN      CNAME   zephyr.cs.vu.nl

flits           86400   IN      A       130.37.16.112
flits           86400   IN      A       192.31.231.165
flits           86400   IN      MX      1 flits
flits           86400   IN      MX      2 zephyr
flits           86400   IN      MX      3 top

rowboat                 IN      A       130.37.56.201
                        IN      MX      1 rowboat
                        IN      MX      2 zephyr


little-sister           IN      A       130.37.62.23

laserjet                IN      A       192.31.231.216
```

7.1-листингінің түсініктемеден кейінгі бірінші жолында домен жайлы негізгі ақпарат келтірілген. Келесі екі жол, *person@cs.vu.nl* адресі бойынша электронды поштаны жеткізбек болған кезде алдымен байланысуды қажет

ететін екі хостты анықтайды. *Zephyr* (арнайы машина) атты хостты алдымен сұрау керек. Сұрау сәтсіз аяқталған кезде хатты *top* атты машинаға жеткізу тиіс. Келесі жолда *star* домені үшін аттар сервері анықталған.

Оқуға ыңғайлы болу үшін қосылған бос жолдан кейін, *star, zephyr* және *top*-қа арналған IP-адрестерді хабарлайтын жолдар орналасқан. Әрі қарай, қандай да бір нақты машинаға жүгінуге мүмкіндік беретін *www.cs.vu.nl* бүркеме ат орналасқан. Бұл бүркеме атты құрастыру *cs.vu.nl* доменіне, тұтынушы өзіне жүгінетін адресті өзгертпей, WWW-серверін өзгертуге мүмкіндік береді. Бұл *ftp.cs.vu.nl* домені үшін де – FTP-сервер – ақиқат.

Flits машинасына арналған бөлімде екі IP-адреске және *flits.cs.vu.nl* адресіне жөнелтілген, поштаны өңдеуге арналған үш мүмкін деген адрестер көрсетілген. Әрине, хатты алдымен *flits* компьютерінің өзіне жеткізуге тырысу керек. Бірақ егер хост өшірілген болса, онда *zephyr* және *top* хостарына жүгіну қажет.

Келесі үш жол компьютерлерге арналған әдеттегі жазбалар, біздің жағдайда *rowboot.cs.vu.nl* үшін. Деректер базасында сақталған ақпарат IP-адрестен, сонымен бірге поштаны жеткізуге арналған бірінші және екінші хосттар атынан тұрады. Артынша, өзі поштаны қабылдай алмайтын машина жайлы жазба орналасқан. Соңғы жол, Интернетке қосылған лазерлік принтерді сипаттауы мүмкін.

7.1.3. Аттар сервері

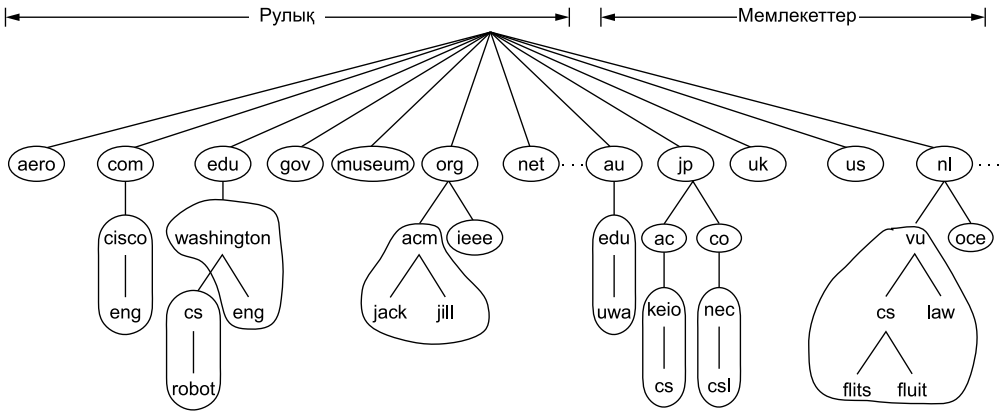
Теорияда бір сервер бүкіл DNS деректер базасын жинақтап, оған келген сұраныстардың барлығына жауап бере алар еді. Іс жүзінде бұл сервердің асыра жүктелгендігі соншалықты, одан ешқандай пайда болмас еді. Одан бетер, егер сервер істен шықса, онда бүкіл Интернет жұмыс істемей қалар еді.

Бүкіл ақпаратты бір жерде сақтаумен байланысты мәселені болдырмас үшін, DNS аттарды сақтау кеңістігі бір-бірімен қиылыспайтын **аймақтарға (zones)** бөлінген. *7.1-суретте* көрсетілген аттар кеңістігін аймақтарға бөлудің бір жолы *7.2-суретте* бейнеленген. Әрбір сызылған аймақта домендер ортақ ағашының бір бөлігі жазылған.

Аймақтар шекарасын орнату толығымен аймақ әкімшілігіне тәуелді. Бұл шешім аймаққа неше аттар сервері керек екендігіне негізделіп қабылданады. Мысалы, *7.2-суретінде* Вашингтон университеті аймағында жеке бөлігінде өз аттар серверімен орналасқан *cs.washington.edu* доменімен емес, *eng.washington.edu* доменімен басқарылатын *washington.edu* аймағы бар. Мұндай шешім ағылшын тілі факультеті өз аттар серверін басқарғысы келмесе, ал оны есептеу техникасы факультеті басқарғысы келген жағдайда қабылдануы мүмкін.

Әр аймақ бір немесе одан көп аттар северімен сәйкестендіріледі. Бұл – аймақтарға арналған деректер базасы орнатылған хосттар. Әдетте, аймақтың

ақпаратты өз дискісіндегі файлдан алатын бір негізгі аттар сервері және бір немесе бірнеше екінші дәрежедегі аттар сервері бар, олар ақпаратты негізгі аттар серверінен алады. Сенімділікті күшейту үшін кейбір аттар сервері аймақтан тыс орналасуы мүмкін.

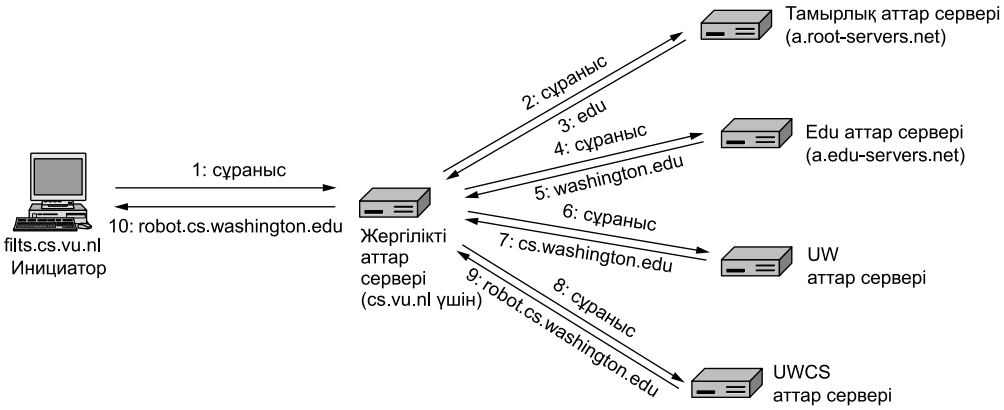


7.2-сурет. Сызылған аймақтарға бөлінген DNS аттар кеңістігінің бір бөлігі

Аты бойынша адресті іздеу үдерісі **аттарды шешу (name resolution)** деп аталады. Танушы домен атын шешуді сұрап жергілікті аттар серверіне жүгінеді. Егер ізделінуші домен осы аттар сервердің жауапкершілігі шеңберінде болса, мысалы, *top.cs.vu.nl* домені *cs.vu.nl* доменінің заң шеңберіне кіреді, онда осы DNS-сервер танушы сұранысына ресурстың **беделді жазбасын (authoritative record)** беріп, өзі жауап қайтарады. Жазба сақталып және оның қалып-күйін басқаратын ресми ақпарат көзінен алынатын жазбаны беделді деп атайды. Сондықтан бұндай жазба, ескіріп қалатын **кэштелген жазбаларға (cached records)** қарағанда әрдайым дұрыс жазба деп саналады.

Алайда, егер қашықтықтағы *flits.cs.vu.nl* тәрізді домен Вашингтон университетіндегі *robot.cs.washigton.edu* үшін IP-адрес таппақ болған жағдай да не болады? Бұл жағдайда, жергілікті қолжетімді кэште сұралған домен жайлы ақпарат жоқ болса, аттар сервері қашықтыққа сұраныс жөнелтеді. Бұл үдерісті *7.3-суретте* бейнеленген мысалмен түсіндірейік. Бірінші қадамда («1» деп белгіленген) жергілікті аттар серверіне сұраныс жөнелтіледі. Бұл сұраныста ізделінетін домен типі (*A*) және класс (*IN*) көрсетілген.

Келесі қадамда иерархия төбесіндегі **тамырлық аттар серверінің (root name servers)** біріне сұраныс жөнелтіледі. Бұл аттар серверінде жоғарғы деңгей доменінің әрқайсысы жайлы ақпарат сақталады. Бұл сұраныс, *7.3-суреттегі* 2-қадамда көрсетілген. Тамырлық сервермен байланысушы үшін, әр серверде осындай бір немесе бірнеше аттар сервері жайлы ақпарат болуы керек. Әдетте, бұл ақпарат DNS-сервер іске қосылған кезде DNS кэшіне жүктелетін жүйелік конфигурация файлында көрсетіледі. Ол – жай *NS* жазбасы және сәйкес *A* жазбасының тізімі.



7.3-сурет. Танушының қашықтықтағы хост атын іздеудегі он қадамы

Аттары қарапайым *a.root-servers.net*-тен *m.root-servers.net*-ке дейін 13 DNS тамырлық сервері бар. Әр тамырлық сервер логикалық түрде бір жеке компьютер бола алар еді. Алайда, бүкіл Интернет тамырлық серверге тәуелді болғандықтан, олар өте қуатты машиналар, ал ол жердегі ақпарат бірнеше рет көшіріледі. Көптеген серверлер әртүрлі географиялық нүктелерде орналасқан және қол жеткізу осы топтағы кез келген құрылғыны адрестеу арқылы жүргізіледі, ал десте жақын адреске жеткізіледі (біз топтағы кез келген құрылғыны адрестеуді бесінші тарауда сипаттағанбыз). Ақпаратты көшіру – сенімділік пен өнімділікті жоғарылатады.

Бұл тамырлық сервердің Вашингтон университетіндегі машина адресін білуі екіталай. Тіпті, ол университеттің аттар серверінің адресін де білмеуі мүмкін, бірақ ол *cs.washington.edu* орналасқан *edu* доменінің аттар серверін білуі тиіс. Үшінші қадамда ол атты және IP-адресті жауап бөлігі ретінде қайтарады.

Жергілікті аттар сервері бұл күрделі жолды әрі қарай жалғастырады. Ол Вашингтон университеті серверінің атын беретін *edu* (*a.edu-servers.net*) аттар серверіне сұраныс жөнелтеді. Бұл үдеріс 4 және 5-қадамдарда бейнеленген. Біз қажет жерге жақындап қалдық. Жергілікті аттар сервері Вашингтон университеті серверіне сұраныс жөнелтеді (6-қадам). Егер ізделінген домен аты ағылшын тілі факультетінде болса, жауап алынады, себебі университет аймағы бұл факультетті қамтиды. Бірақ есептеу техникасы факультеті өзінің жеке аттар серверін іске қосқан, сондықтан сұраныс Вашингтон университеті есептеу техникасы факультеті аттар серверін және IP-адресін қайтарады (7-қадам).

Соңында, жергілікті аттар сервері Вашингтон университеті есептеу техникасы факультетінің аттар серверіне сұраныс жөнелтеді (8-қадам). Бұл сервер *cs.washington.edu* домені үшін жауап береді, сондықтан ол жауап қайтаруы тиіс. Соңында тұжырымды жауып қайтып келеді (9-қадам) және жергілікті аттар сервері оны *flits.cs.vu.nl* береді (10-қадам). Ізделінген ат қабылданды.

Сіз бұл үдерісті көптеген UNIX-жүйелерде орнатылған *dig* тәрізді стандартты программаларды пайдаланып оқи аласыз. Мысалы,

```
dig @a.edu-servers.net robot.cs.washington.edu
```

сияқты жолды теріп, сіз *a.edu-servers.net* аттар серверіне *robot.cs.washington.edu* сұранысын жөнелтіп, жауапты басып шығарасыз. Сөтіп, сіз біздің мысалымыздың 4-қадамында алынған жауапты көресіз және Вашингтон университеті аттар серверінің IP-адрестерін және аттарын білесіз.

Бұл ұзын-сонар сценарийдің түсініктемені қажет ететін үш техникалық сәті бар. Біріншіден, *7.3-суретте* сұраныстың екі түрлі механизмі қолданылады. *flits.cs.vu.nl* хосты жергілікті аттар серверіне сұранысы жөнелткен кезде, бұл сервер сұранысты қайтаратын жауап келгенше *flits* атынан орындайды. Ол жартылай жауаптарды қайтармайды. Олар пайдалы болуы мүмкін, бірақ сұраныстарда ол жайлы айтылмаған. Бұл механизм **рекурсивті сұраныс** (**recursive query)** деп аталады.

Екінші жағынан, тамырлық аттар сервері (әр келесі сұранысқа) жергілікті аттар серверінің рекурсивті сұранысын жалғастырмайды. Ол тек жауап бөлшегін қайтарып, келесі сұранысқа көшеді. Жергілікті аттар сервері жауапты іздеуді жалғастырып, келесі сұраныстарды жөнелтеді. Бұл механизм **қайталамалы сұраныс (iterative query)** деп аталады.

Мысалда көрсетілгендей, атты іздеудің бір үдерісінде екі механизм де қолданылуы мүмкін. Іс жүзінде рекурсивті механизм жақсырақ болып көрінеді, алайда, көптеген серверлер (әсіресе, тамырлық) оны өңдемейді. Олар онсызда асыра жүктелінген. Қайталамалы сұраныс – сұранысты өңдеу жүгін оны туындатқан машинаға артады. Жергілікті аттар серверіне өз доменіндегі хосттарға қызмет көрсету үшін рекурсивті сұранысты қолданған ыңғайлы. Бұл хосттардың барлық аттар серверін аралап шығатындай ұйымдастырылуы міндетті емес, оларға тек жергілікті серверге жүгіну мүмкіндігі болса жеткілікті.

Назар аударатын екінші мәселе – кэштеу. Барлық жауаптар, сонымен бірге бөлшектеніп қайтарылған жауаптар кэште сақталады. Сөйтіп, егер *cs.vu.nl*-дің басқа хосты *robot.cs.washington.edu*-ге сұраныс берсе, онда жауап белгілі. Одан бетер, егер хостты сол домендегі басқа хост сұраса, мысалы, *galah.cs.washington.edu*, онда жауапты тікелей осы атқа жауап беретін аттар серверіне жөнелтуге болады. Сәйкесінше, *washington.edu*-дегі басқа да домендерге келген сұраныстар бірден *washington.edu* доменінен бастауы мүмкін. Кэште сақталған жауаптарды пайдалану сұраныстағы қадамдарды едәуір азайтады және өнімділікті жоғарылатады. Біз сипаттаған сценарий мүмкін деген нұсқалардың ішіндегі ең нашары, себебі кэште пайдалы ақпарат жоқ болып саналады.

Алайда, кэште сақталған жазбалар беделді болып есептелмейді, себебі *cs.washington.edu* доменіндегі өзгерістер осы домен ақпарат көшірмесі сақталған кэштерге автоматты түрде таратылмайды. Сондықтан кэш жазбасы ұзақ өмір сүрмейді. Әр жазбаның *Time_to_live* өрісі бар. Ол қашықтықтағы серверлерге жазбаны кэште қаншалықты ұзақ сақтау керек екендігін хабарлайды. Егер

қандай да бір машина адресін жылдар бойы тұрақты сақтаса, бұл ақпаратты кэште бір күн бойы сақтау жеткілікті. Тұрақсыз ақпараттарды, бірнеше секундтан немесе бір минуттан кейін жойып отыру дұрысырақ болады.

Біз назар аударғымыз келген үшінші сәт – бұл сұраныс және жауап үшін қолданылатын, транспорттық деңгей хаттамасы – UDP. DNS-мәлімдемелер сұраныстар, жауаптар және сервер аты үшін қарапайым форматы бар және жауап алу үдерісін жалғастыруда пайдалануға болатын UDP-дестелерімен жөнелтіледі. Біз бұл форматтың егжей-тегжейіне үңілмейміз. Егер белгілі бір уақыт аралығында (ұзақ емес) жауап келмесе, DNS-клиент сұранысты қайталайды және бірнеше талпыныстан кейін доменнің басқа серверін тексереді. Бұл үдеріс, серверлердің бірі істен шықса немесе дестелердің бірі жоғалса да жұмыс тоқталмайтындай етіп құрастырылған. Әр сұранысқа 16-биттік идентификатор енгізілген, ол жауапқа көшіріледі, сөйтіп, аттар сервері тіпті, бір мезгілде көп сұраныстар түссе де, сұранысқа сәйкес қажет жауапты бере алады.

DNS-ті құрастыру мақсаты қарапайым және түсінікті болса да, бұл бірлесіп жұмыс істейтін миллиондаған аттар серверінен тұратын, өте үлкен және күрделі, таратылған жүйе. DNS тұтынушыларға оқуға ыңғайлы домен аттары және машинаның IP-адрестері арасындағы кілтті байланысты құрастырады. Ол жоғары өнімділік және сенімділікке бағытталған ақпаратты көшіруді және кэштеуді қоса орындайды және өте функционалды болатындай етіп құрастырылған.

Біз қауіпсіздік жайлы ештеңе жазған жоқпыз, алайда, атты адреске және кері түрлендіруді зиян келтіру үшін қолданылса, салдары өте жағымсыз болатынын өздеріңізде түсінетін шығарсыздар. Осы себептен DNS үшін, қауіпсіздікті қамтамасыз ететін DNSsec деп аталатын кеңейтілім құрастырылды. Біз оны сегізінші тарауда сипаттаймыз.

Бұдан басқа, кейде қосымшаларға атты икемді пайдалануға тура келеді, мысалы, контентті атап, осы контент бар жақындағы хосттың IP-адресіне жүгінуге тура келеді. Фильмдерді іздеу және жазып алу да осылайша жүргізіледі. Бұл жағдайда бізді фильмнің көшірмесі бар компьютер емес, фильм өзі қызықтырады. Сондықтан бізге жақын арадағы осы фильм жазылған кез келген компьютердің IP-адресі қажет. Оң нәтижеге қолжеткізу үшін контентті жеткізу желісін (CDN) пайдалануға болады. Мұны DNS көмегімен қалай жүзеге асыруға болатынын біз *7.5-бөлімде* қарастырамыз.

7.2. ЭЛЕКТРОНДЫ ПОШТА

Электронды пошта немесе жиірек **e-mail** деп аталады – осыдан үш он жылдық бұрын пайда болды. Ол Интернет дамуының бірінші күнінен бастап, өзінің жылдамдығы және арзандығы арқасында танымал болып келеді. 1990 жылға дейін ол тек ғылыми ұйымдарда қолданылды. Тоқсаныншы жылдары ол кеңінен танымал бола бастады және содан бері электронды пош-

та арқылы жөнелтілетін хаттар саны экспоненциалды түрде өсе бастады. Күн сайын жөнелтілетін мәлімдемелердің орташа саны **әдеттегі пошта (snail mail)** көмегімен жөнелтілетін хаттар санынан бірнеше есе асып түседі. Соңғы он жылдықта лездік мәлімдеме және IP-телрефония тәрізді желілік коммуникацияның басқа түрлері де кеңінен тарала бастады, алайда, интернет-коммуникацияның «жегін аты» әлі де электронды пошта. Бүгінде электронды пошта өндірістегі компаниялар арасындағы ақпарат алмасу үшін кеңінен қолданылады. Ол қашықтықтағы қызметшілердің күрделі жобаны бірігіп орындауына мүмкіндік береді. Өкінішке орай, әдеттегі пошта тәрізді электронды поштаның 10-мәлімдемесінің 9-ы **спам (spam)** болып келеді (McAfee, 2010).

Кез келген коммуникация тәрізді электронды поштаның өз стилі және келісімдер жиынтығы бар. Жеке алғанда, электронды пошта арқылы хабарласу бейресми және демократиялық сипатта жүргізіледі. Айталық, қандайда бір Ерекше Дәрежелі кісіге (VIP) қоңырау шалып немесе жай қағаз хат жолдай алмайтын адам оған немқұрайлы жазылған электронды мәлімдеме жолдай алады. Көбіне, e-mail арқылы хат алысу, хат алысушылар мәртебесіне емес, хат мазмұнына негізделген, сондықтан мұнда лауазым, жас және жыныс аралық мәселелердің барлығы жойылады. Электронды пошта арқасында студент-практикант жөнелткен жақсы идея, мекеме вице-президенті ұсынған идеядан да озық шығып, жеңіске жетуі мүмкін.

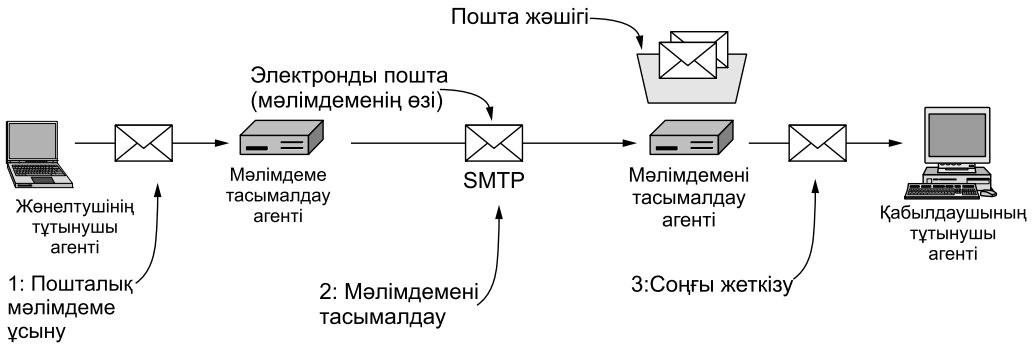
Электронды поштада адамдар ерекше жаргонды және BTW (By The Way - айтпақшы), ROTFL (Rolling On The Floor Laughing – күлкіден жерде аунап жатырмын), IMHO (In My Humble Option – менің сыпайы пікірім бойынша) және т.б. қысқартуларды қолданғанды жақсы көреді. Одан басқа, бәрімізге белгілі «:-)» белгісінен бастап, әртүрлі **смайликтер (smileys)** өте танымал. Осы және басқа да **эмотикондар** мәлімдеменің сарынын білдіруге мүмкіндік береді. Олар тек электронды пошта хаттарында ғана емес, лездік мәлімдемелерде де толып кетті.

Смайликтерде пайдалану уақытынан бері электронды пошта хаттамалары да өзгерді. Алғашқы электронды пошта жүйесі тек файлдарды тасымалдау хаттамасынан және мәлімдеменің бірінші жолында (демек, файлдың) қабылдаушы адресін көрсету келісімінен тұратын. Уақыт өте электронды пошта файл тасымалдаудан аулақтап кетті және бір хатты бірнеше адресатқа жөнелту тәрізді көптеген опциялар қосылды. Тоқсаныншы жылдары мультимедиа, яғни суреттер бар мәлімдемелер және басқа да мәтіндік ақпаратты жөнелту мүмкіндігі пайда болды. Поштаны оқу программалары күрделене түсті, негізгі мәтіннен графикалық тұтынушы интерфейсіне көшу, тұтынушыларға қай жерде жүргеніне тәуелсіз ноутбуктер арқылы поштаға қол жеткізуге мүмкіндік берді. Соңында спамдардың көбеюі, қазіргі заманғы пошта оқу программаларын және бір поштаны жөнелту хаттамаларын, қажетсіз мәлімдемелерді анықтап, жою мәселесіне көңіл аударуға әкелді.

Біз электронды поштаны сипаттау барысында, электронды поштаның сыртқы түрі мен оны оқуға арналған программалар ерекшелігіне емес, мәлімдемелердің тұтынушылар арасында жөнелтілу тәсілдеріне назар аударатын боламыз. Дегенмен, құрылымды толығымен сипаттап, біз пошта жүйесінің тұтынушы көретін және көптеген оқырмандарға белгілі бөлігіне көшеміз.

7.2.1. Құрылым және қызметтер

Бұл бөлімде біз электронды пошта мүмкіндіктері және ұйымдастырылуын қарастырамыз. Пошта жүйесінің құрылымы *7.4-суретте* бейнеленген. Электронды пошта жүйесі (e-mail жүйесі): тұтынушыға электронды поштаны оқуға және жөнелтуге мүмкіндік беретін **тұтынушы агенттері (user agents)** және мәлімдемені жөнелтушіден қабылдаушыға тасымалдайтын мәлімдемелерді **тасымалдау агенттері (message transfer agents)** сияқты екі ішкі жүйеден тұрады. Біз агенттерді бейресми түрде **пошталық серверлер (mail servers)** деп атаймыз.



7.4-сурет. e-mail жүйесінің құрылымы

Тұтынушы агенттері – тұтынушыға электронды пошта жүйесімен әрекеттесуге мүмкіндік беретін, мәтін немесе командаларға негізделген графикалық интерфейс немесе интерфейсті ұсынатын программалар. Оған мәлімдеме және мәлімдемеге жауап жазу, кіріс мәлімдемелерді бейнелеу, хаттарды бумаларға орналастыру, іздеу және жою құралдары кіреді. Жаңа мәлімдемелерді әрі қарай жеткізу үшін пошта жүйесіне жөнелту – **пошталық мәлімдемені жіберу (mail submission)** деп аталады.

Мәлімдемені өңдеу тұтынушы тілегіне қарай жартылай автоматтандырылуы мүмкін. Мысалы, келген пошта сұрыпталып, спамға ұқсас мәлімдемелерге төменгі белсенділік берілуі мүмкін. Кейбір программалардың жауап мәлімдемені автоматты түрде жөнелту тәрізді («Мен демалыстамын, келген кезде сізге жауап беремін») қосымша мүмкіндіктері бар. Тұтынушы агенті сол тұтынушы өз электронды поштасын оқитын компьютерде жұмыс істейді. Бұл қарапайым программа және оның үздіксіз жұмыс істеуі міндетті емес.

Мәлімдеме жөнелту агенттері әдетте, жүйелік үдеріс болып саналады. Олар пошталық сервер машиналарында фондық режимде жұмыс істейді және үнемі қолжетімді болулары керек. Олар пошталық мәлімдерді жөнелтушіден қабылдаушыға **SMTP (Simple Mail Transfer Protocol – пошталық мәлімдемелерді жөнелтудің қарапайым хаттамасы)** көмегімен жүйеде автоматты түрде жылжытып отырулары қажет. Бұл – мәлімдеме жөнелтілетін қадам.

SMTP алғаш рет RFC 821 құжатында анықталған болатын. Әрі қарай оған қазіргі RFC 5321 редакциясына дейін өзгертулер енгізілді. Ол мәлімдемелерді байланыстар арқылы жөнелтіп, жеткізілу статусы және орын алған қателіктер жайлы есепті кері жібереді. Жеткізілгендіктің расталуы өте маңызды, тіпті, заңдық мағынасы бар («Сіздің құрметіңіз, менің электронды поштам сенімсіз, сондықтан сотқа шақыру қағазы жоғалып қалған болар») көптеген қосымшалар бар.

Мәлімдемелерді жөнелту агенттері, сонымен бірге, мазмұны бірдей мәлімдеме көшірмесін адресі электронды поштаның адрестер тізіміне енгізілгендердің барлығына жөнелтетін **жөнелту тізімін (mailing lists)** пайдаланады. Пайдалы қосымша мүмкіндіктер ішінде: хаттың «көшірме қағаз» көшірмесін жөнелтуді (Carbon copy), басқа қабылдаушылар жайлы ескертусіз көшірмені жөнелтуді (Blind carbon copy), белсенділігі жоғары хаттарды, құпия (яғни, шифрланған) пошта, поштаны баламалы қабылдаушыға жіберуді, егер негізгі қабылдаушы қолжетімсіз болса, сонымен бірге поштаны өңдеуді хатшыға тапсыру мүмкіндігін атап кетуге болады.

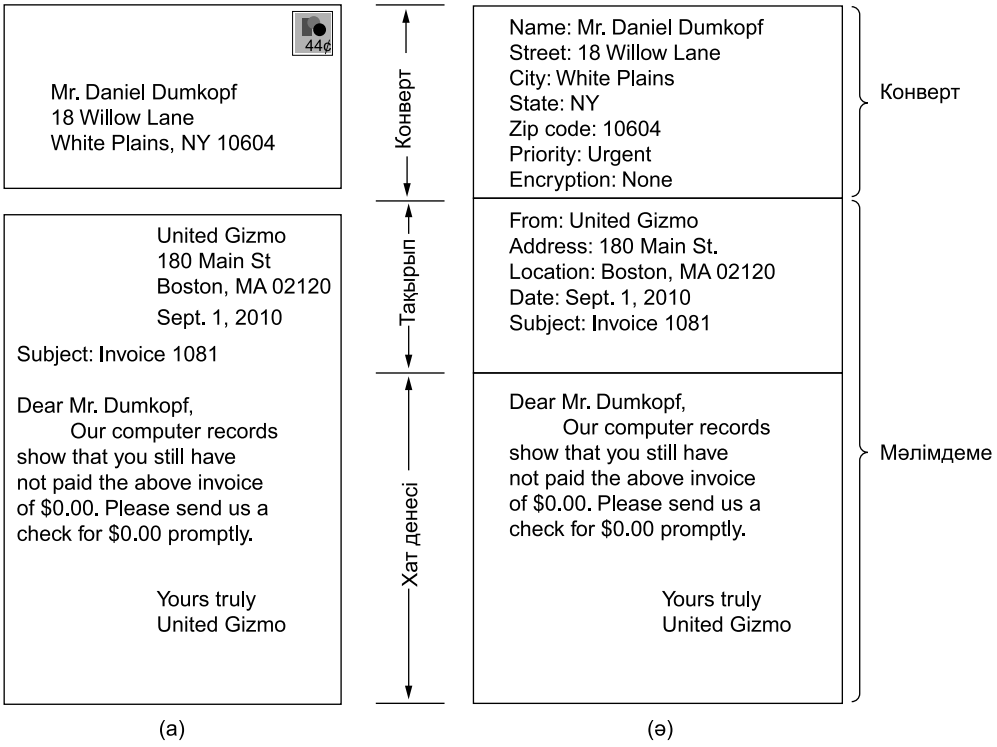
Тұтынушы агенттері мен мәліметтерді жөнелту агенттері арасындағы байланыс үшін пошта жәшіктері және пошталық мәліметтердің стандарты форматы жауап береді. **Пошталық жәшіктер (mailboxes)** тұтынушыға жеткізілген поштаны сақтайды. Оларды пошталық серверлер қолдап отырады. Тұтынушы агенттері тұтынушыға өз пошта жәшігінің мазмұнын көруге мүмкіндік жасайды. Бұл әрекетті орындау үшін, тұтынушы агенті пошталық серверге команда жөнелтіп, пошталық жәшікпен: мазмұнын қарау, мәлімдемелерді жою және т.б. іс-әрекеттер орындауға мүмкіндік алады. Поштаны алудағы соңғы қадам – соңғы тұтынушыға жеткізу (*7.4-суреттегі* 3-қадам). Мұндай құрылымда бір тұтынушы бір пошталық жәшікке қол жеткізу үшін, әртүрлі машинада әртүрлі пошталық агенттерді пайдалана алады.

Мәліметтерді жөнелту агенттері арасында пошта стандартты форматта тасымалданады. Алғашқыда құрастырылған формат RFC 822, кейіннен өзгертулер енгізілді. Ағымдағы версия RFC 5322 деп аталады, оған мультимедиа-контентін және халықаралық мәтінді қолдау енгізілген. Бұл схема MIME деп аталады, ол жайлы біз кейінірек айтамыз. Көбі электронды пошталы әлі де RFC 822 атайды.

Қазіргі заманғы электронды пошта жүйелерінің барлығының негізінде – **конвертті (envelope)** және хат мазмұнын шектеу – негізгі идеясы жатыр. Конверт өз ішіне мәлімдеме енгізеді. Онда мәлімдемені жеткізуге қажет ақпа-

раттың барлығы – қабылдаушы адресі, басымдылық, деңгей, құпиялық және т.б. жазылған, бұл мағлұматтардың барлығы мәлімдеменің өзінен бөлектелген. Мәлімдемелерді жөнелту агенттері әдеттегі пошталық қызмет тәрізді, конвертті маршруттау үшін пайдаланады.

Конверт ішіндегі мәлімдеме: **тақырып (header)** және **хат денесі (body)** сияқты екі жеке бөліктен тұрады. Тақырып тұтынушы агенттеріне арналған басқарушы ақпараттан тұрады. Хат денесі толығымен қабылдаушы-адамға арналған. Конверттер және мәлімдемелер мысалы *7.5-суретте* көрсетілген.



7.5-сурет. Конверттер және мәлімдемелер: а – әдеттегі хат; ә – электронды хат

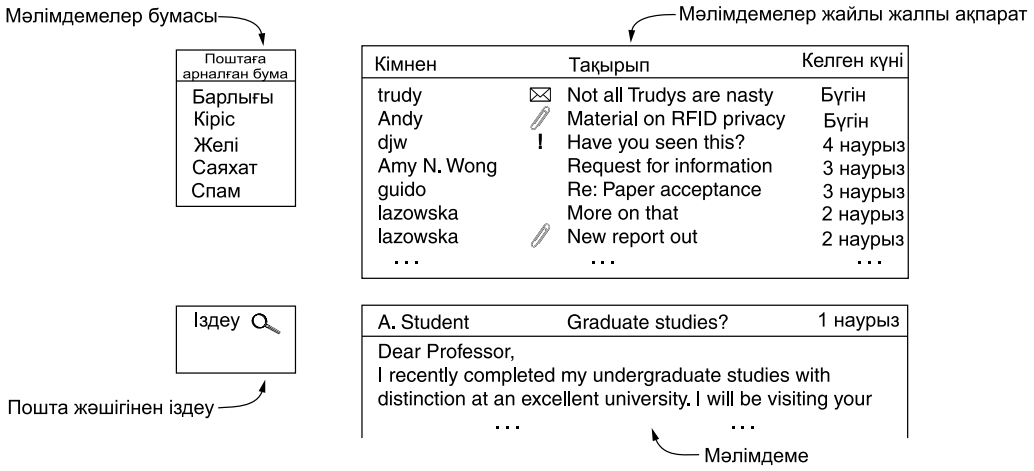
Біз бұл құрылымды, бір тұтынушыдан екіншісіне мәлімдеме жөнелту үшін орындалатын қадамдарды егжей-тегжейлі қарастырып, әңгімелейміз. Бұл саяхат тұтынушы агентінен басталады.

7.2.2. Тұтынушы агенті

Тұтынушы агенті – бұл мәлімдемені құрастыру және қабылдау, сонымен бірге мәлімдеге жауап жазу және пошта жәшігін басқару үшін көптеген командалармен басқарылатын программа (кейде **пошталық редактор – email editor** немесе **«оқығыш» – email reader** деп аталады).

Тұтынушы агенттерінің көптеген танымал түрлері, мысалы: Google gmail, Microsoft Outlook, Mozilla Thunderbird және Apple Mail бар. Олардың сыртқы түрлері бір біріне ұқсамайды. Көптеген тұтынушы агенттерінің графикалық интерфейсі меню немесе шартты белгілерге негізделген және тышқан қолтетігінің болғанын немесе кіші мобилдік құрылғыларда саусақты тигізіп басқарғанды талап етеді. Ертеректегі тұтынушы агенттері, мысалы, Elm, mh және Pine интерфейстері мәтінге негізделген және пернетақтадан бір әріптік командаларды енгізу арқылы жұмыс істейтін. Мәтіндік мәлімдерге байланысты функционалдық айырмашылықтар жоқ.

Тұтынушы агентінің әдеттегі элементтері *7.6-суретте* көрсетілген. Сіз қолданылатын агент қарапайымдау болуы мүмкін, бірақ функциялары сәйкес болып келеді.



7.6-сурет. Тұтынушы агентінің стандартты элементтері

Тұтынушы агенті іске қосылған кезде, ол тұтынушының пошта жәшігіндегі мәлімдемелер жайлы жалпылама ақпаратты көрсетіп тұрады. Жиірек бір мәлімдемеге бір жол сәйкес келеді, олар сұрыптау бойынша таңдалып алынған белгілі бір ретпен орналасады. Жалпылама ақпаратта мәлімдеменің конверттен немесе тақырыптан алынған кілтті өрісі ерекшеленіп тұрады.

Жалпылама ақпараттың жеті жолы *7.6-суретте* көрсетілген. Жолдарда *«Кімнен», «Тақырып»* және *«Қабылданған дата»* өрістері керсетілген ретпен пайдаланылады және онда мәлімдеменің кімнен алынғаны, не жайлы және қашан алынғаны жазылады. Бүкіл ақпарат тұтынушыға ыңғайлы түрде форматталған, ол мәлімдеме жолдарына негізделген, бірақ дәлме-дәл бейнелемейді. Сөйтіп, кімде кім мәлімдемені тақырыпсыз жөнелтсе, олардың хаттары төмен басымдылыққа ие болады.

Басқа да көптеген маркерлер және басқа өрістер болуы мүмкін. Мәлімдеме тақырыбы жанындағы шартты белгілер (*7.6-суретті* қараңыз), мысалы,

оқылмаған поштаны (конверт), қыстырылған файлды (скрепка), жөнелтуші көзқарасынан маңызды мәлімдемені (леп белгісі) білдіруі мүмкін.

Сұрыптаудың бірнеше нұсқалары болуы мүмкін. Ең кең таралғаны қабылданған датаға негізделген, алдымен қабылдаушы оқығанын немесе оқымағанын білдіретін маркері мен жаңалары орналасады. Жалпылама ақпарат және сұрыптау реті тұтынушының тілегі бойынша бапталуы мүмкін.

Тұтынушы агенттері кіріс мәлімдемелерді білгілі бір үлгімен, оқуға ыңғайлы болатындай бейнелеуі тиіс. Тұтынушы қай хатты қашан оқитынын шешу үшін, жиі хаттарды алдын ала қарап шығу ұсынылады (*7.6-суретті* қараңыз). Алдын ала қарау кезінде, хаттың мазмұнын сипаттау үшін кішкене шартты белгілер немесе суреттер қолданылуы мүмкін. Өңдеуді ұсынудың басқа нұсқалары мәлімдемені терезеге сиятындай етіп форматтау, аударма, мазмұнды ыңғайлы басқа формаға түрлендіру (мысалы, сандық дауысқа немесе мәтінді тану) болуы мүмкін.

Мәлімдеме оқылғаннан кейін сіз онымен әрі қарай не істеу керек екенін шешесіз. Бұл **мәлімдемені орналастыру (message disposition)** деп аталады. Опциялар ішінде жою, жауап жазу, басқа тұтынушыға жөнелту және мәлімдемені жәшікте әрі қарай жұмыс үшін қалдыру бар. Көптеген тұтынушы агенттері поштаны сақтау үшін бумалар жиынтығы бар бір кіріс мәлімдемелер жәшігімен жабдықталған. Бумалар тұтынушыға мәлімдемелерді жөнелтушіге, тақырыбына немесе басқа да белгілерге байланысты әртүрлі жерде сақтауға мүмкіндік береді.

Бумалар бойынша сұрыптауды тұтынушы агенті тұтынушы мәлімдемені оқығанша автоматты түрде орындауы мүмкін. Оның мысалы өрістерді және мәлімдеме мазмұнын тексеру, сонымен бірге хаттың спам екендігін анықтауға арналған тұтынушының алдыңғы мәлімдемелер жайлы пікірі бола алады. Көптеген интернет-провайдерлер және компаниялар мәлімдемені маңызды немесе спам ретінде белгілейтін программалық жабдықтамаларды пайдаланады. Сол себепті тұтынушы агенті оларды сәйкес бумаларға орналастыра алады. Интернет-провайдерлер және компаниялардың көптеген мәлімдемелер жиынтығымен жұмыс істеу артықшылығы бар, сондықтан оларда білгілі спамерлердің тізімі болуы мүмкін. Егер жүздеген тұтынушылар бір мезгілде бірдей мәлімдеме алатын болса, онда оның спам болуы ықтимал. Мәлімдемелерді алдын ала сұрыптап және олардың бір бөлігін «спам болуы мүмкін» деп белгілей отырып, тұтынушы агенті тұтынушыны қайсысының керек, қайсысының керек емес екенін анықтауға қажет үлкен жұмыстан босатады.

Ең танымал спам қайсы? Ол **ботнет (botnet, «ботжелі»)** деп аталатын бұзылған компьютерлер жиынтығымен генерацияланады. Ал оның мазмұны сіз тұратын елге байланысты. Азияда бұл жалған дипломдар жайлы ұсыныстар, АҚШ-та – арзан дәрі-дәрмек және күмәнді тауарлар. Сұраныссыз қалған есепшоттар Нигерия банкінде өз танымалдығын жойған емес. Ал, дененің қандай да бір бөлігін үлкейтуге арналған таблеткалар кез келген жерде топ параққа кіреді.

Сонымен қатар тұтынушылардың өздері хаттарды бумаларға таратуды ережесін ойлап шығара алады. Мысалы, ереже: басшыдан келген хат кідіріссіз оқуды талап ететін бумаға орналастырылуы қажет деуі мүмкін. Ал, белгілі бір жөнелтушілер тізімінен келген хаттар кейін оқу үшін басқа бумаға орналастырылуы тиіс. *7.6-суретте* бірнеше бума бейнеленген. Олардың ең маңыздысы – **Кіріс (Inbox)**, қабылданған кезде келісім ешқайда ауыстырылмаған кіріс пошта үшін, ал **Спам (Spam)**, спамға ұқсас пошта үшін.

Пошталық мәлімдемелерді бумаларға таратудан басқа тұтынушы агенті пошта жәшігінде хаттарды іздеудің кең мүмкіндігін ұсынады. Бұл опцияларда *7.6-суретте* көрсетілген. Іздеу мүмкіндігі тұтынушыға хатты жылдам табуға мүмкіндік береді. Мысалы, өткен айда алынған «where to buy Vegemite» жолы бар мәлімдемені жылдам табуға болады.

Электронды пошта жай деректерді тасымалау құралы болған кезден бері біраз уақыт өтті. Қазір провайдерлердің барлығы дерлік көлемі бір гигабайтқа дейінгі, ұзақ уақыт кезеңі арасындағы тұтынушы хаттарын сақтайтын электронды поштаны қолдайды. Ал мұндай үлкен көлемдегі мәліметтерді сақтауды тұтынушы агентінің пошталық мәліметтерді өңдеуге арналған іздеу және ақпаратты автоматты өңдеу кіретін күрделі амалдар жиынтығы қамтамасыз етеді. Жылына мыңдаған мәлімдеме алатын және жөнелтетіндер үшін бұл құралдар өте маңызды.

Тағы бір пайдалы опция – белгілі бір үлгіде мәлімдемелерге автоматты жауап беру. Жауаптың бір нұсқасы – кіріс мәлімдемені басқа адреске жөнелту, мысалы, радио немесе спутниктік антеннаны пайдаланып, *«Тақырып»* жолын пейджерде бейнелеу арқылы тұтынушыны шақыру үшін коммерциялық пейджингілік компания компьютеріне жөнелту. Бұл **автоматты жауап берушілер (autoresponders)** пошта серверінде жұмыс істеуі тиіс, себебі тұтынушы агенті үздіксіз жұмыс істемейді және поштаны тек кейде ғана алып отырады. Осы факторлар салдарынан тұтынушы агенті нағыз автоматты жауап беруді қамтамасыз етпейді. Алайда, автоматты жауап үшін интерфейсті тұтынушы агенті ұсынады.

Автоматты жауап берудің тағы бір мысалы – **агент демалыста (vacation agent)**. Әр кіріс мәлімдемені қарап отыратын бұл программа жөнелтушіге, мысалы, «Саламатсыз, мен демалыстамын. 24-тамызда келемін, сол кезде сізбен байланысамын», – деген бейтарап жауап жібереді. Сонымен бірге, мұндай жауаптарда, тұтынушы жоқ кезде орын алған тығыз жағдайда қалай байланысу керектігі, белгілі бір мәселелер жайлы қандай адамдарға жүгіну керек екендігі және т.б. жазылуы мүмкін. Көптеген агенттер демалыста автоматты жауап кімдерге жөнелтілетіндігі жайлы жазба жүргізеді және ол адамға қайталап жауап беру қажеттілігінен арылтады. Алайда, бұл агенттермен байланысудың қиындықтар бар. Мысалы, дайындалған жауапты үлкен тізім адресаттарына жөнелтуге болмайды.

Енді бір тұтынушының екіншісіне мәлімдеме жөнелту сценарийін қарастырайық. Тұтынушы агенті қолдайтын, біз қарастырмаған функцияның бірі –

мәлімдеме жазу. Бұл үдеріс – мәлімдеме құрастыру, оған жауап жазып, оны жеткізу үшін пошта жүйесінің басқа бөлігіне жөнелту. Мәлімдеме құрастыру үшін кез келген мәтіндік редакторды пайдалануға болса да, әдетте редактор тұтынушы агентіне кіріктіріледі. Бұл адрестаттарды ыңғайлы қосуға және тақырыпты, басқа да ақпаратты қажет өрістерге жазуға мүмкіндік береді. Мысалы, мәлімдемеге жауап жазу кезінде пошталық жүйе жөнелтуші адресін кіріс мәлімдемеден алып, автоматты түрде жауаптағы қажет өріске қоя алады. Басқа жиі кездесетін мүмкіндік – бұл мәлімдеме соңына **қолтаңба блогын (signature block)** қосу, қателерді түзеу, мәлімдеменің ақиқат екендігін растайтын сандық қолтаңбаны есептеу.

Пошталық жүйеге жөнелтілетін мәлімдеменің форматы стандартты және тұтынушы агенті ұсынған ақпарат негізінде түрлендірілуі тиіс. Мәлімдемені жөнелтудің ең маңызды бөлігі – бұл конверт, ал конверттің ең маңызды бөлігі – тағайындалған адрес. Бұл адрес мәлімдемелерді жөнелту агенттері жұмыс істей алатындай форматта болуы тиіс.

Күтілетін адрес форматы – бұл *user@dns-adress*. DNS домен аттары жүйесі жоғарыда, осы тарауда айтылғандықтан, біз бұл сұраққа тоқталмаймыз. Алайда, басқа адрестеу формаларының бар екенін айта кету керек. Жеке алғанда, **X.400** DNS-адрестеріне мүлдем ұқсамайды.

X.400 – бұл бір кезде SMTP-мен бәсекелес болған, ISO мәлімдемелерді өңдеу стандарты. Ол бәсекелестікте SMTP жеңіп шықты, дегенмен, X.400 әлі де қолданыста (негізінен АҚШ-тан тыс қолданылады). X.400 стандартының адрестері DNS-адрестеріне мүлдем ұқсамайды және слэшпен бөлінген *атрибут=мән* жұбынан тұрады, мысалы:

```
/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR/CN
=KEN SMITH/
```

Бұл адресте мемлекет, штат, орналасқан жері, жеке адрес және қабылдаушы аты (Ken Smith) көрсетілмеген. Сонымен бірге басқа да түрлі атрибуттарды көрсету мүмкін, бұл мүлдем атын білмейтін, тек сізге қабылдаушының басқа атрибуттары (мысалы, компания аты және лауазымы) белгілі болса, кез келген адамға электронды поштаны жөнелтуге мүмкіндік береді. X.400 адрестеу формасы DNS-ке қарағанда ыңғайсыз болса да, тұтынушы агенттері үшін бұл даулы сұрақ, себебі олар тұтынушы көзқарасы тұрғысынан ыңғайлы **бүркеме атты (address**, кейде – **nicknames**) қолдайды, олардың көмегімен нақты электронды адрес алынады. Сондықтан тұтынушы әдетте, бұл ыңғайсыз жолдарды енгізбейді.

Поштаны жөнелту шеңберінде біз қарастыратын соңғы сұрақ – тарату тізімі, ол тұтынушыға бір мәлімдемені қабылдаушылар тобына, бір команда көмегімен жөнелтуге мүмкіндік береді. Тарату тізімдерін сақтаудың екі жолы бар. Біріншісі – тұтынушы агенті жергілікті сақтайды. Бұл жағдайда тұтынушы агенті әр қабылдаушыға жеке мәлімдеме жөнелтеді. Сонымен бірге, бұл тізім қашықтықта, мәлімдемені жөнелту агентінде сақталуы мүмкін. Бұл

жағдайда мәлімдеме мәлімдемені тасымалдау жүйесінде тираждалады және көптеген тұтынушыларға мәлімдемені осы тізім бойынша жөнелтуге болады. Айталық, *meadowlark.arizona.edu* мәлімдеме жөнелту агентінде орнатылған *birders* (құсаулаушылар) тізімде құстарды зерттеушілер тобы бар делік. Онда *birders@meadowlark.arizona.edu* адресі бойынша жөнелтілген кез келген мәлімдеме алдымен Аризона штатының университетіне, ал сонан кейін ол жерден жеке мәлімдемелер түрінде тізім мүшелерінің барлығына олардың қай жерде жүргеніне қарамастан жөнелтіледі. Хат жөнелтуші үшін тарату тізімінің әдеттегі жеке адрестен еш айырмашылығы жоқ. Егер жөнелтуші *briders* – бұл тарату тізімі екенін білмесе, ол хатты Gabriel O. Brirders атты профессорға жөнелтіп отырмын деп ойлауы мүмкін.

7.2.3. Мәлімдемелер форматы

Енді тұтынушы интерфейсін қарастырудан электронды пошта мәлімдемелері форматын қарастыруға көшеміз. Тұтынушы агенті жөнелтетін мәлімдемені мәлімдемені жөнелту агенті өңдеу үшін олар белгілі бір стандартқа сәйкес рәсімделуі тиіс. Алдымен біз электронды хаттың RFC 5322 стандартындағы негізгі ASCII-форматын қарастырамыз. Ол – RFC 822-де сипатталған интернет-мәлімдеменің бірегей форматының нұсқасы. Сонан кейін біз бастапқы форматтың мультимедиалық кеңейтілімімен танысамыз.

RFC 5322 – интернет-мәлімдеме форматы

Мәлімдеме қарапайым конверттен (RFC 5321-де SMTP бөлігі ретінде сипатталған), бірнеше тақырып өрісінен, бос жолдан және мәлімдеме денесінен, әр тақырып өрісі (логикалық түрде) өріс атынан білдіретін бір ASCII-мәтіннен, қоснүктеден және (көп жағдайда) өріс мәнінен тұрады. Бастапқы RFC 822 бірнеше ондаған жылдар бұрын құрастырылған, онда конверт және тақырып арасында нақты шектеу жоқ болатын. Стандарт бөлігі RFC 5322 қайта қарастырылса да, оны толығымен жаңарту мүмкін болмады, себебі RFC 822 кеңінен таралып кеткен болатын. Әдетте тұтынушы агенті мәлімдемені құрастырып, оны мәлімдемені жөнелту агентіне береді. Мәлімдемені жөнелту агенті ескі үлгідегі мәлімдеме және конверт қоспасынан тұратын, тақырып өрістері бірінің көмегімен жаңа түрдегі конвертті құрастырады.

Транспорттаумен байланысты тақырыптардың негізгі өрістері *7.3-кестеде* келтірілген. *To:* (Кімге) өрісінде негізгі қабылдаушының DNS-адресі жазылады. Қабылдаушылар саны бірнешеу болуы мүмкін. *Cc:* өрісінде қосымша қабылдаушылардың адрестері көрсетілуі мүмкін. Жеткізу тұрғысынан негізгі қабылдаушы және қосымша қабылдаушы арасында ешқандай айырмашылық жоқ. Олардың арасындағы айырмашылық – психологиялық, ол адамдар үшін маңызды болуы мүмкін, ал пошта жүйесі үшін ешқандай айырмашылық жоқ.

Cc: (carbon copy) – «көшірме қағаз» арқылы жасалған экземпляр термині біршама ескірді, себебі компьютермен жұмыс істегенде көшірме қағаз мүлдем қолданылмайды, дегенмен, ол электронды поштада сенімді жайғасқан. *Bcc:* (Blind carbon copy – соқыр көшірме) өрісі алдыңғы өріске ұқсас, тек бұл жағдайда осы өріс жазбасы мәлімдеменің барлық экземплярларынан, негізгі және қосымша қабылдаушының да экземплярларынан алынып тасталынады. Бұл қасиет бір мезгілде бір мазмұндағы хатты бірнеше қабылдаушыға жөнелтуге мүмкіндік береді және қабылдаушылар хаттың өзінен басқа тағы біреуге жөнелтілгенін білмейді.

7.3-кесте. RFC 5322 стандартының мәлімдемені транспорттаумен байланысты өрістер тақырыбы

Өріс	Мәні
To:	Негізгі қабылдаушы (қабылдаушылар) электронды адресі (адрестері)
Cc:	Қосымша қабылдаушы (қабылдаушылар) электронды адресі (адрестері)
Bcc:	Соқыр көшірме электронды адресі (адрестері)
From:	Мәлімдеме авторы (авторлары)
Sender:	Жөнелтушінің электронды адресі
Receiver:	Маршрут бойынша әр мәлімдемені жөнелту агентінің қосатын жолы
Return-Path:	Жөнелтушіге баратын кері жолды сәйкестендіру үшін пайдаланылуы мүмкін

From: және *Sender:* сияқты екі өріс сәйкесінше мәлімдемені кімнің құрастырып, жөнелткенін білдіреді. Бұл әртүрлі адамдар болуы мүмкін. Мысалы, хатты мекеме басшысы жазып, ал хатшысы оны жөнелтуі мүмкін. Бұл жағдайда басшы *From:* өрісінде, ал хатшы *Sender:* өрісінде жазылады. *From:* өрісі міндетті болып саналады, ал *Sender:* өрісін мазмұны *From:* өрісімен сәйкес болған жағдайда тастап кетуге де болады. Бұл өрістер мәлімдеме жекізілмеген жағдайда жөнелтушіні хабардар ету үшін қажет. Сонымен бірге осы өрісті адрестер бойынша жауап жөнелтілуі мүмкін.

Receiver: өрісі жазылған жолға мәлімдемені жолы бойындағы әр тасымалдау агенті қосады. Бұл өріске агент идентификаторы, мәлімдеменің қабылданған датасы және уақыты, сонымен бірге маршруттау жүйесіндегі қателіктерді түзеуге қажет басқа да ақпараттар жазылады.

Return-Path: өрісін мәлімдеме жөнелтудің соңғы агенті қосады. Бұл өріс жөнелтушіге қалай жету керек екендігін хабарлайды деп болжанған болатын. Теорияда бұл ақпаратты *Receiver:* тақырыптың барлық өрістерінен (жөнелтуші пошта жәшігінен басқасын) жинап алуға болады, алайда, іс жүзінде ол сирек толтырылады және әдетте, онда тек жөнелтуші адресі жазылады.

RFC 5322 стандартының мәлімдемесі *7.3-кестеде* келтірілген өрістерден басқа, тұтынушы агенті немесе тұтынушының өзі пайдаланатын көптеген тақырыптар өрісінен тұруы мүмкін. Жиі пайдаланылатын тақырыптар өрісі *7.4-кестеде* келтірілген. Өрістердің қызметін түсіну үшін кестедегі ақпарат жеткілікті, сондықтан біз барлығын егжей-тегжейлі қарастырмаймыз.

7.4-кесте. RFC 5322 стандарты мәлімдеме тақырыбында қолданылатын кейбір өрістер

Өріс	Мағынасы
Data:	Мәлімдеме жөнелтілген уақыт және дата
Reply-to:	Жауап жөнелтуге арналған электронды адрес
Message-Id:	Келесіде осы мәлімдемеге сілтеме жасау үшін қолданылатын бірегей нөмір
In-Reply-To:	Мәлімдеменің Message-Id: идентификаторына жауап ретінде осы мәлімдеме жөнелтіледі
References:	Басқа да маңызды сілтемелер (Message-Id: идентификаторлары)
Keywords:	Тұтынушы таңдайтын кілтті сөздер
Subject:	Бір жолда бейнелеу үшін мәлімдеме тақырыбының қысқаша мазмұны

Reply-to: өрісі кейде хатты құрастырушы да, жөнелтуші де оған жауап алғысы келмеген жағдайда қолданылады. Мысалы, тауарды өткізу бөлімінің басқарушысы клиенттерді жаңа тауар жайлы хабардар етіп, хат жазуы мүмкін. Хатты оның хатшысы жөнелтуі мүмкін, бірақ *Reply-to:* өрісінде, барлық сұрақтарға жауап беріп, тапсырысты қабылдайтын сату бөлімі менеджерінің адресі көрсетілуі мүмкін. Бұл өрісі жөнелтушінің екі электронды адресі бар болса және ол жауапты жөнелтілген адрес емес, басқа адреске алғысы келген жағдайда да пайдалы болуы мүмкін.

Message-Id: – бұл автоматты генерацияланатын сан. Ол қайталап жөнелтуді болдырмас үшін мәлімдемелерді байланыстыруда (мысалды, хатқа жауап берген кезде) қолданылады.

RFC 5322 құжатында тұтынушы өз қажеті үшін қосымша өрістер құрастыра алатындығы ашық айтылған. RFC 822 форматынан бастап тақырыптар X- жолынан басталады. Болашақта, ресми және жеке тақырыптар арасында шиеленіс туындамас үшін, ешқандай стандартты тақырыптар бұл символдан басталмайды деп кепілдік беріледі. Кейде ақылгөй-студенттер *X-Fruit-of-the-Day* (бүгінгі жеміс) немесе *X-Disease-jf-the-Week* (апта ауыруы) тәрізді өрістерді қосады, қолданылуы заңды болғанымен мағынасы түсініксіз.

Тақырыптан кейін мәлімдеме денесі орналасады. Бұнда тұтынушы қалауынша кез келген мәліметті орналастыра алады. Кейбіреулер өз мәлімдемеле-

рін, танымал және сирек кездесетін цитаттар, саяси арыз және әртүрлі хабарландыру (мысалы, «Жоғарыда айтылған көзқарас үшін АБВ корпорациясы жауапты емес. Ол тіпті, оны түсінбейді де.») енгізілген, күрделі қолтаңбамен аяқтайды.

MIME – электронды поштаның Интернет желісіндегі көпмақсатты кеңейтілімі

ARPANET желісі дәуірінде электронды пошта тек ағылшын тілінде және ASCII символымен жазылған мәтіндік мәлімдемеден тұратын. Мұндай қолданыс үшін бастапқы RFC 822 стандарты жеткілікті болатын: ол тақырыптар форматын анықтап, мәлімдеме мазмұнын толығымен тұтынушы құзырына қалдыратын. 1990 жылдары басталған Интернетті дүниежүзілік қолдану және пошталық жүйе арқылы әртүрлі контентті жөнелту қажеттілігі, бұл амалдың Интернетті пайдалануға үйренген тұтынушыны қанағаттандырмайтындығын көрсетті. Жол үсті белгілері бар (мысалы, француз және неміс тілінде) тілдерде, латын тілінен өзгеше әліпбиі бар (мысалы, иврит немесе орыс) тілдерді немесе әліпбиі жоқ (мысалы, қытай немесе жапон) тілдерде мәлімдемелер жөнелтуді қамтамасыз ету қажет болды. Сонымен бірге, мәтін емес мәлімдемелерді (мысалы, аудио, сурет немесе бинарлы құжаттар және программалар) жөнелту мүмкіндігін қамтамасыз ету керек болды.

Шешім ретінде **MIME (Multipurpose Internet Mail Extensions – электронды поштаның Интернет желісіндегі көпмақсатты кеңейтілімі)** құрастырылды. Олар Интернет арқылы жөнелтілетін пошталық мәлімдемелер үшін де және веб-сайт тәрізді басқа қосымшалар үшін контентті сипаттауда да кеңінен қолданылады. MIME RFC2045-2047, 4288, 4289 және 2049 құжаттарында сипатталған.

MIME стандартының негізгі мақсаты – мәлімдеме денесіне құрылымдар қосу және ASCII-мәлімдеме еместерін кодтау ережесін анықтау арқылы RFC 822 форматын (RFC 5322 форматының алдындағы өткені және MIME құрастырылған кезде қолданыста болған) пайдалануды жалғастыру. RFC 822 стандартынан ауытқымай, MIME-мәлімдеме, әдеттегі мәлімдеме жөнелту агенттері және хаттамалар (ертеде RFC 821-ге, ал қазір RFC 5321-ге негізделген) арқылы жөнелтіледі. Тек тұтынушы өздері үшін құрастыра алатын – жөнелтуші және қабылдаушы программалары өзгертуді қажет етеді.

MIME стандарты, *7.5-кестеде* келтірілген, мәлімдеменің бес жаңа тақырыбын анықтайды. Бірінші тақырып (*MIME-Version:)* мәлімдемені қабылдайтын тұтынушы агентіне MIME-мәлімдемесімен жұмыс істейтінін және осы мәлімдемеде қолданылатын MIME версиясын анықтайды. Егер мәлімдемеде бұл тақырып жоқ болса, онда ол ағылшын тілінде жазылған деп саналады (немесе тек ASCII таңбаларын пайдаланатын тілде) және сәйкесінше өңделеді.

7.5-кесте. MIME қосқан мәлімдеме тақырыптары

Тақырып	Сипаттамасы
MIME-Version:	MIME стандартының версиясын көрсетеді
Content-Description:	Мазмұн сипаттамасы. Мәлімдеме мазмұны жайлы хабарлайтын қарапайым мәтін жолы.
Content-Id:	Бірегей идентификатор
Content-Transfer-Encoding:	Тасымалданатын мәлімдеме денесінің кодталу әдісін көрсетеді
Content-Type:	Мәлімдеме мазмұнының типі және форматы

Content-Description: тақырыбы мәлімдемеде не бар екенін хабарлайтын ASCII-жол. Бұл тақырып тұтынушыға мәлімдемені оқу немесе оқымау жайлы шешім қабылдауға мүмкіндік береді. Егер жолда «Қосаяқ Барбараның суреті» деп жазылса, ал ол қосаяқтарды ұнатпайтын болса, оның мәлімдемені бірден жойып тастауы ықтимал.

Content-Id: тақырыбында мәлімдеме мазмұнының идентификаторы жазылады. Бұл жерде стандартты *Message-Id:* тақырып форматы қолданылады.

Content-Transfer-Encoding: тақырыбында желі арқылы жөнелту үшін мәлімдеме денесіне қолданылатын бума тәсілі көрсетіледі. MIME құрастыру кезіндегі негізгі мәселе – мәлімдеме ASCII форматында болады деп жорамалдайтын мәлімдемелерді тасымалдау хаттамасы (SMTP) болатын. Бұл жағдайда жолда 1000 символдан артық болмау керек. ASCII символдары әр сегізбиттік байттың 7 битін пайдаланады. Орындалушы программа және сурет тәрізді бинарлық деректер бүкіл 8 байтты және кеңейтілген символдар жиынтығын да пайдаланады. Бұл деректердің қауіпсіз тасымалданатындығына ешбір кепілдік жоқ болды. Осының барлығының нәтижесінде бинарлық деректерді, әдеттегі ASCII пошталық мәлімдемесі ретінде бейнелеп, тасымалдайтын тәсіл қажет болды. MIME құрастырылған сәттен енгізілген кеңейтілген SMTP, қазіргі кезде кодталмаған бинарлық деректер пошталық жүйе арқылы әркез дұрыс жеткізілмесе де, іс жүзінде сегіз биттік бинарлық деректерді тасымалдауға мүмкіндік береді.

MIME, деректерді тасымалдауға арналған кодтаудың бес схемасын қамтамасыз етеді (сонымен бірге жаңа схема енгізу мүмкіндігі бар). Олардың ең қарапайымы – қарапайым мәтіндік ASCII мәлімдемені тасымалдау. ASCII симмволы 7 разрядты пайдаланады және жолдағы символдар саны 1000 аспаған жағдайда тікелей электронды пошта хаттамасымен жөнелтіле алады.

Екінші қарапайым схема алдыңғы схемаға ұқсас, бірақ мұнда 8-разрядтық симиволдар, яғни байттың барлық мәндері 0-ден 255-ті қоса қолданылады. Сегіз разрядты кодтауды қолданатын мәлімдемелерде жолдың ең үлкен ұзындығы жайлы ереже сақталуы тиіс.

Нағыз екілік жүйеде кодталған мәлімдемелермен жағдай біршама қиын. Оларға байттағы барлық 8 разрядты пайдаланатын және бір жолда 1000 символдан аспауы керек деген ережені сақтамайтын, еркін екілік файлдар кіреді. Бұл санатқа орындалатын файлдар жатады. Бүгінгі күнде пошталық серверлер бинарлық кодтағы (немесе сегіз биттік) деректерді тасымалдауға мүмкіндік бар екендігін тексере алады және байланыстың екі басында да бұл кеңейтілім қолданбайтын болса, деректер ASCII-ге түрлендіріледі.

Бинарлық деректерді ASCII форматына кодтау **base64 (64-символдық кодтау)** деп аталады. Бұл тәсіл қолданылған жағдайда 24 биттік топтар 6 биттен төрт топқа бөлінеді де, әрқайсысы рұқсат етілген ASCII-символы ретінде тасымалданады. Мұндай кодтауда 6-разрядтық 0 символы «A» ASCII-символымен кодталады, 1 – «B» ASCII-символымен және т.с.с. Әрі қарай 26 кіші әріптер, онан кейін 10 цифр және соңында 62, 63 кодтау үшін сәйкесінше + және / пайдаланылады. == және = тізбектері соңғы топтың сәйкесінше тек 8 немесе 16 биттен тұратынын білдіреді. Жаңа жолға көшу және каретканы қайтару символдары есепке алынбайды, сондықтан жолдар тым ұзын болып көрінбес үшін оларды кодталған символдар ағынының кез келген жеріне қоюға болады. Аса тиімді болмаса да, осы әдіспен кез келген екілік кодты тасымалдауға болады. Бинарлық ақпаратты тасымалдайтын пошталық серверлер пайда болғанша бұл кодттау өте танымал болатын.

Аз ғана ASCII емес символдары бар, ал қалғаны толығымен ASCII-символдан тұратын жол үшін бұл тәсіл біршама тиімсіз. Оның орнына **quoted-printable (дәйексөз баспа коды)** кодын пайдаланған дұрыс. Бұл 127-ден жоғары ASCII-код мәніне сәйкес келетін қарапайым 7-разрядты ASCII, теңдік белгісімен кодталады онан кейін ASCII-код символының он алтылық жүйедегі екі саны жүреді. Соңғы бос орындардан басқа басқару символдары, кейбір пунктуация белгілері және математикалық символдар да кодталады.

Соңында, егер бұл тәсілдерді қолданбауға салмақты негіз бар болса, *Content-Transfer-Encoding:* тақырыбында өз кодтауыңызды көрсетуіңізге болады.

Ең көп қызығушылық тудыратын *7.5-кестеде* көрсетілген соңғы тақырып. Ол мәлімдеме денесінің типін көрсетеді және оның қолданыс аясы электронды пошта шеңберінен әлдеқайда тысқары жатыр. Мысалы, Интернеттен жүктелетін контент MIME символдарымен белгіленеді, бұл браузерге ақпаратты дұрыс бейнелеуге мүмкіндік береді. Ағындық мультимедиа арқылы және нақты уақыт режимінде жөнелтілетін контент пен жағдай осылай, мысалы, IP (VoIP) арқылы берілетін дауыс.

Бастапқыда RFC 1521 құжатында мәлімдеме мазмұнының жеті типі анықталған болатын, олардың әрқайсы қолжетімді бірнеше ішкі типтерге бөлінеді. Ішкі типтер негізгі типтен қисық сызық арқылы (/) бөлінеді, мысалы, «Content-Type: video/mpeg». Одан бері көптеген типтер және ішкі типтер қосылды. Бұл тізім қажеттілік туған кезде үнемі толықтырылып отырады. Желіде оны IANA қолдайды, ол *www.iana.org/assignments/media-types* адресі бойынша қолжетімді.

Типтер және жиі қолданылатын ішкі типтер мысалы, *7.6-кестеде* келтірілген. Олардың әрқайсысына қысқаша тоқталып кетейік. Қабылданғаннан кейін бірден компьютер экранында бейнеленетін қарапайым мәтіндік мәлімдеме *text/plain* комбинациясымен белгіленеді. Ол үшін қосымша өңдеу немесе қайта кодтау қажет емес. Бұл тақырып өрісінің мәні әдеттегі MIME мәлімдемесін аздаған қосымша тақырыптар қосып тасымалдауға мүмкіндік береді. Веб-технологиялар танымал бола бастаған кезде, RFC 822 хаттарының денесінде вебпарақтарды тасымалдауға мүмкіндік беретін жаңа *text/html* типі қосылды (RFC 2854-те). RFC 3023-де кеңейтілмелі белгілеу тілі (eXtensible Varkup Language - XML) үшін *text/xml* ішкі типі анықталған. Интернеттің дамуымен XML құжаттар өсе бастады. Төменде *7.3-бөлімінде* біз HTML және XML қарастырамыз.

MIME-нің келесі типі *image*. Ол қозғалмайтын суреттерді тасымалдауға мүмкіндік береді. Қазіргі таңда суреттерді тығыздау және тығыздаусыз сақтау және тасымалдаудың көптеген түрлі форматтары бар. Кейбір форматтар, сонымен бірге GIF, JPEG және TIFF іс жүзінде барлық броузерлерге кіріктірілген, алайда, көптеген басқа типтер және оларға сәйкес ішкі типтер де бар.

Дыбыс және жылжымалы суреттерді тасымалдау үшін сәйкесінше *audio* және *video* типтері пайдаланылады. Мұндағы video ішкі типінің дыбыс жолынан емес тек вузуалды ақпараттан тұратынына назар аударыңыз. Егер электронды пошта арқылы дауысы бар бейне фильмді тасымалдау қажет болса, онда бейне қатарды бөлек, дыбыстық жолды бөлек тасымалдауға тура келеді. Бұл кодтау жүйесіне тәуелді. MIME стандарты анықтаған алғашқы бейне формат **MPEG (Motion Experts Group – жылжымалы сурет сұрақтары бойынша эксперттік топ)** болатын. Содан бері бірнеше форматтар қосылды. Тұтынушыларға электронды пошта арқылы MP3 форматындағы аудио файлды тасымалдауға мүмкіндік беру үшін, RFC 3003-те бұрыннан келе жатқан *audio/basic* басқа *audio/mpeg* типі қосылды. *video/mp4* және *audio/mp4* типтері жаңа MPEG4 форматындағы бейне және аудио деректерге нұсқайды.

7.6-кесте. MIME стандартының типтері және ішкі типтер мысалы

Тип	Ішкі тип	Сипаттамасы
text	plain, html, xml, css	Түрлі форматтағы мәтін
image	gif, jpeg, tiff	Сурет
audio	basic, mpeg, mp4	Дыбыс
video	mpeg, mp4, quicktime	Бейне фильмдер
model	Vrml	3D-модель
application	onoctet-stream, pdf, javascript, zip	Қосымшалар шығаратын деректер
message	http, rfc822	Мәлімдемені инкапсуляциялау
multipart	mixed, alternative, parallel, digest	Бірнеше типтер комбинациясы

Model типі контенттің басқа типінен кейінірек қосылған болатын. Ол 3D-модельдерді сипаттауға арналған. Алайда, ол кеңінен қолданылмайды.

Application (қосымша) типі, басқа типтермен қолданылмайтын және қосымшадан деректерді түсіндіруді талап ететін барлық форматтарға арналған. Біз *pdf, javascript* және *zip* ішкі типтерін сәйкесінше PDF-құжат, Java-Script программасы және ZIP форматындағы архив мысалы ретінде келтірдік. Бұл контентті қабылдайтын тұтынушы агенті оны бейнелеу үшін сыртқы кітапхананы немесе программаны пайдаланады. Бейнелену нәтижесі тұтынушы агентіне кіріктірілуі мүмкін.

MIME типтерін пайдаланып, тұтынушы агенттері жаңа контенттер пайда болған сайын оларды өңдеу мүмкіндігін кеңейтеді. Бұл – маңызды артықшылық. Екінші жағынан, контенттің көптеген жаңа формаларын қосымшалар өңдейді және интерпретациялайды, бірақ бұл белгілі бір дәрежеде қауіп тудырады. Электронды пошта арқылы «досыңыздан» келген еркін орындалатын программаны іске қосудың қауіпті екені айқын. Мұндай программа компьютердің қол жеткен бөлігіне нұқсан келтіруі мүмкін, әсіресе, егер ол файлды оқып, құрастыратын және желіні пайдалана алатын болса. Құжат форматтары айқын болмасада біршама қауіпті. Бұндай мүмкіндікті есептен алып тастауға болмайды, себебі PDF типінің форматтары маскіленген дамыған программалау тілдері болуы мүмкін. Олар интерпретациялатын және оларға қолжетімді аумақ шектелген болса да, интерпретатордағы қателік бұл құжаттарға шектелген аумақты айналып өтуге мүмкіндік береді.

Бұл мысалдардан басқа, қосымшалардың көптеген ішкі типтері бар, себебі қосымшалардың өздері де аз емес. Мәселен, *octet-stream* типі (байттық ағын) ешқандай өңделмейтін қарапайым байттар тізбегін білдіреді. Ол контентті ешбір сипаттауға мүмкіндік жоқ кезде жазылады (егер қолайлы ешқандай тип жоқ болса). Мұндай ағынды алған тұтынушы агенті тұтынушыға оны файл ретінде сақтауды ұсынады. Әрі қарай өңдеу толығымен, қандай контент алғанын білетін тұтынушыға байланысты.

Соңғы екі тип мәлімдеме құрастырғанда және олармен түрлі әрекет жасағанда өте пайдалы. *Message* типі бір мәлімдемені екіншісіне қоюға мүмкіндік береді. Бұл хатты қайта адрестеу кезінде пайдалы. Егер бір мәлімдеме ішінде RFC 822 стандартының басқа бір мәлімдемесі бар болса, *rfc822* ішкі типін пайдалану керек. HTML форматындағы құжаттар да әдетте, осылайша инкапсуляцияланады. Ал *partial* ішкі типі мәлімдемелерді бірнеше бөлікке бөліп жеке тасымалдауға мүмкіндік береді (мысалы, инкапсуляцияланған мәлімдеме тым ұзын болса). Параметрлер мәлімдемені қабылданғаннан кейін дұрыс ретте қалпына келтіруге мүмкіндік береді.

Соңында, *multipart* типі бірнеше бөліктен тұратын мәлімдеме құрастыруға мүмкіндік береді және әр бөліктің басы және соңы нақты көрсетіледі. *Mixed* ішкі типі әртүрлі форматтағы мәлімдеме бөліктерін құрастыруға мүмкіндік береді. Көптеген пошталық программалар тұтынушыға мәтіндік мәлімдемеге бір немесе бірнеше қосымша қыстыруға мүмкіндік береді. Бұл қосымшалар

multipart типінің көмегімен тасымалданады. *Alternative* ішкі типін пайдаланған жағдайда керісінше, әр бөліктегі мәлімдемелер бірдей болуы тиіс, бірақ басқа түрде немесе кодта. Мысалы, мәлімдеме қарапайым ASCII-мәтіні HTML және PDF форматында жөнелтілуі мүмкін. Дұрыс құрастырылған тұтынушы агенті мұндай мәлімдемені алғаннан кейін алдымен оны тұтынушы талабына сәйкес форматта бейнелуге тырысуы тиіс. Егер бұл қандай да бір себептермен мүмкін болмаса, онда оны HTML форматында бейнелеуі тиіс. Егер бұл да мүмкін болмаса, онда ASCII-мәтін түрінде бейнеленеді. Бөліктерді, тіпті, ескі (MIME-ге дейінгі) тұтынушы агенттері жоқ дегенде қарапайым ASCII-мәтін түрінде бейнелей алатындай, күрделілігінің өсу реті бойынша орналастыру керек.

Alternative ішкі типін, сонымен бірге бір мезгілде бірнеше тілде жөнелтілетін мәлімдемелер үшін қолдануға болады. Осы тұста, Мысырда табылған әйгілі розет тасы *multipart/alternative* типіндегі мәлімдеменің ертедегі нұсқасы бола алады.

Қалған екі ішкі тип мысалына келетін болсақ, *parallel* ішкі типі мәлімдеменің барлық бөліктері бір мезгілде қаралу керек болған жағдайда қолданылады. Мысалы, фильмдерде жиі бейне- және аудио- жолдар болады. Фильмді осы екі жол тізбекпен емес, қатар жүргенде қараған әлдеқайда ыңғайлы. *Digest* ішкі типі бірнеше мәлімдеме бір мәлімдемеге біріктірілген кезде қолданылады. Мысалы, Интернеттегі қандайда бір пікірталас клубы жазылушылар мәлімдемесін жинап, сонан кейін оны *multipart/digest* типіндегі бір мәлімдеме ретінде жөнелтуі мүмкін.

MIME типтері электронды пошта мәлімдемелер үшін қалай қолданылатындығының мысалы *7.2-листингте* келтірілген. Мысалда, туған күнмен құттықтау бірмезгілге екі балама форматта: HTML форматында және аудио жазба түрінде жөнелтіледі. Егер тұтынушының аудио жазбаны тыңдау мүмкіндігі бар болса, онда тұтынушы агенті оны қалыпқа келтіреді. Мысалда дыбыс *message/external-body* ішкі типі ретінде сілтеме арқылы беріледі, сондықтан тұтынушы агенті алдымен дыбыс үшін FTP арқылы *birthday.snd* файлына жүгінуі тиіс. Кері жағдайда мәлімдеме мәтіні экранда толық үнсіздікте бейнеленеді. Хаттың бұл екі бөлігі қос дефиспен бөлінген, онан кейін *boundary* (шекара) параметрінің мәні ретінде көрсетілген жол (тұтынушы анықтайтын) орналасады.

Назар аударыңыздар: *Content-Type* тақырыбы мәлімдемеде үш рет кездеседі. Жоғарғы деңгейде мәлімдеменің бірнеше бөліктен тұратындығы, әр бөлік үшін оның типі және ішкі типі көрсетілген. Соңында, мәлімдеменің екінші бөлігінде ол тұтынушы агентіне сыртқы файл типін көрсетеді. Тақырыптардың барлығы үшін символ регисторының маңызы жоқ болғанымен, біз бұл айырмашылықты көрсету үшін соңғы жағдайда кіші символдарды қолдандық. 7-разрядты ASCII-ден өзгеше сыртқы дене форматы үшін де *content-transfer-encoding* тақырыбы қажет.

7.2-листинг. HTML және аудиодан тұратын *multipart* типіндегі мәлімдеме

```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Жер Күнді бүтін сан рет айналып шықты

Бұл кіріспе сөзбасы. Тұтынушы агенті оны есепке алмайды.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
      access-type=»anon-ftp»;
      site=»bicycle.cs.washington.edu»;
      directory=»pub»;
      name=»birthday.snd»

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

7.2.4. Мәлімдемелерді тасымалдау

Тұтынушы агенті және электронды пошта мәлімдемелері сипаттағаннан кейін, енді тұтынушы агенті оларды жөнелтушіден қабылдаушыға қалай жеткізетіндігін білген дұрыс. Поштаны тасымалдау SMTP хаттамасының көмегімен жүзеге асырылады.

Бұл үшін жөнелтуші машина және қабылдаушы машина арасында транспорттық байланыс орнатып, сонан кейін ол арқылы мәлімдемені жөнелткен дұрыс. SMTP бастапқыда осылайша жұмыс істеген. Алайда, соңғы кезде SMTP-ны қолданудың екі тәсілі пайда болды. Біріншісі – **пошталық мәлімдемені беру (mail submission,** *7.4-суреттегі* e-mail құрылымының бірінші қадамы) үшін. Бұл тәсіл арқылы тұтынушы агенттері мәлімдемені әрі қарай жөнелту мақсатында пошталық жүйеге береді. Екіншісі – мәлімдемені, мәлімдеме жөнелту агенттері арасында тасымалдау (*7.4-суреттегі* екінші қадам).

Мұндай тізбек, мәлімдемені жөнелтуші агенттен қабылдаушы агентке дейінгі бүкіл жол бойында бір қадам ішінде жеткізуге мүмкіндік береді. Соңғы,

нақты жеткізу басқа хаттамалар арқылы жүзеге асырылады. Біз оны келесі бөлімде қарастырамыз.

Бұл бөлімде біз SMTP хаттамасының негіздері және оны кеңейту механизмдері жайлы да айтамыз. Сонан кейін, оны әртүрлі жолмен қабылдау және жөнелту үшін қолдануды талқылаймыз.

SMTP – электронды поштаның қарапайым хаттамасы және оның кеңейтілімі

Интернетте электронды поштаны жөнелту үшін жөнелтуші машина қабылдаушы машинаның 25-портымен TCP-байланыс орнатады. Бұл порт пошталық доменмен тыңдалынады және олардың қарым-қатынасы **SMTP (Simple Mail Transfer Protocol – электронды поштаны тасымалдаудың қарапайым хаттамасы)** арқылы жүзеге асырылады. Бұл сервер кіріс байланысты қабылдап, оны қауіпсіздікке тексереді және мәлімдемені жеткізу мақсатында қабылдайды. Егер хатты жеткізу мүмкін болмаса, онда жөнелтушіге осы хаттың бірінші бөлімінен тұратын қателік жайлы мәлімдеме жеткізіледі.

SMTP хаттамасы – қарапайым ASCII-хаттама. Бұл – кемшілік емес, айырмашылық. ASCII форматындағы мәтінді пайдалану хаттамадағы қателіктерді жеңіл түрлендіруге, тестілеуге және түзетуге мүмкіндік береді. Олар команданы қолдан енгізу арқылы тестіленеді және бұл жағдайда мәлімдеме жеңіл оқылады. Қазіргі кезде қолданбалы деңгейдегі интернет-хаттамалардың көбі осылайша жұмыс істейді (мысалы, HTTP).

Біз пошталық серверлер және олардың жеткізушілері арасындағы қарапайым мәлімдеме тасымалдау жайлы сөз қозғаймыз. Жиырма бесінші портпен TCP-байланыс орнатып, жөнелтуші машина клиент рөліне енеді, сөйтіп, сервер режимінде жұмыс істейтін қабылдаушы машинадан сұраныс күтеді. Сервер сұхбатты өзінің идентификаторы және поштаны қабылдауға дайын екендігі (немесе дайын емес) жайлы жолдан тұратын мәтіндік жолды жөнелтуден бастайды. Егер сервер поштаны қабылдауға дайын болмаса, клиент байланысты үзіп, талпынысын кейін қайталайды.

Егер сервер поштаны қабылдауға дайын болса, клиент поштаның кімнен келгенін және кімге тағайындалғанын хабарлайды. Егер пошта қабылдаушысы бар болса, сервер клиентке мәлімдемені жөнелтуге келісім береді. Сонан кейін клиент мәлімдемені жөнелтеді, ал сервер оның қабылданғандығын растайды. Бақылау қосындысы тексерілмейді, себебі TCP транспорттық хаттамасы сенімді байттық ағынды қамтамасыз етеді. Егер жөнелтушіде тағы пошта бар болса, ол да жөнелтіледі. Пошта толығымен жөнелтілгеннен кейін екі бағыттағы байланыста үзіледі. *7.2-листингтегі* мәлімдемені тасымалдау кезіндегі клиент және сервер арасындағы сұхбат *7.3-листингінде* көрсетілген. Клиент жөнелткен (демек, жөнелтуші) жолдар C:*,* ал сервер жөнелткен (демек, қабылдаушы) – *S:* әрпімен белгіленген.

Алдымен клиент серверге сәлемдеме жолдайды. Сөйтіп, клиенттің бірінші командасы `HELO` түрінде болады, бұл `HELLO` сөзінің төрт символға дейін сәтті қысқартылған түрі. Бұл командаларды не себептен төрт символға дейін қысқарту қажет болғанын қазір ешкім білмейді.

7.3-листинг. Мәлімдемені *alice@cs.washington.edu*-ден *bob@ee.uwa.edu.au*-ге жөнелту

```
S:  220 ee.uwa.edu.au SMTP қызметі дайын
C: HELO abcd.com
S: 250 cs.washington.edu  ee.uwa.edu.auге сәлем жолдайды
C: MAIL FROM: <alice@cs.washington.edu>
S: 250 жөнелтушіні растаймын
C: RCPT TO: <bob@ee.uwa.edu.au>
S: 250 қабылдаушыны растаймын
C: DATA
S: 354 Хатты жөнелтіңіз; Хат соңы «.» символынан тұратын жолмен белгіленеді
C: From: alice@cs.washington.edu
C: To: bob@ee.uwa.edu.au
C: MIME-Version: 1.0

C: Message-Id: <0704760941.AA00747@ee.uwa.edu.au>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Жер Күнді бүтін сан рет айналып шықты
C:
C: Бұл кіріспе сөзбасы. Тұтынушы агенті оны есепке алмайды.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/html
C:
C: <p>Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <bold> Bob </bold>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C: access-type=»anon-ftp»;
C: site=»bicycle.cs.washington.edu»;
C: directory=»pub»;
C: name=»birthday.snd»
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
S: 250 мәлімдеме қабылданды
C: QUIT
S: 221 ee.uwa.edu.au байланысты үземін
```

Біздің мысалымызда мәлімдеме тек бір қабылдаушы жеткізілуі тиіс, сондықтан тек бір `RCPT` (recipient – қабылдаушы сөзінің қысқартылған түрі) командасы қолданылады. Бұл команданы бірнеше рет қолдану мәлімдемені бірнеше қабылдаушыға жөнелтуге мүмкіндік береді. Олардың әрқайсысы жеке расталады немесе қабылданбайды. Кейбір қабылдаушыға мәлімдемені қайта жөнелту сәтсіз болғанымен (мысалы, адресаттың жоқтығына байланысты) ол тізімдегі басқа адресаттарға жеткізілуі мүмкін.

Соңында, төрт символдық командалар синтаксисі қатаң анықталғандығымен, жауап синтаксисі мұншалықты қатаң емес. Ережені тек жол басындағы сандық код анықтайды. Бұл кодтан кейін орналасқанның барлығы түсініктеме мәтін болып саналады және ол хаттаманың нақты жүзеге асырылуына байланысты.

SMTP базалық хаттамасы жақсы жұмыс істейді, бірақ кейбір жағдайларда шектелген. Ол сәйкестендіруді қоспайды. Бұл, мысалдағы `FROM` командасы жөнелтушінің өзіне ұнаған кез келген адресін беруі мүмкін дегенді білдіреді. Бұл спамды жөнелту үшін өте ыңғайлы. Келесі шектеу – SMTP бинарлық деректер емес, ASCII мәлімдемені тасымалдайды. Нақты осы себептен MIME контентін жөнелту үшін base64 коды қажет болды. Алайда, бұл кодтаумен пошта жөнелткен кезде өткізгіштік қабілеттілік жеткіліксіз пайдаланылады, сондықтан үлкен көлемдегі мәлімдеме жөнелтілгенде ыңғайсыз. Үшінші шектеу – SMTP нақты мәлімдемені жөнелтеді. Ол шифрланбайды және еш қорғалмаған.

Осы және мәлімдемені жөнелтумен байланысты басқа да көптеген мәселелерді болдырмас үшін SMTP-ға кеңейтілім қосылған болатын. Ол RFC 5321 стандартының міндетті бөлігі болып саналады. SMTP-не кеңейтілімімен қолдану **ESMTP (Extended SMTP – кеңейтілген SMTP)** деп аталады.

7.7-кесте. SMTP-ның кейбір кеңейтілімдері

Кілтті сөз	Сипаттамасы
AUTH	Клиенті сәйкестендіру
BINARYMIME	Сервер бинарлық мәлімдемені қабылдайды
CHUNKING	Сервер үлкен мәлімдемені бөлшектеп қабылдайды
SIZE	Мәлімдеме мөлшерін жөнелту алдында тексеру
STARTTLS	Қауіпсіз арнаға ауысу (TLS; *8-тарауды* қараңыз)
UTF8SMTP	Интернационалданған адрес

Кеңейтілген версияны пайдаланғысы келген клиенттер алдымен `HELO` орнына `EHLO` жөнелтеді. Егер бұл нұсқа қабылданбаса, сервер әдеттегі SMTP-мен жұмыс істейді, ал тұтынушы стандартты жолмен жүруі тиіс. Егер `EHLO` қабылданса, сервер қандай кеңейтілімдерді қолдайтынын хабарлайды. Бұдан кейін клиент аталған кеңейтілімдердің кез келгенін қолдана алады. Бірнеше

стандартты кеңейтілімдер *7.7-кестеде* келтірілген. Онда кілтті сөздер кеңейтілу механизмінде қолданылатын түрінде берілген және оның қызметі сипатталған. Кеңейтілімдерді бұдан әрі егжей-тегжейлі қарастырмаймыз.

SMTP және осы тарауда қарастырылған хаттамалардың қалай жұмыс істейтінін жақсы түсіну үшін өзіңіз олармен жұмыс істеп көріңіз. Кез келген жағдайда алдымен Интернетке қосылған машинаны табыңыз. UNIX (немесе Linux) жүйесінде командалық жолда *mail.isp.com* орнына провайдердің DNS пошталық сервер атын қойып:

```
telnet mail.isp.com 25
```

командаңы теріңіз. WindowsXP жүйесінде *Іске қосу → Орындау (Start → Run)* батырмаларын басып, сұхбат терезесінде командаңы теріңіз. Vista немесе Windows7 орнатылған компьютерлерде алдымен `telnet` программасын (немесе осындай басқа программаны) орнатып, іске қосу қажет болуы мүмкін. Командаңы орындау нәтижесінде осы машинаның 25 портымен telnet-байланыс (яғни TCP-байланыс) орнатылады. *6.4-кестеде* көрсетілгендей 25 порт SMTP-порт болып саналады (*6.4-кестені* қараңыз, басқа стандартты хаттамалар портымен). Енгізілген командаға жауап ретінде:

```
trying 192.30.200.66…
connected to mail.isp.com
escape character is ‘‘]’
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

жолды аласыз.

Алғашқы үш жол telnet-ке жөнелтіледі және біз үшін болып жатқан жағдайды түсіндіреді. Соңғы жолды сервер қашықтықтағы машина SMTP-на жөнелтіледі және сіздің машинаңызбен сұқбаттасуға және пошта қабылдауға дайын екенін білдіреді. Қолжетімді командалар жайлы білу үшін:

```
HELP
```

командасын теріңіз.

Осы сәттен бастап, егер сервер сізден пошталық мәлімдемелер қабылдауға дайын болса, *7.3-листингте* көрсетілген командалар тізбегімен алмасуға болады.

Пошталық мәлімдемелерді беру

Бастапқыда тұтынушы агенттері поштаны тасымалдайтын мәлімдемелерді тасымалдау агенттері орналасқан машинадан жүктелетін. Мұндай нұсқада мәлімдеме жөнелту үшін тек тұтынушы агентінің жоғарыда сипатталған диалогты пайдаланып, жергілікті пошталық сервермен байланысу мүмкіндігі болуы керек. Алайда, бұл нұсқа кеңінен қолданылмайды.

Тұтынушы агенттері көбіне, ноутбуктерде, үй компьютерлерінде және

мобильді телефондарда жұмыс істейді, ал олар Интернетке үнемі қосылып тұрмайды. Мәлімдемелерді жөнелту агенттері Интернетке тұрақты қосылып тұрған провайдерлер және ірі компаниялар серверлерінде жұмыс істейді. Бұл айырмашылық, тұтынушы Бостонда саяхаттап жүріп пошталық мәлімдеме жөнелту үшін, өзінің әдеттегі Сиэтлдегі пошталық серверіне жүгінуі тиіс дегенді білдіреді.

Бұл қашықтықтан орындалатын коммуникация өздігінен ешқандай мәселе туындатпайды. Нақты осындай жағдайлар үшін TCP/IP хаттамасы құрастырылған болатын. Алайда, қашықтықтағы тұтынушы басқа жерге хат жолдау үшін провайдерлер немесе компаниялар пошталық серверіне мәлімдеме жөнелту керек, ал бұл жағдайға олар құлшыныссыз қарайды. Мұндай сервер қоғамдық болып саналмайды. Бұдан басқа, бұл **ашық пошта станциясы (open mail relay)** түрлі спамерлерді қызықтырады. Бұл нағыз жөнелтушіні жасырып, мәлімдемені спам ретінде сәйкестендіруді қиындатуға болатындықтан орын алып отыр.

Осы ерекшеліктерді есепке ала отырып, SMTP әдетте, `AUTH` кеңейтілімі бар хаттарды жөнелту үшін пайдаланылады. Бұл кеңейтілім сервер поштамен жұмыс істеуді растау үшін, тұтынушы деректерін (тұтынушы аты және құпия сөзді) тексеруге мүмкіндік береді.

Поштаны берерде SMTP пайдаланудың тағы да бірнеше ерекшеліктері бар. Мысалы, 25 порттың орнына 587 порт іске қосылады және SMTP-сервер тұтынушы агенті жөнелткен мәлімдеме форматын тексеріп, түзетуі мүмкін. Хаттарды жөнелту кезінде SMTP-ның қолданылуы жайлы көбірек білу үшін RFC 4409-ды қараңыз.

Мәлімдемені тасымалдау

Поштаны жөнелтуші (мәлімдемені) агент тұтынушы агентінен мәлімдеме алған кезде ол SMTP пайдаланып, мәлімдемені поштаны тасымалдаудың қабылдаушы агентіне береді. Бұны жүзеге асыру үшін жөнелтуші бір тағайындалған адресті пайдаланады. *7.2-листингіндегі bob@ee.uwa.edu.au* адрестелген мәлімдемені қараңыз. Қандай пошталық серверге жеткізілуі тиіс?

Пошталық серверді дұрыс анықтау үшін DNS сұралады. Алдыңғы бөлімде біз DNS құрамына MX (mail exchanger – пошталық мәлімдемелермен алмасу жазбасы) тұратын түрлі жазбалардың қалай кіретінін айтқан болатынбыз. Бұл жағдайда DNS сұранысы *ee.uwa.edu.au* доменіне MX жазбасын жазу үшін орындалады. Бұл сұраныс бір немесе бірнеше пошталық серверден аттарды және IP-адрестердің реттелген тізімін қайтарады.

Сонан кейін поштаны тасымалдаудың жөнелтуші агенті поштаны тасымалдаудың қабылдаушы агентімен хабарласу үшін, пошталық сервер IP-адресінің 25 портында TCP-байланыс орнатады. Мәлімдемені тасымалдау үшін SMTP қолданылады. Поштаны тасымалдаудың қабылдаушы агенті кейін Боб мәлімдемені оқу үшін оны *bob* тұтынушысы пошталық жәшігіне

салады. Таратылған пошталық инфроқұрылым бар болған жағдайда жергілікті жеткізудің бұл қадамы мәлімдеменің компьютер арасында ауысуынан тұруы мүмкін.

Поштаны жеткізудің осы үдерісінің арқасында, пошта пошталық мәлімдемелерді жеткізудің бастапқы агентінен соңғы агентіне бір қадамда жеткізіледі. Мәлімдемені тасымалдау сатысында аралық сервер жоқ. Алайда, бұл үдеріс бірнеше рет қайталануы мүмкін. Бұған біз бір мысал келтірдік (мәлімдемені жөнелту агенті тізімді пайдаланған жағдайды). Бұл жағдайда мәлімдеме қабылдаушылар тізімін алады. Сонан кейін ол тізімнің жеке мүшелеріне бөлінеді де, оның жеке адресіне жөнелтіледі.

Поштаны қайта бағыттаудың тағы бір мысалы келесідей болады: Боб Массачусетск технологиялық институтын бітіруі мүмкін, сәйкесінше ол *bob@alum.mit.edu* адресі бойынша қолжетімді. Поштаны бірнеше аккаунттардан жинаудың орнына Боб осы адрес бойынша келген поштаны *bob@ee.uwa.edu* адресіне бағыттауы мүмкін. Бұл жағдайда *bob@alum.mit.edu* поштасына жөнелтілген хат екі рет жеткізіледі. Олар алдымен *alum.mit.edu* пошталық серверіне, сонан кейін *ee.uwa.edu.au* серверіне жөнелтіледі. Егер біз пошталық мәлімдемелерді жөнелту агенттері тұрғысынан қарайтын болсақ, бұл қадамдардың әрқайсысы толыққанды жеке жеткізу болып саналады.

Сонымен бірге спам жайлы да ұмытпау керек. Бүгінгі таңда он мәлімдеменің тоғызы осы категорияға жатады (McAfee, 2010). Ешкімнің де мұндай поштаны алғысы келмейді, бірақ одан қашып құтылу күрделі, себебі олар әдеттегі мәлімдемедей болып маскіленеді. Спам мүмкіндігін төмендету үшін алдымен мәлімдемені жөнелтпес бұрын қосымша тексеру жүргізген дұрыс. Бобқа жазылған мәлімдеме *alice@cs.washington.edu* адресінен жөнелтілген болатын. Поштаны тасымалдаудың қабылдаушы агенті поштаны тасымалдаудың жөнелтуші агентіне DNS арқылы жүгіне алады. Бұл TCP-байланыстың екінші басындағы IP-адрестің және DNS-аттың сәйкестігін тексеруге мүмкіндік береді. Жалпы жағдайда қабылдаушы агент жөнелтуші доменді DNS-те тексеріп, оның пошта жөнелту саясатын қолдайтындығын анықтай алады. Бұл ақпарат көбіне, TXT- және SPF-жазба түрінде беріледі. Онда басқа да тексерулер жүргізуге болатындығы жайлы жазылады. Мысалы, *cs.washington.edu*-ден жөнелтілетін пошта әрқашан *june.cs.washington.edu* хостынан жөнелтілуі мүмкін. Егер пошта тасымалдаудың жөнелтуші агенті *june* болмаса, онда мәселе туындайды.

Егер бұл тексерулердің ешқайсысы нәтижелі болмаса, онда поштаның жасанды адрестен жөнелтілуі ықтимал. Бұл жағдайда мәлімдеме қабылданбайды. Алайда, ол бұл тексерістердің барлығынан өтсе де, оның спам еместігіне кепілдік жоқ. Пошта тек оның күтілген желі бөлігінен келгендігін тексереді. Бұның мағынасы, пошта жөнелткен кезде спамерлер оны нақты, дұрыс адрестен жөнелтуі тиіс. Бұл спамды дер кезінде анықтап, жоюға мүмкіндік береді.

7.2.5. Мәлімдемені соңғы жеткізу

Біздің пошталық мәлімдемеміз жеткізіліп қалды. Ол Боб пошта жәшігіне келді. Енді мәлімдемені бейнелеу үшін, оның көшірмесін Бобтың тұтынушы агентіне беру керек. Бұл *7.4-суреттегі* құрылымның 3-қадамы. Тұтынушы агенті және поштаны тасымалдау агенті бір компьютерде жұмыс істеп, тек әртүрлі үдерісті анықтайтын жағдайда, бұл қиын мәселе емес еді. Поштаны тасымалдау агенті – пошта жәшігі файлының соңына жаңа мәлімдеме жазады, ал тұтынушы агенті – пошта жәшігі файлында жаңа мәлімдеме бар екендігін тексереді.

Бүгінгі күнде дербес компьютерде, ноутбукте немесе мобильді телефонда жұмыс істейтін тұтынушы агенті провайдер немесе компания пошталық серверінде орналасуы ықтимал. Тұтынушылар, қай жерде жүргендеріне қарамастан өз пошталарының қолжетімді болғанын қалайды. Олар үшін пошта үй компьютерінде, жұмыста, іс сапарда болғанда ноутбукте немесе демалыста интернет-кафеде қолжетімді болуы тиіс. Оларға желімен үнемі байланыста болмай, тек қажет болған жағдайда шығыс мәлімдемені жөнелту және кіріс мәлімдемені қабылдау үшін Интернетке уақытша қосылып жұмыс істеу мүмкіндігі болу керек. Одан бетер, әр тұтынушы осы сәтте қандай компьютерде жұмыс істейтіндігіне байланысты бірнеше тұтынушы агентін жүктей алуы керек. Тіпті, бірнеше тұтынушы агенті параллель жұмыс істеуі мүмкін. Бұл жағдайда тұтынушы агенті пошта жәшігінің мазмұнын бейнелеуі және онымен қашықтықтан жұмыс істеуге мүмкіндік беруі тиіс. Бұл мақсатта тек SMTP емес, әртүрлі бірнеше хаттама іске қосылуы мүмкін. SMTP – бұл деректерді итермелеуге арналған хаттама. Мәлімдемені тасымалдау үшін оған қашықтықтағы сервермен байланысу қажет. Сөйтіп, соңғы жеткізу екі себеппен: пошта жәшігі поштаны тасымалдау агентінде сақталатындықтан және SMTP мәлімдемені жөнетпек болғанда тұтынушы агенті Интернетке қосылулы болмауы мүмкін болғандықтан орындалмауы мүмкін.

IMAP – Интернеттегі электронды поштаға қолжеткізу хаттамасы

Мәлімдемені соңғы жеткізу үшін қолданылатын негізгі хаттамалардың бірі – **IMAP (Internet Mail Access Protocol - Интернеттегі электронды поштаға қолжеткізу хаттамасы)**. Хаттаманың 4 версиясы RFC 3501-де анықталған. IMAP пайдалану үшін пошталық сервер 143 портты тексеретін IMAP-серверді іске қосады. Тұтынушы агенті IMAP-клиентті іске қосады. Бұл клиент сервермен байланысып, *7.8-кестеде* көрсетілген командалардың қажеттісін жөнелте бастайды.

Клиент алдымен қорғалған транспортты (мәлімдеме және команда құпиялығын сақтау үшін) іске қосады, сонан кейін логин және құпия сөзді енгізеді немесе серверде іске қосылады. Жүйеге кіргеннен кейін бумалар және мә-

лімдемелерді немесе мәлімдеме бөлігін көру үшін, хаттарды кейіннен өшіру үшін «қанат белгімен» белгілеуге және бумаларға орналастыру үшін көптеген командаларды пайдалануға болады. Түсініссіздік туындамас үшін, біз бұл тарауда «бума» (folder) сөзін пайдаланамыз, себебі пошта жәшігі бірнеше осындай бумалардан тұрады. Алайда, IMAP спецификациясы бума орнына «пошта жәшігі» (mailbox) терминін қолданады. Сөйтіп, тұтынушының бірнеше IMAP пошта жәшігі бар, олардың әрқайсысын тұтынушы бума ретінде көреді.

7.8-кесте. IMAP командалары (4-версия)

Команда	Сипаттама
CAPABILITY	Сервер мүмкіндіктерін атап шығу
STARTTLS	Қауіпсіз транспортты іске қосу (TLS, *8-тарауды* қараңыз)
LOGIN	Тұтынушы атын және құпия сөзді пайдаланып, серверге кіру
AUTENTICATE	Басқа әдіспен іске қосылу
SELECT	Буманы таңдау
EXAMINE	Тек оқуға арналған буманы таңдау
CREATE	Бума құрастыру
DELETE	Буманы өшіру
RENAME	Буманың атын өзгерту
SUBSCRIBLE	Активті жиынтыққа бума қосу
UNSUBSCRIBLE	Активті жиынтықтан буманы өшіру
LIST	Қолжетімді бумаларды атап шығу
LSUB	Активті бумаларды атап шығу
STATUS	Бума статусын білу
APPEND	Бумаға мәлімдеме қосу
CHECK	Таңдап алынған буманың қалып-күйін қарау («қалы-күйге» нақты ненің кіретіні сервердің нақты жүзеге асырылуына байланысты). Бума үшін бақылау нүктесін құрастыру
FETCH	Бумадағы мәлімдемелерді қарау
SEARCH	Бумадағы мәлімдемені табу
STORE	Мәлімдемелер белгісін өзгерту
COPY	Бумада мәлімдеме көшірмесін жасау
EXPUNGE	Белгіленген мәлімдемелерді өшіру
UID	Бірегей идентификаторды пайдаланып команданы шақыру
NOOP	Ешқандай әрекет орындамау
CLOSE	Белгіленген мәлімдемелерді өшіріп, буманы жабу
LOGOUT	Жүйеден шығып, байланысты жабу

IMAP хаттамасының түрлі мүмкіндіктері бар, мысалы, поштаны келу реті, хат атрибуттары (мысалы, «маған алдымен Элис хаттарын беріңіз») бойынша сұрыптау. Сонымен бірге, серверде белгілі бір шартты қанағаттандыратын клиенттерді көру үшін іздеу құралдары болуы мүмкін.

IMAP – ертеде құрастырылған соңғы жетткізу хаттамасы **POP3-тің (Post Office Protocol, version 3 – пошта бөлімінің хаттамасы, 3-версия)** жақсартылған версиясы. Ол RFC 1939-да анықталған. POP3 хаттамасы қарапайым, бірақ оның мүмкіндікттері аз және әдеттегі қолданыста қауіпсіз емес. Әдетте, пошта пошталық серверде қалмайды, ол компьютерде тұтынушы агентімен жүктеледі. Бұл сервер жұмысын жеңілдетеді, бірақ тұтынушы жұмысын қиындатады. Поштаны бірнеше компьютерден оқу әлдеқайда күрделі және егер тұтынушы агенті орналасқан компьютер істен шықса, бүкіл пошта жоғалады, оны қалыпқа келтіруге мүмкіндік болмайды. Осылай бола тұра POP3 әлі де қолданыста.

Сонымен бірге, хаттама пошталық сервер және тұтынушы агенті арасында жұмыс істейтін және оны бір компания қамтамасыз етуі мүмкін болғандықтан, жеке хаттамалар да қолданылуы мүмкін. Мұндай, жеке хаттамасы бар пошта жүйесінің мысалы Microsoft Exchange бола алады.

Веб-пошта

Кең және қарқынмен дамып келе жатқан пошталық қызметті ұсынуда IMAP және SMTP-ға балама таңдау – мәлімдемені жөнелту және қабылдау үшін веб-интерфейсті пайдалану. Кеңінен қолданылып келе жатқан **веб-пошта жүйесіне (Webmail)** Google Mail, Microsoft Hotmail және Yahoo! Mail кіреді. Веб-пошта – бұл программалық жабдықтаманың бір мысалы (бұл жағдайда тұтынушы пошта агенті) веб-технологияны пайдаланып, қызмет түрі ретінде ұсынылады.

Мұндай құрылымда провайдер пошта қызметтерін әдеттегідей SMTP-ны пайдаланып, 25 порт арқылы мәлімдемелерді қабылдауды басқарады. Алайда, тұтынушы агенті өзгеше. Ол жеке программа емес, веб-парақ арқылы ұсынылатын тұтынушы интерфейсі. Бұл тұтынушылар поштаға қолжеткізіп, мәлімдеме жөнелту үшін өзіне ұнаған кез келген браузерді пайдалана алады деген сөз.

Біз әзірше Интернеттегі және Дүниежүзілік Өрмектегі веб-технологиялар жайлы айтқан жоқпыз, бірақ оның қысқаша мазмұны келесідей: тұтынушы провайдердің пошталық веб-парағына кірген кезде ол Тұтынушы аты және Құпия сөз өрістерін толтыруды қажет ететін форманы көреді. Құпия сөз және Тұтынушы аты серверге жөнелтіліп, олардың дұрыстығы тексеріледі. Егер жүйеге ену ойдағыдай жүргізілсе, сервер тұтынушының пошталық жәшігін тауып, пошталық жәшік мазмұны бейнеленетін веб-парақ құрастырады. Бұл веб-парақ клиент браузеріне жөнелтіледі.

Тұтынушы өзінің пошта жәшігін бейнелейтін парақтың көптеген элементтерін пайдалана алады, сондықтан мәлімдемені оқуға, өшіруге және т.б. әрекеттер орындауға болады. Интерфейс тұтынушының әрекеттеріне жақсы жауап қайтару үшін, веб-параққа жиі JavaScript программасы қосылады. Бұл программа белгілігі бір оқиғаға (мысалы, тышқан қолтетігін шертуге) жауап ретінде жергілікті тұтынушы машинасынан іске қосылады және келесі хатты немесе жаңа хаттарды көрсету үшін мәлімдемелерді фондық режимде серверге және серверден жүктей алады. Бұл модельде поштаны ұсыну қарапайым веб-хаттамаларды пайдаланып, деректерді URL бойынша жөнелтуді арқылы жүргізіледі. Веб-сервер мәлімдемелерді біз жоғарыда қарастырған, әдеттегі поштаны жеткізу жүйесіне орналастырады. Бұл хаттамалар парақтың поштылық мәлімдеме болуымен емес, веб-парақттарды шифрлаумен байланысты.

7.3. ДҮНИЕЖҮЗІЛІК ӨРМЕК (WWW)

Дүниежүзілік өрмек (WWW, World Wide Web, қысқалық үшін жиі жай «веб») – бүкіл Интернет бойынша миллиондаған машиналарда орналасқан, байланысқан контентке қолжеткізудің негізі болып келетін құрылым. Пайда болғаннан бергі он жыл ішінде ол Швейцариядағы физиканың жоғарғы энергия бойынша эксперименттер құрылымын үйлестіру құралынан қызығушылықтары әртүрлі миллиондаған адамдар «Интернет» деп қабылдайтын, қосымшаға айналды. Бұл қосымшаның үлкен танымалдығы, тіпті, жаңа қосылған адамда онымен ешбір қиындықсыз жұмыс істей алатын жағдайға әкелді. Сонымен бірге, түсті графикалық интерфейс арқасында Дүниежүзілік өрмек іс жүзінде, Африка құмырсқасынан бастап, яшмадан жасалған фарфорға дейінгі, кез келген тақырыпта үлкен көлемдегі ақпаратты ұсынады.

Дүниежүзілік өрмек 1989 жылы, Швейцарияда **CERN (Conseil Europeen pour la Recherche Nucleaire)** Еуропалық ядролық зерттеулер орталығында құрылған болатын. Бастапқыдағы негізгі мақсат – мүшелері әртүрлі елдерде және әртүрлі сағаттық аумақтарда тұратын, үлкен топтар арасындағы әрекеттесуді орнықтыру болатын. Олар элементар бөлшек физикасы төңірегінде жүргізілген эксперименттер барысында үнемі өзгеріп отыратын жұмыс жайлы есептермен, сызбалармен, суреттермен, фото-суреттермен және басқа да құжаттармен алмасулары керек болатын. Бір-бірімен байланысқан құжаттар арасында өрмек құру идеясы, CERN орталығының физигі Тим Бернерс-Лиге (Tim Berners Lee) келді. 1991 жылдың желтоқсан айында, Техас штатының Сан-Антонио қаласында өткен Hypertext’91 конференцияда жария көрсету болды. Бұл көрсетілім басқа ғалымдардың назарын аударды. Марк Андрессен (Marc Andreessen) Иллинойс университетінде алғашқы графикалық браузер Mosaic-ті құрастыруды бастады. Программа 1993 жылдың ақпан айында жарық көрді.

Қалғанының барлығы бүгінгі таңда тарих. Mosaic-тің танымал болғандығы соншалық, оның авторы Марк Андрессен бір жылдан кейін өзінің жеке

Netscape Communication Corp. компаниясын құрастыруды ұйғарды. Компания Дүниежүзілік өрмек үшін программалық жабдықтама (немесе қысқаша веб үшін ПЖ) құрастырумен айналысты. Келесі үш жыл ішінде Netscape Navigator және Microsoft-тың Internet Explorer арасында нағыз «браузерлер шайқасы» басталды. Екі жақтыңда құрастырушылары, өз программаларын жаңа функциялармен толықтырып (демек, қателіктермен де), бәсекелесінен асып түсіп, жаңа нарықтың үлкен бөлігін қамтуға тырысты.

1990 және 2000 жылдарда веб-сайттар және веб-парақтар, сонымен бірге веб-контенттер, сайттар саны миллионмен, ал парақтар миллиардпен есептелгенше, экспонента бойымен өсті. Бұл сайттардың аздаған бөлігі кеңінен танымал болды. Қазір осы сайттар және олардың артында тұрған компаниялар вебтің бүгінгі түрін шындап анықтап отыр. Мысал ретінде кітап дүкенін (Amazon, 1994 жылы құрылған, нарықтың бағасы $50 млрд), бүрге базарын (eBay, 1995, $30 млрд), іздеуіш (Google, 1998, $150 млрд) және әлеуметтік желіні (Facebook, 2004, жеке компания $15 млрд аса бағаланады) келтіруге болады. 2000 жылдың сол кезеңінде, көптеген веб-компаниялардың бағасы бір түн ішінде жүздеген миллион доллардан асып, ертеңіне құлап жатты (олардың тек жарнама ретінде құралғаны белгілі болған кезде), тіпті, олардың аттары да бар. Олар **«даткомдар эрасы» (dot com era)** деп аталады. Веб-тегі жаңа идеялар қазір де осылайша дамиды. Олардың көбін студенттер ұсынады. Мысалы, Facebook жобасын ойлап тауып, іске қосқан Марк Цукерберг сол кезде Гарвард студенті болған, ал Google іске қосылған кезде Сергей Брин және Лари Пейдж Стенфорд студенті болған. Мүмкін сіз келесі боларсыз.

1994 жылы CERN және Массачусетск технология институты (M.I.T., Massachusetts Institute of Technologies) **WWW-концорциумын** (**World Wide Web Consortium**, кейде қысқаша **W3C**) құрастыру жайлы келісімге отырды. WWW-концорциумының мақсаты – Дүниежүзілік өрмектің әрі қарай дамуы, хаттамаларды стандарттау және жеке сайттар арасындағы әрекеттесуді ынталандыру. Бернерс Ли концорциум директоры болды. Дүниежүзілік өрмек жайлы көптеген кітаптар жазылса да, ол жайлы жаңа ақпаратты алуға болатын жер – Дүниежүзілік өрмектің өзі. Концорциумның үй парағын *http:/www.w3c.org* адресі бойынша табуға болады. Қызығушылық танытқан оқырман осы парақтан концорциумның басқа құжаттары және қызметі жайлы ақпарат орналасқан парақтар адресін таба алады.

7.3.1. Құрылым жайлы ұсыныс

Тұтынушы тұрғысынан Дүниежүзілік өрмек **веб-парақ (Web pages)** формасындағы, үлкен санды контенттен тұрады, олар қысқалық үшін көбіне **парақтар (pages)** деп аталады. Әр парақтың өзімен байланысқан, әлемнің кез келген нүктесінде орналасқан басқа парақтарға сілтемесі болуы мүмкін. Тұтынушылар сілтеме арқылы басқа параққа көше алады (мысалы, жәй тышқан қолтетігімен шерту арқылы) және сілтеме көрсетіп тұрған парақ

браузер терезесінде бейнеленеді. Бұл үдерісті шексіз қайталауға болады. Қазір **гипермәтін (hypertext)** деп аталатын, өзара байланысқан парақтар идеясын алғаш рет 1945 жылы электротехникамен айналысатын Массачусетск институтының профессоры Ванневар Буш (Vannevar Buch) ұсынған болатын. Іс жүзінде ол коммерциялық компьютерлерден де бұрын пайда болған, университеттерде оның көлемі үлкен бір бөлмені алатын, ал қуаты қалта калькуляторынан аспайтын, дөрекі түптұлғасы болған.

Парақтар **браузер** (**browser)** деп аталатын арнайы программалармен қаралады. Ең танымал браузерлер – Firefox, Internet Explorer және Chrome. Браузерлер тұтынушыға сұралған парақты ұсынады, оның контентін интерпретациялайды және талап бойынша форматталған парақты экранға шығарады. Контент мәтін, сурет және форматтау командаларының үйлесімі ретінде ұсынылып, қарапайым құжат немесе бейне немесе белгілі бір графикалық интерфейсі бар және сол аумақта тұтынушы жұмыс істей алатын программа ретінде көрінуі мүмкін.

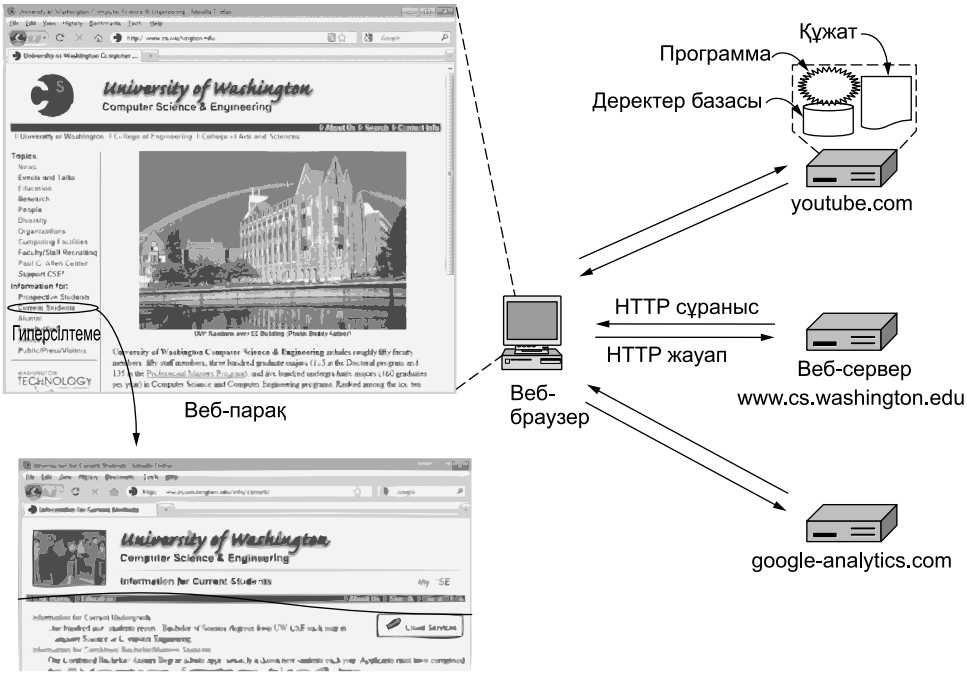
Веб-парақ мысалы, *7.7-суреттің* сол жақ жоғарғы бұрышында көрсетілген. Бұл – Вашингтон университетінің информатика және есептеу техникасы факультетінің парағы. Онда мәтін және графикалық элементтер бар (әрине жақсылап көрінетін масштабта емес). Парақтың кейбір бөліктері сілтеме арқылы басқа парақтармен байланысқан. Басқа параққа сілтемені білдіретін мәтін шартты белгі, сурет және т.б. **гиперсілтеме (hyperlink)** деп аталады. Сілтеме арқылы көшу үшін тұтынушы тышқан қолтетігі көрсеткішін ерекшеленген аумаққа орналастырады, тышқан қолтетігі көрсеткішінің формасы өзгереді, сол кезде оған шерту керек. Сілтеме арқылы көшу браузерге басқа парақты жүктеу керектігін хабарлайды. Бастапқыда, парақ бетінде ажырату үшін, сілтемелер астын сызу арқылы немесе басқа түспен ерекшеленеді. Қазіргі кезде құрастырушылар веб-парақтардағы сілтемелердің түрін үнемі қадағалап отырады, сондықтан олар шартты белгі түрінде немесе сол аумақта тышқан қолтетігі көрсеткіші пайда болғанда сырт пішінін өзгеруі мүмкін. Сілтеменің сыртқы түрін көрер көзге ерекше етіп, оларға жарамды интерфейсті қамтамасыз ету – сайт құрастырушылар мәселесі.

Факультет студенттері өздеріне арналған сілтеме арқылы көшіп, қосымша ақпарат ала алады. Сілтеме қорғалған өріске шерткен кезде екпінді болады. Осы кезде браузер желіден жаңа парақты алады және оны экранда бейнелейді (*7.7-суреттің* сол жақ төменгі бұрышында көрсетілгендей). Алғашқы парақта бұдан басқа ондаған сілтемелер бар. Әр жаңа парақ, алғашқы парақ орналасқан машинада немесе Жер бетінің қарама-қарсы бөлігіндегі компьютерде орналасқан контенттен тұруы мүмкін. Тұтынушы үшін ол байқалмайды. Браузер сұралған парақты тұтышушының қатысынсыз жеткізеді. Сөйтіп, тұтынушы контентті қарап отырып, үнемі компьютерлер арасында қозғалыста бола алады.

Парақтарды бейнелеудің негізгі қағидасы *7.7-суретте* көрсетілген. Браузер веб-парақты клиент жақта бейнелейді. Әр парақ сұранысты бір немесе бірнеше

серверлерге жөнелту арқылы бейнеленеді. Серверлер парақ контентін жөнелту арқылы жауап береді. Парақты бейнелеуге қажет сұраныс-жауап хаттамасы – бұл SMTP жағдайындағыдай, TCP арқылы жұмыс істейтін қарапайым мәтіндік хаттама. Ол **HTTP (HyperText Transfer Protocol – гипермәтінді тасымалдау хаттамасы)** деп аталады. Контент – дисктен оқылатын қарапайым құжат немесе программа жұмысының нәтижесі немесе деректер базасына жөнелтілген сұраныс нәтижесі болуы мүмкін. Егер парақ үнемі бірқалыпты бейнеленетін тұрақты құжат болса, онда ол **статикалық (static page)** деп аталады. Егер парақ керісінше, программа сұранысы бойынша құрастырылатын немесе өзі қандай да бір программадан тұратын болса, онда ол **динамикалық (dynamic page)** деп аталады.

Динамикалық парақ контенті әр шақырылған сайын әртүрлі болуы мүмкін. Мысалы, электронды дүкеннің негізгі парағы әртүрлі тұтынушы үшін әртүрлі болады. Егер кітап дүкенінің клиенті өткен жолы мистикалық романды алған болса, дүкеннің негізгі парағына кірген кезде ол жаңа триллер бейнесін көруі мүмкін, ал жаңа рецептерге қызығатын адам осы кезе ол беттен жаңа аспаздық кітапты көруі мүмкін. Біз веб-сайттардың кім нені сатып алатындығын қалай қадағалайтынын қысқаша айтып кетеміз. Егер егжей-тегжейіне тоқталмайтын болсақ, мәселенің барлығы cookie файлында (Тіпті, аспаздыққа қызықпайтындар үшін де).



7.7-сурет. Дүниежүзілік өрмек құрылымы

7.7-суретте браузер екі парақты жүктеу үшін: *cs.washington.edu*, *youtube.com* және *google-analytics.com* сияқты үш серверге жүгінеді. Осы серверлерден алынған контент бірігіп, браузермен бейнеленеді. Бейнелеу кезінде контент типіне байланысты, өңдеу типі іске қосылады. Мәтін және графикалық элементтерден басқа, бейне файлды ойнату немесе парақтың бір бөлігі болып саналатын, жеке тұтынушы интерфейсі жазылған скрипті іске қосу қажет болуы мүмкін. Біздің жағдайымызда *cs.washington.edu* серверінен негізгі парақ, *youtube.com* серверінен онда жазылған бейне файл жүктеледі, ал *google-analytics.com* серверінен тұтынушыға көрінетін ешнәрсе жүктелмейді. Бұл сервер сайтқа кірушілерді тіркейді. Кейінірек біз трекерлер жайлы айтамыз.

Клиент жақ

Енді *7.7-суретке* сүйене отырып, веб-браузер жақты егжей-тегжейлі қарастырайық. Іс жүзінде браузер – бұл веб-парақты бейнелей алатын және екпінді парақ бетіндегі элементтерге тышқан қолтетігі көрсеткішімен шерткенді ажырата білетін программа. Элементті таңдау кезінде браузер гиперсілтеме бойынша жүріп, серверден сұралған парақты алады.

Дүниежүзілік өрмек құрастырылған кезде, бір парақтан келесі бір парақка сілтеменің болуы, парақтарға ат беру және орналастыру механизмін құрастыруды қажет ететіндігі белгілі болды. Таңдап алынған парақты бейнелемес бұрын, ең маңыздысы:

1. Парақ қалай аталады?
2. Ол қай жерде орналасқан?
3. Оған қалай қол жеткізуге болады?

– деген сұрақтарға жауап беру болды.

Егер әр параққа бірегей ат берілсе және оларды сәйкестендіруде ешқандай бір мағыналық болмаса. Осылай бола тұра ол мәселені шешпес еді. Парақтар және адамдар арасында параллель келтірейік. АҚШ-та әр адамның түгел дерлік әлеуметтік сақтандыру нөмірі бар. Бұл нөмір бірегей идентификатор болып саналады, себебі екі адамның әлеуметтік сақтандыру нөмірі бірдей болмайды. Дегенмен, егер сізде қандай да бір адамның тек осы сақтандыру нөмірі бар болса, сіз оның адресін және ол адамға қандай тілде, ағылшын, испан немесе қытай тілінде хат жазу керектігін біле алмайсыз. Дүниежүзілік өрмектегі орын алатын мәселе де осыған ұқсас.

Таңдап алынған шешім парақты барлық сұрақтар бірден жауабын табатындай етіп сәйкестендіреді. Әр параққа, оның Дүниежүзілік өрмектегі аты болып саналатын **URL (Uniform Resource Locator – ақпараттық ресурстың жүйеленген көрсеткіші)** адресі берілді. URL: **хаттама** (сонымен бірге **схема - scheme** деп те аталады), парақ орналасқан машинаның DNS-аты және жеке файлды бірегей анықтайтын жол (оқуға арналған файл немесе машинада іске қосуға арналған программа) болып үшке бөлінеді. Жалпы жағдайда жол

– файлдар каталогы құрылымын моделдейтін иерархиялық атау. Алайда, жолды интерпретациялау – сервердің жұмысы. Каталогтың нақты құрылымы бейнеленбеуі де мүмкін.

Мысал ретінде *7.7-суретте* көрсетілген URLпарақты келтірейік:

http://www.cs.washington.edu/index.html

Бұл – URL: хаттама (*http*), хосттың DNS-аты (*www.cs.washington.edu*) және жол аты (*index.html*) сияқты үш бөліктен тұрады.

Тұтынушы гиперсілтемеге тышқан қолтетігі көрсеткішімен шерткен кезде, браузер сілтеме көрсетіп тұрған парақты жүктеуге әкелетін бірнеше әрекетті орындайды. Сілтеме таңдалғаннан кейін орын алатын әр әрекетті қарастырайық:

1. Браузер URL анықтайды (парақтың таңдап алынған элементі бойынша).
2. Браузер DNS қызметінен *www.cs.washington.edu* серверінің IP-адресін сұрайды.
3. DNS 128.208.3.88 жауабын қайтарады.
4. Браузер 128.208.3.88 машинасының 80-портымен (HTTP-хаттамаға арналған жалпыға белгілі порт) TCP-байланыс орнатады.
5. Браузер /*index.html* файлын алуға HTTP-сұраныс жөнелтеді.
6. *www.cs.washington.edu* сервері HTTP-жауап ретінде парақты, мысалы, /*index.html* файлын жөнелтеді.
7. Егер парақта бейнелеуге қажетті URL бар болса, онда браузер осы үдерісті пайдаланып, басқа URL-ді алады. Бұл жағдайда URL-де бейнелеуге қажет *www.cs.washington.edu* алынған көптеген суреттер, сонымен бірге *youtube.com* алынған бейне және *google-analitics.com* скрипт бар.
8. Браузер /*index.html* парағын *7.7-суретте* көрсетілгендей бейнелейді.
9. Егер біраз уақыт ішінде сол серверлерге басқа сұраныс түспесе, TCP-байланыс үзіледі.

Көптеген браузерлер өзінің ағымдағы орындап жатқан әрекетін экранның төменгі бөлігіндегі қалып-күйді қатарында көрсетіп отырады. Бұл тұтынушыға төмен өнімділік себебін түсінуге көмектеседі: мысалы, DNS қызметі немесе сервер жауап бермейді немесе парақты тасымалдау кезінде желі асыра жүктелген.

URL-дизайн бұл мағынада шектелмеген және браузерлерге әртүрлі ресурстарды өңдеу үшін көптеген түрлі хаттамаларды пайдалануға мүмкіндік береді. Іс жүзінде URL көптеген басқа хаттамалар үшін де анықталған. URL схемасының біршама қарапайымдалған формасы *7.9-кестеде* келтірілген.

Осы тізімге қысқаша тоқталайық. *http* хаттама – бұл барлық веб-серверлер сөйлейтін, Дүниежүзілік өрмектің ана тілі. **HTTP** – бұл **HyperText Transfer**

Protocol (гипермәтінді тасымалдау хаттамасы) қысқартылған түрі. Ол жайлы кейінірек егжей-тегжейлі әңгімелейміз.

ftp – Интернетте файлдарды тасымалдау хаттамасы, файлдарға FTP арқылы қолжеткізу үшін қолданылады. FTP Дүниежүзілік өрмектің алдында пайда болды. Ол отыз жылдан бері қолданыста. Веб – командалық жолға команда енгізу арқылы құрастырылған интерфейс орнына тышқан қолтетігі жұмысына негізделген қарапайым интефейсті ұсына отырып, әлемнің түкпір-түкпірінде орналасқан түрлі FTP-серверлердегі файлдарға жеңіл қол жеткізуге мүмкіндік береді. Ақпаратқа қол жеткізу тәсілінің жеңілдеуі – Дүниежүзілік өрмектің танымалдығының өсу және оның жылдам даму себептерінің бірі.

7.9-кесте. URL-дің кейбір стандартты схемалары

Атау	Қолданылуы	Мысал
http	Гипермәтін (HTML)	http://www.ee.uwa.edu/~rob/
https	Қауіпсіздігі қамтамасыз етілген гипермәтін	https://www.blank.com./accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Жергілікті файл	file://usr/suzanne/prog.c
mailto	Поштаны жөнелту	mailto:JohnUser@asm.org
rtsp	Мультимедианы ағындық тасымалдау	rtsp://youtube.com/montypython.mpg
sip	Мультимедиалық қоңырау	sip:eve@adversary.com
about	Браузер ақпараты	about:plugins

Жергілікті файлға *file* хаттамасын пайдаланып немесе жай файл атын жазып, веб-парақ тәрізді қол жеткізуге болады. Бұл амалды пайдалану үшін сервердің қажеті жоқ. Әрине, бұл тек жергілікті файлдар үшін ғана жұмыс істейді, қашықтықтағы файлдар үшін емес.

mailto хаттамасы іс жүзінде веб-парақты сұрамайды, бірақ бәрібір оның пайдасы бар. Оның көмегімен поштаны веб-браузер арқылы жөнелтуге болады. Көптеген браузерлер *mailto* сілтемесі арқылы көшуге тұтынушы агентін іске қосумен жауап қайтарады, ол жерде алдын ала қосылған адреске мәлімдеме жазуды бастауға болады.

rtsp және *sip* хаттамалары мультимедиялық ағындар сессиясын, сонымен бірге аудио және бейне қоңырауларды тасымалдауға арналған.

Соңында *about* хаттамасы браузер жайлы ақпаратты ұсынады. Мысалы, егер сіз *about:pulgins* сілтемесі арқылы көшсеңіз, көптеген браузерлер, браузер кеңейтілімі (плагиндер) арқасында қол жетімді MIME типтер тізімі көрсетілген парақты бейнелейді.

Қысқаша айтқанда, URL Дүниежүзілік өрмек арқылы жылжу үшін ғана емес, сонымен бірге FTP және электронды пошта тәрізді ескі хаттамаларды және жаңа аудио, бейнені іске қосу үшін, жергілікті файлдарға және брау-

зер ақпараттарына ыңғайлы түрде қолжеткізу үшін де құрастырылған болатын. Осы әдіс арқасында жоғарыдағы қажеттіліктер үшін тұтынушы интерфейсін ұсынатын барлық программалар керек емес. Ал Интернетке қолжеткізу толығымен бір программаға кіріктірілген: веб-браузер. Бұл ойдың алдымен Щвейцариядағы зерттеу зертханасында жұмыс істейтін британ физигіне келгенін болмаса, оны программалық жабдықтамалар шығарушы компаниясының жарнама бөлімінің ойлап тапқан әдемі жоспары деуге болар еді.

Барлық осы үздік сапасына қарамастан, Дүниежүзілік өрмекті үнемі пайдаланудың өсуі URL схемасының біраз кемшіліктерін анықтады. URL жеке хостқа нұсқайды, кейде парақтың қайда орналасқандығын көрсетпей өзіне нұсқаған дұрыс. Мысалы, егер сілтеме көп жасалатын парақтар көптеп көшіріліп, желінің әртүрлі бөліктерінде сақталса, бұл трафикті азайтуға көмектесер еді. Алайда, біз: «Маған *xyz* парағы керек және оның қай жерден келетіні бәрібір», – деп айта алмаймыз.

Бұл мәселені шешу үшін URL **URI-ға (Uniform Resource Identifier – ресурстың әмбебап идентификаторы)** дейін жинақталды. Кейбір URI ресурстың орналасқан орнын емес атын көрсетеді. Мұндай URI **URN (Uniform Resource Name – ресурстың жүйеленген аты)** деп аталады. URI-дың жазылу ережесі RFC 3986-да көрсетілген, ал оның басқа қолданылу схемаларын IANA анықтайды. URI-дың *7.9-кестеде* көрсетілгендерден басқа көптеген түрлері бар, біз тек қазіргі таңда Дүниежүзілік өрмекте кеңінен қолданылатындары атап шықтық.

MIME типтері

Жаңа (және әр) парақты бейнелеу үшін браузер оның форматын түсінуі керек. Барлық браузерлер кез келген парақты бейнелеу үшін стандартталған HTML тілінде жазылады. Қазіргі таңда ол – Интернетте көпшілік қабылданған тіл. Ол жайлы біз төменде әңгіме қозғаймыз.

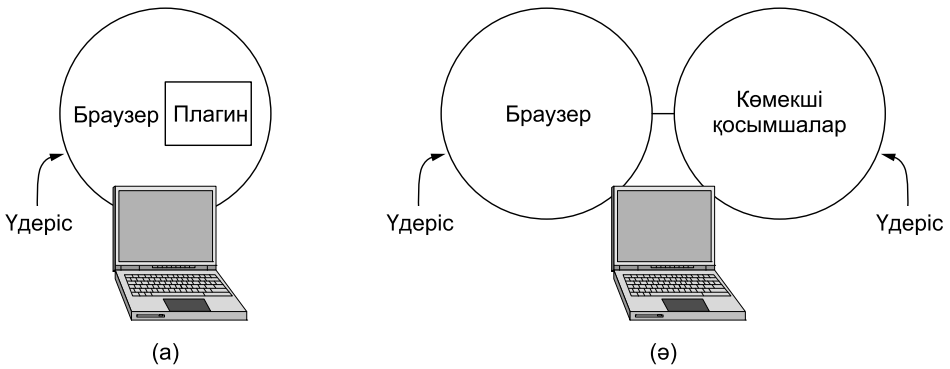
Іс жүзінде браузер HTML интерпретаторы болғанымен, көптеген браузерлер Дүниежүзілік өрмекте қозғалуды жеңілдететін әртүрлі батырмалармен және функциялармен жабдықталған. Көптеген браузерлердің алдыңғы параққа қайтып оралу және келесі параққа көшу (егер тұтынушы бұның алдында кері оралған болса), сонымен бірге тұтынушы таңдап алған ағашқы параққа тікелей оралу батырмасы бар. Браузерлер сонымен бірге ағымдағы параққа бетбелгі қою меню командасын және бетбелгілер тізімін бейнелеуді қолдайды. Бұл кез келген параққа тышқан қолтетігі көрсеткішімен бір шертіп қол жеткізуге мүмкіндік береді.

Біздің мысалымызда көрсетілгендей, HTML-парақта мәтін және гипермәтіннен басқа контенттің әртүрлі элементтері болуы мүмкін. Қатаң айтсақ, барлық парақтардың HTML болуы міндетті емес. Парақтар MPEG форматындағы бейнеден, PDF форматындағы құжаттан, JPEG форматындағы

суреттен, MP3 форматындағы әннен және жүздеген әртүрлі типтегі файлдан тұруы мүмкін. Браузерде сәйкестендіруге келмейтін парақпен мәселе туындайды.

Браузер көлемін және оған әртүрлі типтегі файлдар (саны жылдам өсіп келе жатқан) интерпретаторын қосып мүмкіндігін кеңейткенше, әдетте, жалпылама шешім қабылданады. Сервер жауап ретінде қандайда бір парақты қайтарған кезде онымен бірге қосымша ақпарат жөнелтіледі. Бұл ақпарат парақтың MIME-типін көрсетеді (*7.6-кестені* қараңыз). *Text/html* типіндегі парақтарды басқа да кіріктірілген типтер тәрізді, браузер тікелей шығарады. Ал егер бұл MIME-тип үшін ішкі сәйкестендіру мүмкін болмаса, онда браузер парақты қалай бейнелу керектігін өзінің MIME-типтер кестесі бойынша анықтайды. Бұл кестеде әр типке сәйкес бейнелеу программасы келтірілген.

Бейнелеудің: іске қосылатын модуль, **плагин (plug-in)** немесе көмекші қосымша арқылы сияқты екі тәсілі бар. Іске қосылатын модуль – бұл браузер қатты дискідегі арнайы каталогтан алатын ерекше сыртқы код. Браузер оны *7.8-суретте* көрсетілгендей, өзінің кеңейтілімі ретінде орнатады. Плагиннің стандартты мысалы ретінде PDF, Flash және Quick-time арналған, құжаттармен жұмыс істеп, аудио және бейнені қарауға арналған плагиндерді айтуға болады. Іске қосылатын плагиндер браузер ішінде жұмыс істейтін болғандықтан, олар түрін өзгертетін ішкі параққа қолжеткізе алады. Өз жұмысын аяқтағаннан кейін (әдетте, бұл тұтынушының басқа параққа көшуімен баланысты) плагин браузер жадысынан жойылады.



7.8-сурет. Іске қосылатын модулы бар браузер (а); қосалқы қосымша (ә)

Әр браузердің іске қосылатын модульдерді жұмысын жүзеге асыратын процедуралар жиынтығы бар. Бұл браузер плагинмен әрекеттесу үшін қажет. Мысалы, стандартты процедураға бейнелейтін деректерді беретін браузердің стандартты коды бар. Осы процедуралар жиынтығы іске қосылатын модуль интерфейсін құрайды және әр нақты браузер үшін арнайы болып келеді.

Браузер іске қосылатын модульге бұдан да басқа өзінің белгілі бір процедуралар жиынтығын ұсынады. Олардың ішінде браузер интерфейсіне

әдетте, тарату және жадыны босату, браузер қалып-күй қатарына мәлімдеме шығару және оның параметрлерін сұрау процедурасы кіреді.

Плагинді пайдаланбас бұрын оны орнату керек. Бұл, тұтынушы плагин шығарушының веб-сайттан орнатылатын файлдың көшірмесін алады дегенді білдіреді. Орнатылатын файлды іске қосу, плагинді ашып, браузерді MIME-типті тіркейді және осы типті модульмен сәйкестендіреді. Әдетте, танымал плагиндер браузерлерде алдын ала жүктеледі.

Браузер мүмкіндігін кеңейтудің екінші тәсілі – **көмекші қосымшаларды (helper application)** пайдалану. Көмекші қосымшалар – бұл жеке үдеріс ретінде жұмыс істейтін, толыққанды программа. Ол *7.8 ә-суретінде* көрсетілген. Ол браузермен байланысты болмағандықтан, олардың арасында ешқандай интерфейс жоқ. Көмекші қосымшалар браузерден бейнелеуді қажет ететін деректер орналасқан уақытша файл атын алады да, оны ашып контентті бейнелейді. Әдетте, көмекші қосымша ретінде браузерден жеке жұмыс істейтін үлкен программалар жүреді, мысалы, Microsoft Word немесе Power Point.

Көптеген көмекші қосымшалар *application* (қосымша) MIME-типін пайдаланады. Нәтижесінде көптеген ішкі тип анықталған болатын, мысалы *application/vnd.ms-powerpoint* Power Point файлдары үшін. Мұнда *vnd* файлды ұсынушыға тәуелді форматты білдіреді. Сөйтіп, URL Power Point файлына тікелей нұсқай алады және тұтынушы оған шерткен кезде Power Point автоматты түрде іске қосылып, қажет файлды өңдеп, бейнелейді. Көмекші қосымшаларды тек *application* MIME-типі ғана қолданбайды. Мысалы, Adobe Photoshop *image/x-photoshop* пайдаланады.

Сонымен, браузерге ешқандай өзгеріс енгізбей, кез келген типтегі файлды өңдейтіндей баптауға болады. Қазіргі заманғы веб-серверлерде типтер және ішкі типтердің жүздеген комбинациясы бар. Жаңа программа орнатқан сайын жаңа тип пайда болады.

Шиеленіс көзі – көптеген плагиндер және көмекші қосымшалардың бір типті өңдейтіндігі, мысалы, *video/mpg*. Нәтижесінде, соңғы болып тіркелген программа өз жазбасымен қолданыстағы MIME типтер қауымдастығын өшіріп, типті өзіне қалдырады. Бұның салдары, әр орнатылатын программа браузердің кейбір типтерді бейнелеу тәсілін өзгертуі мүмкін.

Браузерлер жергілікті файлдармен, желімен байланыссыз, қашықтықтағы серверден ақпарат сұрамай жұмыс істей алады. Алайда, браузерге қандайда бір жолмен файлдың MIME-типін анықтау керек. Стандартты тәсіл – операциялық жүйе деңгейінде файл кеңейтілімін MIME-типпен сәйкестендіру. Стандартты конфигурацияда *foo.pdf* файлын ашу талпынысы оны браузерде *application/pdf* плагині көмегімен ашуға әкеледі, ал *bar.doc* файлы Word көмекші қосымшасында ашылады.

Бұл жерде де шиеленіс туындауы мүмкін, себебі көптеген программалар, мысалы, `.mpg` файлдарын қолдағылары келеді. Кәсіби программа, әдетте, орнату кезінде қолданылатын MIME типтерді және кеңейтілімдерді таңдауды ұсынатын жалаушаларды шығарады. Сөйтіп, тұтынушы өзіне қажетін таңдай-

ды және қолданыстағы қауымдастықты кездейсоқ өшіруден сақтайды. Көпшілік тұтынушыға бағытталған программалар көбіне, тұтынушылардың көбі MIME-типтері жайлы білмейді деп болжайды және мүмкін дегеннің барлығын, бұрын орнатылған программаларға қарамастан, жаулап алғысы келеді.

Жаңа файлдар типін қолдаудағы браузер мүмкіндігін кеңейту – бұл ыңғайлы, алайда, кейбір мәселелердің туындауына да әкелуі мүмкін. Windows орнатылған машинадағы браузер .exe кеңейтілімі бар файлды алған кезде, бұл орындалатын файл және оған ешқандай көмекші құрал керек емес деп шешеді. Демек программаны жай іске қосу керек. Алайда, бұл көзқарас ақпаратты қорғау жүйесіндегі күрделі қателік болуы мүмкін. Зиянкестерге тек танымал адамдар, мысалы, киноактерлер немесе спортшылар, суреттерінен тұратын сайт құрастырып, вирусқа сілтеме қою керек. Бұл жағдайда суретке тышқан қолтетігі көрсеткішімен бір шерту, тұтынушы машинасында әрекет ететін, болжамсыз, мүмкін өте қатерлі программаның іске қосылуына әкелуі мүмкін. Осындай қажетсіз жағдайларды болдырмас үшін Firefox және басқа да программалар белгісіз программалар таңдап іске қосуға бағытталған, алайда, тұтынушылардың барлығы бірдей қауіпсіздікті таңдаудың, ыңғайлылықтан маңызды екенін түсінбейді.

Сервер жақ

Клиент жақ туралы жеткілікті айтылды. Енді сервер жақ жайлы айталық. Біз білетіндей, тұтынушы URL енгізген кезде немесе гиперсілтемеге шерткен кезде, браузер URL-ға құрылымдық сараптама жасайды және *http://* және келесі қисық сызық арасында жазылған ақпаратты ізделетін DNS аты ретінде қабылдайды. Қажет сервердің IP-адресін алып, браузер осы сервердің 80-портымен TCP-байланыс орнатады. Бұдан кейін URL-дің қалған бөлігін құрайтын, сервердегі параққа нұсқайтын команда жөнелтіледі. Сервер браузерге сұралған файлды бейнелеу үшін қайтарады.

Бір қарағанда, веб-сервер *6.1-листингте* көрсетілген серверге ұқсас. Бұл серверге, нағыз веб-сервер берілгендей, іздеп тауып, желі бойымен жөнелту қажет файл аты беріледі. Екі жағдайда да негізгі айналымда сервер:

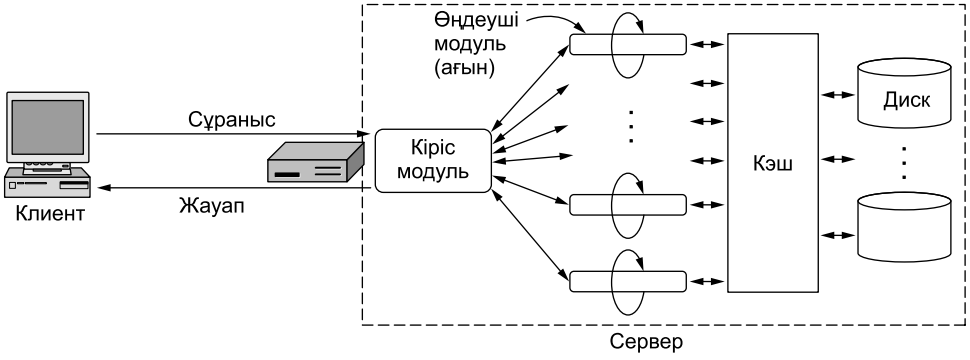
1. Клиенттен (браузерден) келген кіріс TCP-байланысты қабылдайды;
2. Сұралған файлдың аты болып саналатын, параққа баратын жолды алады;
3. Файлды алады (дискіден);
4. Файл мазмұнын клиентке жөнелтеді;
5. TCP-байланысты үзеді.

Қазіргі веб-серверлердің мүмкіндіктері әлдеқайда кең, алайда, олардың жұмысындағы ең маңызды, контентті сұраған кезде орындалатын осы жоғарыда аталған қадамдар. Егер контент динамикалық болса, онда үшінші қадамды контентті қайтаратын программаны іске қосумен (жолмен анықталған) алмастырылуы мүмкін.

Алайда, веб-серверлер бір секунд ішінде көптеген сұраныстарға жауап беру үшін, басқа жолмен жұмыс істейді. Қарапайым жолмен жұмыс істеудің бір мәселесі – файлға қол жеткізудің қиындығы. Файлды дискіден оқу программа жұмысына қарағанда тым баяу жүреді және операциялық жүйе шақырулары на байланысты, бір файл дискіден бірнеше рет оқылуы мүмкін. Тағы бір мәселе, бірмезгілде бірнеше сұраныс өңделуі мүмкін. Файл тым үлкен болуы мүмкін және ол тасымалданып жатқанда басқа сұраныстар оқшауланады.

Мәселені шешудің жолы соңғы сұралған *n* файлды немесе бірнеше гигабайт контент санын жадыда кэштеу. Файлды дискіден сұрамас бұрын, сервер кэш мазмұнын тексереді. Егер сұралған файл кэште бар болса, онда дискіге жүгінбей, оны бірден клиентке жөнелтуге болады. Тиімді кэштеу үшін үлкен жады көлеміне және кэшті тексеруге және оның мазмұнын басқаруға қосымша уақыт қажет болатынына қарамастан, қорытынды уақыттан келетін ұтым үстеме шығын мен бағаны ақтайды.

Сұраныстарды тізбектеп өңдеу мәселесінің бір шешімі – **көп ағындық (multithreaded)** сервер құрастыру. Жүзеге асырылудың бір нұсқасы сервер барлық кіріс сұраныстарды қабылдайтын кіріс модулінен тұрады деп есептейді және *7.9-суретте k* өңдеуші модуль көрсетілген. Барлық *k+1* ағын бір үдеріске тиесілі, сондықтан өңдеуші модульдер үдеріс адрестік кеңістігінде кэшке қолжеткізе алады. Сұраныс келген кезде кіріс модулі оны қабылдап, оның сипаттамасы жайлы қысқаша анықтама жасайды. Сонан кейін жазба өңдеуші модульдердің біріне беріледі.



7.9-сурет. Кіріс және өңдеуші модулі бар көп ағынды веб-сервер

Өңдеуші модуль қажет файлды алдымен кэштен іздейді. Егер ол кэште бар болса, ол файлға көрсеткіш қосып жазбаны жаңартады. Егер ізделген файл жоқ болса, өңдеуші модуль дискіге жүгініп, файлды кэшке жазады (бұл жағдайда, кэштегі кейбір файлдар өшірілуі мүмкін). Дискіден алынған файл кэшке жазылып, клиентке жөнелтіледі.

Бұл схеманың артықшылығы – бір немесе бірнеше өңдеуші модульдер дискілік немесе желілік операциялардың (мұндай модульдер орталық про-

цессор қуатылығын шығындамайды) аяқталуын күтіп, оқшауланып тұрған кезде, басқа модульдер басқа сұраныстарды өңдей береді. *k* өңдеуші модуль бар болғанда, өнімділікті бір ағындық сервермен салыстырғанда *k* рет өсіруге болады. Әрине, дискі немесе желі шектеуші фактор болған кезде, бір ағындық модельді жақсарту үшін бірнеше дискі немесе әлдеқайда жылдам желінің болғаны дұрыс.

Қазіргі заманға веб-серверлердің жай аттар жолын қабылдап, файлды жөнелтуден басқа да функциясы өте көп. Іс жүзінде нақты сұранысты өңдеу әлдеқайда күрделі болуы мүмкін. Осы себептен көптеген серверлерде әр өңдеуші модуль әрекеттер тізбегін орындайды. Кіріс модулі әр кіріс сұранысын бірінші қолжетімді модульге береді. Модуль сұранысқа не қажеттігіне байланысты, оны төменде көрсетілген қадамдар жиынтығын орындап өңдейді. Бұл қадамдар TCP-байланыс және қандайда бір қорғалған транспорттық механизм (*8-мақалада* сипатталатын SSL/TLS тәрізді) орнатылығаннан кейін орындалады.

1. Сұралған веб-парақ атын есептеу.
2. Веб-парақ үшін қолжеткізуді бақылауды іске асыру.
3. Кэшті тексеру.
4. Сұралған парақты дискіден оқу немесе оны құрастыратын программаны іске қосу.
5. Жауаптың қалған бөлігін анықтау (мысалы, MIME типі).
6. Жауапты клиентке қайтару.
7. Сервер активтілігі журналыны жазба қосу.

Бірінші қадам қажет, себебі кіріс сұраныста файлдың нақты аты немесе программа жолдық литерал түрінде болмауы мүмкін. Онда тасымалдауға қажет кіріктірілген шартты белгілер болуы мүмкін. Мысалы, URL: *http://www.cs.vu.nl/* болуы мүмкін. Мұнда файл аты жоқ. Бұл URL-ды үнсіз келісім бойынша қабылданатын файл атымен толықтыру қажет. Әдетте, бұл – *index.html*. Келесі бір жалпы ереже – *~user/* тұтынушы веб-каталогы. Бұл ережелер қатар пайдаланылуы мүмкін. Сөйтіп, файлдың нақты аты белгілі бір каталогтағы *index.html* болғанына қарамастан, авторлардың бірінің үй парағы (AST):

http://www.cs.vu.nl/~ast/

адрес бойынша қолжетімді болады. Сонымен бірге, қазіргі заман браузерлері тұтынушы тілі және браузердің үнсіз келісім бойынша ПЖ тәрізді конфигурация жайлы ақпаратты көрсете алады (мысалы, итальян немесе ағылшын тілі). Бұл серверге сәйкес тілдегі, сонымен бірге, егер мобильді құрылғы болса кішкене суреттері бар веб-парақты таңдауға мүмкіндік береді. Файлдық каталогқа және программаларға жолды қалай бейнелеу керектігі жайлы көптеген келісімдер бар болғандықтан, ат кеңейтілімі – бұл жай бос сөз емес.

Екінші қадам, параққа байланысты қолжеткізуге шектеудің бар екендігін тексереді. Парақтардың барлығы бірдей көпшілікке қолжетімді емес. Клиенттің парақты қарауға құқығы бар екендігін анықтау клиенттің өзіне (мысалы,

берілген тұтынушы аты және құпия сөзге байланысты) немесе клиенттің DNS және IP кеңістігінде орналасуына байланысты. Мысалы, парақты қарауға тек қандайда бір компания қызметкелерінің ғана құқығы болуы мүмкін. Ал бұның жүзеге асырылуы сервердің құрылымына байланысты. Мәселен, танымал Apache серверінде қолжеткізуі шектеулі файлдар орналасқан каталогта шектеулер тізімі көрсетілген *.htaccess* файлы бар.

Үшінші және төртінші қадам парақты алу. Өңдеу ережесі оны кэшттен алуға болатындығын анықтайды. Мысалы, жұмыс істеуші программалар құрастырған файлдар үнемі кэште сақталмайды, себебі әр іске қосылған сайын олардың нәтижесі әртүрлі болуы мүмкін. Уақыт өткен сайын файлдардың өзінің де мазмұнының өзгергендігін тексеру қажет (ескі версияларды өшіріп тастауға болады). Егер парақ программаның іске қосылғанын қажет етсе, онда программа параметрлерін немесе бастапқы мәндерін анықтаумен мәселе туындауы мүмкін. Бұл мәндер жолдан немесе сұраныстың басқа бөлігінен алынады.

Бесінші қадамда парақ мазмұнына байланысты, жауаптың басқа бөліктері анықталады. Мысалы, MIME-типі. Ол файл кеңейтілімінен, конфигурация файлынан және басқа да ақпарат көзінен шығуы мүмкін.

Алтыншы қадам парақты желіге қайтарады. Өнімділікті жоғарылату үшін, клиент және сервер бір TCP-байланысты әртүрлі парақты бірнеше рет алу үшін қолданылуы мүмкін. Мұндай қолдану түрі бірлесіп қолданылатын байланыста сұранысты бейнеленудің қандай да бір логикасын құрастыруды және қайтарылатын жауаптың қажет сұраныспен сәйкестендірілуін талап етеді.

Жетінші қадам әкімшілік мақсатта жүйелік журналда жазба құрастырып, басқа маңызды статистиканы сақтайды. Мұндай журналдар тұтынушы тәртібі жайлы пайдалы ақпараттың бар екендігіне сарапталуы мүмкін, мысалы, адамдардың сайттағы параққа қандай ретпен кіретіндігі.

Cookie файлы

Желіні жоғарыда сипатталғандай қолдануға парақтардың тәуелсіз сұраныстары кіреді. Байланыс сеансы ұғымы жоқ. Клиент браузері серверге сұраныс жөнелтеді және жауап ретінде файлды алады. Бұдан кейін сервер бұл клиентті біркезде көргенін ұмытып кетеді.

Бұл модель көпшілікке арналған құжаттарды алуға ыңғайлы және ол Дүниежүзілік өрмек жаңа құрастырылған кезде жақсы жұмыс істеген. Алайда, сервердің өзгергендігіне байланысты, бұл нұсқа әртүрлі парақтарды түрлі тұтынушыларға қайтару үшін жарамайды. Бұл жұмыс тәртібі веб-сайтпен көпшілік үздіксіз әрекеттесу үшін қажет. Мысалы, кейбір сайттарда (мысалы, газеттер сайтында) тұтынушының тіркелуін талап етеді, ал кейде сайттан ақпарат алу ақылы болып келеді. Тіркелген тұтынушылардан келген сұраныс және басқа барлық тұтынушылардан келген сұранысты ажырату мәселесі туындайды. Екінші мысал, электронды коммерция. Сервер тұтынушы қоржыны жайлы ақпаратты қалай жинауы керек, егер ол қоржынды дүркін-дүркін

толтырып отырса? Үшінші мысал – Yahoo тәрізді веб-портал! Тұтынушыға бастапқы парақтың түрін өз қалауынша жеке баптауға рұқсат беріледі (мысалы, оған бірден сүйікті спорт командасы жайлы жаңалықтар немесе бағалы қағаздар курсын көрсететіндей етіп). Сервер қандай тұтынушымен жұмыс істеп отырғанын білмесе, қалайша парақты дұрыс бейнелей алады?

Алғашқыда шешім анық болып көрінер: сервер тұтынушыны IP-адресі бойынша тани алады. Алайда, бұл идея жұмыс істемейді. Біріншіден, көптеген тұтынушылар ресурстары бөлінген компьютерлерде жұмыс істейді (мысалы, үйде) және де IP-адрес бойынша тек компьютерді сәйкестендіруге болады, тұтынушыны емес. Одан бетер, көптеген компаниялар NAT пайдаланады, сондықтан барлық клиенттердің шығыс дестерлерінің IP-адресі бірдей. Яғни, NAT арқылы жұмыс істейтін компьютерлер серверге бірдей болып көрінеді. Оның үстіне көптеген ISP-лар тұтынушыларға IP-адресті DHCP арқылы тағайындайды. Уақыт өте IP-адрестер өзгеруі мүмкін, сондықтан сервер үшін сіз көршіңіздей болып көрінуіңіз мүмкін. Осы аталған себептермен, сервер тұтынушыны қадағалау үшін IP-адресті қолдана алмайды.

Бұл мәселені шешу үшін, қатты сынға алынған **cookie-файлдар** ұсынылған болатын. Бұл ат ескі программистік сленгілік сөзден келген болатын. Программа процедураны шақырып, оның орнына кейіннен қандайда бір жұмысты орындағанда қажет болатын затты алып отырған. Міне, сол «зат» cookie деп аталатын. Бұл орайда UNIX файлдар дискрипторын және Windows объектілер дискрипторын cookie ретінде қарастыруға болады. Бұл арнайы маркерлер алғаш рет 1994 жылы Netscape браузерінде пайдаланылған болатын. Қазір олар RFC 2109-да біріктірілген.

Тұтынушы парақты сұраған кезде сервер өз жауабын cookie-файл тәрізді қосымша ақпаратпен жасақтандырды, бұл кішкене (4 Кб-қа дейін) аталған жол, север оны браузермен сәйкестендіре алады. Бұл сәйкестендіру тұтынушыны нақты анықтауға мүмкіндік бермейді, алайда, ол IP-адреске қарағанда дәлірек болып келеді. Браузерлер алынған маркерлерді біраз уақыт, клиент қатты дискідегі cookie каталогында сақтайды, сондықтан cookie-файл браузер қайта шақырғанда сақталады, тек тұтынушы бұл функцияны өшіріп тастамаса. Шындап келгенде, бұл файлда вирус болуы мүмкін, алайда, олар тек ақпараттық дерек ретінде қарастырылатындықтан, вирустың активтелуіне және жүйеге нұқсан келтіруіне ресми мүмкіндік жоқ. Алайда, хакердің браузер қателіктерін пайдаланып, cookie-дегі вирусты активтендіру мүмкіндігі бар.

Cookie маркерінде *7.10-кестесінде* көрсетілгендей бес өріс бар. *Домен (Domain*) маркер келген домен аты. Браузерлер сервердің домен аттарын дұрыс айтқанын тексереді деп болжанады. Әр доменде, бір клиентпен байланысты 20-дан аспайтын маркер сақталуы тиіс. *Жол (Path)* өрісі сервердегі каталогтар құрылымына жолды, соның ішінде каталогтар ағашының маркер пайдалана алатын бөлігін көрсетеді. Бұл өрісте жиі, ағаштың толығымен қолжетімді екенін білдіретін «/» белгісі болады.

Мазмұн (Content) өрісі *аты=мән* түрінде болады. Ат та, мән де сервердің қалауынша еркін болуы мүмкін. Бұл өрісте маркердегі негізгі ақпарат орналасады.

Жарамдық (Expires) өрісі маркердің жарамдылық уақытын көрсетеді. Егер бұл өріс жоқ болса, сервер cookie-ді программадан шығысымен лақтырып тастайды. Мұндай маркер **тұрақсыз (nonpersistent cookie)** деп аталады. Егер дата және уақыт көрсетілсе, ол **тұрақты (persistent cookie)** деп аталады. Ол жарамдылық уақыты өткенше сақталады. Жарамдық өрісінде көрсетілген уақыт – гринвичтік. Cookie-ді клиент қатты дискіден жою үшін, сервер жарамдық уақыты өткенін көрсетіп қайта жөнелтеді.

7.10-кесте. Бірнеше cookie мысалы

Домен	Жол	Мазмұн	Жарамдық	Қорғалған
toms-casino.com	/	customerID=297793521	15-10-10 17:00	Иә
jills-store.com	/	Cart=1-00501;1-07031;2-13721	11-1-11 14:22	Жоқ
aportal.com	/	Prefs=Stk;CSCO+ORCL;Spt:Jets	31-12-20 23:59	Жоқ
sneaky.com	/	UserID=4627239101	31-12-19 23:59	Жоқ

Соңында, *Қорғалған (Secure)* өрісінде браузер оны тек ақпаратты тасымалдаудың қорғалған арнасын нақты SSL/TLS (біз оны *8-тарауда* қарастырамыз) пайдаланатын серверге ғана қайтару керектік индексі орнатылады. Бұл қасиет электрондық коммерцияда, банк ісінде және басқа да ақпаратты қорғау маңызды қосымшаларда пайдаланылады.

Сонымен, біз маркерді алу жайлы білдік. Қалай қолданылады? Браузер қандай да бір веб-сайтқа парақты алуға сұраныс жіберер алдында cookie каталогын тексереді. Сұраныс жіберілетін доменнен (тек сол доменнен) келген маркерлер ізделінеді. Барлық табылған маркерлер сұраныспен бірге жөнелтіледі. Сервер оны алып, өз қажеттілігіне байланысты интерпретациялайды.

Cookie-ні пайдалану мүмкіндіктерін қарастырайық. *7.10-кестедегі* бірінші cookie-ні *toms-casino.com* домені жөнелткен, ол клиентті сәйкестендіру үшін пайдаланылады. Егер бір аптадан кейін сол клиент, кезекті ақша сомасын шашуға сайтқа қайтып оралса, браузер cookie жөнелтеді, сондықтан сервер ескі танысын бірден таниды. Тұтынушы идентификаторын алып сервер ол жайлы жазбаны деректер базасынан таба алады және осы ақпаратты сәйкес веб-парақты генерациялау үшін пайдаланады. Ойыншының құмарлығына байланысты оған покер, бүгінгі жарыс кестесі немесе жай ойын автоматы ұсынылады.

Екінші cookie-маркер *jills-store.com* келген. Бұл жағдайдағы сценарий бойынша клиент, өзіне зат таңдап, дүкен қыдырып жүр. Ұнаған зат табылса, тауардың шартты белгісіне шерттеді. Сервер тауарды клиенттің сатып алған тауарлар тізіміне қосады (тізімді сервер қолдап отырады) және тапсырыс берген

тауар кодынан тұратын cookie-маркер құрастырып, оны клиентке жөнелтеді. Клиент электронды дүкенде қыдыруын жалғастырады және әр парақта жаңа тауарға шертеді, әр жаңа парақ сұранысына жауап ретінде серверге cookie жөнелтіледі. Таңдап алған тауарлар жиналған сайын cookie ақпараты толықтырылады. Соңында, тұтынушы «ЕСЕП АЙЫРЫСУҒА КӨШУ» батырмасына шерткен кезде, сатып алулар жайлы толық ақпарат жазылған cookie, сұраныспен бірге серверге жөнелтіледі. Сөйтіп, клиенттің қандай тауарлар сатып алғысы келетіні серверге белгілі болады.

Үшінші cookie-маркер веб-порталдан келді. Тұтынушы портал сілтемесіне шерткен кезде браузер оған Cisco және Oracle акциялары бағасының белгілеулері, сонымен бірге New York Jets футбол матчның нәтижесі көрсетілген парақты жөнелту керектігі көрсетілген cookie жібереді. Cookie-файлдың ең үлкен мөлшері 4 Кб болғандықтан, парақты егжей-тегжейлі баптауға жеткілікті орын қалады. Мысалы, сіз оған ауа райы мәліметін, арнайы ұсыныстар, ірі газеттер мақалаларының тақырыптарын және т.б. қосуыңызға болады.

Cookie-файлды пайдаланудағы ең даулы мәселе – тұтынушы әрекеті. Веб-сайт операторлары тұтынушылар олардың парақтары бойымен қалай қозғалатынын біледі және жарнамашылар жеке тұтынушының қараған сайт профилін және жарнаманы құрастырады. Мәселе тұтынушының өз әрекетін қадағалап отырғанын білмейтіндігінде, тіпті, өзара-байланыспаған егжей-тегжейлі профильде де, веб-сайтта да. Дегенмен, **веб-қадағалау (Web-tracking)** маңызды бизнес. Веб-маниторингпен айналысатын Alexa компаниясының анықтамасы бойынша жарнамалық баннерлерді ұсынып, қадағалау жүргізетін DoubleClick әлемнің 100 ірі веб-сайттарының бірі ретінде танылған. Операторлар үшін сайттың пайдаланылуын қадағалайтын Google Analitics-ті ірі 100 000 сайттардың жартысынан көбі пайдаланады. Сервер cookie файлдарының көмегімен тұтынушы активтігін қадағалайды. Олардың көмегімен сайтқа кірушілер санын, әрқайсысының неше парақ қарағанын білуге және осы деректер бойынша статистика құрастыруға болады. Серверге тұтынушыдан бірінші рет сұраныс келген кезде, әрине, онымен бірге ешқандай маркер келмейді. Сондықттан сервер санауыш мәні 1-ге тең cookie-ні кері жөнелтеді. Ал тұтынушының парақтарды келесі қарауында әрине, cookie бірге жөнелтіліп отырады. Санауыш әр кез инкременттеліп, тұтынушыға жөнелтіледі. Сөйтіп, санауыш бойынша неше тұтынушының бірінші парақты қарап, сайттан шығып кеткенін, нешеуінің екі парақ қарағанын және т.с.с. біліп отыруға болады.

Тұтынушының сайт бойымен жылжуын қадағалау қиынырақ мәселе. Бұл келесідей жүзеге асырылады. Жарнамалық агенттік, мысалы, Қара Жарнамашы ірі веб-сайттармен байланысып, олардың сайттарында өз клиенттерінің тауар баннерлерін орналастырады және ол үшін сайтқа ақша төленеді. Сайттың әр бетінде орналастыруға болатын GIF түріндегі жарнама баннер орнына, оған URL беріледі және ол сайттың әр парағына орналастырылады. Осы URL-дің әрқайсының жол түріндегі бірегей идентификаторы бар, мысалы:

http://www.sneaky.com/382674902342.gif

Тұтынушы осындай жарнама орналасқан *P* парағына алғаш рет кірген кезде, браузер әдеттегідей, HTML-файлды қабылдайды. Браузер оны қарап отырып, *www.sneaky.com* суретіне сілтемені көреді. Әрине ол суретті алуға сұраныс жөнелтеді. GIF-пен бірге тұтынушының бірегей идентификаторы 4627239101 (*7.10-кестені* қараңыз) бар cookie келеді. Қара Жарнамашы осылайша, осындай бірегей идентификаторы бар тұтынушының *P* парағына кіргенін белгілейді. Бұл өте қарапайым жүзеге асырылады, себебі сұралған файлға сілтеме (*382674902342.gif*) тек P парағында ғана бар. Әрине бір жарнама мыңдаған әртүрлі парақтарда орналасуы мүмкін, бірақ олардың әрқайсысының өз аты бар. Жарнаманың әр данасын жеткізу үшін жарнама компаниясы тапсырыс берушіден белгілі бір сомадағы ақы алуы мүмкін.

Сонан кейін, тұтынушы Қара Жарнамашы баннері орналасқан параққа кірген кезде, баузер серверден HTML-файлды алып, айталық, *http://www.sneaky.com/193654919923.gif* атты суретке сілтемені көреді де, сол файлға сұраныс жібереді. Бұның алдында *sneaky.com* доменінен cookie алынғандықтан, браузер оны тұтынушы идентификаторымен кері жөнелтеді. Сөйтіп, Қара Жарнамашы (ҚЖ) өз жарнамасы орналасқан екінші параққа тұтынушының кіргенін біледі.

Уақыт өте ҚЖ тұтынушы құмарлығының толық тізімін жасайды және бұл кезде оның баннерге шертуі қажет емес те. Әрине, тұтынушы аты белгісіз болып қалады (дегенмен, IP-адрес бар және деректер базасы бойынша оның атын анықтауға болады). Тұтынушы ҚЖ бірігіп қызмет істейтін сайттардың бірінде өз атын енгізсе болғаны оған толық веб-құжаттама жасап, сатуға болады. Осындай құжаттамаларды сатудың пайдалы болғандығы соншалықты, ҚЖ мүмкіндігінше көптеген сайттармен қызметтес болып, неғұрлым көп ақпарат жинауға тырысады.

Егер ҚЖ Суперқара Мега жарнамашы болғысы келсе, оның хабарландырулары әдеттегі классикалық баннер сілтемесіндей болмауы керек. Көлемі бір пиксел, парақты артқы фон түсімен бірігіп кететін (яғни көрінбей қалады) «Хабарландыру» да тұтынушыны қадағалау кезінде дәл осындай нәтиже береді: браузер көлемі 1x1 gif-суретке сұраныс береді және cookie кері жөнелтеді.

Жоғарыда сипатталған тұтынушы қозғалысын қадағалауға байланысты cookie желідегі құпиялық жайлы пікірталаста ең басты мәселе болды. Бұл бизнестің ең сорақысы, көптеген тұтынушылар өзі жайлы қандайда бір ақпарат жинақталып жатқандығы жайлы мүлдем білмейді және бұндай әрекеттен өзін қорғалғанмын деп есептейді, себебі ол ешқандай баннерге шертпейді де. Сондықтан сайттағы тұтынушы қозғалысын қадағалайтын cookie-ні көбі **тыңшы-программа (spyware)** деп қабылдайды. Өз браузеріңізде сақталған cookie файлды қарап шығыңыз. Көптеген браузерлер бұл ақпаратты ағымдағы құпиялық баптауымен бірге көрсетеді. Түсініксіз идентификаторлармен бірге сіз ол жерден аттар, адрестер және құпия сөздерді көре аласыз. Ол жерде сіздің кредиттік картаңыздың нөмірі жоқ деп сенеміз, бірақ ақпараттың шамадан тыс асыра пайдаланылатыны көрініп тұр.

Кейбір тұтынушылар өз-өздерін жұбату үшін браузерлерін кез келген cookie қабылдамайтындай етіп баптайды. Алайда, бұл жаңа мәселе туындатуы мүмкін, себебі көптеген веб-сайттар тұтынушымен cookie-маркермен алмаспаса дұрыс жұмыс істей алмайды. Бұдан басқа көптеген браузерлер тұтынушыға **«үшінші жақ» cookie (third-party cookies)** оқшаулауға мүмкіндік береді. Демек, бастапқы парағында отырған сайттан емес, мүлде басқа веб-сайттан келетін, мысалы, *sneaky.com* Р парағымен әрекеттескенде алынатын cookie. Бұл cookie оқшауланған сайттар арасында көшуді қадағалауды болдырмайды. Сонымен бірге, cookie қалай падаланылатынын (немесе пайдаланылмайтынын) дұрыс бақылау үшін, браузер кеңейтілімін орнатуға болады. Cookie жайлы пікірталас жалғасып жатқан кезде, көптеген компаниялар құпиялық саясат нұсқаларын құрастыруда, яғни асыра пайдалану болмас үшін ақпаратпен қалай бөлісетіндіктерін ойластыруда. Әрине, бұндай саясат тек компанияның алынған ақпартты қалай пайдаланатынын анықтайды. Мысалы, «Біз сіз жайлы жинаған ақпаратты өз қалауымызша пайдалана аламыз» деген сөз, компания оны сата алады дегенді білдіреді.

7.3.2. Статикалық веб-парақтар

Дүниежүзілік өрмектің негізгі идеясы – веб-парақтың серверден клиентке қозғалуы. Қарапайым веб-парақтар статикалық болып келеді, яғни, қандай да бір файл-серверде орналасқан, әр қараған кезде бірдей болып көрінеді. Алайда, парақтың статикалық болуы ол браузерде бейнеленгенде онымен еш әрекет орындалмайды деген емес. Бейне файлы бар парақ статикалық бола алады.

Жоғарыда айтылғандай, HTML Дүниежүзілік өрмектің ана тілі, көптеген веб-парақтар осы тілде жазылған. Ұстаздар веб-парағы әдетте статикалық HTML-парақ, компаниялар парағы әдетте, веб-парақ дизайнымен айналысатын компания құрастырған динамикалық парақ болып келеді. Бұл бөлімде біз статикалық HTML-парақтар жайлы қысқаша айтып кетеміз, себебі олар әрі қарай айтылатын ақпараттың негізі болып келеді. HTML-мен таныс оқырмандар бірден келесі бөлімге, динамикалық контент және веб-қызметке көшулеріне болады.

HTML – веб-парақты белгілеу тілі

HTML (HyperText Markup Language) Дүниежүзілік өрмекпен қатар пайда болды. HTML көмегімен веб-парақтарға мәтін, графика, бейне, сонымен бірге, басқа парақтарға көрсеткіштер және т.б. орналастыруға болады. Ол белгілу тілі, яғни құжатты форматтауды сипаттайтын тіл болып келеді. «Белгілеу» (markup) термині, ертеде көркемдік редактор арнайы белгілер көмегімен типографқа (сол кезде осындай мамандық болған) құжатты басып шығару үшін қандай қаріпті пайдалану керектігін көрсететін кезден бас алады. Сөйтіп, белгілеу тілінде форматтаудың егжей-тегжейлі командалары жазыла-

ды. Мысалы, HTML тілінде командасы **мәтін аумақ басын жартылай қалың қаріппен басу керек** дегенді білдіреді, ал осындай аумақтың соңы дегенді білдіреді. Белгілеу тілінің басқа мысалдары, LaTeX және TeX академиялық авторларға жақсы таныс.

Анық форматтау тілдерінен белгілеу тілінің негізгі артықшылығы, ол контенттің қалай бейнеленетіндігін ерекшелейді. Сөйтіп, парақ жазу өте қарапайым: браузер мәтіндегі белгілеу командаларын түсініп, контентке қолдануы керек. HTML-файлдағы кіріктірілген стандартты белгілеу командалары арқылы веб-браузермен кез келген веб-парақты оқып, қайта форматтауға болады. Форматтауды өзгерту мүмкіндігі өте маңызды, себебі бір нүктеге 24 бит бөлінетін бейнелеу мүмкіндігі 1600×1200 нүкте экран үшін жасалған веб-парақты, бір нүктеге 8 бит, бейнелеу мүмкіндігі 640×320 нүкте мобильді телефон экранында қарау мүмкіндігі болу керек.

Мұндай құжаттарды стандартты мәтіндік редактор көмегімен құрастыру мүмкін болса да, көбі осылай жасайды да, сонымен бірге, жұмыстың көп бөлігін өзіне алатын (нәтижені тұтынушының егжей-тегжейлі бақылауын төмендету арқасында), мәтінді редакциялайтын арнайы редакторларды (мәтіндік процессорлар) және HTML пайдалану мүмкіндігі бар.

HTML-де жазылған қарапайым веб-парақ және оның браузерде қалай бейнеленетіні *7.10-суретте* көрсетілген. Парақ тақырыптан және парақ денесінен тұрады. Бүкіл веб-парақ HTML тілінде **тэг (tags)** деп аталатын <html> және </html>, форматтау командаларының арасында орналасқан. Көптеген браузерлер бұл команда жоқ болса да парақты дұрыс бейнелейді. *7.10 а-суретінде* көрсетілгендей, веб-парақ тақырыбы <head> және </head> тэгтерінің жақшасына алынған, ал парақ денесі <body> және </body> тэгтерінің арасында орналасқан. Тэгтер ішіндегі командалар **директива (directives)** деп аталады. Барлығы болмаса да көптеген HTML-тэгтердің форматы осындай, яғни <something> бір нәрсенің басын, ал </something> соңын белгілейді.

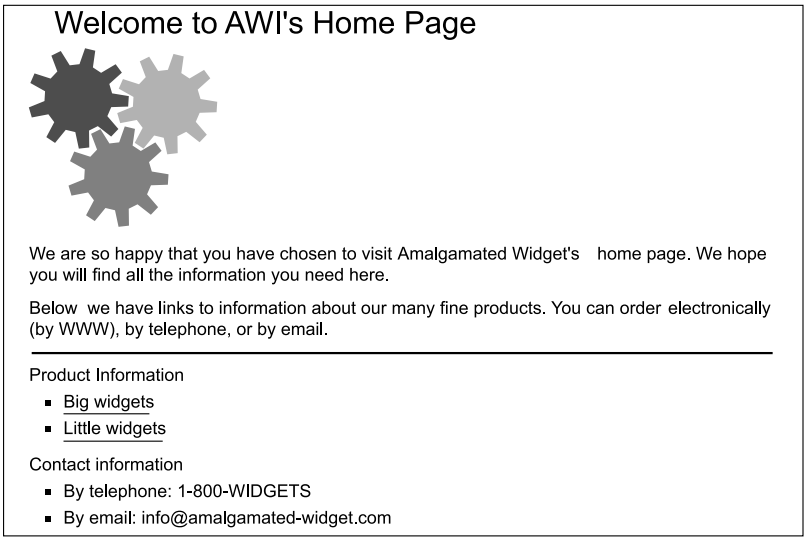
Тэгтердегі символ регистры маңызды емес. Мысалы, <head> және <HEAD> бірдей, алайда, төменгі регистор үйлесімдік сипаттамасы жақсырақ. HTML-мәтін форматы, жолдардың орналасуы маңызды емес. HTML-мәтінді өңдеу программасы артық бос орындарды және жаңа жолға көшулерді есепке алмайды, себебі ол мәтінді берілген бейнелеу аумағына сиятындай етіп форматтайды. Сәйкесінше, бастапқы HTML-құжат жеңіл оқылу үшін оған кез келген сандағы табуляция белгісін, бос орын және жаңа жолға көшу символын қосуға болады. Және керісінше, мәтіндегі абзацты бөлу үшін бастапқы HTML-мәтінге бос жол қосу жеткіліксіз, браузер оны есепке алмайды. Бұл жағдайда нақты тэгті пайдалану керек.

Кейбір тэгтердің **атрибуттар (attributes)** деп аталатын аталған параметрлері болуы мүмкін. Мысалы, *7.10-суреттегі* <img> тэгі парақта мәтінмен бірге суретті орналастыру үшін пайдаланылады. Бұл тэгтің екі атрибуты бар: src және alt. Біріншісі сурет URL-ін ұсынады. HTML-стандарт қандай

форматтағы суреттердің рұқсат етілгендігін анықтайды. Іс жүзіндей барлық браузерлер GIF және JPEG файлдарын қолдайды. Браузерлер басқа да форматты қолдауы мүмкін, бірақ бұл таяқтың екі басы тәрізді. Егер тұтынушы TIFF форматындағы суретті қолданатын браузерге үйренген болса, ол өз парағында осындай суреттерді орналастырып, басқа браузерлердің оны есепке алмағанын көріп таңғалуы мүмкін.

```
<html>
<head>
<title> AMALGAMATED WIDGET, INC. </title>
</head>
<body> <h1> Welcome to AWI›s Home Page </h1>
<img src=»http://www.widget.com/images/logo.gif» ALT=»AWI Logo»> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widgets</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
<li> <a href=»http://widget.com/products/big»> Big widgets </a> </li>
<li> <a href=»http://widget.com/products/little»> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
<li> By telephone: 1-800-WIDGETS </li>
<li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

а)



(ә)

7.10-сурет. веб-парақ мысалының HTML-коды (а); Форматталған парақ (ә)

Екінші атрибут көмегімен сурет қандайда бір себеппен бейнеленбей қалған жағдайда шығатын мәтінді анықтауға болады. HTML стандарты әр тәг үшін мүмкін деген атрибуттар тізімін және олардың мәнін анықтайды. Барлық аттрибуттардың жеке аты болғандықтан олардың реті маңызды емес.

Формалды түрде, HTML құжатты жазған кезде ISO 8859-1 халықаралық стандартының Latin-1 символдар жиынтығын қолдану керек, бірақ пернетақталары тек ASCII-символды қолдайтын тұтынушылар арнайы символдарды, мысалы, è символын енгізу үшін арнайы басқару символдар тізбегін қолдануға болады. Бұл тізбектер амперсанд белгісінен басталып, нүктелі үтір белгісімен аяқталуы керек. Мысалы, ` ` - бос орын, `&egrave;` - è символын, ал `&eacute;` - é символын білдіреді. <, > және & символдары қордағы символдар болғандықтан оларды мәтінде бейнелеу үшін басқарушы тізбек: `<` (less than – «кіші» белгісі), `>` (greater than «үлкен» белгісі), `&` (ampersand - амперсанд) қолданылады.

Тақырыптың негізгі элементі, парақ аты `<title>` және `</title>` тәгтерінің арасында орналасады. Оның ішіне кішігірім мета ақпарат орналастыруға болады, біздің мысалымызда ол жоқ. Ол парақ бетінде бейнеленбейді, бірақ кейбір браузерлер оны парақ терезесін белгілеу үшін пайдаланады.

7.10-суретте бірнеше тақырып қолданылған. Олардың әрқайсысы `<hn>` түріндегі тәг көмегімен жазылады, мұндағы, n – 1-ден 6-ға дейінгі сан. `<h1>` ең маңызды тақырып, `<h6>` - маңыздылығы ең төмен тақырып. Оның экранда бейнеленуі браузерге байланысты. Әдетте, нөмірі ең кіші тақырыптар үлкен қаріппен бейнеленеді. Сонымен бірге браузер әртүрлі тақырыпты әртүрлі түспен бейнелей алады. Әдетте, `<h1>` тақырыбы экранда ірі, жартылай қалың қаріппен бейнеленеді және тақырыптың алдында және соңында бір жолдан қалдырып ерекшеленеді. Ал `<h2>` тақырыбы одан кішірек қаріппен бейнеленеді, сонымен бірге тақырып алдындағы және одан кейінгі бос жол интервалы да кішірейеді.

`<b>` және `<i>` тәгтері сәйкесінше жартылай қалың (boldface) және курсив (italics) қаріпін білдіреді. `<hr>` тәгін пайдаланғанда парақта үзілу және көлденең сызық бейнеленеді.

Абзац басы `<p>` тәгімен бейнеленеді. Браузер оны бос жол және азат жол қосып бейнелеуі мүмкін. Көптеген жалқау HTML-программистердің `</p>` абзац соңы тәгін пайдаланбайтынын айта кету керек.

HTML тілі тізім және кіріктірілген тізім құрастырудың бірнеше механизмін ұсынады. *7.10-суреттегідей* реттелмеген тізімдер (unordered list) `<ul>` тәгімен басталады. Тізімнің жеке пункттері бұрын `<li>` тәгімен белгіленетін. `<ol>` тәгі реттелген тізімге (ordered list) сәйкес келеді. Оны пайдаланған кезде абзацтар және реттелмеген тізімнің жеке пункттері «•» белгісімен белгіленеді. Реттелген тізім пункттерін браузер автоматты түрде нөмірлейді.

Соңында, біз гиперсілтемеге келдік. Оның қолданылу мысалын сіз *7.10-суреттен* көре аласыз. Гиперсілтеме `<a>` (anchor -бетбелгі) және `</a>` тәгтерінің көмегімен енгізіледі. Бұл тәгтің әртүрлі параметрлері болуы мүмкін,

олардың ішіндегі ең маңыздысы `href` (гиперсілтеме, URL). `<a>` және `</a>` тәгтерінің арасында орналасқан мәтін экранда бейнеленеді. Егер бұл мәтін таңдап алынса, онда браузер гиперсілтеме нұсқап тұрған парақты ашады. Сонымен бірге, `<a>` және `</a>` тәгтерінің арасында басқа элементтерде, мысалы, сурет (img тәгі) орналасуы мүмкін. Бұл жағдайда егер тұтынушы суретке шертсе, сілтеме бойынша көшу орындалады.

HTML-дың біз қарапайым мысалда келтірмеген бұдан да басқа көптеген тәгтері және атрибуттары бар. Мәселен, `<a>` тәгінің мәтін ортасында сілтеме жасауға мүмкіндік беретін `name` параметрі бар. Оған осы парақтың басқа жерінен сілтеме жасауға болады. Бұл веб-парақ «жергілікті» гиперсілтемелерден тұратын мазмұндамадан басталған кезде өте ыңғайлы. Мазмұндама пунктіне тышқан қолтетігі көрсеткішімен шерту қажет орынға жылдам көшуге мүмкіндік береді. Тағы бір мысал ретінде `<br>` тәгін келтіруге болады. Ол браузерді жаңа жолға көшуе мәжбүрлейді.

Тәгтерді түсінудің ең жақсы жолы – оларды қызметін көру. Оны көру үшін кез келген веб-парақты таңдап алып, өз браузеріңізде HTML түрінде ашып, оның қалай құрастырылғаны көру. Көптеген браузерлердің Парақтың бастапқы кодын көру немесе осы тәріздес меню қатары бар. Осы меню қатарын таңдаған кезде форматталған парақ емес оның бастапқы HTML-мәтіні бейнеленеді.

Біз Дүниежүзілік өрмектің ертедегі даму сатысындағы тәгтер жайлы қысқаша айттық. HTML әлі де даму үстінде. *7.11-кестеде* HTML келесі версияларына қосылған мүмкіндіктері көрсетілген. HTML 1.0 Дүниежүзілік өрмек пайда болғандағы HTML версиясына жатады. Дүниежүзілік өрмек пайда болғаннан кейінгі бірнеше жыл ішінде 2.0, 3.0 және 4.0 версиялары бірінбірі жылдам алмастырып отырды. HTML 4.0-ден кейін HTML 5.0-ге жол ашылу үшін он жылдай уақыт өтті. Осы стандарт браузерлердің әртүрлі контентті өңдеу тәсілдерін біріктіретін негізгі жаңарту болғандықтан, HTML 5.0 әлі де құрастырылу үстінде және ол 2012 жылдан бұрын аяқтала қоймас. Осылай бола тұрса да, негізгі браузерлер HTML 5 мүмкіндіктерін қолдайды.

HTML версияларына қосылған жаңартулар тұтынушылар көргілері келетін мүмкіндіктермен байланысты. Бұл мүмкіндіктерді стандартталмаған жолмен (мысалы, плагиндер көмегімен) пайдалануға тура келеді. Мысалы, алғашқы екі версияда кестелер бар болатын, бірақ ол тек HTML 3.0 қосылған. HTML-кесте бірнеше жолдан, ал, жолдың әрқайсысы бірнеше ұяшықтан тұрады. Ұяшықта әртүрлі деректер болуы мүмкін (мысалы, мәтін, сурет, тіпті, басқа кесте). HTML 3.0 енгізілгенше кесте қажет болған авторлар кесте бейнеленген сурет тәрізді ерекше әдіске жүгінуге тура келді.

HTML 4.0 алғашқы версиялардан бірнеше жаңа қасиеттерімен ерекшеленеді. Онда мүмкіндігі шектеулі адамдарға арналған қолжеткізу тәсілі, объектілер енгізу (параққа тек сурет қана емес басқа объектілерді қосуға мүмкіндік беретін `img` тәгінің жалпылама түрі), сценарийлер жазу тілін (скриптер) қолдау қосылған, бұл динамикалық парақтардың дамуына түрткі болды және т.б.

7.11-кесте. HTML версияларының арасындағы кейбір айырмашылықтар

Объект	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Гиперсілтеме	x	x	x	x	x
Сурет	x	x	x	x	x
Тізім	x	x	x	x	x
Активті карталар және сурет		x	x	x	x
Формалар		x	x	x	x
Математикалық формулалар			x	x	x
Құрал-саймандар панелі			x	x	x
Кестелер			x	x	x
Қолжетімділікті қамтамасыз ету мүмкіндіктері				x	x
Оъектілерді кіріктіру				x	x
Стильдер кестесі				x	x
Скрипттер				x	x
Бейне және аудио					x
Векторлық графика					x
XML-ді бейнелеу					x
Фондық ағындар					x
Браузерде ақпаратты сақтау					x
Сурет салу аумағы					x

HTML 5.0-дің қазіргі кезде Дүниежүзілік өрмекте кеңінен қолданылатын мультимедианы өңдеуге арналған көптеген опциялары бар. Бейне және аудио парақта орналасып, браузермен ойнатыла алады және бұл тұтынушыдан плагин орнатуды талап етпейді. Графикалық объектілердің растрлық форматының орнына (JPEG және GIF тәрізді) браузерде векторлық формат түрінде сурет салуға болады. Сонымен бірге, браузерде фондық ағындарды есептеу және қорға қолжеткізу тәрізді скрипттердің орындалу мүмкіндіктерін қолдау кеңейді. Осы мүмкіндіктердің барлығы, құжаттан бұрын тұтынушы интерфейсі бар әдеттегі қосымшаға ұқсас веб-парақтарды қолдауға мүмкіндік береді. Дүниежүзілік өрмек нақты осы бағытта дамуда.

Ақпарат енгізу және формалар

HTML-дың біз талқыламаған бір маңызды мүмкіндігі бар – ақпарат енгізу. HTML-дың бірінші версиясы тек біржақты байланысты қамтамасыз ететін.

Тұтынушылар ақпарат ұсынушылардан тек парақты ала алатын, ал ақпаратты кері жөнелту қиын болатын. Веб-парақ арқылы тауарға тапсырыс беру, есепке алу картасын толтыру, кілтті сөз бойынша іздеу және басқа көптеген әрекеттер үшін екі жақты трафик қажет екені тез түсінікті болды.

Енгізілген ақпаратты тұтынушыдан серверге жөнелту (браузер арқылы) екі түрдегі қолдауды қажет етеді. Біріншіден, HTML-дың осы бағытта деректер тасымалдауы керек. Оның қалай жүзеге асырылатынын біз келесі бөлімде қарастырамыз, бұл үдерісте *POST* тәсілі қолданылады. Екіншіден, енгізілген ақпаратты жинақтап біріктіретін, тұтынушы интерфейсі элементтерін ұсыну қажет. Осы функцияны атқаратын **форма (forms)** HTML 2.0 қосылды.

Форманың мәтін енгізетін өрістері және таңдау жасауға, сонан кейін жинақталған ақпаратты парақ иесін жөнелтуге арналған батырмалары бар. Форма, *7.4-листингте* көрсетілгендей, HTML-дың басқа бөліктері тәрізді жазылған. Осы HTML-мәтінге сәйкес келетін форма түрі *7.11-суретте* көрсетілген. Форманың статикалық компонент екеніне назар аударыңыздар. Кімнің пайдаланғанына байланыссыз олар түрін өзгертпейді. Біз кейіннен талқылайтын динамикалық контент, программаны жөнелту арқылы ақпарат жинаудың күрделі түрін ұсынады. Программа қызметі браузер ортасына байланысты.

Басқа формалар тәрізді ол `<form>` және `</form>` тәгтерінің арасында орналасады. Бұл тәгтің атрибутында енгізілген дерекпен не істеу керектігі жазылған, біздің жағдайымызда деректерді тағайындалған URL-ге жөнелту үшін *POST* тәсілі пайдаланылады. Бұл тәгтер арасына кірмеген мәтін тек бейнеленеді. Парақ авторы форманың экрандағы түрін бақылау үшін оның ішінде барлық әдеттегі тәгтерді пайдалануға болады (мысалы, `<b>`).

Формада деректер енгізу үшін терезенің үш түрі қолданылады. Олардың әрқайсысы `<input>` тәгін пайдаланады. Бұл тәгтің бейнелу аумағының мөлшерін, ерекшеліктерін анықтау үшін көптеген параметрлері бар. Кең таралған формалар – бұл тұтынушы мәтін енгізетін бос өрістер, жалаушалар және деректерді серверге жөнелтуді инициализациялайтын *«Жөнелту»* батырмасы.

Бірінші типтегі терезе – бұл *«Аты»* сөзінен кейін орналасқан мәтіндік аудан. Терезе мөлшері 46 символ. Бұл өріске тұтынушы өз атын енгізеді деп болжанады. Енгізілген тұтынушы аты, әрі қарай өңдеу үшін `customer` айнымалысында мәтіндік жол ретінде сақталады. Келесі форма терезелерінде тапсырыс беруші адресі: көше, қала, штат және ел сұралады. Бұл өрістер арасында `<p>` тәгі қойылмағандықтан браузер оларды мүмкіндігінше бір жолда бейнелеуге тырысады (жеке параграфтар түрінде емес). Браузер тұрғысынан бұл абзац жеке алты элементтен тұрады – үш жол, үш тереземен аралас. Келесі жолда тұтынушының кредиттік картасының нөмірі және оның жарамдылық мерзімі сұралады. Кредиттік карта нөмірін Интернет арқылы тек сәйкес қауіпсіздік шаралары қолданылған кезде ғана жөнелтуге болады. Бұл мәселе *8-тарауда* толығырақ қаралатын болады.

Кредиттік карта жарамдылық мерзімінен кейін, біз үшін жаңа басқару элементтері орналасқан – ауыстырып қосқыштар. Олар бірнеше нұсқаның біреуін таңдау керек болған кезде қолданылады. Олар белгіленген радиостанцияларға жылдам қол жеткізуге арналған автомагнитолладағы батырмаларды еске салады. Бұл типтегі ауыстырып қосқыштар үнемі бір топқа біріктіріледі. Бір ауыстырып қосқышты таңдау осы топтағы басқа ауыстырып қосқыштарды автоматты түрде өшіреді, ауыстырып қосқыштардың сыртқы түрі қолданыстағы интерфейске тәуелді. «Зат мөлшері» бланк пунктінде де екі ауыстырып қосқыш қолданылады. Ауыстырып қосқыштар тобы `<input>` тәгінің `name` параметрінің мәнімен анықталады. Ауыстырып қосқыштар тобын анықтау үшін `<radiobutton> …</radiobutton>` тәрізді арнайы тәгтер жақшасы қарастырылмаған.

`Value` параметрі тұтынушының қандай батырманы басқандығын анықтайды. Мысалы, тұтынушы есеп айырысу үшін қандай кредиттік картаны таңдайтынына байланысты `cc` айнымалысына «mastercard» немесе «visacard» мәтіндік жолының мәні меншіктеледі.

7.4-листинг. Тапсырыс бланкі үшін HTML-мәтін

```
<html>
<head> <title> AWI КЛИЕНТІНІҢ ТАПСЫРЫС БЛАНКІСІ </title> </head>
<body>
<h1> Затқа тапсырыс беру бланкісі </h1>
<form ACTION=»http://widget.com/cgi-bin/order.cgi” method=POST>
<p> Аты <input name=»customer» size=46> </p>
<p> Адрес <input name=»address» size=40> </p>
<p> Қала <input name=»city» size=20> Штат <input name=»state» size =4>
Ел <input name=»country» size=10> </p>
<p> Кредиттік карта нөмірі <input name=»cardno» size=10>
Жарамдық мерзімі <input name=»expires» size=4>
M/C <input name=»cc» type=radio value=»mastercard»>
VISA <input name=»cc» type=radio value=»visacard»> </p>
<p> Зат мөлшері: Үлкен <input name=»product» type=radio
value=»қымбат»>
Кіші <input name=»product» type=radio value=»арзан»>
Жеткізу <input name=»express» type=checkbox> </p>
<p><input type=submit value=»Тапсырысты жөнелту»> </p>
Сіздің AWI фирмасына осы затқа тапсырыс бергеніңіз үшін рақмет
айтамыз.
Бұл дұрыс таңдау! </form>
</body>
</html>
```

Бланктегі екі ауыстырып қосқыштар тобынан кейін `checkbox` (жалауша) типті басқару элементі орналасқан. Ол алдыңғы ауыстырып қосқыштарға ұқ-

сас, себебі ол да екі қалып-күйдің (орнатылған/алып тасталған) бірінде бола алады, бірақ топқа біріктірілмейді және осы типтес басқа элементтерге тәуелсіз, оған тышқан қолтетігі көрсеткішімен шерту арқылы қосылып, өшіріледі.

Соңында, біз `submit` (растау) типіндегі батырмаға келіп жеттік. Бұл жағдайда `value` параметрінің жолында батырма жазбасы көрсетілген. Тұтынушы `submit` батырмасына басқан кезде браузер барлық жинақталған ақпаратты үлкен бір жолға біріктіріп, серверге, `<form>` тәгінің бөлігі ретінде жазылған URL-ге өңдеуге жөнелтеді. Қарапайым кодтау тәсілі қолданылады. Деректер өрісі & (амперсанд) ажыратылады, ал бос орындар + белгісімен ал мастырылады. Біздің мысалымызда, серверге жөнелтілетін осындай жол төменде *7.5-листингінде* көрсетілгендей болуы мүмкін.

7.11-сурет. Тапсырыс бланкінің форматталған парағы

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&
state=NY&country=USA&cardno=1234567890&expires=6/14&cc=mastercard&
product=cheap&express=on
```

7.5-листинг. Тұтынушы енгізген ақпарат тұратын браузердің серверге жөнелтетін болжамалы жауабы

Бұл мәлімдеме серверге бір мәтіндік жол (сіздер бірнеше жол көріп отырсыздар, бұл тек парақ ені жеткіліксіз болғандықтан) түрінде жөнелтіледі. Алынған ақпаратпен не істеу керектігін сервер өзі шешеді, ол оны осындай деректерді өңдейтін программаға беруі ықтимал. Біз оны кейін талқылаймыз.

Енгізілетін ақпараттың осы қарапайым мысалда келтірілмеген түрлері бар. Мысалы, `password` (құпия сөз) және `textarea` (мәтіндік аумақ) типтері бар. `Password` терезесі `text` (үнсіз келісім бойынша орнатылатын және атауды қажет етпейтін мәтіндік тип) терезесіне ұқсас, алайда, мәтін терген

кезде символдар экранда бейнеленбейді. `Textarea` аумағы да үнсіз келісім бойынша типке ұқсас, алайда, ол жерде бірнеше жол мәтін болуы мүмкін.

Таңдау жасалатын ұзын нұсқалар тізімін бейнелеу үшін `<select>` және `</select>` тәгтері қолданылады. Мұндай тізімдер жиі ашылатын меню түрінде беріледі. Егер `multiple` (көптік) параметрі берілсе, онда оларды жалаушалармен, егер берілмесе – ауыстырып қосқышпен салыстыруға болады.

Соңында, тұтынушы өзгерте алатын бастапқы мәндерді (немесе үнсіз келісім бойынша) белгілеу әдісі бар. Мысалы, `text` аумағында `value` өрісі берілген. Өріс мәні тұтынушы оны редакциялайтындай немесе өшіре алатындай етіп бейнеленеді.

CSS – стильдердің каскадты кестесі

HTML-дың бастапқы мақсаты құжаттың сыртқы пішінін емес, оның құрылымын анықтау болатын. Мысалы,

```
<h1> Дебораның суреттері </h1>
```

жолы браузерге тақырыпты ерекшелеу керектігін хабарлайды, алайда, қаріп гарнитурасы, мөлшері және түсі жайлы ешнәрсе айтылмаған. Бұның барлығын браузер өз қалауынша таңдайды: оның нақты манитор қасиетінен (мысалы, ондағы нүктелер саны) тұратын артықшылықтары бар. Қандай да бір сәтте, дизайнерлердің құрастырылатын парақты толығымен өздері бақылағылары келді. Осы кезден бастап құжат түрінің қандай болатынын анықтайтын тәгтер пайда болды, Мысалы,

```
<font face=”helvetica” size=”24” color=”red”> Дебора суреттері </font>
```

Сонымен бірге элементтердің экрандағы нақты орнын анықтайтын тәсілдер қосылды. Бұл тәсілдің қиындығы – мұндай парақтарды құрастыру адамды жалықтырып жібереді. Нәтижесінде басқа платформаларға ауыстырылу қасиеті жоқ үлкен HTML-мәтін пайда болады. Парақ құрастырушы браузерінің экранында жақсы көрінгенмен, осы браузердің басқа версиясында және басқа браузерде мүлдем басқаша бейнеленеді.

Ыңғайлы және балама таңдау стильдер кестесін пайдалану. Мәтіндік редакторындағы стильдер кестесі авторларға мәтінді физикалық емес логикалық стильмен сәйкестендіруге мүмкіндік береді, мысалы, «курсив» орнына «бірінші абзац». Әр стильдің сыртқы пішіні жеке анықталған. Сөйтіп, егер автор бірінші абзацтың қаріпін 14 пт. көк курсивтен 18 пт. қызғылт, қалыңға ауыстырғысы келсе, бүкіл құжатты түзету керек емес, тек бір анықтаманы ауыстыру жеткілікті.

HTML 4 пайда болуымен Дүниежүзілік өрмекке стильдер кестесі – **CSS (Cascading Sryle Sheets – стильдердің каскадты кестесі)** енгізілді. Алайда, олар браузерлерде тек 2000 жылдан бастап кеңінен қолданыла бастады. CSS тәгпен белгіленген контенттің сыртқы пішіні бағынатын ережені сипаттау үшін қарапайым тілді анықтайды. Мысал қарастырайық, айталық, AWI ақ-

шыл-сары фонда, мәтіні қара-көк түспен, Arial қарпімен терілген әдемі парақты қалайды делік. Бірінші деңгейдегі тақырып қаріпінің мөлшері негізгі мәтін қарпі мөлшерінен 100% үлкен болуы керек, ал екінші деңгейдегі тақырып – 50%. CSS бұл ережені *7.6-листингте* анықтайды.

7.6-листинг. CSS мысалы

```
body {background-color:linen; color:navy; font-family:Arial;}
h1 {font-size:200%;}
h2 {font-size:150%;}
```

Өздеріңіз көргендей, стильді анықтау жинақы болуы мүмкін. Әр жол өзіне қатысты элементті және оның қасиеттерінің мәнін көрсетеді. Элемент қасиеті үнсіз келісім бойынша HTML-гі барлық элементтерге қолданылады. Сөйтіп, `body` құжаит денесіндегі абзац мәтінінің стилын анықтайды. Сонымен бірге, түс түрлерінің қолайлы шартты белгілері де бар (мысалы, red - қызыл). Алдын ала анықталмаған стиль параметрлерін браузер үнсіз келісім бойынша толтырады. Мұндай амал стильдер кестесін анықтауды қосымша етеді: парақ онсыз да ойдағыдай бейнеленетін болады.

Стильдер кестесін HTML файлына орналастыруға болады (мысалы, `<style>` тәгін пайдаланып), бірақ әдетте, ол сілтеме жасалатын жеке файлда сақталады. Мысалы, AWI парағындағы `<head>` тәгін, *7.7-листингте* көрсетілгендей, каскадты стильдер жайлы ақпаратты *awistyle.css* файлынан оқитындай етіп өзгертуге болады. Бұл мысалда сонымен бірге MIME типі *text/css* түрінде анықталады.

7.7-листинг. CSS стильдер кестесін қосу

```
<head>
<title> AMALGAMATED WIDGET, INC. </title>
<link rel=»stylesheet» type=»text/css» href=»awistyle.css» />
</head>
```

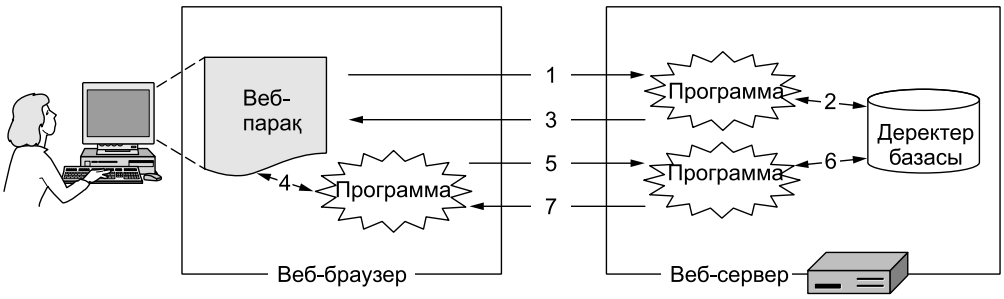
Мұндай стратегияның тек артықшылығы бар. Біріншіден, ол бір стильдер жиынтығын бірнеше веб-сайт парақтарына пайдалануға мүмкіндік береді. Сөйтіп, біз парақтардың бір үлгідегі сыртқы пішінін аламыз, тіпті, оларды әртүрлі авторлар түрлі уақытта құрастырса да, сонымен бірге барлық HTML файлдарды емес, тек CSS файлын өзгертіп, бүкіл сайттың сыртқы түрін өзгерте аламыз. Бұл тәсілді, C программалау тіліндегі `#include` директивасы арқылы тақырыптар файлын қосумен салыстыруға болады. Ол жердегі бір макрокоманданың бір анықтамасын өзгертіп, сіз оны осы тақырып қосылған барлық программалық файлдарда өзгертесіз. Екінші артықшылық, жүктелетін HTML файлдар көлемі кішірейеді, себебі браузер CSS файлына сілтемесі бар барлық файлдар үшін. оның бір көшірмесін жүктейді. Оған жеке веб-парақ үшін барлық анықтамалардың жаңа көшірмесін жүктеудің қажеті жоқ.

7.3.3. Динамикалық веб-парақтар және веб-қосымшалар

Біз осы уақытқа дейін қарастырған статикалық парақтар моделі парақты өзара ыңғайлы байланыстырылған мультимедиялық құжат ретінде қарастырады. Бұл модель Дүниежүзілік өрмектің ертедегі ақпараттың көп бөлігі онлайн орналасқан кездегі даму мақсатына сәйкес келген. Қазіргі кезде веб көбіне, қосымшалар және қызметтер үшін қолданылады. Мысал ретінде интернет-дүкендегі тауар сатып алуды, кітапхана каталогы бойынша іздеуді, картаны оқып білуді, пошта жөнелтуді, сонымен бірге бірнеше тұтынушы-ның құжатты бірлесіп өңдеуін келтіруге болады.

Пайдаланудың бұл жаңа түрлері дәстүрлі қосымшаларға ұқсас (мысалы, поштамен жұмыс істеуге арналған программалар немесе мәтіндік редакторлар). Айырмашылық – бұл қосымшалар браузерде іске қосылады, ал тұтынушылар деректері Интернеттің деректер өңдеу орталықтарындағы серверлерде сақталады. Олар веб-хаттамаларды пайдаланып, ақпаратты Интернет арқылы алады, ал браузер тұтынушы интерфейсін бейнелейді. Мұндай амалдың артықшылығы – тұтынушыға қосымшаны жеке орнатудың қажеті жоқ, ол өз деректеріне әртүрлі компьютерлерден қол жеткізе алады және ол қызмет көрсету операторларында сақталады. Әрине, бұл қосымшаларды ірі провайдерлердің ақысыз ұсынатындығы да өте маңызды. Бұл модель есептеулер тұтынушы компьютерінен бірлесіп пайдаланылатын Интернеттегі сервер кластерлеріне ауыстырылатын, **бұлттық есептеулердің (cloud computing)** таралған формасы болып саналады.

Веб-парақтар, егер олар қосымша ретінде жұмыс істеу керек болса, бұдан былай статикалық бола алмайды. Динамикалық контент қажет. Мысалы, кітапхана каталогының парағы осы сәтте қандай кітаптардың қолжетімді және қандай кітаптардың қолда, сол себепті қолжетімсіз екенін көрсетуі керек. Уақыттың әртүрлі кезеңіндегі акция құнын көріп, пайда және шығысты есептеу үшін, тұтынушының қор биржасы парағымен әрекеттесуі де осыған ұқсас болуы керек. Бұл мысалдар арқылы, динамикалық контентті генерациялайтын программаның браузерде немесе серверде (немесе екі жерде де) іске қосылғанын қалай түсінуге болады.



7.12-сурет. Динамикалық парақтар

Біз бұл бөлімде осы екі жағдайды кезек-кезек қарастырамыз. Жалпы жағдай *7.12-суретте* бейнеленгендей. Мысалы, өздеріңізге тұтынушы көше атын енгізгеннен кейін оған сол төңіректің картасын ұсынатын, картамен жұмыс істеу қызметін елестетіп көріңіз. Сұраныс қабылданғаннан кейін веб-сервер сұралған жердің картасын көше және басқада географиялық ақпарат сақталған деректер базасының көмегімен парақты құрастыратын программаны алуы керек. Бұл әрекет 1-3-қадамдар ретінде көрсетілген. Сұраныс (1-қадам) сервердегі программаның іске қосылуына әкеледі. Программа деректер базасын сұрап, қажет парақты генерациялайды да (2-қадам) оны браузерге қайтарады (3-қадам).

Алайда, бұл толық динамикалық контент емес. Қайтарылған парақтың өзі браузерде іске қосылатын программадан тұруы мүмкін. Біздің мысалымызда программа тұтынушыға маршрутты тауып, оны әртүрлі деңгейде егжей-тегжейлі зерттеуге мүмкіндік береді. Ол парақ масштабты тұтынушы сұранысына байланысты үлкейтіп немесе кішірейтіп жаңартады (4-қадам). Кейбір операцияларды орындау үшін программаға серверден көптеген деректер қажет болуы мүмкін. Бұл жағдайда программа серверге сұраныс жөнелтеді (5-қадам). Сервер деректер базасынан қажет ақпаратты тауып (6-қадам), оны жауап ретінде жөнелтеді (7-қадам). Сонан кейін программа параққа өзгертулерді енгізуді жалғастырады (4-қадам). Сұраныс және жауаптар фондық режимде өңделеді, тұтынушы ол жайлы білмеуі де мүмкін, себебі URL және парақ аты әдетте, өзгермейді. Клиент жақта орындалатын программасы бар парақтар, тек сервер жақта орындалатын программалардан тұратын парақтарға қарағанда әлдеқайда ыңғайлы интерфейс ұсынуы мүмкін.

Веб-парақ мазмұнын сервер жақта динамикалық генерациялау

Веб-парақ мазмұнын сервер жақта динамикалық генерациялауды егжей-тегжейлі қарастырайық. Сервер жақта веб-парақты генерациялауды қажет ететін қарапайым жағдай – форманы пайдалану. Тұтынушы тапсырыс формасын толтырып, Тапсырысты жөнелту батырмасына басатын жағдайды қарастырайық (*7.11-суретті* қараңыз). Тұтынушы батырмаға басқан кезде серверге, URL-ге анықталған форма бойынша *тұтынушы енгізген деректері* бар сұраныс жөнелтіледі (бұл жағдайда POST тәсілі бойынша *http://widget.com/cgi-bin/order.cgi*-ке). Бұл деректер программаға немесе скриптке өңдеуге берілуі тиіс. Сөйтіп, URL, белгілі бір программаның іске қосылуын инициализациялайды, ал енгізілген деректер бастапқы мән ретінде қабылданады. Бұл жағдайда өңдеу, тапсырысты AWI ішкі жүйесіне енгізуді: клиент жазбасын жаңартуды және кредиттік катадан ақшаны есептен шығаруды қарастырады. Осы сұранысқа жауап қайтаратын парақ өңдеу барысында қандай оқиғаның орын алатындығына байланысты. Статикалық парақ жағдайындағыдай нәтиже бекітілмеген. Егер тапсырыс ойдағыдай өңделсе, онда қайтарылатын парақта тауарды жеткізу датасы көрсетілуі мүмкін. Егер сұраныс дұрыс өңделмесе,

онда қайтарылатын парақта тауардың қолда жоқтығы, кредиттік картаның қабылданбағаны немесе басқа да себептер көрсетілуі мүмкін.

Сервердің файлды іздеудің орнына программаны қалай іске қосатыны веб-сервер құрылымына байланысты. Бұл веб-хаттамамен анықталмайды. Осы себептен интерфейс сайт иесі – компанияның талаптары бойынша құрастырылуы мүмкін. Браузерге барлық егжей-тегжейді білу қажет емес. Браузер тек сұранысты құрастырады және парақты қабылдайды.

Осылай болса тұрса да, веб-сервер үшін программаны іске қосуға арналған стандартты API құрастырылған болатын. Бұл интерфейстердің болуы құрастырушылардың түрлі серверлерді веб-қосымшалар есебінен кеңейтуге аз күш жұмсауға мүмкіндік береді. Сіздерге түсінікті болу үшін, біз екі API-ды қысқаша қарастырамыз.

API – динамикалық парақтар сұранысын өңдеу тәсілі. Ол Дүниежүзілік өрмек пайда болғаннан бері қолжетімді. Ол **CGI (Common Gateway Interface – жалпы шлюздік интерфейс)** деп аталады және RFC 3875 құжатында анықталған. CGI, веб-серверлерге деректерді қабылдай алатын (мысалы, формадан) қолданбалы программалармен және скриптермен әркеттесіп, жауап ретінде HTML-парақ генерациялайтын интерфейс ұсынады. Бұл программалар тұтынушы таңдап алған кез келген тілде және қарапайымдылық үшін скриптерді пайдаланып жазылуы мүмкін. Өзіңізге ұнайтын Python, Ruby, Perl немесе басқа тілді таңдап алыңыз.

CGI арқылы іске қосылатын програмамалар URL-ге көрінетін CGI-BIN каталогында сақталу керек деген келісім бар. Сервер сұранысты осы каталогтағы программа атына бейнелеп, программаны жеке үдеріс ретінде іске қосады. Ол программаға сұраныспен бірге, кіріс деректер түрінде жөнелтілген кез келген деректерді береді. Программа шығысында браузерге жөнелтілетін веб-парақ алынады.

Біздің мысалымызда *order.cgi* программасы *7.5-листингте* көрсетілгендей формаға енгізілген деректермен шақырылады. Ол параметрлерді сараптап, сұранысты өңдейді. Форма толтырылмаған жағдайда HTML тапсырыс формасы қайтаруы керек деген келісім өте пайдалы. Сөйтіп, программа форманың қандай түрде берілгендігін білетін болады.

Біз әңгіме қозғайтын екінші API жоғарыда жазылғаннан ерекше. Бұл тәсіл HTML-парақтарға кішігірім скриптерді енгізуді талап етеді. Олар серверде орындалады, олардың қызметі парақты генерациялау. Мұндай скриптерді жазудың танымал құралы **PHP (PHP: Hypertext Preprocessor – PHP: Гипермәтіндік процессор)**. Оны пайдалануда сервер PHP түсінуі керек (стильдер кестесі пайдаланып жазылған парақты талдау кезінде браузер CSS-ті түсінгендей). Әдетте, сервер PHP-де жазылған веб-парақты *htm* немесе *html* емес, *php* кеңейтілімі арқылы ажыратады.

CGI пайдаланғаннан PHP-ді пайдалану жеңіл. Форманы PHP-мен өңдеу мысалы *7.8.a-листингінде* келтірілген. Листингтің жоғарғы бөлігінде қарапайым формасы бар HTML-парақ келтірілген. Бұл жолы `<form>` растау батырмасы

басылғаннан кейін параметрлерді өңдеу үшін *action.php* іске қосылу керектігіне нұсқайды. Бұл мысалда форма мәтін енгізілетін екі өрістен тұрады, біріншісі тұтынушы атын, екіншісі – жасын сұрайды. Тұтынушы формамен жұмысты аяқтағаннан кейін, серверге жоғарыдағы мысалда көргендей стандартты жол жөнелтіледі. Бұл жол өңделіп, `name` және `age` айнымалыларының мәндері алынады. Бұдан кейін, *7.8. ә-листингінде* көрсетілгендей *action.php* скрипті өз жұмысын бастайды. Ол жауапты даярлайды. Срипт жұмысы – php-команданы орындау. Егер тұтынушы «Барбара» және «24» деректерін енгізсе, онда оған *7.8. б-листингінде* көрсетілгендей HTML-файл жөнелтіледі. Осылайша форманы PHP көмегімен өңдеу қарапайым жүргізіледі.

7.8-листинг. Формасы бар веб-парақ (а); Форманы өңдеуге арналған PHP-скрипт (ә); Бастапқы мәндер сәйкесінше «Барбара» және «24» болғандағы PHP-скрипт жұмысының нәтижесі (б)

```
<html>
<body>
<form action=»action.php» method=»post»>
<p> Өз атыңызды енгізіңіз: <input type=»text» name=»name»> </p>
<p> жасыңызды енгізіңіз: <input type=»text» name=»age»> </p>
<input type=»submit» value=»Растау» >
</form>
</body>
</html>
```
(а)

```
<html>
<body>
<h1> Жауап: </h1>
Hello <?php echo $name; ?>.
Болжау бойынша келесі жылы сізге <?php echo $age + 1; ?>
</body>
</html>
```
(ә)

```
<html>
<body>
<h1> жауап: </h1>
Саламатсың Барбара.
Болжау бойынша саған келесі жылы 33 болады </body>
</html>
```
(б)

Қолданудың қарапайымдылығына қарамастан, PHP – Дүниежүзілік өрмек және деректер базасы серверлерімен әрекеттесудің қуатты програмалау тілі. PHP-де С тіліне тән айнымалы, жол, массив және көптеген басқарушы құрылымдардың барлығы бар, алайда, енгізу/шығару әдеттегі *printf*-ке қарағанда

әлдеқайда қуатты. PHP-дің ашық түрдегі бастапқы коды бар, ол тегін таратылады және кеңінен қолданылады. PHP, ашық түрдегі бастапқы коды бар және әлемде кең таралған веб-сервер болып саналатын Apache сервері үшін арнайы құрастырылған болатын. PHP жайлы толық ақпаратты Valade (2009) басылымынан алуға болады.

Сонымен, біз динамикалық HTML-парақты генерациялаудың екі түрлі: CGI-скрипттері және PHP-ді енгізу арқылы жолын білеміз. Бұдан басқа тағы бірнеше тәсілдер бар. **JSP (JavaServer Pages – Java серверінің парағы)** – жалпы PHP-ге ұқсас, тек парақтың динамикалық бөлігі Java тілінде программаланады. JSP көмегімен жазылған файлдар парағының аттас кеңейтілімі бар: *.jsp*. **ASP .NET (Active Server Pages .NET – .NET активті сервер парағы)** – бұл PHP және JSP-ге Microsoft жауабы. Бұл жерде динамикалық конттентті генерациялау үшін, Microsoft жасап шығарған. NET желілік қосымшалар ортасында құрастырылған жекеменшік программалар пайдаланылады. Осы тәсіл көмегімен құрастырылған парақтар файлының *.aspx* кеңейтілімі бар. Осы үш техника арасындағы таңдау негізінен саяси (Microsoft қодына қарсы ашық бастапқы код) тұрғыда жүргізілуде. Технология тұрғысынан бұл тәсілдердің барлығының мүмкіндіктері бірдей.

Клиент жақта динамикалық веб-парақты құрастыру

CGI және PHP скрипттері енгізілетін деректер және серверде орналасқан деректер базасымен әрекеттесу мәселелерін шешеді. Олар формадан кіріс ақпаратты қабылдап, бір немесе бірнеше деректер базасынан қажетті ақпаратты іздеп тауып, нәтиже ретінде HTML-парақты генерациялай алады. Алайда, бұл тәсілдердің ешқайсысы тұтынушымен тікелей әрекеттесуге, мысалы, тышқан қолтетігі көрсеткішінің қозғалысына мүмкіндік бермейді. Бұл үшін HTML-параққа енгізілген және сервер емес, клиент жақта орындалатын скрипттер қажет. HTML-дің 4.0 версиясынан бастап, осындай скритпттерді `<script>` тәгінің көмегімен қосу мүмкіндігі пайда болды. Осы интерактивті веб-парақты құрастыруда қолданылған технологиялар жиі **динамикалық HTML (dynamic HTML)** деп қате аталады.

Клиент жаққа сценарий жазуға арналған анағұрлым танымал тіл – **JavaScript**. Біз оны төменде қысқаша қарастырамыз. Аттарының ұқсастығына қарамастан JavaScript, Java тіліне мүлдем ұқсамайды. Скриптер жазуға арналған басқа тілдер тәрізді ол жоғары деңгейлі. JavaScript-тің бір жолымен сұхбат терезесін құрастыруға, тұтынушы енгізуінің күту цикліне кіруге және алынған жолды айнымалыда сақтауға болады. Тілдің осыншалық жоғары деңгейлігі интерактивті веб-парақ құрастыруға өте ыңғайлы. Екіншіден, JavaScript-тің стандартталмауы және рентген сәулесіндегі дрозофил-шіркейден де жылдам мутацияланауы, белгілі бір платформаға тәуелсіз программа жазуды қиындатады. Осылай бола тұрсада, тіл ерте ме, кеш пе тұрақтанады деп сену керек.

JavaScript-те жазылған программа мысалы *7.9-листингте* келтірілген. *7.8. а-листингідегідей* программа тұтынушы атын және жасын сұрайтын форма құрастырады және осы деректер бойынша келесі жылы оның жасы нешеде болатынын болжап, айтып береді. Скрипт денесі PHP мысалына ұқсас. Негізгі айырмашылық Растау батырмасын хабарлауда және осы хабарлаудағы меншіктеуді анықтауда. Меншіктеу операторы браузерге батырмаға басқан кезде `response` скриптін іске қосып, оған параметр ретінде форманы беру керектігін хабарлайды.

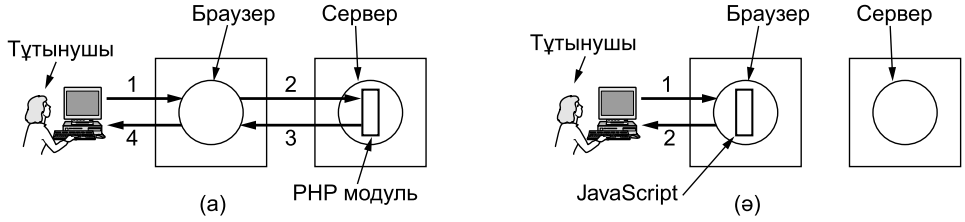
Мұнда, `response` функциясы мүлдем жаңаша хабарланады. Хабарландыру, әдетте, тақырып, оның фон түсі және т.б. ақпаратты сақтайтын HTML-файл тақырыбында орналасқан. Функция формадан `name` өрісінің мәнін алып, оны жол ретінде `person` айнымалысында сақтайды. Сонымен бірге `age` өрісінің мәні де алынады. Ол `eval` функциясының көмегімен бүтін типке келтіріліп, сонан кейін оның мәнінен 1 қосылып, нәтиже `years` айнымалысында сақталады. Осыдан кейін құжат жазу үшін ашылады, оған төрт жол жазылады да (бұл үшін `writeln` тәсілі қолданылады) құжат жабылады. Көптеген HTML тәгтерден байқағаныңыздай құжат HTML-парақ болып келеді. Браузер дайын құжатты экранға шығарады.

7.9-листинг. Форманы өңдеу үшін JavaScript пайдалану

```
<html>
<head>
<script language=»javascript» type=»text/javascript»> function response(testJorm) {
var person = testJorm.name.value;
var years = eval(testJorm.age.value) + 1;
document.open();
document.writeln(«<html> <body>»);
document.writeln(“Саламатсың « + person + «.<br>»); document.writeln(«Болжау бойынша келесі жылы сағаң « + years + « болады.»);
document.writeln(«</body> </html>»);
document.close();
}
</script>
</head>

<body>
<form>
Өз атыңызды енгізіңіз: <input type=»text» name=»name»>
<p>
Жасыңызды енгізіңіз: <input type=»text» name=»age»>
<p>
<input type=»button» value=»Растау»
onclick=»response(this.form)»>
</form>
</body>
</html>
```

PHP және JavaScript кодты HTML-файлда орналастыру бойынша ұқсас болғанымен, өңдеуді мүлде басқа жолмен жүргізетіндігін түсінген дұрыс. *7.8-листингтегі* PHP мысалында тұтынушы *«Растау»* батырмасына басқаннан кейін браузер барлық енгізілген ақпаратты бір ұзын жолға жинақтап, PHP-парақ сұранысы ретінде серверге жөнелтеді. Сервер PHP-файлды жүктеп, сол жерде браузер бейнелейтін жаңа HTML-парақты құрастыру үшін орналасқан PHP-скриптті іске қосады. Браузер оның программамен генерацияланғаны білмейді. Өңдеу *7.13 а-суретінде* 1-4-қадам ретінде көрсетілген.

JavaScript мысалында (*7.9-листингті* қараңыз) «Растау» батырмасына басқаннан кейін браузер парақтағы JavaScript әрекетін өзі орындайды. Сервермен ешқандай әрекеттесу жүргізілмейді. Бұл *7.13 ә-суретінде* 1-2-қадамында көрсетілген. Осының салдары ретінде нәтиже экранда бірден пайда болады, ал PHP-ді пайдаланған кезде нәтиже бейнеленген парақтың келу уақыты бірнеше секунды құрайды.



7.13-сурет. Сервер жақтағы PHP-скрипт (а); клиент жақтағы JavaScript сценарийі (ә)

Бұл айырмашылық JavaScript PHP-ден «жақсы» дегенді білдірмейді. Олардың қолданылу аймақтары әртүрлі. PHP (онымен бірге JSP және ASP анық емес) қашықтықтағы деректер базасымен әрекеттесу керек болған жағдайда қолданылады. JavaScript (және клиент жақтағы басқа да тілдер, мысалы, VBScript) тұтынушымен оның өз компьютері шеңберінде әрекеттесу керек болған жағдайда қолданылады. Әрине, бірмезгілде PHP-ді де, JavaScript-ті де пайдалануға болады. Біз бұл жайлы қысқаша айтып кетеміз.

JavaScript – жоғары интерактивті веб-парақ құрастырудың жалғыз құралы емес. Windows платформасындағы оның балама таңдауы Visual Basic-ке негізделген **VBScript**. Әртүрлі платформада қолдануға болатын тағы бір тәсіл **апплеттерді (applets)** пайдаланумен байланысты. Бұл **JVM (Java Virtual Machine – Java виртуалды машинасы)** деп аталатын виртуалды компьютер машиналық нұсқауларына компиляцияланған, Java тіліндегі кішігірім программалар. Апплеттер HTML-парақтарға енгізіліп (`<applet>` және `</applet>` тәгтерінің арасына), JVM-үйлесімді браузерлермен интерпретацияланады. Java-апплеттер орындалмас бұрын интерпретациядан өтетін болғандықтан, интерпретатор «нашар әрекеттерді» болдырмауға көмектеседі. Теориялық тұрғыдан осындай мүмкіндік бар. Іс жүзінде апплет құрастырушылар Java енгізу/шығару ағынындағы шексіз қателіктерді анықтады.

Microsoft корпорациясының Sun фирмасының Java-апплеттеріне жауабы веб-парақтарға **ActiveX басқарушы элементтерін (ActiveX controls)** – x86 процессорының машиналық тілі үшін компиляцияланған және аппараттық деңгейде орындалатын программаны қосу болды. Бұл қасиет оларды интерпретацияланатын Java-апплеттеріне қарағанда әлдеқайда шапшаң және иілгіш етеді, себебі олар қарапайым программа орындайтын әрекеттердің барлығын орындайды. Internet Explorer парақта ActiveX басқарушы элементтерін көрген кезде ол оны жүктеп, сәйкестендіреді. Алайда, бөтен программаны жүктеу, біз *8-тарауда* қарастыратын ақпаратты қорғау мәселелерін туындатуы мүмкін.

Брузерлердің барығы түгелдей дерлік программаларды Java да, JavaScript-те де интерпретациялай алатын болғандықтан, жоғары дәрежеде интерактивті парақ құрастырғысы келетін құрастырушы кем дегенде, осы екі технологияның бірін таңдай алады. Ал егер ол үшін платформа маңызды болмаса, онда оларға ActiveX қосылады. Жалпы ереже:

- JavaScript программалау қарапайымдылығын қамтамасыз етеді;
- Java-апплеттер орындалу жылдамдығын қамтамасыз етеді;
- ActiveX басқарушы элементтері орындалу жылдамдығы жағынан барлық технологиялардан ілгері келеді.

Барлық браузерлерде қатаң түрде JVM-ның бір версиясы жүзеге асырылған, ал JavaScript бірдей жүзеге асырылған екі браузерді табу өте қиын. Сондықтан Java-апплеттер JavaScript-ке қарағанда бір платформадан екіншісіне жеңіл алмастырылады. JavaScript арналған үлкен көлемдегі (жиі 1000 парақтан астам) кітаптар көп. Мысалы, Flanagan (2010) қарауға болады.

AJAX – асинхронды JavaScript және XML

Күрделі веб-қосымшаларға ыңғайлы тұтынушы интерфейсі және қашықтықтағы веб-серверлерде орналасқан деректерге қарапайым қол жеткізу қажет. Клиент жақта (мысалы, JavaScript көмегімен) және сервер жақта (мысалы, PHP көмегімен) скрипті жазу – бұл осы мәселе шешімін ұсынатын негізгі технологиялар. Бұл технологиялар әдетте, **AJAX (Asynchronous JavaScript and XML – асинхронды JavaScript және XML)** деп аталатын біршама өзгеше комбинацияны пайдаланылады. Көптеген толық функционалды веб-қосымшалар, Google-ден бастап Gmail, Maps және Docs тәрізді, AJAX жазылған.

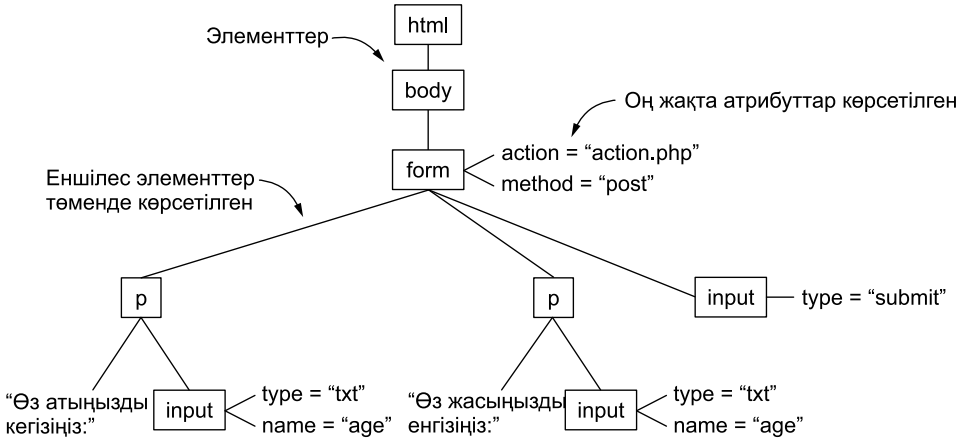
AJAX тіл болмағандықтан біршама оғаштау. Ол дәстүрлі қолданбалы жүйелер тәрізді ыңғайлы және функционалды бола алатын, веб-қосымша жасау үшін бірлесіп жұмыс істейтін технологиялар жиынтығы.

Бұл келесі технологиялар:

1. Ақпаратты парақ түрінде бейнелеуге арналған HTML және CSS.
2. DOM (Document Object Model – құжаттардың объектілік моделі) көру кезінде парақ бөліктерін алмастыру үшін.

3. XML (eXtensible Markup Language – кеңейтілмелі белгілеу тілі) программалар сервермен қосымшалар деректерімен алмасу үшін.
4. Асинхронды амал – оның көмегімен программалар XML-деректерді жөнелтеді және қабылдайды.
5. JavaScript – осы технологиялардың барлығын байланыстыратын тіл.

Осының барлығы белгілі бір жиын болғандықтан, біз олардың әрқайсысын жеке қарастырып, веб-қосымша жасаудағы үлесін бағалаймыз. Біз HTML және CSS-ті жоғарыда қарастырдық. Олар контентті және олардың қалай бейнелену керектігін сипаттау стандарт болып келеді. HTML және CSS-ті құрастыра алатын кез келген программа веб-браузерді бейнелеу құралы ретінде пайдалана алады.



7.14-сурет. *7.8 а-листингіндегі* HTML үшін DOM ағашы

DOM – программаға қолжетімді HTML-парақты ұсыну. Бұл ұсыныстың, HTML-элементтер құрылымын бейнелейтін ағаш тәріздес құрылымы бар. Мысалы, *7.8 а-листингінде* келтірілген HTML-мәтіннің DOM-ағашы *7.14-суретте* бейнеленген. Тамырда бүкіл HTML блогын ұсынатын, *html* элементі орналасқан. Бұл элемент, *body* элементіне қатысты ата-аналық болып келеді, ал өз кезегінде *form* үшін ата-аналық болып келеді. *form* элементінің екі атрибуты бар, сіз оларды оң жақта көре аласыз, олардың бірі форма тәсіліне (*POST*) қатысты және формамен орындалатын әрекеттер үшін тағайындалған (URL-сұраныс). Бұл элементтің үш еншілес элементі: екі абзац тәгі және форматы ақпарат енгізуге арналған бір тәг бар. Төменгі бөлікте не элемент, не жолдық мәтін тәрізді тұрақты болып келетін ағаштың жапырақтары орналасқан. DOM моделінің мағынасы – ол программаларға парақ бөліктерін алмастырудың қарапайым амалын ұсынады. Сөйтіп, парақты толығымен қайта жазу қажеттілігі туындамайды. Тек өзгерту енгізілген түйін алмастырылады. Өзгеріс

енгізілген түйін алмастырылғаннан кейін, браузер парақтың сәйкес бөлігін алмастырады. Мысалы, егер DOM-да парақтың сурет орналасқан бөлігі өзгертілген болса, браузер осы суретті жаңартады, ал парақтың қалған бөлігі өзгеріссіз қалады. Біз DOM-ның қалай жұмыс істейтінін *7.9-листингтегі* JavaScript мысалында, браузер терезесінің төменгі бөлігінде жаңа жол пайда болу үшін, *document* элементіне жолдары қосқан кезде көрдік. DOM – өзгермелі парақ құрастырудың тамаша амалы.

Үшінші технология, **XML** – құрылымдық контентті сипаттауға арналған. HTML контентті форматтаумен араластырады, себебі ол ақпаратты ұсыну тәсілімен байланысты. Алайда, веб-қосымшалардың кең таралуымен байланысты, құрылымдық контентті және оны ұсынуды ажырату қажеттілігі туындап отыр. Мысалы, желідегі қандайда бір кітапқа ең жақсы бағаны іздейтін программаны қарастырайық. Веб-парақ HTML-де жазылған жағдайда, программаға кітап аты және оның бағасы қай жерде көрсетілгендігін анықтау өте қиын.

Осы себептен, WWW (W3C) консорциумы XML-ді құрастырды (Bray және басқалар, 2006). XML – веб-контентті автоматты түрде өңдеу үшін құрылымдайды. XML-дың HTML-дан ерекшелігі ол үшін нақты анықталған тәгтер жоқ. Әр тұтынушы өз тәгін құрастыра алады. XML-құжаттың қарапайым мысалы *7.10-листингте* келтірілген. Ол кітаптар тізімі болып келетін `book_list` атты құрылымды анықтайды. Әр кітаптың: аты, авторы және шыққан жылы үш өрісі бар. Бұл құрылымдар өте қарапайым. Құрылымдағы өрістер қайталануы (мысалы, авторлар ұжымы болса), қосымша өрістер (мысалы, аудио-кітап URL-і) және балама таңдау өрістері (мысалы, кітап дүкені URL, егер кітап бар болса немесе аукцион сайт URL, егер ол сатылып кетсе) болуы мүмкін.

7.10-листинг. Қарапайым XML құжат

```
<?xml version=»1.0» ?>
<book_list>
<book>
<title> адам тәртібі және күш-қуатты үнемдеу қағидалары </title>
<author> Джордж Зипф </author>
<year> 1949 </year>
</book>
<book>
<title> Коммуникациялық математикалық теориясы </title>
<author> Клод Э.Шенон </author>
<author> Уоррен Вивер </author>
<year> 1949 </year>
</book>
<book>
<title> Мың тоғыз жүз сексен төрт </title>
<author> Джордж Оруэл </author>
<year> 1949 </year>
</book>
</book_list>
```

Бұл мысалда үш өрістің әрқайсысы бөлінбейтін бөлік болып саналады, бірақ теориялық тұрғыдан оларды әрі қарай бөлуге болады. Мысалы, іздеуді және форматтауды жақсылап қадағалау үшін «автор» өрісі:

```
<author>
<first_name> Джордж </first_name>
<last_name> Зипф </last_name>
</author>
```

болып рәсімделуі мүмкін.

Әр өріс ішкі өріске (ал ішкі өріс келесі ішкі өрістерге және т.с.с.) еркін сан ретінде бөлінуі мүмкін.

7.10-листингінде бейнеленген файл, үш кітап тізімін жай ғана анықтайды. Ол браузер және серверлерде жұмыс істейтін программалар арасында ақпаратты тасымалдау үшін қолайлы, бірақ құжаттың веб-парақ түрінде қалай ұсынылу керектігі жайлы ешнәрсе хабарламайды. Бұны білу үшін ақпаратты жинайтын программа 1949 жыл кітаптар үшін өте қолайлы жыл болған екен деп, аттары курсивпен ерекшеленген HTML-ды беруі мүмкін. Бұдан басқа, **XSLT (eXtensible Stylesheet Language Transformations – кеңейтілмелі белгілеу тілінің стильдік транформациясы)** деп аталатын тіл қалайша XML-ді HTML-ге трансформациялау керектігін анықтайды. XSLT CSS-ке ұқсас, бірақ оның мүмкіндіктері әлдеқайда көп. Біз оны егжей-тегжейлі қарастырмаймыз.

Деректерді HTML емес, XML түрінде ұсынудың тағы бір артықшылығы программаларға оны сараптау жеңіл. Бастапқыда HTML-да қолмен жазылатын (қазір де көп өзгере қойған жоқ), сондықтан ол көбіне, біршама «шикі» болып көрінеді. Кейде, `</p>` тәрізді жабылатын тәгтер қалып кетеді. Ал басқаларының, мысалы, `<br>` тәгінің сәйкес жабылатын тәгі жоқ. Сонымен бірге, басқа тәгтер қажет жерде емес, кез келген жерде орналасуы мүмкін. Көптеген браузерлерге мұндай кодты сараптау оңай емес. XML құрылымы әлдеқайда қатаң. Тәг аттары және атрибуттары әрқашан кіші әріппен жазылады, тәгтер пайда болу ретіне кері ретпен жабылуы тиіс (немесе белгілі бір тәгке жабылатын тәгтің керек еместігі қатаң түрде анықталады), ал атрибуттар мәні тырнақшаға алынады. Анықталудың қатаңдығы сараптауды жеңілдетеді және нәтиже күтілгендей болады.

HTML тіпті, XML терминдерінде де анықталған. Бұл алмасу **XHTML (eXtended HyperText Markup Language – кеңейтілген гиперсілтемелік белгілеу тілі)** деп аталады. Жалпы алғанда, бұл талаптары қатаң бекітілген HTML тілі. XHTML парақтары XML ережелеріне қатаң сәйкес болуы керек, кері жағдайда оны браузер қабылдамайды. Осылайша, төмен сапалы веб-парақтар және браузер мен парақтар үйлесімсіздігі жойылды. Мақсат – XML жағдайындағыдай программалар жеңіл өңдейтін парақтар құрастыру (бұл жағдайда веб-қосымшалар). XHTML 1998 жылдан бері енгізілсе де, ол ұзақ уақытқа дейін кеңінен тарала алмады. HTML-ді пайдаланушылар XHTML не үшін керек екенін түсіне алмады, сондықтан браузерлер оны көпке дейін қолдамады. Бүгінде, жаңа стандартқа көшу анағұрлым ыңғайлы болу үшін,

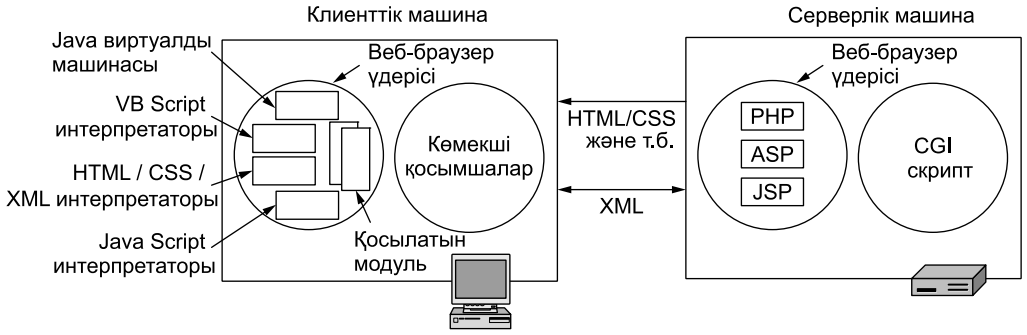
HTML 5.0 парақтар не HTML-де, не XHTML-де жазылатындай етіп анықталған. Нәтижесінде XHTML HTML-ды алмастыруы керек, бірақ бұл алмасу аяқталу үшін көп уақыт өтеді.

XML-де программалар арасындағы коммуникациялық тіл ретінде танымал болды. Әрекеттесу HTTP хаттамасының (келесі бөлімде сипатталған) көмегімен жүргізілген кезде веб-қызмет (Web servise) деп аталады. Программалар арасында қашықтықтағы процедураны шақыруға арналған, веб-қызметті программа тіліне және платформасына тәуелсіз жүзеге асыратын, **SOAP (Simple Object Access Protocol – объектілерге қолжеткізудің қарапайым хаттамасы)** хаттамасына назар аудару қажет. Клиент XML-мәлімде формасында сұраныс құрастырады да, оны http хаттамасын пайдаланып серверге жөнелтеді. Сервер дәл осындай формадағы жауап жібереді. Осылайша, әртүрлі платформадағы қосымшалар бірлесіп жұмыс істей алады.

AJAX-ке оралып, біз тек XML-дің браузер және сервер арасында жұмыс істейтін программалар ішіндегі деректер алмасудың пайдалы форматы екенін айта кеткіміз келеді. Алайда, деректерді алған немесе жөнелткен кезде, браузерде жылдам жауап қайтаратын және жауапты күту уақытында дисплейді оқшауландырмайтын интерфейс құрастыру үшін скрипттердің **асинхронды енгізу/шығару** операцияларын орындау мүмкіндігі болуы керек. Мысалы, браузерде айландыруға болатын картаны қарастырайық. Скрипт тұтынушы картаны айландыра бастады деген мәлімдеме алған кезде, қаралып жатқан карта бөлігі қабылданған шекараға жақын болса, ол серверден карта жайлы көп деректер сұрауы мүмкін. Сұралған деректер алынғанша интерфейс қатып қалмауы керек, кері жағдайда ол қолданыстан алынып тасталуы мүмкін. Оның орнына айландыру еш кедергісіз жүруі тиіс. Деректер алынған кезде скрипт ол жайлы мәлімдеме алады және жаңа деректерді пайдаланады. Егер барлығы жоспарлағандай болса, жаңа деректер оларға деген қажеттілік туындағанша алынады. Қазіргі заманғы браузерлерде коммуникацияның осындай моделін қолдау бар.

Жұмбақтың соңғы бөлігі – жоғарыда айтылған технологияларға қол жеткізу үшін AJAX біртұтас біріктіретін скрипттер тілі болып саналады. Көп жағдайда бұл JavaScript тілі, алайда, VBScript тәрізді балама таңдау да бар. Біз ертеректе JavaScript-ке қарапайым мысал келтірген болатынбыз. Бұл бір қарағандағы қарапайымдылық сіздерді алдамасын. JavaScript-тің көптеген жеке ерекшеліктері бар, бұл – C және Java тілінің мүмкіндіктерін қамтыған толыққанды программалау тілі. Онда айнымалылар, жолдар, массивтер, объектілер, функциялар және әдеттегі барлық басқарушы құрылымдар, сонымен бірге, оның браузер және веб-парақ ерекшеліктерімен байланысты интерфейсі бар. JavaScript тышқан қолтетігі көрсеткішінің қозғалысын қадағалай алады, осының арқасында кенеттен ашылатын меню жасауға және парақты әлдеқайда интерактивті етіп құрастыруға болады. Ол парақтарға қол жеткізу үшін DOM-ды пайдалана алады, сонымен бірге HTML және XML-мен де жұмыс істеп, асинхронды HTTP-коммуникациялар жүргізе алады.

Динамикалық парақтар жайлы мәселені аяқтамас бұрын, жоғарыда айтылған технологияларға жеке мысал келтіріп, ойымызды тұжырымдай кетейік. Толық веб-парақтар серверде орналасқан түрлі скрипттер көмегімен құрастырылуы мүмкін. Скрипттер PHP, JSP немесе ASP.NET тәрізді серверлік кеңейтілім тілінде жазылуы немесе CGI жеке үдерістерін іске қосуы мүмкін, сөйтіп кез келген тілде жазылады. Бұл мүмкіндіктер 7.*15-суретте* бейнеленген.



7.15-сурет. Динамикалық парақты құрастыру үшін қолданылатын түрлі технологиялар

Браузер веб-парақты алған кезде, олар әдеттегі HTML, CSS және басқа да MIME түріндегі парақ ретінде өңделеді және қарапайым бейнеленеді. Браузерде іске қосылатын плагиндер және браузерден тыс іске қосылатын көмекші қосымшалар, браузер қолдайтын MIME кеңейту үшін инсталяциялануы мүмкін.

Динамикалық контентті клиент жақта да құрастыруға болады. Веб-парақтарда орналасқан программалар JavaScript, VBScript, Java және басқа да тілдерде жазылуы мүмкін. Бұл программалар еркін есептеулер жүргізіп, дисплейді жаңарта алады. AJAX көмегімен веб-парақтағы программалар сервермен XML-файлдар және басқа да деректермен асинхронды алмаса алады. Бұл модель дәстүрлі қосымша тәрізді түрлі веб-қосымшаларды қолдайды. Жалғыз айырмашылық – олар браузерде жұмыс істейді және Интернеттегі серверлерде сақталған ақпаратты қабылдайды.

7.3.4. HTTP – гипермәтінді тасымалдау хаттамасы

Енді біз веб-контент және веб-қосымшаның не екенін білгеннен кейін, осы ақпараттың барлығын, веб-серверден клиентке және кері тасымалдайтын хаттамаға тоқталатын кез келді. Бұл – RFC 2616-да анықталған **HTTP (HyperText Transfer Protocol – гипермәтінді тасымалдау хаттамасы)**.

HTTP – сұраныс-жауап қағидасы бойынша жұмыс істейтін қарапайым хаттама. Ол TCP арқылы іске қосылады және клиенттің серверге қандай мәлім-

демелер жөнелтетінін және қандай жауап алатынын анықтайды. Сұраныстар және жауаптар тақырыбы SMTP-дағыдай ASCII берілген. Олардың мазмұны да SMTP-дағыдай MIME форматына ұқсас. Осы қарапайым модель Дүниежүзілік өрмектің бастапқы сатыдағы танымалдық себептерінің бірі, себебі ол оның жылдам даму және таралуына себепкер болды.

Бұл бөлімде біз HTTP-дің қазіргі қолданыстағы түрінің маңызды қасиеттеріне тоқталамыз. Алайда, егжей-тегжейге көшпес бұрын, оның Интернеттегі қолданыс түрінің дамып, өзгеріп отырғанын ескерте кеткен жөн. HTTP – қолданбалы деңгей хаттамасы, себебі ол TCP үстінде жұмыс істейді және вебпен сәйкестендіріледі. Сондықтан біз оны осы тарауда қарастыруды жөн көрдік. Ал контентті бір желіден екінші желіге тасымалдау тұрғысынан HTTP транспорттық хаттамаға көп ұқсас. Бұл үдерістер, үнемі веб-браузер және веб-сервер әрекеттесу шеңберінде жүреді. Медиаплеер HTTP-ді қолданып, серверге жүгініп, альбом жайлы ақпарат ала алады. Вирусқа қарсы программа HTTP-ді пайдаланып соңғы жаңартуларды жүктей алады. Құрастырушылар – қандай да бір жоба бойынша файлдарды алу үшін. Тұрмыстық электроника, мысалы, сандық фотолар рамкасы алу үшін, HTTP-ді жиі қолданады – сервер оларды сыртқы әлеммен байланыстыратын интерфейс. Компьютерлер арасындағы коммуникация жиі HTTP арқылы жүреді. Мысалы, әуекомпания сервері машинаны жалға беру серверімен байланысу үшін SOAP-ті (HTTP арқылы XML RPC) пайдаланады, сөйтіп, турды сатып алу кезінде, автомобильді қызмет түрі ретінде ұсынып, қорда сақтай алады. HTTP-ді пайдаланудың кеңеюімен бірге осы бағыттағы дамуда жалғасуы ықтимал.

Байланысу

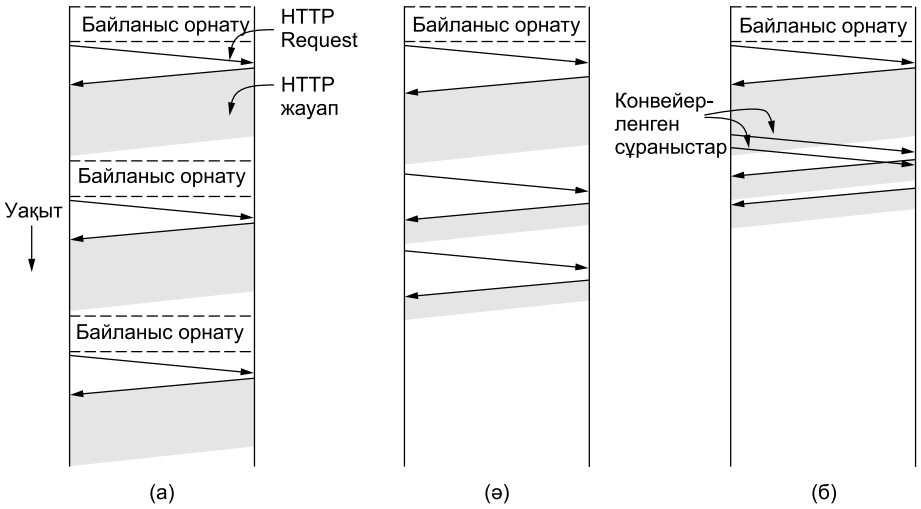
Браузер және сервердің әдеттегі әрекеттесуі, сервердің 80 портымен TCP-байланыс орнатылуына негізделген, бейресми түрде бұл процедура міндетті емес. TCP-ді пайдаланудың маңыздылығы, сервер де, браузер де тым ұзын мәлімдемелер, сенімділік және асыра жүктеуді қадағалау жайлы ойланбайды. Бұның барлығын TCP хаттама қамтамасыз етеді.

Ертеде, HTP 1.0-де бір сұранысқа бір жауап келетін. Осыдан кейін TCP-байланыс үзілетін. Ол кезде веб-парақ толығымен HTML-мәтіннен тұратын және осындай әрекеттесу әдісіне тән болатын. Кейіннен орташа веб-парақ көптеген шартты белгілерге сілтеме және басқа да безендіру элементтеріне толы бола бастады. Бір шартты белгіне тасымалдау үшін TCP-байланыс орнату тиімсіз және қымбат бола бастады.

Осы ой **тұрақты байланысты (persistent connection)** қолдайтын HTTP1.1 хаттамасын құрастыруға әкелді. Бұл сұранысты жөнелту, жауап алу, ал сонан кейін қосымша сұраныстар мен жауаптар жөнелтіп, қабылдау мүмкіндігі пайда болды дегенді білдіреді. Бұл стратегия байланысты **қайталап пайдалану (connection reuse)** деп аталады. Сөйтіп, байланысты қайта-қайта орнатып,

үзумен байланысты үстеме шығындар азайды. Сонымен бірге сұраныстарды конвейерлеу мүмкіндігі, яғни 2-сұранысты 1-сұранысқа жауап келгенше жөнелту туды.

Осы үш жағдайдағы өнімділік айырмашылығы *7.16-суретте* көрсетілген. Бірінші, а бөлігінде біз бірінен соң бірі жөнелтілген үш сұраныс көреміз және олардың әрқайсысы үшін жеке байланыс орнатылады. Бұл сұраныстар бір серверде орналасқан екі суреті бар парақпен жөнелтілген деп жорамалдаймыз. Сурет URL бастапқы парақты алғаннан кейін анықталады, сондықтан олар кейіннен алынады. Қазіргі кездегі қарапайым парақта, қабылданып, бейнелеуді қажет ететін 40 шамадай объектілер орналасады. Бұл біздің суретімізді тым үлкен етер еді. Сондықтан біз мысалымызға тек екі суретті орналастырдық.



7.16-сурет. HTTP: а – көптік байланыс және тізбектік сұраныстар; ә – тұрақты байланыс және тізбектік сұраныстар; б – тұрақты байланыс және конвейерлік сұраныстар

7.16 ә-суретінде парақ тұрақты байланыс арқылы алынған. Демек, TCP-байланыс басынан бастап ашылады, сонан кейін сол үш сұраныс алғашқыдағыдай бірінен кейін бірі жөнелтіледі. Сұраныстар жөнелтілгеннен кейін байланыс жабылады. Жүктелудің жылдам аяқталатынын ескеріңіз. Бұның екі себебі бар: біріншіден, қосымша байланыс орнатуға уақыт жұмсалмайды. Әр TCP-байланыс үшін қосымша уақыт қажет, кем дегенде, оның расталуына. Екіншіден, суреттердің келуі жылдам орындалады. Неліктен? Бұның барлығы желідегі асыра жүктелуді TCP қадағалағандықтан. Байланыс орнатылғаннан кейін, TCP желілік жолды сараптағанша өткізгіштік қабілеттілікті жоғарылату үшін, бірден баяу старт процедурасын пайдаланады. Осындау «қыздыру» кезеңінің салдарынан көптеген қысқа TCP-байланыстар арқылы ақпарат тасымалдауға, бір ұзақ TCP-байланысты қолдануға қарағанда әлдеқайда көп уақыт

жұмсалады. Соңында, *7.16 б-суретінде* бір тұрақты байланыс орнатылған, ал сұраныстар конвейерлік түрде беріледі. Екінші және үшінші сұраныс шапшаң жөнелтіледі, себебі суреттер жүктеліп, бейнелену керектігі жайлы тұжырымдама жасау үшін, бастапқы парақтың үлкен бөлігі алынған болатын. Нәтижесінде осы сұраныстарға жауап келеді. Бұл тәсіл сервердің жұмыс уақытын қысқартады, сондықтан ол өнімділікті әрі қарай жоғарылатады. Алайда, тұрақты байланыс өтеу керек. Келесі жаңа сұрақ – «байланысты қай кезде жабу керек?». Сервермен байланыс парақ жүктеліп жатқан кезде үзілмеу керек. Ал сонан кейін ше? Тұнынушының серверден тағы бір парақты сұрай тын сілтемеге басу ықтималдығы жоғары. Егер байланыс ашық қалса, онда келесі сұранс бірден жөнелтілуі мүмкін. Алайда, клиенттің жақын арада серверге сұраныс жолдауына еш кепілдік жоқ. Іс жүзінде клиент және сервер біраз уақыт кезеңінде (мысалы, 60 с) өткенше тұрақты байланысты сақтайды. Егер осы уақыт арасында бірде бір сұраныс жөнелтіліп, бірде бір жауап қабылданбаса, немесе тым көп байланыстар ашық болса, онда олардың кейбіреуін жабу керек болады.

Зейінді оқырман, біз әлі де атамаған тағы бір комбинацияның бар екенін байқаған болар. Бір сұранысты бір TCP-байланыс арқылы жөнелтуге болады, бірақ бұл байланыстарды параллель орнату керек. Бұл **параллель байланыстар (parallel connection)** тәсілін браузерлер тұрақты байланыс пайда болғанша кеңінен қолданған болатын. Оның кемшілігі дәл тізбектік байланыстағыдай – қосымша қызметтік операциялар, бірақ өнімділік әлдеқайда жоғары. Бұның себебі, байланысты параллель орнату және санын ұлғайту бірақ уақытты жасырады. Біздің мысалымызда, екі суретті бейнелеу байланысы бір мезгілде орнатылуы мүмкін. Алайда, бір сервермен үлкен көлемдегі TCP-байланыс орнату ең жақсы ой емес. Себебі, TCP әр байланыс үшін асыра жүктелуді жеке қадағалайды. Нәтижесінде байланыстар бір бірімен бәсекеге түседі, бұл дестелердің қосымша жоғалауына және олардың жеке байланыстарға қарағанда агрессивті желі тұтынушысы болуына әкеледі. Тұрақты байланыстар деңгейі жоғары және оларды пайдалану параллель байланысқа қарағанда ұнамды, себебі оларда қажетсіз шығындар және асыра жүктелу мәселесі жоқ.

Тәсілдер

HTTP-дің веб-технологияларда пайдалану үшін арнайы құрастырылғанына қарамастан, ол болашақта объектіге бағытталған қосымшаларда қолдану үшін әдейі әмбебап етіп жасалады. Осы себептен веб-парақтардың әдеттегі сұраныстарына қосымша, **тәсілдер** деп аталатын арнайы операциялар құрастырылған. Олар өздерінің өміршеңдігі үшін SOAP технологиясына қарыздар.

Әр сұраныс бір немесе бірнеше ASCII-мәтін жолынан тұрады және бірінші жолдағы бірінші сөз шақырылатын тәсілдің аты болып келеді. Кіріктірілген тәсілдер *7.12-кестеде* келтірілген. Тәсіл аттары символдар регистрына сезімтал, яғни *GET* тәсілі бар, ал *get* – жоқ.

7.12-кесте. HTTP-сұраныстардың кіріктірілген тәсілдері

Тәсіл	Сипаттамасы
GET	Веб-парақты оқу сұранысы
HEAD	Веб-парақ тақырыбын оқу сұранысы
PUT	Веб-парақты сақтау сұранысы
POST	Ат берілген ресурсқа қосу (мысалы, веб-параққа)
DELETE	Веб-парақты жою
TRACE	Кіріс сұранысты бейнелеу
CONNECT	Болашақта пайдалану үшін қорда сақталған
OPTIONS	Белгілі бір параметрлерді сұрау

GET тәсілі серверден MIME стандартына сәйкес кодталған парақты сұрайды (жалпы жағдайда ол объект деп есептеледі, бірақ іс жүзінде ол жай файл). Серверге келетін сұраныстың үлкен бөлігін нақты *GET* сұранысы құрайды. *GET*-тің әдеттегі формасы:

GET filename HTTP/1.1,

Мұндағы, *filename* сұралған параққа нұсқайды, ал 1.1 – қолданыстағы хаттама версиясы.

HEAD тәсілі парақсыз тек мәлімдеме тақырыбын сұрайды. Бұл тәсіл көмегімен индекстік ақпаратты немесе жай осы URL-дің жұмыс қабілеттілігін тексеруге болады.

POST тәсілі форма расталған кезде пайдаланылады. Ол да *GET* тәсілі тәрізді SOAP веб-қызметтері үшін қолданылады. Онда да URL сақталады, бірақ жай ғана парақты табудың орнына ол деректерді серверге жөнелтеді (яғни форма мазмұнын немесе RPC параметрлерін). Сонан кейін сервер URL-ге байланысты деректерді өңдейді, әдетте оны объектіге бекітеді. Нәтижесінде, мысалы, бір зат сатылуы немесе процедура шықырылуы мүмкін. Соңында, тәсіл алынған нәтижесімен парақты қайтарады.

Қалған тәсілдер желілік ресурстарды қарау үшін сирек қолданылады. *PUT* тәсілі *GET* тәсіліне кері жұмыс істейді: ол парақты оқымайды, жазады. Бұл тәсіл қашықтықтағы серверде веб-парақтар жиынтығын құрастыруға мүмкіндік береді. Сұраныс денесінде парақ көрсетіледі. Ол MIME көмегімен кодталуы мүмкін. Бұл жағдайда *PUT* командасынан кейінгі жолға түрлі тақырыптар, мысалы, сұралған операция бойынша абонент құқығын растаушы аутентификациялау тақырыбы қосылуы мүмкін.

DELETE тәсілі парақты жояды немесе веб-серверге парақты жою керектігін нұсқайды. *PUT* тәсіліндегідей мұнда аутентификация және осы операцияны орындауға рұқсат өте маңызды.

TRACE тәсілі реттеуге негізделген. Ол серверге сұранысты кері жөнелтуді бұйырады. Бұл тәсіл әсіресе сұраныстар дұрыс өңделмейтін және клиент

сервердің нақты қандай сұранысты алғандығын білгісі келетін жағдайда пайдалы.

CONNECT тәсілі тұтынушыға веб-кэш тәрізді дәнекер-құрылғы арқылы серверге қосылуға мүмкіндік береді.

OPTIONS тәсілі клиентке серверден парақты сұрап, ол жерде қолдануға болатын тәсілдер және тақырыптар алуға мүмкіндік береді.

Серверден әр сұранысқа, қалып-күй жолы, сонымен бірге қосымша ақпарат (мысалы, веб-парақ немесе оның бөлігі) бар жауап келеді. Қалып-күй жолы сұраныстың сәтті орындалғандығы немесе сәтсіздік себебі көрсетілген үш разрядты қалып-күй кодынан тұруы мүмкін. Бірінші разряд барлық жауаптарды *7.13-кестеде* көрсетілгендей, бес негізгі топқа бөлуге арналған. Бірден (1xx) басталатын кодтар іс жүзінде сирек қолданылады. 2-ден басталатын кодтар сұраныстың сәтті өңделгендігін және деректердің (егер сұралса) жөнелтілгендігін білдіреді. 3xx кодтары клиентке бағын басқа жерде немесе басқа URL-ді сынауын немесе өзінің жекеменшік кэшін пайдалануын хабарлайды (төменде талқылаймыз).

7.13-кесте. Сервер жауабындағы қалып-күй кодтарының тобы

Код	Мәні	Мысалы
1xx	Ақпарат	100 = сервер клиент сұраныстарын өңдеуге келісті
2xx	Сәттілік	200 = сұраныс сәтті өңделді; 204 = мазмұны жоқ
3xx	Жол көрсету	301= парақ ауыстырылған; 304 = кэштелгген парақ қолжетімсіз
4xx	Клиент қателігі	403 = қолжеткізу қателігі; 404 = парақ табылмады
5xx	Сервер қателігі	500 = сервердің ішкі қателігі; 503 = кейінірек тағы бір талпынып көріңіз

Төрттен басталатын кодтар сұраныстың клиентпен байланысты қандай да бір себептермен сәтсіздікке үшырағанын білдіреді: мысалы, жоқ парақ сұралды немесе сұраныс қате жазылған. Соңында, 5xx коды, програма қателегі немесе уақытша асыра жүктелу салдарынан орын алған сервердің ішкі қателіктері жайлы хабарлайды.

Мәлімдемелер тақырыбы

Сұраныс жолынан кейін (мысалы, *GET* тәсілінің аты жазылған) қосымша ақпараты бар басқа жолдар болуы мүмкін. Олар **сұраныстар тақырыбы (request headers)** деп аталады. Бұл ақпаратты, процедура шақырылғанда ұсынылатын параметрлермен салыстыруға болады. Өз кезегінде жауаптарда **жауап тақырыптары (response headers)** болуы мүмкін. Кейбір тақырыптар екеуінде де кездесуі мүмкін. Олардың ішіндегі маңыздылары *7.14-кестеде* келтірілген. Бұл тізім өте ұзын, өздеріңіз түсінетіндей әр сұранысқа және жауапқа тақырыптар жиынтығы сәйкес келуі мүмкін.

7.14-кесте. HTTP хаттамасы мәлімдемелерінің кейбір тақырыптары

Тақырып	Типі	Мазмұны
User-Agent	Сұраныс	Браузер және оның платформасы жайлы ақпарат
Accept	Сұраныс	Клиент қолдайтын парақ типі
Accept-Charset	Сұраныс	Клиент қолдайтын символдар жиынтығы
Accept-Encoding	Сұраныс	Клиент қолдайтын кодтау типтері
Accept-Language	Сұраныс	Клиент түсінетін табиғи тілдер
If-Modified-Since	Сұраныс	Соңғы жаңарту датасы және уақыты
If-None-Match	Сұраныс	Жаңартудан кейін жөнелтілген тәгтер
Host	Сұраныс	Сервердің DNS-аты
Authorization	Сұраныс	Клиенттің жеке меншік идентификаторлар тізімі
Referer	Сұраныс	Алдыңғы сұраныс келген URL
Cookie	Сұраныс	Ертеректе алынған cookie-файлды серверге жөнелтті
Set-Cookie	Жауап	Сервер клиенттің cookie сақтағанын қалайды
Server	Жауап	Сервер жайлы ақпарат
Content-Encoding	Жауап	Мазмұнды кодтау типі (мысалы, gzip)
Content-Language	Жауап	Парақта қолданылатын табиғи тіл
Content-Length	Жауап	Парақтың байтпен берілген көлемі
Content-Type	Жауап	MIME парақ типі
Content-Range	Жауап	Парақ контентінің бөлігін сәйкестендіреді
Last-Modified	Жауап	Параққа енгізілген соңғы жаңартулар датасы және уақыты
Expires	Жауап	Парақ жарамсыз болып саналатын дата және уақыт
Location	Жауап	Клиент сұранысын басқа адреске қайта жөнелту кезінде клиентке берілетін команда
Accept-Ranges	Жауап	Сервер берілген көлемдегі парақтарға сұраныс қабылдауға дайын
Date	Сұраныс / Жауап	Мәлімдеме жөнелтілген дата және уақыт
Range	Сұраныс / Жауап	Парақ бөлігін сәйкестендіру
Cache-Control	Сұраныс / Жауап	Кэшті қалай өңдеу керектігі жайлы нұсқаулар
ETag	Сұраныс / Жауап	Парақ контентіне арналған тәг
Upgrade	Сұраныс / Жауап	Жөнелтуші ауысқысы келетін хаттама

User-Agent тақырыбы клиентке серверді өз браузерінің версиясы (мысалы, *Mozilla/5.0* және *Chrome/5.375.125*) жайлы хабардар етуіне мүмкіндік береді. Бұл ақпарат серверге өз жауаптарын нақты браузерге сәйкестендіруге мүмкіндік береді, себебі әртүрлі браузерлердің мүмкіндіктері мен жұмыс тәртібі әртүрлі.

Accept басталатын төрт тақырып серверге, клиент қабылдауға дайын (егер олардың жиынтығы шектелген болса) ақпарат типі жайлы хабарлайды. Кестеде келтірілген бірінші тақырып, клиентпен дұрыс қабылданатын MIME типін анықтайды (мысалы, *text/html*). *Accept-Charset* тақырыбы клиенттің қандай символдар жиынтығын көргісі келетіні жайлы хабарлайды (мысалы, ISO-8859 немесе Unicode-1-1). *Accept-Encoding* тақырыбында тығыздау тәсілдерінің мәселесі жайлы сөз қозғалады (мысалы, gzip). Соңында, *Accept-Language* клиенттің құжаттарды қандай тілде оқуға (мысалы, испан) дайын екендігін хабарлайды. Егер сервердің бірнеше парақтың біреуін таңдау мүмкіндігі бар болса, онда ол алынған ақпараттарға сүйеніп, олардың ішіндегі ең жақсысын таңдап алады. Егер сұранысты қанағаттандыру мүмкін болмаса, онда қателік кодын қайтарады және сұраныс сәтсіз болып саналады.

If-Modified-Since және *If-None-Match* тақырыптарын кэш пайдаланады. Олар клиентке парақты тек кэште қолжетімді көшірме жоқ болған кезде ғана қайта жөнелтуді сұрауға мүмкіндік береді. Кэштеу жайлы біз кейінірек айтамыз.

Host тақырыбы сервер жайлы сөз қозғайды. Оның мәні URL-ден алынады. Бұл тақырып міндетті. Неліктен? Себебі кейбір IP-адрестерге бірнеше DNS-аттар қызмет көрсетуі мүмкін, сондықтан серверге сұранысты кімге жөнелту керектігін қалай да ажырату керек.

Authorization тақырыбы қорғалған парақ сұралған кезде қажет. Клиент көмегімен сұралған парақты қарауға құқығыңыздың бар екенін растауға болады.

Клиент дұрыс жазылмаған *Referer* тақырыбын сұралған парақпен байланысты URL-ді беру үшін пайдаланады. Көбіне, бұл алдыңғы парақ URL.

Бұл тақырып бір парақтан бір параққа көшкен кезде өте пайдалы, себебі серверге клиенттің белгілі бір параққа қандай жолмен келгенін анықтай алады.

Cookie-дің RFC 2616-да емес RFC 2109 сипатталғанына қарамастан оларды сипаттау үшін де тақырыптар бар. *Set-Cookie* тақырыбы сервердің cookie файлды клиенттерге қалай жөнелтетіндігін анықтайды. Егер бұл тақырыпты сервер орнатса, онда клиент оны көріп cookie-ді өзінде сақтап, келесі сұраныста *Cookie* тақырыбының көмегімен серверге қайтарады деп жорамалданады (RFC 2965-те cookie файлдардың жаңартылған тақырыптары мен кейіннен шыққан спецификациясының бар екендігіне назар аударыңыз. Бірақ олар кең таралмаған).

Жауаптарда көптеген басқа тақырыптар пайдаланылады. *Server* тақырыбы серверге өз программалық жабдықтамасының версиясын сипаттауға мүмкіндік береді. Келесі бес тақырып *Content* сөзінен басталады және серверге жөнелтілетін парақтың қасиеттерін сипаттауға мүмкіндік береді.

Last-Modified тақырыбында жөнелтілетін параққа соңғы жаңартулар енгізілген дата және уақыт жазылады. Ал *Expires* тақырыбы парақты неше уақыт қолжетімді болатындығын хабарлайды. Парақты кэштеуде бұлардың екеуінің де маңызы зор.

Location тақырыбын сервер клиент өз сұранысын басқа URL-ге қайталап жіберу керектігін хабарлау үшін енгізеді. Мұндай жағдай парақ басқа адреске «көшкенде» немесе бірнеше URL бір параққа нұсқаған жағдайда туындайды (мүмкін басқа сервер орналасқан парақ айнасына). Бұл амалды көбіне, негізгі парақтары com доменінде орналасқан компаниялар пайдаланады, алайда, клиенттер ол жерден клиент IP-адресіне немесе клиент таңдаған тілге сүйене отырып, ұлттық немесе аймақтық парақтарға қайта жөнелтіледі.

Егер парақтың көлемі тым үлкен болса, клиент оны бірден толығымен қабылдағысы келмеуі мүмкін. Кейбір серверлер бір жөнелткендегі парақ көлеміне шектеу қоятын сұраныстарды қабылдауы мүмкін. Егер парақ көлемі тым үлкен болса, онда ол бірнеше кіші бөліктерге бөлініп жөнелтіледі. *Accept-Ranges* тақырыбы сервер парақтың осындай бөліктеріне сұранысты қолдайтындығын хабарлайды.

Енді екі бағытта да пайдаланылатын тақырыптарға көшеміз. *Date* тақырыбы сұраныста да, жауапта да қолданылады. Онда мәлімдеменің жөнелтілген датасы және уақыты көрсетіледі, ал *Range* тақырыбында парақ көлемі байтпен көрсетіледі және ол жауап ретінде жөнелтіледі.

Etag тақырыбы парақ контентінің атын білдіретін қысқа тәгті береді. *Cache-Control* тақырыбы парақты қалай кэштеу керектігі жайлы нақты нұсқаулар береді (жиі, қалай кэштемеу керектігін).

Upgrade тақырыбы келесі HTTP хаттамасы тәрізді жаңа коммуникация хаттамасына көшу немесе деректерді қорғалған түрде тасымалдау үшін пайдаланылады. Ол клиентке қандай нұсқаларды қолдайтынын, ал серверге қандай нұсқамен жұмыс істейтінін хабарлауға мүмкіндік береді.

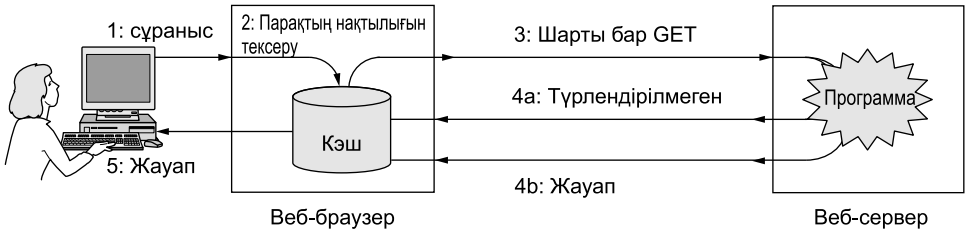
Кэштеу

Біз ертеректе қараған парақтарымызға жиі қайтып ораламыз, ал байланысқан веб-парақтарда көбіне бір ресурстар орналасады. Мысал ретінде сайтпен жылжу үшін қолданылатын суреттерді, сонымен бірге скрипттер және стильдердің стандартты кестесін келтіруге болады. Парақтың бұл ресурстардың барлығын бейнелеген сайын қайталап алған үнемсіз болар еді, себебі браузерде олардың көшірмесі бар.

Алынған парақты әрі қарай пайдалану үшін сақтау **кэштеу (caching)** деп аталады. Бұл тәсілдің артықшылығы – кэште сақталған парақты қайталап пайдалануға болады және мұнда тасымалдауды қайталау қажет емес. HTTP-дің клиенттерге парақты еш қауіпсіз қайта пайдалануын білуге көмектесетін, кіріктірілген кэштеуді қолдауы бар. Бұл қолдау трафик және күту уақытын азайтып, өнімділікті жоғарылатады. Бұның құны – браузер парақтарды сақтауы керек, ол өзін ақтайды, себебі ақпаратты жергілікті сақтау ешқандай күрделі шығынды қажет етпейді. Парақ әдетте, дискіде сақталады, сондықтан оларды браузер келесі жолы іске қосылғанда пайдалануға болады.

HTTP кэштеуіндегі даулы сауал – сақталған парақтың келесі шақырылғандағы түріне сәйкес келетіндігі. Мұндай тұжырымдаманы тек URL сәйкестігінен жасауға болмайды. Мысалы, URL соңғы жаңалықтар бейнеленген парақты беруі мүмкін. Бұл парақтың URL өзгеріссіз қалғанымен, контенті жиі өзгеріп отырады. Екінші жағынан, парақта грек құдайлары және Рим мифологиясының тізімі болуы мүмкін. Бұл парақ соншалықты жиі жаңартылмайды.

HTTP бұл мәселені шешудің екі стратегиясын пайдаланады. Олар *7.17-суретте* сұраныс (1-қадам) және жауап (5-қадам) арасындағы өңдеу формасы түрінде бейнеленген. Бірінші стратегия – парақтың соңғы нұсқа екенін тексеру (2-қадам). Кэш сұралады, онда берілген URL парағының көшірмесінің жаңартылған версия бар (демек, соңғы нұсқа), серверден жаңа көшірмені алудың қажеті жоқ. Кэште сақталған парақ бастапқыда серверден алынған кезде, онымен бірге *Expires* тақырыбы қайтарылады. Оның мазмұны ағымдағы дата және уақытпен бірге парақтың соңғы нұсқа екенін анықтауға көмектеседі.



7.17-сурет. HTTP кэштеуі

Алайда, парақтардың барлығы, оның қай кезде қайта алынуы керек екенін көрсететін, ыңғайлы *Expires* тақырыбымен келе бермейді. Соңында, болжау жасау, әсіресе, болашақты болжау өте қиын. Бұл жағдайда браузер эвристикалық ережені пайдалануы мүмкін. Мысалы, егер парақ соңғы жылда өзгермесе (*Last-Modified* тақырыбында көрсетілгендей), онда оны соңғы бір сағат ішінде өзгере қоймайды деп болжауға болады. Дегенмен, бұған кепілдік жоқ, бұл ой ақталмауы мүмкін. Мысалы, биржа түнде жабылса, оның парағы бірнеше сағат жаңартылмайды, алайда, келесі сауда сессиясы басталысымен өзгертулер тұрақты енгізіліп отырады. Сөйтіп, кэштің ақталуы уақыт кезеңіне байланысты өзгеріп отырады. Осы себептен эвристикалық ережені абайлап қолдану керек, осылай бола тұра олар жиі жақсы жұмыс істейді.

Белгілі бір уақыт аралығында өзгерген парақты іздеу – кэшті пайдаланудың ең пайдалы әдісі, себебі бұл жағдайда сервермен байланысу қажет емес. Өкінішке орай, ол әркез жұмыс істемейді. Сервер *Expires* тақырыбын абайлап падалануы керек, себебі параққа қай кезде өзгеріс енгізілгені белгісіз болуы мүмкін. Сонымен, кэште сақталған көшірмелер жаңа болуы мүмкін, бірақ клиент оны біле алмайды.

Бұл жағдайда екінші стратегия пайдаланылады. Ол – серверден сақталған парақ көшірмесінің соңғы нұсқа екені жайлы ақпарат сұрау. Бұл сұраныс

conditional GET (шарты бар GET) деп белгіленеді. Ол *7.17-суретте* 3-қадам ретінде көрсетілген. Егер сервер кэште сақталған көшірме соңғы нұсқа екенін білсе, оны растап қысқа жауап қайтарады (4а-қадамы). Кері жағдайда ол толық жауап қайтаруы тиіс (4ә-қадамы).

Көшірменің осы сәтте жарамды екенін анықтайтын тағы бірнеше тақырыптар өрісі бар. Клиент *Last-Modified* тақырыбының арқасында параққа соңғы рет өзгерістің қай кезде енгізгенін біледі. Ол осы уақытты серверге *If-Modified-Since* тақырыбын пайдаланып жөнелтіп, парақты соңғы кэштелуден кейін өзгерген жағдайда ғана жіберуін сұрай алады. Басқа жағдайда сервер парақты *ETag* тақырыбымен бірге қайтарады. Ол бақылау қосындысына ұқсас, парақ контентінің қысқа аты болып саналатын тәгті жөнелтеді, тек бұл одан да жақсырақ. (Бұл, біз *8-тарауда* қарастыратын криптографиялық хэш болуы мүмкін). Клиент сақталған көшірменің соңғы нұсқа екенін серверге, сақталған көшірмелер тәгі жазылған, *If-None-Match* тақырыбын жөнелтіп біле алады. Егер кез келген тәг сервер жауап ретінде жөнелту керек болған контентпен сәйкес келсе, кэште сақталған сәйкес көшірмені пайдалануға болады. Бұл тәсілді парақ жаңартылғанда және оны пайдалану ыңғайлы немесе қажет емес екенін анықтауда қолдануға болады. Мысалы, сервер қандай тілдің және MIME типінің ұнамды екеніне байланысты, бір URL үшін әртүрлі контент жіберуі мүмкін. Бұл жағдайда тек модификация датасы кэште сақталған көшірменің жаңа екенін анықтауға көмектеспейді. Соңында, осы екі кэштеу стратегиясын *Cache-Control* тақырыбының директивасымен жоққа шығарылуы мүмкін екеніне назар аударыңыз. Оларды қажеттілік жоқ кезде кэштеуді шектеу үшін пайдаланылады (мысалы, кэштемеу). Мысал ретінде әр алған сайын өзгеріп отыратын динамикалық парақтарды келтіруге болады. Авторизациялауды талап ететін парақтарды да кэштеу қажет емес.

Кэштеу жайлы әлі де көп айтуға болады, бірақ біз көлеммен шектеулеміз, сондықтан тек екі маңызды жағдайды айта кетеміз. Біріншіден, кэштеу тек браузерде ғана қолданылмайды. Әдетте, HTTP-сұраныстар бірнеше кэштер арқылы берілуі мүмкін. Браузерге қатысты емес кэштеуді пайдалану прокси-кэштеу (proxy caching) деп аталады. Кэштеудің әрбір жаңа деңгейі тізбекпен жоғары берілетін сұраныстар санын азайтуы мүмкін. Әртүрлі тұтынушылар сұрайтын парақтарды кэштеуден прокси-кэштеуді іске қосу арқылы пайда көру – провайдерлер мен ұйымдардың әдеттегі ісі. Біз прокси-кэштеуді толығырақ *7.5-бөлімде* қарастырамыз. Екіншіден, кэштеу өнімділікті едәуір жоғарылатады, бірақ көпшілік күткендей емес. Себеп – танымал және танымал емес веб-ресурстар бар, олардың көбі тым ұзын (мысалы, бейне). Сирек кіретін веб-ресурстардың «ұзын құйрығы» кэште орын алады, ал кэш-жады өңдейтін сұраныстар саны көлем өсуімен бірге өспейді. Веб-кэштің сұраныстардың жартысынан көбін өңдеу ықтималдығы жоғары емес (Breslau және басқалар, 1999).

HTTP-ді пайдалану мысалдары

HTTP мәтіндік хаттама болғандықтан, сервермен терминал арқылы әрекеттесуді (браузерге қарама-қарсы) жеңіл ұйымдастыруға болады. Тек сервердің 80-портымен TCP-байланыс орнату қажет. Оқырманға келесі сценариймен тәжірибе жүргізуге мүмкіндік туып отыр. Ол көптеген UNIX қабықшаларында және Windows командалық терезесінде (telnet программасы іске қосылған кезде) жұмыс істейді. Сонымен командалар тізбегі:

```
telnet www.ietf.org 80 > log
GET/rfc.html HTTP/1.1
Host: www.ietf.org
```

Бұл командалар тізбегі, *www.ietf.org* адресі бойынша орналасқан, IETF веб-серверінің 80-портымен telnet-байланыс (демек, TCP-байланыс) орнатады. Әрі қарай *GET* командасы орналасқан. URL жолы және тасымалдау хаттамасы көрсетілген. Онан кейін *Host* тақырыбы бар міндетті жол орналасқан. Ол серверге сұраныстар тақырыбының аяқталғанын хабарлайды. Сервер аттарын және URL-ді өзгерте отырып, түрлі тақырыптар және парақтар түрін көруге болады.

7.3.5. Мобильді веб

Дүниежүзілік өрмек көптеген компьютерлерде, сонымен бірге мобильді телефондарда пайдаланылады. Желілік ресурстарды мобильді байланыс құрылғысы көмегімен көру өте пайдалы болуы мүмкін. Алайда, онымен бірнеше техникалық мәселелер туындайды, себебі контенттердің көп бөлігі дербес компьютерлерге және кең байланыс арналарына арналып құрастырылған болатын. Бұл бөлімде біз мобильді құрылғылар немесе **мобильді веб (mobile Web)** арқылы Дүниежүзілік өрмекке қолжеткізу қалай жүзеге асырылатынын сипаттаймыз.

Жұмыстағы және үйдегі стол үсті компьютерлеріне қарағанда мобильді телефонды желілік ресурстарды қарау үшін пайдалану біршама қиын. Оның себептері:

1. Парақтар және үлкен суреттер бейнелену тиіс экранның кішілігі;
2. URL және басқа да ұзын жолдарды ыңғайлы енгізуге арналған енгізуге арналған енгізу ақпарат мүмкіндігі шектеулілігі;
3. Арнаның өткізгіштік жолағы радио арна мүмкіндіктерімен шектеулі, мысалы 3G және бұл тым қымбат;
4. Тұрақсыз жұмыс істеуі мүмкін байланыс;
5. Көлемі, бағасы және аккумулятор сыйымдылығына байланысты шектеулі есептеу мүмкіндіктері.

Осы қиындықтар, үстел үсті компьютеріне арналған қарапайым контентті мобильді құрылғылармен пайдалану жағымсыз тәжірибеге әкелуі мүмкін.

Мәселені шешудің бірінші жолы, мүмкіндіктері шектеулі сымсыз құрылғылар үшін құрастырылған жаңа хаттамаларды құрастыруға негізделген. **WAP (Wireless Application Protocol – сымсыз қолжеткізу хаттамасы)** осы стратегияның мысалы ретінде кеңінен танымал. WAP-ты құрастыру талпыныстарын 1997 жылы, Nokia, Ericsson және Motorola тәрізді ірі мобильді құрылғылар өндірушілері бастады. Алайда, оларды тосын сыйлар күтіп тұрды. Келесі 10 жыл ішінде желінің өткізгіштік қабілеттілігі және мобильдің құрылғылардың мүмкіндіктері он есесге өсіп кетті. 3G форматындағы дерек-тері бар қызметтер және үлкен түрлі түсті экрандары, жылдам процессорлары бар мобьильді телефондар және сымсыз 802.11 желісімен жұмыс істеу мүмкіндіктері пайда болды. Кенеттен мобильді құрылғылар әдеттегі веб-браузерді пайдалану мүмкіндігіне ие болды. Телефон және үстел үсті компьютерлерінің арасындағы айырмашылық ешуақытта да жойылмайды, әлі де болады, бірақ әртүрлі хаттамаларды құрастыру негізінде жатқан техникалық мәселелер шешіледі.

Сайттар мобильді телефон экранында бейнелеуге ыңғайлы контент ұсыну үшін, дербес компьютер және мобильді құрылғыларда бір хаттаманы пайдалану идеялары кеңінен танымал болып келеді. Веб-серверлер сұраныс тақырыбына қарап, веб-парақтың қандай версиясын телефон немесе компьютер үшін ұсыну керектігін анықтай алады. Бұл жағдайда *User-Agent* тақырыбы өте пайдалы, себебі ол браузер версиясын көрсетеді. Осылайша, веб-сервер сұранысты алған кезде тақырыпқа қарап iPhone үшін суреттері кіші, мәтіні аз, навигациясы қарапайым және ноутбук үшін қасиеттер жиынтығы толық веб-парақты қайтарады.

W3C бұл көзқарасты біраз қолдайды. Мысалы, ол мобильді веб-контент үшін ең жақсы іс-тәжірибені стандарттайды. Осындай 60 іс-тәжірибеден тұратын тізім бірінші спецификацияда берілген (Рабин және МакКетиНевил, 2008). Бұл іс-тәжірибенің көбі кэштеу тиімділігін ұлғайту және парақ көлемін кішірейтуге үлкен үлес қосты, соның ішінде тығыздау арқылы, себебі коммуникация есептеуден қымбатқа түседі. Осы әдістің арқасында сайт иелері әсіресе, үлкен сайт, контенттің мобильді нұсқасын құрастырады, себебі ол желіге телефон арқылы кіретіндер үшін өте қажет. Осындай тұтынушыларға көмектесу үшін парақтарды белгілеу енгізілген, оларды мобильді телефоннан қарауға (сәтті) болады.

Тағы бір пайдалы құрал – **XHTML Basic** деп аталатын HTML-дың қысқартылған версиясы. Бұл тіл XHTML деп аталады және мобильді телефон, теледидар, қалта компьютері, сауда автотматтары, пейджер, автомашина, ойын автоматтары, тіпті, сағаттарда пайдалануға арналған. Осы себептен бұндай версия стильдер кестесін, скрипттерді немесе фреймдерді қолдамайды, бірақ стандартты тәгттердің көбі онда бар. Олар 11 модульге топтастырылған. Кейбіреулері қажетті, кейбіреулері опция. Олардың барлығы XML-де анықталған. Осы модульдер және тәгтер мысалы *7.15-кестеде* келтірілген.

7.15-кесте. XHTML Basic модульдері және тәгтері

Модуль	Қажетті ме?	Функциясы	Тәгтер мысалы
Structure (Құрылым)	Иә	Құжат құрылымы	body, head, html, title
Text (Мәтін)	Иә	Ақпарат	br, code, dfn, em, hn, kbd, p, strong
Hypertext (Гипермәтін)	Иә	Гиперсілтеме	a
List (Тізімдер)	Иә	Элементтер тізімі	dl, dt, dd, ol, ul, li
Forms (Формалар)	Жоқ	Форманы толтыру	form, input, label, option, textarea
Tables (Кестелер)	Жоқ	Тікбұрышты кестелер	caption, table, td, th, tr
Image (Суреттер)	Жоқ	Суретті орналастыру	img
Object (Объектілер)	Жоқ	Апплеттер, карталар және т.б.	object, param
Meta-Information (Метаақпарат)	Жоқ	Қосымша ақпарат	meta
Link (Сілтеме)	Жоқ	<a> ұқсас	link
Base (База)	Жоқ	URL санау нүктесі	base

Алайда, парақтардың барлығы мобильді құрылғыларда жақсы жұмыс істейтіндей жобаланбайды. Сол себепті, қосымша амал – **контентті түрлендіру (content transformation)** немесе **кодын өзгерту (transcoding)**. Бұл, компьютер мен сервер арасындағы байланысты қамтамасыз ететін компьютер конттентті алып, оны мобильді құрылғыға жарамды болатындай етіп түрлендіреді дегенді білдіреді. Қарапайым түрлендіру – үлкен суреттердің көлемін қысқарту үшін төмен рұқсат қабілеттілікке түрлендіру. Бұдан басқа да қарапайым және пайдалы түрлендірулер болуы мүмкін. Кодты өзгерту мобильді вебтің пайда болуымен белгілі бір сәттілікпен қолданыла бастады (мысалы, Fox және басқалар (1996) басылымын қараңыз). Алайда, екі түрлендіруді қолданғанда мобильді контент және сервер мен кодты өзгертуші шешімдерінің арасында кейбір келіспеушілік болады. Мысалы, веб-сайт мобильді Интернет тұтынушы үшін, транскодер сурет форматын өзгерте алатындай сурет және мәтіннің белгілі бір комбинациясын ғана таңдап алуы мүмкін.

Біз әзірше хаттамалар емес, контент жайлы сөз қозғадық, себебі мобильді вебті жүзеге асыру үшін ең көп қиындық тудыратын контент. Хаттамаларды да қысқаша еске сала кетейік. Желі пайдаланатын HTTP, TCP және IP хаттамалары бейнеленбейтін ақпараттарды тасымалдай отырып, арнаның едә-

үір бөлігін, мысалы, тақырыпты толтыруы мүмкін. Бұл мәселені шешу үшін WAP және басқа да тәсілдерде арнайы мақсаттағы хаттамалар анықталған. Көбіне олардың қажеті жоқ. Осы кіттаптың *6-тарауында* сипатталған ROHC (Robust Header Compression) тәрізді тақырыпттарды тығыздау технологиясы осындай хаттамалардағы жанама ақпараттар көлемін қысқарта алады. Сөйтіп, өткізгіштік қабілеттілігі жоғары және төмен байланыстар үшін бір хаттамалар жиынтығын (HTTP, TCP, IP) пайдалануға болады. Өткізгіштік қабілеттілігі төмен байланысты қолдану үшін тек тақырыпты тығыздау механизмін қосу жеткілікті.

7.3.6. Веб-іздеу

Дүниежүзілік өрмек жайлы әңгімемізді аяқтамас бұрын, көптеген адамдардың ойлары бойынша ең сәтті веб-қосымша – веб-іздеу жайлы айта кетейік. 1998 жылы Стенфорд студенттері Сергей Брин және Лари Пейдж ең жақсы іздеу маханизмін жазу үшін «Google» атты компанияның негізін қалады. Олар кейіннен өзін өте сәтті етіп көрсеткен идеямен қаруланған еді. Идея – парақтағы кілтті сөздер саны емес, басқа парақтардағы сілтеменің параққа неше рет көрсеткені, парақ маңыздылығының ең жақсы көрсеткіші екенін бағалайтын алгоритм. Мысалы, көптеген парақтар Cisco басты парағына сілтейді, бұл «Cisco» сөзін кілтті сөз ретінде енгізген тұтынушы үшін сөзді көп пайдаланатын, бірақ компанияға еш қатысы жоқ параққа қарағанда әлдеқайда маңызды.

Олардың ойлары дұрыс болып шықты. Мұндай әдіс шын мәнінде, жақсы іздеу механизмін жазуға көмектесті және тұтынушылар оны бағалады. Венчурлық капиталмен қолданған Google компаниясы шексіз өсті. 2004 жылы компания акционерлік және оның құны $23 млрд. болды. 2010 жылға дейін ол бүкіл әлем бойынша, деректерді өңдеу және сақтау орталықтарында миллионға жуық серверлерді қамтыды.

Белгілі бер дәрежеде, іздеу – қосымшалардың бірі және ескі қосымша, себебі ол Дүниежүзілік өрмек пайда болғаннан бері дамып келеді. Алайда, веб-іздеу өте қажетті, себебі Күніне миллиардтан астам сұраныс өңделеді. Әртүрлі ақпаратты іздеуші адамдар іздеуді бастапқы нүкте ретінде пайдаланады. Мысалы, Сиэтлде вегемитті сатып алу үшін қандай парақты қарау керектігін білмейміз, бірақ іздеу жүйесі қажетті ақпарат жазылған парақты біледі және оған жылдам бағыттайды деген үміт бар.

Дәстүрлі түрде веб-іздеуді жүзеге асыру үшін тұтынушы өз браузерінде веб-іздеу сайтын ашады. Негізгі іздеу сайттары – Google, Yahoo! және Bing. Сонан кейін тұтынушы форманы пайдаланып, қажет сұранысты жөнелтеді. Іздеу жүйесі релевантты парақты, суретті және т.б. тауып, оны динамикалық парақ түрінде қайтару үшін, бұл сұранысты деректер базасына жөнелтеді. Бұдан соң тұтынушы табылған сілтемелер арқылы көше алады.

Веб-іздеу – желіні ұйымдастыруға және пайдалануға әсер ететін, талқылауға тұрарлық, қызықты тақырып. Біріншіден, веб-іздеудің парақты қалай табатыны қызықты. Веб-іздеу жүйесінің сұранысты өңдеу үшін парақтардан тұратын деректер базасы болуы керек. Әр HTML-парақта басқа парақтарға сілтеме болуы мүмкін, сондықтан барлық қызықты нәрсеге (немесе, кем дегенде іздеу сұраныстарында жиі кедесетін) әйтеуір бір жерде сілтеме бар. Демек, теориялық тұрғыдан, кішігірім парақтар жиынтығынан бастап, сілтемелер арқылы көше отырып, вебте орналасқан барлық парақтарды қарап шығуға болады. Бұл үдеріс **веб-кроулинг (Web crawling)** немесе парақтарды қарап өту деп аталады. Оны барлық іздеу жүйелері пайдаланады.

Веб-кроулингпен байланысты бір қиындық – барлық парақ түрлерін табу мүмкін емес. Статикалық құжаттарды алып, сілтеме арқылы көшу – бұл жеңіл. Алайда, көптеген парақтар тұтынушының жөнелткен ақпаратына байланысты әртүрлі парақ беретін программадан тұрады. Мысал ретінде онлайн-дүкен каталогын алуға болады. Каталогта түрлі тауарға сұраныс бойынша, азықтүлік деректер базасынан құрастырылған динамикалық веб-парақтар болуы мүмкін. Контенттің бұл типі жеңіл табуға болатын статикалық парақтардан өзгеше. Осындай динамикалық парақтарды веб-іздеу жүйелері қалай табады? Жауап қарапайым: көбіне, таппайды. Бұндай жасырын контент типі **терең Дүниежүзілік өрмек (deep Web)** деп аталады. Бұндай контентті қалай іздеу керек – қазіргі кезде ғалымдар шешімін таппақ болып, жұмыс істеп жатқан, ашық мәселе (Мадхаван және басқалар (2008) басылымын қараңыз). Сонымен бірге, сайттар, іздеу жүйелеріне сайттың қандай бөлігін қарауға және қандай бөлігін қарауға болмайтыны жайлы хабарлайтын парақтар құрастыру (robots. txt деген атпен танымал) керек деген келісім бар.

Екінші күрделі сұрақ – қиылысқан гиперсілтемесі бар деректерді қалай өңдеу керек. Индекстеу алгоритмі деректер жиынымен жұмыс істеу үшін, парақтар бір жерде сақталуы тиіс. Бағалу нәтижесі әртүрлі, бірақ негізгі іздеу жүйелерінде Дүниежүзілік өрмектің көрінетін бөлігінен жинақталған миллиардтаған парақтар каталогы бар. Парақтың орташа көлемі 320 Кбайт деп бағаланады. Демек, Дүниежүзілік өрмектің іздеу жүйелері кірген бөлігі 20 петабайт немесе 2×10^{16} байт. Бұл сан тым үлкен болса да, Интернет деректер өңдеу орталықтарында осынша деректер көлемі өңделіп, сақталады (Чанг және басқалар, 2006). Мысалы, егер ақпаратты дискіге сақтау терабайт үшін 20 доллар болса, 2×10^4 Тбайт $400 000 тұрады. Бұл Google, Microsoft және Yahoo! тәрізді компаниялар үшін көп емес. Сонымен бірге желі кеңіген сайын дискі бағасы төмендеп келеді, сондықтан жақын болашақта Дүниежүзілік өрмекті сақтау үлкен компаниялар шеңберінде шешілетін мәселе.

Осы деректерден пайдалы ақпаратты табу – басқа маңызды мәселе. Сіздер XML-дың программаларға деректер құрылымын жеңіл шығарып алуға көмектесетінін бағалай аласыздар, ал арнайы форматтар шешілмеген мәселелерді қосады. Форматтар конверсиясы, тіпті, бір тілден екінші тілге ауыстыру да маңызды сұрақ. Деректер құрылымын анықтаудың өзі де

мәселенің тек жартысы, ал оның өзі – осы деректер құрылымның астарында не жатқанын түсіну. Осы мәселені шешу – релевантты парақтарды тауып, іздеу сұраныстарына жауап беруде үлкен пайда әкелер еді. Басты мақсат – сұрақтарға жауап алу мүмкіндігі, мысалы, біздің қалада арзан және сапалы тостерді қай жерде сатып алуға болады?

Веб-іздеудің үшінші аспектін – ат берудің жоғары деңгейін құрастыру. Сізге ұзын сонар URL-ді есте сақтаудың қажеті жоқ, егер веб-парақты дәл сондай сәттілікпен (немесе одан да жоғары) адам аты бойынша табуға болатын болса. Біз атты есте сақтау, URL-ді есте сақтаудан жеңіл деп жорамалдаймыз. Бұл стратегия өте сәтті болып шықты. DNS-аттар компьютерлердің IP-адрестерін ысырып шығарғандай, веб-іздеу URL-ді ысырып шығарып келеді. Іздеудің тағы бір жағымды жағы ол қателіктер мен жаңылыстарды түзетеді, ал URL-ді қате жазып, басқа парақты аласыз.

Соңында, веб-іздеу бізге желінің нақты құрылғысына аз әсер ететін, бірақ интернет-қызметтерге – қаржылық жарнама үлкен әсер ететін мүмкіндіктерді ұсынады. Жарнама – экономика қорғауышы, желі осының арқасында өсіп келеді. Баспа жарнамадан басты айырмашылығы, жарнаманы тұтынушының нені іздегеніне негіздей отырып ұсыну мүмкіндігі, сөйтіп релеванттықты жоғарылату. Механизмнің бұлай өзгеруі – іздеу нәтижесіне релевантты жарнамаларды қосу (Edelman және басқалар, 2007). Әрине, бұл жаңа модель **сілтеме бойынша шерту санын өсіру (click fraud)** тәрізді жаңа мәселелердің туындауына себеп болды. Программа тұтынушының тышқан қолтетігі көрсеткішімен сілмемеге шертуін иммитациялайды, бұл әділетсіз жолмен табылған құралдарды аударуға әкеледі.

7.4. АУДИО ЖӘНЕ БЕЙНЕНІ АҒЫНДЫҚ ТАСЫМАЛДАУ

Веб-қосымша және мобильді веб – желіні пайдалану аумағында соңғы жылдарғы тамаша өнертабыс, алайда, жалғыз бұлар емес. Көптеген тұтынушылар үшін аудио және бейне желілік технологиялардың әулие Грааль тостағаны болып көрінеді. «Мультимедиа» сөзі құрастырушыларды да, саудагерлерді де қызықтырады. Бірі мультимедиада шексіз қызықты техникалық мәселелерді, мысалы, IP арқылы дауысты жөнелту немесе тапсырыс бойынша бейнені жеткізу, мүмкін интерактивті, ал басқалары – қыруар табыс көзін көреді.

Интернет арқылы аудио және бейнені тасымалдау жетпісінші жылдары пайда болғанымен, тек XXI ғасырдың басында **мультимедианы нақты уақытта (real-time audio** және **real-time video)** тасымалдау мүмкін болды. Нақты уақыт трафигі веб-трафиктен мағынасы болу үшін, тасымалдау белгілі бір жылдамдықпен жүру керектігімен өзгешеленеді. Бұдан басқа, үнемі үзіле беретін бейнені баяу темппен қарау ешкімді қызықтыра қоймайды. Керісінше, Дүниежүзілік өрмекке жүгінгенде желіде қысқа мерзімді тұрып қалулар болуы

мүмкін, ал парақты жүктеуге көп немесе аз уақыт (белгілі бір шекарада) кетуі мүмкін, бірақ бұл күрделі мәселе болмайды.

Осындай өзгерістерді қамтамасыз ету үшін, екі оқиға орын алды. Біріншіден, компьютерлер айтарлықтай қуатты болды, олар микрофон және камералармен жабдықталды, сондықтан аудио және бейне деректерді жеңіл енгізіп, өңдеп және шығару мүмкіндігі пайда болды. Екіншіден, Интернеттің өткізгіштік қабілеттілігімен байланысты мәселелер шешілді. Интернет ортасына апаратын ұзын сілтемелер бірнеше гигабайт жылдамдықпен жұмыс істейді, ал кеңжолақты қолжеткізу және 802.11 сымсыз байланысы Интернет шетіндегі тұтынушыларға дейін жетеді. Бұл даму интернет-провайдерлерге үлкен көлемдегі трафиктерді әдеттегі 56 Кбит/с телефон модемдері арқылы тасымалдауға мүмкіндік берді. Осының арқасында тұтынушылар Интернетке 100-1000 есе жылдам қосыла алады.

Арна өткізгіштік қабілеттілігінің кеңеюімен аудио және бейне трафик өсті, бірақ оның түрлі себептері бар. Телефон сөйлесулері арнаның аз ғана бөлігін алады (64 Кбит/с, тығыздағанда одан да аз), бірақ телефон қызметі қашанда қымбат болатын. Компаниялар телефон шоттарын қысқарту үшін, телефон трафигін Интернетке аударуға болатындығына назар аударды. Skype тәрізді программалар тұтынушыларға интернет-байланысты пайдаланып, тегін телефон қоңырулар шалуға мүмкіндік берді. Әдеттегі дауыс қоңырауын арзан тасымалдау әдісін көрген телефон компаниялары, желіге шығуға арналған құрылғыларды пайдаланып жылдам дами бастады. Нәтижесінде интернет-желі арқылы тасымалданатын **«IP үсті дауыс» (Voice over IP, VoIP)** немесе **интернет-телефония (Internet telephony)** деп аталатын дауыстық деректер бірнеше есеге өсті.

Аудиоға қарағанда бейне арна өткізгіштік қабілеттілігінің біраз бөлігін алады. Сапалы интернет-бейне шамамен, 1 Мбит/с-ке дейін тығыздалады, ал әдеттегі DVD-фильм шамамен 2 Гбайтты алады. Интернетке кең жолақты қол жеткізу пайда болғанша фильмдерді тасымалдау мүмкін емес болатын. Қазір бәрі өзгерді. Жылдам қолжеткізу пайда болғалы тұтынушылардың желіде орналасқан бейнені үйде көру мүмкіндігі пайда болды. Көбі осылай жасайды да. Тұтынушылардың төрттен бір бөлігі күнде танымал бейне сайт YouTube-ке кіреді. Бүгінде бейне фильмдерді жалға алуды Интернет жүктеулерге алмастырды. Ал Интернетте орналасқан бейнероликтердің жалпы көлемі интернет-трафик құрылымын мүлдем өзгертті. Қзіргі таңда Интернетте орналасқан ақпараттың үлкен бөлігі – бұл бейне, ал бірнеше жылдан кейін бейне үлесіне интернет-трафиктің 90%-ы келетін болады (Cisco, 2010).

Өткізгіштік қабілеттілік жолағының ені бейне және аудионы тасымалдау үшін жеткілікті екенін ескерсек, конференция өткізу және мультимедианы тасымалдаудың басты мәселесі – желілік кідіріс. Аудио және бейне нақты уақытта ұсынуды талап етеді, демек, олар пайдалы болу үшін белгілі бір жылдамдықпен қалыпқа келтірілуі тиіс. Ұзақ кідірістер интерактивті болуы керек қоңыраулардың олай болмай шығатындығын білдіреді. Егер сіз спутниктік те-

лефон арқылы сөйлескен болсаңыз бұл мәселені түсінесіз. Бұл жағдайда тіпті, жарты секунд кідіріс ашуландырады. Музыканы және фильмді желі арқылы қалыпқа келтіргенде абсолютті кідіріс ешқандай рөл атқармайды, ол тек файлдың қай кезде ашылатынына әсер етеді. Бірақ тұрақсыз **синхронизация (jitter)** деп аталатын кідірістің мәні бар. Бұл кідірісті плеер маскілеуі тиіс, кері жағдайда аудио түсініксіз, ал бейне үнемі секіріп тұратын болады.

Біз бұл бөлімде осы мәселені қарап анықтауға көмектесетін кейбір стратегияларды талқылаймыз, сонымен бірге аудио және бейне сессияны орнатуға көмектесетін хаттамаларды қарастырамыз. Аудио және бейне жайлы айтқаннан кейін, әртүрлі тәсілдер пайдаланылатын үш жағдайды қарастырамыз. Бірінші және ең қарапайым жағдай – желіде сақталған бейнені тасымалдау, мысалы, YouTube файлдарын қарау. Келесі жағдай – жанды мультимедиалық контентті ағындық тасымалдау. Бұған мысал ретінде, радио және телестанциялар Интернет арқылы хабарлайтын, онлайн-радио және теледидарды (IPTV) келтіруге болады. Соңғы және ең күрделі жағдай – бұл, Skype арқылы қоңырау тәрізді, интерактивті аудио және бейне конференциялар.

Интернет мәнмәтінінде **мультимедиа (multimedia)** термині жиі аудио және бейне дегенді білдіреді. Нақтырақ айтсақ, мультимедиа екі немесе одан да көп аудио көзбе-көз ақпарат құралдарын пайдалану дегенді білдіреді. Осы анықтама бойынша біздің кітабымыз мультимедиа объекті бола алады, себебі мұнда мәтін және графика (сурет) бар. Осылай бола тұрғанымен сіз өзіңізге мультимедианы басқаша елестетіңіз, сондықтан біз бұл терминді **уақыт бойынша созылатын медианы (continuous media)** белгілеу үшін пайдаланамыз, яғни белгілі бір уақыт кезеңінде ойналатын. Іс жүзінде бұл жиі аудио және бейне, демек, дыбыс және қозғалыстағы бейне.

Осындай анық анықтама болғандығына қарамастан, көпшілік «мультимедиа» дегенде тек таза дыбыстық ақпарат деп қабылдайды, мысалы, интернет-телефония немесе интернет-радио. Шын мәнінде, бұл дұрыс емес. Бұл жағдайда әлдеқайда сәтті сөзтіркесі **ағындық ақпарат (streaming media)** болып табылады, алайда, біз үйір инстинктіне беріліп, нақты масштабта берілетін аудио деректерді «мультимедиа» деп атайтын боламыз.

7.4.1. Сандық дыбыс

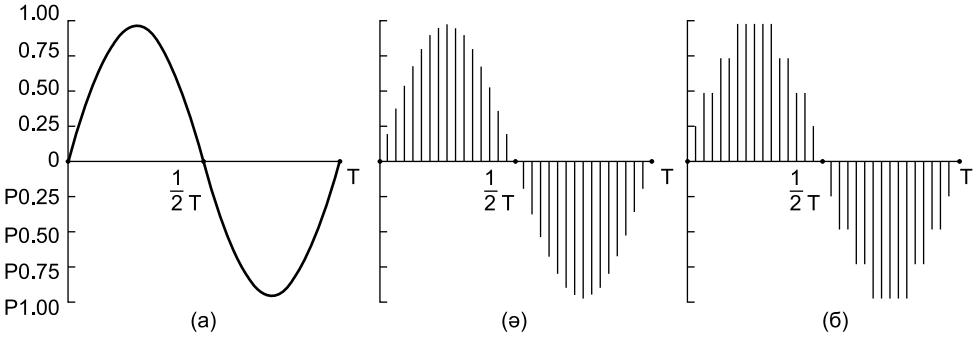
Дыбыстық толқын бірөлшемді акустикалық толқын (қысым толқыны) болып келеді. Осындай толқын құлаққа жеткен кезде, дабыл жарғағы тербеліп, ішкі құлақтың жіңішке сүйектерін тербелте бастайды, нәтижесінде нерв жүйесі бойынша бастағы миға бүлкілдеген сигнал жөнелтіледі. Бұл бүлкілдеуді тыңдаушы дыбыс ретінде қабылдайды. Дәл осылайша, акустикалық толқын микрофонға әсер еткенде, ол дыбыс амплитудасын уақыт функциясы ретінде қабылдайтын электр сигналын құрастырады.

Адам құлағының 20-дан 20 000Гц жиілік диапазонындағы сигналды есту қабілеті бар, ал жануарлар, мысалы, иттер оданда жоғары жиілікті есті алады.

Құлақ қабылдайтын, дыбыс қаттылығы, амплитудаға қатысты логарифмдік түрде өзгереді, сондықтан амплитудалары А және В екі дыбыстың күші әдетте, $\mathbf{10\log_{10}(A/B)}$ ретінде **дицибелмен (дБ)**, өлшенеді. Егер естудің төменгі шегін (2×10^5Па шамадағы қысым) жиілігі 1 кГц синусоидалық толқынды 0 дБ ретінді алсақ, онда әдеттегі сөйлесу қаттылығы сәйкесінше, 50 дБ болады, ал 120 дБ дыбыс-та ауру сезгіштік шегі орын алады, бұл 1 млн. амплитудаға сәйкес келеді.

Адам құлағы бірнеше секунд жалғасатын дыбыс өзгерісіне өте сезімтал. Көз керісінше, мұндай қысқа уақытты өзгерістерді байқай алмайды. Сонымен, мультимедианы тасымалдау кезіндегі бірнеше миллисекундтық күлтілдеу (джиттер) сурет сапасына қарағанда дыбыс сапасына көбірек әсер етеді.

Сандық аудио – аудиотолқынды қайта қалыпқа келтіру үшін қолданылатын сандық ұсыну. Дыбыстық толқындарды **аналогтық-сандық түрлендіруші (АСТ)** көмегімен сандық формаға түрлендіруге болады. АСТ-ға кірісінше, электр кернеуі беріледі, ал шығысында екілік сан құрастырылады. *7.18 а-суретінде* синусоидалық толқын мысалы келтірілген. Осы сигналды сандық түрде ұсыну үшін, біз сигнал мәнін *7.18 ә-суретінде* көрсетілгендей ΔT тең уақыт кезеңінде өлшей аламыз. Егер дыбыстық толқын таза синусоидалық емес, бірнеше синусоидалық толқын қосындысын құрайтын болса және оның құрама бөліктерінің ең жоғары жиілігі *f*-ке тең болса, онда Найквист теоремасына (*2-тарауды* қараңыз) сәйкес сигналды әрі қарай қалпына келтіру үшін дискреттеу жиілігі $2f$ сигнал мәнін өлшеу жеткілікті. Жиілігі үлкен сигналды өлшеудің қажеті жоқ, себебі жоғары жиіліктер сигналға кірмейді.



7.18-сурет. Толқын: а – синусоидалық; ә – дискреттеу; б – санауды 4 битпен кванттау

Кері үдерісте сандық мәндер аналогтық электр кернеуіне түрлендіріледі. Бұл **сандық аналогтық түрлендіруші (САТ)** көмегімен жүзеге асырылады. Сонан кейін дыбысты күшейткіш аналогтық кернеуді акустикалық толқынға айландырады да адамдар дыбысты есті алады. Мысалы, *7.18 б-суретінде* санаулар тек -1,00-ден +1,00-ге дейін 0,25 қадаммен, 9 мәнді қабылдай алады. Сегіз биттік кванттауда әр санау 256 әртүрлі мәннің бірін қабылдай алады. 16-биттік кванттауда сигналды бұданда жоғары дәлдікпен кодтауға болады,

себебі әр сигнал мәніне 65 536 түрлі мәндердің бірін сәйкестендіре алады. Шекті сан мәнін қабылдай алатын кванталған сигналдың дұрыс сәйкес келмеуі салдарынан болған қателік **кванттау шуы (quantization noise)** деп аталады. Сигналдың әр санауы ұсынылатын биттер саны жеткіліксіз болған кезде бұл шудың қаттылығы соншалық, бастапқы сигналдың өзгергенін де, шуды да ажыратуға болады.

Сандық дыбысты пайдаланудың екі жақсы мысалы телефон (егер жаңа сандық АТС қолданылса) және аудио-компакт-диск бола алады. Телефон жүйесінде қолданылатын импульсті-кодтық модуляциялауда, секундына 8000 мың рет өлшенетін, сегіз биттік санау пайдаланылады. Қабылданатын бұрмалануды азайту үшін шкала сызықтық емес болып келеді және секундына 8000 өлшеу жүргізген кезде 4кГц-тен жоғары жиілік жоғалады. Солтүстік Америка және Жапонияда кодтау кезінде **μ-заңдылығы (μ-law)** қолданылады. Еуропада және әлемнің көптеген басқа елдерінде кодтау кезінде **A-заңдылығы (A-law)** қолданылады. Әр кодтау 64 000 бит/с деректер ағынын қамтамасыз етеді.

Аудио-компакт-дискке дискреттеу жиілігі 44 100 Гц санға айналдырылған дыбыстық сигнал жазылады, нәтижесінде жиілігі 22 кГц дыбысты сақтау алады, бұл адамдар қабылдай алатын сапалы дыбыс, бірақ жақсы әуенді бағалай алатын иттер арасында өте төмен сапа болып саналады. Әр санауға 16 бит бөлінеді, оның мәні сигнал амплитудасына пропорционал. Өлшеулердің адам құлағы динамикалық диапазоны бір миллионнан аса мәнді қабылдай ататындығын көрсетсе де, 16-биттік санаудың тек 65 536 әртүрлі мәнді қабылдай алатынына назар аударыңыз. Сөйтіп, CD-сападағы аудио телефон арқылы берілетін аудиодан әлдеқайда жақсы болса да, 16 биттік санауды пайдалану едәурі кванттау шуын береді (динамикалық диапазон толық қамтылмаған, компакт-дискінің дыбыс сапасына ешқандай сөгіс жоқ). Кебір аудиофил фанаттар осы уақыттқа дейін CD-жазбаны емес, минутына 33 айналым ұзақ ойнайтын пластинканы таңдайды, себебі пластинканың 22 кГц шекті жиілік шектеуі және кванттау шуы жоқ (Алайда, егер оны ұқыпты ұстамаса, онда сызықтар пайда болады). Әрқайсысы 16 бит секундына 44 100 санау кезінде тығыздалмаған CD-сападағы аудиоға монофондық сигнал үшін 705,6 Кбит/с және стерефондық сигнал үшін 1,411 Мбит/с өткізгіштік қабілеттілік қажет.

Дыбысты тығыздау

Аудио деректер бейне деректер тәрізді үлкен өткізгіштік қабілеттілікті қажет етпейтіндігіне қарамастан, арнаның қажет өткізгіштік жолағын және тасымалдау уақытын қысқарту үшін аудио жиі тығыздатылады. Барлық тығыздау жүйелерінде екі алгоритм болуы керек: бірі деректерді орналасқан жерде тығыздау үшін және екіншісі оларды ашу үшін. Әдебиеттерде бұл алгоритмдер сәйкесінше **кодтау (encoding)** және **қайта кодтау (decoding)** алгоритмі деп аталады. Біз де осы терминологияны пайдаланатын боламыз.

Тығыздау алгоритмдерінде белгілі бір дәрежедегі ассиметрия бар және

ол жайлы білген дұрыс. Біз қазір аудионы қарастырып отырсақ та бұл аспект бейнеге де қатысты. Көптеген қосымшалар үшін мультимедиа-құжат бір рет тығыздалады (мультимедиа-серверде сақталғанда). Бұл ассиметриялық кодтау алгоритмінің баяу және қымбат құрылғыны қажет ететіндігін білдіреді, ал қайта кодтау алгоритмі жылдам және арзан құрылғыда жұмыс істеуі керек. Танымал аудио және бейне сервер операторы өзінің бүкіл кітапханасын тығыздау үшін бірнеше компьютерді сатып алуды ойлауы мүмкін және әуен тыңдау немесе фильм көру үшін сайтқа кірген тұтынушыларынан да соны талап етсе – бұл жақсы ой бола қоймас. Бүгінгі күнде қолданыстағы тығыздау жүйелері өте көлемді, бұның барлығы кодтау өте баяу және күрделі болса да, қайта кодтау жылдам және қарапайым болу үшін жасалып отыр. Екінші жағынан Skype арқылы қоңыраулар тәрізді жанды бейне және аудио үшін баяу кодтау жарамсыз. Ол нақты уақыт режимінде жұмыс істеуі тиіс. Демек, нақты уақыт режиміндегі мультимедиа дискіге сақтаулы аудио және бейнеден ерекше алгоритмдер мен параметрлерді пайдаланады. Жиі аздап тығыздау қолданылады. Симметрияны екінші бұзылуы – кодтау/қайта кодтау үдерісі қайтымды болуы міндетті емес. Яғни файлды тығыздап, тасымалдап және қайта қалпына келтірген кезде тұтынушы бүкіл ақпаратты соңғы битіне дейін алуы тиіс. Мультимедиаға қатысты бұл талап өзекті емес. Әдетте, кодтау және қайта кодтаудан кейін аудио және бейне сигнал дыбысы (немесе бейне көрнісі) дәл бастапқыдай болады деген шартпен, түп нұсқадан өзгеше болуы мүмкін. Қайта қодталған файл кодталған түп нұсқаға толық сәйкес келмесе ақпаратты тасымалдау **шығынмен (lossy)** жүргізіледі. Егер кіріс және шығыс файлдары сәйкес келсе, тасымалдау **шығынсыз (lossless)** жүреді. Шығынды жүйелер өте қажет, себебі ақпараттың біраз бөлігін жоғалтып біз жақсы тығыздауға ие боламыз.

Телефон сымдары арқылы халықаралық байланыс ертеден қымбат болатын, сондықтан адам дауысы тәрізді аудиодан тұратын **дыбыстық кодерлермен (voice coders - vocoders)** орындалатын жұмыстар өте көп. Дауыс 600-ден 6000 Гц диапазонды қамтиды және сөйлеушінің дауыстық аппараты, тілі және жағының ерекшеліктеріне байланысты, механикалық үдеріс арқасында туындайды. Кейбір дауыстық кодерлер сөздерді кішігірім параметрлер жиынтығына (мысалы, резонаторлар көлемі және формасына) әкелу үшін дауыстық жүйе моделін және 2,4Кбит/с тең деректер көлемін пайдаланады. Алайда, бұл құрылғыларды қарастыру біздің кітабымыздың шеңберінен тыс қары жатыр.

Біз, әдеттегі CD-сапаға жақын аудионы Интернет арқылы тасымалдауға тоқталамыз. Мұндай аудионы тасымалдау кезінде де көлемді қысқарту жағымсыз. Тағыздалмаған, стерео сападағы аудионы тасымалдау үшін ені 1,411 Мбит/с арна қажет болады, бұл бейне және басқа да веб-трафикке аз орын қалдырады, көптеген кеңжолақтық арналарды тежейді. Тығыздау арқасында деректерді тасымалдау жылдамдығын бір ретке төмендетуге болады және бұл жағдайда сапа елеусіз аз жоғалады немесе мүлде жоғалмайды.

Тығыздау және қайта қалыпқа келтіру сигналды өңдеуді талап етеді. Қуанышқа орай дыбыс және бейне роликтерді компьютерлер арнайы ПЖ көмегімен жеңіл өңдей алады. Іс жүзінде тұтынушыға әртүрлі ақпарат көздерінен алынған медиа деректерді жазып, бейнелеп, түзетіп және сақтауға арналған көптеген программалар бар. Бұл көп сандағы музыка және фильмдердің интернет арқылы қолжетімді болуына әкелді – бірақ бұның бәрі заңсыз – нәтижесінде сотқа көптеген артистер және құқық иелерінен шағым түсуде.

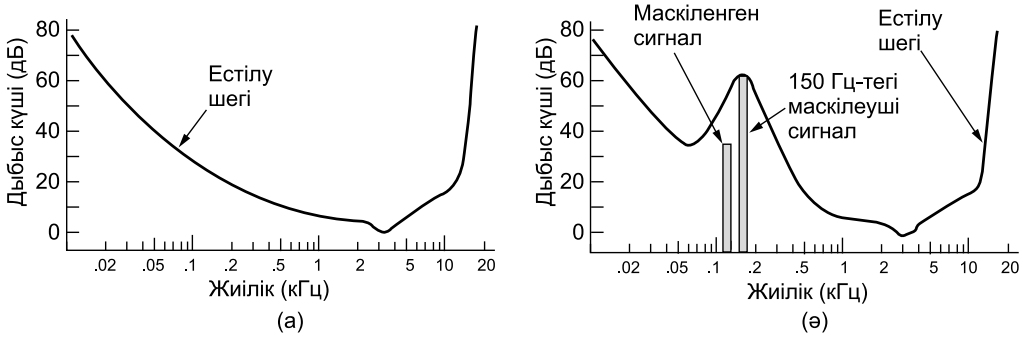
Аудио файлдарды тығыздау үшін көптеген алгоритмдер құрастырылған болатын. Ең танымал форматтар **– MP3 (MPEG audio layer 3 – MPEG аудио 3 деңгей)** және **MP4** файлдарында пайдаланылатын **ACC (Advanced Audio Coding – жақсартылған аудио кодттау)**. Шатаспас үшін MPEG аудио және бейненің тығыздалуын қамтамасыз ететінін есте сақтаңыздар. MP3 MPEG-1 стандартының 3 версиясы емес, аудионы тығыздау блогына (*3-бөлім*) жатады. Іс жүзінде MPEG 3 версиясы шыққан жоқ, тек MPEG-1, MPEG-2 және MPEG-4 шықты. ACC – MP3-тен кейін пайда болған формат. Ол үнсіз келісім бойынша MPEG-4 аудионы кодтау үшін пайдаланылады. MPEG-2-де кодтаудың: MP3 және ACC сияқты екі нұсқасы да пайдалануға болады. Енді түсінікті болған шығар. Стандарттардың ең жақсы жағы – бұл үнемі таңдау бар. Егер сізге бірі ұнамаса, басқасын таңдауға болады.

Дыбысты тығыздаудың екі тұжырымдамасы бар. **Сигнал формасын кодтау (waveform coding)** кезінде сигнал Фурье түрлендіруі бойынша компоненттерге ажыратылады. Екінші тарауда біз оны Фурье тізбегіне ажырату нәтижесінде алынған амплитуда және уақыт функциясы түрінде мысал келтіргенбіз (*2.1 а-суретін* қараңыз). Әр компонент амплитудасы ең аз бұрмалаумен кодталады. Бұндағы мәселе – сигнал формасын мүмкіндігінше ұқыпты және биттерді аз жұмсап тасымалдау.

Келесі концепция **естілімдік кодтау (perceptual coding)** деп аталады. Ол адамның есту аппаратының кемшіліктеріне негізделген және тыңдаушы нақты сигналды мен кодталған сигнал арасындағы айырмашылықты сезбейтіндей етуге мүмкіндік береді. Бірақ осциллографта бұл айырмашылық көрінетін болады. Естілімдік кодтау негізделген ғылым **психоакустика (psychoacoustic)** деп аталады. Ол адамның дыбысты қабылдауын зерттейді. MP3 және ACC форматтары да естілімдік кодтауды пайдаланады.

Естілімдік кодтаудың кілтті қасиеті – бір дыбыс келесі бір дыбысты **маскілей** алады. Өзіңізді жаздың жайма шуақ кешінде көгалда, флейта мен оркестрге арналған жанды концертті тыңдап, медитациялап отырмын деп елестетіп көріңіз. Кенеттен, кен балғасы бар бір топ жұмысшылар пайда болып, көрші көшедегі асфальтты аша бастады делік. Өкінішке орай, флейта дауысын енді ешкім естей алмайды. Флейтаның әзіз дауысын кен балғасының дауысы маскіледі. Егер осы жағдайды деректерді тасымалдау тұрғысынан қарастыратын болсақ, мұнда тек кен балғасы жұмыс істейтін жиілік диапазонын кодтау жеткілікті, себебі бұл шудан флейтаны есту мүмкін емес. Белгілі

бір жиілік диапазонындағы қатты дауыстардың басқа диапазондағы баяу дауысты «жасыру» (қатты дауыс жоқ болса оны естуге болар еді) мүмкіндігі **жиіліктік маскілеу (frequency masking)** деп аталады. Тіпті, жұмысшылар кен балғасы жұмысын тоқтатқаннан кейін де біраз уақытқа дейін тыңдарман флейта дауысын ести алмайды. Бұл қатты дауыс пайда болған кезде адам құлағының дыбысты күшейту коэффициенті күрт төмендейді және кен балғасы жұмысын тоқтатқаннан кейін оның қалыпты жағдайға оралуына уақыт керек. Бұл әсер **уақытша маскілеу (temporal masking)** деп аталады.



7.19-сурет. Есту шегі жиілік функциясы тәрізді (а); маскілеу әсері (ә)

Осы әсерлердің сапалық сипаттамасынан сандық сипаттамасына көшу үшін тәжрибе 1-ді жүргізуді елестетейік. Тыныш бір аудиторияда отырған адам компьютердің дыбыстық картасымен байланысқан құлаққапты киді делік. Компьютер жиілігі 100 Гц, күші біртіндеп өсетін дыбысты генерациялайды (таза синусоидалық дыбыстық толқын). Тәжірибеге қатысушы адам дыбысты естісімен пернетақта батырмасын басуы керек. Компьютер батырма басылған сәттегі дыбыс күшін есте сақтайды. Тәжірибені осылайша 200 Гц, 300 Гц жиілікте және т.с.с есту жиілігінің жоғарғы шегіне жеткенше қайталайды. Тәжірибені бірнеше көптеген адамдармен қайталау керек. *7.19 а-суретінде* орташаланған естілу шегінің дыбыс жиілігіне тәуелділігі екі осьте де логарифимдік масштабпен көрсетілген. Бұл қисыққа қарап келесідей тұжырымдама жасауға болады: амплитудасы есту шегінен төмен жатқан жиіліктерді ешуақытта кодтаудың қажеті жоқ. Мысалы, егер 100 Гц жиілікте дыбыс күші 20 дБ тең болса, онда бұл дыбысты кодтаудың қажеті жоқ және дыбыстың естілу сапасы төмендемейді, себебі 100 Гц жиілікте 20 дБ есту шегінен төмен жатыр (*7.19 а-суретін* қараңыз).

Енді тәжірибе 2-ні қарастырайық. Компьютер тәжірибе 1-дегі әрекетті қайталасын, бірақ әр тестілік жиілік үстіне, айталық 150 Гц жиіліктік тұрақты амплитудасы бар синусоидалық дыбыстық толқыны қосылып отырады. Біз 150 Гц-ке жақын орналасқан жиіліктер үшін есту шегінің күрт өсетінін байқаймыз. Бұл *7.19 ә-суретіндегі* графикте көрсетілген.

Соңғы тәжірибеден келесідей тұжырымдама жасауға болады: қандай сигналдардың жақынарадағы қуатты сигналдармен маскіленетінін біле отырып, біз сәйкес сигналдарды есепке алмай, кодтамауымызға және осылайша биттерді үнемдеуімізге болады. *7.19 ə-суретінен* 125 Гц жиілікті сигналды толығымен есепке алмауға болатыны көрініп тұр және айырмашылықты ешкім байқамайды. Тіпті, кез келген диапазонда қатты сигнал тоқтағаннан кейін уақытша маскілеу қасиетін білетін жағдайда, біраз уақыт аралығында (құлақ қуаттылығы кіші дыбысқа бейімделгенше) осы жиілікті кодтамауға болады. MP3 және AAC алгоритмдерінің мақсаты – дыбыс күшін алу үшін сигналды Фурье тізбегіне таратып, сонан кейін мүмкіндігінше аз биттер санымен кодталатын тек маскіленбеген жиілікті тасымалдау.

Енді негізгі қағиданы білгеннен кейін кодтаудың қалай жүргізілетінін қарастырайық. Дыбысты тығыздау, AAC үшін 8-ден 96 кГц арасындағы сигнал амплитудасын өлшеу арқылы, ал кейде дыбыс CD сапасына жақын болу үшін жай 44,100 кГц жиілікпен орындалады. Өлшеуді бір (моно) немесе екі (стерео) арна арқылы жүргізуге болады. Сонан кейін қажет шығыс биттік жылдамдық таңдап алынады. MP3 алгоритмінің көмегімен компакт-дискке жазылған рок-н-рол стереофондық жазбасын 96 Кбит/с дейін, тіпті, рок-н-рол фанаттары да сезбейтіндей сапа жоғалтумен тығыздауға болады. Егер біз MP3-ке фортапиано концертін жазғымыз келсе, онда бізге биттік жылдамдығы кем дегенде 128 Кбит/с AAC форматы қажет. Айырмашылық рок-н-ролдағы сигнал/шу қатынасының фортапиано концертіне қарағанда әлдеқайда жоғары екендігінде (әрине, тек техникалық жағынан). Әрине, кіші биттік жылдамдықты таңдап, төмен сападағы туындыны алуға болады.

Бұдан кейін есептеулер кішігірім топтармен өңделеді. Әр топ алдын ала жиілік диапазонын ерекшелейтін сандық фильтрлер жиынтығы арқылы өтеді. Маскіленетін жиіліктерді анықтау үшін ақпарат психоакустикалық модельге енгізіледі. Келесі қадамда биттер қоры жиіліктер диапазоны арасында таратылады. Бұл жерде биттердің ең көп саны маскіленбеген спектрлік қуат диапазонына беріледі, ал кіші бөлігі – маскіленетін диапазонға. Соңында, биттік тізбектер, жиі кездесетін сандарға қысқа, ал сирек кездесетін сандарға ұзын код беретін Хаффман (Huffman) коды арқылы шифрланады. Егер сізді бұл тақырып қызықтырып, ол жайлы көбірек білгіңіз келсе, Бранденбург (Brandenburg, 1999) кітабына жүгініңіз.

7.4.2. Сандық бейне

Енді құлақ жайлы бәрін білгеннен кейін көзге көшетін уақыт келді. Сіздің қоятын сұрағыңыздың алдын алып, келесі бөлімде мұрын жайлы әңгіме болмайтынын айта кетейік. Адам көзінің бір ерекшелігі бар: көрініс көздің ішкі тор қабығында пайда болған кезде ол бірнеше миллисекунд сақталып, сонан кейін басқа көрініске орын береді. Егер суреттер секундына 50 көрініс жылдамдықпен ауысып отыратын болса, көз бейненің дискретті

екенін байқамайды. Осы қағиданы қозғалыстағы көріністі бейнелейтін бейне жүйелерінің барлығы пайдаланады.

Бейнені ұсынудың ең қарапайым жолы – бейнені құрайтын тікбұрышты элементтер – **пиксельдер** жиынтығынан тұратын кадрлар тізбегі. Бейнелену үшін әр пиксель жеке бит болуы мүмкін, не қара, не ақ. Алайда, мұндай жүйенің сапасы өте нашар. Өзіңіздің сүйікті графикалық редакторыңыздың көмегімен түрлі түсті бейнені ақ-қара (сүр түстер емес, нағыз ақ-қара) түске түрлендіріп көріңіз. Келесі қадам – сүр түстің 256 деңгейін көрсету үшін, бір пиксельге 8 битті пайдалану. Осы схема бойынша жоғары сапалы ақ-қара түсті бейнені алуға болады. Түрлі-түсті бейне үшін көптеген жүйелер түстің әр негізгі компонентіне: қызыл, жасыл және көк (RGB) түске 8 битті пайдаланады. Әр түс қызыл, жасыл және көк түстерді белгілі бір қоюлығын сызықтық араластыру арқылы алынатын болғандықтан осындай ұсыным болуы мүмкін. Бір пиксельге 24 бит пайдаланған кезде біз 16 млн. түстер аламыз, бұл адам көзі ажырататын түстер санынан әлдеқайда көп.

Түрлі түсті сұйық кристаллды компьютер немесе теледидар дисплейінде әр пиксель, аралықтармен бөлінген қызыл, жасыл және көк түстер ішкі пиксельдерінен тұрады. Суреттер ішкі пиксельдердің білгілі бір анықталған қоюлығы көмегімен бейненеледі, ал көз түс компоненттерін араластырады.

Әдетте, көріністер секундына 24 кадр (35-мм кинопленкадағыдай), 30 кадр (америка теледидарындағыдай – NTSC жүйесі) және 25 кадр (бүкіл әлемге таралған PAL жүйесіндегі теледидардағыдай) жылдамдықпен алмасып отырады. Дәлірек айтсақ, АҚШ-дағы әдеттегі теледидар бейнені секундына 29,97 рет алмастыру арқылы таратады. Теледидар ақ-қара болған кезде кадрлар секундына 30 рет алмасатын, түрлі түсті бола бастағаннан кейін инженерлер түстер жайлы ақпаратты тасымалданатын ағынға енгізу үшін кадрлардың алмасу жиілігін секундына 29,97 кадрға дейін төмендетті. Компьютерлер кадрды секундына 30 рет алмастыруы тиіс. PAL NTSC-дан кейін пайда болды, іс жүзінде онда секундына 25,000 кадр пайдаланылады. Біздің әңгіметіміздің айтылмаған тұсы қалмас үшін, Францияда французша сөйлейтін Африкада және бастыс Еуропа бөлігінде қолданылатын үшінші жүйе – SECAM жайлы айта кету керек. Ол батыс Еуропаға, сол жердегі халықтың шығысгермандық теледидарды (PAL) қарап, өзге идеяны қабылдауы үшін батыс Германиядан келді. Бірақ қазір көптеген елдер PAL-ға ауысуда.

Телехабар үшін секундына 25 кадр – бірқалыпты қозғалысты тасымалдау үшін өте жақсы сапа, бейне жұп және тақ жолдар жаймасы бойынша екі **өріске (fileds)** бөлінбейді. Екі өріс (әрқайсы жайманың жартысы) тізбекпен, секундына 60 (NTSC) немесе дәл 50 (PAL) жол береді. Мұндай жүйе **алмасу (interlacing)** деген атпен белгілі. Компьютерде көруге арналған бейне **тізбектік (progressive)**, яғни алмасу қолданылмайды, себебі монитор бейнекартасында буфер бар. Буферге жаңа суретті секундына 30 қоюға болады, ал экрандағы сурет жыпылықтау болмас үшін, секундына 50, тіпті, 100 рет жаңартылуы мүмкін. Аналогтық теледидардың компьютердегідей буфері жоқ. Алмасуды

пайдаланатын компьютерде жылдам қозғалыстағы көрініс бейнеленген кезде анық белгіленген көрініс шеттерімен бірге көлденең сызықты байқауға болады – бұл әсер **комбинг (combing)** деп аталады.

Интернет арқылы тасымалданатын кадрлар мөлшері қатты өзгеріп отырады, себебі үлкен фреймдер үшін үлкен өткізгіштік жолақ қажет, ал бұл үнемі қолжетімді бола бермейді. Айыру қабілеті төмен бейне 320×240 пиксель болуы мүмкін, ал толық экранды – 640×480. Бұл мөлшерлер сәйкесінше алғашқы компьютерлік мониторларға және NTSC форматындағы теледидарға қатысты. Мөлшер қатынасы (aspect ratio) немесе суреттік пиксельмен берілген енінің биіктігіне қатынасы 4:3, стандартты теледидардағыдай. **HDTV (High-Definition TeleVision – айыру қабілеті жоғары теледидар)** бейне файлдары 1280-де 720 пиксель айыру қабілетімен жүктеле алады. Фильмдер мөлшер қатынасына (3:2) дәлірек сәйкес келу үшін кеңэкранды файлдардың мөлшер қатынасы 16:9. Салыстыру үшін стандартты DVD-бейнені келтіруге болады. Олардың мөлшер қатынасы әдетте, 720×480 пиклель, ал мүмкіндігі жоғары Blu-ray (көк-күлгін лазер негізінде жұмыс істейтін) дискілердікі әдетте, 1080-де 720 пиксель.

Интернетте пиксельдер саны тарих болып кетті, себебі медиапеерлер бір суретті әртүрлі мөлшерде бейнелей алады. Бейне – үлкейтуге немесе кішірейтуге болатын компьютер экранындағы жай тағы бір терезе. Пиксельдер саны көп болған сайын сурет сапасы жоғары, ол үлкейткен кезде сапасын жоғалтпайды. Алайда, көптеген мониторлар пиксельдер саны HDTV-ден де көп суреттерді (демек, бейнені де) бейнелей алады.

Бейнені тығыздау

Біздің сандық бейне жайлы пікірталасымыздан Интернет арқылы тасымалдау үшін бейнені тығыздау сынды екені түсінікті болу керек. Тіпті, стандартты сападағы, суреттер мөлшері 640×480 пиксель, әр түс жайлы 24 бит ақпаратты бар және секундына 30 кадр бейнені тасымалдау үшін 200 Мбит/с керек. Бұл қарапайым тұтынушыларды айтпағанда, көптеген компаниялардың Интернетке қосылған жылдамдығынан әлдеқайда көп. Біз әзірше тек бір бейне ағын жайлы айтып отырмыз. Тығыздалмаған бейнені тасымалдау іс жүзінде мүмкін емес, асып кеткенде ауқымды желі арқылы, тек жақсы тығыздаудың мүмкін екендігін сенім арту қалады. Қуанышқа орай, соңғы бірнеше жыл ішінде осы бағытта көптеген зерттеулер жүргізілді, сондықтан көптеген тығыздау техникасы жән алгоритмдері пайда болды. Олар бейне тасымалдауға мүмкіндік береді.

Интернет арқылы бейне тасымалдау үшін көптеген стандартты және патенттелген форматтар қолданылады. Ең танымал кодттау – MPEG және оның формалары. Бұл кеңейтілімі mpg, mp4 файлдарда және басқа да контейнерлік форматтағы файлдарда қолданылатын ашық формат. Осы бөлімде біз бейненің қалай тығыздалатынын білу үшін MPEG-ті, алдымен, біз JPEG форматындағы

бейненің қалай тығыздалатынын қарастырамыз. Бейне – бұл суреттер тізбегі (дыбыспен бірге). Бейненің тығыздаудың бір жолы – тасымалданатын әр суретті тығыздау. Бірінші жуықтауда MPEG – бұл тасымалданатын әр кадрды JPEG көмегімен тығыздау және артықтықты болдырмауға арналған қосымша опциялар.

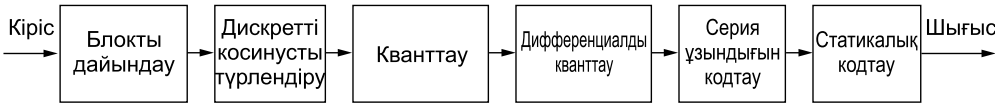
JPEG стандарты

JEPG (Joint Photographic Experts Group – фото суреттерді машиналық өңдеу үшін біріктірілген эксперттер тобы) стандарты жартылай түстік суреттерді тығыздауға арналған (мысалы, фото сурет). Оны стандартқа жауап беретін ITU, ISO, IEC және басқа да компанияларда жұмыс істеген эксперттер тобы құрастырған болатын. Бұл формат кеңінен қолданылады (кеңейтілімі jpg файлдарын қараңыз) және жиі 10 есе және одан да жоғары тығыздалады.

JPEG халықаралық 10918 стандартында анықталған. Іс жүзінде ол алгоритмнен гөрі, сатып алулар тізімін еске салады. Оған кіретін төрт тәсілдің ішінен бізді қызықтыратыны тек біреу – ақпаратты бөлшектеп жоғалтуы бар тізбектік тәсіл. Бұдан бөлек біз JPEG-тің 24-биттік RGB-бейне суретті кодтау үшін қалай қолданылатын, егжей-тегжейіне тоқталмаймыз.

Алгоритм *7.20-суретте* бейнеленген. Бірінші қадам – блоктарды даярлау. Нақтырақ болу үшін JPEG алгоритмінің кірісінде, *7.21 а-суретте* көрсетілгендей мөлшері 640×480, бір пиксельге 24 бит, RGB сурет деп болжаймыз. RGB – тығыздауға арналған түстердің ең жақсы моделі емес. Көз **хроматикалық деректерге (chrominance)** емес, **жарыққа (luminance)**, яғни бейне сигнал түсіне әлдеқайда сезімтал. Сөйтіп, біз алдымен, *Y* жарықты және *R*, *G, B* компоненттерінің екі хроматикалық сипаттамасын: *Cb* және *Cr* есептейміз. Келесі формулалар 8-биттік мәндер (0-ден 256-ға дейінгі) үшін пайдаланылады.

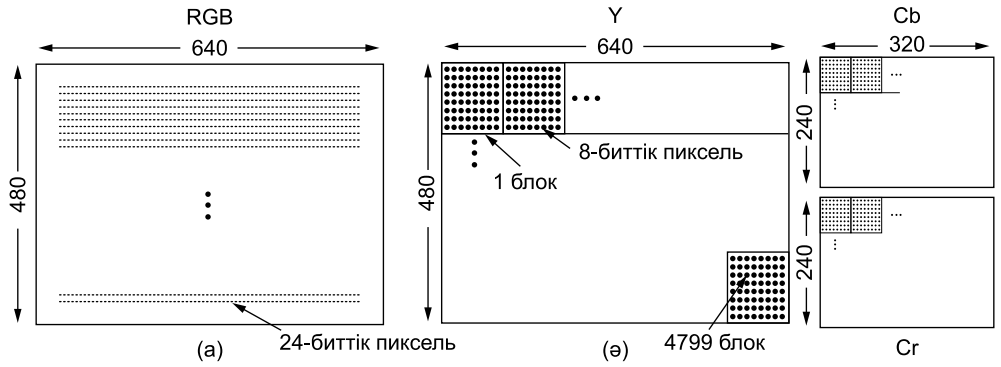
$$Y=16+0,26R+0,50G+0,09B;$$
$$Cb=128+0,15R-0,29G-0,44B;$$
$$Cr=128+0,44R-0,37G+0,07B.$$



7.20-сурет. JPEG-ті жоғалтумен тізбектік кодтау қадамдары

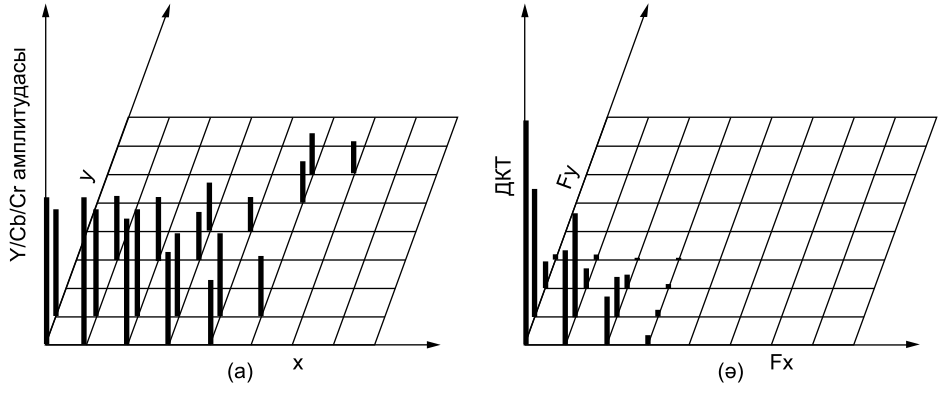
Y, Cb және *Cr* үшін жеке матрицалар құрастырылады. Сонан кейін *Cb* және *Cr* матрицаларында 4 пиксельден тұратын квадратты блоктар орташаланып бір блокқа біріктіріледі және сурет 320×240 мөлшеріне дейін қысқартылады. Тығыздау жоғалтумен жүреді, бірақ көз оны байқамайды, себебі ол үшін ең бастысы – жарық. Дегенмен, деректердің жалпы көлемі екі есе кішірейеді.

Енді нөлді диапазон центріне орналастыру үшін, барлық үш матрицаның әр элементінен 128-ді азайтамыз. Соңында әр матрица 8×8 блоктарға бөлінеді. *Y* матрицасында 4800 блок, қалған екеуінде, *7.21 ә-суретінде* көрсетілгендей – 1200 блоктан бар.



7.21-сурет. Кіріс RGB деректері (а); Блоктар даярланғаннан кейін (ә)

JPEG кодтауындағы екінші қадам – 7200 блоктың әрқайсысына жеке **ДКТ пайдалану (DCT, Discrete Cosine Transformation – ДКТ, дискретті косинустық түрлендіру)**. Әр ДКТ кірісі – бұл 8×8 ДКТ-коэффициенттер матрицасы. ДКТ (0,0) элементі – блоктың орташа мәні. Қалған элементтер әр кеңістіктік жиіліктің спектралдық қуаты қандай екенін көрсетеді. Әдетте бұл элементтер *7.22-суретте* көрсетілгендей, (0,0) координат басынан қашықтаған сайын жылдам нөлге дейін кішірейеді.



7.22-сурет. *Y* матрицасының бір блогы (а); ДКТ коэффициенті (ә)

ДКТ аяқтала салысымен, JPEG кодтауы **кванттау (quantization)** деп аталатын үшінші қадамға көшеді. Бұл қадамда аса маңызды емес ДКТ коэффициенттері өшіріледі. Бұл жоғалтуы бар түрлендіру ДКТ 8×8 матрицасын-

дағы әр коэффициентті кестеден алынған салмаққа бөлу арқылы жүргізіледі. Егер салмақтың барлығы 1-ге тең болса, түрлендіру кезінде ешнәрсе болмайды. Алайда, егер салмақ күрт өссе, онда үлкен айыру қабілеті төмендейді.

ДКТ коэффициенттері

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Квантталған коэффициенттер

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

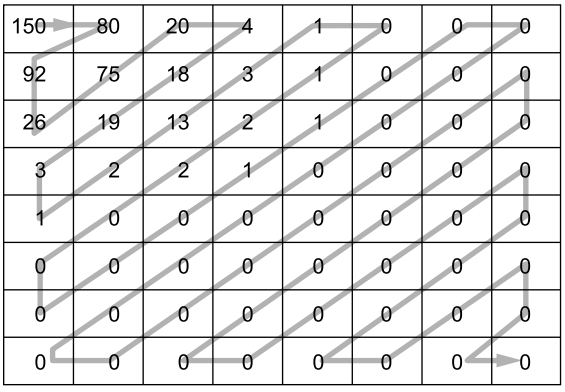
Кванттау кестесі

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

7.23-сурет. ДКТ квантталған коэффициенттерін есептеу

Бұл қадам мысалы *7.23-суретінде* келтірілген. Бұнда біз бастапқы ДКТ матрицасын, дискреттеу кестесін және әрбір ДКТ элементін дискреттеу кестесіндегі сәйкес элементке бөлгенде алынған нәтижені көреміз. Бұл кесте мәндері JPEG стандартының бөлігі болып саналмайды. Жоғалтуы бар тығыздау мүмкін болу үшін, әр қосымша оны өзбетінше ұсынуы тиіс.

Төртінші қадамда әр блоктың (0,0) мәні азаяды (сол жақ жоғарғы бұрышта), ол алдыңғы блоктың сәйкес элементі айырмасына алмастырылады. Бұл элементтер сәйкес блоктың орташа мәндері болғандықтан олар баяу өзгеруі тиіс, сондықтан айырма мәніне алмасқан кезде олардың мәні кішірейеді. Басқа мәндер үшін айырма есептелмейді.



7.24-сурет. Квантталған мәндердің берілу реті

Бесінші қадамда 64 элемент сызықтандырылады және бүкіл массивке топтық кодтау қолданылады. Блокты солдан оңға қарай, сонан кейін жоғарыдан төмен сканерлеу барлық нөлдерді жинамайды, сондықтан *7.24-суретте*

көрсетілгендей сканерлеудің иір-иір моделі қолданылады. Бұл мысалда ол матрица соңында 38 тізбектік нөлдерді береді. Бұл жол нөлдер саны 38 екенін хабарлайтын бір көрсеткішке дейін қысқартылуы мүмкін. Мұндай техника **топтық кодтау** немесе **серия ұзындығын кодтау (run-length encoding)** деп аталады.

Бізде енді суретті ұсынатын (түрлендірілген түрде) сандар тізімі бар. Алтыншы қадамда сандар сақтау немесе тасымалдану үшін Хаффман тәсілінің көмегімен кодталады, мұнда ең жиі кездесетін санға қысқа код беріледі.

JPEG күрделі болып көрінуі мүмкін, алайда, ол іс жүзінде осындай. Дегенмен, жиырма есеге дейін тығыздау мүмкіндігі осыған тұрарлық. JPEG суреттерін қайта кодтау алгоритмді кері ретпен жібереді. Іс жүзінде JPEG симметриялы: кодтау және қайта кодтауға бірдей уақыт кетеді. Бірақ тығыздау алгоритмдерінің барлығы бірдей симметриялы емес, қазір біз оны көреміз.

MPEG стандарты

Біз істің дәл мәніне енді жеттік: **MPEG (Motion Picture Experts Group – қозғалыстағы сурет сұрақтары бойынша эксперттік топ)** стандарттары. Бейнені тығыздаудың көптеген жеке алгоритмдері болғанымен, бұл стандарттар бейнені тығыздауда қолданылатын негізгі алгоритмдерді анықтайды. Олар 1993 жылдан бері халықаралық стандарт болды, себебі фильмде дыбыс та, суретте бар, ал MPEG дыбысты да, бейнені де тығыздай алады. Біз дыбыс пен суреттің тығыздалуы жайлы айттық, енді бейненің тығыздалуына көшейік.

MPEG-1 (MP-3 аудио кіретін) стандарты алғаш рет 1993 жылы жарияланып, осы уақытқа дейін кеңінен қолданылады. Оның негізгі мақсаты шығыстағы бейнеақпаратты бейнемагнитофондағыдай сапамен, 40 есе тығыздап, 1 Мбит/с жылдамдықпен тасымалдау болатын. Осындай бейне Интернет вебсайттарында пайдалануға қолайлы. Бейнемагнитофонның не екенін білмесеңіздер ол үшін қам жемеңіздер, MPEG-1 бейнені CD-де сақтау үшін де пайдаланылады. Егер сіз CD болған уақытқа да ілікпесеңіз, онда біз MPEG-2-ге көшеміз. Бұл стандарт 1996 жылы шықты. Оны бейнені телетрансляция сапасында тығыздау үшін құрастырған болатын. Қазір бұл стандарт кеңінен таралған, себебі ол бейнені DVD үшін (ал мұндай бейне үнемі Интернетке түседі) кодтаудың және сандық телехабар (DVB) негізі болып саналады. Әдетте, DVD сапасындағы бейне 4-8 Мбит/с тасымалдау жылдамдығына негізделіп кодталады.

MPEG-4 стандартында екі бейне формат бар. Оның біріншісі 1999 жылы шықты, бейнені объектіге бағытталған ұсыныс көмегімен кодтайды. Бұл түсірілімді жасанды құрастырылған суретке, мысалы, ауа райы жайлы айтып тұрған жүргізушіні картаға қоюға мүмкіндік береді. Мұндай құрылым программаларға бейне деректермен жеңіл әрекеттесе алады. Екінші формат 2003 жылы шықты, ол **H.264** немесе **AVC (Advanced Video Coding – бейнені ілгерлемелі кодтау)** деп аталады. Оның мақсаты – алдыңғы форматқа қара-

ғанда бейнені қажет өткізгіштік қабілеттіліктен 2 есе төмен және сапасын сақтап, желі арқылы жақсы сапамен тасымалданатындай етіп тығыздау. Бұл кодтаушы жоғары сападағы бейне үшін пайдаланылады.

Бұл стандарттар көптеген егжей-тегжейімен ерекшеленеді. Кейіннен шыққан стандарттарда, бертінде шыққан стандарттарға қарағанда кодтау опциялары және мүмкіндіктері көп. Алайда, біз бұл егжей-тегжейлерге тоқталмаймыз. Бейне тығыздау алгоритмнің уақыт өте толығымен өзеруінің емес, көптеген кішігірім жақсартулардың арқасында жақсарып келеді. Сондықтан біз оның барлығын жалпылама айтып кетеміз.

MPEG аудионы да, бейнені де тығыздайды. Аудио және бейне кодтаушылар бір-біріне тәуелсіз жұмыс жасайтын болғандықтан, қабылдауышта оларды синхрондау қажет. Бұл мәселенің шешімі – екі кодтаушыға да бірегей уақыт санауын және белгілерді енгізу. Бұл белгілер кодталған шығысқа енгізіледі және қабылдауышқа жөнелтіледі, сөйтіп қабылдауыш аудио және бейне ағындарын синхрондайды.

MPEG-бейнені тығыздау фильмдердегі: кеңістіктік және уақыттық сияқты екі артықтықшылықты пайдаланады. Кеңістік артықшылығы әр кадрды жеке JPEG көмегімен кодтап жоюға болады. Бұл амал жиі қолданылмайды, әдетте, тек әр кадрға кездейсоқ қол жеткізу қажет болған кезде (мысалы, бейнені редакциялаған кезде). Бұл нұсқада JPEG тығыздау деңгейі алынады.

Бірінен соң бірі жүретін кадрлар жиі бірдей екенін есепке алып, қосымша тығыздауға қол жеткізуге болады. Бұл әсер бір қарағанда онша да пайдалы емес болып көрінеді, себебі көптеген режиссерлер бір сценадан екіншісіне көшуді әр 3-4 с сайын орындайды (кез келген фильмдегі екі минут ішінде өзгеретін сценаларды санап шығыңыз). Осылай бола тұрса да 75 немесе одан да көп ұқсас кадрлардан тұратын массив, әр JPEG-ті жеке тығыздағанға қарағанда әлдеқайда қаттырақ тығыздалады.

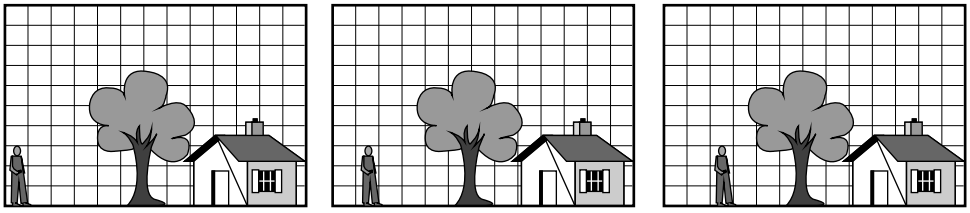
Артқы көрінісі және камера тұрақты, ал бір-екі актер баяу қозғалып отыратын сценалар үшін, көрші кадрлардың барлық пиксельдері бірдей болады. Мұндай жағдайда алдыңғы кадрды алып, түрлі JPEG-пен іске қосу жағдайды жеңілдетеді. Алайда, камера актер соңынан жүріп немесе жақындап отыратын сценаларда бұл техниканың көмегі аз. Қозғалысты теңелтетін әдіс қажет. MPEG осы теңелтуді орындайды және MPEG-тің JPEG-тен басты айырмашылығы осында.

MPEG шығысында үш түрлі:

1. I-(Intracoded – өз-өзінде кодталған) кадрлар (өзіне өзі енгізілген, тығыздалған қозғалыссыз кадрлар);
2. P-(Predictive) кадрлар (алдыңғы кадрдан блоктың айырмашылығы бар);
3. B-(Bidirectional) кадрлар (алдыңғы және келесі кадрдан блоктық айырмашылығы бар) аламыз.

Бірінші типтегі I-кадрлар қозғалыссыз суреттер болып келеді. Оларды JPEG немесе соған ұқсас техника көмегімен кодтауға болады. Мұндай кадрлар үш себеппен дүркін-дүркін кездесуі тиіс (мысалы, секундына екі рет). Біріншіден, MPEG тұтынушылар саны көбейіп немесе азайып отыратын көп адрестік тасымалдау үшін қолданылуы мүмкін. Егер кадрлардың барлығы екіншісінен бастап алдыңғы кадрға тәуелді болса, онда бірінші кадрды жіберіп алған адам, келесі кадрлардың ешқайсын қайта кодтай алмайды. Екіншіден, қандай да бір кадр қателікпен алынса, қалғандары оқылмайды. Үшіншіден, бірінші типтегі кадрсыз, суреттер шатаспас үшін декодерге әр кадрды есептеуге тура келеді.

P-кадрлар, керісінше, кадрлар арасындағы айырмашылықты кодтайды. Олар **макроблоктар (macroblocks)** идеясына негізделген, мысалы, 8x8 түстер жарықтығы үшін, 16×16 пиксельді жабады. Макроблок алдыңғы кадрдағы дәл осындай немесе сәл айырмашылығы бар іздеу арқылы кодталады. P-кадрлардың қай жерде пайдалы болатындығының мысалы *7.25-суретте* келтірілген. Біз бұнда артқы көрінісі бірдей, тек кейіпкердің орны әртүрлі тізбектік кадрларды көреміз. Кейіпкер орналасқан макроблоктар белгілі бір қашықтыққа ығысады және оны іздеуге тура келеді.



7.25-сурет. Үш тізбектік кадр

MPEG стандарттары іздеудің қандай аралықты жүргізілуі тиіс екенін немесе аудандарды шартты түрде сәйкес келеді деп есептеу үшін қаншалықты ұқсас болуы керектігін анықтамайды. Барлығы әрбір жеке жағдайдағы қолданысқа байланысты. Мысалы, кейбір жүзеге асырылуларда макроблокты іздеу сол позицияда немесе *x* осі бойынша және y осі бойынша подзициясынан жүргізілуі мүмкін. Әр позиция үшін жарықтық матрицасында сәйкестіктер саны есептеледі. Сәйкестік саны ең үлкен позиция (бұл сан қажет ең кіші саннан асқан жағдайда) жеңімпаз болып жарияланады. Кері жағдайда макроблок жоқ деп есептеледі. Әрине, бұдан да күрделі алгоритмді енгізуге болады.

Егер макроблок табылса, ол алдыңғы кадрдағы макроблок айырмасы көмегімен кодталады (жарық және хроматикалық сипаттамалар бойынша). Әдетте, бұл айырмашылықтар матрицасы кейіннен дискретті косинустық түрлендіруге, кванттауға, сериялар ұзындығын кодтауға және Хаффман тәсілі бойынша кодтауға ұшырайды.

Макроблок мәні шығыс ағында ауысу векторымен бірге жүреді (макроблоктың әр бағытта қаншалықты ауысқандығы), ал сонан кейін осы айырма-

шылықты кодтау жүреді. Егер алдыңғы кадрда макроблок табылмаса, онда оның ағымдағы мәні I-кадрдағыдай кодталады. Әрине, бұл алгоритм тым ассиметриялы. Іс жүзінде ол әр макроблокты шектеу үшін, алдыңғы кадрдағы әрбір даулы позицияның қаншалықты алысқа ауысқанына қарамастан тексере алады. Бұл амал кодталған MPEG ағынын баяу кодтау арқасында кішірейтеді. Ол фильмдер кітапханасын бір реттік кодтау үшін жарамды, бірақ бейне конференцияны кодтауға жарамайды.

Әр жеке жағдайда «табылған» макроблоктың не екендігі анықталады. Осының арқасында сапа және алгоритм жылдамдығы өзгеріп отырады, алайда, жарамды шығыс MPEG үнемі табылады.

MPEG қайта кодтау әзірше тікелей жүргізіледі. I-кадрларды қайта кодтау JPEG суреттерін қайта кодтауға ұқсас жүргізіледі. P-кадрларды қайта кодтау, толығымен қайта кодталған макроблоктар және алдыңғы кадрмен айырмашылықтан тұратын макроблоктардан келесі кадрды жеке буферде құрастыру үшін, декодерден алдыңғы кадрларды буферлеуді талап етеді. Жаңа кадрдың әр макроблогы рет-ретімен жинақталады.

B-кадрлар P-кадрларға ұқсас, алайда, қолданыс жағдайында этолондық макроблок алдыңғы кадрда да, келесі кадрда да болуы мүмкін. Бұл қосымша еркіндік қозғалысты жақсы өтейді. Бұл, мысалы, объектілер біраз уақыт бірін бірі басып қалатын жағдайда пайдалы. B-кадрды кодтау үшін кодтаушы: өткендер, ағымдағылар (кодталушы) және келесі кадрлар тізбегін есте сақтауы тиіс. Қайта кодтау да біршама күрделі және көп уақыт алады. Бұның себебі, жеке B-кадр өзі тәуелді келесі кадрлар қайта кодталмайынша ол да қайта кодталмайды. Сөйтіп, B-кадрлар бейнені максималды тығыздауға мүмкіндік бергенімен, күрделілік және арнайы буферлеу қажеттілігіне байланысты кеңінен қолданылмайды.

MPEG стандарттарында осы техникалардың көптеген жақсартулары бар, бұл жақсы тығыздауға қолжеткізуге мүмкіндік береді. AVC бейнені 50:1 қатынасымен тығыздауды пайдалана алады, бұл арнаның өткізгіштік қабілеттілік талабын 50 есеге қысқартады. AVC жайлы көбірек білгіңіз келсе, Салливан және Виганд (Sullivan and Wiegand, 2005) кітабын оқыңыз.

7.4.3. Сақталған медиафайлдарды ағындық тасымалдау

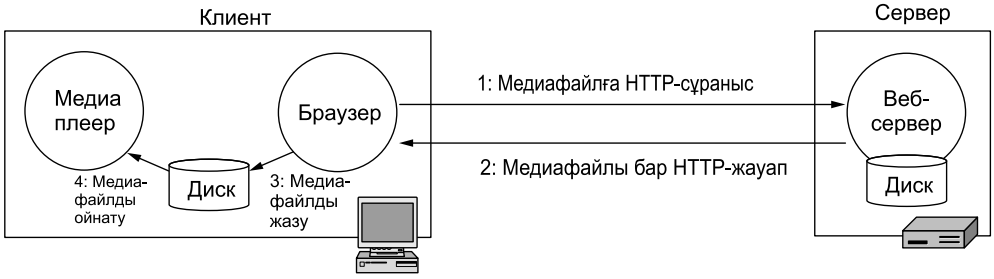
Енді желілік қосымшаларға көшейік. Біз алдымен файлда сақталған медиа ақпаратты ағындық тасымалдау жайлы сөз қозғаймыз. Ең көрнекті мысал ретінде Интернет арқылы бейне файлдарды тамашалауды келтіруге болады. Бұл – **VoD**-тың **(Video on Demand – сұраныс бойынша бейне)** бір формасы. Сұраныс бойынша бейненің басқа формалары Интернет бөлігіне жатпайтын провайдер желісіне фильмдер тасымалдауда қолданылады (мысалы, кабельдік желі).

Келесі бөлімде біз нақты уақыттағы медианы тасымалдау, мысалы, теледсигналды IP арқылы тасымалдау (IPTV) және интернет-радио жайлы ай-

тамыз. Сонан кейін біз, үшінші жағдай – нақты уақыттағы конференцияны қарастырамыз. Мысал ретінде дауысты IP арқылы тасымалдау немесе Skype арқылы өткізілетін телеконференцияларды келтіруге болады.

Бұл жағдайда аудио және бейнені желі арқылы тасымалдауға қойылатын талаптар өте жоғары, себебі біз кідірістер және оның өзгерісіне (синхрондау тұрақсыздығына) ерекше көңіл бөлуіміз керек.

Интернетте сақталған медиафайлдарды трансляциялайтын, әуен және бейнесі бар сайттар өте көп. Іс жүзінде сақталған медиафайлдармен трансляцияламай жұмыс істеген ыңғайлы. Мысалы, сіз Apple-дің iTunes тәрізді бейнені жалға беретін сайт құрастырғыңыз келді делік. Әдеттегі веб-сайт тұтынушыларға бейнені алдымен жүктеп, сонан кейін қарауды ұсынады (ақысы төленгеннен кейін әрине). Бұл тізбектер қадамы *7.26-суретте* көрсетілген. Келесі мысалмен салыстырғандағы айырмашылық жақсы байқалу үшін, біз оларды егжей-тегжейлі қарастырамыз.



7.26-сурет. Медиафайлды Дүниежүзілік өрмек көмегімен жүктеу арқылы ойнату

Браузер тұтынушы фильм атын шерткен кезден бастап әрекет жасайды. Бірінші қадамда фильм сілтемесі нұсқаған веб-серверге фильмнің HTTP-сұранысы жөнелтіледі. Екінші қадамда сервер фильмді алып (MP4 немесе басқа бір форматтағы әдеттегі файл), оны браузерге жөнелтеді.

Файлдың MIME-типі арқылы (мысалы, *video/mp4*) браузер оны қалыпқа келтіру жолын анықтайды. Біздің жағдайымызда медиаплеер пайдаланылады, ол көмекші қосымша ретінде көрсетілген, бірақ плагин қолданылуы да мүмкін. Браузер фильмді дискіден жүктеу үшін сақтайды (3-қадам). Сонан кейін медиаплеер іске қосылады, оған қабылданған фильм аты беріледі. Соңғы 4-қадамда, медиаплеер файлды оқып, фильмді ойната бастайды.

Бұл қадамдар тізбегі дұрыс. Тұтынушы фильмді көреді. Жүктеу қарапайым файлды жүктеу болып қалады, нақты уақытпен байланысты ешқандай желілік мәселелер жоқ. Жалғыз күрделі мәселе, бейне жазба алдын ала толығымен желі арқылы тасымалдануы қажет. Көптеген тұтынушылар «тапсырыс бойынша бейнені» көрмес бұрын оны бір сағат күткілері келмейді. Бұл модельді аудиоға қатысты пайдаланылған кезде де мәселелер туындайды. Егер әуен файлы 4 Мбайт болса (MP3 форматындағы әуені бар аудио файлдың әдеттегі көлемі),

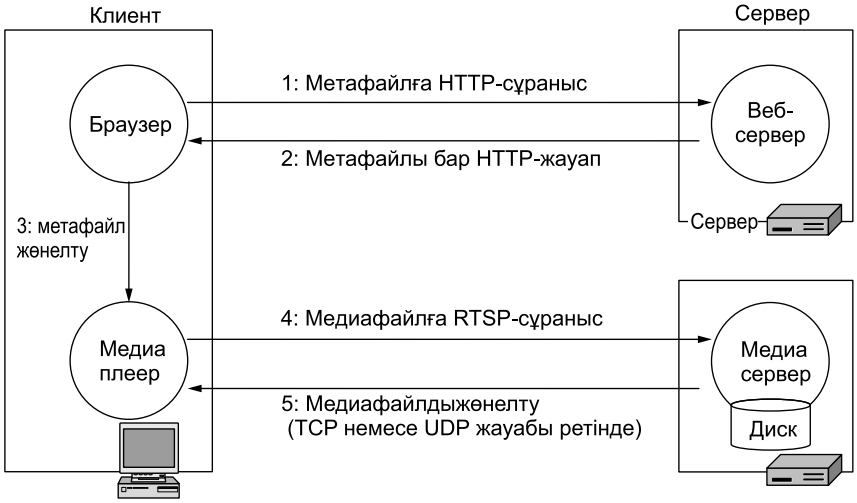
ал тұтынушы ақпаратты 1 Мбит/с жылдамдықпен қабылдайтын болса, ол әуенді тыңдамас бұрын 30 с тыныштықты тамашалайды. Әуен сүйгіштердің барлығына бірдей бұл ұнай қоймайды.

Браузер жұмысына өзгеріс енгізбей бұл мәселені қалай айналып өтуге болады? Бейне сайттар *7.27-суретте* көрсетілген схеманы пайдалана алады. Фильм аты жазылған гипперсілтемемен байланысқан файл іс жүзінде бейне файл емес **метафайл (metafile)** болып келеді. Метафайл әдетте өте қысқа. Ол тек атаудан (және бірнеше кілтті сипаттамалар болуы мүмкін) тұрады. Әдеттегі жағдайда ол шамамен:

```
rtsp://joes-movie-server/movie-0025.mp4
```

болатын бір мәтіндік жолдан тұрады.

Браузер парақты әдеттегідей қабылдайды. Сонан кейін 1- және 2-қадамдар орын алады. Келесі қадамда ойнатқышты іске қосып, әдеттегідей оған уақытша файлдың атын береді (3-қадам). Ойнатқыш уақытша файлда филімді алуға болатын URL-дің көрсетілгендігін көреді. Ол *joes-movie-server* байланысып, бейнероликті сұрайды (4-қадам). Сонан кейін фильм медиапеерге беріледі (5-қадам). Бұл амалдың артықшылығы – медиаплеер қысқа метафайл жүктегеннен кейін, жылдам іске қосылады. Бұдан кейін браузер қолданылмайды. Медиафайл плеерге тікелей беріледі және файл толығымен жүктелгенше оны көре бастауға болады.



7.27-сурет. Медиафайлды Дүниежүзілік өрмек және медиасервер көмегімен ойнату

7.27-суретте біз екі серверді көрсеттік, себебі метафайлда көрсетілген сервер жиі метафайлға сілтеме жасайтын веб-севермен сәйкес келе бермейді. Одан бетер, көбіне, ол HTTP-сервер емес, арнайы мультимедиалық сервер. Көрсетілген мысалда бұл сервер **RTSP (Real Time Streaming Protocol – нақ-**

ты уақыттық ағындық хаттама) хаттамасын пайдаланады, оны сілтемеге қарағанда түсінуге болады, ол *rtsp* – схема атынан басталады.

Мультимедиа ойнатушысы (медиаплеер):

1. Тұтынушы интерфейсін басқару;
2. Тасымалдау қателіктерін өңдеу;
3. Тығыздалған деректерді қалыпқа келтіру;
4. Синхрондау тұрақсыздығын болдырмау (джиттер) сияқты 4 негізгі мәселені шешеді.

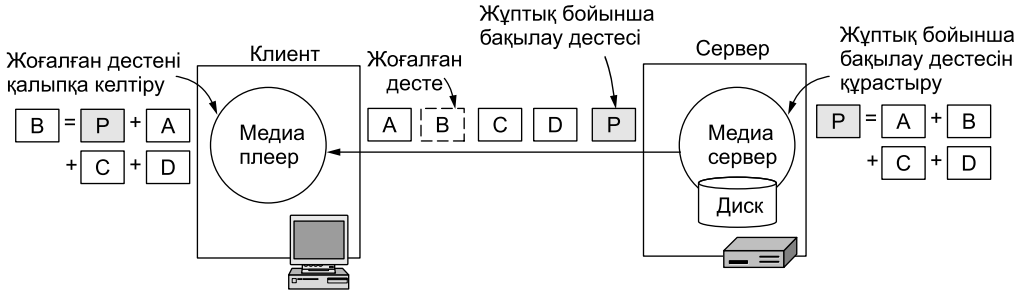
Мультимедианы қалыпқа келтіруге арналған қазіргі заманға программалардың көбінің әдемі интефейсі бар. Әдетте олардың **«скин» (skins)** деп аталатын, сыртқы пішінді өзгертуге мүмкіндік беретін басқару панелі бар. Тұтынушы өз қалауынша өзгерте алады. Біз жоғарыда айтқандай ойнатушы тұтынушымен сұхбаттасуды жүзеге асырады.

Ойнатушының басқа есептері желілік хаттамалармен байланысты. Біз тасымалдау кезіндегі қателіктерді жеңуден бастап, олардың барлығына тоқталамыз. Қателіктерді жеңу, HTTP тәрізді TCP негізіндегі транспорт немесе RTP тәрізді UDP негізіндегі транспорттың қолданылуына байланысты. Іс жүзінде оның екеуі де қолданылады. TCP негізіндегі тасымалдауды қолдану кезінде медиаплеерге қателіктерді түзетудің қажеті жоқ, себебі TCP қайта тасымалдау арқылы жоғары сенімділікті қамтамасыз етеді. Бұл қателікті жеңудің өте қарапайым жолы, кем дегенде медиаплеер үшін, бірақ соңғы қадамдарда джиттерді жоюды қыйындатады. RTP тәрізді UDP-ға негізделген тасымалдау балама нұсқаны ұсынады. Біз оны *6-тарауда* қарастырдық. Бұл хаттамалар қайта тасымалдауды пайдаланбайды. Сөйтіп, кептеліс немесе қателік салдарынан дестенің жоғалуы медианың кейбір бөлігінің жетпей қалуына әкеледі. Осы мәселені талқылап көрейік. Тасымалдау кезінде жоғалулар үлкен мәселе, себебі тұтынушы әуен немесе фильмдегі үлкен бос орындарға қуана қоймайды. Алайда, бұл жоғалулар әдеттегі файлды тасымалдау кезіндегідей күрделі емес, себебі медианың кішкене бір бөлігінің жоғалуы бүкіл файлды жоққа шығармайды. Бейне жағдайында егер секундына 25 кадрдың орнына 24 кадр ойнатылса, тұтынушы оны байқай қоймас. Аудио жағдайында кішігірім үзілістер уақыт бойынша жақын дыбыстармен маскіленеді. Тұтынушы мұқият тыңдамаса оны байқамайды.

Алайда, бұның барлығы үзілістер өте қысқа болған жағдайда ғана ақиқат болады. Тасымалдау немесе қабылдау қателігі жиі толық дестенің немесе бірнеше дестенің жоғалуына әкеледі. Медиа тасымалдау кезінде дестелер жоғалу әсерін азайту үшін: FEC және интерливинг сияқты екі стратегия қолданылады. Біз олардың әрқайсысына тоқталамыз.

FEC (Forward Error Correction – қателікті алдын-ала түзету) біз *3-тарауда* қолданбалы деңгейге қатысты қарастырған қателікті түзетуді кодтау болып келеді. Жеке алғанда оның мысалы дестеаралық бақылау (Shacham және McKenny, 1990). Жөнелтілетін деректердің әр төрт десте үшін, *7.18-суретте*

көрсетілгендей (*A*, *B*, *C* және *D* дестелері) бесінші, жұптық бойынша бақылау дестесі (parity packet) құрастырылып жөнелтілуі мүмкін. Жұптық бойынша бақылау дестесі, P жұптық бойынша бақылау қосындысы қағидасынан немесе «НЕМЕСЕ жоққа шығару» бойынша молшылық ақпараттан тұрады және мұнда әр төрт дестенің биттер қосындысы есепке алынады. Бес дестеден тұратын дестелер тобының көпшілігінің барлық дестелері жеткізіледі деп үміттенеміз. Бұл жағдайда қабылдаушы жақта жұптық бойынша бақылау дестесі еленбейді. Ал егер тек бақылау дестесі жоғалса, ол мәселе туындатпайды.



7.28-сурет. Жоғалуды болдырмас үшін жұптық бойынша бақылау дестесін пайдалану

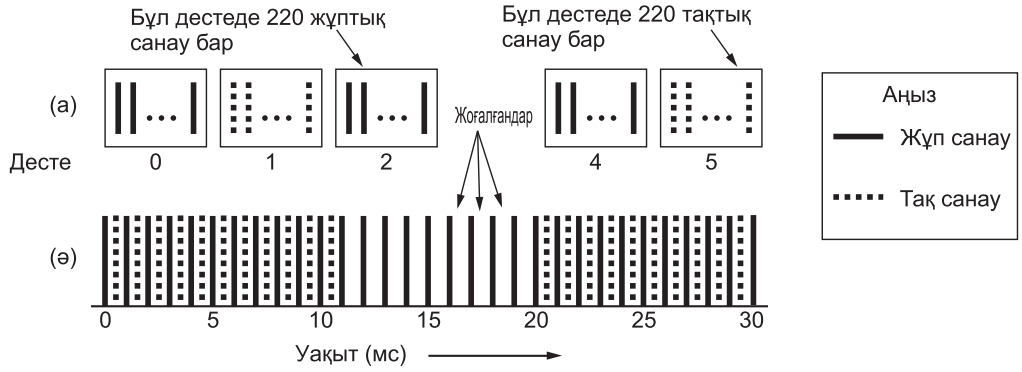
Алайда, кей жағдайда тасымалдау кезінде қандай да бір деста, мысалы, *7.28-суреттегідей* B дестесі жоғалуы мүмкін. Медиаплеер тек 3 деректер дестесін, *A*, *C, D* және P бақылау дестесін алады. Бұл схеманы пайдаланған кезде жоғалған деста биттері P дестесі деректерінің көмегімен қалыпқа келтірілуі мүмкін. Дәлірек айтсақ, егер «+» символымен «НЕМЕСЕ жоққа шығару» немесе «модуль-2 бойынша қосуды» белгілесек, B дестесін «НЕМЕСЕ жоққа шығарудың» $X+Y+Y=X$ қасиеті көмегімен $B=P+A+C+D$ формуласы арқылы қайта құрастыруға болады.

FEC медиаплеерге байқалатын жоғалу деңгейін кейбір жоғалған дестелерді қалыпқа келтіру арқылы қысқарта алады, бірақ ол белгілі бір деңгейге дейін ғана жұмыс істейді. Егер бес дестеден тұратын бір топтың екі дестесі жоғалса, онда жоғалған ақпаратты қалыпқа келтіру мүмкін емес. FEC-дің тағы бір маңызды қасиеті – жоғалудан сақтауға төленетін баға. Әр төрт деста бестікке айналады, сондықтан арна ені бойынша талап 25%-ға өседі. Қайта кодтауға кететін уақытта өседі, себебі осы уақытқа дейін жеткізілген дестелерді қайта қалыпқа келтірмес бұрын бізге бақылау дестесін күтуге тура келеді.

Жоғарыда айтылған техниканың тағы бір қызық тұсы бар. *3-тарауда* біз қателікті анықтау үшін жұптық бойынша бақылауды пайдалануды сипаттаған болатынбыз. Бұнда біз қателікті түзету жайлы айтып отырмыз. Анықтау да, түзету де қалайша мүмкін болмақ? Жауап – бұл жолы бізге қай дестенің жоғалғаны белгілі. Жоғалған деректер **қиратылған (erasure)** деп аталады. Үшінші тарауда, бірнеше биттер қателіктен тұратын кадрларды қарастырған

кезде, біз нақты қайсысы екенін білмейтінбіз. Бұл жағдай қираған дестежағдайынан қиындау. Сөйтіп, дестежоғалған кезде жұптық бойынша бақылау дестесі қателікті түзетуді қамтамасыз ете алады, ал *3-тарауда* айтылған жағдайда қателікті тек анықтау мүмкін. Жақында, көпадрестік тасымалдау жайлы сөз қозғаған кезде, біз бақылау дестесінен алуға болатын тағы бір күтпеген артықшылықпен танысамыз.

Екінші стратегия – **интерливинг (interleaving – кезектесу)** деп аталады. Бұл әдіс тасымалдау алдында медиа ретін араластыру немесе интерливингтеу және оны қабылдаған кезде сұрыптау немесе қайта интерливингтеуге негізделген. Сөйтіп, араластыру арқасында бірінен соң бірі орналасқан дестелер жоғалмайды және медианы ойнату кезінде бір үлкен үзіліс болмайды. Мысалы, дестеде 220 сэмплден тұруы мүмкін, әрқайсысында 16-биттік сандар жұбы бар, әдетте, бұл 5 мс әуенді ойнату үшін жеткілікті. Егер осы үлгілер рет-ретімен жөнелтілген болса, дестенің жоғалуы ойнату кезінде 5 мс үзіліске әкеледі. Оның орнына тасымалдау *7.29-суретте* көрсетілгендей жүзеге асырылады. 10 мс аралығындағы барлық жұп сэмплдер бір дестеде, ал тақтары бір дестеде жөнелтіледі. Енді үшінші дестенің жоғалуы әуендегі 5 мс үзілісті емес, 10 мс ішіндегі бос және әуенмен толған қысқа аралықтардың алмасуын білдіреді. Егер плеер алдыңғы және келесі үлгілерді пайдаланып интерполяцияны пайдаланатын болса, жоғалуды оңай жеңуге болады. Нәтижесінде біз 10 мс аралығындағы уақытша жоғалуды аламыз, бірақ айтарлықтай үзіліс болмайды.



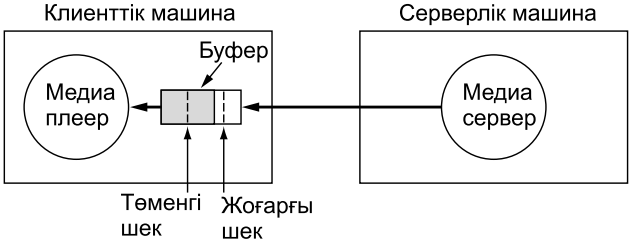
7.29-сурет. Дестеде тек жұп немесе тақ мысалдар тасымалданған кезде, айыруды уақытша төмендетеді, бірақ үзіліс тудырмайды

Интерливингтің бұл схемасы тек тығыздау жоқ кезде жұмыс істейді. Алайда, егер тығыздалған ағында сэмплдер шекарасын анықтау мүмкін болса, онда интерливинг схемасын (жеке сэмплдер емес, уақыт кезеңдері үшін) тығыздаудан кейін де пайдалануға болады. RFC 3119 тығыздалған аудиомен жұмыс істейтін схеманы ұсынады.

Интерливинг ұнамды техника, себебі оны қолданған кезде FEC-дегідей өткізу арнасының қосымша өткізгіштік қабілеттілігі қажет емес. Алайда, ол да FEC тәрізді күту уақытының ұлғаюына әкеледі, себебі бірнеше дестенің жеткізілуін күту керек (сэмплдерді рет-ретімен орналастыру үшін).

Медиаплеердің үшінші мәселесі – тығыздалған деректерді қалпына келтіру. Бұл есеп есептеу тұрғысынан жіті болғанымен өте қарапайым. Қиындық тек медианы қайта кодтауда, егер желілік хаттама тасымалдау кезінде қателіктерді түзетпесе. Көптеген тығыздау схемаларында тығыздау кезінде пайдаланылған, алдыңғы келген деректер қайта кодталмаса, кейіннен келетін деректер де қайта кодталмайды. Деректерді UDP негізінде тасымалдаған кезде дестелер жоғалуы мүмкін. Сөйтіп, кодтау үдерісі дестенің жоғалуына қарамастан қайта кодтау мүмкін болатындай етіп ұйымдастырылуы тиіс. Осы талаптар себінен I-, P- және B-кадрлар қолданылады. Алдыңғы кадрлардың кез келгенінің жоғалуын ескеру үшін әр I-кадр басқаларына тәуелсіз қайта кодталуы мүмкін.

Төртінші мәселе – нақты уақыт режимінде жұмыс істейтін барлық жүйелердің соры джиттерді жою. Біз *6.4.3-бөлімде* сипаттаған жалпы шешім буферді пайдалануға негізделген. Барлық ағындық жүйелер *7.30-суретте* көрсетілгендей, ойнату алдында 5-10 с медианы буферлеуден бастайды. Аудио анық, ал бейне секірмес үшін ойнату барысында медиа үнемі буферден алынады. Бастау алдындағы кідіріс буферге **төменгі шектен (low-water mark)** төмен түсуге мүмкіндік береді. Буфер ешуақытта бос болмау үшін деректер ұдайы келіп тұруы тиіс. Егер буфер бос болып қалса, ойнату тоқталады. Буферлеудің артықшылығы – жаңа деректер кептеліс салдарынан кешіккен жағдайда да буферленген медиадеректерді жаңа деректер келіп, буфер толғанша қалыпты жағдайда ойнатуға болады.



7.30-сурет. Медиаплеер медиасерверден келген кіріс ақпаратты буферлейді және медианы тікелей желіден емес, буферден ойнатады

Қанша деректің буферленуі керектігі және медиасервердің буферді қаншалықты жылдам толтыратындығы қандай транспорттың пайдаланылатындығына байланысты: TCP немесе UDP.

Айталық, RTP тәрізді UDP негізіндегі транспорт қолданылады делік. Әрі қарай өткізу жолағының ені аздаған жоғалулар және аздаған қосымша

ақпаратпен, медиасерверден медиаплеерге дестелерді тасымалдауға жеткілікті делік. Бұл жағдайда дестелер медиа ойналатын жылдамдықпен тасымалдануы мүмкін. Әр дестe желі арқылы тасымалданады және ойнату алдындағы бастапқы кідірістен кейін, медиаплеер медианы ойната алатындай қажет уақытта келіп отырады. Ақпараттық тек біраз бөлігін ғана буферлеу қажет, себебі кідіріс тұрақты. Егер интерливинг немесе FEC пайдаланылатын болса, онда үлкенірек бөлікті, кем дегенде жеке қайта кодтауға болмайтын дестелер жиынтығын буферлеу қажет болады. Алайда, буферленген ақпарат саны тек болмашы ұлғаяды.

Өкінішке орай, бұл сценарий екі себеппен мүмкін емес. Біріншіден, арна ені желілік маршрутқа байланысты өзгеріп отырады, сондықтан медиасервер үшін тасымалдау басталмай тұрып өткізу жолағының жеткіліктілігі белгісіз. Қарапайым шешім – медианы бірнеше айыруда кодтау және тұтынушыға өз Интернеті үшін жарамды айыруды таңдауға мүмкіндік беру. Жиі тек екі деңгей қолданылады: жоғары сапа, айталық 1,5 Мбит/с немесе үлкен не кіші сапа, айталық 512 Кбит/с немесе оданда төмен.

Екіншіден, біршама джиттер болады немесе медиа бөлігінің жөнелтуші серверден медиаплеерге дейінгі жолды өткенше неше уақыт кетеді. Бұл джиттер екі себеппен орын алады. Желіде жиі бірнеше бәсекелес ағындар тасымалданады. Бұл тұтынушы бір веб-парақта фильм қарай отырып, басқа веб-парақты да қарайды. Мұндай трафик медианы алу жылдамдығында толқыныс тудырады. Бұдан бетер, бізге дестелер емес, бейнекадрлар және аудио бөліктерін жеткізу маңызды. Компрессия арқасында бейне кадрлар мазмұнына байланысты үлкен немесе кіші болуы мүмкін. Тұрақты пейзажды тасымалдағанға қарағанда, қозғалысты тасымалдау үшін әдетте, бір биттен көп қажет. Егер желінің өткізгіштік қабілеттілігі тұрақты болса, белгілі бір уақыт кезеңінде жеткізілетін медиа саны өзгеруі мүмкін. Джиттер немесе кідірісті өзгерісі үлкен болған сайын, буферде сақталатын ақпарат та көп болуы тиіс.

Енді медианы жөнелту үшін пайдаланылатын транспорт HTTP тәрізді TCP-ға негізделген делік. Оларды ретімен орналастыру үшін қайта тасымалдау және дестелердің жеткізілуін күту арқылы TCP медиаплеердегі джиттерді ұлғайтады, тіпті, едәуір болуы да мүмкін. Нәтижесі – үлкен көлемдегі буфер және онда сақталған үлкен көлемдегі ақпарат. Бұның өз артықшылығы бар. TCP деректерді желіде тасымалдау мүмкін жылдамдықпен жөнелтеді. Кейде ойнату, жоғалту орын алған жағдайда тоқталуы мүмкін. Бірақ көп жағдайда, желі деректерді медиаплеер файлды ойнатын жылдамдықтан жылдам жеткізеді. Бұл аралықтарда буфер толтырылады және алдағы уақыттағы кідірістерді болдырмайды. Егер желі жылдамдығы медианы ойнатуға қажет жылдамдықтан жоғары болса, жиі осылай болады да, медиаплеер іске қосылған кезде буфер жылдам толады және алдағы уақытта ол босап қалмайды.

TCP немесе UDP пайдаланғанда және жылдамдық медианы ойнатуға қажет жылдамдықтан жоғары болғанда, медиафайлдың қаншалықты үлкен бөлігін медиаплеер буферге жүктейді. Файл толығымен жиі жүктеледі.

Алайда, осыншалықты үлкен бөлікті буферге жүктеу қажет емес, сонымен бірге ол сақтау үшін үлкен орын керек және буфердің босап қалуын болдырмау міндет емес. Бұл қажет емес болған жағдайда медиаплеер буферді толтырудың **жоғарғы шегін (high-water mark)** анықтайды. Бұның мәні - сервер деректерді буфер жоғарғы шекке дейін толғанша жөнелтеді. Бұдан кейін медиаплеер жөнелтуді тоқтатуды сұрайды. Тоқтатуға сұраныс жеткенше деректер келуін жалғастыра беретін болғандықтан, жоғарғы шек және буфер соңы арасындағы арақашықтық қандайда бір Х-тан осы уақыт аралығында жөнелтіліп үлгерілетін деректер саны үлкен болуы керек. Х өткізгіштік қабілеттіліктің шектелуі салдарынан орын алатын арнадағы кідіріске сәйкес келеді (демек, өткізгіштік қабілеттіліктің кідіріске көбейтіндісі). Сервер тасымалдауды тоқтатқаннан кейін буфер босай бастайды. Буфердегі деректер төменгі шекке жеткен кезде ойнатушы мультимедиа серверін тасымалдауды қайта бастауын сұрайды. Буфердің бос болып қалуын болдырмас үшін, төменгі шекті есептеген кезде де өткізгіштік қабілеттіліктің кідіріске көбейтіндісін есепке алу керек және осыған сәйкес серверге тасымалдауды қайталауға сұраныс жөнелту уақытын да есептеуі тиіс.

Медиа ағынын бастау және тоқтату үшін ойнатушы серверді қашықтықтан басқаруды жүзеге асыруы тиіс. Ол сәйкес басқару механизмі ұсынатын RTSP хаттамасыммен қамтамасыз етіледі. Ол RFC 2326-да анықталған. Қалыпқа кетірудің басталуынан және тоқтауынан басқа файл белгілі бір позицияға дейін артқа және алдыға айналдырылуы, сонымен бірге жылдамдатылған немесе баяулатылған ойнату пайдаланылуы мүмкін. Алайда, бұл стандарт деректер ағынын анықтамайды, дегенмен, әдетте, бұл UDP арқылы RTP немесе TCP арқылы HTTP. RTSP негізгі командалары *7.16-кестед*е келтірілген. Олар HTTP-мәлімдеме тәрізді қарапайым мәтіндік форматта келтірілген және әдетте, TCP арқылы тасымалданады. RTSP UDP арқылы да жұмыс істей алады, себебі әр команда расталады (егер ол расталмаса қайта жөнелтіледі).

Нақты уақыт трафигіне TCP жарамайтындай болып көрінгенімен, іс жүзінде ол жиі қолданылады. Негізгі себеп оның UDP-ға қарағанда желіаралық экрандар арқылы жеңіл өтуінде, әсіресе, егер ол HTTP-порт арқылы іске қосылса.

7.16-кесте. Ойнатушының серверге жөнелтетін RTSP командалары

Команда	Сервер әрекеті
DESCRIBE	Мультимедиалық деректер параметрлерін көрсетеді
SETUP	Ойнатушы және сервер арасында логикалық байланысты орнатады
PLAY	Клиентке деректер жөнелтуді бастайды
RECORD	Клиенттен деректерді қабылдауды бастайды
PAUSE	Деректер тасымалдауды тоқтады
TEARDOWN	Логикалық байланысты тоқтады

Көптеген әкімшіліктер желіаралық экранды сырттан орынсыз кірулерден қорғауға болатындай етіп баптайды. Көп жағдайда 80-ші порт (Дүниежүзілік өрмекке арналған HTTP) арқылы TCP байланыс орнатуға рұқсат беріледі. Бұл портты оқшаулау әпсәтте тұтынушылар наразылығына әкеледі. Алайда, көптеген порттар оқшауланады, тіпті, RTSP және RTP пайдаланатын 554-ші және 5004-ші порттарды қоса. Сөйтіп, желіаралық экран арқылы сигнал жөнелтудің қарапайым әдісі – веб-сайттың әдеттегі HTTP-жауап (кем дегенде, желіаралық экран үшін) жөнелтуші HTTP-сервер бола қалуы. TCP-дің басқа да артықшылықтары бар. Ол сенімділікті қамтамасыз ететін болғандықтан, клиентке медианық дәл көшірмесін ұсынады. Бұл тұтынушыға, деректердің жоғалуынан қорықпай, қаралған кадрды жеңіл кері айналдыруға мүмкіндік береді. Соңында, TCP файлдың ең үлкен бөлігін ең үлкен мүмкін деген жылдамдықпен буферлейді. Буфердегі орын арзан болғанда (мысалы, сақтау үшін дискі пайдаланылатын болса), медиаплеер медианы көру барысында жүктей алады. Жүктеу аяқталған кезде тұтынушы фильмді үзіліссіз көре алады, тіпті, байланыс үзілген жағдайда да. Бұл қасиет мобильді телефондарды пайдаланған жағдайда пайдалы, себебі қосылу мүмкіндігі қозғалыс кезінде күрт өзгеруі мүмкін.

TCP кемшілігі – көру басындағы күту уақытын ұлғайту (TCP-дің іске қосылуынан), сонымен бірге төменгі шектің жоғарылығы. Алайда, бұл тасымалдау жылдамдығы қалыпқа келтіру жылдамдығынан едәуір жоғары болған жағдайда сирек қиындық туғызады.

7.4.4. Медианы нақты уақытта тасымалдау

Дүниежүзілік өрмекте тек сақталған медиа ғана танымал емес. Нақты уақыттағы медианы тасымалдау да үлкен сұранысқа ие. Аудионы Интернет арқылы тасымалдау мүмкін бола бастағаннан кейін, коммерциялық радио және телестанциялар оны пайдалана бастады. Жақын арада Интернет арқылы хабарланатын студенттік станциялар пайда болды. *Студенттер* жекеменшік бағдарламаларын жүргізе бастады.

Қазір адамдар және кез келген көлемдегі компаниялар аудио және бейнені тасымалдайды. Нақты уақытта хабарлау инновация бесігі болды, жаңа стандарттар және технологиялар пайда болды. Интернет арқылы жанды хабарлауды негізінен телестанциялар пайдаланады. Оны **IPTV (IP Tele Vision – IP-теледидар)** деп атайды. Мұндай хабарлауды радиостанциялар да, мысалы, BBC пайдаланады. Ол **интернет-радио** деп аталады. IPTV де, интернет-радио да түрлі оқиғаларды, мода көрсетілімдерінен бастап, футболдан халықаралық чемпионаттарын және крикеттен іріктеу жарыстарына дейін, бүкіл әлемге хабарлайды. IP арқылы жанды хабарлауды кабельдік теледидар провайдерлері жекеменшік хабарлау жүйесі технологиясы ретінде пайдаланады. Оны жиі үлкендер мен жануарлар паркіне дейінгі аз бюджетті жобалар сайттары да

пайдаланады. Қазіргі заманғы технологиялармен әркім жанды хабарлауды жылдам әрі арзан бастай алады.

Жанды хабарлаудың бір әдісі бағдарламаларды дискіге жазуға негізделген. Көрермендер сервер мұрағатына қосылып, кез келген бағдарламаны жүктеп, тыңдай алады. **Ішкі каст (podcast)** - осындай жолмен алынған файл. Кесте бойынша жүретін бағдарламалар үшін, контентті бағдарлама аяқтағаннан кейін сақтау да мүмкін, сондықтан мұрағат жанды хабарлаудан жарты сағатқа немесе оданда аз уақытқа қалып отырады.

Іс жүзінде бұл амал мультимедианы тасымалдаумен толық сәйкес келеді. Біз талқылаған техникалардың барлығын жеңіл пайдалануға болады және тұтынушылар өздеріне ұнаған бағдарламаны мұрағаттан ала алады.

Келесі әдіс Интернет арқылы жанды хабарлау. Көрермендер жүріп жатқан медиа ағынға теледидардан жүріп жатқан бағдарламаға қосылғандай қосылады. Алайда, медиаплеерлер тоқтату немесе артқа бұрау тәрізді қосымша мүмкіндіктерді ұсынады. Жанды медиаконтент, тұтынушы көруге дайын болғанша плеермен тасымалдануын және буферленуін жалғастырады. Браузер тұрғысынан бұл сақталған бейнені тасымалдаумен бірдей. Плеерге контенттің нақты уақытта жөнетіліп жатқаны немесе ақпарат көзінің файл екендігі маңызды емес. Әдетте, плеер оны анықтай алмайды да (жанды тасымалдаудағы жалғыз ерекшелік – контентті алдыға айналдыру мүмкін емес).

Механизмнің біз жоғарыда талқылағанмен ұқсастығын ескере отырып, біздің әңгіміміздің көптеген бөлігі осы жағдайға байланысты болады, алайда, маңызды ерекшеліктер де бар.

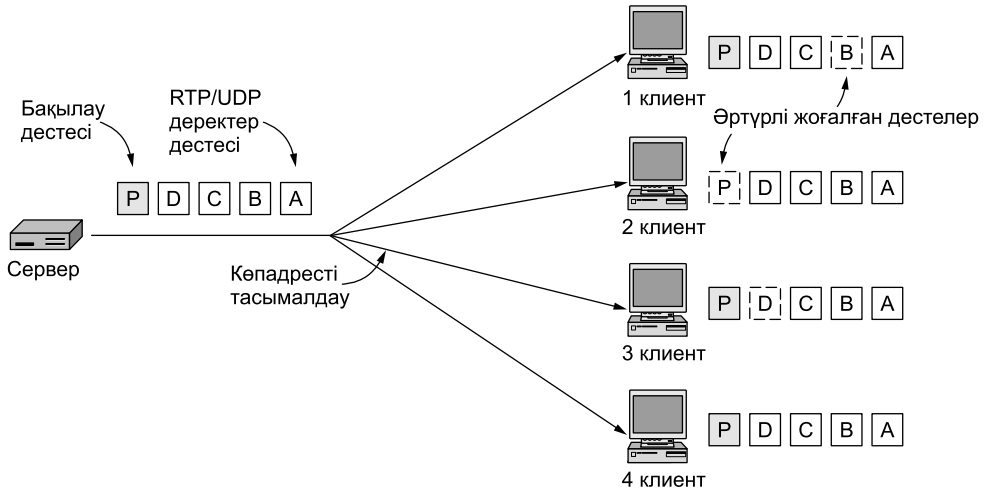
Синхрондау тұрақсыздығын (джиттер) тегістеу үшін клиент жақта буферлеудің әлі де маңызды екенін айта кету керек. Іс жүзінде контентті нақты уақытта тасымалдаған кезде деректердің үлкен бөлігін буферлеуге тура келеді (бұл тұтынушының қалыпқа келтіруді тоқтатуы мүмкін екендігімен байланысты). Файл тасымалданып жатқан кезде медиа қалыпқа келтіруге қажет жылдамдықтан үлкен жылдамдықпен берілуі мүмкін. Бұл жағдайда буфер жылдам толады және джиттердің орнын толтырады (буферлеу қажет болмаса плеер ағынды тоқтатады). Контентті нақты уақытта тасымалдаудың бұдан ерекшелігі контентті қабылдау жылдамдығы (қалыпқа келтіру) контентті жөнелту жылдамдығымен (генерациялау) сәйкес келеді. Контент бұдан жылдам жөнелтіле алмайды. Демек, мүмкін деген джиттерді игеру үшін, буфер үлкен болуы керек. Әдетте, қалыпқа келтіру басының кідірісі 10-15 с, бұл күрделі мәселе болып саналады.

Келесі маңызды ерекшелік, әдетте, нақты уақытта беріліп жатқан оқиғаларды бірмезгілде мыңдаған телекөрермендер қарайды. Бұл жағдайда мәселенің шешімі – топтық адрестеуді пайдалану. Сақталған медиафайлдарды ағындық тасымалдау кезінде бұның қажеті жоқ, себебі тұтынушылар әртүрлі контентті әртүрлі уақытта сұрайды. Сөйтіп, көптеген тұтынушыларға тасымалдау тасымалдаудың бір уақытта жүретін көптеген жеке сессияларынан тұрады.

Көпадресті ағындық трансляциялау схемасы келесідей жұмыс істейді. Сервер, IP бойынша көпадресті тасымалдауды (IP multicast) пайдаланып, топтағы адресаттарға әр дестені бір рет жөнелтеді. Бағдарламаға қосылғысы келген барлық клиенттер топқа, медиасерверге RTSP-мәлімдеме жөнелту арқылы емес, IGMP арқылы қосылады, себебі медиасервер бұл уақытта жанды ағынды тасымалдап жатады (бірінші клиент қосылған жағдайдан басқа жағдайда). Ағын жергілікті болуы үшін де осы қажет.

Көпадресті тасымалдау бірден көпке жеткізу қызметі болғандықтан, медиа UDP-транспорт арқылы RTP-дестелерде беріледі. TCP тек бір жөнелтуші және бір қабылдаушы арасында жұмыс істейді. UDP сенімділікті қамтамасыз етпейтін болғандықтан кейбір дестелер жоғалуы мүмкін. Жоғалулар деңгейін тиісті деңгейге дейін төмендету үшін біз FEC және интерливингті пайдалана аламыз.

FEC пайдаланған жағдайда, *7.31-суретте* бақылау дестесі мысалымен көрсетілгендей, көпадресті тасымалдаумен пайдалы әрекеттесу орын алады. Дестелер көптеген адрестерге тасымалданған кезде әртүрлі клиентте әртүрлі дестелерді жоғалтуы мүмкін. Мысалы, 1-клиент B дестесін, 2-клиент P бақылау дестесін, 3-клиент *D* дестесін жоғалтты, ал 4-клиент ешбір дестені жоғалтқан жоқ.



7.31-сурет. Бақылау дестесі бар медианы көпадресті тасымалдау

Алайда, клиенттердің әртүрлі дестелерді жоғалтқанына қарамастан, бұл мысалда барлық клиенттер жоғалтқан ақпаратты қалыпқа келтіре алады. Мұнда тек әр клиенттің бір дестеден артық жоғалтпауы маңызды, сонда жоғалған десте қабылданған ақпарат негізінде қалыпқа келтіріледі. Нонненмахер және басқалар (1997) осы идеяны сенімділікті жоғарылату үшін қалай пайдалану керектігін сипаттады.

Клиенттер саны көп сервер үшін медианы RTP- және UDP-дестелерде көпадресті тасымалдау ең тиімді әдіс болып саналады. Әйтпесе, сервер N клиент болғанда N ағынды тасымалдауға тиіс болады, бұл сервер жақтан желінің үлкен өткізгіштік қабілеттілігін қажет етеді.

Сіздердің тағалуларыңыз мүмкін, бірақ іс жүзінде Интернет бұлай жұмыс істемейді. Әдетте, әр тұтынушы сервермен TCP-байланыс орнатады және медиа осы байланыс арқылы тасымалданады. Клиент үшін бұл сақталған медиафайлды тасымалдаумен бірдей болып көрінеді. Дәл осы жағдайдағыдай, осындай сәтсіз шешімнің бірнеше себебі бар.

Бірінші себеп, IP арқылы көпадресті тасымалдау Интернетте кеңінен таралмаған. Кейбір интернет-провайдерлер және желілер оны қолдайды, бірақ әдетте ол желі шекарасынан тысқары қолжетімді, бұл ауқымды хабарлауға тосқауыл қояды. Келесі себеп, біз ертеде талқылаған TCP-дің UDP алдындағы артықшылығы. TCP арқылы тасымалдауды іс жүзінде Интернеттің барлық тұтынушылары қабылдай алады, әсіресе, егер хабарлау желіаралық экрандар үшін HTTP тәрізді жүрсе, ал медианы сенімді жеткізу тұтынушыларға контентті жеңіл артқа айналдыруға мүмкіндік береді.

Дегенмен, UDP және көпадресті тасымалдау пайдаланылатын провайдер желісінің ішінде маңызды бір жағдай бар. Мысалы, кабельді хабарлаумен айналысатын компания теледидар арналарын тұтынушыға дәстүрлі телехабарлау орнына IP-технологияны пайдаланып, теледидар приставкасы арқылы жөнелтуі мүмкін. IP-ді бейне бағдарламаларды тасымалдау үшін пайдалану, біз жоғарыда айтқандай жалпы жағдайда IPTV деп аталады. Компания өз желісін толығымен қадағалайтын болғандықтан, ол желіні көпадресті тасымалдауды қолдайтындай және оның арнасының өткізгіштік қабілеттілігі UDP негізінде хабарлауға жеткілікті болатындай етіп құрастыруы мүмкін. Бұның барлығы клиентке байқалмайды, себебі IP-технология **еркін қозғалуға арналған аудан** ішінде жұмыс істейді.

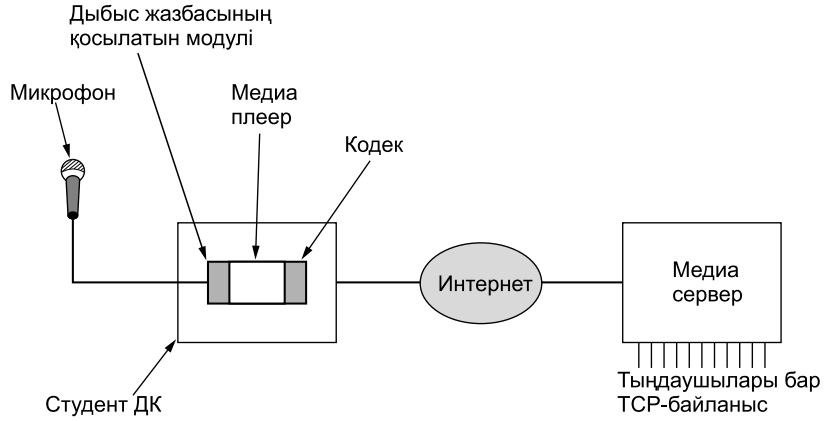
Қызмет көрсету тұрғысынан мұндай желі қарапайым кабельдік теледидар тәрізді көрінеді, бірақ хабарлау UDP жұмыс істейтін компьютер, ал теледидар компьютерге жалғанған жай манитор рөлін атқаратын телеприставка көмегімен, IP арқылы жүргізіледі.

Интернет жағдайына қайта оралсақ, TCP арқылы нақты уақыт режимінде тасымалдау кемшілігі – сервердің әр клиентке медианың жеке көшірмесін жөнелту қажеттілігінде. Бұл клиенттер саны көп болмаған, әсіресе, аудионы тасымалдаған жағдайда мүмкін, келеңсіз жағдай – арнаның өткізгіштік қабілеттілігі жеткілікті болу үшін, сервердің Интернетке жақсы қосылуын қамтамасыз ету. Бұл үй серверін пайдалануды емес, провайдерден серверді жалға алуды болжайды. Мұндай қызметтер нарығы кең, сондықтан баға үлкен болмайды.

Іс жүзінде тіпті, студент медиасервер орнатып, оны радиостанция тәрізді басқара алады. Мұндай станцияның негізгі компоненттері *7.32-суретте* көрсетілген. Негізгі компонент сапалы дыбыс картасы және микрофоны бар

әдеттегі дербес компьютер. Әртүрлі форматтағы (мысалы, MP4) аудионы қабылдап, кодтау үшін танымал ПЖ пайдаланылады, ал тыңдау үшін әдетте, медиаплеер колданылады.

Станция құрастыратын аудио деректер ағыны желіге жақсы жалғанған немесе ішкі каста тәрізді Интернеттегі мультимедиа серверге, сақталған файлдарды немесе нақты уақытта тасымалдау үшін жөнелтіледі. Сервер медианы көптеген TCP-байланыстар арқылы таратады. Онда сонымен бірге станция жайлы және қолжетімді контентке сілтемесі бар парақтары бар веб-сайт орналасады. Барлық қажет компоненттері бар коммерциялық программалар және сонымен бірге icecast тәрізді ашық программалық жабдықтамалар да кездеседі.



7.32-сурет. Студент радиостанциясы

Алайда, клиенттер саны көп болған жағдайда, бір серверден әр клиентке медианы тасымалдау үшін TCP-ді пайдалану мүмкін емес. Ағындық хабарлаудың үлкен сайттары үшін, клиент ең жақынына қосыла ататындай, географиялық тұрғыдан бытырап орналасқан бірнеше сервер пайдаланылады. Бұл схема, біз осы тараудың соңында талқылайтын, контентті тарату желісі болып саналады.

7.4.5. Нақты уақыттағы конференциялар

Ерте уақытта дауыстық қоңыраулар жалпы қолданыстағы коммутациялық телефондық желілер арқылы берілетін. Сонан кейін Интернет және Дүниежүзілік өрмек пайда болды. Жылдар өте тасымалданатын деректер көлемі өсе бастады, ал 1999 жылы телефон желісіндегі деректер көлемі және сөздер теңесті (екеуінде секундтағы биттер санымен өлшеуге болады, себебі қазір телефон жүйесінің түбінде сандық PCM-кодтау жатыр). 2002 жылға дейін ақпараттық

трафик реті сөздік трафиктен үлкен болды және оның өсуі (экспоненциалды!) жалғасып келеді. Ал сөздік трафик іс жүзінде өзгеріссіз сақталып отыр.

Осының нәтижесінде телефон желілерін Интернет ығыстыра бастады. Бүгінде дауыстық трафик интернет-технологияларды пайдалану арқылы тасымалданады және желілік өткізгіштік жолағының аз ғана бөлігін алады. Бұл технология **IP-телефония (VoIP – Voice over IP, дауысты IP үстімен тасымалдау)** немесе **интернет-телефония** деген атпен танымал.

IP-телефония бірнеше форматта қолданылады, оған экономикалық факторлар әсер етеді (Демек: оны ақшаны үнемдейтін болғандықтан пайдаланады).

Формалардың бірі әдеттегі телефонға ұқсас (ескі сәнді?) Ethernet-ке қосылып, желі арқылы шақырулар жөнелтеді. Осы модельді оқу жобасы үшін құрастырған кезде Пер Андерсон (Pehr Anderson) және оның жолдастары M.I.T. соңғы курсының студенттері болатын. Өз жобалары үшін олар «B» бағасын алды. Бұған қанағаттанбай, олар 1996 жылы NBX компаниясының негізін қалады. Компания IP-телефонияның осы түрінің бірінші ашушысы болды және үш жылдан кейін 3Com-ды 90 млн долларға сатты. Компаниялар бұл әдісті жақсы көреді, себебі ол оларға жеке телефон тораптарынан құтылып, бар желілерді пайдалануға мүмкіндік береді.

Келесі бір әдіс – халықаралық қоңыраулар үшін телефон желісін құрастырғанда IP технологиясын пайдалану. АҚШ тәрізді елдерде мұндай желілер алыс арақашықтыққа балама байланыс үшін қолжетімді, ол үшін нөмір алдында арнайы кодты теру жеткілікті. Дауыстық сигналдар (сэмплдер) желіге жөнелтілетін дестелерге орналастырылады және желіден шыққаннан кейін дестіден алынады. IP-құралдар әдеттегі телекоммуникациялық құралдардан әлдеқайда арзан болғандықтан, қызмет түрі де арзан болып келеді.

Артқа шегініс ретінде: бағадағы айырмашылық тек техникалық тұрғыдан ғана емес. Көптеген ондаған жылдар бойы телефон байланысының қызметі, телефон компанияларына олардың шығындары үшін белгілі бір көлемдегі кіріс пайызына кепілдік беріп келген, реттелуші монополия болды. Олардың шығынды өсіріп отырғаны таңғаларлық жағдай емес, мысалы, үлкен көлемдегі молшылық техникалық құралдарды сенімділікті жоғарылатумен ақтап отырды (телефон жүйесіндегі мүмкін деген тұрып қалулар 40 жыл ішінде екі сағаттан аспады немесе жылына орта есеппен үш минут). Бұл әсерді жиі «алтын жалатқан телефон бағанасы» деп атады. Реттеу тоқталғаннан кейін, әрине, бұл әсер азайды, бірақ мұрағатталып қалған құралдар әлі де бар. Ақпараттық технологиялар индустриясы ешуақытта мұндай жолмен әрекет еткен емес, сондықтан ол үнемі әлдеқайда тиімді және ешқандай артық заттан тұрған емес.

Алайда, біз тұтынушыларға әлдеқайда көрнекті IP-телефония формасына тоқталамыз: бір компьютерді басқа бір компьютерге қоңырау шалу үшін пайдалану. Мұндай форма танымал бола бастады, себебі мониторлар микрофондармен, динамиктермен, камералармен, сонымен бірге аудио және бейне файлдарды өңдеу үшін жылдам процессорлармен жабдықтала бастады, сөйтіп адамдар Интернетке үйде отырып, өткізгіштік қабілеттілігі жоғары арналар

арқылы қосыла бастады. Танымал мысал – 2003 жылы пайда болған Skype программалық жабдықтамасы. Skype және басқа да компанияларда әдеттегі телефон нөмірлеріне де, IP-адресі бар компьютерлерге де жеңіл қоңырау шалуға мүмкіндік береді.

Желінің өткізгіштік қабілеттілігі өскен сайын дауыстық шақыруларға бейне шақырулар қосылды. Бастапқыда бейне шақырулар компания шеңберінде орындалды. Бейне конференция жүргізу жүйесі екі немесе бірнеше жерлер арасында бейне алмасу үшін құрастырылды, бұл әртүрлі жерде жүрген қызметкерлерге жиналыс кезінде бір-бірін көруге мүмкіндік берді. Алайда, Интернетке кеңжолақтық қосылу және бейнені тығыздау программалық жабдықтамалары арқасында, бейне конференция үйде де мүмкін бола бастады. Тек қана аудиоға арналған Skype тәрізді құралдардың қазір бейнені шақыру мүмкіндігі де бар, сөйтіп, әлемнің әр бұрышындағы достар және туыстар бір-бірін тыңдап қана қоймай, көре де алады.

Біздің көзқарасымыз бойынша, Интернеттегі дауыстық және бейне шақырулар ағындық мультимедиа мәселесіне жатады, алайда, олардың сақталған файлды немесе оқиғаны тікелей трансляциялауға қарағанда үлкен шектеулері бар. Қосымша шектеу – екі жақтық әңгімелесуге қажет төмен күту уақыты. Телефон желісі қолайлы пайдалану үшін 150 мс-ге дейін біржақтық күту уақытын қамтамасыз ете алады, үлкен кідіріс білініп, қатысушыларды мазаландыра бастайды (Халықаралық шақырулардың күту уақыты 400 мс-ге дейін болуы мүмкін, бұл параметр бойынша олар тұтынушыға ыңғайлы болудан алшақ).

Мұндай төмен күту уақытына қолжеткізу өте қиын. Әрине, мультимедианы 5-10 с буферлеу көмектеспейді (себебі ол спорттық оқиғаны радио арқылы трансляциялауда жұмыс істейді). Бұның орнына бейне және дауыстық IP-телефония күту уақытын кішірейтудің әртүрлі тәсілдерін пайдалану арқылы құрастырылуы тиіс. Бұл мәселе TCP орнына UDP-ны таңдауға әкеледі, сондықтан тексерілген TCP тасымалдары кідіріске кем дегенде екі еселенген бір өтуді қосады. Файлда кейбір күту уақыттарын тіпті, UDP пайдаланғанда да кішірейту мүмкін емес. Мысалы, Сиэтл және Амстердам арақашықтығы шамамен, 8000 км. Бұл арақашықтық үшін оптыталшық бойымен сәуле жылдамдығымен таралу кідірісі 40 мс құрайды. Осы кідірісті қысқартқысы келген адамға табыс тілейміз! Іс жүзінде желі арқылы таралу кідірісі бұдан да үлкен болады, себебі бұдан да көп арақашықтықты өтеді (биттер үлкен шеңбер доғасымен қозғалады) және тасымалдау кідірісі бар, себебі әр IP-маршруттауыш дестені сақтап, қайта жөнелтеді. Бұл бекітілген кідірістер жалпы күту уақытының көп бөлігін алады.

Келесі кідіріс көзі деста көлемімен байланысты. Әдетте, үлкен дестені пайдалану, желілік өткізгіштік қабілеттілікті пайдаланудың ең жақсы жолы, сондықтан олар тиімдірек. Алайда, дискреттеу жиілігі 64 Кбит/с дыбыстық сигнал үшін көлемі 1 Кбайт деста 125 мс уақытта толатын болады (егер санау тығыз болса, тіпті, одан да көп). Мұндай кідіріс жалпы күту уақытынан асып

кетер еді. Бұдан басқа, егер 1 Кбайт десте жылдамдығы 1 Мбит/с кеңжолақтық арна бойымен жөнелтілетін болса, онда тасымалдау 8 мс алады. Сонан кейін десте екінші беттегі кеңжолақтық байланыс арқылы өту үшін тағы 8 мс қосылады. Үлкен дестелердің жарамсыз екені түсінікті.

Бұның орнына IP-телефония жүйелері күту уақытын өткізгіштік қабілеттілікті тиімді пайдалану есебінен қысқарту үшін, қысқа дестелерді пайдаланады. Олар дыбыстық сануларды кіші бөліктермен жүктейді, әдетте, 20 мс. Жылдамдық 64 Кбит/с болғанда бұл 160 байт деректер және тығыздалғанда одан да кіші. Бұндай дестелерді пайдаланғанда кідіріс 20 мс құрайды. Тасымалдау кідірісі де аз болады, себебі десте қысқа. Біздің мысалымызда шамамен, 1 мс-қа дейін қысқарады. Қысқа дестелерді пайдаланған кезде бір бағыттағы Сиэтл-Амстердам дестесі үшін ең кіші кідіріс жарамды 181 мс-тан (40+125+16) жарамды 62 мс-қа (40+20+2) дейін.

Біз әлі программалық жабдықтамалар уақыттының қосымша шығындары жайлы айтқан жоқпыз, ол да жарамды кідірістің бір бөлігін шығындайды. Бұл ерекшелік бейне үшін ақиқат, себебі бар өткізгіштік қабілеттілікке сәйкес болу үшін бейнені тығыздау қажет. Ең үлкен тығыздау деңгейін қамтамасыз ететін сақталған файлды ағындық тасымалдауға қарағанда, мұнда күрделі есептеуі бар кодерге уақыт жоқ. Екеуі де – кодер де, декодер де жылдам әрекет етуі тиіс.

Қазіргі заманғы медиасэмплдерді ойнату үшін буферлеу әлі де қажет (түсініксіз дыбысты немесе үзік-үзік бейне болмас үшін), бірақ буфердегі деректер көп болмау тиіс, себебі қалған жарамды кідіріс уақыты миллисекундпен өлшенеді.

Егер десте ұзақ уақыт келмесе, ойнатушы жоқ сануларды қалдырып кетеді, тұтынушыға дестенің жоғалғанын білдірмес үшін айналадағы шуды немесе фрагментті қайталауы мүмкін. Джиттерді басқару үшін пайдаланылатын буфер көлемі және жоғалған мультимедиа арасында келесі қажет. Кіші көлемдегі буфер күту уақытын қысқартады, бірақ джиттер салдарынан болатын жоғалуларды ұлғайтады. Сөйтіп, буфер көлемі кіші болған сайын, жоғалуларды тұтынушы сезетін болады.

Аңғарымпаз оқырман біз бұл тарауда желілік деңгей хаттамасы жайлы ешнәрсе айтпағанымызды байқаған болар. Желі күту уақытын немесе қызмет көрсету сапасын қамтамасыз ету механизмдерін пайдаланып джиттерді азайтуы мүмкін. Бұл мәселенің осыған дейін көтерілмеу себебі – ағындық тасымалдау қолданыстағы күту уақытымен орындалуы мүмкін, тіпті, жанды трансляциялау кезінде де. Егер күту уақыты маңызды болмаса, джиттер мәселесін шешу үшін түйін (хост) жақтағы буфер жеткілікті. Алайда, нақты уақыттағы конференция кезінде рұқсат етілген кідіріс аумағында қалу үшін, әдетте, желінің күтуді де, джиттерді де (күруді өзгерту ретінде) қысқартқаны маңызды. Бұл тек желілік өткізгіштік қабілеттілік жоғары және әркім жақсы қызмет сапасын алатын кезде ғана маңызды емес.

Біз *5-тарауда* осы мақсатқа қолжеткізетін, қызмет сапасын қамтамасыз етудің екі механизмін сипаттадық. Бірінші механизм – бұл дестелердің әртүр-

лі кластарға жататындығын белгілеп, желі шекарасында әртүрлі өңдеуге ие болатын, дифференциалды қызмет көрсету. IP-телефония дестелері үшін сәйкес белгілеулер – төмен кідіріс. Іс жүзінде жүйе жалпы белгілі мәнге дифференциалды қызмет көрсету кодын: қызмет көрсету класы – *«Тығыз жөнелту» (Expedited Forwarding);* қызмет көрсету типі – *«Төмен кідіріс» (Low Delay)* деп орнатады Бұл қолжеткізудің кеңжолақтық арнасы үшін өте пайдалы, себебі бұл арналар веб немесе басқа трафик байланысты пайдалану үшін бәсекеге түсетін кезде асыра жүктелуі мүмкін. Тұрақты желілік жолмен анықталатын кідіріс және джиттер кептеліспен ұлғаяды. 1 кбайт әр дестені арна арқылы 1 Мбит/с тасымалдау үшін 8 мс қажет және IP-телефонияның дестесі кезекте веб-трафиктен кейін тұрса, бұл кідірісті өзіне қабылдайды. Алайда, егер IP-телефония дестесінің төмен кідіріс маркері бар болса, онда олар веб-трафикті айналып кезек алдына тұрады және күту уақытын азайтады.

Кідірісті қысқартуға мүмкіндік беретін екінші механизм – өткізгіштік қабілеттіліктің жеткілікті екеніне көз жеткізу. Егер қолжетімді өткізгіштік қабілеттілік тұрақты емес, тасымалданатын деректер ағынының жылдамдығы өзгеретін (тығыздалған бейнедегідей) және кейде өткізгіштік қабілеттілік жеткіліксіз болатын болса, онда кезек және кідіріс ұлғаяды. Бұл тіпті, дифференциалды қызмет көрсетумен де орын алады. Жеткілікті өткізгіштік қабілеттілікке кепілдік беру үшін, желі бөлігі қорға қойылуы мүмкін. Мұндай мүмкіндік біріктіріп қызмет көрсетумен қамтамасыз етіледі. Өкінішке орай, ол кеңінен таралмаған. Оның орнына желілер тасымалданатын күтілулі ақпараттар көлемі үшін жобаланады немесе клиенттерге белгілі бір деңгейдегі ақпарат көлемін тасымалдау үшін қызмет көрсету деңгейі жайлы келісім ұсынады. Кептеліс және қажетсіз кідірістерді болдырмас үшін қосымшалар осы деңгейден төмен әрекет етуі тиіс. Үйде күнделікті бейне конференция өткізу үшін тұтынушы желі өткізгіштік қабілеттілігімен анықталатын бейне сапасын немесе желілік жолды тексеріп, сәйкес сапаны автоматты таңдайтын программалық жабдықтаманы таңдай алады.

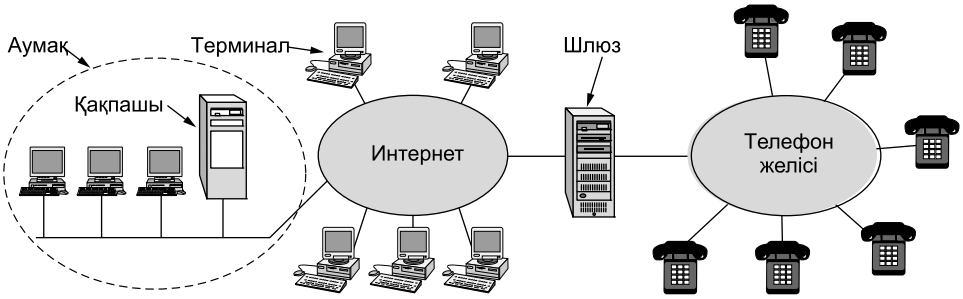
Жоғарыда келтірілген факторлардың кез келгені күту уақытын жарамды ете алады, сондықтан нақты уақыттағы конференц-байланыс әрқайсысына көңіл бөлуді қажет етеді. IP-телефонияның қысқа шолуы және осы факторлардың сараптамасын Goode (2002) қараңыз.

Енді ағындық мультимедиа үшін күту уақыты мәселесін талқылағаннан кейін, біз конференция жүргізу жүйесінің келесі маңызды мәселесіне көшеміз. Бұл – шақыруларды қалай орнатып, қалай аяқтау мәселелері. Біз осы мақсатта кеңінен қолданылатын екі хаттаманы қарастырамыз – H.323 және SIP. Skype – бұл тағы бір маңызды жүйе, бірақ оның ішкі құрылымы жабық.

H.323

Дауыстық және бейне шақырулар Интернет арқылы жүзеге асырыла бастағанша, егер әр өндіруші өз хаттамалар стегін құрастыратын болса, жүйенің

ешуақытта жұмыс істемейтіні түсінікті болған. Осы мәселе орын алмас үшін, қызығушылық танытқандар Халықаралық телекоммуникация одағы (ITU) астында бірігіп, бірегей стандартты құрастыруды бастады. 1996 жылы ITU «Бейнетелефон жүйелері және кепілденген қызмет сапасын ұсынбайтын жергілікті есептеу желілерінің құрылғылары» тақырыбымен **H.323** индексі бар ұсыныстарды шығарды. Мұдай атау тек телефон индустриясында ғана туындауы мүмкін еді. Бұл ұсыныстар 1998 жылы қайта қаралып, H.323-тің жаңа нұсқасы «Дестелерге негізделген мультимедиа-коммуникация жүйелері» деген атпен шықты.



7.33-сурет. Интернет-телефония үшін H.323 құрылымының моделі

H.323 нақты хаттаманы сипаттамайды, керісінше, интернет-телефония құрылымы жайлы жалпы түсініктеме береді. Құжаттан сөзді кодтауға, байланыс орнатуға, сигнал, деректер және т.б. тасымалдауға арналған әртүрлі арнайы хаттамаларға көптеген сілтемелер табуға болады, бірақ олардың сипаттамасы келтірілмейді. Жалпылама модель *7.33-суретте* бейнеленген. Ортада, Интернетті телефон желісімен байланыстыратын **шлюз (gateway)** орналасқан. Ол Интернет жақтан H.323 хаттамасын және «телефон» жақтан жалпы қолданыстағы коммутацияланатын телефон желісі хаттамасын қолдайды. Коммуникациялық құрылғылар **терминалдар** деп аталады. Жергілікті есептеу желісінде өз заң шеңберінде (аумағында) орналасқан соңғы түйінді басқаратын, **қақпашы-машина (gatekeeper)** болуы мүмкін.

Телефон желісінің жұмысын көптеген хаттамалар қамтамасыз етеді. Біріншіден, аудио және бейнені кодтайтын және қайта кодтайтын хаттама қажет. Бір дауыстық арнаны стандартты телефондық ұсыну, ағыны 64 Кбит/с (жиілігі секундына 8000 жиілік санауына 8 бит) сандық аудио тәрізді кодталады, ол **G.711-де** анықталған. H.323 жүйелерінің барлығы G.711 қолдауы міндетті. Осылай болса да, сөзді кодтаудың басқа хаттамаларын да қолдауға рұқсат етілген (бірақ міндетті емес). Олар тығыздаудың басқа алгоритмдерін пайдаланады және сапа мен өткізгіштік қабілеттіліктің біршама өзгеше келісіміне әкеледі. Бейне үшін, біз ертеде H.264 қоса талқылаған MPEG тығыздау формасы қолданылады.

Бірнеше тығыздау алгоритмдерін пайдалану рұқсат етілгендіктен, терминалдарға осы хаттамалардың бірін пайдалануға келісімге келетін арнайы

хаттама қажет. Мұндай хаттама **H.245** деп аталады. Ол сонымен бірге байланыстың басқа да параметрлерді, мысалы, биттік жылдамдықты келістіруге мүмкіндік береді. RTP арналарын басқару үшін RTCP қажет. Бұдан басқа, байланысты орнату және үзу, тональды шақыруды қамтамасыз ету, қоңырау дауысын генерациялау және телефон жүйесінің басқа стандартты функцияларын жүзеге асыру хаттамалары қажет. Стандартты ITU **Q.931** қолданылады. Терминалдарға қақпашы-машинамен келісім жүргізу үшін (егер ол жергілікті желіде бар болса) хаттама керек. Ол үшін жүйеде **H.225** хаттамасы жұмыс істейді. Осы хаттама басқаратын, дербес компьютер және қақпашы арасындағы арна **RAS (Registration / Admission / Status – Тіркеу / Қолжеткізу / Статус)** арнасы деп аталады. Сонымен бірге ол терминалдарға аймаққа кіруге және одан шығуға, өткізгіштік қабілеттілікті сұрауға және босатуға, қалып-күй жайлы деректерді жаңартуға және т.б. мүмкіндік береді. Соңында, тікелей деректерді тасымалдау үшін хаттама қажет. Бұл аумақта UDP арқылы RTP жұмыс істейді. Ол әдеттегідей RTCP-пен басқарылады. Барлық осы хаттамалардың иерархиясы *7.34-суретте* көрсетілген.

7.34-сурет. H.323 хаттамалар стегі

Бұл хаттамалардың бір-бірімен қалай әрекеттесетінін түсіну үшін, жергілікті желі терминалы болып келген (қақпашысы бар) және қашықтықтағы телефонға қоңырау шалушы дербес компьютер (ДК) жағдайын қарастырайық. Бастапқыда ДК-ге қақпашыны табу керек, сондықтан ол 1718 порты арқылы кеңтарату тәсілімен арнайы UDP-десте таратады. Қақпашы жауабынан ДК оның IP-адресін біледі. Енді компьютер қақпашыда тіркелуі керек. Ол үшін ол қақпашыға UDP дестесінде RAS жөнелтеді. Тіркеу өткеннен кейін компьютер қақпашыға өткізгіштік қабілеттілікті қорда сақтау жайлы өтініш жасайды (RAS қол жеткізу мәлімдемесі). Тек осы ресурс бөлінгеннен кейін ғана байланыс орнатуды бастауға болады. Өткізгіштік қабілеттілікті алдын ала қорда сақтау, қақпашыға шығыс тораптағы байланыс санын шектеуге мүмкіндік береді, ол өз кезегінде қажетті қызмет сапасын қамтамасыз етуге қызмет етеді.

Қатаң айтқанда, телефон жүйелері де осы жұмысты орындайды. Сіз телефон трубкасын көтерген кезде, жергілікті абоненттік бекетке сигнал жөнелтіледі. Егер бекетте тағы бір қоңырауды өңдеуге қуат жеткілікті болса, он үздіксіз гудокты генерациялайды. Кері жағдайда сіз ешқандай дыбысты естімейсіз. Қазіргі таңда жүйенің көлемі соншалықты үлкен, сондықтан сіз үнемі үздіксіз гудокты естисіз, бірақ ертеде, телефон жаңа пайда бола бастаған кезде оған үнемі бірнеше секунд қажет болатын. Сондықтан егер сіздің немерелеріңіз телефон нөмірін термес бұрын үздіксіз гудок не үшін керек десе, сіз оларға не деп жауап беру керектігін білесіз. Алайда, ол уақытқа дейін тұрақты телефон станциялары қалмауы да мүмкін.

Енді, телефон қоңырауын жүзеге асыру үшін, ДК қақпашымен TCP-байланыс орнатады. Телефон байланысын орнату кезінде телефон желісінің байланысқа бағытталған, дәстүрлі хаттамалары қолданылады. Сондықтан TCP қажет. Екінші жағынан, телефон жүйесінде телефон аппаратына өзінің бар екендігі жайлы хабарлауға мүмкіндік беретін ешқандай RAS жоқ, сондықтан H.323 құрастырушылары RAS мәлімдемені жөнелту үшін UDP-ны да, TCP-ді де қолдана алды, бірақ олар UDP - үстеме шығындары аз хаттама – UDP таңдады.

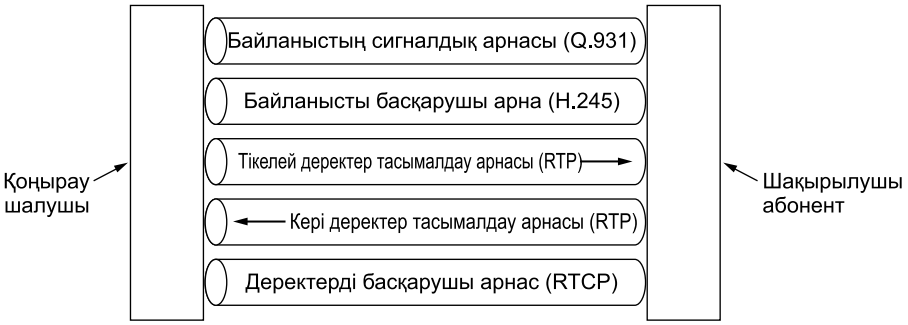
Терминалға өткізгіштік қабілеттілік бөлінген кезде ол TCP-байланыс арқылы *SETUP* мәлімдемесін жөнелтеді (Q.931). Ол жерде шақырылатын абонент нөмірі (немесе IP-адресі және порт, егер қашықтықтағы компьютер шақырылса) көрсетіледі. Қақпашы, сұраныстың сәтті қабылданған фактісін растап, *CALL PROCEDING* Q.931-мәлімдемесімен жауап қайтарады. Сонан кейін қақпашы *SETUP* мәлімдемесін шлюзге жөнелтеді.

Бір жағынан компьютер, ал екінші жағынан телефон коммутаторы болып келетін шлюз әдеттегі телефонға әдеттегі қоңырау шалуды жүзеге асырады. Шақырылушы абоненттің ақырғы телефон станциясы өзінің әдеттегі жұмысын атқарады (абоненттің телефоны сыңғырлайды) және ДК қоңыраулар сериясы басталғандығын жайлы хабардар етіп, *ALERT* Q.931-мәлімдемесін кері жөнелтеді. Абонент телефон трубкасын көтерген кезде ақырғы телефон станециясы компьютерге байланыстың орнатылғандығын хабарлап, *CONNECT* мәлімдемесін жөнелтеді.

Байланыс орнатылғаннан кейін қақпашы бұл үдеріске қатыспайды, алайда, шлюз екі жақты байланысты қамтамасыз етіп, жұмысын жалғастыра береді. Дестелер қақпашыны айналып, тікелей шлюз IP-адресіне бағытталады. Бұл жағдайды екі жақ арасындағы әдеттегі арнамен салыстыруға болады. Бұл шын мәнінде, биттер тасымалданатын, қарапайым физикалық деңгей байланысы. Екі жақты бірі де қарсы беттің қандай болып көрінетіндігін білмейді.

Байланыстың ұнамды параметрлері жайлы келісім үшін H.245 хаттамасы пайдаланылады. Бұл жерде үнемі ашық тұратын H.245-тің арнайы басқарушы арнасы қолданылады. Әр жақ өз мүмкіндіктерін хабарлаудан бастайды. Мысалы, бейнені қолдау (H.323 бейнені қолдай алады), конференц-байланыс, қолданыстағы кодек және т.б. жайлы айтуға болады. Әр жақ қарсы жақтың

мүмкіндігін білгеннен кейін, бір бағытталған екі арна ұйымдастырылады. Олармен белгілі бір кодектер байланысады және оларға белгілі бір параметрлер тағайындалады. Екі жақта әртүрлі құрылғылар орнатылуы мүмкін болғандықтан, бір бағытталған арналардың әрқайсысы өз кодекін пайдаланатын жағдай болуы мүмкін. Келісімге қол жеткеннен кейін деректерді тасымалдауды бастауға болады (RTP хаттамасы арқылы). Басқаруды асыра жүктелуді қадағалайтын RTCP жүргізеді. Егер бейне тасымалданатын болса, RTCP дыбысты және бейне тізбекті синхрондауды жүзеге асырады. *7.35-суретте* логикалық арналардың әртүрлі түрлері көрсетілген. Екі жақтың бірінде телефон трубкасы орнына қойылғаннан, қоңырау аяқталғаннан кейін қажет емес ресурстарды босату үшін, Q.931 арнасы арқылы байланыстың аяқталғандығы жайлы сигнал беріледі.



7.35-сурет. Сөйлесу кезіндегі қоңырау шалушы және шақырылушы абоненттер арасындағы логикалық арналар

Байланыс үзілгеннен кейін шақырушы ДК қақпашымен қайта байланысып, оған қорда сақталған өткізгіштік қабілеттілікті босатуды сұрап, RAS мәлімдеме жөнелту керек. Алайда, оның орнына ол жаңа қоңырауды жүзеге асыруы да мүмкін.

Біз осы уақытқа дейін H.323-тегі сапа жайлы сөз қозғаған жоқпыз, ал бұл нақты уақыттағы конференцияны сәтті жеткізудің маңызды бөлігі. Мәселе, QoS H.323 қарастыратын аумаққа кірмейтінінде. Егер деректер тасымалданатын желі ДК және шлюз арасында джиттерсіз тұрақты байланысты қамтамасыз ете алатын болса, онда біздің жолымыз болды және қызмет көрсету сапасы жақсы, ал кері жағдайды өкінішке орай сапа нашар болады. Алайда, кез келген телефон қоңырауында телефон желісінің дизайнының арқасында үзілістер болмайды.

SIP – байланыс орнату хаттамасы

H.323 стандартын ITU құрастырған. Интернет-қауымдастығында ол әдет тегі телекоммуникация өнімі: тым үлкен, күрделі және жеткілікті түрде иілгіш

емес болып көрінген. Дауысты IP үстімен тасымалдаудың қарапайым және иілгіш жүйесін жасау үшін арнайы IETF комитетін құрастыру ұйғарылды. Комитет қызметінің негізгі нәтижесі **SIP (Session Initiation Protocol – байланыс орнату хаттамасы)** болды, оның соңғы версиясы, 2002 жылы RFC 3261 сипатталған. Хаттама Интернет арқылы телефон байланысын орнату, бейнеконференциялар үшін жүйе ұйымдастыру технологиясы және басқа мультимедиалық байланыстар ұйымдастыру әдістеріне тоқталады. Толық хаттамалар жиынтығынан тұратын H.323-ттен айырмашылығы, SIP – түрлі инттернет-қосымшалармен әрекеттесе алатын бірегей модуль. Мысалы, телефон нөмірі URL түрінде анықталады, демек веб-парақтарда гиперсілтемелер орналастыруға болады, тұтынушы осы сілтемеге шертіп телефон байланысын орната алады (*mailto* схемасы шамамен, осылайша электронды хат жазып, сілтемеде көрсетілген адрес бойынша жөнелте алады).

SIP хаттамасы екіжақты байланыс орнатуға да мүмкіндік береді (демек, әдеттегі телефон байланысы), көпжақты (қатысушылардың әрқайсысы басқаларды тыңдап, сөйлей алады) және кеңтаратылымды (қатысушылардың біреуі ғана сөйлейді, ал қалғандары тыңдайды) байланыс орнатуға да мүмкіндік береді. Байланыс сеансы кезінде аудио-, бейне- және басқа деректерді тасымалдауға болады. Бұл мүмкіндік, мәселен, нақты уақытта қатысушылар саны көп желілік ойын ұйымдастырған кезде қолдануға болады. SIP тек байланысты орнату, басқару және үзумен айналысады. Деректерді тасымалдау үшін басқа хаттамалар, мысалы, RTP/RTCP. SIP – TCP және UDP үстінде жұмыс істейтін, қолданбалы деңгей хаттамасы пайдаланылады.

SIP хаттамасы шақырылушы абонентті іздеуден (осы сәтте өзінің үйдегі компьютерінен алыста болуы мүмкін) бастап, оның мүмкіндігін анықтау, телефон байланысын орнату және үзу механизмдерін қолдау тәрізді түрлі қызметтер ұсынады. Қарапайым жағдайда SIP қоңырау шалушы және шақырылушы абоненттер компьютерлері арасында байланыс сеансын орнатады. Нақты осы жағдайды біз төменде қарастырамыз.

SIP телефон нөмірлері *sip* схемасымен URL түрінде беріледі. Мысалы, *sip:ilse@cs.university.edu* сізді хосты *cs.university.edu* DNS-атымен анықталатын, Ilse атты тұтынушымен байланыстырады. Сонымен бірге, SIP URL-і IPv4, IPv6 форматындағы адрестерден немесе нақты телефон нөмірлерінен тұруы мүмкін.

SIP хаттамасы мәтіндік болып келеді, ол HTTP моделінде құрастырылған. Екі жақтың бірі бірінші жолында тәсіл аты көрсетілген ASCII-мәлімдеме жөнелтеді, ал төменірек параметрлерді беруге арналған тақырыптардан тұратын қосымша жолдар орналасады. Көптеген тақырыптар MIME стандартынан алынған, бұл SIP-ке қолданыстағы интернет-қосымшалармен әрекеттесуге мүмкіндік береді. Базалық спецификациямен анықталатын алты тәсіл *7.17-кестеде* келтірілген.

Байланыс сеансын орнату үшін қоңырау шалушы шақырылушы абонентпен TCP-байланыс орнатып, оған *INVITE* мәлімдемесін жөнелтуі, не

осы мәлімдемені UDP-дестеде жөнелтуі тиіс. Екі жағдайда да, екінші және келесі жолдардың барлығында орналасқан тақырыптар, қоңырау шалушының мүмкіндіктері жайлы ақпарат, мультимедиа типі және форматтарынан тұратын мәлімдеме денесінің құрылымын сипаттайды. Егер шақырылушы абонент қоңырауды қабылдаса, ол жауап ретінде, HTTP-ге ұқсас үш орынды нәтиже кодын жөнелтеді (осы кодтар тобы *7.13-кестеде* келтірілген, 200 коды қоңыраудың қабылданғанын білдіреді). Нәтиже коды бар жолдан кейін қоңырау шалушы абонент, сонымен бірге, өз мүмкіндіктері жайлы деректерді, мультимедиа және оның форматтарын хабарлауы мүмкін.

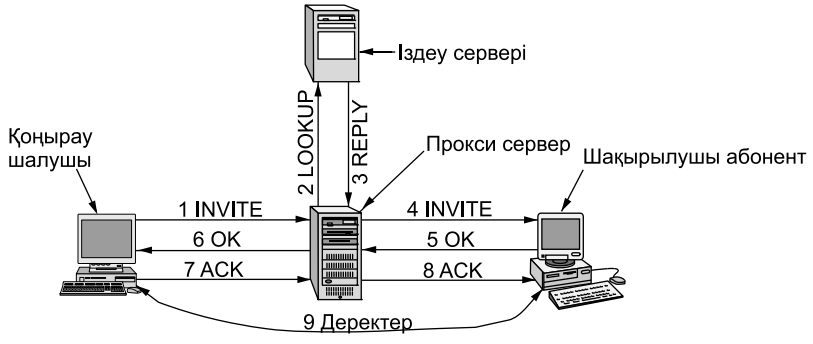
7.17-кесте. SIP тәсілдері

Тәсіл	Сипаттамасы
INVITE	Байланыс сеансын орнатуға сұраныс
ACK	Сеанстың орнатылғанын растау
BYE	Сеанстың аяқталуына сұраныс
OPTIONS	Хост мүмкіндіктерін сұрау
CANCEL	Сұранысты болдырмау
REGISTER	Қайта адрестеу серверіне тұтынушының ағымдағы орны жайлы хабарлау

Тұтынушымен байланыс үштік қолалысу хаттамасы бойынша орнатылады, қоңырау шалушы хаттама жұмысын аяқтау үшін *ACK* және 200 кодын қабылдау растауын жөнелтеді.

Екі жақтың кез келгені сеанстың аяқталғандығы жайлы сұраныс жөнелте алады, ол үшін *BYE* тәсілі қолданылады. Қарсы беттен растау алынғаннан кейін сеанс аяқталған болып саналады.

OPTIONS тәсілі машина мүмкіндіктерін сұрау үшін пайдаланылады. Әдетте бұл қоңырау шалушы бет ойлағандай сеанс типінің (мысалы, IP үстімен дауыс тасымалдау) қолданатындығын анықтау үшін, байланыс сеансын іске қосар алдында орындалады.



7.36-сурет. Прокси-сервер және SIP хаттамасының қайта адрестеуін пайдалану

REGISTER тәсілі SIP хаттамасының мүмкіндіктеріне жатады, тұтынушы үйде жоқ болса да, оны іздеп, байланыс орнатады. Осы тәсіл көрсетілген мәлімдеме, осы сәтте кімнің қайда жүргендігі жайлы ақпаратты сақтайтын, SIP іздеу серверіне (location server) жөнелтіледі. Кейіннен осы сервер көмегімен абонентті табуға талпынуға болады. Осы жағдайда қолданылатын қайта адрестеу операциясы *7.36-суретте* көрсетілген. Бұл суреттен біз қоңырау шалушының прокси-серверге *INVITE* мәлімдемесін жөнелткенін көреміз. Бұл мүмкін қайта адрестеуді байқалмайтындай етеді. Прокси абонентті іздеп табуға талпынады және оған табылған адрес бойынша *INVITE* мәлімдемесін жөнелтеді. Әрі қарайғы қарым-қатынас үштік қолалысу кезіндегі мәлімдемелер тізбегін коммутациялауды білдіреді. *LOOKUP* және *REPLY* мәлімдемелері SIP хаттамасына кірмейді, бұл сатыда, іздеу серверінің типіне байланысты, кез келген жарамды хаттаманы пайдалануға болады.

SIP хаттамасының біз бұл жерде толығымен сипатталмаған, басқа да көптеген мүмкіндіктері бар. Олардың ішінде шақыруды күту, қоңырауды бейнелеу, қоңырау шалушыны шифрлау және сәйкестендіру функциясы бар. Бұдан басқа, Интернет және телефон жүйесі арасында сәйкес шлюз бар болса, компьютерден әдеттегі телефонға қоңырау шалуға мүмкіндік бар.

H.323 және SIP салыстырмалы сараптамасы

H.323 және SIP екі жақты да, көп жақты да байланысты қолдайды. Соңғы құрал ретінде компьютер де, әдеттегі телефон да болуы мүмкін. Екі жағдайда да екі жақ параметрлер, деректерді шифрлауды және RTP/RTCP хаттамалар мүмкіндігі жайлы алдын ала келіседі. Жинақталған салыстырмалы *7.18-кестесі* барлық ұқсастықтар мен айырмашылықтарды көрсетеді.

Ұқсас қасиеттер және сипаттамалар жиынтығына қарамастан, хаттамалар концепциясы және философиясы бойынша ерекшеленеді. H.323 – телефон индустриясына тән нағыз ауыр салмақты стандарт. Ол толық хаттамалар стегін сипаттайды және ненің рұқсат етілгенін, ненің рұқсат етілмегенін дәл сипаттайды. Мұндай көзқарас, әр сатыда жақсы анықталған хаттамаларға әкеледі, сөйтіп желілер арасындағы әрекеттесуді жеңілдетеді. Алайда, бұның құны – болашақта пайда болатын қосымшаларға қиын сәйкестендірілетін, үлкен, күрделі және қатаң стандарт.

SIP, керісінше, жұмысы қысқа мәтіндік мәлімдемелермен алмасуға негізделген, әдеттегі интернет-хаттама. Бұл – Интернеттің басқа хаттамалармен жеңіл әрекеттесетін кішігірім модул, алайда, телефон жүйесінің қолданыстағы сигнал хаттамаларымен нашар сәйкестенеді. Деректер тасымалдау жүйесінің моделі IETF ұсынған IP үсті болғандықтан, модульдік қағада пайдаланылады, ол қажетті түрде итілгіш және жаңа қосымшаларға жеңіл сәйкестендіріледі. Бұл хаттаманың кемшілігі – стандартты белгілеуде интерпретациялауды пайдалану салдарынан, желіаралық әрекеттесу кезінде қиындықтар туындауы мүмкін.

7.18-кесте. H.323 және SIP салыстыру

Аспект	H.323	SIP
Құрастырушы	ITU	IETF
Телефон жүйесімен үйлесімдік	Толық	Көп жағдайда
Интернетпен үйлесімдік	Көп уақыттан бері бар	Бар
Құрылым	Монолитті	Модульді
Аяқталғандық	Толық хаттамалар стегі	SIP тек байланыс орнатуды қамтамасыз етеді
Параметрлер жайлы келіссөздер жүргізу	Екі жақта жүргізеді	Екі жақта жүргізеді
Шақыру кезіндегі сигнал	TCP үстіндегі Q.931	TCP немесе UDP үстіндегі SIP
Мәлімдемелер форматы	Екілік	ASCII
Мультимедиалық деректерді тасымалдау	RTP/ RTCP	RTP/ RTCP
Көпжақты байланыс	Бар	Бар
Мультимедиалық конференциялар	Мүмкін	Мүмкін емес
Адрестеу	URL немесе телефон нөмірі	URL
Байланысты үзу	Анық немесе TCP-байланысты үзу	Анық немесе тайм-аут бойынша
Мәлімдемелермен алмасу (ins-tant messaging)	Жоқ	Бар
Деректерді шифрлау	Бар	Бар
Стандартты сипаттау көлемі	1400 парақ	250 парақ
Жүзеге асыру	Тым үлкен және күрделі	Орташа
Статус	Кең таралған, әсіресе, бейне	Әсіресе, сөз үшін жақсы балама

7.5. КОНТЕНТТІ ЖЕТКІЗУ

Интернет, телефон желісі тәрізді, коммуникацияның барлық аумағын жаулап алуға ұмтылды. Алдымен, академиктер есептерді орындау үшін желі арқылы тіркеліп, жеке машиналармен байланысатын. Адамдар бір-бірімен байланысу үшін электронды поштаны ұзақ қолданды және қазір бейне- және дауыстық IP-телефонияны қолданады. Алайда, Дүниежүзілік өрмектің өсуімен Интернет коммуникация құралынан гөрі, контентті сақтаушы кор болды. Көптеген адамдар Дүниежүзілік өрмекті қажет ақпараты табу үшін пайдаланады және фильмдер, әуендер және программаларға ортақ қолжеткізуді ұсынатын үлкен көлемдегі файл алмасу желілері бар. Акценті контентпен ығыстыру анық бола бастады, қазір Интернеттің өткізгіштік қабілеттілігінің үлкен бөлігі сақталған бейнені тасымалдауға жұмсалады. Контентті тарату мәселесі коммуникация

мәселесінен өзгеше болғандықтан олар желіге басқа талаптар қояды. Мысалы, егер Салли Итумен сөйлескісі келсе, ол оның мобильді телефонына IP-телефония арқылы қоңырау шалады. Коммуникация белгілі бір компьютермен болуы керек, Пол компьютеріне қоңырау шалудың мағынасы жоқ. Бірақ егер Иту өз командасының крикет бойынша соңғы матчын көргісі келсе, ол оны осы бейнені ұсынған кез келген компьютерден алуға қуанышты болады. Оған бұл кімнің компьютері, Саллидікі ме немесе Полдыкі ме немесе Интернеттегі белгісіз біреудің компьютері ме маңызды емес. Сөйтіп, жұмыс сапасына әсер ететіні (және заңдылық) болмаса, контент үшін мекен жай маңызды емес.

Тағы бір ерекшелік – контентті ұсынатын кейбір веб-түйіндер ерекше танымал болып кетті. Бұның жарқын мысалы – YouTube. Бұл сайт тұтынушыларға кез келген ойға келмейтін тақырыпта, өздері жасаған бейнелерімен бөлісуге мүмкіндік береді. Көптеген адамдар оны жасағысы келеді. Қалғандары көргісі келеді. YouTube осы өткізгіштік қабілеттілікке талабы жоғары бейненің, күнделікті интернет-трафиктің 10%-ын құрайтын бөлігін шығарады. Осындай сұраныс деңгейін басқару үшін, ешбір сервер жеткілікті қуаттылық және сенімділік деңгейін қамтамасыз ете алмайды. Оның орнына YouTube және басқа да үлкен контент ұсынушылар өздерінің жеке контент тарату желілерін құрастырады. Бұл желілер көптеген клиенттерге жақсы өнімділік және қолжетімділікті қамтамасыз етіп, контентті ұсыну үшін, бүкіл әлем бойынша орналасқан деректер сақтау орталықтарын пайдаланады.

Контентті тарату үшін пайдаланылатын тәсілдер уақыт өте дамыды. Өрмек алғашқы өсе бастаған кезде, оның танымалдығы оның жойылып кетуіне әкелді. Контентке деген өспелі сұраныс сандары себебінен сервер және желілер жиі асыра жүктеледі. Адамдар WWW-ді World Wide Wait (Дүниежүзілік Күту) дей бастады. Тұтынушылар сұранысына жауап ретінде Интернет ядросының өткізгіштік қабілеттілігі өсті және әлдеқайда жылдам желі байланысты Интернеттің шеткі аймақтарына әкелді. Өткізгіштік қабілеттіліктің өсуі жұмысты жақсартудың кілті еді, бірақ ол шешімнің тек жартысы болатын. Шексіз кідірістерді қысқарту мақсатында, зерттеушілер контентті таратуда өткізгіштік қабілеттілікті пайдаланудың түрлі құрылымдарын дамытты.

Осы құрылымдардың бірі – **CDN (Content Distribution Network – контентті тарату желісі)**. Мұнда жеткізуші Интернеттегі машиналар жиынтығын шашыратып орналастырып, оларды клиенттерге контентті тарату үшін пайдаланады. Бұл – үлкен ойыншылар таңдауы. Баламалы таңдау - **P2P (Peer-to-Peer – пирингтік желі, теңқұқықты түйіндер желісі)** желісі. Мұнда компьютерлер жиынтығы арнайы орнатылған серверсіз немесе қандай да бір орталық басқару пункітінсіз, бір-біріне контент ұсыну үшін, өз ресурстарын ортақ қорға салады. Бұл идея адамдардың қиялын жаулап алды, себебі бірлесе отырып, көптеген кіші ойыншылар үлкен әсер туғыза алады.

Бұл бөлімде біз Интернеттегі контентті тарату мәселесін және іс жүзінде пайдаланылатын кейбір шешімдерді қарастырамыз.Мазмұн танымалдығын және интернет-трафикті қысқаша қарастырғаннан кейін, біз қуаттты веб-сер-

верлердің қалай құрастыру керектігін және веб-клиенттер үшін өнімділікті жақсартуда кэштеуді қалай пайдалану керектігін, сонан кейін біз контентті таратуға арналған екі негізгі құрылымды: CDN және P2P желілері қарастырамыз. Біз олардың құрылымы және қасиеттерінің әртүрлі екенін көреміз.

7.5.1. Контент және интернет-трафик

Желілерді жобалап, құрастыру үшін, олардың қандай трафикті тасымалдайтынын түсіну керек. Мәселе сервердегі контентке қол жеткізуге, мысалы, компания офистарынан Интернеттегі көптеген машиналары бар және желіге қосылудың тамаша мүмкіндіктерін ұсынатын деректерді сақтау және өңдеу орталықтарына ығысқан кезде. Тіпті, кішкене серверді ұстап, нақты машинамен үйде немесе Интернетке кеңжолақтық қосылуы бар офиста жұмыс істегенше, қазіргі кезде Интернеттегі деректерді сақтау және өңдеу орталықтарында виртуалды машинаны жалға алған жеңіл және арзан.

Қуанышқа орай интернет-трафикке қатысты тек екі факты білу жеткілікті. Біріншісі – ол жылдам өзгереді, тек егжей-тегжейі ғана емес толығымен де. 1994 жылға дейін трафиктің үлкен бөлігі FTP арқылы дәстүрлі файлдарды (компьютерлер арасында программа және деректердің қозғалысы үшін) және электронды поштаны тасымалдау болып келген. Сонан кейін веб пайда болып, экспоненциалды өсе бастады. Веб-трафик 2000 жылдары FTP және электронды алыс артта қалдырды. 2000 жылдарда әуендер, сонан кейін фильмдерді ұсынған P2P-желілер іске қосылды. 2003 жылға интернет-трафиктің үлкен бөлігін вебті артта қалдырып, P2P-трафик алды. Шамамен 2000 жылдың соңында контентті тарату тәсілін пайдаланатын, YouTube тәрізді сайттардан шығатын ағындық бейне P2P-ден аса бастады. Cisco жорамалы бойынша 2014 жылға дейін интернет-трафиктің 90%-ы қандай да бір формадағы бейне алатын болады (Cisco, 2010).

Трафик көлемі әркез маңызды емес. Мысалы, IP-телефония серпілісі кезінде, тіпті, 2003 жылы іске қосылған Skype-қа дейін, ол диаграммада елеусіз серпіліс болып қалған, себебі дыбысқа деген өткізгіштік қабілеттілік талабы мән бойынша, бейнеге қарағанда екі есе төмен. Десек те, IP-телефония желіні басқа қырынан жүктейді, себебі ол күту уақытына сезімтал. Басқа мысал, 2004 жылы іске қосылған Facebook-тен бастап, үздіксіз өсіп келе жатқан онлайн әлеуметтік желілер. 2010 жылы Facebook алғаш рет бір күнде Google-ден де көп тұтынушыларды алды. Тіпті, трафикті қарамағанның өзінде де әлеуметтік желілер маңызды (ал ол жерде трафик өте үлкен), себебі олар адамдардың Интернет арқылы әрекеттесуін өзгертеді.

Біз жасаған қорытынды: «сейсмикалық жылжулар» – интернет-трафиктегі маңызды өзгерістер – жылдам және жүйелі түрде болып тұрады. Келесіде не болады? Біз сіздерге оны кітаптың 6 басылымында міндетті түрде айтамыз.

Интернет-трафик жайлы екінші маңызды факт – ол өте ассиметриялы. Біз жұмыс істейтін кейбір мәндер өзінің орта шамасына жақын. Мысалы, көптеген

үлкен адамдардың бойы орташаға жақын. Біршама биік және біршама бойы қысқа адамдар бар, ал тым биік және тым қысқа адамдар өте аз. Осындай қасиеттермен елді мекеннің үлкен бөлігін жаулап алатын диапазонды табуға болады.

Интернет-трафик бұлай құрастырылмаған. Ұзақ уақыт аралығында аздаған трафигі бар кішігірім веб-түйіндер тізбегі және оданда кіші трафигі бар сайттардың үлкен бөлігі белгілі болған. Бұл ерекшелік желілерді жасау тілінің бір бөлігі. Ертедегі қозғалыс жайлы мақалалардағы **дестелер поезды (pocket trains)** терминінде, байланыс арқылы өтетін үлкен санды дестелерде экспресс-поездар идеясының бар екендігі жайлы айтылатын (Jain және Routhier, 1986). Бұл **өз-өзіне ұқсастық (selfsimilarity)** түсінігі ретінде нысандандырылған, біздің мақсатта оны тіпті, әртүрлі уақыт масштабында қарастырғанда да, көптеген ұзақ және көптеген қысқа аралықтардан тұратын желілік трафик ретінде түсінуге болады (Leland және басқалар, 1994). Кейінгі жұмыстарда трафиктің ұзын ағындарын **пілдер (elephants)**, ал қысқаларын **тышқандар (mice)** деп атайды. Негізгі ой мынада: тек бірнеше пілдер және көптеген тышқандар бар, бірақ пілдер маңызды, сол себепті олар осындай үлкен.

Веб-контентке қайтып оралсақ, әлбетте, сәйкес ассиметрия бар. Бейнені жалға алу, кітапхана және де басқа да осыған ұқсас ұйымдармен жұмыс тәжірибесі барлық фильмдердің (кітаптардың) бірдей танымал еместігін көрсетеді. Егер жалға беру пунктінде *N* фильм бар болса,онда танымалдар тізімінің *k* орындағы фильмге нақты тапсырыстың үлесі шамамен C/k екені тәжірибеде дәлелденген. Мұндағы, C – үлес қосындысын 1-ге толықтыратын сан, нақтырақ:

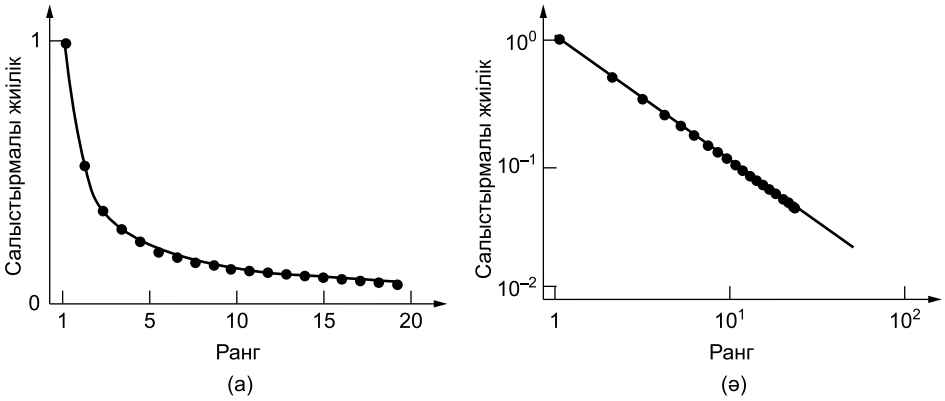
$$C = \frac{1}{(1+1/2+1/3+1/4+1/5+...+1/N)}$$

Осылайша, мысалы, ең танымал фильм танымалдық тізімінде 7-болып тұрған фильмнен 7-рет жиі алынады. Бұндай тәуелдік **Ципфа заңы (Zipf's law)** деп аталады (Zipf, 1949). Бұл заңдылық, үлкен мәтіндегі сөзді пайдалану жиілігі, оның рангіне кері пропорционал екенін байқаған, Гарвард университетінің лингвистика профессоры Джордж Ципфаның құрметіне осылай аталған. Мысалы, жиі кездесетін сөздер тізіміндегі қырқыншы сөз, сексенінші сөзге қарағанда екі есе, ал жүз жиырмасыншы сөзге қарағанда үш есе жиі кездеседі.

Ципфа таратылуы *7.37. а-суретінде* көрсетілген. Ол аздаған танымал элементтердің және көптеген танымал емес элементтердің бар екенін бейнелейді. Осындай түрдегі таратуды көру үшін деректерді *7.37. ә-суретіндегідей* екі ось бойынша да логарифмдік масштабта орналастырған дұрыс. Нәтижесінде түзу аламыз.

Веб-парақтардың танымалдығын зерттеу оның Ципфа заңдылығына шамалап сәйкес келетінін көрсетті (Breslau және басқалар, 1999). Ципфа таратылуы – **дәрежелік заңдар (power laws)** деген атпен танымал таратулардың бірі. Дәрежелік заңдар сірә адам қызметтерінің көптеген аясында бар, мыса-

лы, қала тұрғындарының таралуы және байлықты тарату. Олар сонымен бірге, бірнеше ірі ойыншылар, көптеген кішігірім ойыншылар бар жағдайды да сиипаттайды және жағдайды екі ось бойынша да логарифмдік масштабта берілген түзу ретінде бейнелейді. Кейіннен Интернет топологиясын шамалап, дәрежелік заңдылық бойынша сипаттауға болатыны белгілі болды (Faloutsos және басқалар, 1999). Сонан кейін зерттеушілер Интернеттің барлық мүмкін деген қасиеттерін логарифмдік масштабта бейнелеп, түзуді көріп, «Дәрежелік заңдылық» дейді.



7.37-сурет. Ципфа таратылуы: а – сызықтық масштаб;
ә – екі ось бойынша да логарифмдік масштаб

Алайда, логарифмдік масштабтағы түзуден гөрі, бұл таратудың желіні жобалап, пайдалану үшін нені білдіретіні маңыздырақ еді. Контенттер көбіне Ципфа таратуы бар немесе дәрежелік заңдылыққа бағынышты болғандықтан, Интернеттегі танымал веб-түйіндердің осыған ұқсас заңдылықтарға бағыныштылығы түбегейлі болып көрінеді. Бұл өз кезегінде «орташа сайт» түсінігі – сәтсіз ұсыныс екенін білдіреді. Сайттарды танымал немесе танымал емес ретінде сипаттаған дұрыс. Танымал сайттар, сірә маңызды шығар, себебі танымал сайттардың азғана бөлігі интернет-трафиктің үлкен бөлігіне жауап береді. Мүмкін бұл таңғаларлық жай емес шығар, бірақ танымал емес сайттар да маңызды. Танымал емес сайттарға бағытталған трафиктің қосындысы, жалпы трафиктің үлкен бөлігін алуы мүмкін, себебі олардың саны өте көп. Көптеген танымал емес нұсқаның қосынды мән түсінігі танымалдандырылған болатын, мысалы, *The Long Tail* («ұзын құйрық») кітабымен (Anderson, 2008a).

7.37 а-суретіндегідей, кемуді көрсететін қисықтар, әдеттегідей, бірақ олардың барлығы бірдей емес. Жеке алғанда, кему деңгейі қалған материал санына пропорционал болған жағдайлар (мысалы, тұрақсыз радиоактивті атомдардағыдай) Ципфа заңына қарағанда әлдеқайда жылдам кемитін **экспоненциалды кемуді (exponential decay)** көрсетеді. Айталық, t уақыт сәтінен кейін қалған атомдар саны әдетте, $e^{-t/\alpha}$ түрінде беріледі, мұндағы α тұрақ-

тысы кемудің қаншалықты жылдам жүретінін көрсетеді. Экспоненциалды кему және Ципфа заңдылығының арасындағы айырмашылық – бірінші жағдайда құйрық соңын есепке алмауға болады, ал Ципфа заңдылығы жағдайында оның жалпы салмағы елеулі, сондықтан есепке алмауға болмайды.

Ассиметриялы әлемде тиімді жұмыс істеу үшін, біз веб-түйінін екі түрінде құрастыра алуымыз керек. Танымал емес сайттарды басқару жеңіл. DNS-ті пайдаланып, әртүрлі көптеген сайттар іс жүзінде олардың барлығын басқарып отырған, Интернеттегі бір компьютерге нұсқауы мүмкін. Екінші жағынан, танымал сайттарды қолдау қиын. Ол үшін қуаттылығы жеткілікті бір де бір компьютер жоқ, сонымен бірге бір компьютерді пайдалану, компьютер істен шыққан жағдайда сайттың миллиондаған тұтынушылар үшін қолжетімсіз болуына әкеледі. Бұл сайттарды басқару үшін біз контентті басқару жүйесін құрастыруымыз керек. Біз оны әрі қарай қарастырамыз.

7.5.2. Серверлік фермалар және веб-прокси

Біз осы уақытқа дейін қарастырған желілік жобалар көптеген клиенттік машиналармен әрекеттесетін, бір серверлік машинадан тұратын. Жақсы сипаттамалары бар үлкен веб-түйін құрастыру үшін біз не сервер жағында, не клиент жақта өңдеу жылдамдығын өсіруіміз керек. Сервер жақта, әлдеқайда қутты веб-серверді, компьютерлер кластері бір сервер ретінде әрекет ететін серверлік ферманы ұйымдастыру арқылы тұрғызуға болады. Клиент жақта жақсы өнімділікке, жақсы кэштеу тәсілдерінің көмегімен қол жеткізуге болады. Жеке алғанда, кэштеуші прокси (веб-прокси) клиенттер тобы үшін үлкен жалпы кэшті қамтамасыз етеді.

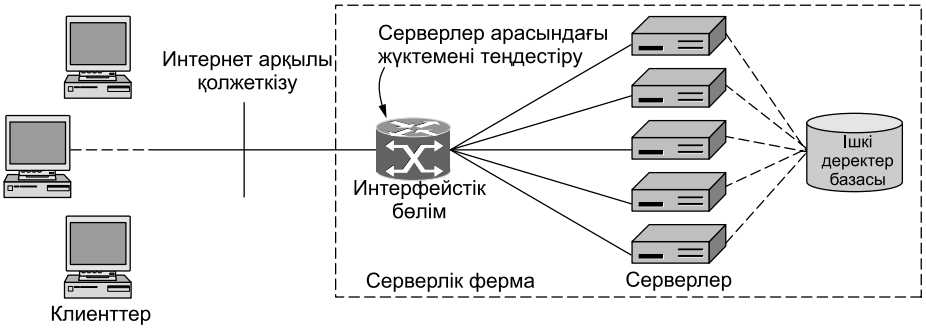
Біз осы тәсілдердің әрқайсынсын сипаттаймыз. Алайда, бұл тәсілдердің бірі де ең үлкен веб-сайт құрастыру үшін жеткіліксіз екенін айта кеткен жөн. Ең танымал сайттар үшін, әртүрлі жерде орналасқан компьютерлерді пайдаланатын, контентті тарату тәсілдері қажет. Бұл тәсілдерді біз келесі бөлімде қарастырамыз.

Серверлік фермалар

Бір машинаның қанша өткізгіштік қабілеттілігі бар болса да, ол тек желілік сұраныстың белгілі бір санына қызмет көрсете алады, сонан кейін ол асыра жүктеледі. Бұл жағдайдағы шешім – веб-сервер құрастыру үшін бірнеше компьютерді пайдалану. Бұл *7.38-суретте* көрсетілген **серверлік ферма (server farm)** моделіне әкеледі.

Бір қарағанда қарапайым болып көрінетін бұл модельдің қиындығы – серверлік ферманы құрайтын компьютерлер жиынтығы клиенттер үшін тұтас логикалық веб-сайт болып көрінуі тиіс. Кері жағдайда біз параллель жұмыс істейтін бірнеше веб-сайтты аламыз. Серверлер жиынтығының бір веб-сайт

етіп көрсетудің бірнеше жолы бар. Бұл шешімдердің барлығы серверлердің әрқайсысы кез келген клиенттен келген сұранысты өңдей алады деп болжайды. Ол үшін әр серверде веб-сайттың көшірмесі болуы керек. Осы мақсатта ортақ ішкі деректер базасы бар серверлер байланыстары үзік сызықтармен көрсетілген.



7.38-сурет. Серверлік ферма

Шешімнің бірі – серверлік ферманың серверлеріне сұранстарды тарату үшін DNS-ті пайдалану. Веб-сайт URL-і үшін DNS сұраныс жасалған кезде, DNS-сервер серверлердің өзгермелі IP-адрестерін қайтарады. Әр клиент бір IP-адреске жүгінеді, әдетте, тізімде бірінші тұрған. Сөйтіп, бастапқыда ойлағандай, әртүрлі клиент әртүрлі серверге бір веб-түйінге қол жеткізу үшін жүгінеді. DNS тәсілі CDN технологиясының негізінде жатыр, осы бөлімнің соңына қарай біз оған қайтып ораламыз.

Басқа шешімдер, келіп түскен сұраныстарды серверлік ферманың серверлері арасында тартатын **интерфейстік бөлікке (front end)** негізделген, тіпті, клиент бір тағайындалған IP-адресті пайдаланып, серверлік фермамен әрекеттескеннің өзінде де. Интерфейстік бөлік әдетте, арналық деңгейдегі желілік коммутаторды немесе IP-маршруттауышты, яғни фреймдер немесе дестелерді басқаратын құрылғыны бейнелейді. Шешімдердің барлығы осы құрылғылар (немесе серверлер) желілік немесе транспорттық деңгей немесе қосымшалар деңгейінің тақырыптарын қарап, оларды стандартталмаған жолмен пайдалануға негізделген. Веб-сұраныс және жауап TCP-байланыс ретінде тасымалданады. Дұрыс жұмыс істеу үшін интерфейстік бөлік бір сұраныстың барлық дестелерін бір серверге жөнелтуі тиіс.

Интерфейстік бөліктің қарапайым схемасы – барлық кіріс сұраныстарды барлық серверлерге жөнелту. Әр сервер алдын ала келісім бойынша, сұраныстың тек бір бөлігіне ғана жауап береді. Мысалы, 16 сервер деректер көзінің IP-адресіне қарап, IP-адрестерінің соңғы 4 биті олардың белгілі бір селекторына сәйкес келетін сұраныстарға жауап беруі мүмкін. Өткізгіштік қабілеттілікті шығындау тұрғысынан бұл ысырап болып көрінгенімен, жиі жауаптар сұранысқа қарағанда әлдеқайда ұзын болатындықтан бұның ешқандай ерсілігі жоқ.

Жалпы жағдайда интерфейстік бөлік құрылымының нұсқасы IP-, TCP- және HTTP-тақырыптарды тексеріп, оларды серверлер арасында еркін таратуы мүмкін. Тарату **жүктемені теңдестіру (load balancing)** саясаты деп аталады, себебі оның мақсаты – сервердің жұмысшы жүктемесін теңдестіру. Саясат қарапайым немесе күрделі болуы мүмкін. Қарапайым саясат серверлерді бірінен кейін бірін кезектеп немесе шеңбер бойымен қайталап пайдаланудан тұруы мүмкін. Бұл жағдайда интерфейс, бір сұраныстың бөлігі болатын әр сұраныстың бейнесін есте сақтап, оларды бір серверге жөнелтіп отыруы керек. Бұдан басқа, сайтты бір серверден сенімді ету үшін, интефейс серверлердің істен шығуы жайлы ақпаратты алып және істен шыққан серверге сұраныс жөнелтуді тоқтатып отыруы керек.

Көп жағдайда, NAT тәрізді, бұл жалпы схема қауіпті болып келеді, немесе кем дегенде, өте осал, себебі біз көпдеңгейлі хаттамалар қағидасына қайшы келетін құрылғыны жасадық: басқару мақсатында әр деңгей өзінің жеке тақырыбын пайдалануы тиіс және пайдалы жүктемедегі ақпаратты басқа мақсатта пайдаланбауы керек. Бірақ адамдар бәрібір осындай жүйелерді жобалайды және жоғары деңгейдің өзгеру салдарынан жүйе істен шықса өздері таңғалады. Бұл жағдайда интерфейс – желілік коммутатор немесе маршруттауыш, бірақ ол транспорттық немесе одан да жоғары деңгей ақпаратына сүйеніп әрекет ете алады. Мұндай құрылғы **«аралық түйін» (middlebox)** деп аталады, себебі ол хаттамалар стегіне сәйкес ешқандай жұмысы жоқ желілік жол ортасына кіріктірілген. Бұл жағдайда интерфейсті, қолданбалы деңгейге дейін барлық деңгейлерді аяқтайтын, серверлік ферманың ішкі бөлігі ретінде қарастырған дұрыс (сондықтан осы деңгейлердің тақырыбындағы барлық ақпаратты пайдалана алады).

Осылай десек те, NAT тәрізді, бұл схема іс жүзінде пайдалы. TCP-тақырыптарды қарау арқылы, тек IP ақпаратты пайдаланғаннан да жақсы жүктеме теңдестігіне қол жеткізуге болады. Мысалы, бір IP-адрес бүкіл компаниянікі болып, көптеген сұраныстар жасауы мүмкін. Тек TCP немесе жоғары деңгейлер ақпаратын қарап қана бұл сұраныстарды әртүрлі серверлерге жөнелтуге болады.

HTTP тақырыптарды қарау себебі біршама басқа. Көптеген веб-әрекеттер деректер базасына қолжеткізу және оның модификациясы болуы мүмкін, мысалы, клиент өзінің соңғы сатып алуларын қараған кезде. Бұл сұранысты алған серверге ішкі деректер базасына сұраныс жасауға тура келеді. Осы тұтныушының келесі сұраныстарын да осы серверге жөнелткен пайдалы болады, себебі осы тұтынушы жайлы ақпарат оның кэшінде бар. Бұның ең қарапайым жолы – веб-куктерді пайдалану (немесе тұтынушыларды айыруға көмектесетін басқа ақпаратты) және оларды анықтау үшін HTTP тақырыптарды қарап шығу.

Қорыта келе, біз бұл жобаны веб-сайттар үшін сипаттағанымызбен, серверлік ферманы басқа түрдегі серверлер үшін құрастыруға да болады. UDP үсті медиа ағындар сервері бұған мысал бола алады. Бұл жағдайда қажет болатын

жалғыз өзгеріс – интерфейс осы сұраныстар жүктемесі теңдестігін қолдауға қабілетті болуы тиіс (оларда веб-сұранысқа қарағанда, хаттама тақырыбының басқа өрістері болады).

Веб-прокси

Желілік сұраныстар және жауаптар HTTP пайдалану арқылы жөнелтіледі. *7.3-бөлімде* біз браузерлердің жауаптарды қалай кэштейтіндігін және оны келесі сұраныстар үшін қалай бірнеше рет қолдануға болатынын сипаттадық. Тақырыптардың түрлі өрістерін және ережелерді браузер веб-парақтың кэштелген көшірмесінің әлі де өзекті екенін анықтау үшін пайдаланылады. Бұнда біз бұл материалды қайталамаймыз.

Кэштеу жұмысты жақсартады, жауап уақытының ұзақтығын және жүктелімін қысқартады. Егер браузер кэштелген парақтың өзектілігін өзі анықтай алса, онда ол оны желілік трафиксіз, әп-сәтте кэшттен таңдап алады. Алайда, тіпті, браузер серверді парақтың өзектілігі жайлы сұраған жағдайда да жауапқа жұмсалатын уақыт және желілік жүктеме азаяды, әсіресе, үлкен парақтар үшін, себебі тек кіші мәлімдеме жөнелтіледі.

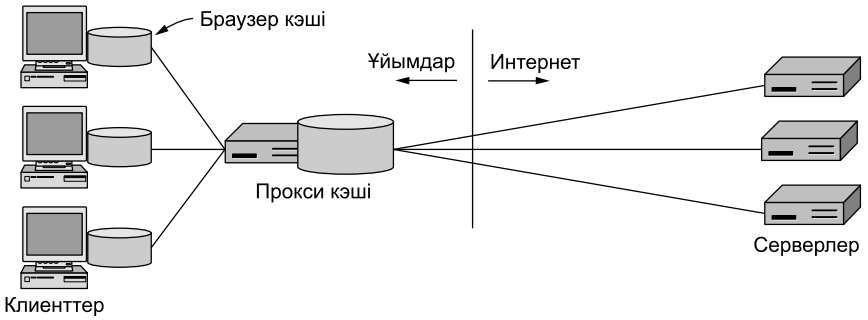
Осылай бола тұра, браузер тек тұтынушы кірген веб-парақтардың барлығын кэштей алады. Біздің танымалдық жайлы әңгімемізден, сіздер көптеген адамдар бірнеше рет кіретін, танымал парақтардың тек кішкене бөлігінен басқа көптеген танымал емес парақтар бар екені естеріңізде болар. Іс жүзінде бұл браузермен кэштеу тиімділігін шектейді, себебі нақты осы тұтынушы тек бір рет кірген көптеген парақтар бар. Бұл парақтарды үнемі серверден алу керек.

Кэшті тиімді пайдаланудың жалғыз жолы – оны бірнеше тұтынушы үшін ортақ ету. Осылайша, бір тұтынушы үшін алынған парақ, басқа тұтынушыға дәл осы параққа сұраныс жасағанда оған да қайтарылуы мүмкін. Кэштеусіз браузер екі тұтынушы үшін де парақты серверден алуы тиіс. Әрине, бұл ортақ қолжеткізу программалармен қайтарылатын, аутентификацияны қажет ететін, шифрланған трафик, парақтар және кэштелмейтін парақтар (мысалы, биржаның ағымдағы бағалары) үшін қолданылмайды. Динамикалық парақтар, әсіресе программалармен жасалған – бұл кэштеу тиімсіз және өсіп келе жатқан жағдай. Алайда, көптеген тұтынушыларға көрінетін және кімнің оларға жүгінгеніне тәуелсіз бірдей болып көрінетін веб-парақтар саны көп (мысалы, суреттер).

Веб-прокси (Web proxy) тұтынушыларға кэшке ортақ қол жеткізуді қамтамасыз ету үшін қолданылады. Прокси – бұл әлдекімнің атынан әрекет ететді, мысалы, тұтынушыны агентінің. Проксидің бірнеше түрі бар. Мысалы, ARP-прокси басқа бір жерде орналасқан (және өзі үшін жауап бере алмайтын) тұтынушы атынан жүргізілетін ARP сұраныстар үшін жауап береді. Веб-прокси веб-сұраныстарды оның тұтынушысы атынан орындайды. Ол әдетте, веб-жауаптарды кэштеуге жауап береді, ол ортақ қолданыста болғандықтан, оның браузердікінен де үлкен кэші бар.

Ұйымдар үшін проксиді пайдаланудың әдеттегі нұсқасы – барлық тұтынушылар үшін бір веб-проксиді пайдалану. Ұйымдар – компания немесе интернет-провайдер. Екі жағдайда да тұтынушылар үшін веб-сұраныс жылдамдығын ұлғайту, сонымен бірге өткізгіштік қабілеттілік қажеттілігін кеміту пайдалы. Соңғы тұтынушы үшін, әдетте, пайдалануына тәуелсіз тұрақты баға тағайындалады, ал көптеген компаниялар және интернет-провайдерлер үшін төлем ақы пайдаланылған өткізгіштік қабілеттілікке тәуелді.

Проксиді пайдалану конфигурациясы *7.39-суретте* бейнеленген. Әр браузер сұраныстарды нақты сервер парағына емес, проксиге жөнелтетіндей етіп бапталған. Егер проксиде бұл парақ бар болса, ол оны лезде қайтарады. Кері жағдайда, прокси парақты серверден алып, болашақта пайдалану үшін кэшке қосады және клиентке сұрағанын қайтарады.



7.39-сурет. Веб-браузерлер және веб-серверлер арасындағы көмекші кэш

Нақты сервер орнына сұраныстарды проксиге жөнелтуден басқа, клиенттер браузер кэшін пайдаланып, өз кэштеулерін де жүргізеді. Браузер сұранысты өз кэшінен қанағаттандыруға талпынғаннан кейін ғана проксиге жүгінеді. Осылайша, прокси кэштеудің екінші сатысын қамтамасыз етеді.

Кэштеудің қосымша деңгейлерін қамтамасыз ету үшін келесі проксилер қолданылуы мүмкін. Әр прокси (немесе браузер) өзінен **жоғары тұрған проксиге (upstream proxy)** сұраныс жолдайды. Әр жоғары орналасқан прокси өзінен **төмен орналасқан прокси (downstream proxy)** немесе браузер үшін кэштеу жүргізеді. Осылайша, компания браузерлері, интернет-провайдер проксиін пайдаланатын компания проксилерін пайдалануы, ал интернет-провайдер проксиі өз кезегінде веб-сервермен тікелей әрекеттесуі мүмкін. Алайда, іс жүзінде мүмін деген артықшылықтардың үлкен бөлігіне қолжеткізу үшін, жиі біз *7.39-суретте* көрсеткен бір сатылы прокси-кэштеу жеткілікті. Мәселе – тағы да танымалдықтың ұзын құйрығы. Веб-трафикті зерттеулер, жалпы қолданыстағы кэш, тұтынушылар саны кішігірім компания қызметкерлерінің санынан (айталық, 100 адам) аспаған жағдайда ғана тиімді екенін көрсетті. Тұтынушылар саны өскен кезде ортақ кэшті пайдаланудың тиімділігі елеусіз болады, себебі танымал емес сұраныстар сақтау кеңістігінің аздығынан кэштелмейді (Wolman және басқалар, 1999).

Веб-прокси, жиі оларды пайдалану шешімінің факторы болып келетін, қосымша артықшылықты қамтамасыз етеді. Осындай артықшылықтардың бірі – контентті сүзбеден өткізу. Әкімшілік проксиде сайттардың қара тізімін немесе жасауы немесе өзі жасайтын сұраныстарды сүзбеден өткізуі мүмкін. Мысалы, көптеген әкімшіліктер жұмыс уақытында YouTube-тен бейне тамашалайтын (немесе одан бетер, порнография) қызметкерлерді жақтырмайды және сәйкес сүзбе орнатады. Проксидің тағы бір артықшылығы – прокси тұтынушының бірегейлігін серверден қорғаған кездегі құпиялық және анонимдік.

7.5.3. Контентті жеткізу желілері

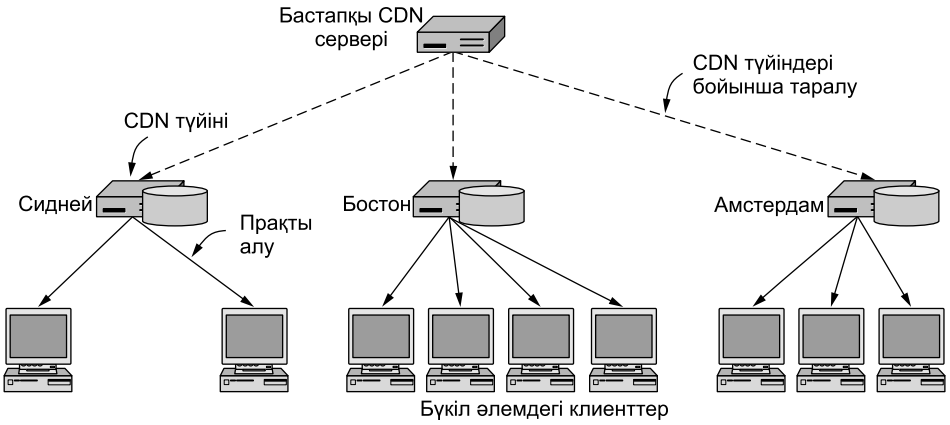
Серверлік фермалар және веб-проксилер үлкен сайттар құрастыруға және желінің жұмысын жақсартуға көмектеседі, бірақ олар контентті ауқымды масштабта жеткізуге міндетті веб-сайттар үшін жеткілікті түрде пәрменді емес. Бұл сайттар үшін басқа амал қажет.

CDNs (Content Delivery Networks – контентті жеткізу желілері) дәстүрлі веб-кэштеу идеясының аяғын басына келтіреді. Клиенттер парақ көшірмесін жақын арадағы кэштен іздегеннің орнына, парақ иелерінің өздері оларды әржердегі түйіндер жиынтығына орналастырып, клиенттер жақын жердегі түйінді сервер ретінде пайдалану үшін оны өздері бағыттайтын болды.

Деректерді CDN таратқан жағдайдағы олардың жүру жолдарының мысалы *7.40-суретте* бейнеленген. Бұл – ағаш. CDN-нің бастапқы сервері контент көшірмесін CDN-дегі басқа түйіндерге таратады. Біздің мысалымызда, Сидней, Бостон және Амстердамда орналасқан түйіндерге. Бұл үдеріс үзік сызықтармен көрсетілген. Сонан кейін клиенттер парақтарды жақын арадағы CDN түйіндерінен алады. Бұл үдеріс біркелкі сызықпен көрсетілген. Осылайша, Сиднейде орналасқан екі клиенттің екеуі де парақтың көшірмесін Сиднейден алады, олар парақты, мүмкін Еуропада орналасқан бастапқы серверден алмайды.

Ағаш тәріздес құрылымды пайдаланудың үш артықшылығы бар. Біріншіден, контентті тарату, үлкен көлемдегі CDN түйіндерін пайдаланып (және тарату CDN түйіндері арасындағы жіңішке орын болған кезде, ағаштағы деңгейлер саны үлкейеді), кез келген қажет клиенттер санына дейін масштабталуы мүмкін. Ағаш тәріздес құрылым клиенттер санына тәуелсіз тиімді. Бастапқы сервер асыра жүктелмеген, себебі ол клиенттермен CDN ағашының түйіндері арқылы әрекеттеседі, ол парақтың әр сұранысына өзі жауап бермейді. Екіншіден, әр клиент парақты, алыстағы серверден емес, жақын арадағы серверден жеткізу арқасында жақсы өнімділікке ие болады. Бұл байланыс орнату уақытының қысқа және осының арқасында TCP баяу старты жылдам жүретіндігіне байланысты, ал ең қысқа желілік жолдың Интернеттегі кептеліс аумағы арқылы өту ықтыималдығы төмендейді. Соңында, желінің жалпы жүктелуі

де азайтылады. Егер CDN түйіндері жақсы орналасқан болса, онда әр парақ желінің әр бөлігіне тек бір рет беріледі. Бұл өте маңызды, себебі кім де кім ақырында өткізгіштік қабілеттілік үшін ақы төлейді.



7.40-сурет. CDN тарату ағашы

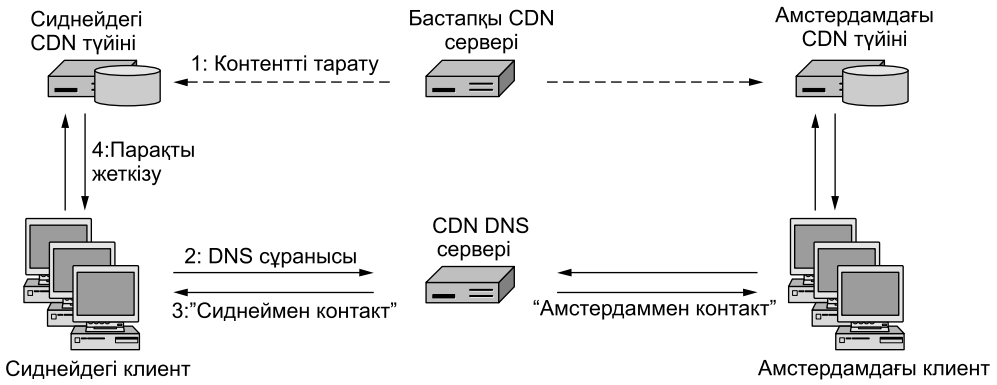
Тарату ағашын пайдалану идеясы қарапайым. Бұл ағашты клиенттердің пайдалануын ұйымдастыру одан да жеңіл мәселе. Мысалы, прокси-серверлер шешімді қамтамасыз етуі қажет тәрізді. *7.40-суреттен*, егер әр клиент Сиднейде, Бостонда және Амстердамда кэштелуші веб-прокси ретінде CDN-нің бір түйінін пайдалануға бапталған болса, тарату ағаш тәріздес болатынын көруге болады. Алайда, іс жүзінде бұл стратегия үш себептен жұмыс істемейді. Бірінші себеп, клиенттер желінің бұл бөлігінде әртүрлі компанияларға қарасты болуы мүмкін, сондықтан әртүрлі веб-проксиді пайдаланады. Естеріңізге сала кетейік, кэштеу әртүрлі ұйымдар үшін жалпылама болмайды және тұтынушылар саны көп болғанда кэштеу пайдасы шектеулі, сонымен бірге қауіпсіздік тұрғысынан да. Екіншіден, CDN көп болғанымен бір клиент тек бір прокси кэшті пайдалана алады. Қандай CDN-ді ол прокси ретінде пайдалануы керек? Соңында, іс жүзіндегі маңызды мәселе, проксиді клиент өзі баптайды. Олар CDN контентін тарату артықшылықтарына бапталуы, бапталмауы да мүмкін және CDN бұған ештеңе істей алмайды.

Бір деңгейлі дистрибутивті ағашты қолдаудың тағы бір жолы – **айнаны (mirrornig)** пайдалану. Бұл әдісте де алдыңғыдағыдай бастапқы сервер контентті CDN түйіндерінде қайталайды. CDN желісінің әртүрлі жеріндегі түйіндер айна деп аталады. Бастапқы сервердегі веб-парақтардың әртүрлі айналарға нақты сілтемесі бар, олар тұтынушыға парақтың қай жерде орналасқаны жайлы мәлімет береді. Мұндай схема тұтынушыға жақын арадағы айна адресін қолмен теріп, контенті жүктеу үшін пайдалануға мүмкіндік береді. Айнаны пайдаланудың әдеттегі мысалы – үлкен программа дестесін, мысалы, АҚШ-тың батыс және шығыс жағалауларындағы, Азия және Еуропадағы

серверлерде орналастыру. Айна сайттары әдетте, өзгермейді және бұл сайттар таңдауы айлар, тіпті, жылдар бойы өзгеріссіз қалады. Бұл – сыналған және тексерілген техника. Алайда, тарату бұл жағдайда тұтынушыға байланысты, айна – бұл әртүрлі веб-сайттар, тіпті, олар бір-бірімен байланысқан болса да.

Алдыңғы екі амал қиындықтарын жеңіп шыққан үшінші амал DNS-ті пайдаланады және **DNS қайта тағайындауы (DNS redirection)** деп аталады. Айталық, клиент URL адресі *http://www.cdn.com/page.html* парағын алғысы келеді. Бұл парақты алу үшін браузер *www.cdn.com* IP-адресі бойынша анықтау үшін DNS-ті пайдаланады. Бұл жағдайда DNS-ті қарау әдеттегідей жүреді. DNS хаттамасы қолданылады, браузер *cdn.com* үшін аттар серверінің IP-адресін таниды. Ал енді нағыз ақылды ой іске қосылады. Аттар сервері CDN-мен басқарылады. Әрбір сұраныс үшін бір IP-адресті қайтару орнына ол сұраныс жөнелткен клиенттердің IP-адрестерін қарап, әртүрлі жауаптар қайтарады. Айталық, егер Сиднейдегі клиент CDN аттар серверінен *www.cdn.com* IP-адресін жөнелтуді сұраса, аттар сервері Сиднейдегі CDN түйінінің адресін қайтарады, ал егер тура осындай сұранысты Амстердамдағы клиент сұраса, аттар сервері Амстердамдағы түйінінің IP-адресін қайтарады.

Бұл стратегия DNS семантикасына заңды түрде сәйкес келеді. Біз аттар серверінің өзгермелі IP-адрестер тізімін қайтара алатынын алдын ала көреміз. Сараптаудан кейін Сиднейдегі клиент парақты Сиднейдегі CDN түйінінен алады. Келешекте осы «сервердегі» қалған парақтарда DNS кэштеу көмегімен, Сиднейдегі түйіннен алынатын болады. Қадамдардың толық тізбегі *7.41-суретте* көрсетілген.



7.41-сурет. DNS-ті пайдаланып, клиенттерді жақын арадағы CDN түйініне бағыттау

Жоғарыда сипатталған мәселедегі күрделі сұрақ – ең жақын CDN түйінін табу және оны қалайда орындау. «Ең жақын арадағы» түсінігін анықтауда негізгі фактор географиялық емес. Клиент және CDN түйінін сәйкестендіруді анықтауда кем дегенде, екі факторды назарға алу керек. Бірінші фактор – желілік арақашықтық. Клиент CDN түйіне дейін қысқа және сыйымдылығы

үлкен желілік жолға ие болуы керек. Бұл жүктеуді жылдамдатады. IP-адрес бойынша клиент және оның мекен жайын анықтау үшін CDN алдын ала анықталған бейнелеуді пайдаланады. Таңдап алынған түйін түзу бойынша ең қысқа арақашықтықта орналасуы да, олай болмауы да мүмкін. Желілік жол ұзындығының кейбір комбинациясы және ондағы шектелген сыйымдылық маңызы рөл атқарады. Екінші фактор – осы кездегі CDN түйініндегі жүктеме. Егер CDN түйіні асыра жүктелген болса, ол асыра жүктелген веб-сервер тәрізді жауапты баяу жөнелтеді, бұл біз ең алдымен құтылмақ болған мәселе. Сондықтан кейбір клиенттерді біршама алыс орналасқан, бірақ аз жүктелген түйіндерге жіберіп, CDN түйіндерінің жүктелуін теңдестіруге тура келеді.

Контентті тарату үшін DNS-ті пайдалануды 1998 жылы, Дүниежүзілік өрмек өзінің ерте даму салмағы астында тұншығып жатқан кезде, Akamai алғаш рет құрастырды. Akamai алғашқы ең үлкен CDN және осы индустрияның көш басшысы болды. DNS-ті клиенттерді жақын арадағы түйіндемен байланыстыру үшін пайдаланудан да ақылдысы бизнесті ынталандыру болды. Компаниялар Akamai-ға өз контенттерін тұтынушыға жеткізгендігі үшін төлем ақы төледі, сондықтан олар клиенттерге ыңғайлы, жақсы жауап қайтаратын веб-сайттарға ие болды. CDN түйіндері желі байланысы жақсы жерлерде орналасуы керек, ол бастапқыдан-ақ интернет-провайдерлердің ішкі желісі дегенді білдіреді. Интернет-провайдерлер үшін өз желілерінде CDN түйіні орналасуының артықшылығы, CDN түйіні оларға қажет (және олар төлем ақы төлейтін) жоғары орналасқан желілік өткізгіштік қабілеттіліктің санын тек прокси-сервермен шектейді. Бұдан басқа, CDN түйін интернет-провайдердің клиенттерге жауап қайтаруын жақсартады, бұл провайдерлерді клиенттер алдында жақсы етіп көрсетеді, басқа да CDN түйіні бар бәсекелес провайдерлер алдында артықшылық береді. Интернет-провайдер үшін ешбір ауыртпалығы жоқ бұл артықшылықтар CDN түйінін орнатуды тартымды етеді. Осылайша, контент жеткізушінің де, интернет-провайдердің де, клиенттің де ұпайы түгел, ал CDN пайда жасайды. 1998 жылдан бастап бұл бизнеске басқа компаниялар да кірді, сондықтан қазір бұл көптеген жеткізушілермен бәсекелес сала.

Бұл сипаттау көптеген компаниялардың өз CDN-ін түрғызғысы келмейтінін білдіреді. Оның орнына олар өз контентін жеткізу үшін CDN жеткізушінің, мысалы, Akamai қызметін пайдаланады. Басқа компанияларға CDN қызметін пайдалануға рұқсат беру үшін, біз өз сипаттауымызға тағы бір соңғы қадамды қосуымыз керек.

Келісім шартқа қол қойылғаннан кейін веб-сайт иесі өз контентінің дистрибутивін CDN-ге береді. Бұл контент CDN түйіндеріне жөнелтіледі. Бұдан басқа, веб-сайт иесі өз веб-парақтарындағы контентке апаратын барлық сілтемелерді қайта жазады. Контентті веб-сайтпен байланыстыру орнына, парақтар контентпен CDN арқылы байланысады. Осы схема жұмысының мысалы ретінде, *7.11. а-листингінде* көрсетілген Ұлпа бейне веб-парағының бастапқы кодын қарастырайық. Ол алдын ала өңделгеннен кейін *7.11. ә-листингіне* түрлендіріліп, Ұлпа бейне серверде *www.fluffyvideo.com/index.html* парағы ретінде орналастырылған.

Тұтынушы өз браузерінде *www.fluffyvideo.com/index.html* URL терген кезде, негізгі парақты (HTML) әдеттегі тәсілмен алуға мүмкіндік беріп, DNS Ұлпа бейне сайтының IP-адресін қайтарады. Тұтынушы гиперсілтемелердің кез келгеніне шерткен кезде, браузер DNS-тің *www.cdn.com* табуын сұрайды. Іздеу барысында DNS-сервер CDN-німен әрекеттесу орын алады, ол жақын арадағы CDN түйінінің IP-адресін қайтарады. Сонан кейін, браузер CDN түйініне әдеттегі HTTP-сұранысты жөнелтеді, мысалы, */fluffyvideo/koalas.mpg* суретін алуға. URL парақты тұтынушыға қайтару үшін жолды *fluffyvideo* бастап сәйкестендіреді, осылайша CDN түйіні өзі қызмет көрсететін әртүрлі компаниялардың сұраныстарын бөле алады. Соңында, бейне алынды, тұтынушы әдемі ұлпа жануарларды тамашалайды.

7.11-листингі. Бастапқы парақтың HTML-мәтіні (а);
сол парақ, тек CDN-ге ауысқаннан кейін (ә)

(а)

```
<html>
<head> <title> Ұлпа бейне </title> </head>
<body>
<h1> Ұлпа бейне материалдарының тізімі </h1>
<p> Төмендегі ақысыз мысалдарға шертіңіз. </p>
<a href=»koalas.mpg»> Бүгінгі коалалар </a> <br>
<a href=»kangaroos.mpg»> Қызықты кенгурулар </a> <br>
<a href=»wombats.mpg»> Әдемі вомбаттар </a> <br>
</body>
</html>
```

(ә)

```
<html>
<head> <title> Ұлпа бейне </title> </head>
<body>
<h1> Ұлпа бейне материалдарының тізімі </h1>
<p> Төмендегі ақысыз мысалдарға шертіңіз. </p>
<a href=»http://www.cdn.com/fluffyvideo/koalas.mpg”> Бүгінгі коалалар </a> <br>
<a href=»http://www.cdn.com/fluffyvideo/kangaroos.mpg”> Қызықты кенгурулар </
a> <br>
<a href=»http://www.cdn.com/fluffyvideo/wombats.mpg”> Әдемі вомбаттар </a>
<br>
</body>
</html>
```

CDN-де сақталатын контентті бөлгеннен кейінгі парақ иесінің стратегиясы - контент иесіне бақылауды беру, ал деректердің үлкен бөлігін CDN-ге ауыстыру. Көптеген «кіріс» парақтарының көлемі кішкене ғана, себебі ол тек HTML-мәтіннен тұрады. Бұл парақтар жиі үлкен файлдармен, мысалы, бейнемен не-

месе суретпен байланысады. Ал үлкен файлдар CDN-мен қолданады, тіпті, CDN тұтынушылар үшін толығымен ашық болса да. Сайт дәл бастапқыдай болып көрінеді, тек ол жылдамырақ жұмыс істейді.

Сайтардың ортақ CDN-ді пайдалануының тағы да басқа артықшылықттары бар. Веб-сайтқа деген болашақ сұранысты болжау қиын. Жиі **«flash crowds» (веб-сайтқа жүгінудің күрт өсуі)** деген атпен танымал, үлкен сұраныс толқындары болып тұрады. Мұндай үлкен толқын, мысалы, тауардың жаңа версиясы шыққанда, мода көрсетілімдері өткенде немесе басқа да оқиғалар немесе компания жаңалықта көрінген кезде орын алуы мүмкін. Тіпті, бұрын танымал емес, адам бармайтын саз-балшық веб-сайт, егер жаңалықтарда айтылып, оған танымал сайттарда сілтеме қойылса, кенеттен Интернет орталығы болуы мүмкін. Көптеген сайттар трафиктің тым үлкеюін көтере алмайтындықтан, осындай толқыннан кейін істен шығып жатады.

Бір жағдайды қарастырайық. Әдетте, Florida Secretary of State сайты аса жүктелмеген, бірақ сіз ол жерден Флорида штатының корпарациялары, нотариусы және мәденит оқиғалары, сонымен бірге сайлау кезіндегі дауыс берулер жайлы ақпаратты ала аласыз. Кейбір себептермен, 2000 жылдың 7 қарашасында (Америка президентінің сайлау күні, Буш Горға қарсы) көптеген адамдарды кенеттен осы сайтта жарияланған сайлау нәтижесі қызықтыра бастады. Сайт кенеттен әлемдегі ең көп сұранысқа ие веб-сайттардың бірі болып шыға келді, нәтижесінде ол істен шықты. Егер ол жерде CDN қолданылса, ол мүмкін істен шықпас ба еді.

CDN-ді қолданып, сайт контентті орналастырудың үлкен сыйымдылығы-на қолжеткізеді. Ең үлкен CDN-нің әлем бойынша әртүрлі елдерде орналасқан, ондаған мың серверлері бар. Уақыттың әр сәтінде тек кішігірім сайттар ғана жүгінулер толқынын иеленетіндіктен (анықтама бойынша), бұл сайттар толқын қайтқанша жүктемені басқару үшін CDN сыйымдылығын пайдалана алады. Осылайша, CDN сайтқа ұсынылатын сыйымдылықты жылдам үлкейтеді.

Жоғарыда айтылғандар Akamai жұмысының қарапайымдатылған сипаттамасы. Іс жүзінде көптеген егжей-тегжейлердің маңызы бар. Біздің мысалымызда көрсетілген CDN түйіндері әдетте, машина кластерлері болып келеді. DNS қайта адрестеуі екі сатыда жүргізіледі: бірі клиентті жақын арадағы желі бөлігіне бағыттайды, екіншісі жүктемені желінің осы бөлігіндегі түйіндер арасында таратады. Сенімділіктің де, жұмыс жылдамдығының да маңызы бар. Клиентті бір машинадан келесі машина кластеріне ауыстыру мүмкіндігі болу үшін, екінші сатыдағы DNS жауаптары қысқа TTL түрінде ұсынылады, сондықтан клиент белгілі бір адрестерді қысқа уақыт мерзімінде қайталап отырады. Соңында, осы уақытқа дейін біз сурет және бейне тәрізді, статикалық объектілерді таратуды қарастырдық, бірақ CDN сонымен бірге динамикалық түрде құрастырылатын парақтарды, ағындық медианы және тағы басқа көптеген контенттерді де қолдай алады. CDN жайлы толығырақ Dillay және басқалар (2002) басылымнан оқуға болады.

7.5.4. Біррангілі түйіндер желісі (пирингілік желілер)

Өз контентін тарату үшін бүкіл әлемде орнарласқан, 1000-даған түйіндерден тұратын CDN-ді әркім орната алмайды. (Іс жүзінде әлем бойынша 1000 виртуал машинаны жалға алу қиын емес, себебі хостинг индустриясы жақсы дамыған және бәсекеге дайын. Алайда, CDN тек түйінді алудан басталады). Қуанышқа орай, басқалардың барлығы үшін қолдануға ыңғайлы және үлкен көлемдегі контентті тарата алатын, балама бар. Бұл P2P (біррангілі желі) желісі.

P2P желілерінің таралуы 1999 жылы басталды. Қосымшаның алғашқы кеңінен таралуы, көпшілік заңды бұзу үшін пайдаланылды: Napster желісінің 50 млн тұтынушысы, үлкен пікірталас тудырып, сот шешімімен Napster желісі жабылғанша, иелерінің рұқсатынсыз авторлық құқығы қорғалған өлеңдермен алмасып отырды. Алайда, теңрангілі түйіндер технологиясының заңды мақсатта да көптеген жақсы жерлері бар. Басқа жүйелер де тұтынушылар қызығушылығын тудырып, дами бастады, ал P2P-трафик веб-трафиктен жылдам өсіп кетті. Қазіргі күнде BitTorrent – P2P-дің ең танымал хаттамасы. Ол бейнені (лицензияланған және қоғамдық), сонымен бірге басқа да трафикті тарату үшін кеңінен қолданылады және бүкіл интернет-трафиктің үлкен бөлігін құрайды. Біз оны осы бөлімде қарастырамыз.

P2P (Peer-to-Peer – біррангілі желілер немесе **пирингілік желілер)** желісіндегі файлдарға ортақ қолжеткізу идеясы – контентті тарату жүйесін құрастыру үшін, көптеген компьютерлер өз ресурстарын біріктіреді. Мұндағы компьютерлер көбіне, қарапайым үйдегі компьютерлер. Оларға Интернеттегі деректерді өңдеу орталығындағы машина болу міндетті емес. P2P-дегі компьютерлер пирлер (peers, біррангілі желі мүшелері-түйіндер) деп аталады, себебі олардың әрқайсысы, басқасына қатысты оның контентін алып, клиент ретінде және басқа түйіндерге контент ұсынып сервер ретінде әрекет ете алады. Біррангілі түйіндер жүйесі, CDN-ге қарағанда арнайы инфрокұрылымның жоқтығымен қызықты. Контентті тарату мәселесіне әрқайсысы атсалысады, қандай да бір орталықтан жиі бақылау жоқ кезде.

Көптеген адамдар P2P желісімен шабытталған, себебі бұл оларға кішкене адамға көмек көрсету тәрізді болып көрінеді. Себеп, CDN жұмысына үлкен компания қажет, ал P2P желісіне кез келген компьютері бар адам қосыла алатындығында емес. Контент тарату үшін P2P желілерінің өте үлкен сыйымдылығы бар, ол веб-сайттардың ең үлкеніне сәйкес келе алады.

Орта шамадағы, әрқайсысының өткізгіштік қабілеттілігі 1 Мбит/с желіге кеңжолақтық қосылуы бар, N тұтынушыдан тұратын P2P желісін қарастырайық. P2P желісі контентінің жалпы сыйымдылығы немесе тұтынушылар Интернетке трафик жөнелте алатын жылдамдығы N Мбит/с құрайды. Жүктеме сыйымдылығы немесе тұтынушылар трафикті ала алатын жылдамдық та N Мбит/с құрайды. Әр тұтынушы бір мезгілде, әр бағытта 1 Мбит/с жылдамдықпен трафикті жүктеп және қабылдай алады.

Бұның айқын болуы міндетті емес, бірақ контентті тарату үшін бүкіл сиымдылықты тиімді падаланыла алады, тіпті, бір файл көшірмесін барлық тұтынушылар ортақ қолданған жағдайда да. Бұны түсіну үшін, тұтынушылар бинарлық ағашқа ұйымдастырылған деп елестетіңіз. Әрбір жапырақ емес тұтынушы файлды басқа екеуіне жөнелтеді. Файлдың бір көшірмесі ағаш бойымен барлық тұтынушыларға жеткізілуі тиіс. Жүктеменің өткізгіштік қабілеттілігін мүмкіндігінше осынша көп тұтынушыларға қолдану үшін (демек, төмен күту уақытымен көптеген файл тарату үшін) бізге тұтынушылардың желілік қызметін конвейрлеу керек. Әр тұтынушы жаңа фрагментті ағаштың жоғары бөлігінен алып, бұрын алынған фрагментті ағашпен төмен жөнелте алады. Осылайша, фрагменттердің кішігірім бөлігі жөнелтіліп (ағаш тереңдігіне тең), конвейр іске қосыла салысымен барлық жапырақ тұтынушылар бір-біріне файл жөнелтумен болады. Жапырақ емес тұтынушылар саны *N/2* болғандықтан, бұл ағаштың өткізгіштік қабілеттілік жүктемесі - *N/2* Мбит/с. Біз осы амалды пайдаланып, жапырақ және жапырақ емес төбелер рөлін ауыстырып, басқа *N/2* Мбит/с өткізгіштік қабілеттілікті пайдаланатын ағаш құрастыра аламыз. Осы құрылым толығымен бүкіл сиымдылықты пайдаланады.

Бұл P2P желілерінің өзін-өзі масштабтайтындығын білдіреді. Олардың мүмкін деген жүктеме көлемі, тұтынушылар тарапынан жүктеме қажеттілігімен бірге өседі. Олар әрқашанда, қандайда бір арнайы инфроқұрылым қажетінсіз, «жеткілікті түрде үлкен». Кері жағдай: тіпті, үлкен веб-түйіннің бекітілген көлемі бар және ол не тым үлкен немесе тым кіші болады. Әрқайсысының 10 Гбит/с өткізгіштік қабілеттілігі бар, 100 кластерден тұратын сайтты қарастырайық. Бұл үлкен көлем тіпті, тұтынушылар саны аз болғанда да көмектеспейді. Сайт ақпаратты *N* тұтынушыға *N* Мбит/с-тен жылдам жеткізе алмайды, себебі шектеу веб-сайтта емес, тұтынушы жақта. Ал әрқайсысы 1 Мбит/с-тен миллионнан аса тұтынушы болған кезде, веб-сайт барлық тұтынушыларды жүктемемен қамтамасыз ететіндей, деректерді қажетінше жылдам жөнелте алмайды. Тұтынушылардың бұл саны тым көп болып көрінеді, алайда, үлкен BitTorrent желілер (мысалы, Pirate Bay) 10 млн артық тұтынушыларының бар екенін хабарлайды. Бұл біздің мысал терминінде 10 Тбит/с-тен аса!

Сіз бұл ұқыпсыз санауларға біршама (немесе, ең жақсысы едәуір) скептицизммен қарағаныңыз жөн, себебі олар жағдайды қарапайым етіп көрсетеді. P2P желілеріне татымды шақырулар – тұтынушылар әртүрлі формада және көлемде, сонымен бірге олардың жүктелуі және сыйымдылығы әртүрлі бола тұрсада, өткізгіштік қабілеттілікті жақсы пайдалану. Алайда, бұл цифрлар P2P-дің үлкен шамасын көрсетеді.

P2P-ді маңызды ететін тағы бір себеп бар. CDN және басқа да орталықтандырылған қызметтер провайдерлерді тұтынушылардың үлкен саны жайлы – веб-серфинг ұнатуынан бастап, интернет-дүкендердегі сатып алулары, мекен жайы және электронды поштасы тәрізді жеке ақпаратты табуға мәжбүрлейді. Бұл ақпараттар, жақсы және жекелендірілген қызмет көрсету немесе адам-

дардың жеке өміріне рұқсатсыз басып кіру үшін қажет. Екінші нұсқа жаңа өнімнің бір бөлігі ретінде әдейі жасалуы немесе кездейсоқ жария ету немесе рұқсат етілмеген жария ету салдарынан болуы мүмкін. P2P жүйесінде барлығын үстінен бақылап отыратын, бір жеткізуші болуы мүмкін емес. Бұл P2P жүйесінің міндетті түрде құпиялықты қамтамасыз етеді дегенді білдірмейді, себебі тұтынушылар белгілі бір дәрежеде бір-біріне сенеді. Бұл – тек олар бір орталықтан басқарылатын жүйеден ерекше құпиялық формасын қамтамасыз ете алады дегені. P2P жүйелері қазір файлдарды бірлесіп пайдалану (мысалы, сақтау, тасымалдау) үшін зерттеліп жатыр және бұл артықшылықтың маңызды екенін уақыт көрсетеді.

P2P технологиясы екі байланысқан жолмен дамып келеді. Практикалық тұрғыдан – күнделікті қолданылатын жүйелер бар. Бұл жүйелердің ең жақсы танымалдары BitTorrent хаттамасына негізделген. Академиялық тұрғыдан – P2P жүйелеріне орталықтандырылған компоненттерге тәуелсіз, толық жүйе ретінде жақсы жұмыс істеуге мүмкіндік беретін **DHT (Distributed Hash Table – хэш-кестені тарату)** алгоритмдеріне үлкен қызығушылық байқалады. Біз бұл екі технологияны да қарастырамыз.

BitTorrent

BitTorrent хаттамасын 2001 жылы Коэн Брахом түйіндер жиынтығына файлдарға жылдам және жеңіл ортақ қол жеткізуді қамтамасыз ету үшін құрастырған болатын. Көптеген браузерлер веб-сервермен байланысу үшін HTTP хаттамасын қолданады және осы хаттаманы қолдайтын, еркін тарататын ондаған клиенттер де бар. Хаттама *www.bittorrent.org-ma* ашық стандарт ретінде қолжетімді.

Әдеттегі біррангілі түйіндер (пирингтер) жүйесінде, мысалы, BitTorrent көмегімен құрастырылған, тұтынушылардың әрқайсында басқа тұтынушылардың қызығушылығын тудыратын қандай да бір ақпарат бар. Бұл ақпарат еркін программалық жабдықтама, әуен, бейне, фотосуреттер және т.с.с. болуы мүмкін. Контентке ортақ қолжеткізуді қамтамасыз ету үшін үш мәселені шешу керек.

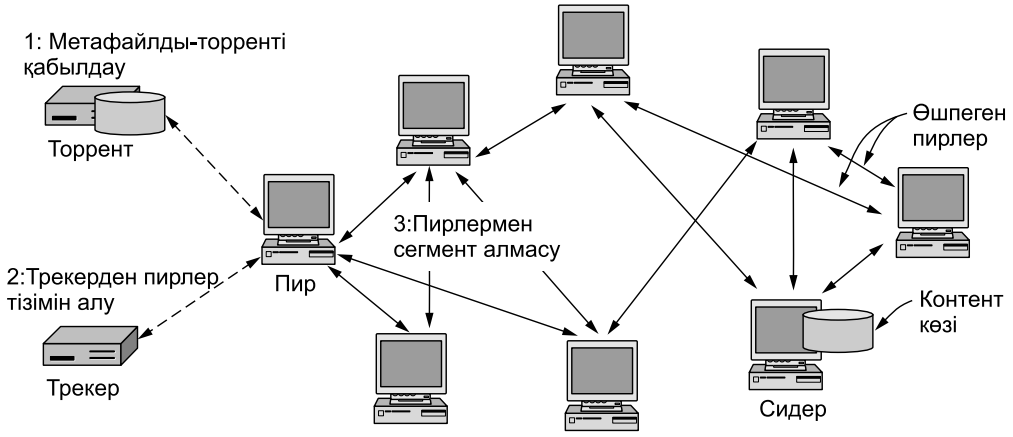
1. Пир өзін қызықтыратын контенті бар және оны жүктегісі келетін басқа пирлерді қалай табады?
2. Әркімге жылдам жүктеуді қамтамасыз ету үшін, пирлер контент қалай көшіреді?
3. Контенті басқалар үшін жүктегенде, сонымен бірге өзі үшін жазып алған кезде пирлер бір-бірін қалай ынталандырады?

Пирлердің барлығында бірдей бүкіл контент бола бермейді, кем дегенде бастапқыда, сондықтан бірінші мәселе орын алады. BitTorrent қабылдаған әдіс – әрбір контент жеткізушісі үшін, **торрент (torrent)** деп аталатын контент сипаттамасын жасау. Торрент контенттен әлдеқайда кіші және пир оны

басқа пирлерден жүктейтін деректер тұтастығын тексеру үшін пайдаланады. Контентті жүктегісі келген басқа тұтынушылар алдымен торренті алулары керек, айталық, оны контентті жарнамалайтын веб-парақтан табуы керек.

Торрент – бұл екі түрлі ақпараттан тұратын, жай, белгілі бір форматтағы файл. Бірінші түрі – пирды торрент мазмұнына әкелетін сервер **трекер** деп аталады. Екінші ақпарат түрі – контенттің бөліктері болып саналатын, бір көлемдегі фрагменттер тізімі немесе **сегменттер (chunks)**. Әртүрлі торренттер үшін түрлі сегменттер көлемі қолданылуы мүмкін, әдетте, 64-тен 512 Кбайтқа дейін. Торрент файлында әр сегменттің 160-биттік SHA-1 хэш сегменті ретінде аты көрсетіледі. Біз SHA-1 тәрізді криптографиялық хэштерді *8-тарауда* қарастырамыз. Әзірше, хэш – әлдеқайда ұзын және қауіпсіз бақылау қосындысы деп санаңыз. Торрент файлдың сегменттерінің және хэш мазмұнының көлемі контенттен кемдегенде үш ретке төмен, сондықтан ол жылдам жөнелтіледі.

Торрентте сипатталған контентті жүктеу үшін пир алдымен торрент трекерімен әрекеттеседі. **Трекер (tracker)** – бұл контентті активті жүктейтін және тасымалдайтын, қалған барлық пирлер тізімін қолдайтын сервер. Пирлердің бұл жиынтығын **үйір (swarm)** деп атайды. Үйір мүшелері, өздерінің әлі де активті екенін, сонымен бірге үйірден кететіндігін хабарлап, үнемі трекермен әрекеттесіп отырады. Жаңа пир үйірге қосылу үшін трекермен әрекеттескен кезде, трекер оған үйірдің басқа пирлері жайлы хабарлайды. Торрентті алу және трекермен әрекеттесу – бұл *7.42-суретте* көрсетілген контентті жүктеудің алғашқы екі қадамы.



7.42-сурет. BitTorrent

Екінші мәселе – контентті жылдам жүктелуді қамтамасыз ететіндей етіп бөлу. Бастапқы үйір құралған кезде, кейбір пирлерде контентті құрайтын барлық сегменттер болуы керек. Бұл пирлерді **себушілер (seeders)** деп атайды. Үйірге қосылатын басқа пирлерде ешқандай сегменттер болмайды, олар контентті жазып алатын пирлер.

Пир үйірде болған кезде ол өзінде жоқ сегменттерді басқа пирлерден жазып алады және өзінде барды басқа қажет еткен пирлерге жүктейді. Бұл алмасу *7.42-суретте* контентті таратудың соңғы қадамы ретінде көрсетілген. Біраз уақыт өткеннен кейін, пир сегменттердің үлкен санын жинайды, контент толығымен жүктелгенше. Пир үйірді кез келген уақытта тастап кетуі (қайта қосылуы) мүмкін. Әдетте, пир үйірде қысқа уақыт мерзімінде қалады, өз жеке жүктелімі біткенше. Жаңадан пайда болып, жоғалып жатқан пирлер салдарынан үйірдегі «аққыштық» үлкен болуы мүмкін.

Жоғарыда сипатталған тәсіл жақсы жұмыс істеу үшін, әр сегмент көптеген пирлерде қолжетімді болуы керек. Егер әрқайсысы сегментті бір ретпен алу керек болса, онда көптеген пирлер келесі сегмент себушілеріне тәуелді болар еді. Бұл жіңішке орын тудырар еді. Оның орнына пирлер бір-бірімен өздерінде бар сегменттер тізімімен алмасады. Сонан кейін жүктеу қиын, сирек сегменттерді таңдайды. Мұндағы ой, сирек сегментті жүктеген кезде оның көшірмесі жасалады, бұл сегментті басқа пирлер үшін қолжетімді етеді. Егер бұны барлық пирлер орындаса, қысқа уақыттан кейін барлық сегменттер қолжетімді болады.

Үшінші мәселе ең қызықтысы болуы мүмкін. CDN түйіндері контентті тек тұтынушыларға жеткізуге бапталған. Ал P2P түйіндері олай емес. Бұл тұтынушылар компьютері, ал тұтынушыларға басқа тұтынушыға жүктеуге көмектесудің орнына, кинофильмді жазып алу әлдеқайда маңызды. Жүйеден ресурсты ешбір үлес қоспай алушы түйіндер **фрирайдерлер (freeriders)** немесе **личерлер (leechers – сүлік)** деп аталады. Егер олар тым көп болса, жүйе жақсы жұмыс істемейді. Ертедегі P2P жүйелері олармен жұмыс істеген (Saroiu және басқалар, 2003) BitTorrent олардың санын азайтуға тырысуда.

BitTorrent клиенттерінде қабылданған әдіс – жүктеу кезінде жақсы тәртіп көрсеткен пирлерді марапаттау. Әр пир басқа пирлерге ретсіз жүгінеді, олардан сегмент алады және оларға сегмент жөнелтеді. Пир жүктеудің жоғары өнімділігін қамтамасыз ету үшін, тек аздаған пирлермен сегмент алмасуын жалғастырады және жақсы әріптес табу үшін, басқа пирлерді де ретсіз байқап көреді. Кездейсоқ жүгінулер жаңа қосылған пирлерге бастапқы сегментті алуға мүмкіндік береді, сонан кейін ол басқалармен осы сегментпен алмаса алады. Осы сәтте түйін сегмент алмасып жатқан пирлер **сөнбегендер (unchoked)** деп аталады.

Біраз уақыттан кейін бұл алгоритм жүктеу және жазып алу мүмкіндіктері сәйкес келетін пирлерді сәйкестендіруі керек. Пир басқа пирлерге неғұлым көбірек көмектессе, ол жауапты соғұрлым ұзақ күтуі мүмкін. Пирлер жиынтығын пайдалану да жоғары өнімділік үшін пирлер өткізгіштік қабілеттілігін қанықтыруға көмектеседі. Екінші жағынан, егер пир басқа пирлерге сегменттер жөнелтпесе немесе өте баяу жөнелтсе, ол ерте ме кеш пе кесіліп тасталған немесе **сөніп қалған (choked)** болуы мүмкін.

Сөндіруші алгоритмі кейде, коорпорацияны қайталап әрекеттесуі үшін ынталандыратын, **«тіске-тіс» (tit-for-tat)** стратегиясын іске асыру ретінде си-

патталады. Алайда, ол клиенттерді жүйені алдаудан қорғамайды (Piatek және басқалар, 2007). Дегенмен, кездейсоқ тұтынушылар үшін фрирайдты қиындату механизмі мен мәселеге назар аудару BitTorrent танымалдығына көмектескен болуы мүмкін.

Біздің талқылауымыздан байқағандай, BitTorrent өзінің ауқымды терминдері бар. Торренттер, үйір, личеры, сидеры, трекеры, сонымен бірге сөгістер, сөнгендер, тыңдау және т.с.с. Қосымша ақпаратты BitTorrent жайлы қысқа мақаладан (Коэн және басқалар, 2003) және Желіден, *www.bittorrent.org* сайтынан оқи аласыздар.

DHT – хэш-кестені тарату

Файл алмасу P2P желілерінің пайда болуы 2000 жылдары зерттеушілер бірлестігінде үлкен қызығушылық тудырды. P2P жүйелерінің негізгі мақсаты – CDN және басқа да жүйелердегідей бір орталықтан басқарылуды болдырмау. Бұл маңызды артықшылық болуы мүмкін. Жүйе өте үлкен көлемге дейін өскен кезде, бірорталықтан басқарылатын компоненттер істен шығудың жіңішке орны бола бастайды. Сонымен бірге, орталық компоненттерді бақылау нүктесі ретінде пайдалануға болады (мысалы, P2P желісін өшіру үшін). Алайда, ертедегі P2P жүйелері жартылай орталықсыздандырылған немесе егер олар толығымен орталықсыздандырылған болса, олар тиімсіз болар еді.

Біз осы қазір сипаттаған, дәстүрлі BitTorrent формасы тең рангілі тасымалдаулар және әр үйір үшін орталық трекерді пайдаланады. Тең құқықты түйіндер жүйесінде осы трекерді орталықсыздандыру қиынға соғады. Басты мәселе - қандай пирлерде ізделініп отырған контенттің бар екенін қалай анықтауға болады. Мысалы, әр тұтынушыда басқа тұтынушылар алғысы келетін деректердің бір немесе бірнеше элементі болуы мүмкін: ән, сурет, программа, файл және т.с.с. Басқа тұтынушылар оларды қалай табады? Кімде не бар екені жайлы тізімін құрастыру – қарапайым, бірақ орталықтандырылған. Әр пирдің өз тізімі болғаны көмектеспейді. Бірақ олар таралады. Алайда, пирлер үшін өзекті контент тізімін қолдау (себебі контент жүйе бойымен қозғалады) жұмысы тым үлкен болғандықтан, бұл талпыныс үмітсіз.

Зерттеушілер бірлестігі қолға алған мәселе – P2P үшін толығымен таратылған және жақсы сипаттамалары бар индекстер құрастыру мүмкін бе? Біз бұнда үш затты меңзеп отырмыз. Біріншіден, әр түйін басқа түйіндер жайлы тек азғана ақпаратты сақтайды. Бұл қасиет индекс өзектілігін сақтау үлкен шығынды қажет етпейді дегенді білдіреді. Екіншіден, әр түйін индекс бойынша жазбаны жылдам таба алады. Кері жағдайда, бұл тиімсіз индекс. Үшіншіден, әр түйін индексті басқа түйіндер пайда болып және жоғалып жатқан кезде де пайдалана алады. Бұл қасиет индекс өнімділігі түйіндер санымен бірге өседі дегенді білдіреді.

Бұл сұрақтың жауабы «Иә». 2001 жылы төрт түрлі шешім алынды. Бұл – Chord (Stoica және басқалар, 2001), CAN (Ratnasamy және басқалар, 2001), Pas-

try (Rowstron және Drusxel, 2001) және Tapestry (Zhao және басқалар, 2004). Басқа шешімдер кейіннен ойлап табылды, соның ішінде іс жүзінде қолданылатын Kademlia (Maymounkov және Mazieres, 2002). Бұл шешімдер **DHT (Distributed Hash Tables – таратылған хэш-кестелер)** деген атпен белгілі, себебі индекстің негізгі қызметі – кілт және мән арасындағы сәйкестікті орнату. Бұл хэш-кестеге ұқсас және шешім, әрине, таратылған версиялар болып саналады.

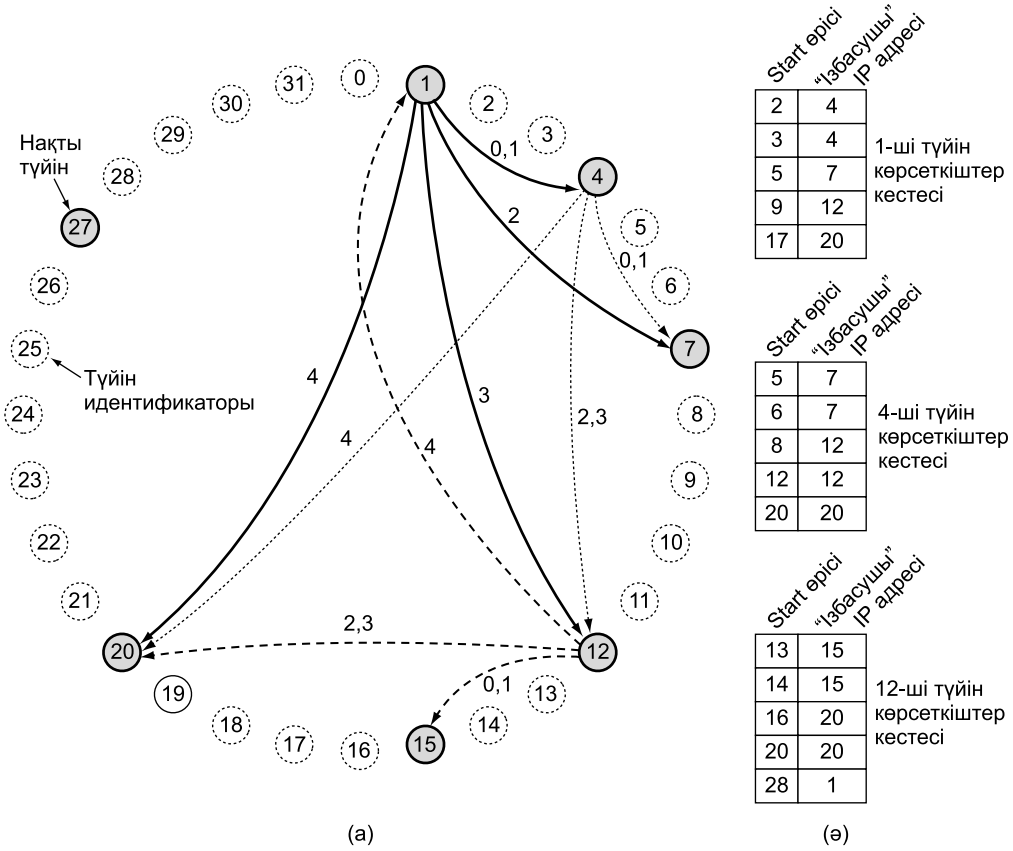
DHT өз жұмысын түйіндер арасындағы тұрақты құрылымды пайдаланып жүзеге асырады. Бұл пирлердің барлық байланыстарын пайдаланатын, дәстүрлі P2P жүйелерінен біршама ерекше. Осы себептен DHT **құрылымдалған P2P-желісі** деп атайды. Дәстүрлі P2P хаттамалары **құрылымдалмаған P2P-желісін** тұрғызады.

DHT Chord сипаттайық. Сценарий ретінде, BitTorrent-те дәстүрлі пайдаланылатын орталықтандырылған трекерді толығымен таратылған трекерге қалай алмастыру керектігін қарастырайық. Бұл мәселені шешу үшін Chord-ты пайдалануға болады. Бұл сценарийде толық индекс – компьютер контентті жүктеу үшін қосыла алатын барлық үйірлер тізімі. Индексті қарауға арналған кілт – торренттің контентке берген сипаттамасы. Ол контентті жүктеуге болатын үйірді контентті құрайтын барлық сегменттердің хэш-функциясы ретінде бірегей сәйкестендіреді. Әр кілт үшін индексте сақталған мән үйірге кіретін пирлер тізімін құрайды. Бұл пирлер – контентті жүктеу үшін әрекеттесуге болатын компьютерлер. Контентті, мысалы, кинофильмді, жүктегісі келетін адамда тек торренттің сипаттамасы бар. DHT жауап беруге тиіс сұрақ – орталықтандырылған деректер базасы жоқ жағдайда, адам фильмді жүктейтін нақты пирді (BitTorrent-тің миллион түйіндерінің ішінен) қалай табады?

Chord DHT *n* қатысушы түйіндерден тұрады. Біздің сценарийімізде бұл – BitTorrent іске қосылған түйіндер. Әр түйіннің байланысуға болатын IP-адресі бар. Толық индекс түйіндер арасында таратылған. Бұл әр түйін сегментті және басқа түйіндер пайдаланатын индекстің бір бөлігін сақтайды дегенді білдіреді. Chord басты рөлі – ол түйіннің IP-адресі немесе контент атын (мысалы, фильмнің) пайдаланып емес, виртуалды кеңістіктегі идентификаторларды пайдаланып, осы индекстерді басқарады. Идентификаторлар тұжырымдамасы – олар сақина бойымен өсу реті бойынша реттелген *m*-битті сандар болып келеді. Түйін адресін идентификаторға айналдыру үшін хэш-функция көмегімен *hach m*-разрядты екілік санға сәйкестендіріледі. Chord *hach* ретінде SHA-1-ді пайдаланады. Бұл – біз BitTorrent-ті сипаттаған кезде айтқан хэш-функция. Біз оны *8-парауда* криптографияны талқылаған кезде қарастырамыз. Әзірше оның ұзындығы айнымалы байт жолын кездейсоқ 160-разрядты екілік санға бейнелейтін жай функция екенін айтып кеткен жеткілікті. Сондықтан біз оны кез келген IP-адресті **түйін идентификаторы (node identifier)** деп аталатын 160-разрядты санға айналдыру үшін пайдалана аламыз.

*7.43. а-суретін*де *m=5* болғандағы түйіндер идентификаторының шеңбері көрсетілген (Әзірше доғаның ортада екенін елемеңіз). Кейбір идентификаторлар түйіндерге сәйкес келеді, көбі сәйкес келмейді. Бұл мысалда идентифи-

каторлары 1, 4, 7, 12, 15, 20 және 27 түйіндер нақты түйіндерге сәйкес келеді және штрихталған. Қалғандары жоқ.



7.43-сурет. Шеңбер бойымен реттелген түйіндердің 32 идентификаторынан тұратын жиынтық (а); Штрихталғандары нақты машиналарға сәйкес келеді. Доғалар 1, 4 және 12 түйіндерінің көрсеткіштеріне сәйкес келеді. Доғалардағы белгілер – кестелер индексі. Көрсеткіштер кестесінің мысалы (ә)

Енді, *successor (k)* функциясын, шеңбер бойынша, сағат тілі бағытымен *k* кейінгі бірінші нақты түйін идентификаторы ретінде анықтайық. Мысалы, *successor* (6)=7, *successor* (8)=12, ал *successor* (22)=27.

Кілтте 160-разрядты санды алумен тұрғызылады – контент атын *hach* (демек, SHA-1) көмегімен хэштеу арқылы. Біздің сценариймізде контент аты – торрент. Сондықтан, *torrent* (торренті сипаттау файлы) кілтіке түрлендіру үшін *key=_hach(torrent)* есептейміз. Бұл есептеу, жай жергілікті *hach* функциясы.

Жаңа үйірді іске қосу үшін, түйін индекске (*torrent, my-IP-adress*) тұратын жаңа кілт-мән жұбын қосуы керек.

Бұны жүзеге асыру үшін түйін *successor(hach(torrent))*-ды *my-IP-adress*-ті сақтауын сұрайды. Осылайша, индекс түйіндер арасында кездейсоқ таралады.

Қателіктерге тұрақты болу үшін, деректерді p түйіндерде сақтауға әртүрлі p хэш-функцияны пайдалануға болады. Бірақ біз бұл жерде қателікке тұрақтылық жайлы сөз қозғамаймыз.

DHT құрастырылғаннан кейін, біраз уақыттан соң басқа бір түйін торрентті тауып, үйірге қосылып, контентті жазып алғысы келеді. Түйін *torrent* іздейді, алдымен ол *key* алу үшін оны хэштейді, сонан кейін сәйкес мән сақталған түйіннің IP-адресін табу үшін *successor(key)* пайдаланады. Мән – үйірдегі пирлер тізімі. Түйін тізімге өз IP-адресін қосып, BitTorrent хаттамасы бойынша контентті жүктеп, басқа пирлермен әрекеттесуі мүмкін.

Бірінші қадам қарапайым, ал екіншісі – қарапайым емес. Белгілі бір кілтке сәйкес түйіннің IP-адресін табу үшін, әр түйінге белгілі бір әкімшілік деректер құрылымын қолдап отыру керек. Олардың бірі – түйіндер идентификаторы шеңберіндегі өз ізбасушысының IP-адресі. Мысалы, *7.43-суретте* 7-түйін 4-түйін үшін ізбасушы, ал 12 – 7-түйін ізбасушысы.

Іздеу келесідей жүзеге асырылуы мүмкін. Сұраушы түйін өз ізбасушысына ізделінетін IP-адрес және кілт көрсетілген дестені жөнелтеді. Деста шеңбер бойымен, ізделінетін түйін идентификаторының ізбасушысын анықтағанша қозғалады. Бұл түйін өзінде кілтке сәйкес ақпараттың бар екендігін тексереді, егер ақпарат бар болса, оны IP-адресі бар сұраған түйінге тікелей жөнелтеді. Алайда, үлкен бір рангілі жүйеде барлық түйіндерді тізбек бойымен іздеу тиімсіз, себебі ізделінетін түйіндердің орташа саны *n/2*-ге тең. Іздеу жылдамдығын жоғарылату үшін әр түйін Chord-де **көрсеткіштер кестесі (finger table)** деп аталатын кестені қолдайды. Көрсеткіштер кестесінде *m* жазба бар, олар 0-ден *m-1*-ге дейін индекстелген және әрқайсысы өзінің нақты түйініне сілтейді. Әр жазбаның екі өрісі: *start* және *7.43 ә-суретінде* үш түйін үшін көрсетілгендей, ізбасушының IP-адресі бар. *k* идентификаторы бар түйіндегі *i* нөмірінің мәнін жазу үшін өрісте мәні:

$$start = k + 2^i (mod 2^m)$$

Ізбасушы IP-адресі = *successor(start[i]*)

болады. Әр түйіннің азғана түйіндер IP-адресін есте сақтайтынын айта кетейік және олардың көбі термин бойынша түйін идентификаторына жақын.

Көрсеткіштер кестесін пайдаланғанда *k* түйінде *key*-ді іздеу келесідей жүзеге асырылады. Егер, *key k* және *successor(k)* аралығында болса, онда *key* жайлы ақпараты бар түйін *successor(k)* болып саналады және іздеу тоқтатылады. Кері жағдайда, көрсеткіштер кестесінде, *start* өрісі *key*-дің ең жақын ізбасушысы болып саналатын жазба ізделінеді. Нақты осы жазбадағы IP-адреске іздеуді жалғастыру жайлы сұраныс жөнелтіледі. Бұл *key*-ге жақынырақ және тағы бірақ қосымша сұраныстардан кейін жауап табылатындығына үміт бар. Іс жүзінде межеге дейінгі арақашықтық іздеу екі есе қысқартылатын болғандықтан, іздеудің орташа саны құрайтындығын көрсетуге болады.

Бірінші мысал ретінде 1-түйінде *key=3* іздеуді қарастырайық. 1-түйін 3-тің өзі және оның 4-ізбасушы арасында жатқанын білетін болғандықтан, ізделінетін түйін 4, сондықтан іздеу 4-түйіннің IP-адресін қайтарумен аяқталады.

Екінші мысал ретінде 1-түйінде *key=16* іздеуді қарастырайық. *key=16* 1-ші және 4-түйін арасында жатпайтындықтан көрсеткіштер кестесіне жүгінуге тура келеді. 16-ның ең жақын ізбасушы - 9, сондықтан сұраныс 9-жазба IP-адресі бойынша, 12-түйінге жөнелтіледі. 12-түйін жауапты білмейді, сондықтан ол 16-түйіннің алдындағы ең жақын түйінді іздейді, сөйтіп 14-түйін табалады, ал ол 15-түйін IP-адресіне әкеледі. Сұраныс сонда жөнелтіледі. 15-түйін 16-ның өзі және ізбасушы (20) арасында жатқанын көреді, сөйтіп 1-түйінге 20-түйіннің IP-адресін жөнелтеді.

Түйінде үнемі қосылып, өшіп отыратын болғандықтан, Chord осы әрекеттерді басқара алуы керек. Біз, жүйе іске қосылған кезде түйіндер саны өте аз болды деп болжаймыз, сондықтан бірінші шеңберді және көрсеткіштер кестесін құру үшін олар тікелей ақпарат алмаса алды. Әрі қарай автоматтандырылған процедура қажет. Жаңа *r* түйіні қосылғысы келгенде, ол осы кезде бар бір түйінмен әрекеттесіп, оның *successor(r)* IP-адресін табуын сұрауы қажет. Сонан кейін жаңа түйін *successor(r)* оның ізбасушысын анықтайды. Енді жаңа түйін оларды *r*-ді шеңберге, өздерінің араларына қосуды сұрайды. Мысалы, егер *7.43-суретке* 24-түйін қосылғысы келсе, ол кез келген түйінді, *successor(24)* табуды сұрайды, біздің жағдайымызда ол 27-түйін. Сонан кейін ол 27-түйінді оның ізбасушысы (20-түйін) жайлы сұрайды. Бұдан кейін ол 20-шы және 27-түйінге өзінің бар екенін хабарлайды, әрі қарай 20-түйін 24-түйінді ізбасушы, ал 27-түйін 24-түйінді өзінің ізашары ретінде пайдалана бастайды. Сонымен бірге, 27-түйін 21-түйіннен 24-түйінге дейінгі кілттер диапазонын 24-түйінге береді. Енді 24-түйін толығымен іске қосылды.

Алайда, енді көптеген көрсеткіштер кестесінде қате ақпарат жазылған. Оны түзету үшін әр түйін, дүркін-дүркін *successor* сұрап, әр көрсеткішті қайталап есептейтін фондық үдерісті қолдап отырады. Осы сұраныстардың бірі жаңа түйінді көрсеткен кезде, көрсеткіштің сәйкес жазбасы жаңартылады. Түйін дұрыс өшкен кезде, ол кілттерді өзінің ізбасушысына тапсырады және оны оған өшетіндігі жайлы хабарлайды, сондықтан өшетін түйіннің ізашары оның ізбасушысымен байланысады. Егер түйін істен шықса, мәселе туындайды, себебі оның ізашушысының нақты ізбасары жоқ. Бұл мәселені жеңілдету үшін, әр түйін тек өзінің нақты ізбасушысын ғана емес, сонымен бірге реті бойынша *s-1* істен шыққан түйіндерді жіберіп, шеңберді қосу үшін, келесі *s* ізбасушыларды қадағалап отырады.

DHT құрастырылғаннан бері, оған байланысты көптеген зерттеулер жүргізілді. Олардың қаншалықты көп екенін білу үшін «Желілер аумағындағы зерттеулердің қайсысы жиі дәйек сөз алынады?» деген сұрақ қоялық. Сіздер Chord (Stoica және басқалар, 2001) теңесудің қиын екенін көресіздер. Осы зерттеулерге қарамастан DHT қосымшалары баяу пайда бола бастады. BitTorrent-тің кейбір клиенттері DHT-ты біз сипаттағандай толық таратылған трекерді қамтамасыз ету үшін пайдаланады. Үлкен коммерциялық бұлттық қызметтер де, мысалы, Amazon компаниясының Dynamo-сы тәрізді, DHT тәсілдерін қосады (DeCandia және басқалар, 2007).

7.6. ТҮЙІНДЕМЕ

ARPANET-те ат берудің өзі о баста оңай болған жоқ. Мәтіндік файлда барлық хосттар аттары және оларға сәйкес IP-адрестер көрсетілді. Әр түн сайын барлық машиналарға осы файл жүктеліп отырды. Бірақ ARPANET Интернетке алмастырылған кезде оның көлемі шексіз ұлғайды, бұл ат берудің әлдеқайда талғампаз және динамикалық схемасын құруды талап етті. Қазіргі кезде Интернеттегі домендерге ат беру, домен аттары қызметі (DNS) деп аталатын иерархиялық схема бойынша жүзеге асырылады. Ол Интернетке қосылған барлық машиналарды ағаштар жиынтығына ұйымдастырады. DNS жүйесінің жоғарғы деңгейінде жалпыға белгілі рулық домендер орналасқан, *com, edu* және екіжүзге жуық ұлттық домендерді қоса алғанда. DNS бүкіл әлемде орналасқан, таратылған деректер базасы ретінде ұйымдастырылған. DNS-серверге жүгіну арқылы, үдеріс Интернеттегі домен атын алып, оны IP-адреске түрлендіру арқылы нақты компьютермен әрекеттеседі.

Электронды пошта – Интернеттің ең танымал қосымшаларының бірі. Оны қазірде барлығы, мектеп жасындағы жас баладан бастап, қарттарға дейін пайдаланады. Электронды поштаның көптеген жүйелері RFC 5321 және 5322 сипатталған стандарттарға сәйкес келеді. Мәлімдемелер MIME-тақырыптарында көрсетілген ASCII-тақырыптардың және әртүрлі типтегі деректерден тұрады. Пошта жеткізу үшін тасымалдау агенттеріне беріледі және олардан веб-қосымшаларды қоса, басқа да тұтынушылар агенттеріне беру үшін алынады. Жөнелтілген пошта, хост-жөнелтуші және хост-қабылдаушы арасында TCP-байланыс орнататын, SMTP хаттамасы бойынша жеткізіледі.

Дүниежүзілік өрмек көптеген адамдар интернет деп қабылдайтын қосымша болып саналады. Бастапқыда ол HTML тілінде жазылған және браузер және сервер арасындағы TCP-байланыс арқылы жүктелетін, басқа парақтарға гиперсілтемесі бар парақтарды өңдеуге арналған қосымша болатын. Қазіргі кезде контенттің көбі динамикалық болып келеді, не сервер жағында (мысалы, PHP бар) немесе браузер жақта (мысалы, JavaScript бар). Сервердегі динамикалық парақтар ішкі деректер базасымен қоса алғанда, электронды сауда және іздеу тәрізді веб-қосымшаларды пайдалануға мүмкіндік береді. Электронды пошта тәрізді браузер динамикалық парақтары толыққанды қосымшаларға айналып келеді, олар браузер ішінде жұмыс істейді және қашықтықтағы сервермен байланысу үшін веб-хаттамаларды пайдаланады. Кэштеу және тұрақты байланыс Дүниежүзілік өрмек өнімділігін жақсарту үшін кеңінен қолданылады.

Арна енінің кеңейгеніне және мобильді құрылғылар қуатының өскеніне қарамастан, Дүниежүзілік өрмекті мобильді телефондар арқылы пайдалану оңай іс емес. Веб-сайттар жиі суреттері кішірейтілген және экраны кішкентай құрылғылар үшін қозғалуы қарапайымдатылған парақтардың түрлендірілген версияларын жөнелтеді.

Веб-хаттамалар машиналар арасындағы коммуникация үшін жиі қолда-

нылады. Компьютерде жеңіл өңделетін контентті сипатау үшін, HTML-ге қарағанда XML ұнамдырақ. SOAP, HTTP көмегімен XML-мәлімдемелерді тасымалдайтын, RPC-механизм болып саналады.

2000 жылдан бері, Интернет дамуының негізінде сандық аудио және бейне жатыр. Қазіргі кезде интернет-трафиктің үлкен бөлігі – бұл бейне. Оның едәуір бөлігі веб-сайттардан аралас хаттамалар (RTP/UDP және RTP-HTTP/TCP қоса) көмегімен тасымалданады. Нақты уақыттағы медиа көптеген тұтынушыларға жөнелтіледі. Оған әртүрлі бағдарламаларды тасымалдайтын радио- және телестанциялар жатады. Аудио және бейне, сонымен бірге нақты уақыттағы телеконференциялар үшін қолданылады. Көптеген қоңырау шалуларда дәстүрлі телефон желілері емес, IP үстіндегі дауыс пайдаланылады, сонымен бірге олар үшін бейнеконференциялар да қолданылады.

Аздаған өте танымал веб-сайттар және көптеген аса танымал емес сайттар бар. Танымал сайттарға қызмет көрсету үшін контентті тарату желілері құрастырылды. Олар клиентті жақын арадағы серверге бағыттау үшін DNS-ті пайдаланады. Серверлер бүкіл әлем бойынша, деректерді өңдеу орталықтарында орналасқан. Балама ретінде P2P-желілері бірнеше машинаға фильм тәрізді контентті бөліп сақтауға мүмкіндік береді. Машина саны өскен сайын контентті тарату сыйымдылығы да өседі, сондықтан мұндай желілер ірі сайттармен бәсекелесе алады.

СҰРАҚТАР

1. Көптеген коммерциялық компьютерлердің әртүрлі және бірегей үш идентификаторы бар. Олар қандай болып келеді?
2. 7.1-листингінде *laserjet* сөзінен кейін нүкте қойылмаған. Неліктен?
3. Кибертреррорист бүкіл әлемдегі барлық DNS-серверлерді құрауға мәжбүрлейтін жағдайды қарастырыңыз. Бұл Инттернетті пайдалануды қалайша өзгертуі мүмкін?
4. DNS TCP орнына UDP пайдаланады. Егер DNS-десте жоғалса ол автоматты түрде қалпына келтірілмейді. Бұл жағдай мәселе туындата ма? Егер туындатса ол қалай шешіледі?
5. Джон біртума домен атын алғысы келеді және доменнің екінші атын генерациялау үшін рандомизация программасын пайдаланады. Ол домендік атты *com* доменінде тіркемек. Генерацияланған домендік ат 253 белгіден тұрады. Домен осындай атты тіркеуге рұқсат бере ме?
6. Компьютердің бір DNS аты және бірнеше IP-адресі бола ала ма? Бұл қалай болуы мүмкін?
7. Соңғы кезде жеке веб-сайты бар компаниялар саны күрт өсті. Нәтижесінде *com* доменінде мыңдаған фирманың сайты бар, бұл жоғары деңгейдің осы доменіне қызмет көрсететін сервердің асыра жүктелуіне әкеледі. Осы мәселені ат беру схемасын өзгертпей (демек, жаңа жоғары деңгей доменін құрастырмай) шешетін тәсілді ұсыныңыз. Мүмкін, сіздің шешіміңіз клиенттік программаға өзгерту енгізуді қажет ететін шығар.
8. Кейбір электронды пошта жүйелері *Content-Return:* өрісін қолдайды. Бұл өрісте хат адресатқа жеткізілмеген жағдайда оның мазмұнын кері қайтару керектігі көрсетіледі. Бұл өріс конверт құрамына кіре ме, әлде хат тақырыбына кіре ме?
9. Электронды пошта жүйесі e-mail адрестер кітабын сақтайды. Оның көмегімен тұтынушы қажет адресті таба алады. Осындай кітап бойынша іздеу жүргізу үшін адресаттар, аты стандартты компоненттерге (мысалы, аты, трегі) бөлінуі тиіс. Сәйкес халықаралық стандартты құрастыру үшін шешуді қажет ететін кейбір мәселені талқылаңыз.
10. Көптеген қызметкерлері бар ірі заң фирмасы әр қызметкеріне жеке электронды адрес ұсынады. Әр қызметкердің e-mail адресі логиннен, @ белгісінен, фирма атынан және com доменінен тұрады. Алайда, фирма логин форматын дәл анықтаған жоқ, сондықтан кейбір қызметкерлер өз аттарын, кейбіреулері – тегін, ал екінші бірі аты-жөнінің бірінші әріптерін пайдаланады. Енді фирма форматты дәл анықтамақ болды, мысалы,

    аты.тегі@фирма_аты.com.

    Бұл форматты барлық қызметкерлер қолданылуы тиіс. Бұны аз шығынмен және қалайша жүзеге асыруға болады.

11. Ұзындығы 4560 байт екілік файл бар. Файлды base64 жүйесінің көмегімен кодтағаннан кейін оның ұзындығы қандай болады? CR+LF символдар жұбы әр 110 байт сайын және мәлімдеме соңына енгізіледі.
12. Мәтінде көрсетілмеген бес MIME типін атаңыз. Ақпаратты браузер баптауынан немесе Интернеттен алуға болады.
13. Айталық, сіз досыңызға MP3-файлын жөнелткіңіз келеді, алайда сіздің досыңызға қызмет көрсететін провайдер кіріс пошта көлеміне 1 Мбайтқа дейін шектеу қояды, ал файл 4 Мбайт. Осы мәселені RFC 5322 және MIME-ні пайдаланып шешуге бола ма?
14. Айталық, Джон өз істеріне қатысты барлық хаттар келетін жұмыстағы электронды пошта адресіне, хаттардың барлығын өзінің жеке, жұбайы екеуі бірлесіп пайдаланатын электронды адреске қайта жөнелтетін автоматты жауап беру механизмін орнатты. Джонның жұбайы бұны білмейді және жеке жәшігіндегі демалыстық доменді іске қосып қойды. Джон қайта жөнелтуді іске қосты, бірақ жұмыстағы пошта жәшігіне демалыстық доменді орнатқан жоқ. Осы жәшікке хат келгенде не болады?
15. RFC 5322 тәрізді кез келген стандартта грамматиканың дәл сипаттамасы болуы тиіс – бұл әртүрлі іске асырулардың әрекеттесуі үшін қажет. Тіпті, ең қарапайым элемент дәл анықталуы тиіс. Мысалы, SMTP тақырыптарында символдар арасында бос орын қалдыруға болады. Осы бос орындардың шындыққа жанасатын екі балама анықтамасын айтып беріңіз.
16. Демалыстық домен тұтынушы агенттің бе, әлде мәлімдеме жөнелту агентінің бір бөлігі болып санала ма? Оның тұтынушы агенті көмегімен бапталатыны түсінікті, бірақ жүйенің қандай бөлігі автоматты жауаптарды жөнелтумен айналысады? Жауабыңызды түсіндіріңіз.
17. Chord алгоритмінің қарапайым версиясында теңқұқықты іздеу үшін көрсеткіштер кестесі қолданылмайды. Оның орнына олар шеңберді екі бағытта тізбек бойынша қарап шығады. Түйін іздеуді қандай бағытта жүргізу керектігін анықтай ала ма? Жауабыңызды түсіндіріңіз.
18. IMAP тұтынушыларға қашықтықтағы пошта жәшігінен поштаны сұрауға және жүктеуге мүмкіндік береді. Бұл IMAP-ты пайдаланатын кез келген клиенттік программа кез келген сервердегі пошта жәшігіне жүгіну үшін пошта жәшіктерінің ішкі форматы стандартталған болу керек дегенді білдіре ме? Өз жауабыңызды аргументтермен толықтырыңыз.
19. *7.43-суреттегі* Chord шеңберін қарастырыңыз. Айталық, 18-түйін кенеттен онлайн шықты делік. Бұл суретте көрсетілген көрсеткіштер кестесінің қандай бөлігіне әсер етеді?
20. Webmail қандай хаттамаларды пайдаланады: POP3, IMAP немесе екеуінде? Егер бұл екеуінің тек біреуі болса, онда неліктен ол? Егер екеуі

де болмаса, онда нақты қолданылатын хаттама бұл екеуінің қайсысына ұқсас?

21. Веб-парақты тасымалдаған кезде MIME тақырыптармен бірге алынады. Неліктен?

22. Тұтынушы Internet Explorer және Firefox браузерлерінде MIME-типі бірдей сілтемеге шерткен кезде екі түрлі көмекші қосымшалардың ашылуына әкелуі мүмкін бе? Жауабыңызды түсіндіріңіз.

23. Бұл жайлы мәтінде айтылмағанымен, URL-дің DNS аттар орнына IP-адресті пайдаланатын нұсқасы бар. Неліктен DNS-ат санмен айғақтауға болмайтынын түсіндіру үшін осы ақпаратты пайдаланыңыз.

24 Стэнфорд университеті математика факультетінің қызметкері жаңа құжат жазып, әріптестерінің пікірін білу үшін осы құжатты FTP арқылы таратпақ болды. Ол құжатты *ftp/pub/forReview/newProof.pdf* каталогына орналастырды. Осы құжаттың URL-і қандай болады?

25. 7.10-кестеге сәйкес *www.aportal.com* клиент артық көретін парақтарды cookie түрінде сақтайды. Мұндай шешімнің кемшілігі: cookie көлемі 4 Кбайтпен шектелген және тұтынушы жайлы көп деректерді сақтау керек болса (мысалы, оның парақта көптеген биржалық тұжырымдамалар, түрлі тауарлар категориясы жайлы арнайы ұсыныстар және т.б.), онда бұл сипаттамалар тез арада 4 Кбайттық шекке жетуі мүмкін. Осы мәселе туындамайтын, тұтынушы жайлы ақпаратты сақтаудың балама әдісін ұсыныңыз.

26. Қандай да бір «Жалқауларға арналған банк» өзінің жалқау клиенттері үшін арнайы онлайн банктік жүйе ұйымдастырғысы келеді. Жүйеде тіркеліп, құпия сөз арқылы сәйкестендірілгеннен кейін тұтынушы клиенттің идентификаторы көрсетілген cookie-файл алады. Осылайша, жүйеге әрбір кірген сайын ол өзінің сәйкестендіру деректерін енгізбейді. Сізге бұл идея ұнай ма? Ол жұмыс істей ме? Бұл идея қаншалықты жақсы деп ойлайсыз?

27. а) Келесі HTML-тәгті оқыңыз:

```
<h1 title="this is the header"> HEADER 1 </h1>.
```

Браузер қандай жағдайда *TITLE* атрибутын пайдаланады және қалай?
ә) *TITLE* атрибутының *ALT* атрибутынан айырмашылығы неде?

28. HTML тілінде гиперсілтеме суретпен қалай ұқсастырылады? Мысал келтіріңіз.

29. *Тұтынушы_аты@домен_аты.com* электронды адресіне сілтемесі бар HTML-парақ жазыңыз. Тұтынушы осы сілтемеге шерткен кезде не болады?

30. Университет студенттерін есепке алатын XML-парақ жазыңыз. Әр студент үшін бірегей аты, мекенжайы және орташа балл көрсетілетін болсын.

31. Аталған әр жағдай үшін (1) мүмкін бе және (2) PHP-скрипті пайдаланған жақсы ма, әлде JavaScript пайдаланған дұрыс па? Неліктен?
    а) Кез келген ай күнтізбесі, 1752 жылдың қыркүйегінен бастап.
    ә) Амстердамнан Нью-Йоркқа дейінгі рейстер кестесі.
    б) Тұтынушы енгізген полиномды коэффициентімен шығару.

32. JavaScript-те, кірісте 2-ден үлкен бүтін сандық мәнді қабылдайтын және осы енгізілген санның қарапайым екендігін хабарлайтын программа жазыңыз. JavaScript-тегі if және while синтаксистері C және Java-дағы сәйкес өрнектерге ұқсас. Еркін модуль бойынша арифметикалық өрнекті орындау: %. Егер x санының квадрат түбірін табу керек болса, *Math.sqrt(x)* функциясын пайдаланыңыз.

33. HTML-парақ:

```
<html> <body>
<a href="www.info-source.com/welcome.html"> </a>
</body> </html>
```

    кодтан тұрады.
    Тұтынушы гиперсілтемеге шерткен кезде TCP-байланыс ашылады және серверге қандай да бір жолдар тізбегі жөнелтіледі. Осы жолдарды атап шығыңыз.

34. *If-Modified-Since* тақырыбын кэште сақталған парақтың өзектілігін тексеру үшін пайдалануға болады. Сәйкес сұраныстар суреттері, дыбыстар, бейне және т.б. бар, сонымен бірге қарапайым HTML парақтарға да жіберіледі. Сіз қалай ойлайсыз, осы тәсілдің тиімділігі қарапайым HTML парақтарға қарағанда JPEG суреттері бар парақ үшін жоғары бола ма? «Тиімділік» сөзінің мағынасын жақсылап ойланыңыз, содан кейін жауабыңызды түсіндіріңіз.

35. Өте маңызды спорт оқиғасы күні (айталық, танымал спорт түрі бойынша халықаралық чемпионат өтетін күні) тұтынушылардың үлкен саны осы оқиғаның ресми сайтына кіруге асығады. Бұл жағдай 2000 жылғы Флоридадағы сайлау кезіндегі трафиктің күрт өсуіне ұқсас па? Неліктен?

36. Жеке провайдерге контентті жеткізу желісі ретінде жұмыс істеудің мағынасы бар ма? Егер бар болса, бұл жүйе қалай жұмыс істеуі керек? Егер мағынасы жоқ болса, бұл идеяның нашар жағы неде?

37. Айталық аудио үшін CD тығыздауды пайдаланбайды. Ойнау ұзақтығы екі сағатқа тең дискіге неше мегабайт дерек болуы керек?

38. *7.18 б-суретінде*, 9 деңгейлік сигналды ұсыну үшін 4-биттік саналымды пайдаланғандықтан, кванттау шуы пайда болатыны көрсетілген. Бірінші саналым (нөлде) дәл болып келеді, ал қалғандары – дәл емес. Уақыт сәтінің 1/32, 2/32, және 3/32 кезеңдерінде алынған саналымдардың салыстырмалы қателігі нешеге тең?

39. Психоакустикалық модельді интернет-телефония жүйесінің қажет өткізгіштік қабілеттілігін азайту үшін пайдалануға бола ма? Егер болатын болса, онда модель жұмыс істейтін шарттар қандай (егер олар бар болса)? Болмаса, неліктен?

40. Аудиоағын сервері бір бағытта 100 мс кідіріс беретін ойнатушыдан қашықта орналасқан. Ол деректерді 1 Мбит/с жылдамдықпен береді. Егер ойнатушының 2 Мбайт буфері бар болса, осы буфердің жоғарғы және төменгі толу шектерінің орналасуы жайлы не айтуға болады?

41. Дауысты IP үстімен жөнелткен кезде, ағындық аудионы жөнелткендегідей желі аралық экранмен мәселе туындайды ма? Жауапты талқылаңыз.

42. Өлшемі 1200×800 пиксел және секундына 50 кадрда бір пикселге 16 бит тығыздалмаған түрлі түсті суретті тасымалдау үшін қандай жылдамдық қажет?

43. MPEG кадрындағы бір биттегі қателік бірден артық кадрға нұқсан келтіре ала ма? Өз жауабыңызға дәлел келтіріңіз.

44. 50 000 клиентке қызмет көрсететін бейне сервер мысалын қарастырайық. Әр клиент ай сайын үш фильм көреді. Айталық, фильдердің үштен екі бөлігін көру 21:00-де басталады. Сервер осы кезеңде бірмезетте неше фильм тасымалдауы тиіс? Егер әр фильмді тасымалдау үшін 4 Мбит/с қажет болса, сервердің желімен байланысуы үшін OC-12 түріндегі неше байланыс қажет?

45. Айталық, 10 000 фильм сақталған бейне серверге қол жеткізу үшін Ципфа заңдылығы сақталады. Сервер жадысында 1000 ең танымал фильм, ал қалған 9000 фильм дискіде сақталған делік. Сұраныстардың қандай бөлігі жадыға бағытталады? Осы мәнді сандық түрде есептейтін кішігірім программа жазыңыз.

46. Қандай да бір жаман адамдар бүкіл әлемге танымал *www.microsoft.com* ұқсас және тұтынушы адресті терген кезде кездейсоқ кішкене қателік жіберіп кіре алатын, домен атын тіркеді делік. Осындай домендерге, кем дегенде, бес мысал келтіріңіз.

47. *www.сөз.com* атымен тіркелген көптеген сайттар бар, мұнда сөз – әдеттегі ағылшын сөзі. Әрбір келтірілген категория үшін бес вебсайттан тұратын тізім жасаңыз және олардың мағынасын қысқаша сипаттаңыз (мысалы, *www.cosmos.com* – ғарыш мәселелеріне арналған сайт). Категориялар тізімі: жануарлар, азық-түлік, тұрмыс заттары, дене бөлшектері. Соңғы категория үшін тек белден жоғары орналасқан дене бөлшектерін көрсетіңіз.

48. *6.1-листингін* HTTP 1.1. хаттамасымен жұмыс істеу үшін GET командасын пайдаланатын нақты веб-серверге айналдырып, қайта жазыңыз. Ол Host мәлімдемесіне әрекет етуі тиіс. Сервер дискіден жақын арада сұралған файлдарды кэштеуі және сұраныстарға файлды мүмкіндігінше кэштен алып қызмет көрсетуі тиіс.

8-ТАРАУ
ЖЕЛІЛЕРДЕГІ ҚАУІПСІЗДІК

Алғашқы пайда болған он жылдықта компьютерлік желілер алдымен университет зерттеушілері электронды поштамен мәлімет алмасу және корпорация қызметкерлері принтерді бірлесіп пайдалану үшін қолданды. Мұндай жағдайда қауіпсіздік сұрақтары аса көп назарды талап етпеді. Алайда, қазір, миллиондаған адамдар өз банк есепшоттарын басқару, салық декларациясын толтыру және интернет-дүкендерден тауар сатып алу үшін желіні пайдаланатын жағдайда көптеген кемшіліктер анықталып, желілік қауіпсіздік мәселесі өте маңызды болып отыр. Біз бұл тарауда желілік қауіпсіздікті әртүрлі тұрғыдан қарастырамыз, күтпеген кедергілерді нұсқаймыз және желі қауіпсіздігін нығайтатын алгоритмдер мен хаттамаларды талқылаймыз.

Қауіпсіздік тақырыбы өте кең және адам күнәларымен байланысты көптеген мәселелерді қамтиды. Қарапайым түрде, қауіпсіздік – бұл басқа тұтынушыға арналған мәлімдемені бөгде адамдар оқи алмайды, тіпті, одан бетер өзгерте алмайды деген кепілдік. Қауіпсіздік – бұл қашықтықтағы қызмет түрлеріне құқығы жоқ адамдардың заңсыз қол жеткізу талпынысын болдырмау. Қауіпсіздік жүйесі: «Есепшотты жұмаға дейін төлеу керек» деген мәлімдемені салық қызметі жазды ма, әлде бұл бұрмалау ма, осыны анықтай білуі керек. Бұдан басқа, қауіпсіздік жүйесі мәлімдемені ұстап қалу және қайта қалпына келтіру және осы мәлімдемені жөнелткенін мойындағысы келмейтін адамдармен байланысты мәселелерді шешеді.

Қауіпсіздік мәселелерінің көбі өзі үшін пайда табу немесе басқаларға залал келтіргісі келетін зиянкестер салдарынан орын алады. Бірнеше кеңінен таралған

бұзақылық түрлері *8.1-кестеде* келтірілген. Осы тізімнен желі қауіпсіздігін қамтамасыз ету, жәй программалық қателіктерді түзетуден әлдеқайда күрделі түсінікті болды деп ойлаймыз. Жиі ақылды, өзіне кәміл сенімді, жақсы қаржыландырылған қарсыластан айланы асырып кету керек болады. Сонымен бірге, кездейсоқ бұзақыны тоқтата алатын шаралар қауіпті қылмыскерге әсер ете қоймайтыны белгілі. Полиция жинаған статистика, көптеген шабуылдардың сырттан (байланыс тораптарын тыңдайтын адамдармен), ал іштен – қызғаншақ немесе бірнәрсеге көңілі толмайтын адамдармен жасалатынын көрсетеді. Демек, қауіпсіздік жүйесі бұл фактіні де ескеруі тиіс.

Алғашқы жақындауда желі қауіпсіздігі мәселесін төрт қиылысқан аудандарға бөлуге болады: құпиялық, сәйкестендіру, міндеттердің қатаң түрде орындалуын талап ету және біртұтастықты қамтамасыз ету.

Құпиялық (конфеденциялдық) – ақпараттың қуатталмаған, арамза тұтынушылардың қолына түсуін болдырмау дегенді білдіреді. Желі қауіпсіздігі жайлы сөз қозғағанда алдымен ойға осы мәселе келеді. Сәйкестендіру – әңгімелесушіге құпия ақпаратқа қол жеткізуге мүмкіндік бермес немесе онымен іскерлік қарым-қатынасқа түспес бұрын, оның кім екенін анықтауға мүмкіндік береді. Міндеттердің қатаң түрде орындалуын талап ету – бұл қол қоюға байланысты. Сіздің клиентіңіз, шынымен де электронды пошта арқылы бағасы 89 цент, сол жақ бұрандалы он миллион винтке сұраныс беріп, кейіннен ол баға 69 цент болған десе, өз сөзіңіздің дұрыстығын қалай дәлелдейсіз? Соңында, біртұтастықты бақылау – сіз қабылдаған мәлімдемені жол бойы зиянкестер түрлендірмегеніне және қолдан жасамағанына қалайша сенімді болуға болады?

8.1-кесте. Бұзақылар түрлері және олардың мақсаты

Бұзақы	**Мақсат**
Студент	Әуестіктен біреудің хатын оқу
Хакер	Бөтен біреудің қауіпсіздік жүйесінің мықтылығын тексеру; деректер ұрлау
Сауда агенті	Тек Андорраның емес, бүкіл Еуропаның өкілі болып көріну
Бизнесмен	Бәсекелестік стратегиялық маркетинг жоспарын барлау
Жұмыстан қуылған қызметкер	Жұмыстан қуып жібергені үшін фирмадан кек алу
Бухгалтер	Компания ақшасын ұрлау
Биржа брокері	Электронды пошта арқылы клиентке берген уәдені орындамау
Алаяқ	Кредиттік карта нөмірін сату мақсатында ұрлау
Тыңшы	Қарсыластың әскери немесе өндірістік құпиясын білу
Террорист	Бактериялогиялық қару өндірісі құпиясын ұрлау

Осы аспектілердің барлығы (құпиялық, сәйкестендіру, міндетті қатаң түрде орындау және біртұтастықты қамтамасыз ету) дәстүрлі жүйелерде кездеседі, бірақ біршама ерекшеліктері бар. Құпиялық және біртұтастық тапсырыс хаттары және құжаттарды өртенбейтін сейфте сақтау арқылы жүзеге асырылады. Бүгінгі күнде пошта поезын ұрлау Джесси Джеймс заманынан әлдеқайда қиын.

Бұдан басқа, адамдарға бірегей қағаз құжатты оның көшірмесінен айыру қиын емес. Тексеру үшін нағыз банктік чектің фотокөшірмесін жасаңыз. Дүйсенбі күні нағыз чекті қолма-қол ақшаға айналдырып көріңіз. Ал, келесі сейсенбі күні, чектің фотокөшірмесін қолма-қол ақшаға айналдырып көріңіз. Банк қызметшісінің тәртібін бақылауға тырысыңыз. Өкінішке орай, нағыз чек пен оның электронды көшірмесін бір-бірінен ажырату қиын. Банктер бұған үйренгенше біраз уақыт керек.

Адамдар бір-бірін түрлі жолдармен таниды, мысалы, беп әлпетіне қарап, даусы және жазу ерекшелігі бойынша. Қағаз құжаттардың нақтылығы баспа бланктердегі қолтаңбамен, мөрмен, рельефті белгілермен және т.б. қамтамасыз етіледі. Жасандылықты қағаз, сия және жазу ерекшеліктері мамандар анықтауы мүмкін. Электронды құжаттармен жұмыс жасағанда бұның бәрі қолжетімсіз. Демек басқа шешімдер қажет.

Біздің шешімдеріміздің мағынасын талқыламас бұрын, бірнеше минут уақытымызды бөліп, желілік қауіпсіздік жүйесінің хаттамалар стегінің қай деңгейіне жататындығын анықтап алғанымыз дұрыс болар. Ол үшін қандай да бір орын табу қиын емес болар. Әр деңгей өз үлесін қосу керек. Физикалық деңгейдегі жасырын тыңдаумен тасымалдаушы кабельдерді (немесе оптытылшықты) жоғары қысым көмегімен инертті газбен толтырылған герметикалық құбырға орналастыру арқылы күресуге болады. Құбырды тесуге әрекеттену газдың шығуына әкеледі, нәтижесінде қысым төмендеп, дабыл қағылады. Бұндай техника кейбір әскери жүйелерде қолданылады.

Арналық деңгейде екінүктелі торап арқылы тасымалданатын дестелер тропқа берілер алдында шифрлануы, ал қабылданар алдында қайта қалыпқа келтірілуі мүмкін. Бұның егжей-тегжейі тек арналық деңгейге белгілі және жоғары деңгейлер тіпті не болып жатқанын білмеуі де мүмкін. Алайда, егер десте бірнеше маршруттауыштан өту керек болса, онда мұндай шешім жұмыс істемейді, себебі дестені әр маршруттауышта қайта қалпына келтіріп отыру керек болады, бұл дестені маршруттауыш ішіндегі шабуыл алдында қорғансыз етеді. Бұдан басқа, мұндай тәсіл қорғауды қажет ететін жеке сеанстарды (мысалы, интернет-дүкендерде сатып алуларды жүзеге асыру) қорғауға мүмкіндік бермейді. Осылай десек те, байланыс арнасында **шифрлау (link encryption)** деп аталатын тәсілді кез келген желіге жеңіл қосуға болады және ол жиі пайдалы болып келеді.

Желілік деңгейде сырттан келген күмәнді дестелерді қабылдамайтын желіаралық экрандар орналастырылуы мүмкін. IP-қорғауда осы деңгейге жатады.

Транспорттық деңгейде байланысты бір басынан екінші басына дейін толығымен шифрлауға болады. Ең жақсы қорғанысты тек осындай толассыз шифрлау ғана қамтамасыз етеді.

Соңында сәйкестендіру және міндетті қатаң түрде орындау тек қолданбалы деңгейде шешілуі мүмкін.

Сонымен, желідегі қауіпсіздік – бұл барлық деңгейлерді қамтитын мәселе екені белгілі болды, сондықтан оған жеке тарау арналады.

Бұл тараудың үлкен, маңызды және көптеген техникалық сипаттамалардан тұратындығына қарамастан, қазіргі кезде желілік қауіпсіздік мәселесі орынсыз болып көрінуі мүмкін. Себебі қауіпсіздік жүйесіндегі «тесіктердің» көбі қызметкерлердің олақтығынан, ақпаратты қорғау процедурасына ұқыпсыз қарайтындығынан, өздері қорғайтын нысанға жауапкершілікпен қарамауынан, көптеген жүзеге асырулар қателігінен орын алатыны белгілі. Осы аталған қателіктер қуатталмаған тұтынушылардың жүйеге енуіне және «әлеуметтік инженерия» деп аталатын тәсілдерді пайдаланып, клиенттерді алдап пароль-дерін ашуға жол береді. Байланыс тораптарын тыңдап, алынған деректерді қайта шифрлайтын интеллектуал-қылмыскер салдарынан орын алатын мәселелер үлесі салыстырмалы түрде аз. Өзіңіз ойлап көріңіз, адам далада тауып алған магниттік кредит картасымен кез келген банк бөліміне келе алады, PIN-код ұмытып қалдым, ал кодты жазып қойған қағазды үйде әмиянымның ішінде тастап кетіппін десе, онда құпия шифрды бірден «есіне салады» (клиентпен жақсы қарым-қатынасты сақтап қалу үшін не істемейсің). Бұндай жағдайда әлемдегі ешбір криптографиялық жүйе көмектесе алмайды. Бұл ойда Росса Андерсон (Anderson, 2008a) кітабы шынымен таң қалдырады. Онда әртүрлі өндіріс салаларындағы қауіпсіздік жүйелерінің қойған міндетті атқара алмайтындығының жүздеген мысалдары келтірілген. Бұл келеңсіздіктердің орын алу себебі, жұқалап айтқанда, іс жүргізу ұқыпсыздығы немесе қарапайым қауіпсіздік ережелерін сақтамау. Осылай десек те, басқа фактолар жоқ жағдайда электрондық коммерцияны тұрғызудың техникалық негізі – криптография.

Физикалық деңгейден басқа барлық деңгейлерде желілерде ақпаратты қорғау криптография қағидаларына негізделеді. Сондықтан біз қауіпсіздік жүйесін қарастыруды криптография негіздерін егжей-тегжейлі талқылаудан бастаймыз. *8.1-«Криптография» бөлімінде* біз базалық қағидаларды қарастырамыз. Ары қарай «Ашық кілттерді басқару» бөліміне дейін (*8.2-8.5 бөлімдері*) криптографияда қолданылатын деректер құрылымының кейбір дәстүрлі мысалдарын қарастырамыз. Осыдан кейін біз осы тұжырымдамалардың компьютер желісінде қалай жұмыс істейтінін талқылып, теорияны тәжірибемен байланыстырамыз. Осы тараудың соңында технологияны және әлеуметтік сұрақтарға байланысты біраз ойлар айтылады.

Істі бастамас бұрын осы тарауда сөз болатын сұрақтарға тоқтала кетейік. Материалдар жинастыра отырып біз операциялық жүйелер немесе қосымшалармен емес, нақты компьютерлік желілермен байланысты (олардың

жіктерін ажырату қиын болса да) сұрақтарға көп бөлуге тырыстық. Мысалы, тұтынушының биометрияны пайдаланып қуатталуы, құпия сөзді қорғау, буфердің толуына байланысты шабуылдар, сайтаралық скриптинг көмегімен кодты енгізу, вирустар, құрттар және т.б. жайлы ешнәрсе айтылмайды. Осылардың барлығы *«Қазіргі заманғы операциялық жүйелер»* (Tanenbaum, 2007) кітабының *9-тарауында* айтылған. Жүйе деңгейіндегі қауіпсіздікті қамтамасыз етуге қызыққандар осы кітапқа жүгінуіне болады.

Сонымен, бастайық.

8.1. КРИПТОГРАФИЯ

Криптография (cryptography) сөзі гректің «құпия хат» деген мағынаны білдіретін сөзінен шыққан. Криптографияның бірнеше мыңдаған жылдардан тұратын ұзақ және қызықты тарихы бар. Біз бұл тарауда оның тек кейбір сәттерін алдағы ақпаратқа кіріспе болсын деп, еске салып кетеміз. Криптографияның толық тарихымен танысқысы келетіндерге Kahn (1995) кітабын оқуды ұсынамыз. Қазіргі жағдайдың егжей-тегжейін білу үшін Kaufman және басқалар, 2002 басылымын қараңыз. Криптографияның математикалық аспектілері жайлы Stinson (2002) кітабынан оқыңыз. Burnett және Paine (2001) басылымында баяндау тілі біраз формалды (математикалық тұрғыдан).

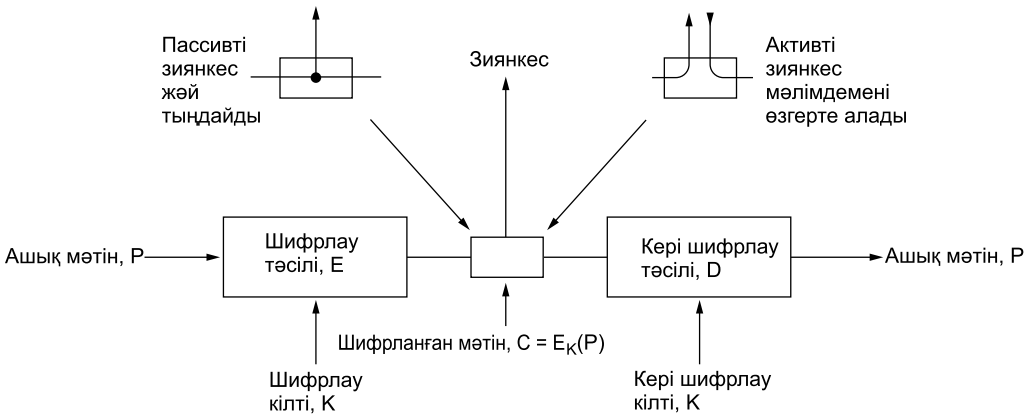
Кәсіби тұрғыда «шифр» және «код» сөздерінің бір-бірінен айырмашылығы бар. **Шифр (cipher)** – мәлімдеменің лингвистикалық құрылымына тәуелсіз символдық немесе биттік түрлендіруді білдіреді. **Код (code)**, керісінше, толық бір сөзді басқа сөзбен немесе символмен алмастырады. Қазіргі кезде кодттар қолданылмайды, алайда оның жақсы тарихы бар. Екінші дүниежүзілік соғыс кезінде америка әскерінің Тынық мұхитта қолданған коды ең жақсы код болып саналады. Құпия келіссөздер жүргізу үшін навахо үнділерінің тілі сөздерімен әскери терминдер белгіленген. Мысалы, *chay-dagahi-nail-tsaidi* (тікелей аудармасы: тасбақаны өлтіруші) танкке қарсы қару дегенді білдіреді. Навахо тілі дауыс ырғағына өте сезімтал (мағынасы түсінікті болу үшін дыбысты төмендету немесе жоғарылату қолданылады), өте күрделі, жазба түрі жоқ. Бірақ оның ең басты артықшылығы бірде-бір жапондық ол жайлы мүлдем ештеңе білмейді.

1945 жылдың қыркүйек айында *San Diego Union* газеті осы кодты былайша сипаттады: «Үш жыл бойы әскери теңізші қай жерде түсседе, жапондықтардың құлағы тек басқа дыбыстармен араласқан бейтаныс бүлкілдек шуды естіді. Мұның барлығы тибет монахының ұранын немесе ыстық су құйылған бөтелкенің босауын еске салады». Жапондықтар бұл кодты бұза алмады, навахо үнділерінің тілін білушілердің көбі жақсы қызметі және батылдығы үшін жоғары әскери марапаттарға ие болды. АҚШ-тың жапон кодын қалпына келтіріп, ал жапондықтардың навахо тілін білмегендігі Тынық мұхитта американдықтардың жеңіп шығуында басты рөл ойнады.

8.1.1. Криптография негіздері

Криптография өнерін тарихи пайдаланып, дамытып келе жатқан келесі төрт мамандықтың өкілдері: әскерилер, дипломатия корпусы, күнделік жүргізуші адамдар және көңілдестер. Олардың ішінде осы аумақтағы дамуға маңызды үлес қосқандар – әскерилер. Әскери ұйымдарда құпия мәлімдемелер дәстүрлі түрде шифрлау және тасымалдау төлемақысы нашар шифрлаушыларға беріліп отырды. Мәлімдемелер көлемінің үлкендігі бұл жұмысқа санаулы дегдар мамандарды тартуға мүмкіндік бермейді.

Компьютерлер пайда болғанға дейін криптографиядағы негізгі тежеуіш фактордың бірі, шифрлаушының қажет түрлендірулерді жиі ұрыс алаңында, қарапайым құрылғы көмегімен орындау мүмкіндігі болды. Сонымен бірге, бір криптографиялық тәсілден екіншісіне ауысу өте күрделі, себебі ол көптеген адамдарды қайта оқытуды қажет етеді. Дегенмен де, шифрлаушыны қарсыластардың ұстап алу қауіпі, қажет болған жағдайда криптографиялық тәсілдерді алмастыру амалын дамытуға итермеледі. Осы қайшылықтарға толы талаптар *8.1-суретте* көрсетілген шифрлау-керішифрлау үдерісі моделіне әкелді.



8.1-сурет. Шифрлау – кері шифрлау үдерісі моделі
(симметриялы кілті бар шифр үшін)

Ашық мәтін (plaintext) деп аталатын, шифрлау қажет мәлімдемелер екінші кіріс параметрі **кілт (key)** болып келетін функция көмегімен түрлендіріледі. **Шифрланған мәтін (ciphertext)** деп аталатын шифрлау үдерісінің нәтижесі әдетте радио немесе байланыстырушы арқылы жөнелтіледі. **Қарсылас** немесе **зиянкес (intruder)** тыңдап, шифрланған мәтінді көшіріп отыр деп жорамалданады. Алайда, осы мәлімдеме арналған қабылдаушыдан бір ерекшелігі – зиянкес кері шифрлау кілтін білмейді, сондықтан мәлімдемені кері шифрлау өте қиынға соғады, тіпті мүмкін емес. Кейде зиянкес байланыс

арнасын тек тыңдап қоймай (пассивті зиянкес), мәлімдемені жазып алып, кейіннен ол қабылдаушыға жеткенше өз мәлімдемесін қосуға немесе бірегей мәлімдемеге түрлендіруге (активті зиянкес) мүмкіндігі бар. Шифрларды бұзу өнері **криптосараптау (cryptanalysis)** деп аталады. Шифрды құрастыру (криптография) және оны бұзу (криптосараптау) өнері бірге **криптология (cryptology)** деп аталады.

Әдетте ашық мәтінді, шифрланған мәтінді және кілтті белгілеу үшін арнайы жазба шартты белгілері қолданылады. Біз *K* кілтінің көмегімен P ашық мәтінін шифрлаған кезде C шифрланған мәтін алынатынын білдіретін $\boldsymbol{C = E_k(P)}$ формуласын пайдаланатын боламыз. Сәйкесінше, $\boldsymbol{P = D_k(C)}$ C шифрланған мәтінді ашық мәтінді қалпына келтіру үшін кері шифрлау дегенді білдіреді. Осы екі формуладан:

$$\boldsymbol{D_k(E_k(P)) = P}$$

шығады. Мұндай жазба шартты белгісі E және *D* қарапайым математикалық функция деп жорамалдайды. Олар, шынында да қарапайым функция. Мұндағы жалғыз оқулық, бұл екі функцияның да екі параметрі бар, оның бірін біз аргумент (кілт) түрінде емес, оны мәлімдемеден ажырату үшін төменгі индекс түрінде жаздық.

Криптографияның негізгі ережесі – криптосараптаушыға (шифрды сындырушы) шифрлау тәсілі белгілі деп болжау. Басқа сөзбен айтқанда, қылмыскер *8.1-суреттегі* E шифрлау және *D* кері шифрлау тәсілінің қалай жұмыс істейтінін біледі. Жаңа тәсілді құрастыру, тестілеу және енгізуге үлкен күш пен шығын қажеттілігіне байланысты, ескі тәсіл көпшілікке белгілі болып қалған жағдайда шифрлау алгоритмін құпия сақтау тиімсіз. Ал тәсілді көпшілікке белгілі болған кезде құпия деп болжау одан да көп шығын әкеледі.

Бұл жерде шифрлау кілті көмекке келеді. Кілт, шифрлау нәтижесінің көптеген нұсқаларының ішіндегі анықтауышы қысқа жолдан тұрады. Бірнеше жылда бір өзгеретін шифрлау тәсіліне қарағанда кілтті қажеттілігінше жиі өзгертуге болады. Осылайша, біздің базалық моделіміз тұрақты және көпшілікке белгілі тәсіл болып келеді, мұнда параметр ретінде құпия және жеңіл өзгертілетін кілт пайдаланылады. Мұндағы негізгі ой криптосараптаушыға тәсіл белгілі, ал құпиялық іргетасы бірегей кілт деп болжаймыз, бұл **Керкгоф қағидасы (Kerckhoff's principle)** деп аталады. Оны алғаш рет 1883 жылы, фломанд әскери криптографы Аугуст Керкгоф (Auguste Kerckhoff, 1883) айтқан болатын. Сонымен,

Керкгоф қағидасы былай дейді:
Шифрлау алгоритмдері жалпыға белгілі, тек кілт құпия.

Алгоритмнің құпиялығына көп көңіл бөлудің қажеті жоқ. Саудада **түсініксіздік есебіндегі қауіпсіздік (security by obscurity)** деп аталатын алгоритмді құпия сақтап қалу талпынысы сәтсіз аяқталады. Сонымен бірге, өз алгоритмін жария етіп құрастырушы жаңа жүйені бұзғысы келетін көптеген

ғылым-криптосарапшылардың тегін кеңесін алады және өзінің білімін, ақылдылығын көрсетеді. Егер алгоритм жарияланғаннан кейін ұзақ уақыт аралығында оны ешкім бұза алмаса, демек бұл алгоритмнің берік болғаны.

Іс жүзінде тек кілт құпия сақталатын болғандықтан, негізгі мәселе оның ұзындығы. Қарапайым кодтық құлыпты қарастырайық. Оның негізгі жұмыс қағидасы – сіз тізбек бойынша ондық сандарды енгізесіз. Бұны барлығы біледі, бірақ кілт құпия сақталады. Ұзындығы екі саннан тұратын кілттің 100 нұсқасы бар. Ұзындығы үш саннан тұратын кілттің 1000 нұсқасы бар, ал кілттің ұзындығы алты цифр болған жағдайда комбинациялар саны миллионға жетеді. Кілт ұзын болған сайын шифр сындырушының **еңбек шығын көрсеткіші (work factor)** жоғары болады. Кілт ұзарған сайын кілт мәнін қарапайым талдау арқылы жүйені бұзуға жұмсалатын еңбек шығын көрсеткіші экспоненциалды түрде өседі. Тасымалданатын мәлімдеме құпиялығы қуатты алгоритм (ашық) және ұзын кілтпен қамтамасыз етіледі. Іниіз өзіңіздің электронды поштаңызды оқымасын десеңіз, ұзындығы 64 екілік разрядтан тұратын кілт жеткілікті. Коммерциялық жүйелерді ұзындығы 128 биттен тұратын кілтті пайдаланған орынды. Өз мәтіндеріңізді дамыған елдер үкіметінен қорғау үшін, кем дегенде ұзындығы 256 бит кілт қажет.

Криптосараптаушы тұрғысынан алғанда криптосараптау мәселесі үш негізгі нұсқадан тұрады. Біріншіден, криптосараптаушыда сәйкес ашық мәтінсіз шифрланған мәтіннің қандай да бір бөлігі бар. Бастапқы мән ретінде **тек шифрланған мәтін (ciphertext-only)** болатын есептер әртүрлі газеттердің ребустар бөлігінде жиі басылады. Екіншіден, криптосараптаушыда шифрланған мәтінмен, сәйкес ашық мәтіннің қандай да бір бөлігі болуы мүмкін. Бұл жағдайда біз **белгілі ашық мәтін мәселесімен (known plaintext)** жұмыс істейміз. Соңында, криптосараптаушыда ашық мәтіннің кез келген бөлігін өз қалауынша шифрлау мүмкіндігі болған жағдайда біз мәселенің үшінші нұсқасын, **еркін ашық мәтін мәселесін (chosen plaintext)** аламыз. Егер криптосараптаушыларға «ABCDEFGHJKL-дің шифрланған түрі қандай болады?» сұрақ қоюға болатын болса, газеттегі есептер оңай шешілер еді.

Криптография ісіне жаңадан келушілер жиі егер шифр бірінші типтегі шабуылды (тек шифрланған мәтін) төтеп берсе, ол жеткілікті түрде сенімді деп ойлайды. Мұндай болжам тым аңғырттық. Көптеген жағдайда криптосараптаушы шифрланған мәтіннің жартысын шеше алады. Мысалы, жүйеге кірер алдында көптеген компьютерлердің сұрайтыны – `login:`. Криптосараптаушы бір-біріне сәйкес келетін шифрланған және ашық мәтіннің бірнеше бөлігін алғаннан кейін, оның жұмысы әлдеқайда жеңілдей бастайды. Құпиялықты қамтамасыз ету үшін, тіпті оппонент өз қалауынша ашық мәтіннің бірнеше бөлігін кодтаған жағдайда да шифрдың бұзылмайтындығына кепілдік беретін, криптографтың көрегендігі қажет.

Шифрлау тәсілдері тарихи екі категорияға бөлініп келді: алмастыру тәсілі және орын ауыстыру тәсілі. Біз оларды қазіргі заманғы криптографияға кіріспе ретінде қысқаша қарастырамыз.

8.1.2. Алмастыру тәсілі

Алмастыру тәсіліне (substitution cipher) негізделген шифрларда әр символ немесе симвмолдар тобы басқа символмен немесе символдар тобымен алмастырылады. Көнеден келе жатқан шифрлардың бірі Юлий Цезарьға (Julius Ceasar) таңылып келген **Цезарь шифры (Caesar cipher)**. Бұл шифр циклдық жылжыту арқылы әліппенің барлық символдарын үш орынға жылжытып басқа символдармен алмастырады. Осылайша *a* әрпі *D*-ға, *b* әрпі *E*-ге, *c* әрпі *F*-ке, ... , *z* әрпі *C*-ға алмастырылады. Мысалы, *attack* сөзі *DWWDFN* сөзіне алмастырылады. Біздің мысалдарымызда ашық мәтін кіші символдармен, ал шифрланған мәтін – бас әріптермен белгіленетін болады.

Цезарь шифрының кейбір жалпылама түрлері әліппені үш символға емес еркін *k* символға жылжыту болып келеді. Бұл жағдайда *k* әліппені циклдық жылжытатын жалпы тәсіл кілті болып саналады. Цезарь шифры өз уақытында Помпен тұрғындарын алдаған болар, алайда одан кейін басқа ешкімді алдай алған емес.

Келесі жақсарту – ашық мәтінде кездесетін әр символға басқа символды сәйкестендіру. Мысалы,

ашық мәтін: a b c d e f g h i j k l m n o p q r s t u v w x y z
шифрланған мәтін: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Мұндай жүйе **дара әліппелік алмастыру шифры** (**monoalphabetic substitution cipher)** деп аталады, бұл шифр кілті толық әліппеге сәйкес келетін 26-символдық жол болып келеді. Біздің мысалымыздағы *attack* сөзі *QZZQEA* болады.

Бір қарағанда мұндай жүйе сенімді болып көрінеді, тіпті егер криптосараптаушыға жалпы жүйе белгілі болса да, мүмкін деген кілттің қайсысын қолдануды білмейді. Бұл жағдайда Цезарь шифрының орына қарапайым теру тәсілін қолдану күмәнді. Тіпті, бір нұсқаны тексеруге 1 нс уақыт жұмсағанның өзінде, барлық кілтті тексеріп шығу үшін бір мезгілде жұмыс істейтін миллион компьютер чиптеріне 10 000 жыл қажет болады.

Осылай бола тұрса да шифрланған мәтіннің кішкене бөлігі бар болған жағдайда мұндай шифр жеңіл бұзылады. Шабуылдау үшін табиғи тілдердің статистикалық сипаттамалары қолданылуы мүмкін. Мысалы, ағылшын тілінде е ең жиі кездесетін әріп. Бұл әріптен кейін жиі *t, o, a, n, i* және т.б. әріптер жүреді. Екі символдан тұратын жиі кездесетін комбинациялар **(биграммалар – digrams)** *th, in, er, re* және *an*. Үш символдан тұратын жиі кездесетін комбинациялар немесе **триграммалар (trigrams)** – *the, ing, and* және *ion*.

Дара әліппелік шифрды бұзбақ болған криптосараптаушы алдымен шифрланған мәтінде әліппедегі барлық әріптердің кездесу жиілігін санап шығады. Сонан кейін ол ең жиі кездескен әріпті *e* әріпімен алмастыруға тырысады, келесі жиі кездескенін – *t* әрпімен. Сонан кейін ол триграммаларға қарап *tXe*

ұқсас тіркесті табуға тырысып, тапқаннан кейін *X* - бұл *h* әрпі деп болжайды. Сәйкесінше, егер *thYt* тіркесі жиі кездессе, онда *Y – a* әрпі болуы мүмкін. Осындай ақпаратқа қолжеткізген криптосараптаушы жиі кездесетін *aZW* триграммасын іздеуі мүмкін, бұл тіркес *and* деп болжанады. Осылайша әріптерді, биграммаларды, триграммаларды іздеуді жалғастырып және әріптер тіркесі ықтималдығын біле отырып, криптосараптаушы бастапқы мәтінді қалпына келтіре алады.

Келесі бір тәсіл бірден сөзді немесе сөз тіркесін толығымен анықтауға негізделген. Мысалы, бухгалтерлік фирмадан алынған келесі шифрланған мәтінді қарастырайық (бес символдан тұратын блоктарға бөлінген).

CTBMN	BYCTC	BTJDS	QXBNS	GSTJC	BTSWX	CTQTZ	CQVUJ
QJSGS	TJQZZ	MNQJS	VLNSX	VSZJU	JDSTS	JQUUS	JUBXJ
DSKSU	JSNTK	BGAQJ	ZBGYQ	TLCTZ	BNYBN	QJSW	

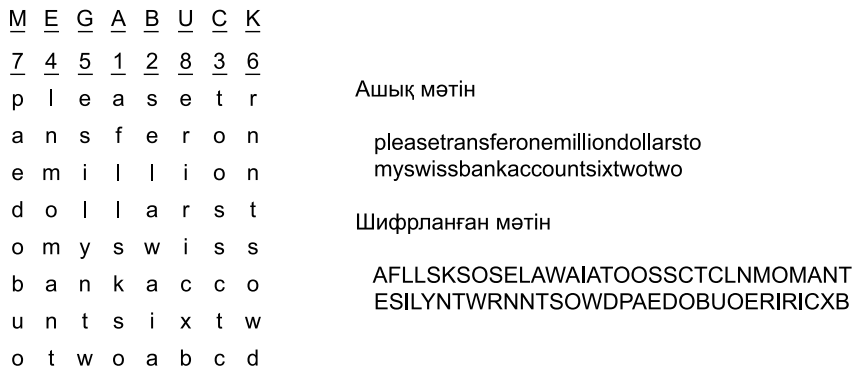
Бухгалтерлік фирма мәлімдемесінде *financial (қаржылық)* сөзі кездесуі тиіс. Осы фактты пайдаланып, сөзде *i* әрпі арасына төрт әріп салып, екі рет кездесетінін ескеріп, бір-бірінен белгілі бір арақашықтықта орналасқан қайталанатын символдарды іздейміз. Нәтижесінде, біз мәтінде осындай 12 орынды анықтаймыз, олар мына позицияларда орналасқан: 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 және 82. Алайда, тек екі жағдайда 31 және 42 позицияларда келесі символ (ашық мәтіндегі *n* әрпіне сәйкес) сәйкес орында қайталанады. Осы екі нұсқаның ішінде a символы тек 31 позиция үшін дұрыс орналасқан. Осылайша, бізге *financial* сөзінің 30 позициядан басталатыны белгілі болды. Ағылшын тілінің лингвистикалық ерекшеліктерін пайдаланып, сөзді толығымен анықтауды ары қарай жалғастыруға болады.

8.1.3. Орын ауыстыру тәсілі

Алмастыру тәсіліне негізделген шифрлар символдар ретін сақтайды, бірақ оларды алмастырады. **Орын ауыстыру тәсілін (transposition ciphers)** қолданатын шифрлар символдар ретін өзгертеді, бірақ символдардың өзін өзгертпейді. *8.2-суретте* бағаналық орын ауыстыруы бар, қарапайым орын ауыстыру шифры көрсетілген. Шифрдың кілті – қайталанатын әріптері жоқ сөз немесе сөз тіркесі. Біздің жағдайда MEGABUCK сөзі кілт қызметін атқарады. Кілттің мақсаты – бағаналарды нөмірлеу. Әліппе басына ең жақын орналасқан әріп астында орналасқан бағана бірінші бағана және т.с.с. ашық мәтін көлденеңінен жолдарда жазылады. Шифрланған мәтін ең кіші кілттік әріптен басталып, бағаналар бойынша оқылады.

Орын ауыстыру шифрын бұзу үшін криптосараптаушы, алдымен осы орын ауыстыру тәсілімен жұмыс істеп отырғанын түсінуі тиіс. Егер *E, T, A, O, I, N* және т.б. символдардың қайталану жиілігіне қарайтын болсақ, олардың әдеттегі ашық мәтін жиілігіне сәйкес екенін көруге болады. Олай болатын болса,

бұл шифрдың орын ауыстыру екені анық, себебі мұндай шифрда әріптер басқа символдармен алмастырылмайды.



8.2-сурет. Орын ауыстыру шифры

Содан кейін бағаналар санын анықтау керек. Көптеген жағдайда мәлімдеме мәнмәтінде сөзді немесе сөз тіркесін анықтауға болады. Мысалы, криптосараптаушы мәлімдемеде *milliondollars* сөзі болу керек деп жорамалдайды. Енді, осы сөз ашық мәтінде бар болуының нәтижесінде, шифрланған мәтінде *MO, IL, LL, LA, IR* және *OS* биграммалары кездеседі. Мұнда *O* символы *M* символынан кейін орналасқан (яғни, қатарынан 4 бағанада орналасқан), себебі олар болжамалы сөз тіркесінде кілт ұзындығына тең арақашықтықта орналасқан. Егер ұзындығы 7-ге тең кілт қолданылса, онда жоғарыда айтылған биграммалардың орнына келесідей биграммалар кездесер еді: *MD, IO, LL, LL, IA, OR* және *NS.* Осылайша, кілттің ұзындығына байланысты шифрланған мәтінде әртүрлі биграммалар жиынтығы құрастырылады. Әртүрлі нұсқаларды сұрыптап, криптосараптаушы жиі кілт ұзындығын жеңіл анықтай алады.

Енді тек бағаналар ретін анықтау қалды. Егер *k* бағаналар саны үлкен болмаса, онда барлық *k(k-1)* мүмкін деген көршілес қос бағана комбинацияларын, кездескен биграммалар жиілігін ағылшын тілінің статистикалық сипаттамаларымен салыстыра отырып қарап шығуға болады. Ең жақсы сәйкес келген қос бағана дұрыс жайғасу деп саналады. Содан кейін қалған бағаналар кезегімен табылған қос бағанамен тексеріледі. Статистикамен ең үлкен сәйкестік көрсеткен биграммалар және триграммалар дұрыс деп жорамалданады. Үдеріс барлық бағаналар реті қалпына келтірілгенше қайталанады. Мәтінді осы сатының өзінде тануға болады деген болжам бар (мысалы, егер біз *million* сөзінің орнына *milloin* сөзін көрсек, онда бір жерде қате кеткені белгілі болады).

Кейбір орын ауыстыру шифрлары кірісте бекітілген ұзындықтағы блок алып шығыста бекітілген ұзындықтағы блокты береді. Мұндай шифрлар толығымен символдардың шығыс блогқа келіп түсетін реті көрсетілген тізіммен анықталады. Мысалы, *8.2-суреттегі* шифрды 64-символдық блок

ретінде қарастыруға болады. Оның кірісін 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, ..., 62 сандар тізбегімен сипатталар еді. Басқа сөзбен айтқанда, төртінші кіріс әрпі a, шығыста бірінші болады, ал содан кейін он екінші – f және т.с.с.

8.1.4. Бір реттік блокнот

Бұзылмайтын шифрды құрастыру іс жүзінде оңай. Оның тәсілі көп жылдардан бері белгілі. Кілт ретінде ұзындығы бастапқы мәтінмен сәйкес келетін еркін биттік жол таңдалады. Ашық мәтінде екілік разрядтар тізбегіне түрлендіріледі, мысалы, стандартты ASCII кодының көмегімен. Соңында, бұл екі жол әр разряды модуль-2 (XOR операциясы) бойынша қосылады. Нәтижесінде, алынған шифрланған мәтінді бұзу мүмкін емес, себебі үлкен бөліктегі кез келген әріп, биграмма немесе триграмма ықтималдығы бірдей болады. Бұл тәсіл **бір реттік блокнот (one-time pad)** деп аталады, теориялық тұрғыдан ол сындырушының осы кездегі немесе болашақтағы есептеу қуаттылығына қарамастан, кез келген шабуылдан қорғаныс болып саналады. Бұл мүмкін емес факт ақпараттар теориясы көмегімен түсіндіріледі: сындырушы үшін шифрланған мәлімдемеде ешқандай ақпарат жоқ, себебі кез келген ашық мәтін ықтималдығы тең кандидат бола алады.

Бір реттік блокнотты қолдану мысалы *8.3-суретте* көрсетілген. Бастапқыда “I love you” сөз тіркесі 7-биттік ASCII-кодқа түрлендіріледі. Содан кейін бір реттік тізбек таңдалады, Тізбек 1. Содан кейін осы тізбек мәлімдемемен модуль-2 бойынша қосылады. Нәтижесінде, қандай да бір шифр алынады. Криптосараптаушыға оны бұзу үшін мүмкін деген барлық бір реттік тізбекті тексеріп, ашық мәтіннің қандай болатынын сараптау керек болады. Мысалы, егер мәлімдемені Тізбек 2 көмегімен (*8.3-суреттті* қараңыз) қалпына келтірмек болса, “Elvis lives” (Элвис тірі) мәтіні алынады. Осы тұжырымдаманың дәлдігін талқылау бұл кітаптың шеңберінен тыс шығып кетеді, алайда біз бастапқы мәтіннің қайта қалпына келмегенін көрдік, бірақ ағылшын грамматикасы тұрғысынан дұрыс сөз тіркесі алынды. Іс жүзінде, 11 символдан тұратын кез келген тізбекті генерациялау үшін ASCII кодында бір реттік блокнот табылады. Шифрланған мәтінде ешқандай ақпарат жоқ дегенде біз осыны айттық: одан ұзындығы сәйкес келетін кез келген мәтінді алуға болады.

Бір реттік кілттер өте қуатты құрал болып келеді, алайда олардың бірнеше кемшіліктері де бар. Біріншіден, мұндай ұзын кілтті есте сақтау мүмкін емес, сондықтан жөнелтуші де, қабылдаушы да кілттің жазба көшірмесін өзімен бірге алып жүруі тиіс. Егер бұл көшірменің бірін қарсылас қолға түсіретіндей қауіп бар болса, жазба көшірмені сақтау қажет емес. Одан басқа, деректердің толық көлемі қолжетімді кілт көлемімен шектеулі. Егер тыңшының жолы болып, ол үлкен ақпаратты қолға түсірсе, ол бұл мәлімдемені орталыққа жібере алмайды, себебі кілттің ұзындығы жеткіліксіз. Тағы бір мәселе, тәсілдің жоғалған немесе кіріктірілген символдарға сезімталдығы. Егер жөнелтуші немесе қабылдаушы синхрондауды бұзса, онда деректер осы жерден бастап бұзылады.

1 Мәлімдеме:
1001001 0100001 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110

1 тізбек:
1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011

Шифрланған мәтін:
0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

2 тізбек:
1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110

2 ашық мәлімдеме:
1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

8.3-сурет. Мәлімдемені шифрлау үшін бір реттік блокнотты пайдалану және басқа кілттік тізбекті қойып, шифрланған мәтіннен еркін ашық мәтінді алу мүмкіндігі

Компьютерлердің пайда болуымен бір реттік блокнот тәсілін іс жүзінде пайдалану мүмкін. Кілтті бірнеше гигабайт ақпарат жазылған арнайы DVD дискте сақтауға болады. Егер оны басында бірнеше минут фильм жазылған бейнекомпакт-диск қорабында тасымалданса, тіпті ерекше күмән тудырмайды да. Әрине, деректер тасымалдау жылдамдығы гигабайтпен саналатын желілерді компакт-дискті әрбір 30 секунд сайын жаңартып отыру қажеттігі тез шаршатады. Демек, бір реттік кілті бар DVD дискті құпия мәлімдемені қабылдаушыға қолма-қол беру керек, ал бұл бір реттік блокнот тәсілін іс жүзінде қолдану ықтималдығын күрт төмендетеді.

Кванттық криптография

Бір реттік тізбекті желі арқылы тасымалдау мәселесінің шешімінің бұл саладан алшақ ғылымнан – кванттық механикадан келгені қызықты. Бұл көп үміттендіретін тәжірибелік аумақ. Егер бұл тәсілді жақсарту мүмкін болса, онда криптография есептерінің барлығын бір реттік тізбек көмегімен шешуге болады, себебі бұл ақпаратты қорғаудың ең сенімді тәсілі. Төменде біз **кванттық криптография** деп аталатын технологияны қысқаша сипаттаймыз. Жеке алғанда біз **BB84** хаттамасын қарастырамыз. Хаттама сипаттамасы алғаш рет жарияланған кезде, оны құрастырушылар құрметіне осылай аталған болатын (Bennet және Brassard, 1984).

Айталық, Алиса атты тұтынушы басқа Боб деген тұтынушыға бір реттік тізбекті жөнелткісі келеді. Алиса және Боб **принципалдар (principals)** деп аталады, олар біздің оқиғамыздың негізгі кейіпкерлері. Мысалы, Боб банкир, ал Алиса онымен іскерлік қарым-қатынасқа түскісі келетін тұлға. Көптеген жылдар бұрын Рон Ривест алғаш рет осы екі атты пайдаланғаннан бері іс жүзінде криптографияға қатысты материалдардың барлығында принципиалдарды белгілеу үшін нақты «Алиса» және «Боб» қолданылады (Rivest және басқалар,

1978). Криптографтар әртүрлі дәстүрді жақсы көреді. Егер біз бұл жерде Алиса және Бобтың орнына Андрей және Белланың қарым-қатынасы жайлы жазатын болсақ бұл тараудың бірде-бір сөзіне ешкім сенбес еді (автордың криптографияға қатысы жоқ адам екені белгілі болар еді). Сондықтан біздің кітабымыздың да кейіпкерлері Алиса және Боб болсын.

Сонымен, егер Алиса және Боб бір реттік тізбекті орната алса, онда олардың әңгімелері толығымен құпия болып қалады. Мәселе мынада: олар бірбіріне DVD бермей құпия кілтпен қайша алмасады? Біз екі тұтынушы да бірбіріне желілік импульстарды жөнелтіп, қабылдай алатын бір оптыталшықты кабельдің екі басында орналасқан деп жорамалдаймыз. Осылай бола тұрса да, осы кабель жолында активті тыңдаушы құрылғыны орналастыра алған ержүрек тыңшы Труди табылды делік. Ол екі бағытта жүріп жатқан сигналдарды да оқи алады делік. Сонымен бірге, ол екі бағытта да жалған мәлімдемелер жөнелте алады. Алиса және Боб үшін жағдай тығырыққа тірелгендей. Міне, осы жерде кванттық криптография көмекке келеді және біздің кейіпкерлерімізде үміт оты пайда болғанын көреміз.

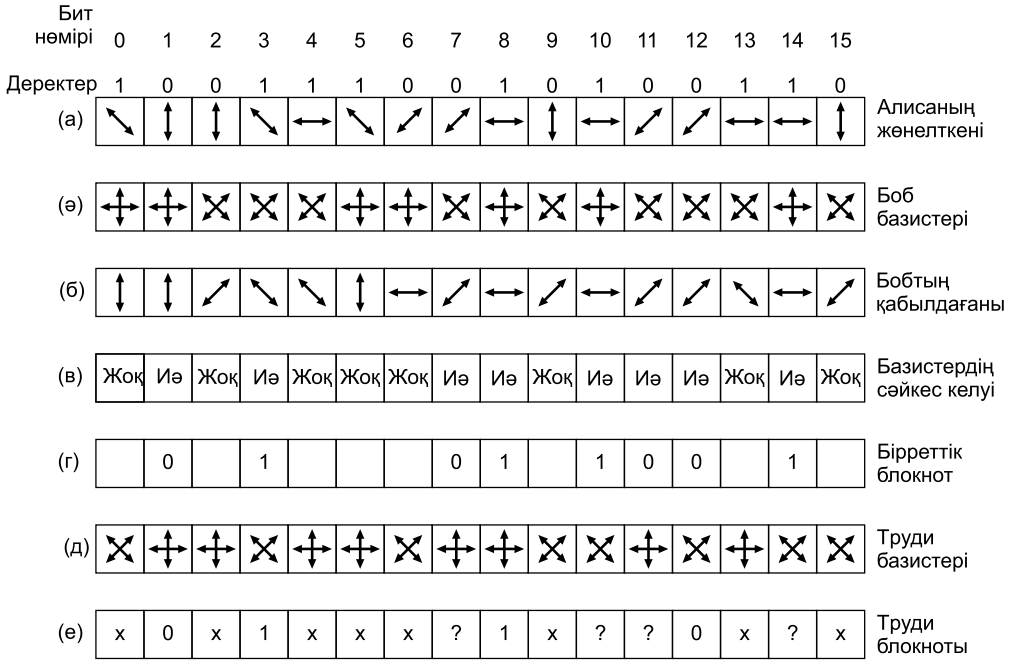
Кванттық криптография жарықтың **фотон** деп аталатын және біршама ерекше қасиеттері бар кіші бөліктермен берілуіне негізделген. Сонымен бірге жарықты поляризациялық фильтр арқылы өткізіп, оның поляризациялануына қол жеткізуге болады. Бұл факт алдымен күннен қорғайтын көзілдірік тағатындарға және фотографтарға белгілі. Жарық сәулесі (яғни, фотон ағыны) осындай фильтр арқылы өтіп, фильтр осі бағытында полярланады (мысалы, тігінен). Егер бұдан кейін сәулені екінші фильтр арқылы өткізсек, онда шығыстағы жарық күші фильтр осьтерінің арасындағы бұрыштың косинусының квадратына тең болады. Егер осьтерді перпендикуляр орналастырса фотондар фильтр арқылы өте алмайды. Осьтердің кеңістіктегі абсолютті бағытталуының мәні жоқ – тек олардың өзара орналасуы маңызды.

Бір реттік блокнотты генерациялау үшін Алисаға поляризациялық фильтрдің екі жиынтығы қажет болады. Бірінші жиынтық тік және көлденең фильтрден тұрады. Ол **түзу сызықты базис** деп аталады. Базис – қарапайым координаталар жүйесі. Екінші фильтрлер жиынтығын біріншісінен 45^{o} бұрылғандығымен ерекшеленеді, демек оны төменгі сол жақ бұрыштан оң жақ жоғарғы бұрышқа бағытталған, ал екіншісі сол жақ жоғарғы бұрыштан оң жақ төменгі бұрышқа бағытталған сызық ретінде елестетуге болады. Бұл **диагональдік базис** деп аталады. Сонымен, Алисаның екі фильтрлер жиынтығы бар және ол оның кез келгенін жарық сәулесінің жолына орналастыра алады. Бірақ төрт әйнек-поляризатор – ғажайып елдегі Алиса ертегісіне қарағанда түк емес. Іс жүзінде, мүмкін деген төрт поляризациясы жоғары жылдамдықтағы электронды әдіспен өзгертілетін кристалл бар. Бобтың да дәл Алисадағыдай құрылғысы бар. Алиса және Бобта екі базистен болуы кванттық криптографияда маңызды рөл атқарады.

Алиса әр базисте бір бағытын нөлмен белгілейді, ал келесісін сәйкесінше бірмен. Төменде көрсетілген мысалда, біз Алиса тік бағытты нөл ретінде, ал

көлденең бағытты – бірлік ретінде таңдап алды деп болжаймыз. Бұған тәуелсіз, «төменгі сол – жоғарғы оң» бағытына нөл мәні, ал «жоғарғы сол – төменгі оң» бағытына – 1 мәні беріледі. Бұл ақпарат Бобқа ашық мәтін ретінде беріледі.

Енді, Алиса, мысалы, кездейсоқ сандар генераторы көмегімен алынған (бұл генератор өзінше күрделі құрал) бір реттік блокнотты алып, Бобқа жөнелтеді. Тасымалдау разрядтармен жүргізіледі, әр бит үшін екі базистің бірі кездейсоқ таңдалынады. Битті тасымалдау үшін фотондық зеңбірек осы бит үшін таңдап алынған базистен өте алатындай бір фотон шығарады. Мысалы, базистер келесідей ретпен таңдалына алады: диагоналдық, түзу сызықтық, түзу сызықтық, диагоналдық, түзу сызықты және т.с.с. осы базистер көмегімен 1001110010100110 тізбегінен тұратын бір реттік блокнотты тасымалдау үшін 8.4 а-суретінде көрсетілген фотондар жөнелтіледі. Осы бір реттік блокнот және оған сәйкес базистер тізбегі үшін әр битке қолданылатын поляризация жалғыз рет анықталады. Бір уақыт бірлігінде бір фотонмен жөнелтілетін биттер **кубиттер (qubits)** деп аталады.



8.4-сурет. Кванттық криптография мысалы

Боб қандай базисті пайдалану керектігін білмейді, сондықтан ол *8.4 ә-суретінде* көрсетілгендей, әрбір келген фотон үшін базисті кездейсоқ ретпен қолданады. Егер осы фотон үшін базис дұрыс таңдалса, Боб дұрыс битті алады. Кері жағдайда биттің мәні кездейсоқ болады, себебі фотон өзінің жеке поляризациясына қатысты 45^{o} бұрылған поляризатор арқылы өтеді және тең

ықтималдықпен бірге немесе нөлге сәйкес бағытқа түседі. Фотондардың бұл қасиеті бүкіл кванттық механикада негізгі қасиеттердің бірі болып саналады. Осылайша, кейбір биттер дұрыс, кейбіреуі дұрыс емес алынады, бірақ Боб олардың қайсысының дұрыс, ал қайсысының дұрыс еместігін анықтай алмайды. Боб алатын ақпарат *8.4 б-суретінде* көрсетілген.

Базистердің қайсысының дұрыс, ал қайсысының дұрыс емес ұсынылғанын Боб қалай біледі? Ол үшін Боб Алисаға ашық мәтінмен әр битті қабылдау кезінде қандай базистер қолданғанын хабарлайды, ал Алиса жауап ретінде (ашық мәтінмен) Бобтың қандай базистерді дұрыс қолданғанын хабарлайды. Бұл *8.4 в-суретінде* көрсетілген. Осы ақпаратты біле отырып Алиса және Боб биттік жолды *8.4 г-суретінде* көрсетілгендей дұрыс болжай алады. Орта есеппен бұл жолдың ұзындығы бастапқы жолдың толық ұзындығының жартысына тең болады, алайда бұл жағдай екі жаққа да белгілі болғандықтан олар жолдың дұрыс болжамын бір реттік блокнот ретінде пайдалана алады. Алиса тек ұзындығы бір реттік блокноттың екі еселенген ұзындығынан сәл асатын биттік жолды жөнелту керек. Мәлесе шешілді.

Тоқтай тұрыңыздар, біз Труди жайлы ұмытып кеттік қой! Ол Алисаның не айтқысы келгенін білгісі келеді, сондықтан ол тасымалдау торабына өз детектр және тасымалдағышын ендірді деп жорамалдадық. Өкінішке орай (Труди үшін), ол да әр фотонды қандай базис арқылы өткізу керектігін білмейді. Ол да базисті Боб тәрізді кездейсоқ таңдай алады. Трудидің оны қалай орындағаны *8.4 д-суретінде* көрсетілген. Боб Алисаға ашық мәтінмен қандай базистерді пайдаланғанын, ал Алиса қайсының дұрыс екенін хабарлаған кезде, Труди де Боб тәрізді қандай биттерді тапқанын, қандайын таппағанын біледі. Суретте көрсетілгендей Труди базистері 0, 1, 2, 3, 4, 6, 8, 12 және 13 позицияларда Алисанықімен сәйкес келді. Алайда, Алисаның жауабынан (*8.4 в-суретін* қараңыз) ол бір реттік блокнотқа тек 1, 3, 7, 8, 10, 11, 12 және 14 биттердің кіретінін біледі. Осы биттердің төртеуі (1, 3, 8 және 12) дұрыс табылды. Труди оларды есте сақтайды. Ал қалған биттер (7, 10, 11 және 14) табылмады және олардың мәндері Труди үшін белгісіз болып қалады. Осылайша, Бобқа 01011001 басталатын (*8.4 г-суретін* қараңыз) бір реттік блокнот белгілі, ал Трудиде қалғаны – үзінділер 01?1??0? (*8.4 е-суретін* қараңыз).

Әрине, Алиса мен Боб өздерінің бір реттік блокноттың бөлігін Трудидің алып қалғысы келетінін біледі, сондықтан ол қол жеткізе алатын ақпарат көлемін азайтуға тырысады. Ол үшін олар тізбекпен біраз әрекеттер орындауы мүмкін. Мысалы, бір реттік блокнотты 1024 биттен тұратын блоктарға бөліп, әр блокты квадраттауы мүмкін, осылайша ұзындығы 2048 бит сандар алу. Содан кейін 2048-биттік сандардың конкотенциясын бір реттік блокнот ретінде пайдалану. Биттік жолдың тек бір бөлігін алып Труди бұл түрлендіруді ешуақытта орындай алмайды. Труди алатын ақпараттың үлесін азайтатын, бастапқы бір реттік блокнотпен орындалатын әрекеттер **құпиялықты күшейту (privacy amplification)** деп аталады. Іс жүзінде блокты квадраттаудың орнына, әр бит әрбір кіріс битке тәуелді күрделі түрлендірулер қолданылады.

Әзіз Труди. Оның бір реттік блокноттың қандай екені жайлы ешбір болжамы жоқ, сонымен бірге өзінің тыңдап отырғаны жайлы бәрінің хабардар екенін де сезеді. Боб Алисамен сөйлесіп отырмын деп ойлау үшін ол әр қабылданған битті Бобқа жөнелтуге тиіс. Өкінішке орай, Труди тек әр қабылданған квантобитті өзі қабылдаған поляризацияны пайдаланып жөнелте алады. Бұл жағдайда поляризацияның шамамен жартысы дұрыс болмайды, бұл Бобтың бір реттік блокнотындағы қателіктерге әкеледі.

Соңында, Алиса деректерді жөнелте бастаған кезде, ол оны қателіктерді алдын ала түзетуі бар күрделі кодты пайдаланып шифрлайды. Боб тұрғысынан, бір реттік тізбектің бір битіндегі қателік – бұл деректер тасымалдаудың бір битінде кеткен қателік. Кез келген жағдайда, нәтиже биттің дұрыс емес мәнін алудан тұрады. Қателікті алдын ала түзеу арқылы жоғалған ақпаратты қалпына келтіріп қана қоймай, сонымен бірге қателіктер санын да есептеуге болады. Егер ол күтілгеннен әлдеқайда көп болса (құрылғыда орын алған қателік ықтималдығымен байланысты), онда торапты Трудидің тыңдап отырғаны белгілі болады және Боб қажет шараларды қолдана алады (мысалы, Алисаның радиоарна ауысуын сұрауы, полиция шақыруы және т.б.). егер Труди фотонды көшіріп, өз көшірмесін өңдеп, ал бастапқы фотонды Бобқа өзгермеген түрде жөнелтсе, онда оның байқаусыз қалуға мүмкіндігі бар еді, алайда бүгінгі күнде фотонды клондау мүмкіндігі жоқ. Тіпті, Труди фотонның көшірмесін алған күнде де, бұл кванттық криптографияның маңызын ешбір төмендетпейді.

Кванттық криптография ұзындығы 60 км оптыталшықты арна бар жерде жұмыс істей алады, бірақ өте қымбат құрылғы қажет. Осыған қарамастан негізгі ойдың өзі көп үміт береді. Кванттық криптография жайлы толығырақ Mullins (2002) кітабынан оқуға болады.

8.1.5. Криптографияның екі іргелі қағидасы

Келесі парақтарда біз көптеген әртүрлі криптографиялық жүйелерді қарастырсақ та, олардың барлығының негізінде өте маңызды екі қағида жатыр. Оларға ерекше назар аударыңыздар – бұл қағидаларды бұзу өте қауіпті.

Молшылық

Бірінші қағида былай дейді: барлық шифрланған мәлімдемелердің белгілі бір дәрежедегі молшылығы болуы тиіс, яғни мәлімдемені түсінуге қажет емес ақпарат болуы керек. Мысал арқылы түсіндірейік. Өзіңізге пошталық тапсырыс арқылы 60 000 атаудан тұратын тауарларды саудалайтын «Couch Potato» (Үй күшік) компаниясын елестетіңіз. Ресурстарды үнемді пайдаланғанына қуанған «Couch Potato» (Үй күшік) компаниясының программистері тапсырыс бланкі 16 байттық клиент атынан, содан кейін 3 байттық тауар өрісі (1 байт

тауар санын белгілеу және 2 байт тауар идентификаторына) тұратын болады деп шешті. Соңғы үш байтты тек «Couch Potato» (Үй күшік) компаниясының клиентіне белгілі үлкен кілт көмегімен кодтау ұйғарылды.

Бір қарағанда мұндай схема сенімді болып көрінеді, жеке алғанда пассивті зиянкес мәлімдемені қайта қалпына келтіре алмайды. Өкінішке орай, бұл схеманы толығымен құнсыздандыратын сынды кемшілігі бар. Айталық, жақын арада жұмыстан қуылып шыққан қызметкер жұмыстан шығарып жібергені үшін «Couch Potato» (Үй күшік) компаниясынан кек алғысы келеді. Кетер алдында ол компания клиенттері тізімінің бір бөлігін ала кетті делік. Бір түн ішінде ол нағыз клиенттер аты көрсетілген жалған тапсырыс құрастырып шығады. Кілттер тізімі жоқ болғандықтан, ол соңғы үш байтқа кездейсоқ сандар жазып, «Couch Potato» (Үй күшік) компаниясына жүздеген тапсырыстар жөнелтеді.

Осы мәлімдемелер «Couch Potato» (Үй күшік) компаниясының компьютеріне келіп жеткен кезде клиент аты бойынша мәлімдемені қайта қалпына келтіру кілті табылады. Өкінішке орай, «Couch Potato» (Үй күшік) компаниясы үшін барлық 3-байттық мәлімдемелер сенімді түрде қабылданады, сондықтан компьютер тапсырыстарды тауар жеткізу үшін басып шығара бастайды. Клиент 837 балалар алтыбақанының орындығын немесе 540 құмсалғышқа тапсырыс берсе ерсі көрінуі мүмкін, бірақ кім біледі, ол балалар ойын алаңын жасақтаумен айналысқан шығар. Осылайша активті зиянкес (жұмыстан қуылып шыққан қызметкер) тіпті өз компьютерінен жіберіп отырған мәлімдеменің мәніне мән бермей, көптеген келеңсіз жағдайлар туындатуы мүмкін.

Бұл мәселені әр мәлімдемеге молшылық ақпаратты қосу арқылы шешуге болады. Мысалы, егер үш шифрланатын байтқа тағы тоғыз, айталық, нөлдік, қоссақ, онда жұмыстан қуылып шыққан қызметкер сенімді болып көрінетін мәлімдемелер ағынын құрастыра алмайды. Бұл әңгіменің ғибраты – активті зиянкес кездейсоқ қоқысты нағыз мәлімдеме ретінде ұсынбас үшін барлық мәлімдемелерде жеткілікті сандағы молшылық ақпарат болуы керек.

Алайда молшылық ақпаратты қосу шифрды бұзушы криптосараптаушы жұмысын жеңілдетеді. Айталық, пошталық тапсырыстар бизнесіндегі бәсекелестік тым жоғары және «Couch Potato» (Үй күшік) компаниясының негізгі бәсекелесі «Sofa Tuber» (Жатыпішер) компаниясы «Couch Potato» (Үй күшік) компаниясының айына неше құмсалғыш сататынын білгісі келеді делік. Ол үшін «жатыпішерлер» «үй күшіктер» компаниясының телефон торабына қосылады. Бастапқы 3-байттық нөмірі бар схемада криптосараптау мүмкін емес еді, себебі кілттің мәнін жорамалдап криптосараптаушы өз жорамалының дұрыстығын тексере алмайтын. Мәлімдемелердің барлығы техникалық тұрғыдан дұрыс болатын. Жаңа 12-байттық схема енгізілісімен криптосараптаушы дұрыс мәлімдемені дұрыс емесінен жеңіл ажырата алады. Сонымен:

№1 криптографиялық қағиада:
Мәлімдемеде молшылық деректер болуы тиіс.

Басқа сөзбен айтқанда, мәлімдемені қайта қалпына келтіргенде қабылдаушы оның түпнұсқа екендігін сараптау арқылы тесере алуы тиіс және күрделі емес есептеулерді орындауы мүмкін. Молшылық активті зиянкестердің қабылдаушыны қоқыс жазылған жалған мәлімдемелермен алдауына қарсы тұру үшін қажет. Сонымен бірге, молшылық ақпаратты қосу пассивті зиянкестерге жүйені бұзуды жеңілдетеді, сондықтан бұл жерде белгілі бір дәрежедегі қайшылықтар да бар. Бұдан басқа, молшылық деректер ешуақытта мәлімдеменің басында немесе соңындағы тізбектік нөлдер формасын қабылдамауы тиіс, себебі мұндай мәлімдемені шифрлаған кезде кейбір алгоритмдер криптосараптаушының жұмысын жеңілдететін, күтпеген нәтиже береді. Бұл мақсатта жәй нөлдер тізбегін пайдаланғаннан, циклдық молшылық код көпмүшесін (CRC) пайдаланған ұтымды, себебі қабылдаушы оның дұрыстығын тексере алады және ол криптосараптаушының жұмысын күрделендіреді. Криптографиялық хэш-функцияны пайдалану әлдеқайда ыңғайлы, біз ол жайлы төменде сөз қозғаймыз. Әзірше, бұл CRC-тің әлдеқайда сенімді нұсқасы екенін есте сақтаңыз.

Кванттық криптографияға орала отырып, молшылықтың бұл технологияда қандай рөл атқаратынын айта кеткіміз келеді. Труди фотондарды ұстап алып отырғандықтан, Боб қабылдап алатын бір реттік блокноттың кейбір биттерінің мәні дұрыс болмайды. Бобқа қателікті анықтау үшін кіріс мәлімдемедегі қандай да бір ақпарат молшылығы қажет. Молшылықтың өрескел түрінің бірі ретінде, бір мәлімдемені бірнеше рет қайталауды қарастыруға болады. Егер қабылданған екі көшірме сәйкес келмесе, Боб не арнадағы шудың тым көптігін немесе біреудің деректерді ұстап алып отырғанын біледі. Әрине, барлығын екі реттен жіберіп отыру тым орынсыз. Бұл жағдайда қателікті анықтап, түзету үшін Хэмминг немесе Рид-Соломон кодын пайдаланған дұрыс. Алайда, нақты мәлімдемені жалған мәлімдемеден ажырату үшін қандай да бір ақпарат молшылығы қажет екені түсінікті болуы тиіс. Бұл, әсіресе, активті зиянкес бар болған жағдайда өте маңызды.

Жарамдық мерзімінің шектеулігі

Криптографияның екінші қағидасы – қабылданған мәлімдеменің жаңа, демек, жақын арада жөнелтілгендігін анықтайтын тәсілдің болуы қажет. Бұл шара ұстап алған ескі мәлімдемелерді қайта жөнелтетін активті зиянкестермен күреске бағытталған. Егер мұндай шара қолданбасақ, онда жұмыстан қуылған «Couch Potato» (Үй күшік) компаниясының қызметкері осы компанияның телефон торабына қосылып, ертеде қабылданған нақты мәлімдемелерді қайталап жөнелтуі мүмкін. Сонымен тұжырымдаманы тиянақтайық:

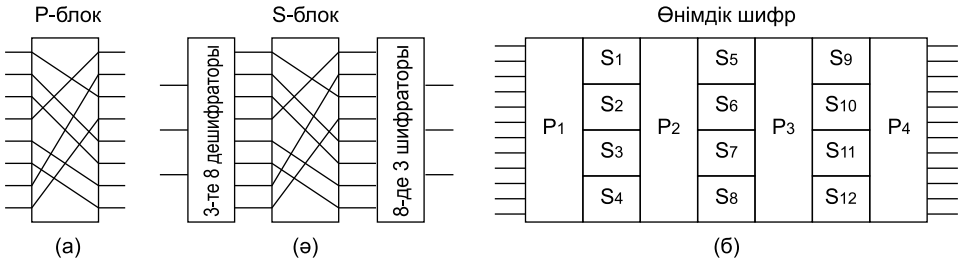
№2 криптографиялық қағида:
Ертеде жөнелтілген мәлімдемелерді қайталап жөнелтумен күресу тәсілдері қажет.

Осындай шаралардың бірі әр мәлімдемеге, айталық, тек 10 с аралығында жарамды, уақыттық штамп қосу. Қабылдаушы алынған мәлімдемелерді көшірмелерін өшіріп, тек 10 с сақтауы тиіс. Жасы 10 с үлкен мәлімдемелер ескірген ретінде еленбейді. Көшірмелерден қорғанудың басқа жолдары төменде қарастырылады.

8.2. СИММЕТРИЯЛЫ КРИПТОГРАФИЯЛЫҚ КІЛТІ БАР АЛГОРИТМДЕР

Қазіргі заманғы криптографияда сол дәстүрлі криптографияның негізгі идеялары қолданылады, яғни орын ауыстыру және алмастыру, бірақ екпіндер басқаша қойылады. Дәстүрлі криптографияда қарапайым алгоритмдер пайдаланылған. Қазір керісінше: мақсат – күрделі, шытырман алгоритм құрастыру, тіпті криптосараптаушының қолына үлкен сандағы шифрланған мәтін түскен күнде де, кілтсіз ол одан ешбір пайда таба алмайды.

Біз қарастыратын шифрлау алгоритмдерінің бірінші класы – **симметриялы кілті бар алгоритмдер (symmetric-key algorithms).** Ол шифрлау және кері шифрлау үшін бір кілт қолданылатындықтан осындай атқа ие болады. Симмертиялы кілті бар алгоритмді пайдалану мысалы *8.1-суретте* көрсетілген. Жеке алғанда, біз кірісте n-биттік ашық мәтінді қабылдап, кілтті пайдаланып, оны *n*-биттік шифрға түрлендіретін, **блоктық шифрды (block ciphers)** егжей-тегжейлі қарастырамыз.



8.5-сурет. Өнімдік шифрдың негізгі блоктары: P-блок (а); S-блок (ә); өнімдік шифр (б)

Криптографиялық алгоритмдер апараттық түрде де (олардың жұмыс жылдамдығын жоғарылатады), программалық түрде де (иілгіштікті жоғарылату үшін) жүзеге асырылады. Біздің талқылауларымыздың үлкен бөлігінің нақты жүзеге асыруға тәуелсіз алгоритмдер мен хаттамалар жайлы болатынына қарамастан, криптографиялық аппараттың құрастырылу қағидаларына тоқтала кеткен пайдалы деп санаймыз. Орын ауыстырулар мен алмастырулар қарапайым электр тізбектерінің көмегімен жүзеге асырылады. *8.5-суретте* сегіз кіріс разрядтарының орын ауыстыру үшін қолданылатын **P-блок (P литерасы**

permutation – орын ауыстыру дегенді білдіреді) деп аталатын құрылғы көрсетілген. Егер кіріс биттерді жоғарыдан төмен нөмірлесек (01234567), онда осы нақты Р-блоктың шығысы келесідей болады: 36071245. Р-блоктың сәйкес ішкі құрылғысының көмегімен (сымдарды дәнекерлеу арқылы) орын ауыстырудың кез келген операциясын жарық жылдамдығымен орындатуға болады, себебі мұнда ешқандай есептеулер жүргізілмейді, сигнал тек кірістен шығысқа беріледі. Бұл шешім Керкгоф қағидасына сәйкес келеді: сындырушы биттердің орнын ауыстыру тәсілінің қолданылатындығын біледі. Алайда, ол орын ауыстыру ретінде негізделген кілтті білмейді.

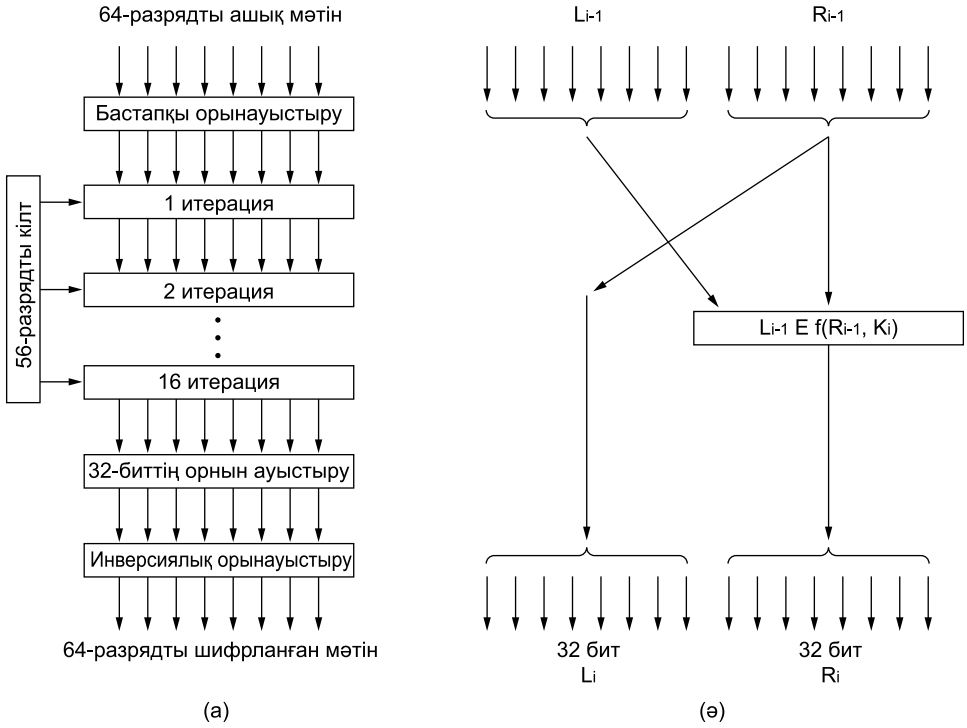
Орын ауыстыруларды *8.5 ә-суретінде* көрсетілгендей **S-блогы** орындайды **(S - substitution – орнына қою, ауыстыру** дегенді білдіреді). Бұл мысалда кіріске 3-биттік ашық мәтін беріледі, ал шығыста 3-биттік шифрланған мәтін пайда болады. Әрбір кіріс сигналы үшін декодердің сегіз шығыс торабының бірінде 1-ді орнату арқылы таңдалып алынады. Қалған тораптардың барлығында 0 орнатылады. Содан кейін, сегіз тораптың барлығы S-блоктың екінші бөлігін құрайтын Р-блок арқылы өтеді. Үшінші сатыда сегіз тораптың бірін кері 3-биттік екілік санға кодтау жүргізіледі. Яғни, 0 - 2 санымен алмастырылады, 1 – 4 санымен және т.с.с. Мұнда да, S-блоктың ішіндегі Р-блок сымдарын қайта жалғап, орын ауыстырудың кез келген нұсқасын жүзеге асыруға болады. Сонымен бірге, мұндай құрылғыны аппаратураға кіріктіріп, жоғары жылдамдықпен жұмыс істеуге болады, себебі шифратор және дешифраторлар тек бір немесе екі вентильдік кідіріс (1 нс кем емес) енгізеді, ал Р-блок ішінде сигналдың таралу уақыты 1 нс кем болмайды.

Бұл элементтердің күшін, *8.5 б-суретінде* көрсетілгендей, осындай құрылғылардан тұратын каскад құрастырған кезде көруге болады. Нәтижесінде алынған құрылғы **өнімдік шифр (product cipher)** деп аталады. Бұл мысалда, бірінші сатыда (P_1) 12 кіріс тораптары орын ауыстырады. Екінші сатыда кіріс үш биттен төрт топқа бөлінеді және олардың әрқайсысында алмастыру операциясы тәуелсіз орындалады (S_1-ден S_4-ке дейін). Бұл тәсіл бірнеше кішкене S-блоктардан үлкен S-блокты құрастырудан тұрады және өте ыңғайлы, себебі кішігірім S-блоктарды аппаратты түрде жүзеге асыру жеңіл (яғни, сегіз разрядты S-блок 256-разрядты қайта кодтау кестесі ретінде жүзеге асырылуы мүмкін), ал үлкен S-блоктарды аппаратты түрде жүзеге асыру ыңғайсыз (мысалы, он екі разрядты S-блок, кем дегенде 212=4096 қиылысқан сымдарды қажет етеді). Бұл тәсіл жалпы шешімнің жеке бір бөлігі болса да, оның қуаты өте жоғары. Өнімдік шифрдың шығысын, үлкен сандағы қосымша сатыларды пайдаланып, кірістің күрделі функциясы етіп құрастыруға болады.

k-биттік тізбекті туындататын k-биттік кіріспен жұмыс істейтін өнімдік шифр кеңінен таралған. Әдетте k мәні 64-тен 256-ға дейінгі аралықта жатады.

8.2.1. DES деректерді шифрлау стандарты

1977 жылдың қаңтар айында Құрама Штаттар үкіметі IBM фирмасы құрастырған өнімдік шифрды құпия емес мәліметтер үшін ресми стандарт ретінде қабылдады. **DES (Data Encryption Standard – деректерді шифрлау стандарты)** деп аталған бұл шифр өндірісте ақпаратты қорғау үшін кеңінен қолданылды. Қазір оның бастапқы нұсқасы сенімді емес, бірақ түрлендірілген нұсқасы әлі де пайдалы. Қазір біз оның жұмыс істеу принципін қарастырамыз.



8.6-сурет. DES деректерді шифрлау стандарты: а – жалпы көрніс;
ә – сатылардың бірінің құрылымы

DES шифрының схемасы *8.6 а-суретінде* көрсетілген. Ашық мәтін 64 биттік блок ретінде шифрланады, нәтижесінде шығыста 64-биттік шифрланған мәтін блоктары алынады. Елу алты разрядты кілтті пайдаланатын алгоритм 19 жеке сатылардан тұрады. Бірінші сатыда ашық мәтіннің 64 разрядын тәуелсіз орын ауыстыру орындалады. Соңғы сатыда кері орын ауыстыру орындалады. Соңғының алдындағы сатыда сол және оң 32-разрядтар орын ауыстырады. Қалған 16 сатының қызметтері ұқсас, бірақ кіріс кілтінің әртүрлі функцияларымен басқарылады. Алгоритм, кері шифрлау сол шифрлаған кілттің көмегімен жүргізілетіндей етіп құрастырылған. Бұл алгоритмнің сим-

метриялы кілт қағидасына сәйкес келуін қамтамасыз етеді. Кері шифрлау кезінде сатылар тек кері ретпен орындалады.

Аралық сатылардың бірінде орындалатын операция *8.6 ә-суретінде* көрсетілген. Әр сатыда кірістегі 32-разрядтан тұратын екі бөліктен шығыстағы 32-разряд құрастырылады. Кірістің оң бөлігі сол бөлікке көшіріледі. Оң жақтағы 32-разряд сол жақ бөліктің, оң жақ бөлік функциясының және $К_i$ осы саты кілтінің модуль-2 бойынша қосындысы болып келеді. Алгоритм күрделігінің барлығы осы функцияда.

Функция тізбек бойынша орындалатын төрт қадамнан тұрады. Алдымен, оң жақтағы 32-разряд бекітілген орын ауыстыру және қайта орындау арқылы 48-разрядты Е санын құрайды. Екінші қадамда Е саны және $К_i$ кілті модуль-2 бойынша қосылады. Содан кейін шығыс 6-рязрядтан сегіз топқа бөлінеді, олардың әрқайсысы тәуелсіз S-блок көмегімен 4-разрядты топтарға түрлендіріледі. Соңында, осы 8×4 разряд Р-блок арқылы өткізіледі.

Он алты сатының әрқайсысында бастапқы кілттің түрлі функциясы пайдаланылады. Алгоритмнің жұмысы басталар алдында кілтке 56-разрядты орын ауыстыру қолданылады. Әр саты алдында кілт 28-разрядтан екі топқа бөлінеді. Әр топ саты нөміріне тәуелді сан разрядына солға бұрылады. Осы операция нәтижесінде тағы бір 56 разрядты орын ауыстыру көмегімен $К_i$ кілті алынады. Әр сатыда 56 разрядты кілттің 48 разряды таңдалып алынады және олардың да орны ауыстырылады.

Кейде DES сенімділігін арттыру үшін **ақтау (whitening)** деп аталатын тәсіл қолданылады. Бұл тәсілдің жұмысы мынадай: шифрды орындамас бұрын, ашық мәтіннің әр блогы еркін 64-битті кілтпен модуль-2 бойынша қосылады, содан кейін DES құрылғысына жөнелтіледі, соңында алынған шифр екінші 64-битті кілтпен модуль-2 бойынша қосылады. Қабылданатын жақта ақтау кері операцияны орындау арқылы алынып тасталады (бұл қабылдаушыда екі ақтау кілті болған жағдайда ғана мүмкін). Бұл тәсілді қолдану кілт ұзындығын өсіретін болғандықтан, кілт мәні кеңістігіндегі толық талдау өте ұзақ болады. Назар аударыңыз: әр блокты ақтау үшін бір кілт қолданылады (яғни, әр мәлімдеме үшін бір ғана ақтау кілті бар).

DES деректерді шифрлау стандарты құрастырылғаннан бері қайшылықтарға толы болатын. Ол IBM корпорациясы құрастырып, патенттеген Люцифер (Lucifer) шифрына негізделген болатын, айырмашылық тек IBM 56-разрядты емес 128-разрядты кілт пайдаланды. Құрама Штаттың федералды үкіметі құпия қолдану үшін қандай да бір шифрды стандарттамақ болған кезде, ол IBM-ді математика және криптография саласындағы әлемдегі ең ірі жұмыс беруші болып саналатын, Ұлттық қауіпсіздік агенттігімен NAS (National security Agency) осы сұрақты «талқылауға» «шақырды». АҚШ Ұлттық қауіпсіздік агенттігі құпиялығы соншалық, тіпті мынадай танымал қалжың бар:

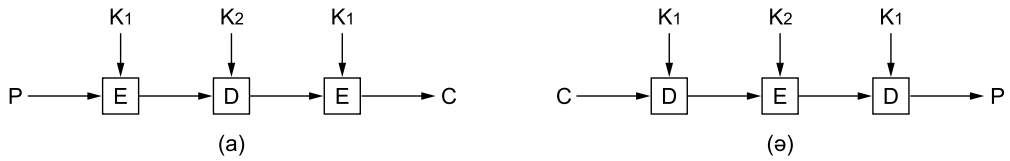
Сұрақ: NAS аббревиатурасы нені білдіреді?
Жауап: No Such Agency – Мұндай Агенттік жоқ.

Осы талқылаудан кейін IBM кілт ұзындығын 128 биттен 56 битке дейін қысқартты және DES стандартын құрастыруды құпия сақтауды шешті. Көбі кілт ұзындығының қысқаруы, NAS-тың DES-ті бұза алуына кепілдік беру үшін жасалды деп ойлады, бірақ қаржыландыруы төмен фирмалардың бұған шамасы келмейді. Жобаны құпияландыру мақсаты Ұлттық қауіпсіздік агенттігі DES шифрын жеңіл бұза алатын жасырын жолды жабу болуы ықтимал. Осы басқарма қызметкері Электротехника және электроника инженерлер институтына (IEEE) криптография бойынша жоспарланған конференцияны болдырмау жайлы ұсыныс жасаған кезде, бұл ұсыныс жайлылық қоспады. Ұлттық қауіпсіздік агенттігі әркез бәріне кедергі жасап келеді.

1977 жылы Стенфорд университетінің криптография саласындағы зерттеулермен айналысатын ғалымдары Диффи (Diffie) және Хеллман (Hellman) DES кодын бұзуға арналған машина құрастырып, оның құнын $20 млн. бағалады. Бұл машина, ашық мәтіннің кішкене бөлігі және оған сәйкес шифрланған мәтін бойынша 256 нұсқаны толық талдау арқылы бір күннің ішінде 56-разрядты кілтті анықтай алды. Қазіргі күнде мұндай машина бар, сатылады және оның құны $10 000 (Kumar және басқалар, 2006).

DES көмегімен үштік шифрлау (Triple DES)

1979 жылы IBM корпорациясы DES стандарты кілтінің тым қысқа екенін түсініп, үштік шифрлау арқылы оның сенімділігін едәуір жоғарылататын тәсіл құрастырды (Tuchman, 1979). Таңдап алынған тәсіл содан бері Халықаралық стандарт 8732 болды, ол *8.7-суретте* бейнелгенген. Бұл жерде екі кілт және үш саты қолданылады. Бірінші сатыда ашық мәтін $К_1$ әдеттегі DES кілтімен шифрланады (суреттегі Encryption блогы). Екінші сатыда DES $К_2$ кілтін пайдаланып кері шифрлау режимінде жұмыс істейді (Decryption блогы). Соңында $К_1$ кілтімен тағы бір шифрлау операциясы орындалады.



8.7-сурет. DES көмегімен үштік шифрлау (а); кері шифрлау (ә)

Бірден екі сұрақ туындайды. Біріншіден, не себепті үш кілт емес, екі кілт қолданылады? Екіншіден, не себепті **EEE (Encrypt Encrypt Encrypt – Шифрлау Шифрлау Шифрлау)** емес, **EDE (Encrypt Decrypt Encrypt – Шифрлау Кері шифрлау Шифрлау)** операциялар тізбегі орындалады? Бар-жоғы екі кілт қолдану себебі – әлемдегі ең параноик криптографтардың өзі (криптография саласында паранойя ауыру емес артықшылық деп саналады) коммерциялық

ұсыныстар үшін ұзындығы 112 бит кілт жеткілікті деп есептейді. 168 разрядқа көшу-сақтау және қосымша кілтті транспорттау үшін қосымша шығындарды талап етеді, ал нақты көмегі шамалы.

Шифрлау, кері шифрлау, шифрлау тізбектерін қолдану қолданыстағы бір кілтті DES-жүйесімен кері үйлесімділікпен түсіндіріледі. Шифрлау және кері шифрлау екі функциялары 64-разрядты сандар жиынтығының арасындағы сәйкестікті орнатады. Криптография тұрғысынан екі функция да бірдей сенімді. Алайда, EEE орнына EDE қолдану үштік шифрлауды пайдаланатын компьютерге әдеттегі бірлік шифрлауды пайдаланатын компьютермен қарапайым $К_1=К_2$ теңдігі арқылы әрекеттесуге мүмкіндік береді. Осылайша, үштік шифрлауды қосымша жұмыс режимі ретінде жеңіл қосуға болады, бұл криптограф ғалымдар үшін қызық емес, бірақ IBM корпорациясы және оның клиенттері үшін маңызды.

8.2.2. AES жақсартылған шифрлау стандарты

Белгілі бір сәтте DES реурсының (тіпті, үштік шифрлаумен бірге) аяқталып келе жатқаны белгілі болды. Осы кезде, **NIST (National Institute of Standards and Technology – Стандарттар және технологиялар ұлттық институты)** – АҚШ федералды агенттігі үшін стандарттар құрастырумен айналысатын, сауда министрлік агенттігі – үкіметке құпия емес деректер үшін жаңа стандарт керек деп шешті. NIST DES байланысты қайшылықтарды анық білді және жаңа стандарт құрастырылғандығы жайлы хабарланысымен, криптография саласында сәл түсінігі барлардың барлығы, үнсіз келісім бойынша, мұнда Ұлттық қауіпсіздік агенттігі кез келген ақпаратты жеңіл бұза алатын бір жол бар деп болжайтындығы да түсінді. Осындай шарттармен ешкім жаңа технологияны пайдалануға келіспес еді және бұл технология танымал болмастан өліп кетер еді.

Осындай алғышарттарға сүйене отырып, NIST үкіметтік бюрократиялық аппарат күтпеген амал қолданды: ол криптографиялық байқауға спонсор болуды шешті. 1997 жылдың қаңтар айында бүкіл әлемнің ғалымдары жаңа стандартқа байланысты **AES (Advanced Encryption Standard – жақсартылған шифрлау стандарты)** деп аталған байқауға өз құрастыруларын ұсынуға шақырылды. Құрастыруларға қойылған талаптар мынадай болды:

1. Алгоритм симметриялы блоктық кілтті пайдалануы тиіс.
2. Құрастырылымның барлық егжей-тегжейі жалпыға қолжетімді болуы керек.
3. Ұзындығы 128, 192 және 256 бит кілттер қолданылуы тиіс.
4. Программалық та, аппараттықта жүзеге асырылу мүмкін болу керек.
5. Алгоритм жалпыға қолжетімді немесе өз беделін түсірмеген түсініктерге негізделуі тиіс.

Салмақты он бес ұсыныс қарастырылды. Жалпыға қолжетімді конференцияларда құрастырушылар өз жобаларын ұсынады, ал оппоненттер әр жобадағы кемшіліктерді табу үшін бар күштерін салады. 1998 жылдың маусым айында стандарттар және технологиялар институты көшбастаған бес жобаны таңдап алды. Таңдау, негізінен келесідей аспектілерге негізделіп жүргізілді: қамтамасыз етілетін қауіпсіздік, тиімділік, қарапайымдылық, иілгіштік, сонымен бірге жадыға деген талап (бұл кіріктірілген жүйелер үшін маңызды). Көптеген сынды ескертулер айтылған тағы да бірнеше конференция өткізілді.

2000 жылдың қазан айында NIST Джон Домен мен Винсент Райменнің Rijndael таңдағанын жариялады. Rijndael аты (Райн-дол деп оқылады) екі автор ата-тегінің қысқартылуынан алынған: Раймен+Домен. 2001 жылдың қараша айынан бастап Rijndael ақпаратты өңдеудің федералды стандарты, FIPS 197 деп жарияланған АҚШ үкіметінің стандарты болды. Байқаудың толығымен ашық түрде өтуінің, сонымен бірге Rijndael техникалық мүмкіндіктерінің және байқауда жеңіп шыққан команда Бельгияның екі жас криптографтарынан тұрғандығының арқасында (олардың қандай да бір бұзу жолдарын көрсетіп, NSA бірлесіп жұмыс істеуі екіталай), Rijndael әлемдегі басым криптографиялық стандарт болды. Қазіргі уақытта AES кейбір микропроцессорларға арналған нұсқаулардың бір бөлігі (мысалы, Intel).

Rijndael ұзын кілттерді және қадамы 32 биттен ұзындығы 128-ден 256 битке дейінгі блоктарды қолдайды. Кілт және блоктар ұзындығы бір-біріне тәуелсіз өзгеріп отыруы мүмкін. Осылай бола тұрса да, AES стандарты блок көлемі 128-битке тең болуы керек дейді, ал кілт ұзындығы – 128, 192 және 256 бит. Алайда, кімде кімнің 192-битті қолдануы екіталай, сондықтан іс жүзінде AES-те екі нұсқа пайдаланылады: 128-биттік блок және 128-биттік кілт, сонымен бірге 128-биттік блок және 256-биттік кілт.

Төменде алгоритмнің сипаттамасы келтірілген, онда біз тек бір жағдайды қарастырамыз – 128/128, себебі осы нұсқа коммерциялық қосымшалар нормасы болады деп жорамалдаймыз. 128-биттік кілт – мәндер кеңістігінің көлемі дегенді білдіреді. Тіпті, Ұлттық қауіпсіздік агенттігі әрқайсысы бір кілтті пикосекундта есептейтін, миллион параллель жалғанған процессорлардан тұратын машина құрастырса да, барлық нұсқаларды талдау үшін 1010 жыл қажет болады. Ол уақытқа дейін күн де сөнеді, ал біздің алыстағы ұрпақтарымыз кілт мәні жазылған баспаны майшам жарығымен оқуға тура келеді.

Rijndael

Математикалық тұрғыдан алғанда Rijndael Галуа өрістері теориясына негізделген, осыған орай құпиялыққа байланысты кейбір қасиеттерді қатаң түрде дәлелдеуге болады. Осылай бола тұрса да, оны математикалық егжей-тегжейіне тоқталмай, С тіліндегі программа коды тұрғысынан қарастыруға да болады.

DES-тегідей Rijndael-да да алмастырулар мен орын ауыстырулар қолда-

нылады. Екеуі де бірнеше итерацияны пайдаланады, олардың саны кілт және блоктың ұзындығына тәуелді: 128-разрядты кілт, 128-биттік блок үшін 10-ға; ең ұзын кілт және үлкен көлемдегі блоктар үшін итерация саны 14-ке тең. Алайда, DES ерекшелігі – барлық операциялар бүтін байттармен орындалады, бұл аппараттық та, программалық та тұрғыдан тиімді жүзеге асыруларды орындауға мүмкіндік береді. Rijndael тәсілінің схемалық алгоритмі *8.1-листингте* келтірілген. Бұл кодтың иллюстрациялық мақсатта қолданылғанына назар аударыңыздар. Қауіпсіздікті қамтамасыз ететін кодтың жақсы жүзеге асырылулары, пайдаланғаннан кейін қабылдағыш жадыны тазарту тәрізді белгілі бір қосымша іс жүзілік қолданыстары болуы тиіс. Мысал үшін Ferguson және басқалар (2010) басылымын қараңыз.

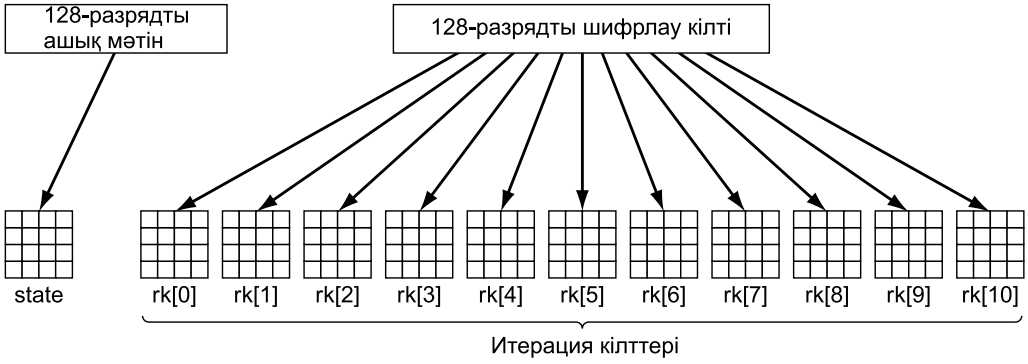
8.1-листинг. Rijndael тәсілінің схемалық алгоритмі

```
#define LENGTH 16          /*деректер блогындағы немесе кілттегі байттар саны*/
#define NROWS 4             /*state массивіндегі жолдар саны */
#define NCOLS 4             /* state массивіндегі бағаналар саны */
#define ROUNDS 10           /* итерация саны */
typedef unsigned char byte;   /* 8-разрядты таңбасыз бүтін сан*/

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
int r;                                  /*цикл санауышы*/
byte state[NROWS][NCOLS];       /* ағымдағы қалып-күй*/
struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1]  /*итерация кілті*/

expand_key(key, rk);                  /* итерация кілттерін құрастыру*/
copy plaintext to state(state, plaintext);  /* ағымдағы қалып-күйді
                                                          инициализациялау*/
xor_roundkey_into_state(state, rk[0]);  /* кілт және ағымдағы қалып-
                                                    күйді модуль-2 бойынша қосу*/

for (r = 1; r <= ROUNDS; r++) {
   substitute(state);           /* әр битті S-блок арқылы өткізу*/
        rotate rows(state);          /*і жолын і байтқа бұру*/
  if (r < ROUNDS) mix_columns(state);  /*араластырушы функция*/
  xor_roundkey_into_state(state, rk[r]); /* кілтті ағымдағы қалып-күймен
                                                          модуль-2 бойынша қосу*/
  }
copy_state_to_ciphertext(ciphertext, state);  /* нәтижені қайтару*/
}
```

Rijndael функциясының үш аргументі бар: *plaintext* – кіріс деректерден тұратын, өлшемі 16 байттық массив, *ciphertext* – шифр жазылатын, өлшемі 16 байттық массив, сонымен бірге *key* – кілт жазылған, өлшемі 16 байт массив. Есептеу барысында деректердің ағымдағы қалып-күйі state байттық массивінде сақталады, оның өлшемі NROWS*NCOLS тең. 128-биттік блоктық деректер үшін бұл массивтің көлемі байтқа тең. 16 байтқа бір блок толығымен сияды.

State массивіне бастапқыдан ашық мәтін жазылады және есептеудің әр сатысында түрлендіріледі. Кейбір сатыларды байттық орын ауыстырулар орындалады. Басқа сатыларда – массив ішінде байттардың орнын ауыстыру. Басқа түрлендірулер де орындалуы мүмкін. Соңында *state* массивінің мәні нәтиже ретінде қайтарылатын, шифрланған мәтін болады. Алгоритм өлшемдері қалып-күйді білдіретін (state) массивімен бірдей 11 массивке кілтті тарату-дан басталады. Бұл массивтер қалып-күй массивтері жазылған, *rk* – массивтер құрылымында сақталады. Осындай құрылымдардың бірі есептеу басында, ал қалған оны – он итерация барысында қолданылады (бір итерацияға біреуі). Әр итерация үшін кілтті есептеу күрделі жолмен жүргізіледі, біз бұл үдерістің егжей-тегжейін қарастырмаймыз. Бұл есептеулер кезінді кілт разрядтарының әртүрлі тобы үшін циклдік бұрулар және модуль-2 бойынша қосулар орындалатынын айта кетсек жеткілікті. Толығырақ Daemen және Rijmen (2002) басылымынан білуге болады.

Келесі қадамда, ашық мәтінді алдағы итерациялар кезінде өңдеу үшін *state* массивіне көшіріледі. Мәтін 4-байт бағаналарға көшіріледі: бірінші 4 байт 0 бағанаға түседі, екінші – 1 бағанаға және т.с.с. Жолдар да, бағаналар да нөлден бастап нөмірленеді, ал итерациялар – 1. Өлшемі 12 байттық массивті құрастыру 8.8-суретте көрсетілген.



8.8-сурет. state және rk массивтерін құрастыру

Есептеудің негізгі циклының басында тағы бір амал орындалады: rk[0] әр разряды state массивімен модуль-2 бойынша қосылады. Басқа сөзбен айтқанда, state массивінің 16 байты, өзі және rk[0]-дің сәйкес байтымен модуль-2 бойынша қосындысымен алмастырылады.

Осыдан кейін ғана негізгі есептеулер басталады. Циклде 10 итерация жүргізіледі, әр итерацияда state массиві түрлендіруге ұшырайды. Әр итерация 4 қадамнан тұрады. Бірінші қадамда state массивінде жеке символдық орын ауыстыру орындалады. Әр байт кезек-кезек S-блок үшін индекс ретінде қолданылады және мәні S-блоктың сәйкес жазбасына алмастырылады. Бұл

қадамда түзу дара әліппелік алмастырушы шифр алынады. Бірнеше S-блок қолданылатын DES айырмашылығы – Rijndael-де S-блок біреу.

Екінші қадамда төрт жолдың әрқайсысы солға бұрылады. Нөлінші жол 0 байтқа (яғни, өзгермейді), 1 жол – 1 байтқа, 2 жол – 2 байтқа, 3 жол – 3 байтқа бұрылады. Бұл қадамның мағынасы деректерді блок айналасында шашу. Бұл *8.5-суретте* көрсетілген орын ауыстыруға сәйкес келеді.

Үшінші қадамда барлық бағаналарды тәуелсіз араластыру орындалады. Бұл матрицалық көбейту операциясының көмегімен жүргізіледі, нәтижесінде әрбір жаңа бағана ескі бағана және тұрақты матрица көбейтіндісіне тең болады. Мұнда көбейту соңғы Галуа өрісін, GF(28) пайдалану арқылы жүргізіледі. Осының барлығы күрделі болып көрінгендігіне қарамастан, матрицаның әр элементі екі рет кестеге жүгіну және үш рет модуль-2 бойынша қосу арқылы есептеледі (Daemen және Rijmen, 2002, Appendix E).

Барлық әрекеттердің қайтымдылығының арқасында кері шифрлау осындай алгоритм көмегімен орындалады, тек қадамдар кері ретпен жүргізіледі. Алайда, мұнда кері шифрлауды шифрлау алгоритмінің өзгертілген кестесін пайдаланып орындауға болатын бір қулық бар. Жақсы программалық жүзеге асыру жиілігі 2 ГГц машинада деректерді 700 Мбит/с жылдамдықпен шифрлай алады. Мұндай жылдамдық MPEG-2 форматындағы бейнені нақты уақыт масштабында шифрлауға жеткілікті. Аппараттық жүзеге асырулар бұдан да жылдам жұмыс істейді.

8.2.3. Шифрлау режимі

Барлық күрделігіне қарамастан, AES (немесе DES, немесе кез келген басқа блоктық код) дара әліппелік ұзын символды (AES 128-биттік символ, DES – 64-биттік символдар пайдаланады) орын ауыстырушы шифр болып келеді. Ашық мәтіннің кез келген бөлігін бір шифрлаушы блоктан өткізген кезде бір шифр алынады. Айталық, егер сіз *abcdefgh* мәтінін DES алгоритмі үшін бір кілтпен 100 рет шифрлауға тырыссаңыз да, 100 бірдей шифр көшірмесін аласыз.

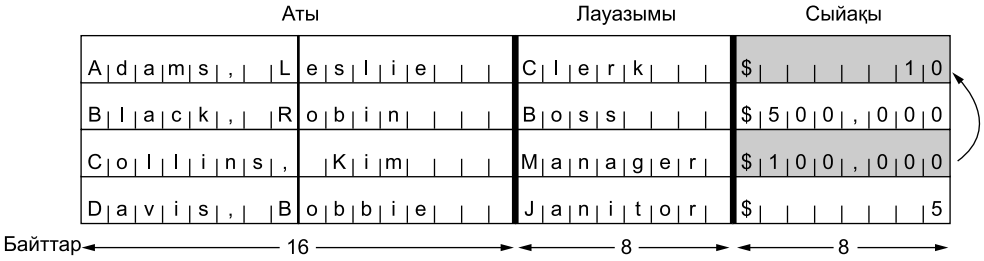
Электронды шифрблокнот режимі

Дара әліппелік алмастыру шифрының бұл қасиетін шифр бөлігін бұзу үшін қалай қолданылатындығын түсіну үшін біз DES стандарты бойынша кодтауды (үштік) біз 64-разрядты блоктарды бейнелеу 128-разрядтығына қарағанда жеңіл болғандықтан ғана қарастырамыз. Мұнда AES-тің де дәл осындай мәселелермен қақтығысатынын есте сақтау керек. Ұзын мәлімдемені кодтаудың ең дұрыс жолы оны 8 байттық (64 бит) жеке блоктарға бөлу. Содан кейін бұл блоктарды кезек-кезек бір кілтпен кодтау. Қажет болған жағдайда соңғы блокты 64 битке дейін толтыруға болады. Мұндай тәсіл, сөздер және оған сәйкес шифрдан тұратын (әдетте бұл бес орынды ондық сандар бола-

тын) ескі шифроблокноттарға ұқсас, **электронды шифроблокнот режимі (Electronic Code Book mode – ECB mode)** деп аталады.

8.9-суретте компания қызметкерлеріне берілетін ынталандыру сыйлық ақы тізбегі көрсетілген компьютерлік файл бейнеленген. Бұл файл әр жазба бір қызметкерге арналған, келесі форматтағыдай тізбектік 32-разрядты жазбадан тұрады: 16 байт – аты, 8 байт – лауазымы, 8 байт – сыйлық ақы. Он алты сегіз байттық блоктың әрқайсысы (0-ден 15-ке дейін нөмірленген) DES шифрымен кодталады.

Лессли қазір ғана бастығымен ренжісіп қалды және үлкен сыйлық ақы алам деп үміттенбейді. Ким, керісінше, бастықтың жақсы көретін қызметкері, бұл барлығына белгілі. Лесли, файл шифрланғаннан кейін оған банкке жөнелтілмей тұрып қолжеткізе алады. Лисли тек шифрланған мәтінге қолжеткізіп, жағдайды өзгерте ала ма?



8.9-сурет. 16 DES-блок түрінде шифрланған ашық мәтін файлы

Бұл жағдайда барлығы өте қарапайым. Леслиге тек шифрланған 12 блокты (Кимнің сыйлық ақысы жазылған жол) көшіріп, оны 4 блокпен (Лесли сыйлық ақысы көрсетілген жол) алмастыру керек. Тіпті, 12 блоктың мәнін білмей ақ, Лесли Рождество қызықты болады деп үміттене алады. (Шифрланған 8 блокты да көшіріп алуға болады, бұл жағдайда өтіріктің шығып қалу ықтималдығы жоғары, оның үстіне Лесли сараң адам емес).

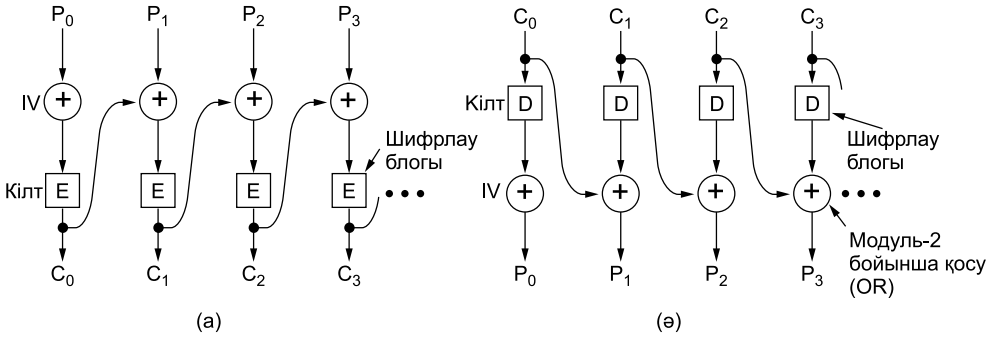
Шифр блоктарын іліндіру режимі

Осындай шабуылдарға төтеп беру үшін шифрларды бір блокты өзгерту, кері шифрланғаннан кейін ашық мәтіннің басқа блоктарының (өзгертілген жерден бастап) бұзылып, қоқысқа айналдыратындай етіп түрлендіруге болады. Осындай тәсілдердің бірі – **шифр блоктарын іліндіру (cipher block chaining).** 8.10-суретте көрсетілген бұл тәсілде, ашық мәтіннің әр блогы шифрланар алдында алдыңғы шифрланған блокпен модуль-2 бойынша қосылады. Мұнда ашық мәтіннің бірдей блоктарына шифрланған мәтіннің бірдей блоктары сәйкес келмейді. Осылайша, шифр дара әліппелік алмастыру шифры болудан қалады. Бірінші блок шифрланған мәтінмен бірге ашық мәтін ретінде

жөнелтілетін, кездейсоқ инициялизация векторымен, **IV (Initialization Vector)**, модуль-2 бойынша қосылады.

Шифр блоктарын іліндіру жұмысын *8.10-суретте* берілген мысал арқылы қарастырайық. Алдымен $C_0 = E(P_0 \text{ XOR } IV)$ есептейміз. Содан кейін $C_1 = E(P_1 \text{ XOR } C_0)$ және т.с.с. кері шифрлау $P_0 = IV \text{ XOR } D(C_0)$ формуласы және т.с.с. жүргізіледі. Назар аударыңыздар, і блогы ашық мәтіннің 0-ден і-1-ге дейінгі барлық блоктарының функциясы болып келеді, сондықтан мәтіннің бірдей блоктары орналасуына байланысты әртүрлі шифрланған блогына түрлендіріледі. Шифрлаудың осындай тәсілін пайдалану кезінде Лесли орындаған түрлендіру, Лесли сыйақысынан бастап, мағынасыз екі блоктың пайда болуына әкеледі. Қауіпсіздің қызметінің алғыр қызметкері үшін бұл жәйт келесі іздестіру жұмыстары үшін еске салу болуы мүмкін.

Сонымен бірге, шифр блоктарын іліндірудің криптосараптаушы жұмысын күрделендіретін артықшылығы бар, себебі ашық мәтіннің бір блогы әртүрлі шифрланған мәтінге түрлендіріледі. Сипатталған тәсіл нақты осы себептерге байланысты қолданылады.



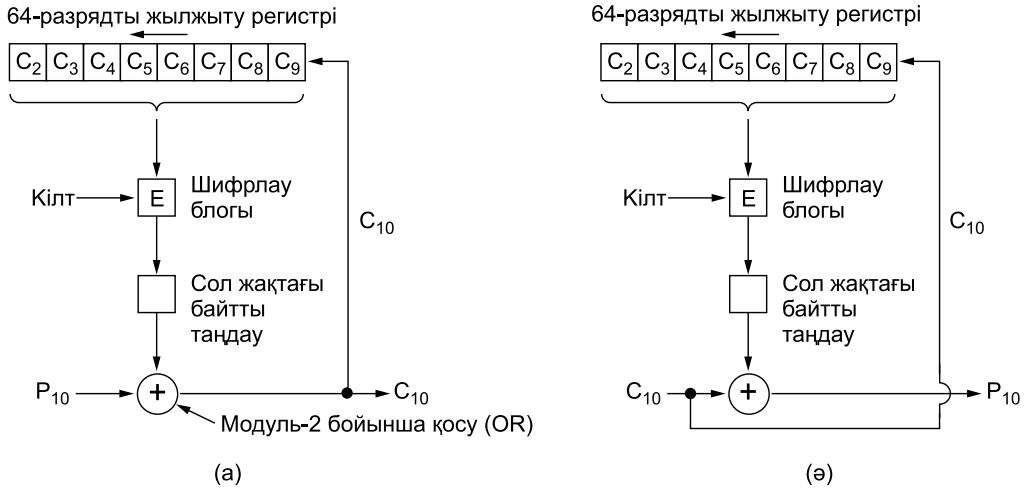
8.10-сурет. Шифрланған блоктарды іліндіру: а – шифрлау; ә – кері шифрлау

Кері байланысты шифрлау

Алайда, шифр блоктарын іліндіру тәсілінің кемшілігі де бар – шифрлау немесе кері шифрлау басталмас бұрын толық 64-биттік деректер блогы пайда болуы тиіс. Сегіз символдық жолды теріп, жауап күту – интерактивті терминалдар тұтынушысы үшін жарамсыз тәсіл. Байттық шифрлау үшін, *8.11-суретте* көрсетілгендей DES-ті (үштік) пайдаланып, **кері байланысты шифрлау режимін (cipher feedback mode)** қолдануға болады. AES стандарты үшін де идея осындай, тек 128-разрядты жылжыту регистрі қолданылады. Суретте біз шифрлау машинасының 0-ден 9-ға дейінгі байттар шифрланып, жөнелтілгеннен кейінгі қалып-күйін көреміз. Ашық мәтіннің оныншы байты келген кезде, *8.11 а-суретінде* көрсетілгендей, DES алгоритмі 64-разрядты шифрланған

блок шығару үшін 64-разрядты жылжыту регистрін өңдейді. Осы шифрланған мәтіннің сол жақтағы байты алынып P_{10}-мен модуль-2 бойынша қосылады. Бұл байт торап арқылы жөнелтіледі. Сонан кейін жылжыту регистрі солға 8 разрядқа жылжиды. Осы кезде C_2 байты регистрдың сол жағынан алынады, ал C_{10} байты оның орнына, C_9 байтының оң жағына орналастырылады. Жылжыту регистры мәнінің ашық мәтіннің алдыңғы мазмұнына байланысты екеніне назар аударыңыздар, сондықтан бастапқы мәнінің қайталанушы фрагменттері әр кез әртүрлі болып кодталады. Шифр блоктарын іліндіру тәсіліндегідей бұл тәсілмен шифрлауды бастау үшін инициализациялау векторы қажет.

Кері байланысты шифрлау пайдалану кезінде кері шифрлау шифрлауға ұқсас. Жеке алғанда, жылжыту регистрінің мәні *кері шифрланбайды*, ал шифрланады, сондықтан P_{10}-ды алу үшін C_{10}-мен модуль-2 бойынша қосылатын байт, C_{10}-ды алу үшін P_{10}-мен модуль-2 бойынша қосылатын байтқа тең. Екі жылжыту регистрінің мәндері бірдей болып тұрған кезде кері шифрлау дұрыс орындалады. Бұл *8.11 ә-суретінде* көрсетілген.



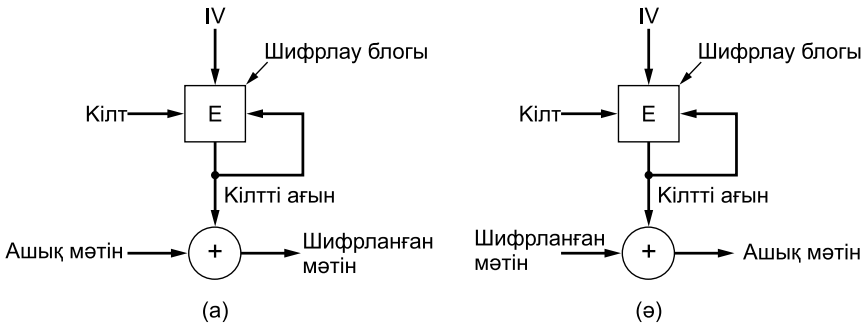
8.11-сурет. Кері байланысты шифрлау режимі: а – шифрлау; ә – кері шифрлау

Кері байланысты шифрлау тәсілімен шифрланған мәтінді тасымалдау кезінде бір бит кездейсоқ инверсияланған кезде, осы режимімен келеңсіз жағдайлар орын ала бастайды. Бұл жағдайда, бұзылған байт жылжыту регистрінде болған кезде шифрланып жатқан 8 байт бұзылған болып саналады. Бұзылған байт регистрден кеткен кезде, шығыста тағы да бұзылмаған, дұрыс ашық мәтін шифрлана бастайды. Осылайша, бір биттің инверсиялануы салыстырмалы түрде оқшауландырылып, мәлімдеменің қалған бөлігін бұзбайды.

Топтық шифр режимі

Осылай бола тұрса да, тасымалдану кезінде бұзылған бір бит ашық мәтіннің 64 битін бұзатын қосымшалар бар, ал бұл тым көп. Осындай қосымшалар үшін **топтық (ағындық) шифрлау режимі (stream cipher mode)** деп аталатын төртінші нұсқа бар. Бұл нұсқаның мағынасы мынада: шығыс блок инициализациялау векторының кілтті пайдаланып шифрлаудан алынады. Содан кейін шығыс блок кілтті пайдаланып тағы бір рет шифрланады, нәтижесінде екінші шығыс блогы алынады. Үшінші блокты алу үшін екінші блок шифрланады және т.с.с. **Кілттік ағын (keystream)** деп аталатын шығыс блоктар тізбегі (еркін ұзындықтағы) бір реттік блокнот ретінде қабылданады және ашық мәтінмен модуль-2 бойынша қосылады. Нәтижесінде, *8.12 а-суретінде* көрсетілгендей, шифрланған мәтін алынады. Назар аударыңыздар: инициализация векторы тек бірінші қадамда пайдаланылады. Содан кейін шығыс блоктар шифрланады. Бұдан басқа, кілттік ағын деректерге тәуелді емес, сондықтан қажет болған жағдайда ол алдын ала есептелуі мүмкін және ол тасымалдау қателіктеріне сезімтал емес. Кері шифрлау үдерісі *8.12 ә-суретінде* көрсетілген.

Кері шифрлау қабылдаушы бетте дәл осындай кілттік ағынды генерациялаудан басталады. Ол тек инициализациялау векторына және кілтке тәуелді болғандықтан, шифрланған мәтінді тасымалдау қателіктері оған әсер етпейді. Осылайша, тасымалданатын шифрдың бір битіндегі қателік кері шифрланған мәтіннің тек бір битіндегі қателікке әкеледі.



8.12-сурет. Топтық шифр: а – шифрлау; ә – кері шифрлау

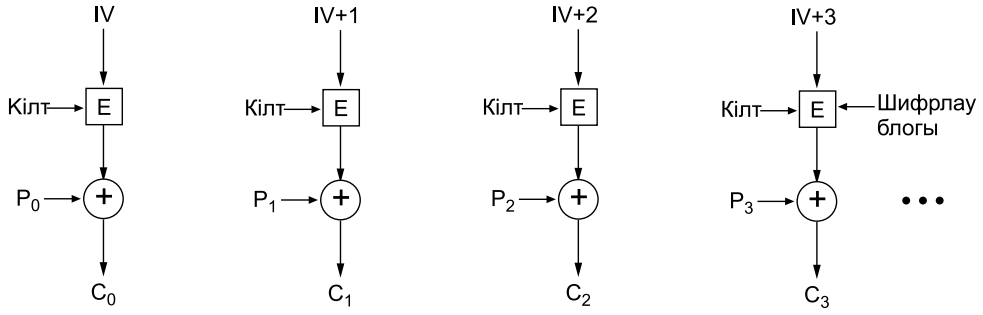
Бір топтық шифрда, кілт және инициализациялау векторының бір парын пайдаланбаған маңызды, себебі әркез бірдей кілттік ағын алынатын болады. Кілттік ағынды қайталап пайдалану, **шифрды кілттік ағынды бірнеше рет пайдалану көмегімен бұзу (keystream reuse attack)** деп аталатын жағымсыз әсерге әкелуі мүмкін. Айталық ашық мәтіннің P_0 блогы кілттік ағын көмегімен шифрланады, нәтижесінде P_0 және K_0-дің модуль-2 бойынша қосындысы алынады. Сонан кейін ашық мәтіннің екінші блогы Q_0 алынады да, тағы сол кілттік ағын көмегімен шифрланады (Q_0 XOR K_0 аламыз). Екі блокты да ұстап алған

криптосараптаушы оларды модуль-2 бойынша қосып, кілтті алып тастайды, нәтижесінде P_0 XOR Q_0 алынады. Егер олардың бірі белгілі болса (немесе оны тауып алуға болатын болса), екіншісін табу қиын емес. Кез келген жағдайда, ашық мәтіннің екі блогының модуль-2 бойынша қосындысын мәлімдеменің статистикалық қасиеттерін пайдаланып бұзуға болады. Айталық, егер ағылшын мәтіні тасымалданатын болса, онда ағында ең жиі кездесетін әріп «е» болады және т.с.с. Қысқасы, ашық мәтіннің екі бөлігінің модуль-2 бойынша қосындысы бар болған жағдайда, сындырушы екі бөлікті де үлкен ықтималдықпен есептеп таба алады.

Санауыш режимі

Электронды шифроблокноттан басқа режимдердің барлығының бір жағымсыз қасиеті бар: шифрланған мәтіннің еркін бөлігіне қолжеткізу мүмкін емес. Мәселен, файл желі арқылы тасымалданады, содан кейін дискте шифрланған күйде сақталады делік. Кейде осылай болады да, егер қабылдаушы компьютер жеңіл ұрланып кететін ноутбук болса. Барлық маңызды деректерді шифрланған күйде сақтау шынында да пайдалы: аппаратураның «жаман ағалар» қолына түскен кезде құпия ақпараттық таралып кету қаупі күрт төмендейді.

Алайда, дисктегі файлдарға қолжеткізуді жиі еркін ретпен жүргізуге тура келеді. Бұл, әсіресе деректер базасы файлдарына қатысты. Егер файл блоктарды іліндіру режимінде шифрланған болса, онда қажет блоктың алдындағы барлық блоктарды алдымен кері шифрлау қажет болады. Мұндай жұмыстың біршама ыңғайсыз екендігіне келісетін шығарсыздар. Осы себептен тағы бір шифрлау режимі енгізілген болатын – **санауыш режимі (counter mode)**. Ол *8.13-суретте* көрсетілген. Мұнда ашық мәтін тікелей шифрланбайды. Оның орнына инициализация векторы және қандай да бір тұрақты шифрланады, нәтижесінде алынған шифр ашық мәтінмен модуль-2 бойынша қосылады. Әрбір жаңа блокты шифрлау кезінде инициализациялау векторы бойынша 1-ге жылжып, файлдың кез келген жерінен кері шифрлау әдісін алуға болады. Мұнда қажет блоктың алдыңғы блоктардың барлығын кері шифрлау керек емес.



8.13-сурет. Санауыш режимінде шифрлау

Санауыш режимінде шифрлаудың пайдалылығына қарамастан оның еске сала кететін бір елеулі кемшілігі бар. Айталық, бұған дейін бір қолданылған К кілтін қайталап қолдандық (басқа ашық мәтінмен, бірақ инициализация векторы сол) және зиянкес екі жағдайда да тасымалданған шифрланған мәтінді ұстап алды делік. Кілттік ағын өзгеріссіз қалады, нәтижесінде кілттік ағынды қайталап пайдалану салдарынан шифрды бұзу қаупі туындайды (біз бұл әсерді топтық шифрды талқылаған кезде айтқан болатынбыз). Криптосараптаушыға тек екі ұстап алынған мәлімдемені модуль-2 бойынша қосса жеткілікті. Бұл кемшілік санауыш режимі толығымен жарамсыз дегенді білдірмейді. Бұл тек инициализация векторларын кілттер тәрізді тәуелсіз және кездейсоқ таңдалу керек дегенді білдіреді. Тіпті, бір кілт екі рет кездейсоқ кездессе де, ақпаратты ұстап алудан ұқсас емес инициализация векторы сақтап қалуы мүмкін.

8.2.4. Басқа шифрлар

8.2-кесте. Кейбір кеңінен таралған симметриялы кілтті криптографиялық алгоритмдер

Аты	Автор	Кілт ұзындығы	Түсініктеме
DES	IBM	56 бит	Қазіргі заманғы жүйелер үшін тым әлсіз
RC4	Рональд Ривест (Ronald Rivest)	1 – 2048 бит	Назар аударыңыз: әлсіз кілттер бар
RC5	Рональд Ривест (Ronald Rivest)	128 – 256 бит	Жақсы, бірақ патенттелген
AES (Rijndael)	Домен (Daemen) , Райммен (Rijmen)	128 – 256 бит	Ең жақсысы
Serpent	Андертсон (Anderson), Байхэм және Кнадсен (Biham) (Knudsen)	128 – 256 бит	Өте күшті
DES	IBM	168 бит	Жақсы, бірақ ескірген
Twofish	Брюс Шнайер (Bruse Schneier)	128 – 256 бит	Өте күшті; кең таралған

AES (Rijndael) және DES – бұл ең танымал симметриялы кілтті криптографиялық алгоритмдер және олар өндірушілердің стандартты таңдауы болып келеді, кем дегенде жауапкершілік тұрғысынан алғанда. (Егер сіз өз өніміңізде AES-ті пайдаланып, оны кімде-кім бұзса, стандартталмаған шифрды пайдаланғандықтан ол бұзылды деп ешкім сізге кінә таға алмайды.) Дегенмен де, табиғатта симметриялы кілтті көптеген басқа шифрлардың бар екендігін айта кеткен дұрыс. Олардың кейбіреулері әртүрлі аппаратты-программалық

өнімдерге кіріктірілген. Олардың ішіндегі кеңінен таралғандары *8.2-кестеде* келтірілген. Бұны шифрлардың комбинациясына да пайдалануға болады, мысалы, AES және Twofish, деректерді алу үшін екі шифрды да бұзу керек болады.

8.2.5. Криптосараптау

Криптографиядағы симметриялы кілттерді пайдалану жайлы әңгімемізді аяқтамас бұрын криптосараптау дамып келе жатқан төрт бағыт жайлы еске сала кеткен дұрыс. Бірінші бағыт **дифференциялды криптосараптау** деп аталады (Biha Shamir, 1997). Ол кез келген блоктық шифрды бұзуға арналған. Алдымен ашық мәтіннің бірнеше битте айырмашылығы бар, қос блогы сарапталады. Мұнда шифрлау барысында әрбір ішкі итерация кезінде орын алатын әрекеттер мұқият сарапталады. Көптеген жағдайда, бір биттік тізбектің қалғандарынан көбірек кездесетінін байқауға болады және ықтималдықтар теориясына негізделіп, осы жағдай шифрды бұзу үшін қолданылады.

Екінші бағыт, **сызықтық криптосараптау** деп аталады (Matsui, 1994). Оның көмегімен белгілі 243 ашық мәтін блогы бар DES шифрын бұзуға болады. Жұмыс принципі ашық мәтіннің кейбір биттерін модуль-2 бойынша қосып, нәтижені шаблондық тізбектер үшін сараптауға негізделген. Егер осы процедураны бірнеше рет қайталаса, жартысы нөлдік, жартысы – 1 мәнге ие болады. Дегенмен де, бұл қатынас жиі екі жақтың біріне ауытқып отырады. Бұл ауытқу қаншалықты аз болса да, криптосараптаушы еңбек шығынын азайтуға пайдалануға болады. Толық ақпараты осы тәсілдің авторы Мацуи (Matsui) материалдарынан алуға болады.

Дамудың үшінші бағыты құпия кілтті есептеуге жұмсалатын электр қуатын сараптаумен байланысты. Әдетте компьютерлердегі кернеу 3 В логикалық бірлікке, ал 0 В – логикалық нөлге сәйкес келеді. Осылайша, логикалық бірлікті өңдеу, логикалық нөлге қарағанда үлкен электр қуатын қажет етеді. Егер криптографиялық алгоритм кілт разрядтары кезегімен өңделетін циклден тұратын болса, онда сындырушы негізгі жүйелік n-гигогерцтік тактілік генераторды баяу генератормен (айталық, жиілігі 100 Гц) алмастырып және орталық процессордың қоректену және жерге тұйықтау аяқтарына «қолтырауын» қыстырып, әр машиналық нұсқаудың пайдаланатын қуатын үлкен дәлдікпен анықтай алады. Осы деректер бойынша құпия кілтті анықтау өте жеңіл болады. Криптосаптаудың бұл тәсілін тек алгоритмді ассемблер тілінде электр қуатын пайдалану жалпы кілтке де, әр итерация кілтіне де тәуелді болмайтындай етіп, тиянақты кодтаумен ғана жеңуге болады.

Төртінші бағыт уақыттық сараптауға негізделген. Криптографиялық алгоритмдердің итерациялық кілттерді тестілейтін, үлкен сандағы шартты операторлары *(if)* бар. Егер *then* және *else* бөліктері әртүрлі уақытта орындалатын болса, онда жүйелік тактілік генераторды баяулатып, барлық қадамдардың ұзақтығын өлшеп итерация кілттерін есептеуге болады. Осы кілттер бойынша жалпы кілттерді есептеуге болады. Электр қуатын және уақытты сараптауды

бірмезгілге пайдалануға болады, бұл криптосараптау жұмысын жеңілдетеді. Электр қуатын және уақытты сараптау жұмыстары біршама таңсық болғанына қарамастан, іс жүзінде олар арнайы қорғанысы жоқ кез келген шифрды сындыра алатын өте қуатты тәсіл бола алады.

8.3. АШЫҚ КІЛТТІ АЛГОРИТМДЕР

Кілтті тасымалдау үдерісі барлық шифрлау жүйелерінде тарихи әлсіз буын болып саналатын. Криптожүйенің өзінің қаншалықты берік болғанына қарамастан, егер зиянкес кілтті ұрлап ала алатын болса, жүйе жарамсыз болып қалады. 1976 жылға дейін криптологтардың барлығы кері шифрлау кілті шифрлау кілтіне (немесе бірін екіншісінен жеңіл шығарып алуға болатындай) сәйкес болу керек деген ұстанымда болатын. Сонымен бірге, кілттер жүйені пайдаланушылардың барлығында болуы тиіс. Осылайша, бұл мәселені жоюға болмайтындай болып көрінетін: кілттер ұрланудан қорғалуы тиіс және оны тұтынушылар арасында тарату керек, сондықтан оны банк сейфінде сақтауға болмайды.

1976 жылы Стенфорд университетінің екі зерттеушісі, Диффи (Diffie) және Хеллман (Hellman) кері шифрлау кілтін шифрлау кілтінен алуға болмайтын, түбегейлі жаңа криптожүйені ұсынды. Олар ұсынған E шифрлау алгоритмі және D кері шифрлау алгоритмі (екеуі де кілтпен параметрленген) келесі талаптарды қанағаттандыруы тиіс болды.

1. $D(E(P))=P$.
2. E-ден D-ны алу өте күрделі.
3. E-ні еркін ашық мәтін көмегімен сындыруға болмайды.

Бірінші талап, егер D кері шифрлау алгоритмін E(P) шифрланған мәтінге қолданған жағдайда біз қайтадан P ашық мәтінін аламыз. Мұнсыз авторластырылған қабылдаушы мәлімдемені кері шифрлай алмайды. Екінші талап, айтпаса да түсінікті. Үшінші талап қажет, себебі зиянкестер алгоритммен қалауларынша тәжірибе жүргізе алады. Мұндай жағдайда шифрлау кілтін жалпыға қолжетімді етпеудің ешбір себебі жоқ.

Бұл тәсіл келесідей жұмыс істейді. Кімде-кім, мысалы, Алиса, құпия мәлімдеме алмақ оймен, алдымен жоғарыда аталған талаптарды қанағаттандыратын екі алгоритм құрастырады. Содан кейін, шифрлау алгоритмі және оның кілті ашық жарияланады, алгоритмнің аты да осыған байланысты **ашық кілтпен шифрлау (public-key cryptography)** деп аталады. Бұны, ашық кілтті, мысалы, Алисаның үй парағында орналастырып жариялауға болады. Алисаның ашық кілтімен параметрленген шифрлау алгоритмін белгілеу үшін біз $\boldsymbol{E}_A$ белгілеуін қолданамыз. Сәйкесінше, Алисаның жеке кілтімен параметрленген кері шифрлау (құпия) алгоритмін біз $\boldsymbol{D}_A$ деп белгілейтін боламыз. Бобта дәл осылай істейді, $\boldsymbol{E}_B$ ашық жариялайды, бірақ $\boldsymbol{D}_B$ құпия ұстайды.

Енді Алиса мен Боб арасында бұған дейін кездеспеген, сенімді арна орнату мәселесін шеше аламыз ба, соны қарастырайық. Алиса және Бобтың екі шифрлау кілті де, $\boldsymbol{E_A}$ және $\boldsymbol{E_B}$ ашық болып саналады. (Жүйенің барлық тұтынушылары, тұтынушы болғаннан кейін өз шифрлау кілттерін ашық жариялай алады.) Енді, Алиса өзінің *P* бірінші мәлімдемесін алып, $\boldsymbol{E_B}$(P) есептеп, оны Бобқа жөнелтеді. Боб оны өзінің $\boldsymbol{D_B}$ құпия кілтінің көмегімен кері шифрлайды, демек ол $D_B\left(E_B\left(P\right)\right) = P$ есептейді. $\boldsymbol{E_B}$(P) мәлімдемесін басқа ешкім оқи алмайды, себебі шифрлау жүйесі жеткілікті түрде сенімді, ал белгілі $\boldsymbol{E_B}$ кілтінің негізінде $\boldsymbol{D_B}$ кілтін алу өте қиын деп саналады. Жауапты жөнелте отырып Боб $E_A\left(R\right)$ да жөнелтеді. Осылайша, Алиса және Боб сенімді құпия байланыс арнасын алады.

Осы жерде қолданылған терминологияға назар аударыңыздар. Ашық кілтпен шифрлау әр тұтынушыда екі кілттің болуын қажет етеді – осы тұтынушыға жіберілетін, мәлімдемені шифрлауға қажет ашық кілт және келген мәлімдемелерді кері шифрлауға қажет жабық кілт. Әрі қарай біз бұл кілттерді, әдеттегі симметриялы кілтті криптографияда шифрлау және кері шифрлау үшін қолданылатын *құпия (secret)* кілттерден ажырату үшін *ашық (public)* және *жабық (private)* деп атайтын боламыз.

8.3.1. RSA алгоритмі

Бұл жердегі жалғыз кедергі, осы талаптардың барлығын қанағаттандыратын алгоритмді табу. Ашық кілтпен шифрлау артықшылығы айқын болғандығына қарамастан, көптеген зерттеушілер осындай алгоритмдері құрастырумен айналысады, кейбіреулері жарияланып та қойды. Осындай жақсы бір тәсілді Массачусетск технология институтының зерттеушілер тобы құрастырды (Rivest және басқалар, 1978). Ол үш құрастырушы тегінің бірінші әріптерімен: **RSA** (Rivest, Shamir, Adleman) деп аталған. Бұл тәсіл осымен ширек ғасыр сынару талпыныстарының бәріне төтеп беріп келеді және өте берік деп саналады. Іс жүзінде қолданылатын көптеген қауіпсіздік жүйелері осы тәсіл негізінде құрастырылған. Осы себептермен Rivest, Shamir, Adleman 2002 жылы «Тьюринг сыйы» (Turing Award) атты ACM сыйлығына ие болды. RSA басты кемшілігі – жеткілікті деңгейдегі қауіпсіздікті қамтамасыз ету үшін ұзындығы, кем дегенде 1024 бит кілт қажет (симметриялы кілтті алгоритмдердегі 128 битке қарсы). Осының салдарынан алгоритм баяу жұмыс істейді.

RSA тәсілінің негізінде сандар теориясының кейбір қағидалары жатыр. Бұл тәсілді қалай қолдану керектігін жалпылама сипаттап кетелік. Егжей-тегжейін сәйкес кітаптардан оқуға болады.

1. Екі үлкен қарапайым санды аламыз: p және q (әдетте ұзындығы 1024 бит).
2. $n=p\times q$ және $z=(p-1)\times(q-1)$ есептейміз.
3. z санымен өзара қарапайым болып келетін d санын таңдап аламыз.

4. e×d көбейтіндісін z санына бөлгендегі қалдық 1-ге тең болатындай e санын табамыз.

Осы параметрлерді алдын ала есептеп алып шифрлауды бастауға болады. Алдымен, әрбір *P* мәлімдемесі $0 \le P < n$ аралығына түсетіндей, ашық мәтінді (биттік жол ретінде қарастырылатын) блоктарға бөлеміз. Егер ашық мәтінді *k* биттен тұратын блоктарға бөлсек бұл қиын емес, мұндағы *k* – ең үлкен бүтін сан, ол үшін $2k < n$.

P мәлімдемесін $\boldsymbol{C = P^e (\mathrm{mod}\ n)}$ шифрлау үшін есептейміз. *C*-ны кері шифрлау үшін $\boldsymbol{C = C^d (\mathrm{mod}\ n)}$ есептейміз. Көрсетілген аралықтарында P-ның кез келген мәні үшін шифрлау және кері шифрлау функциялары өзара қайтымды екенін дәлелдеуге болады. Шифрлауды жүзеге асыру үшін *e* және *n* қажет. Кері шифрлау үшін *d* және *n* қажет. Осылайша ашық кілт *(e, n)* жұбынан, ал жабық кілт *(d, n)* жұбынан тұрады.

Тәсілдің сенімділігі – үлкен сандар көбейткішін табу күрделілігінде. Егер криптосараптаушы n санын (ашық) көбейткіштерге жіктей алса, ол *p* және *q* мәнін таба алар еді, демек *z* санын да. Бұдан кейін *e* және *d* сандарын Евклид алгоритмінің көмегімен табуға болады. Қуанышқа орай, математиктер үлкен сандарды көбейткіштерге жіктеу мәселесін, кем дегенде 300 жыл шешуге тырысып келеді және осы уақыт ішінде жинақталған тәжірибе бұл мәселені тым қиын деп болжауға мүмкіндік береді.

Ривест (Rivest) және оның әріптестері, 500 цифрдан тұратын санды көбейткіштерге жіктеу үшін, дөрекі күшті қолданғанның өзінде, 1025 жыл қажет деп санайды. Ең жақсы танымал алгоритм және бір нұсқауды 1 мкс ішінде орындайтын компьютер қолданылады деп жорамалданады. Егер миллион чип бір мезгілде жұмыс істеп, бір нұсқауды 1 наносекунд ішінде орындайтын болса да 1016 жыл қажет болады. Тіпті, компьютер жылдамдығының экспоненциалды өсуін сақтағанның өзінде, 500 цифрдан тұратын санның көбейткіштерін табу үшін көптеген жылдар қажет болады, ал бұл уақытқа дейін біздің ұрпақтарымыз бұдан да үлкен p және d таңдай алады.

RSA алгоритмінің жұмысы *8.14-суретте* келтірілген. Бұл мысал үшін біз $p=3,\ q=11$ деп алдық, бұдан $n=33,\ z=20$ аламыз. 20 және 7 сандарының ортақ бөлгіштері жоқ болғандықтан, *d* санын 7-ге тең деп алуға болады. Осындай таңдаудан кейін e-ні $7e=1 \pmod{20}$ теңдеуін шешу арқылы табуға болады. Нәтижесінде $e=3$ болады. Шифрланған *C* мәтіні ашық P мәтінен $\boldsymbol{C = P^3 (\mathrm{mod}\ 33)}$ формуласы бойынша алынады. Қабылдаушы мәлімдемені $P=C7 \pmod{33}$ формуласы бойынша кері шифрлайды. Мысал ретінде суретте «SUZANNE» сөзін шифрлау көрсетілген.

Бұл мысал үшін таңдалып алынған қарапайым сандар тым кіші болғандықтан, *P* 33-тен кем болмауы тиіс. Нәтижесінде дара әліппелік алмастыру шифрын аламыз, ол аса таңқаларлықтай емес. Егер біз *p* және *q* сандарын 2^{512} ретіндегідей таңдаған кезде, *n* шамамен 21024 аумағында болар еді. Бұл жағдайда әр блокта 1024-ке дейін немесе DES немесе 16 AES шифрының сегіз разрядына қарсы 128 сегіз разрядты символ алар едік.

Ашық мәтін (P)		Шифрланған мәтін (C)			Кері шифрлаудан кейін	
Символ	Сан	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Символ
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
		Жөнелтуші есептеулері		Қабылдаушы есептеулері		

8.14-сурет. RSA алгоритмі жұмысының мысалы

RSA алгоритмін жоғарыда сипатталған түрде пайдалану, кірістегі бірдей блоктар шығыстағы бірдей блоктарға түрлендірілетін, симметриялы алгоритмді **ECB (Electronic Code Book – электронды шифроблокнот)** режимінде қолдануға сәйкес келеді. Осылайша, деректерді шифрлау үшін блоктарды қандай да бір түрде іліндіру қажет. Алайда, іс жүзінде ашық кілтті RSA алгоритмі тек бір реттік құпия кілтті тасымалдау үшін қолданылады. Бұдан кейін, AES немесе үштік DES тәрізді, қандай да бір симметриялы кілтті алгоритм қолданылады. Үлкен көлемдегі деректерді шифрлау үшін RSA жүйесі тым баяу жұмыс істейді, бірақ ол кілттерді тарату үшін кеңінен қолданылады.

8.3.2. Ашық кілтті басқа алгоритмдер

RSA алгоритмі кеңінен қолданыс тапса да, ол ашық кілті бар жалғыз танымал алгоритм емес. Алғашқы ашық кілтті алгоритм «арқаға асатын сөмке алгоритмі» болатын (Merkle Hellman, 1978). Алгоритмнің басты идеясы – әртүрлі салмақтағы көптеген объектілер бар. Осы объектілердің иесі, объектілер жиынын таңдап, оларды арқаға асатын сөмкеге салып, мәлімдемені кодтайды. Арқа қоржындағы объектілердің жалпы салмағы, мүмкін деген объектілердің тізімі және олардың әрқайсысының сәйкес салмағы бәріне белгілі. Объектілер тізімі рюкзактың ішінде құпия сақталған. Белгілі бір қосымша шектеулерде, мүмкін деген объектілер тізімін белгілі жалпы салмақ бойынша есептеп, анықтау мүмкін емес деп саналды, яғни шешімді тек объектілер тізімін әртүрлі сәйкестіктерді толық талдау арқылы ғана табуға болады. Сол себепті ол ашық кілт алгоритмі негізіне алынды.

Алгоритмнің құрастырушысы Ральф Меркле (Ralph Merkle) өз алгоритмін ешкім сындыра алмайтынына сенімді болғаны соншалық, оны сындырған кез келген адамға $100 сыйақы ұсынды. RSA тобындағы «S», Ади Шамир (Adi Shamir) оны бірден сындырып сыйақыны алды. Бұл Мерклені абыржытқан жоқ. Ол алгоритмді күшейтіп, оны сындырған адамға $1000 сыйақы тағайындады.

RSA тобындағы «R», Рон Ривест (Ron Rivest) алгоритмнің жақсартылған нұсқаны бірден сындырып, сыйақыны алды. Меркле келесі нұсқаға $10 000 тағайындауға батпады, сондықтан RSA тобындағы «А» Леонард Эйдлманның (Leonard Adleman) жолы болмады. Арқаға асатын сөмке алгоритмінің тағы да жақсартылғандығына қарамастан, ол сенімді деп саналмайды және сирек қолданылады.

Ашық кілтті басқа схемалар дискретті логарифмдерді есептеу күрелілігіне негізделген. Бұл қағиданы пайдаланатын алгоритмдерді Эль-Гамал (El Gamal, 1985) және Шнорр (Schnorr, 1991) құрастырған болатын.

Біршама басқа тәсілдер де бар, мысалы, элиптикалық қисықтарға негізделген (Menezes Vanstone, 1993). Алайда, негізгі екі категорияны үлкен сандар көбейткіштерін табу және дискретті лагорифмдер күрделілігіне негізделген алгоритмдер құрайды. Бұл есептер аса күрделі болып саналады, себебі математиктер көптеген жылдар бойына шешуге тырысып келеді, әзірше бәрі сәтсіз.

8.4. САНДЫҚ ҚОЛТАҢБАЛАР

Көптеген қағаз құжаттардың түпнұсқалығы (заңдық, қаржылық және т.б.) авторлық қолтаңбаның бар немесе жоқтығы арқылы анықталады. Фотокөшірмелер құжат деп саналмайды. Компьютерлік мәлімдемелер жүйесі қағаз және сиядан тұратын құжаттардың физикалық қозғалысын алмастыра алу үшін қолтаңба мәселесін шешу керек.

Қолмен қойылған қолтаңбаны алмастыруды құрастыру мәселесі өте күрделі. Іс жүзінде бір жақ екінші жаққа «қол қойылған» мәлімдемені жөнелте алатындай жүйе құрастыру керек, сонымен бірге:

1. қабылдаушы жөнелтуші тұлғаны тексере алатындай;
2. жөнелтуші кейіннен мәлімдеме мазмұнынан бас тарта алмайтындай;
3. қабылдаушы қабылданған мәлімдемені кейіннен өзгерте алмайтындай

болу керек.

Бірінші талап өте маңызды, мысалы, қаржылық жүйелер үшін. Клиент компьютері банк компьютеріне бір тонна алтын сатып алуға тапсырыс берген кезде, банк компьютері тапсырыс берген компьютердің шынымен де есеп-шотынан ақша алынатын компанияға тиесілі екенін тексере алуы керек. Басқа сөзбен айтқанда, банк клиенттің нақтылығын анықтай алуы керек (ал клиент банктің).

Екінші талап банкті алаяқтықтан қорғау үшін қажет. Айталық, банк бір тонна алтын сатып алды және осыдан кейін алтын бағасы бірден күрт түсіп кетті делік. Арсыз клиент кейіннен ол еш уақытта алтын сатып алуға тапсырыс жіберген жоқпын деп, банкті сотқа беруі мүмкін. Банк сотта мәлімдемені көрсете алады, бірақ клиент оны жөнелткенін мойындамайды. Осылайша, **ертеде берілген міндеттен бас тарта алмау (nonrepudiation)** талабы қамтама-

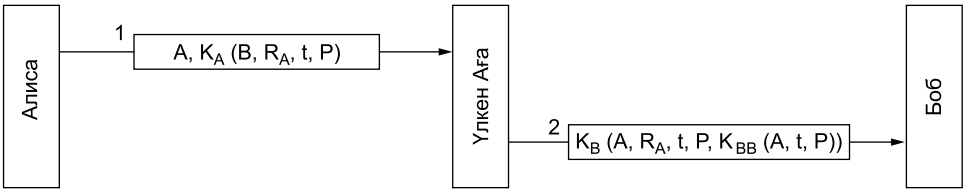
сыз етілу керек. Біз төменде қарастыратын, сандық қолтаңбаны құрастыру тәсілдері осыған да арналған.

Үшінші талап, алтын сатып алғаннан кейін оның бағасы жоғары өсіп кеткен кезде, банк бір тонна алтынның орнына бір құйма сатып алуға тапсырыс берілді деп, қол қойылған мәлімдемені өзгертпек болған жағдайда, клиентті қорғау үшін керек. Осындай сценарий жағдайында банк, алааяқтыққа барып, алтынның қалған бөлігін өзіне алып қалады.

8.4.1. Симметриялы кілтті қолтаңбалар

Сандық қолтаңбаларды жүзеге асырудың бір тәсілі – барлығы бірдей сенім артатын, қандай да бір беделді ұйым құрастыру, мысалы, оны Үлкен Аға (Big Brother, BB) деп аталық. Сонан кейін әр тұтынушы құпия кілтті таңдап, оны Үлкен Аға офисына өзі апарып береді. Осылайша, мысалы, Алисаның құпия кілті, K_A Алисаға және Үлкен Ағаға ғана белгілі.

Алиса өз банкирі Бобқа ашық мәтінмен P қол қойылған мәлімдемені жөнелтпек болған кезде, ол K_A-мен (Алиса кілтімен) $K_A(B, R_A, t,)$ шифрланған мәлімдемесін құрастырады. Мұндағы B – Боб идетификаторы, R_A – Алиса таңдап алған кездейсоқ сан, t – мәлімдеменің жаңадан құрастырылғандығын растайтын уақыт шампы. Содан кейін, Алиса мәлімдемені *8.15-суретте* көрсетілгендей, Үлкен Ағаға жөнелтеді. Үлкен Аға мәлімдеменің Алисадан келгенін көріп, оны кері шифрлап Бобқа жөнелтеді. Бобқа жөнелтілген мәлімдеме Алисаның ашық мәтінінен және Үлкен Аға қолтаңбасынан тұрады $K_{BB}(A, t, P)$. Қол қойылған мәлімдемені алғаннан кейін Боб Алиса тапсырысын орындауға кіріседі.



8.15-сурет. Үлкен Ағаның сандық қолтаңбасы

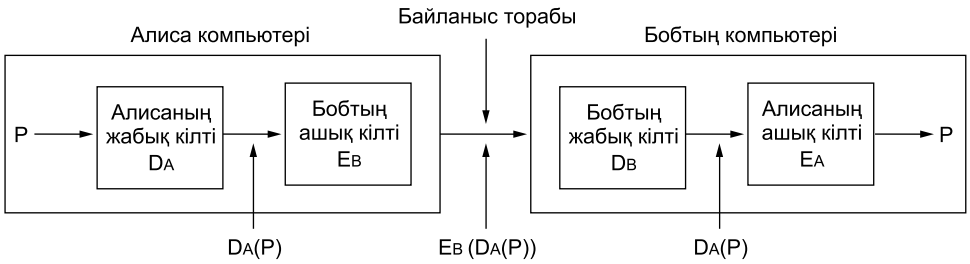
Алиса кейіннен мәлімдеме жөнеткендігінен бас тартса не болады? Әрине, мұндай жанжал орын алса, жанжалдасушылар бір-бірін сотқа береді (кем дегенде Америка Құрама Штатттарында). Соңында, іс сотқа келген кезде, Алиса бар күшін салып, Бобқа мәлімдеме жөнелтпегенін айтып бағады. Сот Бобтан бұл мәлімдеменің алааяқ Трудидан емес, Алисадан келгендігіне неліктен сенімді екенін сұрайды. Боб алдымен, Үлкен Ағаның мәлімдеме K_A Алиса кілтімен шифрланбаған жағдайда оны қабылдамайтынын айтады, ал алааяқтың Үлкен Ағаға Алисадан жалған мәлімдеме жоғалту мүмкіндігі жоқ.

Содан кейін, Боб сотқа мәлімдемені көрсетеді $A: K_{BB}(A,t,P)$. Боб мәлімдемеге Үлкен Аға қол қойғандығын айтады, бұл P мәлімдемесін Бобқа Алисаның жөнелткендігін дәлелдейді. Судья Үлкен Ағаны (барлығы сенетін) осы мәлімдеме соңындағы қолтаңбаны тексеруді сұрайды. Үлкен Аға Бобтың шындықты айтып тұрғанын растағаннан кейін, судья істі Бобтың пайдасына шешеді. Іс жабылады.

Сандық қолтаңба хаттамасына байланысты *8.15-суретте* көрсетілген мәселе теорияда зиянкес екі мәлімдемені де қайта қалпына келтірген жағдайда орын алуы мүмкін. Бұндай ықтималдықты азайту үшін уақыт штампы қолданылады. Одан басқа, Боб жақын арада келген барлық басқа мәлімдемелерді қарап, олардың ішінде осындай R_A-ның бар екендігін тексере алады. Егер мұндай мәлімдеме бар болса, онда мәлімдеме көшірме болып саналады және еленбейді. Уақыттық штампы бойынша өте ескі мәлімдемелер де еленбейді. Қайта қалыпқа келтіру арқылы әп-сәттік шабуылдан қорғану үшін Боб әрбір келген мәлімдемедегі R_A – кездейсоқ сан мәнін тексереді және соңғы сағатта келген осындай сандарды есте сақтайды. Егер соңғы сағат ішінде мұндай мәлімдеме алынбаса, онда Боб келген мәлімдеменің жаңа тапсырыс екеніне сенімді бола алады.

8.4.2. Ашық кілтті қолтаңбалар

Сандық кілттер үшін симметриялы кілтпен шифрлаудың басты мәселесі – барлығы Үлкен Ағаға сенуге келісім беруі тиіс. Бұдан басқа, Үлкен Аға өзі қол қоятын барлық мәлімдемелерді оқуға мүмкіндік алады. Үлкен Аға серверін басқаруға логикалық тұрғыдан ең сенімді кандидат – үкімет, банк немесе нотариалды бюро. Алайда бұл ұйымдарға барлығы бірдей сенбейді. Осылайша, электронды құжатқа қолтаңба алу үшін беделді сенімді ұйым қажет болмағаны дұрыс болар еді.



8.16-сурет. Ашық кілт көмегімен шифрлау арқылы алынған сандық қолтаңба

Қуанышқа орай, бұл жерде ашық кілтпен шифрлау көмекке келеді. Айталық, ашық кілтпен шифрлау және кері шифрлау алгоритмінің әдеттегі *D(E(P))=P* қасиетінен басқа, *E(D(P))=P* қасиеті бар делік. Мысалы, RSA-ның осындай қасиеті бар, сондықтан мұндай болжам бос сөз емес. Бұл жағдайда Алиса Бобқа

Р қол қойылған ашық мәлімдемені $\boldsymbol{E_B\left(D_A\left(P\right)\right)}$ жөнелту арқылы жібере алады. Алиса өзінің $\boldsymbol{D_A}$ кері шифрлау (жабық) кілтін, сонымен бірге Бобтың $\boldsymbol{E_B}$ ашық кілтін білетіндігіне назар аударыңыздар. Сондықтан ол мұндай мәлімдемені құрастыра алады.

Мәлімдемені қабылдап, Боб оны әдеттегідей өзінің $\boldsymbol{D_B}$ жабық кілтін пайдаланып кері шифрлайды және нәтижесінде *8.16-суретте* көрсетілгендей $\boldsymbol{D_A\left(P\right)}$ мәлімдемесін алады. Ол бұл шифрланған мәлімдемені сенімді жерде сақтайды, содан кейін оны Алисаның $\boldsymbol{E_A}$ шифрлау кілтімен кері шифрлап, ашық мәтінді алады.

Бұл жағдайда, сандық қолтаңбаның қалай жұмыс істейтіндігін түсіну үшін Алиса кейіннен Бобқа *P* мәлімдемесін жөнелткендігінен бас тартты делік. Іс сотқа жеткен кезде, Боб судьяға *P* және $D_A\left(P\right)$ көрсетеді. Судья Бобта шын мәнінде $\boldsymbol{D_A}$ кілтімен шифрланған *P* мәлімдеменің бар екендігіне, мәлімдемеге E_A кілтін қолданып көз жеткізеді. Боб Алисаның жабық кілтін білмейді, демек, осы кілтпен шифрланған мәлімдемені ол тек Алисадан ала алады. Өтірік айтқаны және алааяқтығы үшін түрмеде отырып, Алиса ашық кілті бар жаңа қызықты алгоритмдерді құрастырумен айналыса алады.

Ашық кілтті пайдаланып шифрлау схемасы әдемі болғанына қарамастан, оның алгоритмнің өзімен емес, жұмыс істейтін ортаға қатысты елеулі кемшілігі бар. Біріншіден, Боб мәлімдеменің Алисадан келгендігін тек D_A кілті құпия болып қалған жағдай да ғана дәлелдей алады. Егер Алиса өзінің құпия кілтін ашық жариялап қойса, онда бұл аргумент сенімді болудан қалады, себебі мәлімдемені кез келген адам жөнелтуі мүмкін, тіпті Бобтың өзі де.

Мысалы, егер Боб Алисаның биржалық брокері болса да мәселе туындауы мүмкін. Алиса Бобтан бірнеше акция сатып алуға тапсырыс береді. Акциялар сатып алынғаннан кейін олардың құны бірден түсіп кетеді. Бобқа жөнелткен өз мәлімдемесінен бас тарту үшін Алиса полицияға үйіне ұрылар түсіп, құпия кілт жазылған компьютер ұрланғанын хабарлайды. Алиса тұратын елдің немесе штаттың заңына байланысты, ол заң алдында жауапты немесе жауапты емес деп табылуы мүмкін. Әсіресе, егер ол үйінің ұрланғандығын жұмыстан келгеннен кейін, ұрлық жасалғаннан бірнеше сағаттан өткен кезде хабарласа.

Бұл сандық қолтаңба схемасының басқа бір мәселесі, егер Алиса өз кілтін ауыстырғысы келсе орын алуы мүмкін. Мұдай әрекет заңды, одан бетер кілтті дүркін-дүркін ауыстыру оның жоғары сенімділігіне кепілдік береді. Бұл жағдайда, егер іс сот қарауына жететін болса, судья $\boldsymbol{D_A\left(P\right)}$ қолтаңбасына ағымдағы E_A кілтін қолданбақ болады және нәтижесінде *P* мәлімдемесін ала алмайды. Бұл жағдайда Боб ақымақ болып көрінеді.

Іс жүзінде сандық қолтаңба үшін ашық кілті бар кез келген алгоритмді пайдалануға болады. RSA алгоритмі іс жүзінде өндірістік стандарт болды. Ол қауіпсіздікті қамтамасыз етуге негізделген көптеген программаларда пайдаланылады. Алайда, 1991 жылы NIST жаңа Сандық қолтаңба стандарты үшін ашық кілтті Эль-Гамал алгоритмінің **DSS (Digital Signature Standard)** нұсқасын пайдалануды ұсынды. Эль-Гамал алгоритмінің бұл нұсқасы үлкен

сандарды көбейткіштеге жіктеу күрделілігіне емес, дискретті логарифмдерді есептеуге негізделген.

Әдеттегідей, үкіметтің жаңа криптографиялық стандартты зорлап міндеттеуі үлкен шу тудырды. DSS стандартын:

1. шамадан тыс құпиялығы (Эль-Гамал алгоритмін пайдаланатын хаттама АҚШ Ұлттық қауіпсіздік агенттігі құрастырған болатын);
2. тым баяулығы (қолтаңбаны тексеру үшін RSA алгоритмінен 10-нан 40 есеге дейін баяу);
3. тым жаңа (Эль-Гамал алгоритмі жеткілікті түрде тексерілмеген болатын);
4. тым сенімсіз (кілттің бекітілген ұзындығы 512 бит) болғандығы үшін сынаққа алынды.

Келесі өңдеулерден кейін наразылықтың төртінші пункті даулы болды, себебі ұзындығы 1024 бит кілтті пайдалануға рұқсат берілді. Алайда алғашқы екі пункт бойынша наразылық осы күнге дейін сақталуда.

8.4.3. Мәлімдемелер пішіні

Сандық қолтаңбалардың көптеген тәсілдері онда екі түрлі функцияның қатар қолданылуы үшін сынға алынады: сәйкестендіру және құпиялық. Көптеген жағдайда құпиялықсыз, тек сәйкестендіру қажет болады. Оның үстіне, мысалы, егер жүйе құпиялықсыз, тек сәйкестендіруді қамтамасыз ететін болса, сыртқа шығаруға лицензия алу әлдеқайда жеңіл. Төменде бүкіл мәлімдемені шифрлауды қажет етпейтін сәйкестендіру схемасы сипатталады.

Бұл схема кірісте еркін ұзындықтағы ашық мәтін бөлігін қабылдайтын және сол бойынша бекітілген ұзындықтағы биттер жолын есептейтін, қайтымсыз хэш-функция идеясына негізделген. Жиі **мәлімдеме пішіні (message digest, MD)** деп аталатын бұл хэш-функцияның келесідей төрт маңызды қасиеті бар.

1. Берілген P ашық мәтіні бойынша *MD(P)* хэш-функциясының мәнін есептеу жеңіл.
2. Іс жүзінде *MD(P)* сандық қолтаңба бойынша P ашық мәтіннің мәнін анықтау мүмкін емес.
3. Іс жүзінде берілген P үшін $MD(P') = MD(P)$ теңдігі орындалатын P' таңдап алу мүмкін емес.
4. Кіріс тізбектің бір битінің өзгеруі мүлдем ұқсас емес нәтижеге әкеледі.

Үшінші талапты қанағаттандыру үшін хэш-функция нәтижесінің ұзындығы, кем дегенде 128 бит болу керек, тіпті одан да ұзын болғаны жарамды. Төртінші талапты қанағаттандыру үшін хэш-функция кіріс мәндерді өте қатты

бұрмалауы керек. Осы тұста бұл тәсіл біз жоғарыда қарастырған симметриялы кілтті алгоритмдерді еске салады.

Ашық мәтін жағынан мәлімдеме пішіні барлық мәлімдеменің ашық кілтті алгоритм көмегімен шифрлануына қарағанда жылдам есептеледі. Сондықтан мәлімдеме пішіндері сандық қолтаңбалар алгоритмі жұмысын жылдамдату үшін қолданылуы мүмкін. Бұның барлығының қалай жұмыс істейтінін түсіну үшін *8.15-суретте* көрсетілген сандық қолтаңбаны тасымалдау хаттамасын тағы бір қарастырайық. Р ашық мәтінін $K_{BB}(A, t, P)$ бірге жөнелткеннің орнына Үлкен Аға енді Р ашық мәтініне *MD* хэштеу функциясын пайдаланып, мәлімдеме пішіні *MD(P)*-ны есептейді. Сонан кейін ол $K_{BB}(A, t, MD(P))$-ны бесінші элемент ретінде тізімге орналастырып, $\boldsymbol{K_B}$ кілтімен шифрлайды да, оны $K_{BB}(A, t, P)$ орнына Бобқа жөнелтеді.

Дау орын алған жағдайда Боб сотта Р ашық мәтінін де, $K_{BB}(A, t, MD(P))$-ны да көрсете алады. Судьяның сұранысы бойынша Үлкен Аға $K_{BB}(A, t, MD(P))$-ны кері шифрлайды, нәтижесінде сотқа түпнұсқа екендігін Үлкен Аға дәлелдеген *MD(P)* сандық қолтаңба және ашық мәтін ұсынылады. Іс жүзінде осы сандық қолтаңбаға сәйкес басқа ашық мәтін құрастыру мүмкін емес болғандықтан, сот Бобтың шындықты айтып тұрғанына сенеді. Мәлімдеме пішінін пайдалану шифрлау уақытын және сақтау мен транспорттауға кететін шығынды үнемдейді.

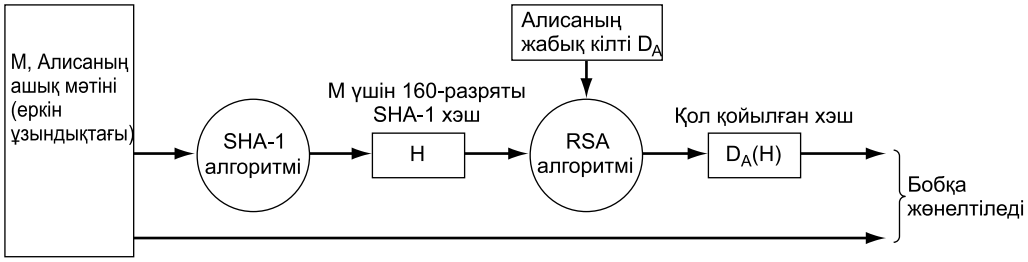
8.17-сурет. Мәлімдеме пішіні қолданылған сандық қолтаңба

Мәлімдеме пішіні сонымен бірге, *8.17-суретте* көрсетілгендей, ашық кілтті шифрлау жүйелерінде мәлімдемені желі бойымен тасымалдау кезінде, мәлімдеме сақтығының кепілдігі ретінде де қолданылады. Мұнда Алиса алдымен өз ашық мәтіні үшін мәлімдеме пішінін есептейді. Содан кейін ол пішінге қол қойып, шифрланған мәлімдеме пішінін және ашық мәтінді Бобқа жөнелтеді. Егер жол бойында Р ашық мәтінін зиянкес ауыстырмақ болса, Боб мәлімдеме пішіні *MD(P)* есептеп, оны бірден сезеді.

SHA-1

Мәлімдеме пішінін есептейтін функцияның бірнеше нұсқасы ұсынылды. Олардың ішінде кең таралғаны **SHA-1 (Secure Hash Algorithm 1 – сенімді хэштеу алгоритмі)** *(NIST, 1993)*. Барлық мәлімдеме пішіні тәрізді ол кіріс биттерді күрделі жолмен араластырады, сондықтан әр шығыс биті әрбір кіріс

битіне тәуелді. SHA-1 АҚШ Ұлттық қауіпсіздік агенттігі құрастырған болатын және NIST батасын алды (FIPS 180-1 федералды стандартында ұласқан). SHA-1 алгоритмі кіріс деректерді 512 биттік блоктармен өңдейді және ұзындығы 160 бит мәлімдеме пішінін құрастырады. Алисаның құпия емес, бірақ қол қойылған мәлімдемені Бобқа жөнелткен әдеттегі жағдайы *8.18-суретте* көрсетілген. Ашық мәтін SHA-1 алгоритмімен өңделеді, шығыста 160-биттік хэш SHA-1 алынады. Оған Алиса қол қояды *(RSA жабық кілтімен)* және ашық мәтінмен бірге Бобқа жөнелтеді.

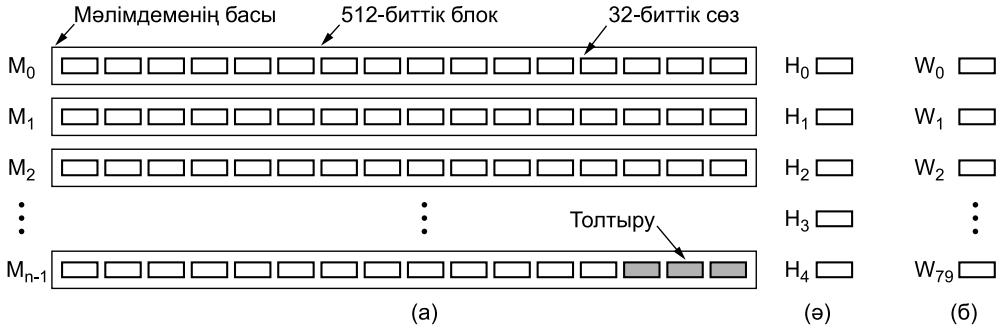


8.18-сурет. SHA-1 және RSA құпия емес мәлімдемелер қолтаңбасын құрастыру үшін пайдалану

Боб мәлімдемені алған кезде SHA-1 көмегімен хэш-функцияны өзі есептейді және Алисаның ашық кілтін қол қойылған хэшке H бастапқы хэшті алу үшін қолданады. Егер олар сәйкес келсе, мәлімдеме дұрыс деп саналады. Труди мәлімдеме ұстап алып, оны H мәні бақылау мәнмен сәйкес келетіндей етіп өзгерте алмайды, Боб Труди жасаған алмастырулардың барлығын бірден біледі. Дербес құқықтығы маңызды, ал құпиялықтың мәні жоқ мәлімдемелер үшін *8.18-суретте* келтірілген схема жиі қолданылады. Есептеуге кететін аздаған шығындармен ол жол бойында енгізілген кез келген өзгертулер жоғары ықтималдықпен анықталады деген кепілдік береді.

Енді SHA-1 қалай жұмыс істейтінін қысқаша қарастырып кетейік. Алдымен SHA-1 алгоритмі мәлімдеменің соңына бірлік битті қосады, содан кейін, нәтижесінде биттердің жалпы саны 512-ге еселі болатындай нөлдік биттер қосылады (кем дегенде, 64 тең). Содан кейін, мәлімдеме ұзындығы жазылған (биттік қосымшаға дейінгі) 64-разрядты сан, 64 кіші биттермен логикалық қосылады (НЕМЕСЕ операциясы). *8.19 а-суретінде* оң жағында қосымшасы бар мәлімдеме көрсетілген, себебі ағылшын мәтіні және суреттер оңнан солға қарай оқылады (яғни, суреттің оң жақ шекарасы оның соңы ретінде қабылданады, ал сол жағы – басы). Есептеу техникасына қатысты алғанда мұндай орналасу байттардың сақталуының кері орналасуына сәйкес келеді (алдымен ең маңызды үлкен биттер беріледі). Мұндай жүзеге асырулар, мысалы, SPARC және IBM 360 және олардың ізбасарларына тән. Алайда қолданылатын техникаға тәуелсіз, SHA-1 биттік қосымшаларды мәлімдеменің соңына орналастырады.

Есептеулерді орындау кезінде SHA-1 бес 32-биттік айнымалылармен $(H_0 ... H_4)$ жұмыс істейді. Бұл айнымалыларда хэш-функция мәндері жинақталады. Олар *8.19 ә-суретінде* көрсетілген. Олардың бастапқы мәндері – стандартпен анықталған тұрақты шамалар.



8.19-сурет. 512 битке еселі ұзындыққа дейін толықтырылған мәлімдеме (а); шығыс айнымалылар (ә); сөздер массиві (б)

Содан кейін M_0 блоктан M_{n-1} блокқа дейін кезек-кезек өңделеді. Ағымдағы блок үшін алдымен 16 сөз, *8.19 б-суретінде* көрсетілгендей, мөлшері 80 сөз *W* көмекші массивінің басына көшіріледі. Қалған 64 сөз келесі формулаларды пайдалану арқылы есептеледі:

$$W_i = S^1\left(W_{i-3}\ XOR\ W_{i-8}\ XOR\ W_{i-14}\ XOR\ W_{i-16}\right)\left(16 \le i \le 79\right)$$

мұндағы $S^b(W)$ 32-разрядты *W* сөзінің *b* битке бұрылуын білдіреді. Енді $(H_0 ... H_4)$ мәндері бойынша A-дан E-ге дейінге айнымалылар инициализацияланады.

Есептеулердің өзін жалған-С келесідей жазуға болады:

```
for (i-0; i<80; i++) {
    temp = S5(A) + fi(B, C, D)+E +Wi +Ki;
    E=D; D=C; C=S30(B); A=temp;
}
```

мұндағы тұрақты K_i стандартпен анықталады. Араластырушы f_i функциясы келесідей жолмен беріледі:

$$f_i(B, C, D) = (B\ AND\ C)OR(NOT\ B\ AND\ D) \qquad (0 \le i < 19)$$

$$f_i(B, C, D) = B\ XOR\ C\ XOR\ D \qquad (20 < i < 39)$$

$$f_i(b, C, D) = (B\ AND\ C)OR(B\ AND\ D)OR(C\ AND\ D) \qquad (40 < i < 59)$$

$$f_i(B, C, D) = B\ XOR\ C\ XOR\ D \qquad (60 < i < 79)$$

Циклдың 80 итерациясынан кейін A ... E айнымалыларының мәндері сәйкесінше $(H_0 ... H_4)$ қосылады.

Алғашқы 512-биттік блокты өңдегеннен кейін келесіне өңдеу басталады. *W* массиві жаңа блоктың көмегімен қайта инициализацияланады, алайда H өзгеріссіз сақталады. Бұл блокты өңдеу аяқталғаннан кейін, келесі блокты өңдеу басталады және т.с.с. мәлімдеменің барлық 512-разрядты блоктары осы кастрюлге түскенше. Соңғы блок өңделгеннен кейін H массивіндегі 32-разрядты сөз, 160-биттік криптографиялық хэш-функцияның мәні ретінде шығарылады. SHA-1 толық коды RFC 3174 келтірілген.

Қазіргі кезде SHA-1 жаңа нұсқаларымен жұмыс жүргізіліп жатыр, 224, 256, 384- және 512-разрядты хэш-функция мәндерімен. Бұл нұсқалар бірге SHA-2 деп аталады. SHA-1-ден олардың хэштері ғана ұзын емес, сонымен бірге SHA-1 кейбір әлсіз тұстарынан құтылу үшін пішіндер де өзгертілді. SHA-2 әзірше кеңінен қолданылмайды, бірақ ол болашақта орын алуы мүмкін.

MD5

Суреттеуіміз толық болу үшін біз тағы бір пішінді қарастырамыз. **MD5 (Message Digest 5 – мәлімдеме пішіні 5)** алгоритмі Рональд Ривест (Ronald Rivest) құрастырған хэш-функция алгоритмінің бесінші нұсқасы болып саналады. Алдымен мәлімдеме 448 бит ұзындыққа дейін толықтырылады (модуль 512 бойынша). Содан кейін оған 64-разрядты сан ретінде қарастырылатын, мәлімдеменің бастапқы ұзындығы қосылады. Нәтижесінде ұзындығы 512 еселі биттер блогы алынады. Есептеуге дайындықтың соңғы қадамында 128-разрядты буфер инициализацияланады. Оның мәні қандай да бір тұрақты мәнге тең етіп алынады.

Содан кейін есептеу басталады. Әр қадамда кіріс мәтіннің 512-разрядты блогы алынып, 128-разрядты буфермен мұқият араластырылады. Тіпті, шығымды болу үшін кастрюлге синустар кестесінің мазмұны қосылады. Синустар олардың нәтижесі кездейсоқ сандар генераторы (жиі тригонометриялық функция қолданылатын) нәтижесіне қарағанда кездейсоқтау болғандықтан емес, кейіннен құрастырушы (тапсырыс беруші) кіре алатын, жасырын айла құрастырылды деген күмән болмас үшін. Үдеріс барлық кіріс блоктар өңделіп біткенше жалғасады. 128-разрядты буфер мазмұны мәлімдеме пішінін құрайды.

Он жылдан артық қолданылып, зерттелгеннен кейін MD5 сәйкестікте анықталды, яғни әртүрлі мәлімдемелер бірдей хэшпен (Sotirov және басқалар, 2008). Бұл пішіндер функциясы үшін қорқынышты нышан, себебі ол функция мәлімдемерлерді ұсыну үшін қауіпсіз қолданыла алмайды дегенді білдіреді. Осылайша, қауіпсіздік бірлестігі MD5 сынған деп есептейді. Оны мүмкін болған жердің барлығында ауыстыру керек және жаңа жүйелер оны өз құрылымының бір бөлігі ретінде қолданбауы керек. Осыған қарамастан, осы кітап жазылған уақытқа дейін бұл алгоритм әлі де қолданыста.

8.4.4. Туған күндер жайлы есептер

Шифрлар әлемінде барлығы бір қарағандағыдай емес. Мысалы, *m* разрядтан тұратын мәлімдеме пішінін бұзу үшін үшін операция қажет деп жорамалданады. Ал, іс жүзінде, егер жарияланған және классикалық болып кеткен **туған күндерге негізделген есепті (birthday attack)** пайдалансақ, онда $2^{m/2}$ операция жеткілікті болады (Yuval, 1979).

Бұл тәсілдің негізінде математика профессоры ықтималдықтар теориясы курсы бойынша жиі мысал ретінде келтіретін есеп жатыр. Сұрақ: туған күндері сәйкес келетін адамдар саны ½-ден асу үшін класта неше студент болуы керек? Көптеген студенттер бұл мән 100-ден көп болу керек деп күтеді. Іс жүзінде ықтималдықтар теориясы бұл сан 23-ке тең болуы керек деп санайды. Бұл мәселенің егжей-тегжейіне тоқталмай біз интуитивті түсінікті жауап береміз: 23 адамнан әрқайсысының туған күні 1/365 ықтималдықпен сәйкес келетін, $(23 \times 22)/2 = 253$ әртүрлі жұптар құрастыруға болады. Енді бұл жауап ғажайып емес болып көрінеді.

Жалпы жағдайда, егер n кіріс (адамдар, мәлімдемелер және т.б.) және k мүмкін шығыстар (туған күндер, мәлімдеме пішіні және т.б.) арасында сәйкестік бар болса, біз $n(n-1)/2$ кіріс жұбын аламыз. Егер $n(n-1)/2 > k$ болса, онда әртүрлі кірісте кем дегенде бір сәйкес шығыстың болу ықтималдығы тым үлкен. Осылайша, пішіндері бірдей екі мәлімдеменің кездесу ықтималдығы $n > \sqrt{k}$ болғанның өзінде жоғары болады. Бұл 64-разрядты мәлімдеме пішіні үлкен ықтималдықпен сындыруға болады (яғни, пішіндері бірдей екі түрлі мәлімдеме табуға болады) дегенді білдіреді.

Іс жүзіндегі мысалды қарастырайық. Мемлекеттік университеттің компьютерлік ғылымдар кафедрасында вакансия және осы орынға екі кандидат пайда болды, Том және Дик. Том факультетте Диктен екі жыл ұзағырақ жұмыс істейді, сондықтан оның кандидатурасы бірінші қарастырылады. Егер ол бұл лауазымға ие болса, демек Диктің жолы болмағаны. Том, кафедра меңгерушісі Мэрилин оның жұмысын жоғары бағалайтынын біледі, сондықтан ол Мэрилиннің істі қарайтын факультет деканына ұсыныс хат жазуын сұрайды. Жөнелтілгеннен кейін хаттың барлығы құпия болады.

Мэрилин өз хатшысы Эленге хатты жазуды тапсырады және хатта қандай ақпараттың болу керектігін айтады. Хат дайын болған кезде, Мэрилин оны қарап, хаттың 64-разрядты пішінін есептеп, қол қойып, факультет деканына жөнелтеді. Кейіннен Элен хаттың өзін электронды пошта арқылы жөнелте алады.

Томның сорына қарай, Элен және Дик арасында роман бар еді және Элен Томды алдамақ болады. Сондықтан ол тік жақшада 32 нұсқасы бар келесідей хатты жазады.

Құрметті декан,
Бұл [*хат* | *үндеу*] менің [*осы күнде* | *осы жылы*] профессор лауазымына

[*кандидат* | *таласушы*] профессор Том Уилсон жайлы [*шын* | *ашық*] пікірімді білдіреді. Мен профессор Уилсонмен алты жыл [*дерлік* | *шамамен*] бойында [*таныспын* | *жұмыс істеймін*]. Ол [*көрнекті* |*өте жақсы*] зерттеуші, [*үлкен талант* | *үлкен мүмкіндіктер*] иесі және [*бүкіл әлемде* | *тек біздің елде ғана емес*] [*көптеген* | *үлкен аумақтағы*] [*күрделі* | *келешегі бар*] сұрақтарға өзінің [*шыншыл* | *жасампаз*] көзқарасымен танымал.

Ол сонымен бірге, [*жоғары* | *тіпті*] [*қадірлі* | *бағалы*] [*оқытушы* | *педагог*] болып саналады. Оның студенттері оның [*сабақтарына* | *дәрістеріне*] [*өте жақсы* | *ең жоғары*] баға береді. Ол [*біздің кафедраның* | *біздің университеттің*] [*ең танымал* | *ең сүйікті*] [*оқытушысы* | *мұғалімі*].

[*Сонымен бірге* | *бұдан басқа*] профессор Уилсонның [*гранттары* | *контракттары*] біздің кафедраның [*фондын* | *қаржы қорын*] [*татымды* | *едәуір*] толтырды. Бұл [*ақшалай* | *қаржылық*] қорлар [*көптеген* | *бірнеше*] [*маңызды* | *арнайы*] бағдарламаларды [*орындауға* | *жүзеге асыруға*] [*жағдай жасады* |*мүмкіндік берді*], мысалы, [*келесідей* | *олардың ішінде*] 2000 жылдың мемлекеттік бағдарламасы бар. Бұл қаражаттарсыз біз үшін [*маңызды* | *елеулі*] бағдарламаны жалғастыру мүмкін емес еді. Мен бұл лауазымды оған беруіңізді сұраймын.

Томның сорына қарай, Элен бұл хатты аяқтасымен екінші хатты жазуға кіріседі:

Құрметті декан,

Бұл [*хат* | *үндеу*] менің [*осы күнде* | *осы жылы*] профессор лауазымына [*кандидат* | *таласушы*] профессор Том Уилсон жайлы [*шын* | *ашық*] пікірімді білдіреді. Мен профессор Уилсонмен алты жыл [*дерлік* | *шамамен*] бойында [*таныспын* | *жұмыс істеймін*]. Ол [*әлсіз* | *жеткілікті дәрежеде талантсыз*] [*зерттеуші* | *ғалым*], өзі айналысатын салада мүлдем танымал емес. Оның жұмыстарында іс жүзінде қазіргі уақыттың [*кілтті* | *басты*] [*мәселелерін* | *сұрақтарын*] түсінушілік жоқтың қасы.

[*Одан бетер*| *Бұдан басқа*], ол қандай да бір [*қадірлі* | *бағалы*] [*оқытушы* | *педагог*] емес. Оның студенттері оның [*сабақтарына* | *дәрістеріне*] [*ең төмен* | *жағымсыз*] баға береді. Ол біздің кафедраның ең танымал емес [*оқытушысы* | *мұғалімі*]. Ол өз [*дәрісінде* | *сабағында*] сұрақ қоюға батылы жеткен студенттерді [*күлкі етін* | *ыңғайсыз жағдайға қалдыратын*] [*әдетімен* | *бейімділігімен*] [*атағы шыққан* | *қайғылы танымал*].

[*Сонымен бірге* | *бұдан басқа*] профессор Уилсонның [*гранттары* | *контракттары*] біздің кафедраның [*фондын* | *қаржы қорын*] [*мүлдем* | *іс жүзінде*] толтырмайды. Егер жаңа қаржы көзін таппасақ біз [*көптеген* | *бірнеше*] [*маңызды* | *арнайы*] бағдарламаларды [*жабуға* | *тоқтатуға*] мәжбүр боламыз, мысалы, [*келесідей* | *олардың ішінде*] 2000 жылдың мемлекеттік бағдарламасы бар. Өкінішке орай, мұндай [*жағдайда* | *шарттармен*] мен оны сізге бұл лауазымға [*ұсына алмаймын* | *ұсынбаймын*].

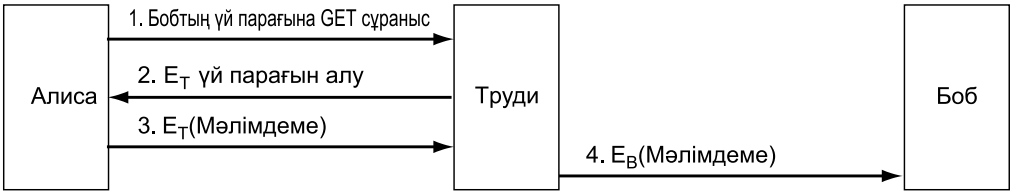
Сонан кейін Элен өз компьютерін екі хаттың әр нұсқасы үшін 2^32 мәлімдеме пішінін есептуге мәжбүрлейді және бұған бір түн уақыт кетеді. Бірінші хаттың бір пішіні екінші хаттың бір пішінімен сәйкес келеді деген үміт бар. Егер сәйкес келмесе, онда Элен тағы бірнеше сөздер және сөз тіркестер нұсқасын қоса алады және демалыс күні тағы да бақ сынап көреді. Айталық, ол осындай сәйкестікті тапты делік. Оң пікірді А хаты, ал теріс пікірді – В хаты деп атайық.

Элен А хатын бекіту үшін Мэрилинге электронды пошта арқылы жөнелтеді. Мэрилин, әрине, хатты бекітіп, 64-разрядты мәлімдеме пішінін есептеп, оған қол қояды да, қол қойылған пішінді факультет деканына жөнелтеді. Элен В хаты тәуелсіз жөнелтеді (жөнелту керек боған А хатының орнына).

Хатты және қол қойылған мәлімдеме пішінін алып, декан В хаты үшін есептеу алгоритмін іске қосады да, пішінінің Мэрилен жіберген хатпен сәйкес келгенін көріп Томды жұмыстан шығарып жібереді. Ол Эленнің мәлімдеме пішімдері бірдей екі хат құрастырып, Мэрилин оқып қол қойған нұсқадан басқасын жөнелткенін білмейді де. (Арты жақсы аяқталған нұсқа: Элен Дикке өзінің хатты алмастырғанын айтады. Ашуланған Дик Эленмен байланысын үзеді. Элен Мэрилинге бәрін айтып, кешірім сұрайды. Мэрилин деканға қоңырау шалады. Соңында Том профессор лауазымына ие болады.) SHA-1 алгоритмін пайдаланған кезде шифрға бұлай шабул жасауды жүзеге асыру өте күрделі, себебі тіпті компьютер секундына 1 трлн пішімдер есептей алса да, екі хат үшін 2^{80} нұсқадан талдап шығу үшін 3200 жылдан аса уақыт қажет болады және ол 100% сәйкестікке кепілдік бермейді. Әрине, егер 1 млн компьютерді параллель жұмыс істеуге мәжбүр етсек, 3200 жылдың орнына 2 апта қажет болады.

8.5. АШЫҚ КІЛТТЕРДІ БАСҚАРУ

Криптографияда ашық кілттерді пайдалану, алдын ала жалпы кілтті болмаса да, сонымен бірге электронды қолтаңбаны үшінші сенімді жақты тартпай құрастырып, құпия деректерді тасымалдауға мүмкіндік береді. Соңында, қол қойылған мәлімдемелер пішіні қабылдаушыға алынған деректердің тұтастығы және сәйкестігін жеңіл және сенімді тексеруге мүмкіндік береді.



8.20-сурет. Ашық кілттер жүйесіне басып ену амалы

Алайда, бір сұрақты жеңіл айналып өттік: егер Алиса және Боб бір-бірін танымаса, олар хабарласу алдында ашық кілтпен қалай алмаса алады? Ашық кілтті веб-сайтта орналастыру – ең тиімді шешім тәрізді болып көрінеді. Алайда бұлай істеуге болмайды, себебі: айталық, Алиса Бобтың ашық кілтін оның веб-сайтынан таппақ болады. Ол оны қалай жүзеге асырады? Сайттың URL-адресін тереді. Браузер Бобтың үй парағының DNS-адресін іздеп, оған *8.20-суретте* көрсетілгендей *GET* сұранысын жөнелтеді. Өкінішке орай, Труди осы сәтте сұранысты ұстап алып, Алисаға жалған веб-парақ адресін жөнелтеді. Жалған веб-парақ ретінде, мысалы, Боб үй-парағының көшірмесі алынып, ол жерге Трудидің ашық кілті орналастырылуы мүмкін. Алиса өзінің бірінші мәлімдемесін көмегімен шифрлағаннан кейін, Труди оны кері шифрлап, оқып, Бобтың ашық кілтінің көмегімен шифрлап, Бобқа жөнелтеді, ал Боб бұл жайлы білмейді. Ең сорақысы, Труди мәлімдемені оқығаннан кейін, шифрламас бұрын өзгертуі мүмкін. Демек, ашық кілтпен алмасудың қандай да бір құпия механизмі қажет.

8.5.1. Сертификаттар

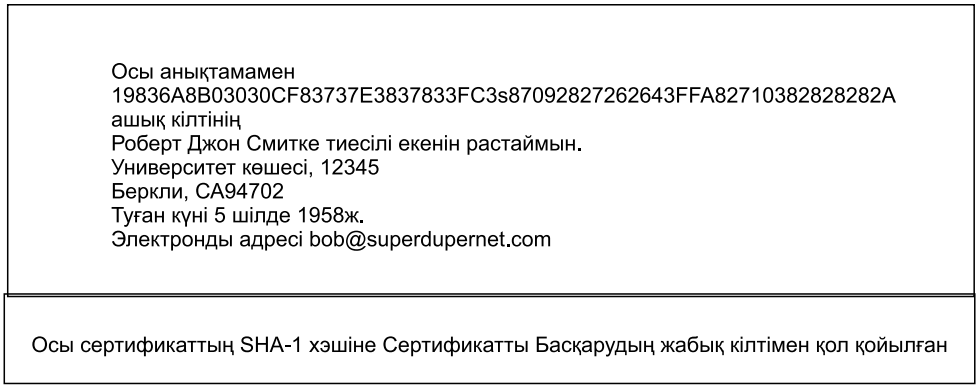
Ашық кілттермен қорғалған алмасуды ұйымдастырудың алғашқы талпынысы – тәулік бойы жұмыс істейтін, талап бойынша **кілттерді тарату интернет-орталығын** құрастыру. Мұндай шешімнің көптеген кемшіліктерінің бірі – бұл шешімді масштабтау мүмкін емес және бұл орталықтың өзі тез арада жіңішке орын болады. Ал егер ол жүктемені көтере алмаса, бүкіл интернет қауіпсіздігі әп-сәтте жоққа айналады.

Осы себептермен, кілтті тарату орталығы тәулік бойы қолжетімді болуын қажет етпейтін, жаңа шешім ұсынылды. Тіпті, орталықтың онлайн режимде жұмыс істеуі де қажет емес. Оның орнына орталыққа, жеке тұлға, сонымен бірге заңды тұлғаларға тиесілі ашық кілттерді сертификаттау міндеті жүктеледі. Қазіргі кезде ашық кілттерді сертификаттаумен айналысатын ұйым **Сертификаттауды басқару (CA – Certification Authority)** деп аталады.

Мысалы ретінде келесідей жағдайды қарастырайық: Боб Алиса және таныс емес басқа да адамдарға өзімен қорғалған байланыс орнатуға рұқсат бергісі келеді. Ол Сертификаттауды басқару орталығына өз ашық кілтімен, төл құжаты немесе басқа құжатымен келіп, өз кілтін тіркеуді сұрайды. Сертификатты басқару орталығы оған *8.21-суретте* көрсетілгенге ұқсас сертификат беріп, хэш SHA-1-ге өзінің жабық кілтімен қол қояды. Содан кейін Боб Сертификаттауды басқару орталығы қызметінің төлемақысын төлеп, сертификат және қолтаңбасы бар хэш жазылған дискті алады.

Сертификаттың негізгі міндеті ашық кілтті принципал атымен (жеке немесе заңды тұлға) байланыстыру. Сертификаттар өздері ешуақытта қорғалмайды және құпия сақталмайды. Мысалы, Боб өз сайтына сертификатты қойып, оған «Бұл жерде менің ашық кілтімнің сертификаттарын қарауға болады» деген

сілтеме қоя алады. Осы сілтеме арқылы көшіп, тұтынушы сертификатты және қолтаңбасы бар блокты көре алады (сертификат хэш SHA-1 қол қойылған).



8.21-сурет. Сертификат және қолтаңбасы бар хэш мысалы

Енді *8.20-суретте* көрсетілген сценарийге қайта қарап көрейік. Труди енді Боб парағына Алиса жөнелткен сұранысты ұстап алып, не істей алады? Труди парақта өзінің жекеменшік сертификатын және жалған парақта қол қойылған блокты қоя алады, алайда, Алиса сертификатты оқып, өзінің Бобпен сөйлесіп отырмағанын бірден сезеді, себебі ол жерде оның аты мүмдем жоқ. Труди ашық кілтті өз кілтімен ауыстырып, «лезде» Бобтың үй парағын өзгерте алады. Осылай бола тұрса да, ол сертификатты SHA-1 алгоритмімен тексеріп, хэш-функция мәнінің Сертификатты басқару және қолтаңба блогы ашық кілтімен есептелген мәнмен сәйкес келмейтінін көреді. Труди Сертификатты басқару жабық кілтіне қол жеткізе алмайтын болғандықтан, ол өзінің ашық кілті орналасқан, өзгертілген парақ хэші бар қолтаңба блогын генерациялай алмайды. Осылайша, Алиса Боб кілтінің тұпнұсқа екендігіне күмән келтірмейді. Біз жоғарыда айтқандай, мұндай схемада Сертификатты басқару орталығының тәулік бойы үздіксіз байланыс режимінде жұмыс істеуі қажет емес. Демек, жүйенің ықтимал жіңішке орны жойылады.

Сертификат ашық кілтті тек принципалмен ғана емес атрибутпен де байланыстыра алады. Мысалы, сертификатта мынадай ақпарат болуы мүмкін: «Бұл ашық кілт 18 жастан үлкен адамға тиесілі». Осылайша принципал статусын растап немесе оның жас бойынша шектеу қойылған кейбір арнайы деректерге қолжеткізуге рұқсаты бар екендігін білуге болады. Мұнда принципалдың аты айтылмауы да мүмкін. Әдетте сертификат иесі оны принципал веб-парағына немесе клиент жасы жайлы абыржыған үдеріске жөнелтеді. Жауап ретінде ашық кілтпен шифрланатын кездейсоқ сан (сертификаттан оқылатын) генерацияланады. Егер клиент оны кері шифрлап қайта жөнелте алса, онда оның шын мәнінде сертификатта көрсетілген қасиеттерге ие екенінің дәлелі болады. Бұл кездейсоқ санды болашақ байланыс үшін сеанстық кілт ретінде де пайдалануға болады.

Тағы бір жағдайда сертификатта атрибут болуы мүмкін: егер әңгіме объектіге бағытталған таратылған жүйе жайлы болса. Әдетте әр объектінің қандай да бір тәсілдер жиынтығы болады. Объект иесі әр клиентке ол пайдалана алатын тәсілдер тізімі көрсетілген сертификат бере алады. Бұл тізім ақпаратты бейнелеу разрядты картасы ретінде (биттік карта) қол қойылған сертификатты пайдаланып ашық кілтпен байланыстыра алады, әрине егер сертификат иесі сәйкес жабық кілттің бар екендігі дәлелдей алса, ол тізімде көрсетілген тәсілдерді қолдана алады. Сертификатты бұлай пайдаланудың ерекшелігі – сертификат иесінің аты жөнін көрсетудің қажет еместігі. Бұл құпиялық қажет болған жағдайда пайдалы.

8.5.2. X.509

Егер қол қойғысы келгендердің барлығы әртүрлі сертификаттар бойынша Сертификатты басқару орталығына жүгінетін болса, әртүрлі форматтарға қызмет көрсету қиын болар еді. Бұл мәселені болдырмас үшін ITU ұйымы арнайы сертификаттар стандартын бекітті. Ол **X.509** деп аталады және Интернетте кеңінен қолданылады. 1988 жылдан бастап стандарттың үш нұсқасы жарық көрді. Біз олардың ең жаңасы – үшінші нұсқаны қарастырамыз.

X.509 стандартына OSI әлемі үлкен ықпал етті, осыған байланысты онда кейбір келеңсіз заттар орын алды, мысалы, кодтау және ат беру қағидалары. Интернеттің даму мәселелік тобы, IETF, осы тұжырымдамалармен келіскені ғажап, әсіресе машиналық адрестерден бастап, транспорттық хаттамалар және электронды хаттар форматына дейін, іс жүзінде барлық аумақта IETF OSI-ді елемейтін және қандай да бір ұғымды зат жасауға тырысатын. X.509-дың IETF нұсқасы RFC 5280 құжатында сипатталған.

Іс жүзінде X.509 – бұл сертификаттарды сипаттау амалы. Сертификаттың негізгі өрістері *8.3-кестеде* келтірілген. Бірінші бағанада келтірілген түсініктемелерден өрістің не үшін керектігі түсінікті. Қосымша ақпарат алу үшін RFC 2459 жүгініңіз.

Мысалы, егер Боб Money Bank банкінің несие бөлімінде жұмыс істейтін болса, оның X.500 адресі келесідей болар еді:

```
/C=US/O=MoneyBank/OU=Loan/CN=Bob/
```

мұндағы, *C* – ел, *O* – ұйым, *OU* – ұйым бөлімі, *CN* – аты. Сертификатты басқару және басқа да мәндер ұқсас аталады. X.500 ат беру жүйесіндегі басты мәселе, егер Алиса *bob@moneybank.com* байланысуға талпынса және осы форматтағы сертификат деректері бар болса, онда бұл сертификаттың оған керек Боб екені анық емес болады. Қуанышқа орай, үшінші нұсқадан бастап X.500 орнына DNS атын пайдалануға рұқсат етілген, сондықтан бұл мәселе сәтті шешілді.

Сертификаттар **OSI ASN.1 (Abstract Syntax National 1 – абстракты синтаксисті жазу жүйесі 1)** көмегімен кодталады. Оны C тілінің құрылымын-

да ұқсатуға болады, тек бұл жазба көп сөзді және біраз ерекше. X.509 жайлы толық ақпаратты Ford және Baum (2000) басылымынан қарауға болады.

8.3-кесте. X.509 стандартындағы сертификаттың негізгі өрістері

Өріс	**Мәні**
Version	X.509 нұсқасы
Serial number	Бұл сан Сертификатты басқару атымен бірге сертификатты бірегей сәйкестендіреді
Signature algorithm	Сертификат қолтаңбасын генерациялау алгоритмі
Issuer	Басқарудың X.500 аты
Validity period	Жарамдық мерзімінің басы және соңы
Subject name	Мән, сертификацияланатын кілт
Public key	Мәнінің ашық кілті және оның алгоритмін пайдаланатын идентификатор
Issuer ID	Сертификат эмитентін (құрастырушысын) жалғыз анықтайтын, міндетті емес идентификатор
Subject ID	Сертификат иесін жалғыз анықтайтын, міндетті емес идентификатор
Extensions	Әртүрлі мүмкін кеңейтілімдер
Signature	Сертификат қолтаңбасы (Сертификатты басқару жабық кілтімен генерацияланады)

8.5.3. Ашық кілті бар жүйе инфроқұрылымы

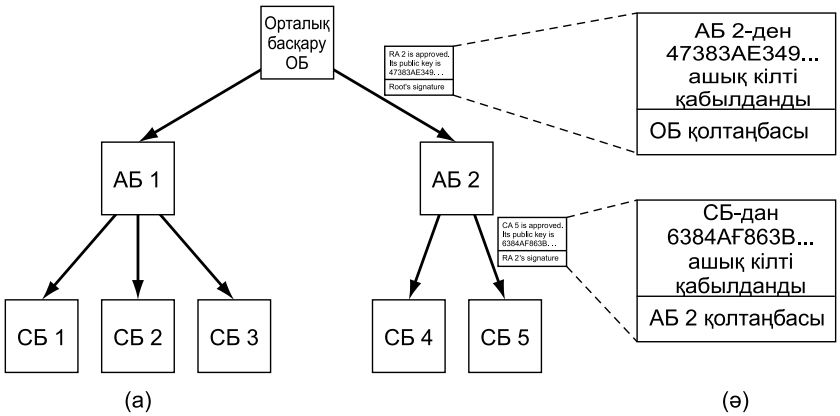
Әрине, бүкіл әлем үшін бір ғана Сертификатты басқару жеткіліксіз екені түсінікті. Үлкен жүктеме салдарынан ол бірден істен шығар еді, сонымен бірге желі қауіпсіздігімен байланысты мәселелер орталығы болар еді. Мүмкін, бір ұйым басқаратын осындай Сертификатты басқару орталықтары жиынтығын құрастыру қажет шығар. Алайда бұл жүктемені тарату мәселесін шешкенмен, кілттің жайылып кетуімен байланысты жаңа сұрақ туындайды. Егер бүкіл әлем бойынша Сертификатты басқарудың жабық кілттері сақталған ондаған серверлері шашыратып орналасқан болса, онда ерте ме, кеш пе кілттің ұрланып немесе басқа бір жолмен жоғалу ықтималдығы өседі. Егер кілт құпия болудан қалса, онда бүкіл әлемнің қауіпсіздік инфроқұрылымын жерлеуге болады. Сонымен бірге, бір ғана Сертификатты басқару орталығы – бұл да қауіп.

Сонымен, Сертификатты басқаруды қандай ұйым басшылыққа алады? Бүкіл әлемдегі алдын ала сенім көрсету үлкен қандай да бір заңды ұйымды елестету өте қиын. Кейбір елдерде мұндай ұйымның үкіметтік ұйым болғаны жарамды, ал кейбірінде – керісінше, қандай ұйым болса да, тек үкіметтік емес.

Осы себептермен ашық кілтті сертификаттаудың балама әдісі құрас-

тырылды. Ол жалпы **PKI (Public Key Infrastructure – ашық кілтті жүйелер инфрақұрылымы)** деген атпен танымал. Біз бұл бөлімде PKI әрекеттеу қағидаларын қарастырып өтеміз, оны түрлендіру жайлы көптеген ұсыныстар болғандықтан кейбір егжей-тегжейлер уақыт өте өзгеруі мүмкін.

PKI көптеген құрама бөліктерден тұрады, оның ішінде тұтынушылар, Сертификатты басқару орталығы, сертификаттардың өзі, сонымен бірге каталогтар. Ашық кілтті жүйелер инфрақұрылымы осы құрама бөліктерді құрылымдық ұйымдастыруды ұсынады және әртүрлі құжаттар мен хаттамаларға қатысты стандарттарды анықтайды. PKI қарапайым түрінің бірі – *8.22-суретте* көрсетілген басқару ирреархиясы. Суретте үш деңгей көрсетілген, алайда іс жүзінде олар одан көп те, аз да болуы мүмкін. Жоғары деңгейдегі Сертификатты басқаруды (root) біз Орталық басқару (ОБ) деп атаймыз. Орталық басқару басқарудың екінші деңгейін сертификаттайды. Олар қандай да бір географиялық аймақты, мысалы, елді немесе континентті басқаратын болғандықтан, біз оны **Аймақтық бөлімдер (АБ – Regional Authorities, RA)** деп атаймыз. Бұл термин стандартталған. Әдетте иеррархия деңгейлерінің атауы стандартталмайды. Аймақтық бөлімдер өз кезегінде жеке және заңды тұлғалар үшін X.509 стандартын эмитациялайтын нақты Сертификатты басқару (СБ) орталықтарын заңдастырады. Орталық басқару ұйымы жаңа Аймақтық бөлімдерді заңдастыру кезінде оларға заң жүзінде танылғандығы жайлы сертификат береді. Сертификатта жаңа АБ ашық кілті және ОБ қолтаңбасы көрсетіледі. Аймақтық басқару да Сертификатты басқару орталықтарымен дәл осылайша әрекеттеседі: СБ орталығының ашық кілті және оның қызметінің заң жүзінде танылғандығы жайлы сертификат беріп, қол қояды.



8.22-сурет. PKI иерархиясы (а); сертификаттар тізбегі (ә)

Сонымен, біздің PKI келесідей жұмыс істейді. Айталық, Алисаға Бобпен хабарласу үшін оның ашық кілті қажет болды. Алиса оны іздеп, СБ 5 орталығы қол қойған сертификатты тауып алады. Алайда, Алиса СБ 5 ешуақытта естіген емес. Іс жүзінде бұл «Сертификатты басқару» орталығы Бобтың он жасар қызы болуы да мүмкін. Алиса СБ 5 барып, оның жұмысының заңды екенін

дәлелдеуін сұрай алады. СБ 5 жауап ретінде АБ 2 алған және СБ 5 ашық кілті көрсетілген сертификатын көрсетеді. Енді СБ 5 ашық кілтімен қамтамасыз етілген Алиса Боб сертификатына, шын мәнінде СБ 5 қолы қойылғанын, яғни оның заңды екеніне көз жеткізе алады.

Әрине, егер АБ 2 Бобтың он екі жастағы ұлы болмаса. Егер Алисаға кенеттен осы ой келетін болса, ол АБ 2 заңды екеніне сұраныс жасай алады. Енді Алиса кімде кімнің емес, нақты Бобтың ашық кілтін алғандығынан күмәнданбайды.

Ал егер Алиса Боб ашық кілтін білгісі келсе ше? Бұны қалай жүзеге асыруға болады? Жұмбақ. ОБ ашық кілтін барлығы біледі деп жорамалдаймыз. Мысалы, ол оның браузерінің ішінде «қыстырулы» болуы мүмкін.

Бірақ біздің Боб өте ақпейіл адам және Алисаны әурелегісі келмейді. Ол Алисаның СБ 5 және АБ 2 заңды екендігін тексергісі келетіндігін біледі, сондықтан сәйкес сертификаттарды өзі жинап, оларды өз сертификатымен бірге жөнелтеді. Енді Алиса ОБ кілтін білгеннен кейін, басқа ұйымдардың заңдылығын кезегімен тексере алады. Заңдылықты растау үшін ешкімді де мазаламайды. Сертификатардың барлығы қол қойылған болғандықтан, ол кез келген құжаттың жалған екендігін ажырата алады. Аяғы ОБ апаратын сертификаттар тізбегін кейде **сенімді тізбек (chain of trust)** немесе **сертификат жолы (certification path)** деп атайды. Осы сипатталған тәсіл іс жүзінде кеңінен қолданылады.

Әрине, ОБ иесін анықтау мәселесі сақталады. Бір емес бірнеше Орталық басқару қажет және олардың әрқайсысымен өз Аймақтық басқару бөлімдер және Сертификатты басқару орталықтар иерархиясын байланыстыру қажет. Іс жүзінде қазіргі заманға браузерлерге кейде **сенімді якорь (trust anchors)** деп аталатын 100-ден аса Орталық басқару ашық кілттерін «қыстырады». Осылайша, өздеріңіз көргендей, сертификаттаумен айналысатын, бүкіл әлемдік бір ұйым мәселесін шешуге болады.

Алайда, браузер шығарушылар қандай сенімді якорьді – сенімді, ал қайсысын – сенімсіз деп санай алады деген сұрақ туындайды. Түбінде барлығы тұтынушының браузер құрастырушыға қаншалықты сенетіндігіне, шешімнің қаншалықты сауатты қабылданғандығына және сенімді якорьлер ашық кілті ықылас білдіргендердің барлығынан белгілі бір төлемақы үшін «қыстырылмағандығына» қаншалықты сенімді екендігіне байланысты болып келеді. Браузерлердің көпшілігі ОБ кілттерін тексеруге (әдетте бұл өзі қол қойға сертификат көмегімен жүзеге асырылады) және күмәнді кілттерді жоюға мүмкіндік жасайды.

Каталогтар

Ашық кілттер жүйесі инфроқұрылымы тағы бір сұраққа жауап береуі тиіс. Ол сертификаттарды сақтау орны (және қандай да бір сенімді якорьге апаратын тізбекті). Тұтынушылардың барлығын өз сертификаттарын өзінде сақтауға

мәжбүрлеуге болады. Бұл қауіпсіз (себебі қол қойылған сертификатты қолдан жасау мүмкін емес), бірақ ыңғайсыз. Сертификаттар каталогы ретінде DNS-ты пайдалану ұсынылды. Бобпен байланыспас бұрын Алисаға домен аттары қызметі (DNS) көмегімен оны IP-адресін білу қажет болады. Сондықтан, неліктен DNS-ке IP-адреспен бірге сертификаттар тізімін қайтармасқа?

Кім-кімге бұл тығырықтан шығу болып көрінуі мүмкін, бірақ кейбіреулер X.509 сертификаттарын сақтау үшін арнайы каталогтары бар сервер қажет деп санайды. Мұндай каталогтар X.500 аттарының көмегімен іздеуді қамтамасыз етер еді. Мысалы, теориялық тұрғыдан, «Маған АҚШ немесе Канаданың кез келген сату бөлімінде жұмыс істейтін, Алиса атты барлық адамдардың толық тізімін беріңіз» деген сұранысқа жауап қайтаратын каталогтар серверінің қызметін елестетуге болады.

Күшін жою

Нақты әлем әртүрлі көптеген сертификаттарға толы, олардың ішінде, мысалы, төл құжаттар, жүргізуші куәлігі. Кейде бұл сертификаттардың күшін жою керек болады (мысалы, ішімдік ішіп рөлге отырғандығы үшін жүргізуші куәлігінің күшін жою). Дәл осы мәселе сандық технологиялар әлемінде де туындайды: сертификатты берген тұлғаны оны қандай да бір шартты бұзғандығы үшін қайта алып алуы мүмкін. Бұны жабық кілт іс жүзінде қорғалған болудан қалған кезде немесе ОБ кілті сенімді жоғалтқан кезді жүзеге асыру керек. Осылайша, ашық кілттер жүйесінің инфрокұрылымы қандай да бір жолмен сертификат күшін жоюды қамтамасыз етуі керек. Күшті жою мүмкінді барлығын қиындатады.

Бұл бағыттағы бірінші қадам барлық ОБ дүркін-дүркін күші **жойылған сертификаттар тізімі (CRL – Certificate Revocation List)** шығаруға мәжбүрлеу. Бұл тізімде барлық күші жойылған сертификаттардың реттік нөмірі жазылады. Сертификаттарда жарамдық мерзімі көрсетілетін болғандықтан, CRL-ге тек жарамдылық мерзімі аяқталмағандарды қосу керек. Жарамдық мерзімі аяқталғаннан кейін сертификаттар автоматты түрде жарамсызданады, сондықтан «ескіру» бойынша күшін жою және басқа себептермен күшін жоюды ажырату керек. Кез келген жағдайда оларды пайдалануға шек қою керек.

Өкінішке орай, күші жойылған сертификаттар тізімінің пайда болуы, сертификатты пайдаланбақ болған адамның алдымен оның осы тізімде бар екендігі тексеруін қажет етеді. Егер сертификат тізімде бар болса, оны пайдаланудан бас тарту керек. Осылай бола тұрса да, сертификат қара тізімнің жаңа нұсқасы шыққан кезде күшін жоюы мүмкін. Демек, ең сенімді жол – сертификат қалып-күйін тікелей ОБ-дан білу және бұл сұранысты сертификатты әрбір пайдалану алдында жөнелту керек болады, себебі оның бірнеше секунд бұрын күшін жоймағандығына ешкім кепілдік бере алмайды.

Кейде күшін жойған сертификатты қайта қалпына келтіру мәселені одан әрі қиындатады. Мысалы, егер сертификатты қайтып алудың себебі қандай

да бір жарна төлу болса, онда қажет жарна төленгеннен кейін сертификатты қайта қалпына келтірмеудің ешбір себебі қалмайды. Сертификаттың күшін жою және қайта қалпына келтіру жағдайын өңдеу, сертификатты ОБ көмегінсіз пайдалану, маңызды қасиетін жоққа шығарады.

Күші жойылған сертификаттар тізімін қай жерде сақтау керек? Оларды да сол сертификаттар сақталған жерде сақтаған жақсы болар еді. Мұндағы стратегияның бірі – ОБ дүркін-дүркін сертификаттар қара тізімін шығарады және жаңартуларды каталогқа енгізуге мәжбүрлейді (күші жойылған сертификаттарды өшіріп). Егер сертификаттарды сақтау үшін каталогты пайдаланбасақ, онда оларды желінің әр жерінде кэштеуге болады. Қара тізім қол қойылған құжат болғандықтан, оны қолдан жасау әрекеті бірден белгілі болады.

Егер сертификаттың жарамдық мерзімі үлкен болса, онда күші жойылған сертификаттар тізімі де ұзын болады. Мысалы, жарамдық мерзімі 5 жыл несие карталарының ішіндегі күші жойылғандар тізімі, жарамдық мерзімі үш ай несие карталарына қарағанда әлдеқайда ұзын болады. Ұзын тізімдермен күресудің стандартты жолы – тізімнің өзін сирек шығару және оларға жаңартуларды жиі енгізіп отыру. Сонымен бірге, бұл тізімдерді таратуға қажет өткізгіштік қабілеттілікті төмендетуге көмектеседі.

8.6. БАЙЛАНЫСТАРДЫ ҚОРҒАУ

Біз қолданбалы құралдарды қарастыруды аяқтадық. Қолданыстағы көптеген тәсілдер және хаттамалар сипатталды. Тараудың қалған бөлігінде осы тәсілдердің желі қауіпсіздігін қамтамасыз ету үшін қалай қолданылатынын қарастырамыз. Сонымен бірге, осы сұрақтың әлеуметтік жағы жайлы да сөз қозғаймыз.

Келесі төрт бөлім байланыс қауіпсіздігіне арналған, яғни биттерді жөнелту пункітінен қабылдау пунктіне дейін құпия және қолдан ауыстыру қаупінсіз қалайша тасымалдау, сонымен бірге, торапқа бөтен биттерді қалай ендірмеуді қарастырамыз. Бұл желі қауіпсіздігінің толық тізімі емес, алайда, аталған сұрақтар солардың ішіндегі ең маңыздысы болып саналады.

8.6.1. IPsec

IETF мәселелік тобы көптеген жылдар бойы Интернетте қауіпсіздіктің жоғымен келісіп келді. Интернет қауіпсіздігін қамтамасыз ету оңай мәселе емес, әсіресе, оның қандай бөлігін қорғау керектігі жайлы қызу тартыстар орын алды. Қауіпсіздік сұрақтары бойынша көптеген эксперттер, шын мәнінде сенімді жүйе толассыз шифрлануы және деректер тұтастығын толассыз қамтамасыз ету керектігіне (яғни, бұл қолданбалы деңгейде жүзеге асырылуы тиіс) сенімді. Бұл ақпарат көзі үдерісі деректер тұтастығына қорғаныс қояды немесе шифрлайды және оны деректерді кері шифрлап, тұтастығын тексеретін

қабылдаушы үдеріске жөнелтеді дегенді білдіреді. Бұл жағдайда кез келген бұзу талпынысын (тіпті, операциялық жүйе деңгейінде, екі жақта да) байқауға болады. Мұндай көзқарастың басты кемшілігі – қауіпсіздікті қамтамасыз ету үшін барлық қосымшаларға өзгертулер енгізу керек болады. Бұл шифрлауды транспорттық деңгейге «түсіру» немесе қолданбалы және транспорттық деңгей арасында арнайы ішкі деңгей ұйымдастыру керек дегенді білдіреді. Ол толассыз болу керек, бірақ қосымшаларға өзгерту енгізуді қажет етпеуі тиіс.

Кері көзқарас – тұтынушылар қауіпсіздік шараларын қолдану керектігін бәрібір түсінбейді және ұсынылған мүмкіндіктерді толық пайдаланбайды. Сонымен бірге, ешкім де қолданыстағы программаларды өзгерткісі келмейді, сондықтан желілік деңгей оның тұпнұсқа екендігін тексеруі немесе мәлімдемелерді тұтынушыларға білдірмей шифрлауы керек. Көп жылдарға созылған тартыста осы көзқарас жеңіп шықты: желілік деңгейге бағытталған қауіпсіздік стандарты құрастырылды. Басты аргумент – желілік деңгейде шифрлау қауіпсіздікке көп көңіл бөлетін тұтынушыларға кедергі жасамайды, ал екінші жағынан – бейберекет тұтынушыларды біраз қорғайды.

Осы пікірталастардың нәтижесінде **IPsec (IP security – IP-қауіпсіздік)** стандарты құрастырылды. Бұл стандарт RFC 2401, 2402, 2406 және басқа да құжаттарда сипатталған. Тұтынушылардың барлығына бірдей байланысты шифрлау қажет емес (сәйкес процедураны орындау есептеу ресурстарының едәуір бөлігін алады). Алайда, шифрлауды міндетті емес етудің орнына, тұтынушыға қажет болған жағдайда ол алгоритмді таңдау ұсынылады. RFC 2410-да бос алгоритмнің қарапайымдылық, жүзеге асыру жеңілділігі және жоғары жылдамдық тәрізді артықшылықтар сипатталады.

IPsec көптеген қызметтер, алгоритмдер және модульдер негізі болып келеді. Қызмет түрлерінің көп болуының бір себебі – көптеген қызметтер үшін барлығы бірдей ақы төлегісі келмейді, сондықтан қажет қызметтер бөлек-бөлек ұсынылады. Негізгі қызметтер мынадай: құпиялық, деректер тұтастығы, мәлімдемелерді қайталау арқылы сындырудан қорғау (зиянкес рұқсатсыз тыңдаған әңгімені қайталаған кезде). Бұның барлығы симметриялы кілтті криптографияға негізделген, себебі мұндай үлкен өнімділік маңызды.

Алгоритмдер жиынтығы не үшін қажет? Мәселен бүгін сенімді деп саналған алгоритм, ертең сындырылуы мүмкін. Егер IPsec нақты алгоритмге тәуелді болмаса, онда стандарт, тіпті бір алгоритм сынған кезде де сақталып қалады.

Әртүрлі модульдер не үшін қажет? Бір TCP-байланысты да, хосттар жұбы арасындағы бүкіл трафикті де және қорғалған екі маршруттауыштар арасындағы бүкіл трафикті және т.с.с. қорғау үшін қажет.

IPsec IP деңгейде болғанына қарамастан байланысқа бағытталғандығы біршама таңқаларлық. Іс жүзінде таңқаларлық ештеңе жоқ. Қауіпсіздікті тек кілт жасап, оны қандай да бір уақыт аралығында пайдалану арқылы ғана қамтамасыз етуге болады. Ал бұл – байланыстың бір түрі. Сонымен бірге барлық байланыстар шығыны үлкен санды дестелер тасымалдау арқылы қайтарылады. IPsec тұрғысынан «байланыс» **қорғаушы байланыс (Security Aconnection -**

SA) деп аталады. Қорғаушы байланыс – бұл арнайы қорғаныс идентификаторы байланысқан, екі соңғы нүкте арасындағы симплексті байланыс. Егер екі бағытта да қорғалған деректерді тасымалдау қажет болса, онда екі қорғаушы байланыс керек. Қорғау идентификаторлары осы сенімді байланыс арқылы жөнелтілетін дестелерде беріледі және қорғалған дестелер жеткізілген кезде кілтті және басқа да маңызды ақпаратты іздеу үшін пайдаланылады.

Техникалық тұрғыдан IPsec екі негізгі бөліктен тұрады. Біріншісі дестеге қорғау идентификаторы, деректер тұтастығын бақылау және басқа да ақпаратты қосуға болатын, екі жаңа тақырыпты сипаттайды. Екінші бөлігі, **ISAKMP (Internet Security and Key Management Protocol – интернет-қауіпсіздік және кілттерді басқару хаттамасы)** – кілттер жасауға арналған. Жұмысты орындау үшін қажет негізгі хаттама – **IKE (Internet Key Exchange – Интернет кілттермен алмасу)**. Міндетті түрде 2 нұсқаны пайдалану керек, себебі, алдыңғы нұсқаның Perlman және Knaufmn (2000) көрсетілгендей күрделі кемшіліктері бар.

IPsec екі режимде жұмыс істей алады. **Транспорттық режимде** IPsec тақырыбы бірден IP тақырыбынан кейін орналастырылады. IP тақырыбының *Protocol* өрісі ары қарай IPsec тақырыбының орналасқандығы түсінікті болатындай етіп өзгертіледі (TCP тақырыбының алдында). IPsec тақырыбында қауіпсіздікке байланысты ақпарат орналасқан – жеке алғанда, қорғаушы байланыс идентификаторы, жаңа реттік нөмір және мүмкін пайдалы жүктеме өрісі тұтастығын тексеру.

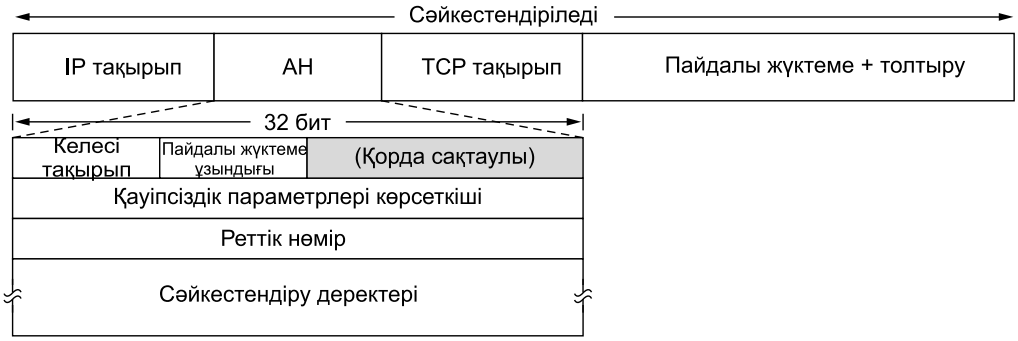
Туннелдеу режимінде бүкіл IP-дестe тақырыбымен бірге жаңа, тақырыбы да жаңа IP-дестe ішіне орналастырылады. Бұл режим туннел соңғы тағайындалған пункттен тыс жерде аяқталған кезде пайдалы. Кейбір жағдайда туннел соңы шлюз, қауіпсіздікті қамтамасыз ететін, мысалы, корпоративті желіаралық экран болып келеді. Ол әдетте **VPN (Virtual Private Network – виртуалды жеке желі)** үшін пайдаланылады. Бұл режимде желіаралық экран өзі арқылы әртүрлі бағытта өтетін дестелерді енгізеді және шығарып алады. Машиналар осылайша ұйымдастырылған кезде компанияның ЖЕЖ-не кепілденген түрде IPsec стандарты бойынша қызмет көрсетіледі. Ол жайлы ешбір қам жеудің қажеті жоқ, барлық жауапкершілікті желіаралық экран өзіне алады.

Сонымен бірге, туннелдеу режимі бірнеше TCP-байланыс бірге біріктіріліп, тұтас шифрланған ағын ретінде өңделетін жағдайда пайдалы, себебі бұл жағдайда зиянкес кімнің кімге қандай десте және неше десте жөнелткенін біле алмайды. Кей жағдайда, тіпті бір адамның екіншіге жөнелткен трафик көлемі де маңызды ақпарат болып саналады. Мысалы, әскери дағдарыс кезінде Пентагон және Ақ үй арасындағы трафик күрт төмендесе және дәл осы кезде Пентагон және Колорадодағы басқа бір әскери база арасындағы трафик күрт өссе, зиянкес бұдан алысқа баратын қорытынды шығаруы мүмкін.

Ағын құрылымын өткен дестелер саны бойынша зерттеу **трафикті сараптау** деп аталады. Егер туннелдеу пайдаланылатын болса, онда мұндай сараптау қиынға түседі. Туннелдеу режимінің кемшілігі – IP-дестe тақырыбын

кеңейтуге тура келеді, бұның салдарынан дестенің қосынды көлемі едәуір өседі. Транспорттық режимде деста көлемі елеуліктей өспейді.

Жаңа тақырыптардың біріншісі **сәйкестендіру тақырыбы (CT – Authentication Header, AH)** деп аталады. Оның көмегімен деректер тұтастығын тексеру және қайталап тасымалдау арқылы сындырудан қорғауды жүзеге асыру орындалады. Алайда оның құпиялыққа ешқандай қатысы жоқ (деректерді шифрлауға). Транспорттық режимде сәйкестендіру тақырыбын пайдалану *8.23-суретте* көрсетілген. IPv4 стандартында ол IP және TCP тақырыптарының ортасында (міндетті емес өрістермен бірге) орналасқан. IPv6 стандартында бұл әлі қосымша тақырып. Ол дәл осылайша қабылданады. Сәйкестендіру тақырыбы форматы шынында да IPv6 қосымша тақырыбының форматына ұқсас. Кейде сәйкестендіру алгоритміне қажет ұзындыққа жеткізу үшін пайдалы жүктемеге *8.23-суретте* көрсетілгендей толықтыру қосылады.



8.23-сурет. Транспорттық режимдегі IPv4 үшін IPsec сәйкестендіру тақырыбы

Сәйкестендіру тақырыбын қарастырайық. *Келесі тақырып* өрісі ары қарай сәйкестендіру тақырыбы орналасқанын білдіріп, ертеде IP тақырыбындағы *Хаттама* өрісінің 51-ге ауыстырылған мәнін сақтайды. Әдетте бұнда TCP (6) арналған код кездеседі. *Пайдалы жүктеме ұзындығы* өрісінде сәйкестендіру тақырыбының 32-разрядты сөзінен 2-ні азайтқандағы сан сақталады.

Қауіпсіздік параметрлері көрсеткіші өрісі – бұл байланыс идентификаторы. Оны жөнелтуші енгізеді және қабылдаушы деректер базасындағы нақты жазбаға сілтеме жасайды. Бұл жазбада жалпы кілт және осы байланыстың басқа да ақпараты орналасқан. Егер бұл хаттаманы IETF емес ITU ойлап тапқан болса, бұл өріс *Виртуалды арна* нөмірі деп аталуы ықтимал еді.

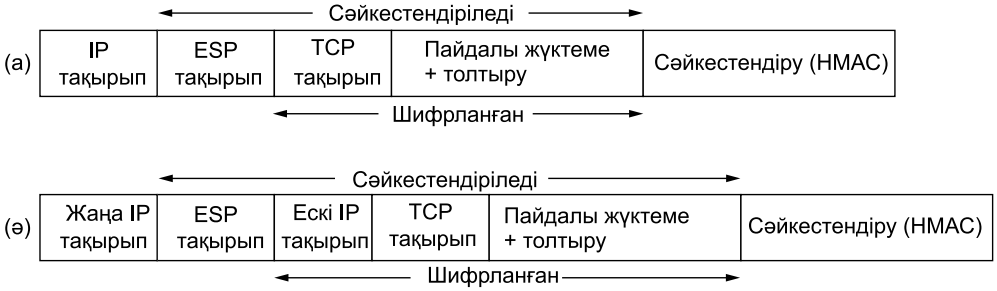
Реттік нөмір өрісі қорғалған байланыс арқылы жөнелтілетін барлық дестелерді нөмірлеу үшін қолданылады. Дестелердің барлығы бірегей нөмірге ие болады, тіпті ол қайталап жөнелтіліп отырса да. Қайталап жөнелтіліп отырған дестенің нөмірі оның тұпнұсқасы нөмірінен ерекше болады дегенді білдіреді (тіпті, TCP нөмірі сол бұрынғы болып қалса да). Бұл өріс қайталап жөнелту арқылы сындыруға орын бермеуді жүзеге асырады. Реттік нөмірлер

ешуақытта қайталанбайды. Егер барлық 2^{32} нөмір қолданылған болса, онда әрекеттесуді жалғастыру үшін жаңа қорғалған байланыс орнатылады.

Соңында, *Сәйкестендіру деректері* атты ұзындығы айнымалы өріс, пайдалы жүктемеге қатысты есептелетін санды қолтаңбадан тұрады. Қорғалған байланыс орнатылған кезде екі жақ санды қолтаңбаны генерациялау үшін қолданылатын алгоритм жайлы келіседі. Мұда ашық кілтпен шифрлау жиі қолданылады, себебі осы типтегі алгоритмдердің барлығы тым баяу жұмыс істейді, ал дестелерді жоғары жылдамдықпен өңдеу қажет. IPsec хаттамасы симметриялы кілтпен шифрлауға негізделген, сондықтан қорғалған байланыс орнату алдында жөнелтуші және қабылдаушы қолтаңбаны есептеуде пайдаланылатын ортақ кілт мәні жайлы келісуі тиіс. Ең қарапайым әдістің бірі десте және ортақ кілт үшін хэш-функцияны есептеу. Ортақ кілт жеке тасымалданбайды. Мұндай схема **HMAC (Hashed Message Authentication Code – хэштелген мәлімдемені сәйкестендіру коды)** деп аталады. Бұл кілтті есептеу SHA-1 және RSA тізбекпен іске қосудан жылдам орындалады.

Сәйкестендіру тақырыбы деректерді шифрлауға мүмкіндік бермейді. Оның басты пайдасы – құпиялықсыз деректер тұтастығын тексеру кезінде білінеді. Деректер тұтастығын Сәйкестендіру тақырыбы арқылы тексерген кезде IP тақырыбының кейбір өрістерінің қамтылатындығын айта кету керек, жеке алғанда бір маршруттауыштан келесі маршруттауышқа өткен кезде өзгермейтіндері. *Өмір уақыты* өрісі, мысалы, әр маршруттауыш арқылы жөнелтілген кезде өзгеріп отырады, сондықтан оны тұтастықты тексеру кезінде қамту мүмкін емес. Алайда ақпарат-көзі IP-адресі қамтылады, осылайша оны зиянкестің алмастыру мүмкіндігіне шектеу қойылады.

IPsec тақырыбына балама ретінде **ESP (Encapsulating Security Payload – инкапсуляциялы қорғалған пайдалы жүктеме)** қызмет көрсетеді. *8.24*-суретте көрсетілгендей бұл тақырып транспоттық режимде де, туннелдік режимде де қолданылады.



8.24-сурет. ESP: а – транспорттық режимде; ә – туннелдік режимде

ESP тақырыбы екі 32-разрядты сөзден тұрады: *Қауіпсіздік параметрлері көрсеткіші* және *Реттік нөмір*. Біз оларды сәйкестендіру тақырыбында кездестіргенбіз. Бұлардан кейін орналасатын үшінші сөз, бірақ техникалық тұрғыдан тақырып бөлігі болып саналмайтын – бұл *Инициализация векторы* (тек

шифрлаудың бос алгоритмі қолданылмаса – онда бұл өріс қалып кетеді). ESP да CT тәрізді тұтастықты HMAC көмегімен тексеруді қамтамасыз етеді, алайда тек тақырыпқа хэшті қосу орнына ол пайдалы жүктемеден кейін орналасады. Бұны *8.24-суреттен* көруге болады. Өрістердің осылайша орналасуының тәсілді аппараттық жүзеге асыруда артықшылығы бар. Артықшылық мәні мынада – HMAC пайдалы жүктеме биттерін желі арқылы тасымалдау кезінде есептеліп, олардың соңына қосылуы мүмкін. Осы себептермен Ethernet және басқа да жергілікті желі стандарттарында молшылықты циклдық бақылау тақырыпқа емес, соңына қосылады. CT тақырыбын пайдаланған кезде дестені буферлеу және қолтаңбаны есептеу қажет, тек содан кейін жөнелтуге болады. Бұл уақыт бірлігінде жөнелтілетін дестелер санының азаюына әкеледі.

Егер ESP CT орындайтын функциялардың барлығын орындаса және тіпті оны әлдеқайда тиімді орындаса, онда CT қажеті не? Бұның басты себебі тарихта. Бастапқыда CT тақырыбы тек тұтастықты қамтамасыз еткен, ал ESP – тек құпиялықты. Кейіннен ESP-ны тұтастықты тексеру үшін пайдалануды үйренді, ал CT құрастырушылары атқарылған қыруар жұмыстардан кейін сәйкестендіру тақырыбының ұмыт қалғанын қаламады. CT пайдасындағы жалғыз артықшылық – оның көмегімен IP тақырыбы бөлігін тексеруге болады, ал ESP бұны орындай алмайды. Осылай бола тұрса да, аргумент өте әлсіз. Тағы бір күмәнді аргумент – CT қолдайтын, бірақ ESP қолдамайтын жүйеге экспортқа лицензия алуда кедергі болатын мәселелер аз, себебі CT құпиялық және шифрлауға қатысы жоқ. Уақыт өте CT ұмыт болуы мүмкін.

8.6.2. Брандмауэрлер

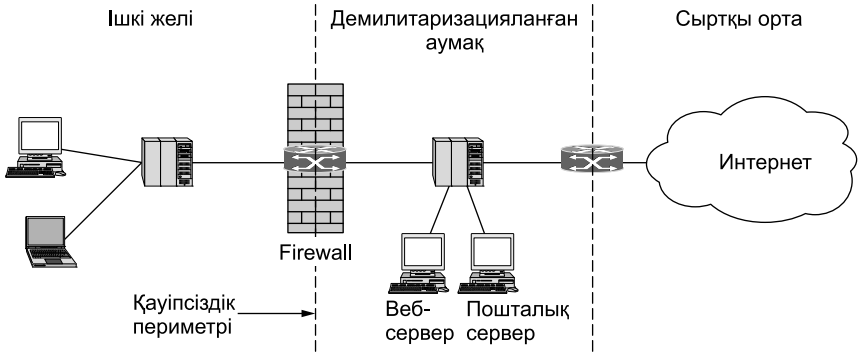
Кез келген компьютерлерді бір-бірімен байланыстыру мүмкіндігі кейбір жағдайда артықшылық, ал кейбір жағдайда кемшілік болып келеді. Интернетте саяхаттау мүмкіндігі үй тұтынушыларына көптеген қуаныштар әкеледі. Корпорацияның қауіпсіздік бөлімінің менеджерлері үшін бұл мүмкіндік сұмдық болып саналады. Көптеген компанияларда желіге қосылған компьютерлерде орналасқан үлкен көлемдегі құпия ақпарат бар – коммерциялық құпиялар, өндірістің даму жоспары, нарық стратегиясы, қаржылық қалып-күйдің аналитикалық есебі және т.б. Бәсекелестер алдында бұл мәліметтердің ашылуының салдары өте нашар болуы мүмкін.

Ақпараттың сыртқа ағып кету қаупінен басқа құпияны сындырып, бағалы деректерді жою мүмкіндігі бар вирустар, құрттар және де басқа сандық індеттер тәрізді зиянды ақпараттың ену қаупі де бар. Олармен күресуге жүйелік әкімшіліктің көп уақыты кетеді. Бұл індетті көбіне жаңа компьютерлік ойындармен ойнағысы келетін, бейберекет қызметкерлер әкеледі.

Осылайша, «жақсы сападағы» ақпаратты іште, ал «зиянды» ақпаратты сыртта ұстап тұратын арнайы құрылғы қажет. Бұл бағыттағы амалдардың бірі IPsec пайдалану. Бұл тәлім деректерді тасымалдау кезінде қорғайды. Алайда, шифрлау желіге ену мүмкіндігі бар вирустар және хакерлерден қорғамайды.

Желіні жағымсыз сырттан енулерден қорғау үшін желіаралық экрандарды орнату көмектесуі мүмкін. Ары қарай біз осы желіаралық экрандарды қарастырамыз.

Желіаралық экран (firewalls, сонымен бірге **брандмауэрлер** деп те аталады) ортағасырлық қауіпсіздікті қамтамасыз етудің қазіргі заманғы жүзеге асырылуы. Олар сарай айналасындағы қорған орынды еске салады. Құрылым мағынасы мынада – сарайға кірушілер және сарайдан шығушылардың барлығы бір көтерілмелі көпір арқылы өтуі тиіс, осы жерде енгізу/шығару полициясы оларды тексере алады. Осы тәсілді желіге де қолдануға болады: компанияның өзара еркін түрде байланысқан бірнеше жергілікті желісі болуы мүмкін, бірақ бүкіл сыртқы трафик *8.25-суретте* көрсетілгендей, электронды көтермелі көпір (желіаралық экран) арқылы өтуі тиіс. Ешқандай басқа жол жоқ.



8.25-сурет. Ішкі желіні қорғайтын желіаралық экран

Брандмауэрлер **дестелік сүзбе (packet filter)** тәрізді жұмыс істейді. Ол әрбір кіріс және шығыс дестесін зерттейді. Белгілі бір шарттарды қанағаттандыратын дестелер сүзбе арқылы өткізіледі. Тексеруден өте алмаған дестелер жойылады.

Сүзбеден өткізу шарттары әдетте мүмкін және шектеу қойылатын жөнелтушілер мен қабылдаушылар тізімі, сонымен бірге кіріс және шығыс дестелерге қолданылатын әрекеттер көрсетілген ережелер немесе кестелер арқылы жүзеге асырылады.

TCP/IP баптауларының жалпы жағдайында қабылдаушы және жөнелтуші жайлы ақпарат IP-адрестен және порт нөмірінен тұрады. Порт нөмірі қажетті қызмет түрлерін анықтайды. Мысалы, 25 порт пошта үшін пайдаланылады, 80 порт – HTTP. Кейбір порттар оқшауланулы болуы мүмкін. Мысалы, компания барлық IP-адрестер үшін TCP және ортақ 79 портпен бірге кіріс дестені оқшаулап қоюы мүмкін. Оны бұрын Finger қызметі электронды адрестерді іздеу үшін кеңінен қолданатын, қазіргі кезде сирек қолданылады.

Қалған порттар бекерге оқшауланбайды. Мұндағы қиындық – желілік әкімшілік қауіпсіздік жайлы армандайды, бірақ сыртқы әлеммен коммуникацияны оқшаулай алмайды. Екінші нұсқа қауіпсіздік үшін ең қарапайым

және ең жақсысы болатын, бірақ тұтынушылар наразылығы толассыз болды. Мұнда *8.25-суретте* көрсетілген **DMZ (DeMilitarized Zone – демилитаризация аумағы)** шешімі көмектеседі. DMZ – бұл қауіпсіздікке қатер төндірмейтін компания желісінің бөлігі. Бұл аймаққа барлығы қосылады. Егер серверді DMZ орналастырса, онда компания сайтына кіру үшін компьютерлер Интернет арқылы сізбен әрекеттесе алады. Желіаралық экран 80 портқа TCP кіріс трафигін оқшаулайтындай етіп бапталуы мүмкін, сондықтан Интернеттке қосылған компьютерлер ішкі желідегі компьютерлерге шабуыл жасау үшін бұл портты пайдалана алмайды. Веб-серверді басқару үшін желіаралық экран ішкі желідегі компьютерлерге веб-сервермен байланысуға рұқсат беруі мүмкін. Қылмыскерлермен күреске қарулану жарысында желіаралық экран күрделі бола бастады. Бастапқыда олар әр дестеге тәуелсіз ережелер жиынтығын пайдаланатын, содан кейін қажет функционалдықты қамтамасыз ететін ережені жазу қиындай бастады, бірақ бүкіл жарамсыз трафикті оқшаулай бастады. **Мәнмәтіндік сүзбесі бар желіаралық экрандар (stateful firewalls)** байланысты қадағалау үшін дестелерді байланыстарға және TCP/IP тақырыптарына бекіте бастады. Осының арқасында келесідей ережелер орнатыла бастады, мысалы, сыртқы веб-сервер ішкі хостқа десте жөнелте алады, тек егер алдымен ішкі хост сыртқы сервермен байланыс орнатса. Мұндай ереже сыртқы серверден келген барлық дестелер енгізілетін немесе енгізілмейтін, қарапайым дестелік сүзбе кезінде мүмкін емес.

Мәнмәтінді есепке алуды күрделендірудің келесі деңгейі – бұл **қолданбалы деңгей шлюзы (application-level gateway)**. Бұл өңдеу кезінде желіаралық экран қосымшаның не істейтіндігін білу үшін дестелерді TCP тақырыбынан тысқарыда тексереді. Бұл жағдайда Интернет желісіндегі сайтқа кіру үшін пайдаланылатын HTTP-трафикті және файлмен алмасу үшін пайдаланылатын HTTP-тафиктен ажыратуға болады. Әкімшілік компания қызметкерлері файлдармен алмаса алмайтындай, бірақ бизнеске қажет сайттарға кіре алатын ережелер жаза алады. Осы тәсілдердің көмегімен шығыс трафикті кіріс трафигіндей зерттеуге болады, мысалы, маңызды құжаттар компания сыртына электронды пошта арқылы кетпес үшін.

Жоғарыда айтылғандардан түсінікті болғандай, желіаралық экран хаттамалар қабатының иерархиялық орналасу стандартын бұзады. Олар желілік деңгей құрылғысы болып келеді, бірақ сүзбеден өткізу үшін транспорттық деңгейді де, қосымшалар деңгейінде бақылайды. Бұл жағдай оларды өте осал етеді. Мысалы, желіаралық экран дестеде қандай трафиктің бар екенін анықтау үшін порттардың стандартты нөмірленуін есепке алады. Стандартты порттар әрине пайдаланылады, бірақ барлық компьютерлер және қосымшалармен бірдей емес. Кейбір пирингілік қосымшалар оларды анықтау (оқшаулау) қиын болу үшін динамикалық портты таңдайды. IPsec немесе басқа схемалар көмегімен шифрлау желіаралық экраннан жоғары деңгей ақпаратын жасырады. Соңында, желіаралық экран өзімен байланысқан компьютерлерге байланыстың не себептен жүзеге аспағанын жылдам хабарлай алмайды. Ол жәй ғана қиылған

кабель қалпында болуы тиіс. Осы себептердің салдарынан желілік пуристтер желіаралық экранды Интернет құрылымының кемшілігі деп санайды. Осылай бола тұрса да, Интернет ең қауіпті орны болуы мүмкін. Желіаралық экрандар осы мәселені шешуге көмектеседі, сондықтан олар әлі де қолданыста болады.

Тіпті мінсіз бапталған желіаралық экран жағдайында да қауіпсіздікпен байланысты көптеген мәселелер қалады. Мысалы, егер тек нақты желіден (мысалы, еншілес компания ЖЕЖ-нен) келген кіріс дестелер енгізілетін болса, брандмауэр әрекет ететін аймақтан тыс орналасқан зиянкес жөнелтуші адресін қолдан жасап, бөгеттен өтіп кетуі мүмкін. Егер компанияның арам қызметкері құпия құжаттарды біреуге жөнелтпек болса, онда ол оны шифрлауы немесе суретке түсіріп алуы мүмкін, бұл жағдайда құжаттар кез келген пошта сарап-таушысы арқылы өте алады. Шабуылдардың 75% желіаралық экран әрекет ету аймағынан тыс орын алса да, желіаралық экран әрекет ету аймағындағы шабуылдардың (мысалы, наразы қызметкер) ең қауіпті болып саналатынын талқыламауға да болады (Verizon, 2009).

Желіаралық экрандардың тағы бір кемшілігі олар бірыңғай периметр немесе бірыңғай қауіпсіздікті қамтамасыз етеді. Егер қорғаныс сынса, бәрінің құрығаны. Осы себептермен желіаралық экрандар көпсатылы қорғаныс үшін қолданылады. Мысалы, желіаралық экран ішкі желіге кірісті қарауылдауы және әр компьютерде де желіаралық экран орналасуы мүмкін. Бір ғана бақылау-өткізу пункті жеткілікті деп санайтын оқырмандар көптен бері халықаралық әуе жолдарымен саяхаттамаған болуы керек.

Сонымен бірге, ешқандай желіаралық экран төтеп бере алмайтын, бүтін-дей бір класс шабуылдары бар. Желіаралық экран негізінде жатқан идея – зиянкестерге жүйеге енуге жол бермеу, ал құпия деректерге – сыртқа шығуға. Өкінішке орай, әлемде сайттар жұмысына зиян келтіруден басқа жақсы ермек таба алмайтын адамдар өте көп. Олар сайттар асыра жүктелуден тұрып қалған-ша заңды мәлімдемелер жөнелтеді. Мысалы, мұндай бұзақылық, байланыс орнатуға *SYN* дестелерін жөнелту болуы мүмкін. Сайттар осы байланысқа кесте бөлігін арнап, жауап ретінде *SYN+ACK* дестесін жөнелтеді. Егер бұзақы жа-уап бермесе, онда кесте жазбасы тайм-аутқа дейін, бірнеше секунд бойы қорда сақтауы болады. Егер бір мезгілде байланысқа мыңдаған сұраныс жөнелтілсе, онда адал тұтынушылардан келген сұраныстар серверге жетпейді, себебі кесте ұяшықтарының барлығы бос емес болып қалады. Мақсаты құпия деректерді алу емес, объект жұмысынна нұқсан келтіру болып саналатын шабуылдар **DoS типті шабуылдар (Denial of Service – қызмет көрсетуден бас тарту)** деп аталады. Әдетте, сұраныс дестесіндегі жөнелтуші адресі жалған болып келеді, сондықтан вандалды табу оңай емес. Үлкен маңызды сайттар үшін DoS шабуылдар әдеттегі шаруа.

Бұндай шабуылдың өте қатаң түрлері де бар. Егер желі бұзақысы бүкіл әлем бойынша орналасқан бірнеше жүздеген компьютерлерді сындыра алса, онда ол оларға бір серверді сұраныстармен толтыруға бұйрық беруі мүмкін. Осылайша, «қирату күші» еселеп қана қоймайды, сонымен бірге бұзақыны

анықтау тіпті мүмкін болмайды, себебі дестелер бұрын өздерін жаман жағынан көрсетпеген әртүрлі компьютерлерден келеді. Шабуылдың бұл түрі **DDoS (Distributed Denial of Service – қызмет көрсетуден таратылған бас тарту)** деп аталады. Бұл бәлемен күресу оңай емес. Тіпті, егер шабуыл жасаушы машина сұраныстың жалған екендігін жылдам анықтай алса да, оны өңдеп, одан бастартуға біраз уақыт кетеді, осы аралықта басқа сұраныстар келеді және нәтижесінде орталық процессор үнемі оларды өңдеумен айналысатын болады.

8.6.3. Виртуалды жеке желілер

Көптеген компаниялардың әртүрлі қалаларда, тіпті кейде әртүрлі елдерде орналасқан бірнеше бөлімшелері бар. Жалпы қолжетімді желі пайда болғанша деректерді тасымалдау үшін бөлінген телефон торабын бірнеше ұйым немесе барлық бөлімшелер арасында байланыс орнатуға жалға алу әдеттегі жағдай болатын. Кейбір компаниялар осындай амалды әлі де пайдаланады. Компанияға тиесілі компьютерлер және бөлінген телефон торабынан тұратын желі – **жеке желі** деп аталады.

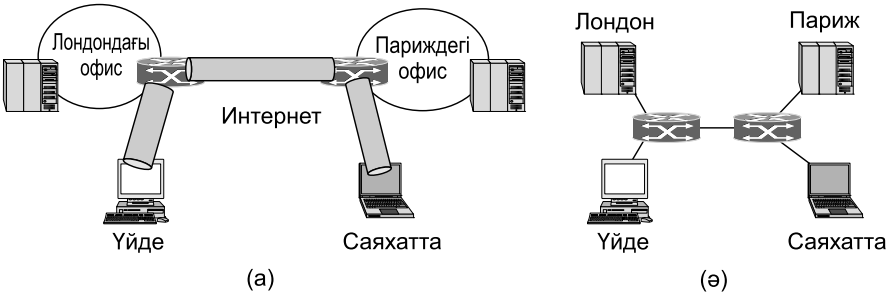
Жеке желілер жақсы жұмысы және жақсы қорғалғандығымен ерекшеленеді. Егер тек бөлінген тораптар қолжетімді болса, онда трафиктің ағып кету мәселесі орын алмас еді және деректерді ұрлап алу үшін бұзақыға торапқа физикалық түрде қосылу керек болар еді, ал бұл оңай шаруа емес. Екі нүкте арасындағы T1 бір бөлінген арнаны жалға алу мыңдаған долларды (айына!) құрайды, ал T3 арнасын жалға алу одан да қымбатқа түседі. Жалпы қолжетімді тасымалдау желісі пайда болған кезде, ал кейінірек Интернет, көптеген компанияларда деректерді тасымалдау (мүмкін дауысты да) үшін осы желіні пайдалану табиғи ынтасы туындады. Сонымен бірге, жеке желідегідей қорғалғандық қасиетін жоғалтқысы келмеді.

Бұл ой жақын арада **вуртуалды жеке желілердің (VPN – Virtual Private Networks)** құрастырылуына әкелді. Бұл әдеттегі жалпы қолжетімді желілер үстінде жұмыс істейтін, жеке желі қасиеттеріне ие оверлейлік желі. Олар «виртуалды» деп аталады, себебі мұндай желілер – иллюзия, сәйкесінше виртуалды арна – бұл нақты арна емес, ал виртуалды жады – нақты жады емес.

Соңғы кезде VPN-ді тікелей Интрернетте ұйымдастыру танымал болып келеді. Жалпы жағдайда әрбір офис желіаралық экранмен жабдықталады және әр офистар жұбы арасында *8.26 а-суретінде* көрсетілгендей Интернет арқылы туннелдеу жүргізіледі. Байланыс үшін Интернетті пайдаланудың тағы бір артықшылығы – туннелді талап бойынша құрастыруға болады және оған, мысалы, Интернетпен байланыс бар болса үйде немесе саяхатта жүрген қызметкер компьютерін қосуға болады. Мұнда иілгіштік бөлінген торапқа қарағанда жоғары, бірақ VPN бар компьютер тұрғысынан топология жеке желідегідей болып көрінеді, ол *8.26 ә-суретінде* көрсетілген. Жүйені іске қосқан кезде әр желіаралық экрандар жұбы қызметтер жиынтығы, режимдер, алгоритмдер

және кілттер тәрізді қорғаушы байланыс параметрлері жайлы келісуі тиіс. Егер туннелдеу режиміндегі IPsec қолданылатын болса, онда кез келген офистер жұбы арасындағы трафикті бір сенімді ағынға жинақтап, қорғалған байланыс орнатуға болады және осының арқасында деректер тұтастығы, құпиялық, тіпті трафикті сараптауға қарсы белгілі бір дәрежедегі иммунитетті қамтамасыз етуге болады. Көптеген желіаралық экрандарға виртуалды жеке желілермен жұмыс істеуге арналған арнайы құралдар енгізілген. Жүйені кейбір әдеттегі маршруттауыштарда да құрастыруға болады, бірақ желіаралық экрандар желілік қауіпсіздік жүйесінің бөлінбейтін бөлігі болғандықтан, интернет және компания арасында айқын шекара жүргізіп, туннелдеуді желіаралық экраннан бастап, желіаралық экранда аяқтау табиғи іс. Осылайша, табиғи және кең таралған комбинация – бұл желіаралық экрандар, виртуалды жеке желілер және туннелдеу режиміндегі IPsec және ESP.

Қорғаушы байланыс орнатылғаннан кейін деректерді тасымалдау басталады. Интернетте жұмыс істейтін маршруттауыш тұрғысынан VPN туннелі арқылы өтетін десте – бұл әдеттегі десте. Оның басқалардан жалғыз айырмашылығы – IP тақырыптан кейін IPsec тақырыбының болуы. Қосымша тақырыптар тасымалдау үдерісіне әсер етпейтін болғандықтан, маршруттауыштарды IPsec тақырыбы қатты алаңдатпайды.



8.26-сурет. а – Виртуалды жеке желі, ә – желі ішінен көрінетін топология

Танымал болып келе жатқан тағы бір мүмкіндік – бұл VPN-ді интернет-провайдер көмегімен жүзеге асыру. MPLS-ті пайдалану арқасында (5-тарауда талқыланған) VPN трафигі үшін компания офистері арасындағы жол интернет-провайдер желісі арқылы орнатылуы мүмкін. Бұл жолдар VPN трафикті басқа интернет-трафиктерден бөледі және белгілі бір өткізгіштік қабілеттілікке немесе басқа қызмет сапасына кепілдік береді.

VPN-нің негізгі артықшылығы – ол барлық тұтынушы программалық жабдықтамасы үшін ашық. Қорғалған байланысты орнату және басқаруды желіаралық экран жүзеге асырады. Желі бапталуында ісі бар жалғыз адам – бұл желілік шлюздерді құрастырып, қолдауға міндетті жүйелік әкімшілік немесе MPLS жолдарын қолдауға тиісті интернет-провайдер әкімшілігі. Басқалар

үшін виртуалды жеке желі бөлігінің бөлінген торап негізіндегі жеке желіден айырмашылығы жоқ. VPN жайлы толығырақ Lewis (2006) басылымынан оқуға болады.

8.6.4. Сымсыз желілердегі қауіпсіздік

Логикалық тұрғыдан толығымен сенімді, яғни VPN, желіаралық экраннан тұратын және іс жүзінде електен су аққандай ағатын жүйені құрастыру ғажайып қарапайым екен. Мұндай жағдай желіде желіаралық экран үстімен деректерді екі бағытта да, радиосигналдар көмегімен тасымалдайтын сымсыз машина бар болғанда орын алуы мүмкін. 8022.11 тәрізді желілердің әрекет ету радиусы бірнеше жүздеген метрді құрауы мүмкін, сондықтан ақпаратты ұстап алғысы келген зиянкес фирма алдындағы автотұраққа келіп, машинада эфирде естілгеннің барлығын жазып отыратын, 802.11-ді қабылдағыш-жөнелткіші бар ноутбукты қалдырып, серуендеп қайтуы мүмкін. Кешке қарай қатты дисктен көптеген маңызды ақпаратты табуға болады. Теорияда бұлай болуы мүмкін емес. Шындығында, теорияда банкттерді тонау да орын алмауы керек.

Қауіпсіздіктің көптеген мәселелері үшін өз өнімінің тұтынушыға қатысты достық жағдайында болуына талпынып, сымсыз базалық станцияларды (қолжеткізу нүктесін) құрастырушыларға рахмет айту керек. Әдетте, егер тұтынушы өз құрылғысын сөмкесінен шығарып, тоққа қосқан кезде ол бірден жұмыс істей бастайды және әрекет ету аймағы шеңберіндегі барлық радиотасымалдағыштар ол айтқан кез келген құпияны есті алады. Егер ол құрылғыны сонан кейін Ethernet қосса, жергілікті желі арқылы өтетін бүкіл трафик жақын арадағы машинада тұрған ноутбукпен ұстап алынуы мүмкін. Сымсыз желі – бұл шындыққа айналған тыңшы арманы: ақпарат өзі қолға келеді, тек ұстап үлгер. Яғни, сымсыз желілерді қауіпсіздік мәселесі әлдеқайда өткір мәселе. Біз бұл бөлімде осындай жүйелерді қауіпсіздендіру тәсілдерін қарастырамыз. Қосымша ақпаратты Nichols және Lekkas (2002) басылымынан табуға болады.

802.11 желілеріндегі қауіпсіздік

802.11 стандартының **802.11i** деп аталатын бөлігі сымсыз түйінге басқа сымсыз түйіндер жұбы жөнелткен мәлімдемені оқуға немесе қандай да бір басқа жолмен араласуға тыйым салатын, арналық деңгей қауіпсіздік хаттамасын сипаттайды. Ол **WPA2 (WiFi Protected Access 2 – WiFi қорғалған қолжеткізу версия 2)** деген сауда атымен жүреді. Қарапайым WPA – бұл 802.11i стандарты ішкі жиыны жүзеге асыратын аралық схема. Оның орына WPA2 пайдаланған дұрыс.

Біз 802.11i қысқаша сипаттаймыз, бірақ алдымен оның 802.11 қауіпсіздік хаттамаларының бірінші буынын, **WEP (Wired Equivalent Privacy – сым-**

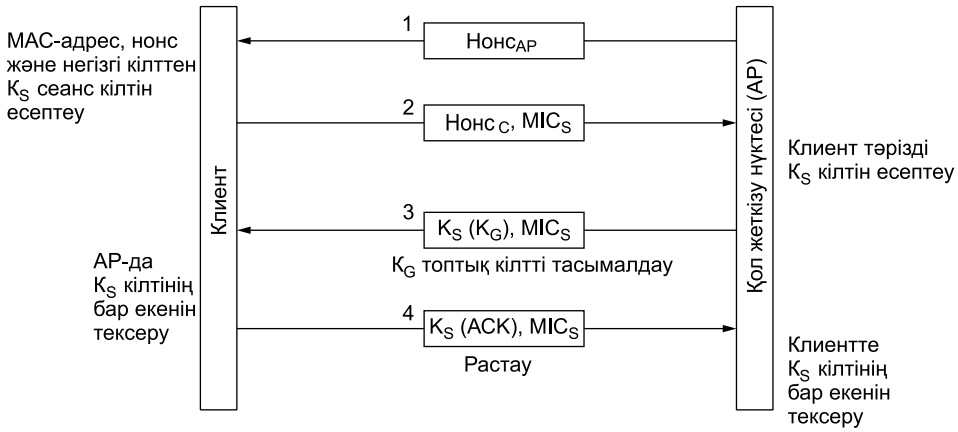
ды желілерге эквивалетті құпиялық) алмастыратынын айта кетейік. WEP-ті желілік стандарт комитеті құрастырған болатын. Бұл NIST AES-ті таңдап алған әдістен ерекше. Нәтижесі қиратушы күшпен тең болды. Не дұрыс болмады? Іс жүзінде барлығы да қауіпсіздік тұрғысынан болды. Мысалы, WEP құпия деректі XOR көмегімен ағындық шифр алынатындай етіп шифрлады. Өкінішке орай, кілтпен әлсіз жұмыс істеу, бұл шығыс деректердің бірнеше рет пайдаланылғанын білдіреді. Осылайша, оларды кері шифрлау өте жеңіл болды. Тағы бір мысал ретінде, тұтастықты тексеру 32-биттік CRC-ке негізделгенін айта кету керек. Бұл тасымалдау қателіктерін анықтау тиімді код, бірақ зиянкесті анықтауға арналған криптографиялық күшті механизм емес.

Осы және басқа да құрастыру кемшіліктері WEP-ті атына жеңіл кір келтіруші болды. Біріншісі іс жүзінде WEP сынғандығын бейнеледі. Бұл Адам Стабблфилд AT&T компаниясының жас қызметкері болған кезде орын алған еді (Stubblefield және басқалар, 2002). Ол Fluhrer және басқалар (2001) жазған шабуылды, уақыттың көп бөлігі менеджерлерді экспериментте пайдалану үшін оған WiFi карта сатып алып беруге иландыруға кеткен бір апта ішінде жүзеге асырып, тексере алды. WEP құпия сөздерін бір минут ішінде сындыра алдатын программаларға қазір еркін қолжеткізуге болады, сондықтан WEP пайдалану көтермеленбейді. Ол әдеттегі қолжеткізуге шектеу қояды, бірақ ешбір нақты қауіпсіздік формасын қамтамасыз етпейді. 802.11i тобы дереу жиналды, WEP сынғандығы белгілі болды. Нәтижесінде 2004 жылдың маусым айындағы формалды стандарт пайда болды.

Біз енді дұрыс бапталып, дұрыс пайдаланған жағдайда нақты қауіпсіздікті қамтамасыз ететін 802.11i сипаттаймыз. WPA2 қолданылатын екі әдеттегі сценарий бар. Біріншісі, компанияның сәйкестендіруді жүзеге асыратын, тұтынушы аттары және клиенттің желіге қолжеткізуге құқығы бар екендігін анықтайтын құпия сөздерді сақтайтын жеке сервері болған жағдайдағы корпоративтік пайдалану. Бұл жағдайда клиенттер өзін сәйкестендіріп, желіге ену үшін стандартты хаттамаларды пайдаланады. Негізгі стандарттар – бұл қолжеткізу нүктесі клиентке сәйкестендіру сервермен сұхбаттасып, нәтижесін бақылауға мүмкіндік беретін **802.1X** және клиент және сәйкестендіру серверінің қалай әрекеттесетінін сипаттайтын **EAP (Extendable Authentication Protocol – сәйкестендірудің кеңейтілген хаттамасы)** (RFC 3748). EAP орта болып келеді, ал қалған стандарттар хаттама мәлімдемесін анықтайды. Біз бұл алмасудың егжей-тегжейіне тоқталмаймыз, қысқа шолуда оның мәні жоқ.

Екінші сценарий – сәйкестендіру сервері жоқ, үй жағдайындағы пайдалану. Оның орнына, клиенттер сымсыз желіге қол жеткізу үшін пайдаланатын, жалпы бірегей құпия сөз бар. Бұл жүйе сәйкестендіру серверіне қарағанда күрделі емес, сондықтан ол үй жағдайында, кішігірім фирмаларда қолданылады, бірақ сенімділік жағынан бірінші сценарийден қалыспайды. Негізгі айырмашылық – сәйкестендіру сервері бар болған кезде әр клиент басқа клиенттерге белгісіз, трафикті шифрлау кілтін алады. Жалпы бірегей құпия сөз болған кезде әр клиент үшін жеке кілт құрастырылады, бірақ барлығының құпия сөзі бірдей және қажет болған жағдайда олар бір бірінің кілтін біле алады.

Трафикті шифрлау үшін қолданылатын кілттер сәйкестендіріп танудың бір бөлігі ретінде есептеледі. Танып айыру клиент сымсыз желімен байланысқаннан кейін бірден жүргізіледі және егер ол сәйкестендіру серверінде бар болса, сәйкестендіру орындалады. Танып айыру басында клиентте не желінің ортақ құпия сөзі бар, не сәйкенстендіру серверінің құпия сөзі бар. Құпия сөз негізгі кілтті алу үшін пайдаланылады. Негізгі кілт дестелерді шифрлау үшін тікелей қолданылмайды. Әр пайдалану аралығы үшін жаңа кілт құрастырып, әртүрлі сеанстар үшін кілтті ауыстырып, негізгі кілтті құпия ұстаудың станрдартты криптографиялық тәжрибесі бар. Танып айыру кезінде нақты сеанс кілті есептеледі.



8.27-сурет. 802.11i-дегі төрт дестелік танып айыру және сеанстық кілтті генерациялау

Сеанс кілті *8.27-суретте* көрсетілген төрт дестелік танып айыру кезінде есептеледі. Біріншіден, **АР (Access Point – қолжеткізу нүктесі)** сәйкестендіру үшін кездейсоқ нөмір жөнелтеді. Осындай қауіпсіздік хаттамасында бір рет қолданылатын кездейсоқ нөмірлер **nonces (нонстар – уақытша мәндер)** деп аталады. Бұл «number used once» – «тек бір рет қолданылатын нөмір» сөз тіркесінің қысқартылған түрі. Клиент те өзінің жекеменшік бір реттік нөмірін таңдайды. Ол, K_s сеанс кілтін есептеу үшін уақытша нөмірді, MAC адресті және АР адресін, сонымен бірге негізгі кілтті пайдаланады. Уақытша нөмір ашық әдіспен жөнелтілуі мүмкін, себебі оның негізінде қосымша құпия ақпаратсыз кілтті есептеу мүмкін емес. Клиенттен келетін мәлімдеме тұтастықты тексерумен қорғалған, ол **MIC (Message Integrity Check – мәлімдеме тұтастығын тексеру)** деп аталады. Бұл тексеру сеанс кілтіне негізделген. АР сеанс кілтін есептегеннен кейін, MIC дұрыстығын және мәлімдеменің шын мәнінде клиенттен келгендігін анықтай алады. MIC – HMAC-тағыдай мәлімдемені сәйкестендіру кодының басқа аты. MIC аты жиі қауіпсіздік хаттамасының орнына, оны **MAC (Medium Access Control – ортаға қолжеткізуді бақылау)** адреспен шатастырмас үшін қолданылады.

Соңғы екі мәлімдемеде АР клиентке K_G ортақ кілтті береді және клиент мәлімдеменің тұпнұсқа екенін растайды. Мәлімдеме деректерін алу клиентке АР дұрыс сеанс кілтінің бар екендігіне көз жеткізуге көмектеседі және керісінше. Ортақ кілт 802.11 LAN бойынша трафикті жөнелту үшін қолданылады. Танып айыру нәтижесінде әр клиенттің өз шифрлау кілті болғандықтан, АР бұл кілттердің ешқайсысын сымсыз желі клиенттерінің барлығына деспе жөнелту үшін қолдана алмайды. Әр жеке көшірмені өз кілтімен әр клиентке жөнелту керек болады. Оның орнына ортақ кілт пайдаланылады, сондықтан кеңтаратылымды трафик бір рет жөнелтіліп, барлық клиенттер алады. Бұл кілт клиенттер желіден кетіп және қайта қосылған сайын жаңартылып отыруы тиіс.

Соңында, біз кілттердің қауіпсіздікті қамтамасыз ету үшін қалай қолданылатынын қарастырамыз. Құпиялықты, тұтастықты және сәйкестендіруді қамтамасыз ету үшін 802.11i екі хаттама қолданылуы мүмкін. WPA тәрізді хаттамалардың бірі **TKIP (Temporary Key Integrity Protocol – кілт тұтастығының уақытша кілті)** уақытша шешім болатын. Ол ескі және баяу 802.11 карталарының қауіпсіздігін қамтамасыз ету үшін құрастырылған болатын, сондықтан оның қауіпсіздігі WPA қарағанда жоғары. Алайда оны қазір сындыруға болады, сондықтан басқа ұсынылған хаттаманы пайдаланған дұрыс – **CCMP.** CCMP деген не? Бұл «Counter mode with Cipher block chaining message authentification code protocol – кері байланысты іліндіру режиміндегі сәйкестендіру хаттамасы бар санауышы режимі» қысқартылған түрі. Біз оны CCMP деп атайтын боламыз. Сіз оны өз қалауыңызша атауыңызға болады.

CCMP тікелей жолмен жұмыс істейді. Ол кілт және көлемі 128 биттік блок көмегімен AES шифрлауын пайдаланады. Кілт сеанс кілтінен шығарылады. Құпиялықты қамтамасыз ету үшін мәлімдемелер санауыш режиміндегі AES шифрланады. Біз бұл шифрлау режимін *8.2.3-бөлімінде* талқылағанбыз. Бұл режим әркез бірдей мәлімдемелерді бірдей биттер жиынтығына шифрлауды болдырмайды. Санауыш режимі санауышты мәлімдемені шифрлау үдерісіне араластырады. Тұтастықты қамтамасыз ету үшін мәлімдеме тақырыптар өрісімен бірге кері байланыс режиміндегі шифрмен кодталады және 128 биттің соңғы блогы MIC ретінде сақталады. Содан кейін мәлімдеме де (санауыш режимінде кодталған), MIC те жөнелтіледі. Клиент те, АР да бұл кодтауды орындай алады немесе сымсыз дестені алған кезде тексере алады. Кең таратылымды немесе топтық мәлімдемелер үшін бұл процедура топтық кілтті қолданумен орындалады.

Bluetooth жүйелеріндегі қауіпсіздік

Bluetooth жүйесінің әрекет ету радиусы 802.11 желілеріне қарағанда әлдеқайда аз, сондықтан зиянкес ғимарат жанындағы автотұрақта қалдырылған машинада ноутбукті қосып шабуыл жасай алмайды. Алайда, қауіпсіздік мәлімдемесі мұнда да маңызды. Мысалы, айталық, Алисаның компьютері Bluetooth стандартындағы сымсыз пернетақтамен жабдықталған. Егер қорғаныс жүйе-

сін орнатпаса, онда Труди стенаның арғы жағындағы офиста отырып Алисаның не теріп жатқанын, шығыс поштасымен қоса, ешбір қиындықсыз оқи алады. Егер сымсыз принтерге жақын жерге орналассаңыз, онда принтерге берілгеннің барлығын ұстап алуға болады (пошта және құпия қағаздарды қоса). Қуанышқа орай, Bluetooth-де Труди тәрізді адамдардың жоспарын бұзатын, жұмысшы қорғаныс схемасы бар. Біз төменде осы схеманың негізгі сипаттамаларына тоқталамыз.

Bluetooth қорғаныс жүйесі (2 версия және одан да кейінгілері) толық әрекетсіздіктен бастап, деректерді толығымен шифрлау және тұтастықты бақылауға дейін төрт режимде жұмыс істей алады. 802.11 жағдайындағыдай, егер қорғаныс жүйесі өшірілген болса (ескі құрылғылар үшін үнсіз келісім бойынша), онда ешбір қауіпсіздік жайлы айтуға болмайды. Көптеген тұтынушылар тықыр таянғанша қорғанысты іске қоспайды. Біз осындай көзқарастың ауылшаруашылық мысалын келтіре аламыз: атқораның есігі аттар жоғалғаннан кейін ғана жабылады.

Bluetooth бірнеше деңгейде қорғанысты қамтамасыз етеді. Физикалық деңгейде бұл үшін жиіліктің секірмелі өзгерісі қолданылады, себебі микро желіде пайда болған кез келген құрылғы жиілік секірісі тізбегін анықтай алуы тиіс, яғни құпия емес. Ақпаратты нағыз қорғау жаңадан қосылған бағынышты құрылғы басқарушы құрылғымен байланыс арнасы сұраған кезде басталады. Bluetooth 2.1 пайда болғанша екі құрылғы да алдын ала орнатылған жабық кілтті пайдаланады деп болжанды. Кейбір жағдайда ол екі құрылғыға да тігіледі (мысалы, мобильді телефонмен бірге сатылатын гарнитурада). Басқа жағдайда бір құрылғыда (мысалы, гарнитурада) кілт тігіледі, ал қосылатын құрылғыға (мысалы, мобильді телефон) тұтынушы кілтті ондық сан ретінде қолмен енгізуі тиіс. Осы түрдегі ортақ кілттер **ашушы кілті (passkeys)** деп аталады. Өкінішке орай, ашушы кілт, жиі «1234» ретінде кодталады немесе басқа да болжаулы мәндер және кез келген жағдайда бұл тек 104 нұсқасы бар, төрт орынды сан болады. Bluetooth 2.1-де құрылғы кодты он алты орынды диапозоннан таңдайды, бұл ашушы кілтті аз болжаулы етеді, бірақ әлі де сенімсіз.

Арнаны орнатар алдында бағынышты және басқарушы құрылғы ашушы кілттің бар екендігін анықтауы керек. Ашушы кілт жоқ болған жағдайда олар арнаның қандай болатындығы жайлы келіседі: шифрланған, тұтастықты бақылауы бар немесе басқаша. Содан кейін, ұзындығы 128 бит сеанс кілті таңдалады. Бұл кілттің кейбір биттері жалпыға қолжетімді болуы мүмкін. Мұндай жеңілдік жүйенің әртүрлі елдердің үкіметі енгізген және кілттің ұзындығы үкімет сындыра алатын ұзындықтан үлкен кілтті пайдалануға немесе экспорттауға тыйым салатын шектеулерге сәйкес болу үшін жасалған.

Шифрлау ағындық E0 шифрын және **SAFER+** қолданып, тұтастықты бақылауды пайдалану арқылы орындалады. Екеуі де дәстүрлі симметриялы кілтті блоктық шифрлар болып келеді. SAFER+-ті AES-те қолдануға тырысқан, бірақ кейіннен бас тартты, себебі ол тым баяу жұмыс істейді. Bluetooth-бен жұмыс істей AES пайда болғанша аяқталған болатын, кері жағдайда Rijndael алгоритмін пайдалануды ықтимал еді.

Ағындық (топтық) шифрды пайдаланып шифрлау үдерісі *8.12-суретте* көрсетілген. Ол жерде ашық мәтін ағын кілтіммен модуль-2 бойынша қосылатындығы көрсетілген. Нәтижесінде шифрланған мәтін алынады. Өкінішке орай, E0 алгоритмі де RC4 тәрізді өте әлсіз (Jacobsson және Wetzel, 2001). Осы кітаптың жазылу кезінде оның әлі сынбағанына қарамастан, сыну қаупі бүкіл G*S*M-тафикке қауіп төндіретін A5/1 шифрына ұқсастығы, мұңды ойларға итермелейді (Biryukov және басқалар, 2000). Көптеген адамдарға (сонымен бірге авторларға) «мысық-тышқан» ойынына ұқсас криптографтар және криптосараптаушылар арасындағы ойынында жиі соңғылардың жеңіске ие болатындығы ғажайып болып көрінеді.

Bluetooth байланысты тағы бір қауіпсіздік мәселесі – жүйе тұтынушыны емес тек құрылғыны сәйкестендіреді. Бұл Bluetooth құрылғысын ұрлап алған ұры құрбанының қаржылық және басқа да есепшоттарына қолжеткізуіне әкеледі. Осылай бола тұрса да, Bluetooth-гі қауіпсіздік жүйесі жоғары деңгейлерде жүзеге асырылған, сондықтан, тіпті қорғанысты сындыру арналық деңгейде орын алса да, әлі де үміт сақталады, әсіресе егер қосымша транзакцияны орындау үшін қандай да бір пернетақта түрінің көмегімен PIN-кодты қолдан енгізуді сұрайтын болса.

8.7. СӘЙКЕСТЕНДІРУ ХАТТАМАЛАРЫ

Сәйкестендіру (аудентификация немесе идентификация) – бұл үдеріс сұхбаттасушы адамның нақты өзін таныстырған адам екендігін анықтау тәсілдері. Қазіргі арам пиғылмен активті енуге талпынастары жиі кездесетін жағдайда, қашықтықтағы үдерістің тұпнұсқалығын тексеру өте күрделі есеп және криптографияға негізделген күрделі хаттамаларды қажет етеді. Осы бөлімде біз қорғалмаған компьютерлік желілерде қолданылатын бірнеше сәйкестендіру хаттамаларымен танысамыз.

Кейде сәйкестендіру және авторизацияны шатыстыратынын айта кету керек. Сәйкестендіру сіздің әңгімелесуші адамыңыздың нақты өзін-өзі таныстырған адам екендігімен байланысты. Авторизация рұқсат берумен байланысты. Мысалы, клиенттік үдеріс файлдық серверге: «Мен Скоттың үдерісімін, *cookbook.old* файлын өшіргім келеді», - дейді. Файлдық сервер келесі екі мәселені шешуі тиіс:

1. Бұл шын мәнінде Скоттың үдерісі ме (сәйкестендіру)?
2. Скоттың *cookbook.old* файлын өшіруге құқығы бар ма (авторизация)?

Тек осы екі мәселеге бірегей оң жауап алынғаннан кейін ғана сұралған әрекет орындалуы мүмкін. Ең бастысы, бірінші сұрақ. Сервер кіммен сөйлесіп тұрғанын анықтағаннан кейін, қолжеткізу құқығын тексеру үшін жергілікті кестелер немесе деректер базасының мазмұнын қарау жеткілікті. Осы себептермен біз бұл бөлімде сәйкестендіру мәселесіне аса көңіл бөлетін боламыз.

Барлық сәйкестендіру хаттамалары пайдаланатын жалпы схема келесі әрекеттерден тұрады. Алиса Бобпен қорғалған байланыс орнатқысы келеді немесе сенімді деп саналатын **Кілттер тарату орталығымен (KDC – Key Distribution Center).** Содан кейін әртүрлі бағытта тағы бірнеше мәлімдеме жөнелтіледі. Мәлімдемелер жөнелтілген сайын Труди атты бұзақы оларды ұстап алып, Алиса мен Бобты алдау немесе келіссөзді бұзу үшін, өзгертіп қайта жөнелтуі мүмкін.

Осылай бола тұрса да, хаттама өз жұмысын аяқтаған кезде, Алиса Бобпен сөйлескеніне, ал Боб Алисамен сөйлескеніне сенімді болулары тиіс. Сонымен бірге, көптеген хаттамаларда әңгімелесушілер болашақ ақпарат алмасуларда қолданатын құпия **сеанс кілтін (session key)** орнатады. Іс жүзінде бүкіл деректер алмасуы құпия кілті бар бір алгоритмнің көмегімен (AES немесе үштік DES) шифрланады, себебі олардың өнімділігі ашық кілтті алгоритмдерге қарағанда жоғары. Дегенмен, ашық кілтті алгоритмдер сәйкестендіру хаттамаларында және сеанс кілтін анықтау үшін кеңінен қолданылады.

Әрбір жаңа байланыс үшін жаңа, кездейсоқ таңдалған сеанс кілтін пайдалану мақсаты – тұтынушылардың жабық және ашық кілттерін пайдаланып жөнелтетін трафикті, зиянкес ұстап алуы мүмкін шифрланған мәтін санын, сонымен бірге үдеріс істен шыққан немесе ядро дампы бөтен қолға түскен жағдайда келетін залалды азайту. Сондықтан байланыс орнатылғаннан кейін үдерісте тек бір уақытша сеанс кілті сақталу керек. Барлық тұрақты кілттер мұқият өшірілуі тиіс.

8.7.1. Ортақ құпия кілтке негізделген сәйкестендіру

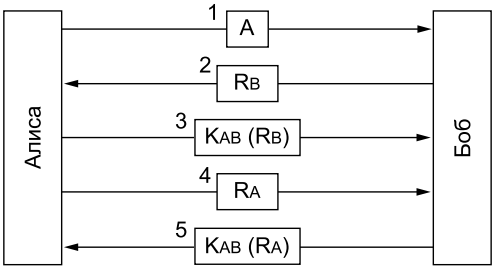
Біздің бірінші сәйкестендіру хаттамамызда біз Алиса және Бобтың ортақ құпия кілті бар деп болжаймыз. Бұл құпия кілт жайлы көзбе-көз кездескен кезде немесе телефон арқылы келісуге болады, кез келген жағдайда желі (қорғалмаған) арқылы емес.

Бұл хаттама негізінде көптеген сәйкестендіру хаттамаларында қолданылатын қағида жатыр: бір жақ екіншісіне кездейсоқ сан жөнелтеді, екінші жақ оны ерекше әдіспен түрлендіріп, нәтижесін қайтарады. Мұндай хаттамалар **шақыру-жауап (challenge-response)** түріндегі хаттамалар деп аталады. Осы және келесі сәйкестендіру хаттамаларында келесі шартты белгілер қолданылатын болады:

A және B – Алиса және Боб;
R_i – шақыру, мұндағы индекс жөнелтушіні білдіреді;
K_i – кілттер, мұндағы индекс кілт иесін білдіреді;
K_s – сеанс кілті.

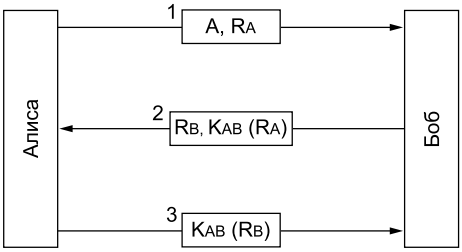
Біздің ортақ кілтті бірінші сәйкестендіру хаттамамыздың мәліметтер тізбегі *8.28-суретте* көрсетілген. Бірінші мәлімдемесінде Алиса Бобқа өзінің A жеке тұлға куәлігін оған түсінікті тәсілмен жөнелтеді. Боб, әрине, бұл мә-

лімдеменің Алисадан, әлде зиянкестен келгенін білмейді, сондықтан ол R_B үлкен кездейсоқ санды таңдап, оны Алисаға жауап ретінде ашық мәтінмен (2 мәлімдеме) жөнелтеді. Сонан кейін Алиса бұл мәлімдемені өзі және Боб үшін ортақ құпия кілтпен шифрлап, K_{FD} (R_B) шифрланған мәтінді 3 мәлімдememен жөнелтеді. Боб мәлімдемені көріп оның Алисадан келгенін түсінеді, себебі зиянкес K_{AB} кілтін білуі мүмкін емес және мұдай мәлімдемені құрастыра алмас еді. Одан бетер, R_B жауабы үлкен сандар кеңістігінде (мысалы, 128-биттік кездейсоқ сан) кездейсоқ таңдалғандықтан, зиянкестің бұл жауапты көруі және алдыңғы сеанстардағы жауаптарды көру ықтималдығы өте аз.



8.28-сурет. Шақыру-жауап хаттамасы кезіндегі екіжақты сәйкестендіру

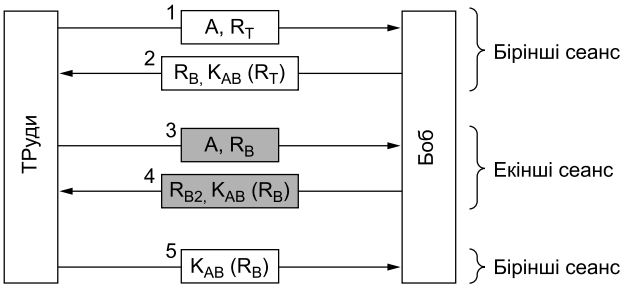
Осы уақыттан бастап Боб Алисамен сөйлесіп отырғанына сенімді, алайда Алиса әлі де күмәнданады. Зиянкес 1 мәлімдемені ұстап алып, R_B жауабын кері жөнелтуі мүмкін. Мүмкін, Боб тірі емес те шығар. Ары қарай хаттама симметриялы жұмыс істейді. Алиса шақыру жөнелтеді, ал Боб оған жауап береді. Енді екі жақ та сөйлескісі келген адаммен сөйлесіп отырғанына сенімді. Осыдан кейін олар, K_s, сеанстың уақытша кілтін орната алады. Сеанс кілтін ортақ K_{AB} кілтімен шифрлап бір-біріне жөнелтеді.



8.29-сурет. Қысқартылған екіжақтық сәйкестендіру хаттамасы

Бұл хаттамадағы мәлімдемелер санын *8.29-суретте* көрсетілгендей әр мәлімдемеде алдыңғы мәлімдемеге жауапты жаңа шақырумен біріктіріп қысқартуға болады. Мұнда Алиса өзі бірінші мәлімдемеде Бобқа шақыру жөнелтеді. Оған жауап беріп Боб осы мәлімдемен өз шақыруын қоса жөнелтеді. Осылайша, бес мәлімдеменің орнына үш мәлімдеме қажет болды.

Бұл хаттама алдыңғыдан жақсы ма? Бір жағынан, иә жақсы – ол қысқа. Бірақ мұндай хаттаманы пайдалауға ұсыныс берілмейді. Кейбір жағдайларда зиянкес бұл хаттаманы **айналық шабуыл (reflection attack)** деген атпен белгілі әдіспен шабуылдауы мүмкін. Жеке алғанда, Труди оны сындыруы мүмкін, егер ол бір мезгілде Бобпен бірнеше байланыс сеансын аша алса. Бұл жағдайды орын алуы ықтимал, егер, айталық Боб – бұл бір мезгілде банкоматпен бірнеше байланыс орнатуға мүмкіндік беретін банк болса.



8.30-сурет. Айналық шабул

Айналық шабуыл схемасы *8.30-суретте* көрсетілген. Ол Труди өзін Алиса деп жариялап, R_T шақыруын жөнелткеннен басталады. Боб әдеттегідей өзінің жеке R_B кілтімен жауап береді. Енді Труди тығырыққа тірелгендей болып көрінеді. Ол енді не істейді? Труди K_{AB} (R_B) білмейді ғой.

Зиянкес 3-мәлімдememен екінші сеанс ашып, шақыру ретінде Бобқа екінші мәлімдемеден алынған оның өз шақыруын жөнелтуі мүмкін. Боб оны шифрлап, кері 4-мәлімдememен K_{AB} (R_B) жөнелтеді. Енді Трудиде қажет ақпарат бар, сондықтан ол бірінші сеансты аяқтайды және екіншіні де үзеді. Боб енді зиянкестің Алиса екендігіне сенімді, сондықтан Трудиға Алисаның банк есепшотына қол жеткізіп, ешбір күмәнсіз Алисаның ағымдағы есепшотынан швейцария банкіндегі құпия есепшотқа ақша аударуға мүмкіндік береді.

Бұл әңгіменің ғибраты мынадай:

Дұрыс жұмыс істейтін сәйкестендіру хаттамасын құрастыру ойлағаннан әлдеқайда күрделі.

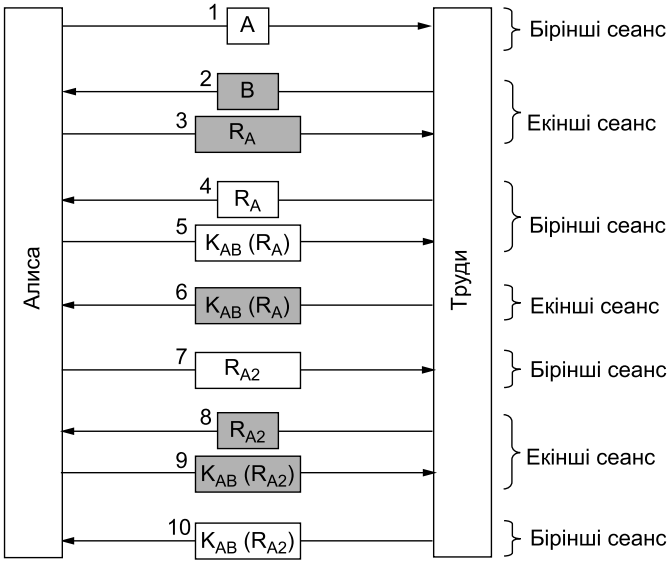
Жиі пайдалы және кең таралған қателіктерді болдырмауға көмектесетін төрт ортақ ережені келтірейік:

1. Сеанс бастамашысы өз жеке басын жауап беруші жақтан бұрын растауы тиіс. Бұл зиянкес өз жеке басын растағанша, өзіне бағалы ақпаратты алуға кедергі жасайды.
2. Екі бөлек ортақ құпия кілтті пайдалану керек: бірі сеанс бастамашысы үшін, ал екіншісі жауап беруші үшін, K_{AB} және K'_{AB}.

3. Бастамашы және жауап беруші жауапты әртүрлі қилыспайтын жиындардан таңдауы тиіс. Мысалы, бастамашы жұп нөмірлерді пайдалануы, ал жауап беруші – тақ нөмірлерді пайдалануы тиіс.
4. Хаттама, бірінші сеанстан ақпарат алынып (немесе керісінше), екінші сеанс іске қосылатын шабуылдарға төтеп бере алуы тиіс.

Егер осы ережелердің кем дегенде біреуі бұзылса, хаттама осал болып саналады. Келтірілген мысалда барлық төрт ереже де бұзылды, сәйкесінше бұл қиратушы нәтижеге әкелді.

Жоғарыдағы, *8.28-суретте* көрсетілген мысалға қайтып орлайық. Бұл хаттама айналық шабуылға төтеп береді деп сенімді түрде айтуға бола ма? Мүмкін. Жағдай өте тұрақсыз. Труди айналық шабуылды пайдаланып, біздің хаттаманы алдай алды, себебі ол Бобпен параллель сеансты ашуға мүмкіндік берді және оған өз жауабын қайтарып, адастырды. Егер Алисаның орнында, жалпы қолданыстағы, параллель байланыс сеанстарын қабылдаушы компьютер болса не болады? Бұл жағдайда Трудидің не істей алатынын көрейік.



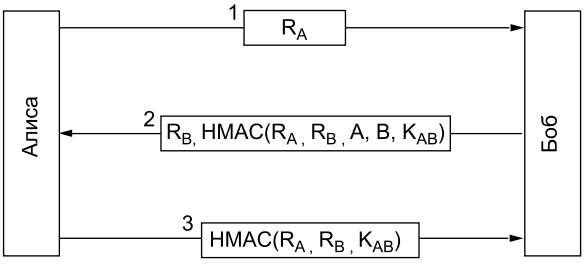
8.31-сурет. 8.28-суретте көрсетілген хаттамаға айналық шабуыл

Трудидің хаттаманы қалай сындыратынын түсіну үшін *8.31-суретке* назар аударайық. Алиса өзінің сәйкестендіру деректерін 1-мәлімдемеде хабарлады. Труди бұл мәлімдемені ұстап алып, 2-мәлімдемені жөнелтіп өз жеке сеансын іске қосады және Боб болып көрінеді. Біз мұнда бұрынғыдай, екінші сеанстың мәлімдемелерін сұр тік бұрыштармен бейнеледік. Алиса 2-мәлімдемерге 3-мәлімдememен «Сен Бобпын деймісің? Бұны растау керек» жауап береді. Осы жерде Труди тұйыққа тіреледі: ол өзінің Боб екендігін растай алмайды.

Труди енді не істей алады? Ол бірінші сеансқа қайтып оралады. Дәл осы кезде бірінші сеанста оның шақыруды жөнелту кезегі келеді. Осы кезде 3-мәлімдемемен алынған R_A жөнелтіледі. Алиса бұл шақыруға 5-мәлімдемемен жауап қайтарады, сөйтіп Трудиге екінші сеанстағы 6-мәлімдемені құрастыруға қажет ақпаратты береді. Труди енді сеансты таңдай алады, себебі ол екінші сеанстағы Алиса шақыруына дұрыс жауап берді. Бірінші сеансты жабуға болады, ал екінші сеансқа қандай да бір ескі санды жөнелтіп, нәтижесінде Алиса мақұлдаған байланыс сеансын алуға болады.

Алайда, Труди тым қиын адам, ол оны өзінің ары қарайғы әрекетімен дәлелдейді. Екінші сеансты тіркеуді аяқтау үшін қандай да бір ескі санды жөнелтудің орнына ол Алисаның 7-мәлімдемені жөнелткенін күтеді (оның бірінші сенсқа шақыруы). Әрине, Труди оған қалай жауап беруді білмейді, сондықтан ол R_{A2} 8-мәлімдеме ретінде жөнелтіп, тағы да айналық шабуыл жасайды. Алиса R_{A2} 9-мәлімдемеге шифрлайды. Труди бірінші сеансқа ауысады және Алисаға 10-мәлімдемеде сұраған санын жөнелтеді. Ол бұл санды қайдан алады? Сірә, Алисадан келген 9-мәлімдемеден болар. Осы сәттен бастап Труди өзін-өзі мақтай алады, оның қолында Алиса мақұлдаған екі байланыс сеансы бар.

Бұл шабуыл біз *8.30-суретте* көрген үш мәлімдемесі бар хаттамаға қарағанда біршама басқа нәтиже берді. Бұл жолы Труди Алиса мақұлдаған екі байланыс сеансын орнатты. Алдыңғы жағдайда Боб мақұлдаған бір байланыс сеансы орнатылған болатын. Тағы да, егер хаттама жоғарыда айтылған барлық төрт ережеге сәйкес болғанда шабуыл нәтижесіз болар еді. Әртүрлі шабуылдар түрінің және оларға қарсы тұру тәсілдерін Bird және басқалар (1993) басылымынан оқуға болады. Бұл басылымда, сонымен бірге дұрыс жұмыс істейтіндігін дәлелдеуге болатын хаттамалар құрастыру әдістері сипатталған. Алайда, бұл хаттамалардың ең қарапайымының өзі өте күрделі, сондықтан біз қазір хаттамалардың басқа класын (дұрыс жұмыс істейтін) қарастырамыз.



8.32-сурет. HMAC пайдаланып сәйкестендіру

Сонымен, жаңа сәйкестендіру хаттамасы *8.32-суретте* көрсетілген (Bird және басқалар, 1993). Біз бұнда IPsec зерттеген кезде талқылаған сол HMAC көреміз. Алиса алдымен Бобқа *nonce* R_A 1-мәлімдеме түрінде жөнелтеді. Боб жауап береді өзінің жеке *nonce* R_B таңдап, оны HMAC бірге жөнелтеді. HMAC

Алиса және Боб *nonce*-сінен, олардың идентификаторларынан, сонымен бірге K_{AB} ортақ жабық кілттен тұратын деректер құрылымын құрастырудан құралады. Содан кейін бұл құрылым хэш-функция көмегімен (мысалы, SHA-1) HMAC орналастырылады. Алиса 2-мәлімдемені қабылдағаннан кейін R_A (бұл мәнді ол өзі таңдаған), ашық мәтін ретінде алынған R_B, екі идентификатор және онсыз да белгілі жабық кілтінің бақытты иесі болады. Осы деректердің барлығына ие болғаннан кейін ол HMAC өз бетінше есептей алады. Егер ол мәлімдемедегі HMAC-пен сәйкес келсе, онда ол Бобпен сөйлесіп отырғанына күмән келтірмейді, себебі Труди K_{AB} білмейді, яғни жөнелтуге тиіс HMAC есептей алмайды. Алисаның Бобқа жөнелткен жауабында екі *nonce*-ден тұратын HMAC бар.

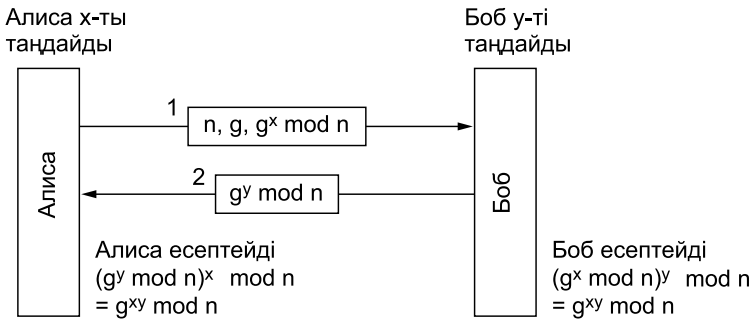
Сұрақ: Труди осындай хаттаманы сындыра ала ма? Жоқ, себебі ол *8.30-суреттегі* жағдайдағыдай, екі жақтың бірін де өзі таңдаған мәнді шифрлауға немесе оған хэш-функцияны пайдалануға мәжбүрлей алмайды. Екі HMac-та жөнелтуші таңдап алған мәннен тұрады. Труди оларды қадағалай алмайды.

HMAC-ты пайдалану – бұл жалғыз амал емес. Жиі қолданылатын балама схема, деректер элементтерін тізбекпен, шифр блоктарын іліндіру арқылы шифрлауға негізделген.

8.7.2. Ортақ кілтті орнату: Диффи-Хеллман кілт алмасу хаттамасы

Сонымен, біз Алиса және Бобта ортақ құпия кілт бар деп болжадық. Енді оларда бұл кілт жоқ деп жорамалдайық (себебі осы уақытқа дейін қолтаңба құрастыру және сертификаттарды таратудың әмбебап PKI инфроқұрылымы жасалған жоқ). Олар бұл кілтті қалай алады? Алиса Бобқа телефон шалып, кілтті телефон арқылы бере алады. Бірақ Боб «Сіз өзіңіздің зиянкес емес Алиса екендігіңізді қалай дәлелдей аласыз?» - деп сұрауы мүмкін. Олар әрқайсысы өз төлқұжатымен, жүргізуші куәлігімен және үш несие картасымен келетін кездесу ұйымдастыруға талпынуы мүмкін, бірақ жұмыс басты адамдар болғандықтан, айлап екеуінде қанағаттандыратын кездесу күнін белгілей алмауы мүмкін. Қуанышқа орай, бейтаныс адамдарға бір күнде ортақ құпия кілтті орнату мүмкіндігі бар, тіпті зиянкес әр мәлімдемені мұқият жазып отырса да.

Бұрын кездеспеген адамдарға ортақ құпия кілтті орнатуға мүмкіндік беретін хаттама **Диффи-Хеллман кілт алмасу хаттамасы (Diffie-Hellman key exchange)** деп аталады және келесідей жұмыс істейді (Diffie және Hellman, 1976). Алиса және Боб, *n* және *g*, екі үлкен қарапайым сан жайлы келіседі, мұнда *(n-1)/2*-де қарапайым сан, бұдан басқа *g* санына қосымша шарттар қойылады. Бұл сандар ашық болуы мүмкін, сондықтан олардың әрқайсысы *n* және *g* таңдап, оны бір-біріне ашық айта алады. Содан кейін Алиса үлкен (мысалы, екілік 1024-разрядты) x санын таңдап, оны құпия ұстайды. Сәйкесінше, Боб үлкен *y* құпия санын таңдайды.



8.33-сурет. Диффи-Хеллман кілтімен алмасу хаттамасы

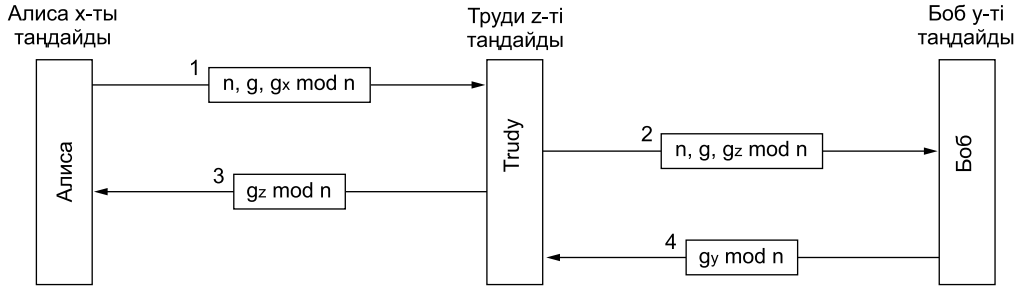
Алиса кілттермен алмасу хаттамасын *8.33-суретте* көрсетілгендей Бобқа *(n, g, g^x* mod *n)* жазылған мәлімдеме жөнелтуден бастайды. Боб Алисаға g^y mod *n* жазылған мәлімдememен жауап қайтарады. Енді Алиса Боб жөнелткен санды алып, оны *x* рет дәрежелейді, сөйтіп *(g^y* mod *n)^x mod n* алады. Боб дәл осындай есептеуді орындап, *(g^x* mod *n)^y mod n* алады. Модульдік арифметикаға сәйкес екі есептеу нәтижесі де g^{xy} mod *n* тең болуы керек. Осылайша, ғажайып таяқша ишаратымен Алиса мен Бобта g^{xy} mod *n* ортақ құпия кілт бар.

Әрине, зиянкес екі мәлімдемені де көреді. Оған *n* және *g* мәндері бірінші мәлімдемеден белгілі. Егер ол х жән у мәндерін есептей алса, ол құпия кілтті қолға түсіре алар еді. Өкінішке орай, g^x mod *n* және *n* біле отырып, х-тың мәнін табу өте қиын. Бүгінгі күнде өте үлкен сан модулінің дискретті логарифмін есептеу белгісіз.

Мысал үшін (тіпті, мүмкін емес) *n=47* және *g=3* мәндерін алайық. Алиса *x=8*, ал Боб *y=10* таңдайды. Екі санда құпия сақталады. Алисаның Бобқа жөнелткен мәлімдемесінде (47, 3, 28) сандары жазылған, себебі 3^8 *mod 47=28*. Боб Алисаға 17 санымен жауап береді. Алиса 17^8 *mod 47* есептейді және 4 алады. Боб 28^{10} *mod 47* және ол да 4 алады. Осылайша, бір-біріне тәуелсіз, Алиса және Боб құпия кілттің мәнін анықтады, ол 4-ке тең. Кілтті табу үшін зиянкеске 3^x *mod 47=28* теңдігін шешу керек болады. Мұны осындай кіші сандар үшін тек толық талдау үшін орындауға болады, бірақ ұзындығы бірнеше жүздеген бит сандар үшін емес. Қазіргі кезде белгілі алгоритмдердің барлығы өте баяу жұмыс істейді, тіпті параллель жұмыс істейтін жоғары жылдамдықты суперкомпьютерлердің өзінде де.

Диффи-Хеллман алгоритмінің осындай әдемілігіне қарамастан оның бір кемшілігі бар: Боб (47, 3, 28) сандарын алған кезде оның зиянкестен (Труди) емес, нақты Алисадан келгеніне қалай көз жеткізеді? Бұны білудің жолы жоқ. Өкінішке орай, Труди *8.34-суретте* көрсетілгендей Алиса және Бобты алдау үшін бұл жағдайда пайдаланып кетуі мүмкін. Мұнда, Алиса және Боб х және у сандарын таңдағанша, Труди өзінің кездейсоқ *z* санын таңдайды. Алиса Бобқа 1-мәлімдемені жөнелтеді. Труди оны ұстап алып, дұрыс *n* және *g* сандарын (ашық мәтінмен жөнелтілген) пайдаланып, оның орнына Бобқа

2-мәлімдемені жөнелтеді, бірақ *x* орнына өзінің *z* санын жазады. Ол Алисаға да кері 3-мәлімдемені жөнелтеді. Кейіннен Боб Алисаға 4-мәлімдемені жөнелтеді. Труди оны да ұстап алып өзінде сақтайды.



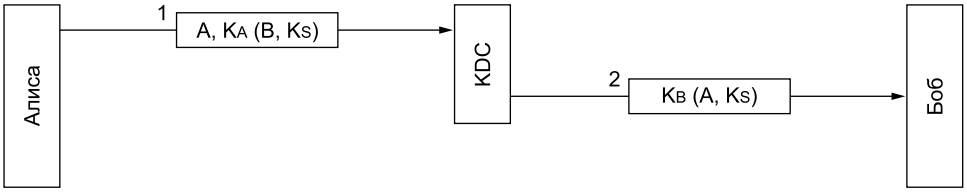
8.34-сурет. «Ортадағы адам» түріндегі шабуыл

Енді барлығы бөлуден қалған қалдықты есептеумен айналысады. Алиса g^{xz} mod n құпия кілтінің мәнін есептейді. Труди де осы есептеуді орындайды (Алисамен сұхбаттасу үшін). Боб g^{yz} mod n есептейді және Труди де осы есептеуді орындайды (Бобпен сұхбаттасу үшін). Шифрланған сеанста Алиса жөнелткен әрбір мәлімдемені Труди ұстап алып, сақтайды, қажет болса өзгертіп Бобқа жөнелтіледі (Трудидің тілегі бойынша). Кері бағытта да дәл осылай болады. Труди барлық мәлімдемені көреді және оларды өз қалауынша өзгерте алады, ал бұл уақытта Алиса және Боб бір-бірімен байланысу үшін оларда қорғалған байланыс арнасы бар деп болжайды. Зиянкестің мұндай әрекеті **«ортадағы адам» (man-in-the-middle attack)** түріндегі шабуыл деп аталады. Бұл шабуылдың тағы бір атауы – **«өрт тізбегі» (bucket brigade attack)**, себебі ертедегі су толы шелекті бір-біріне тізбек бойынша беруші өрт сөндірушілерді еске салады.

8.7.3. Кілт тарату орталықтары көмегімен сәйкестендіру

Сонымен, бейтаныс адаммен ортақ құпия кілт орнатуға қолжеткізейік дедік. Екінші жағынан мүмкін бұны мүлдем орындамаған дұрыс болар ма еді. Сізге *n* адаммен әрекеттесу үшін *n* кілтті сақтау керек болады. Қарым-қатынас шеңбері кең адамдар үшін кілтті сақтау қиын мәселеге айналады, әсіресе егер бұл кілттердің барлығын жеке пластик карталарда сақтау керек болса.

Бұл мәселені шешудің тағы бір жолы сенімді **кілт тарату орталығын (KDC, key distribution center)** ұйымдастыру. Мұндай схемада кез келген тұтынушыда KDC ортақ бір ғана кілт болады. Сәйкестендіру және сеанстық кілттермен орындалатын операциялар KDC арқылы өтетін болады. Екі жақтан және сенімді KDC тұратын, кілтті тарату орталығының көмегімен сәйкестендірудің қарапайым хаттамасы *8.35-суретте* бейнеленген.



8.35-сурет. KDC-орталығы көмегімен сәйкестендіру хаттамасының бірінші талпынысы

Хаттама негізінде жатқан ой қарапайым: Алиса K_S сеанс кілтін таңдайды да, KDC-орталығына Бобпен K_S кілтінің көмегімен сөйлескісі келетінін хабарлайды. Бұл мәлімдеме тек Алиса және кілтті тарату орталығы білетін K_A құпия кілтінің көмегімен шифрланады. Кілтті тарату орталығы мәлімдемені кері шифрлап, ол жерден Боб жеке тұлғасының идентификаторын және сеанс кілтін алады. Содан кейін орталық Алиса жеке тұлғасы идентификаторы және сеанс кілтінен тұратын жаңа мәлімдемені құрастырып, Бобқа жөнелтеді. Бұл мәлімдеме Боб және кілт тарату орталығына белгілі K_B құпия кілтімен шифрланады. Боб мәлімдемені кері шифрлап, Алисаның өзімен сөйлескісі және қандай кілтті қолданғысы келетінін біледі.

Бұл жағдайда сәйкестендіру өздігінен орын алады. KDC 1-мәлімдеменің Алисадан келгенін біледі, себебі басқа ешкім оны Алисаның құпия кілтімен шифрлай алмайды. Сәйкесінше, Боб 2-мәлімдеменің KDC-орталығынан келгенін біледі, себебі олардың ортақ құпия кілті өзінен басқа ешкім білмейді.

Өкінішке орай, бұл хаттаманың күрделі қателігі бар. Трудиге ақша керек, сол себептен ол Алиса үшін өзі орындай алатын қандай да бір заңды қызметті ойлап табады. Содан кейін Труди Алисаға жағымды ұсыныс жасайды да, осы жұмысты алады. Жұмысты орындағаннан кейін Труди сыпайылықпен, Алисаға қызметі үшін төлемақы төлеуін және ақшаны банк есепшотына аударуды сұрайды. Жұмыстың ақысын төлеу үшін Алиса өз банкирі Бобпен сеанс кілтін орнатады. Сонан кейін ол Бобқа Труди есепшотына ақша аударуды сұрап мәлімдеме жөнелтеді.

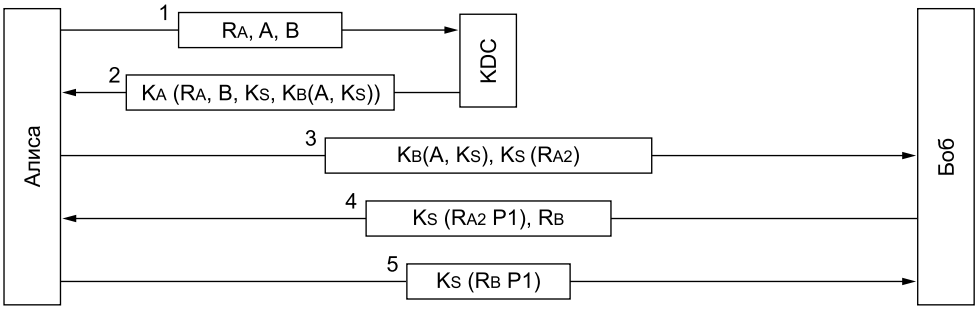
Осы уақытта Труди өзінің қара істеріне қайтып оралады. Ол 2-мәлімдемені (*8.35-суретті* қараңыз) және одан кейінгі ақша аударуға сұранысты көшіреді. Содан кейін ол бұл мәлімдемелерді Боб үшін қайта қалыпқа келтіреді. Боб мәлімдемелерді алып: «Алиса Трудиді қайта жұмысқа алған болу керек. Демек, Труди өз қызметін жақсы атқаратын болу керек», - деп ойлайды. Боб Алиса есепшотынан Труди есепшотына тағы да ақша аударады. Ақша аударуға елуінші сұранысты алғаннан кейін Боб өз офисынан жүгіріп шығып, Трудиді іздейді. Ол Трудиге өзінің өте сәтті бизнесін кеңейту үшін несиеге ақша ұсынбақ болады. Осындай мәселе **қайта қалыпқа келтіру арқылы шабуылдау (replay attack)** деген атқа ие болды.

Бұл мәселені шешудің бірнеше жолы бар. Бірінші шешім – әр мәлімдемеге

уақыт штампын қою. Ескірген мәлімдемелердің барлығы есепке алынбайды. Мұдағы бәленің басы – желідегі жүйелік сағаттарды үлкен дәлдікпен синхрондау мүмкін емес, сондықтан уақыт штампының қандай да бір жарамдылық мерзімі болу тиіс. Труди осы уақыт аралығында қайта мәлімдеме жөнелтіп, хаттаманы алдап кетуі мүмкін.

Екінші шешім – мәлімдемеге бірегей нөмір орналастыру, әдетте ол нонс (*8.6.4-бөлімдегі* nonce қараңыз) деп аталады. Екі жақтың әрқайсысы барлық алдыңғы нонстарды есте сақтауы және бұған дейін қолданылған нонсы бар мәлімдемелердің барлығын есепке алмауы тиіс. Алайда нонстарды өмір бойы сақтау керек болады, әйтпесе Труди бес жыл бұрын қолданылған мәлімдемені қайта қалыпқа келтіруге тырысады. Бұдан басқа, егер машина істен шығу салдарынан нонстар тізімін жоғалтып алса, онда ол қайта қалыпқа келтіру арқылы шабуылдауға сезімтал болады. Сақтау уақытын шектеу үшін уақыт штампы және нонстарды араластыруға болады, кез келген жағдайда хаттама әлдеқайда күрделенуі тиіс.

Сәйкестендірудің бұдан да күрделі тәсілі көпжақты шақыру-жауап хаттамасына негізделген. **Нидхэм-Шредер сәйкестендіру хаттамасы (Need-ham-Schroeder authentication protocol)** осындай хаттаманың жақсы мысалы бола алады (Needham-Schroeder, 1978). Бұл хаттаманың бір нұсқасы *8.36-суретте* көрсетілген.



8.36-сурет. Нидхэм-Шредер сәйкестендіру хаттамасы

Хаттаманың жұмысы Алисаның KDC-орталығына Бобпен сөйлескісі келетіндігі жайлы хабарлауынан басталады. Бұл мәлімдемеде нонсы ретінде R_A үлкен кездейсоқ сан жазылады. Кілттерді тарату орталығы билет деп аталатын, Алисаның кездейсоқ саны, сеанс кілті жазылған 2-мәлімдемені Бобқа жөнелтеді. Кездейсоқ R_A кездейсоқ санды жөнелтудің мақсаты, Алисаға 2-мәлімдеменің қайта қалыпқа келтіріліген емес, жаңа екендігін көрсету. Зиянкес (Труди) 1-мәлімдемеде Боб идентификаторын өз идентификаторына ауыстырып, KDC-орталығы 2-мәлімдеме соңындағы билетті K_B орнына K_T кілтімен (Труди кілті) шифрламас үшін, Боб идентификаторы да 2-мәлімдемеге орналастырылады. Екінші мәлімдеме Алисаға жеткенше зиянкес билетті

ауыстырып жібермес үшін, K_B кілтімен шифрланған билет шифрланған мәлімдеме ішіне орналастырылады.

Сонан кейін Алиса K_S сеанс кілтімен шифрланған билетті R_{A2} жаңа кездейсоқ санмен бірге Бобқа жөнелтеді. Боб 4-мәлімдемеде Алисаға нағыз Бобпен сөйлесіп отырғандығын дәлелдеу үшін $K_S(R_{A2}-1)$ жөнелтеді. Жәй $K_S(R_{A2})$ жөнелтудің мағынасы жоқ, себебі бұл санды зиянкес 3-мәлімдемеден ұрлап алуы мүмкін.

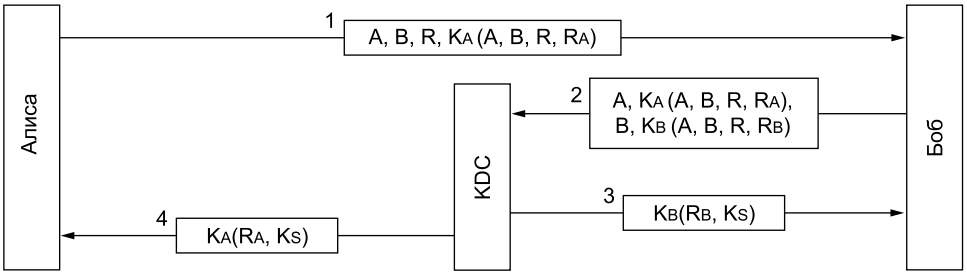
Алиса 4-мәлімдемені алып Бобпен сөйлесіп отырғанына және осы уақытқа дейін қайта қалыпқа келтірілген мәлімдеме болмағанына көз жеткізеді. Кездейсоқ R_{A2} санын жөнелтумен оған $K_S(R_{A2}-1)$ түрінде жауап алу арасында аз ғана уақыт өтті. Бесінші мәлімдеменің мақсаты – Бобты оның шын мәнінде Алисамен сөйлесіп отырғанына және осы байланыс сеансында қайта қалыпқа келтірілген деректердің жоқ екендігіне сендіру. Әрқайсысы қарсы беттің шақыруын құрастырып, оған жауап алуының арқасында, бұл хаттамада ертеде жазылған ақпаратты қайта қалыпқа келтіріп шабуылдау мүмкіндігі жоқ.

Хаттаманың бүкіл орнықтылығына қарамастан оның осал жері бар. Егер зиянкес қандай да бір жолмен ескі K_S сеанс кілтін қолға түсірсе, ол қолға түскен кілттің көмегімен 3-мәлімдемені қайта қалыпқа келтіріп, өзін Алисамын деп таныстырып, Бобпен жаңа сеансты бастай алады (Denning және Sacco, 1981). Бұл жолы зиянкес ешқандай қызмет көрсетпей-ақ, Алисаның есепшотынан ақша ұрлай алады.

Кейіннен Нидхэм және Шредер осы мәселені шешетін хаттама жариялады (Needham Schoeder, 1987). Осы журналдың, дәл осы басылымында Отуэй (Otway) және Рис (Rees) те осы мәселені әлдеқайда қысқа жолмен шешетін хаттамаларын жариялады. *8.37-суретте* Отуэй-Рис хаттамаларының сәл өзгертілген түрі бейнеленген.

Отуэй-Рис хаттамасында Алиса кездейсоқ нөмірлер жұбын құрастырудан бастайды: R – ортақ идентификатор және R_A – Алиса Боб шақыруы ретінде пайдаланатын екі нөмір. Бұл мәлімдемені алғаннан кейін Боб Алисаның шифрланған және сәйкес өз бөлігінен тұратын жаңа мәлімдеме құрастырады. K_A және K_B кілттерімен шифрланған мәлімдеменің екі бөлігі де, Алиса және Бобты сәйкестендіреді, сонымен бірге идентификатор және шақырудан тұрады.

Кілтті тарату орталығы мәлімдеменің екі бөлігіндегі R ортақ идентификаторының сәйкестігін тексереді. Егер зиянкес 1-мәлімдемедегі R-ді немесе 2-мәлімдеменің бөлігін алмастырған болса ол сәйкес келмейді. Егер екі R ортақ идентификаторы сәйкес келсе, KDC Бобтан алынған мәлімдемені айқын деп есептейді. Содан кейін ол K_S сеанс кілтін құрастырып, оны Алиса және Боб кілттерінің көмегімен шифрлап, өздеріне жөнелтеді. Сонымен бірге, әр мәлімдемеде оның зиянкес емес, KDC жөнелткенінің дәлелі ретінде қабылдаушының кездейсоқ саны жазылады. Осы уақытқа дейін Алиса мен Бобтың қолында сеанс кілті болады және олар ақпарат алмасуды бастай алады. Бірінші ақпарат алмасудан кейін-ақ олар өздерінде K_S сеанс кілтін бірдей көшірмесі бар екенін көреді және осымен сәйкестендіру үдерісін аяқтауға болады.



8.37-сурет. Отуэй-Рис сәйкестендіру хаттамасы (сәл өзгертілген)

8.7.4. Kerberos хаттамасының көмегімен сәйкестендіру

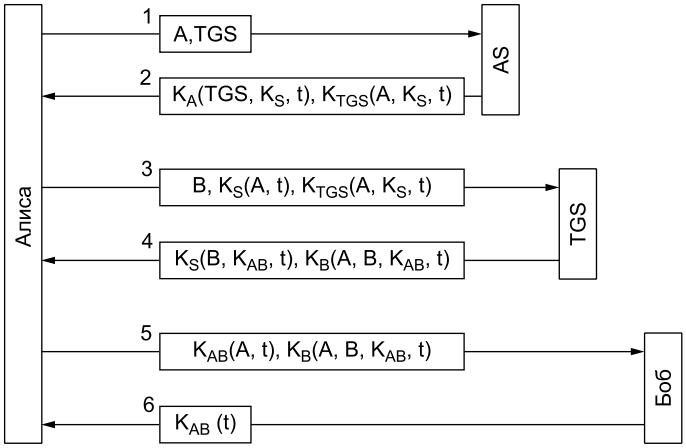
Көптеген нақты жұмыс істейтін жүйелерде (Windows 2000 және одан кейінгі версияларда) Нидхэм-Шредер хаттамасы нұсқаларының біріне негізделген **Kerberos** сәйкестендіру хаттамасы пайдаланылады. Хаттама көне грек мифінде Аидтың шығыс есігін қарауылдаушы үш басты ит Кербердің атымен осылай аталған. Кербер Аидқа бәрін кіргізіп, бірақ ол жерден ешкімді шығармаған. Kerberos хаттамасы Массачусетск технология институтында, жұмыс станциялары тұтынушыларына желілік ресурстарға сенімді қолжеткізуді қамтамасыз ету үшін құрастырылған болатын. Оның Нидхэм-Шредер хаттамасынан негізгі айырмашылығы – желі сағаттарының жақсы синхрондалған деп жорамалдануында. Хаттаманың бірнеше тізбектік версиясы құрастырылған болатын. Хаттаманың V5 версиясы өндірісте кеңінен қолданылады және RFC 4120 сипатталған. Ертедегі V4 версиясында күрделі қателіктер табылғаннан кейін толық бастартылды (Yu және басқалар, 2004). V5 версиясы V4 қарағанда жақсартылды – көптеген кішігірім өзгертулер бар және бірнеше жақсартылған сипатттамалары бар, мысалы, ол ескірген DES негізделмеген. Қосымша ақпаратты Neumann және Ts`o (1994) басылымынан қарауға болады.

Kerberos хаттамасының жұмысына Алиса жұмысшы (клиенттік) станциясынан басқа тағы үш сервер қатысады:

1. Сәйкестендіру сервері (AS, Authentication Server): желіге кірер алдында клиент жеке тұлғасын тексереді.
2. Билет беру сервері (TGS, Ticket Granting Server): «тұпнұсқалықты дәлелдейтін билет» береді.
3. Боб, яғни Алисаға қызмет көрсететін сервер.

AS сәйкестендіру сервері кілттерді тарату орталығы KDC-ке ұқсас, себебі оның әр тұтынушы үшін ортақ құпия сөзі бар. TGS билеттер беру серверінің жұмысы – басқа серверлерді билет иесі, шын мәнінде өзін таныстырған тұлға екендігін дәлелдейтін куәлік беру.

Сеансты бастау үшін, Алиса еркін, жалпыға қолжетімді жұмыс станциясы компьютеріне отырып, өзінің атын және TGS атауын енгізеді. Жұмыс станциясы *8.38-суреттегі* 1-мәлімдемеде көрсетілгендей, енгізілген атты ашық мәтінмен сәйкесттендіру серверіне жөнелтеді. AS сәйкестендіру сервері Алиса жұмыс станциясына сеанс кілтін және TGS билет беру сервері үшін $K_{TGS}(A, K_S, t)$ билетін қайтарады. Сеанс кілті тек Алиса кері шифрлай алатындай, Алиса құпия кілтімен шифрланады. Тек 2-мәлімдеме алынғаннан кейін ғана жұмыс станциясы Алисаның құпия сөзін сұрайды. Осы құпия сөз көмегімен 2-мәлімдеме кері шифрланып, сеанс кілті алынатын K_A кілті құрастырылады. Кері шифрлау орындалғаннан кейін жұмыс станциясы жадыда сақталған құпия сөзді бірден өшіреді. Егер жұмыс станциясында Алисаның орнына Труди тіркелуге талпынса, ол енгізген құпия сөз дұрыс емес болып шығады. Жұмыс станциясы мұны бірден анықтайды, себебі 2-мәлімдеменің стандартты бөлігі дұрыс емес болады.



8.38-сурет. Kerberos V5 хаттамасының жұмысы

Желіде тіркелгеннен кейін Алиса жұмыс станциясына файлдық сервермен, яғни Бобпен байланысқысы келетінін айтады. Бұл жағдайда жұмыс станциясы билет беру серверіне Бобпен хабарласуға билет беруді сұрап, 3-мәлімдемені жөнелтеді. Бұл сұраныстың кілтті элементі $K_{TGS}(A, K_S, t)$. Ол TGS-сервері құпия кілтінің көмегімен шифрланған және жөнелтуші жеке тұлғасын растау үшін қолданылады. Билеттер беру сервері Алиса және Боб пайдаланатын, K_{AB} сеанс кілтін берумен жауап қайтарады. Сервер Алисаға кілттің екі нұсқасын жөнелтеді. Бір кілт K_S сеанс кілтімен шифрланған, сондықтан Алиса оны оқи алады. Екінші кілт – бұл тағы бір билет, ол K_B Боб кілтімен шифрланған және Бобқа оны оқуға мүмкіндік береді.

Зиянкес 3-мәлімдемені көшіріп алып, оны қайта пайдаланбақ болуы мүмкін, бірақ оған мәлімдememен бірге жөнелтілетін *t* уақыт штампы кедергі

жасайды. Зиянкес бұл уақыт штампын жаңаға өзгерте алмайды, себебі ол Алиса билет беру серверімен әрекеттесу үшін қолданатын K_S сеанс кілтін білмейді. Зиянкес 3-мәлімдемені жылдам қайталағанның өзінде де ол жауап ретінде тек 4-мәлімдемені алады, бірақ оны ол бірінші жолы да, екінші жолы да кері шифрлай алмайды.

Бұдан кейін Алиса жаңа билет арқылы Бобпен сеанс орнату үшін оған K_{AB} кілтін жөнелте алады. Бұл мәлімдемелердің де уақыт штампы бар. Жауап ретінде алынатын 6-мәлімдеме (жаңа тексеру ретінде) Алисаның зиянкеспен емес, нақты Бобпен сөйлесіп отырғанын растайды.

Соңында, осы мәлімдемелер сериясымен алмасқаннан кейін, Алиса K_{AB} сеанс кілтін пайдаланып, Бобпен деректер алмаса алады. Егер бұдан кейін Алиса оған басқа сервер, мысалы, Кэрол (Carol, C) керек деп шешсе, онда ол билет беру серверіне 3-мәлімдемеге ұқсас мәлімдеме жөнелтеді, тек ондағы B-ны C-ға алмастырады (яғни, Боб идентификаторын Кэрол идентификаторына ауыстырады). TGS әп-сәтте K_S кілтімен шифрланған мәлімдememен жауап қайтарады. Бұл билетті Алиса Кэролға жөнелтеді, ол Кэрол үшін Алисамен сөйлесіп отырғанының кепілдігі болады.

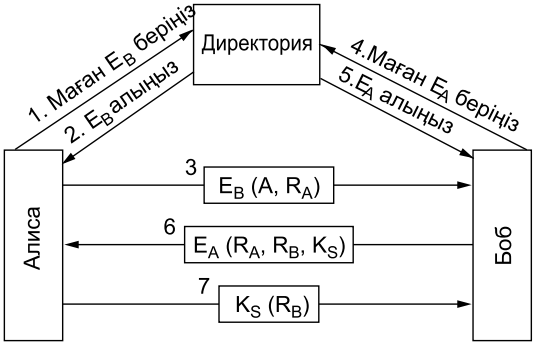
Бұл хаттаманың артықшылығы – енді Алиса желінің кез келген серверіне қолжеткізе алады және сонымен бірге оның құпия сөзі бірде-бір рет желі арқылы берілген жоқ. Іс жүзінде ол тек бірнеше миллисекундқа оның жұмыс станциясында пайда болды. Алайда, назар аударыңыз, әр сервер өзінің авторизациясын орындайды. Алиса Бобқа өз билетін көрсеткен кезде, бұл Бобқа көрсетілген билеттің нақтылығын растайды. Алисаның сервердегі қандай затқа қол жеткізе алатынын Боб шешеді.

Kerberos жүйесін құрастырушылар бүкіл әлем жалғыз бір сәйкестендіру серверіне сенетін болатындығын есепке алмағандықтан, олар әрқайсысының жеке сәйкестендіру сервері (AS) және билет беру сервері (TGS) болатын, бірнеше **ауданның (realms)** болуын қамтамасыз етті. Шалғай ауданда орналасқан серверге билет алу үшін Алиса өз TGS-нен шалғай аудандағы TGS қабылдайтын билет сұрауы керек. Егер шалғайдағы TGS жергілікті TGS тіркелген болса (жергілікті серверлер тәрізді), онда жергілікті TGS Алисаға шалғайдағы TGS-те қабылданатын билет береді. Бұдан кейін ол шалғайдағы TGS-тен осы аудан серверіне билет ала алады. Назар аударыңыздар, шалғайда орналасқан екі жақ бір-бірімен қорғалған байланыс сеансын орнату үшін, әрқайсы қарсы жақтағы TGS-ке сенуі тиіс. Кері жағдайда олар бірге жұмыс істей алмайды.

8.7.5. Ашық кілтпен шифрлау арқылы сәйкестендіру

Өзара сәйкестендіруді ашық кілтпен шифрлау арқылы да орындауға болады. Алдымен Алисаға Бобтың ашық кілтін алу керек болады. Егер PKI инфрақұрылымы ашық кілттерге сертификат беретін каталогтар серверінің

негізінде жүзеге асырылған болса, онда Алиса Бобтың сертификатын ала алады. Бұл *8.39-суретте* 1-мәлімдеме ретінде бейнеленген. Екінші мәлімдемеде жазылған жауап – бұл Боб ашық кілті бар X.509 сертификаты. Қолтаңбаның дұрыстығын тексеріп, Алиса Бобқа өз идентификаторы және нонсы жазылған мәлімдеме жөнелте алады.



8.39-сурет. Ашық кілт көмегімен өзара сәйкестендіру

Боб бұл мәлімдемені алған кезде, ол мәлімдеменің не Алисадан, не Трудиден (зиянкес) келгенін білмейді, бірақ ол сыр бермей каталогтар серверінен Алисаның ашық кілтін беруді сұрайды (4-мәлімдеме). Жақын арада ол 5-мәлімдememен оны алады. Содан кейін ол Алисаның R_A кездейсоқ саны, өзінің жеке нонсы R_B және ұсынылатын сеанс кілті K_S жазылған 6-мәлімдемені Алисаға жөнелтеді. Алиса қабылданған 6 мәлімдемені өзінің жабық кілтімен кері шифрлайды. Алиса ол жерден өзінің R_A кездейсоқ санын көреді және оған қуанады, бұл мәлімдеменің Трудиден емес Бобтан келгенін растайды, себебі Трудидің бұл санның мәнін анықтайтын тәсілі жоқ. Бұдан басқа, R_A кездейсоқ саны мәлімдеменің жаңа екендігін білдіреді. Алиса байланыс сеансын орнатуға келісіп, 7-мәлімдемені жөнелтеді. Боб өзі құрастырған сеанс кілтімен шифрланған өз R_B кездейсоқ санын көрген кезде, ол Алисаның 6-мәлімдемені алғанын және R_A мәнін тексергенін түсінеді. Боб қуанышты.

Зиянкес бұл хаттаманы қандай да бір жолмен алдай ала ма? Ол 3-мәлімдемені құрастырып, Бобты Алисаны тексеруге итермелей алады, бірақ Алиса өзі жөнелтпеген R_A санын көріп, сөйлесуді ары қарай жалғастырмайды. Зиянкес 7-мәлімдемені сенерліктей қайта жасай алмайды, себебі оған R_B шақыру мәні немесе K_S кілті мәні белгісіз және ол Алисаның жабық кілтінің көмегінсіз оларды анықтай алмайды. Сондықтан оның жолы болмайды.

8.8. ЭЛЕКТРОНДЫ ПОШТА ҚҰПИЯЛЫҒЫ

Шалғайда орналасқан екі тұтынушы арасында мәлімдеме тасымалдаған кезде, әдетте мәлімдеме жол бойында ондаған басқа машиналар арқылы өтеді. Олардың кез келгені өзі арқылы өткен поштаны оқып, көшіріп ала алады. Көптеген тұтынушылар бұл жайлы қанша ойласа да құпиялық жоқ. Осылай бола тұрса да, көптеген тұтынушылар электронды пошта арқылы жөнелтілген хаттарды бастығы да, тіпті үкіметте емес, тек тағайындалған адамдар ғана оқыса екен деп тілейді. Бұл тілектер кейбір топтар және жеке құрастырушылардың электронды поштаға криптография қағидаларын қолдануға ынталандырды. Келесі бөлімдерде біз кең таралған электронды поштаны қорғау жүйесі PGP танысамыз, сонымен бірге олардың тағы бірі – S/MIME жайлы жалпы түсініктеме береміз. Қосымша ақпаратты Kaufman және басқалар (2002), Schneier (1995) басылымдарынан қараңыз.

8.8.1. PGP

Біздің бірінші мысалымыз, **PGP (Pretty Good Privacy – жақсы құпиялық)** жүйесі, оны құрастырған бір адам, Фил Циммерман (Phil Zimmermann, 1995a, 1995b). Циммерман желідегі қауіпсіздік қолдаушысы және ол «Егер құпиялық бұзылса, демек ол заңбұзушыларға қолжетімді» дивизін ұстанады. 1991 жылы жарық көрген PGP жүйесі электронды пошта үшін құпиялықты, сәйкестендіруді, сандық қолтаңбаны және тығыздау тәрізді толық кешенді қамтамасыз етеді. Мұның барлығы жеңіл және ыңғайлы формада жүзеге асырылған. Бұдан бетер, барлық программаның бастапқы мәтіні толығымен Интернет арқылы таратылады. Өзінің сапасының, құнының (нөлдік) және әртүрлі платформаларда UNIX, Linux, Windows, Mac OS қоса, орнатылу қарапайымдылығының арқасында PGP жүйесі қазір кеңінен таралған.

PGP деректерді ұзындығы 128-бит кілтті пайдаланатын, **IDEA (International Data Encryption Algorithm – деректерді шифрлаудың халықаралық алгоритмі)** блоктық шифр көмегімен шифрлайды. Ол Швейцарияда, DES ескірген, ал AES әлі құрастырылмаған кезде ойлап шығарылған болатын. Тұжырымдамалық түрде IDEA DES/AES ұқсас: серия разрядтарын араластыру орындалады, бірақ функциялардың жүзеге асырылу егжей-тегжейі DES және AES-тен ерекше. Кілттерді басқару RSA көмегімен жүзеге асырылады, ал деректер тұтастығын қамтамасыз ету есебі үшін MD5 қолданылады. Бұл тәсілдердің барлығын біз ертеде талқылаған болатынбыз.

PGP жүйесінің құрастырылу тарихы бірінші күннен тым шиеленіскен (Levy, 1993). Ол Интернет арқылы еркін таратылғандықтан АҚШ үкіметі Циммерман әскери мүлікті экспорттауға тыйым салушы заңды бұзды деп жариялады. Осы іс бойынша тергеу жұмыстары бес жылға созылды, алайда күндердің бір күнінде екі негізгі себеппен жабылды. Біріншіден, Циммерман PGP-ді өз қолымен Интернетке қойған жоқ және адвокат қорғау позициясын айыптау-

шы өзі ешқашан, ешқандай затты экспорттаумен айналыспаған деп дәйектеді (сонымен бірге, сайт жасау экспорттаумен тең екендігін айта кету керек). Екіншіден, үкімет кенеттен, құпиялықпен байланысты жүктелетін программасы бар кез келген сайт, танк, су асты қайық, әскери ұшақ және ядролық қару тәрізді заттарды саудалау заңының әрекет ету аумағына кіретінін түсінді.

Шынын айтса, экспортқа тиісті заңдар біршама өрескел болып көрінеді. Үкімет программаны веб-парақта орналастыруды заңсыз экспорттау ретінде қабылдауға болады деп шешті және осы іс бойынша бес жыл Циммерманды ығыр қылды. Басқа бір жағынан, егер кімде-кім PGP бастапқы кодын С тілінде кітапта жарияласа (үлкен қаріппен, әр парақ соңында бақылау қосындысымен, бұл сканерлеуді жеңілдетеді) және содан кейін осы кітапты экспорттаумен айналысса, үкімен былқ етпес еді: заң бойынша кітап әскери мүлік болып саналмайды.

PGP жүйесі кенеттен қақтығысқан тағы бір мәселе, патенттік құқыққа қол сұғу болды. RSA патентінің иесі, RSA Security компаниясы PGP-де RSA тәсілін пайдалану патентке қол сұғу болып саналады деп жариялады. Бұл мәселе 2.6-версиясынан бастап шешілді. Бұдан басқа, PGP тағы бір патенттелген алгоритм, IDEA-ны пайдаланады, бұл да бастапқыда біраз сұрақтар туындатты.

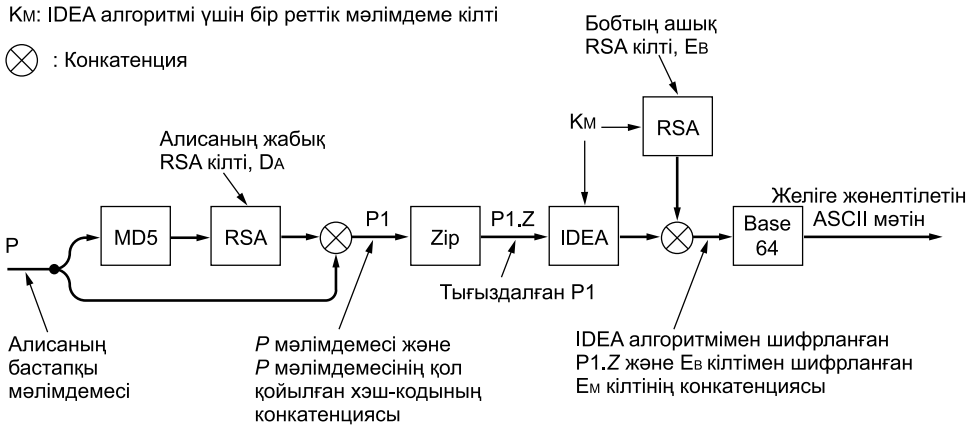
PGP – бастапқы коды ашық жүйе болғандықтан, оның әртүрлі топтар және жеке қызыққан адамдар құрастырған көптеген түрлендірулері пайда болды. Олардың кейбірулері қаруды экспорттау заңын айналып өтуге, басқалары патенттелген алгоритмді қолданбауға тырысты, ал үшіншілері PGP жабық кодты коммерциялық тауарға айналдыруға тырысты. Қаруды экспорттау заңының біраз босаңсығанына (осылай болса да, AES-ті пайдаланушы өнімдерді әлі де АҚШ-нан тыс экспорттауға болмайды), ал RSA патентінің жарамдық мерзімі 2000 жылдың қыркүйек айында аяқталғанына қарамастан, осы мәселелер салдары PGP-дің аттары әртүрлі, бірнеше үйлесімсіз версияларының пайда блып, таралуына әкелді. Төменде PGP-дің дәстүрлі нұсқасы талқыланады, ол ең ескісі және қарапайым. Тағы бір танымал нұсқасы, Open PGP RFC 2440 құжатында сипатталған. Тағы GNU Privacy Guard атап кетуге болады.

PGP жүйесінде жаңа криптографиялық алгоритмдер құрастырылмай, белгілі алгоритмдер әдейі қолданылған. Олардың барлығы әлемнің алдыңғы қатарлы криптосараптаушыларының түбегейлі тексеруінен өткен және бұл алгоритмдердің құрастырылу тарихы оларды әлсіздендіруге тырысатын қандай да бір мемлекеттік ұйымдарда қатысумен былғанбаған. Соңғы қасиет, үкіметке сенбейтіндер үшін өте үлкен артықшылық болып саналады.

PGP жүйесі мәтінді тығыздауды, құпиялықты және сандық қолтаңбаны қолдайды, сонымен бірге кілттерді басқарудың толық құралдарын ұсынады. Бірақ электронды пошта құралдары қолданылмайды. Ол көбіне кірісте ашық мәтінді алып, шығыста base64 форматында ұсынылған, қол қойылған шифрланған мәтін құрастырушы процессорға ұқсас. Әрине, бұл шығыс деректерін электронды пошта арқылы жөнелтуге болады. Кейбір жүзеге асырулар соңғы қадамда, нақты мәлімдемені жөнелту есебін жеңілдету үшін тұтынушы агентіне жүгінеді.

PGP жүйесінің қалай жұмыс істейтінін түсіну үшін *8.40-суреттегі* мысалды қарастырайық. Алиса сенімді жолмен Бобқа ашық мәтінмен қол қойылған P мәлімдемесін жөнелткісі келеді. Алиса мен Бобтың жабық (D_X) және ашық (E_X) RSA-кілттері бар. Айталық, олар бір-бірінің ашық кілттерін біледі делік. Кілттерді тасымалдау әдісін біз кейінірек қарастырамыз.

Алиса өз компьютерінде PGP программасын іске қосудан бастайды. PGP программасы алдымен оның P мәлімдемесін MD5 алгоритмінің көмегімен хэштейді, содан кейін алынған хэшті жабық D_A RSA-кілтінің көмегімен шифрлайды. Боб бұл мәлімдемені алғаннан кейін хэшті Алисаның ашық кілтімен кері шифрлап, оның дұрыстығына көз жеткізеді. Тіпті, қандай да бір зиянкес (Труди) бұл хэшті алып, Алисаның белгілі ашық кілтімен кері шифрлай алса да, MD5 алгоритмінің күші дәл осындай хэшпен басқа мәлімдеме құрастыру мүмкін еместігіне кепілдік береді (есептеу күрделілігіне байланысты).



8.40-сурет. Мәлімдеме жөнелту үшін PGP жүйесін пайдалану

Содан кейін хэш-код және мәлімдеме тұпнұсқасы біртұтас P1 мәлімдемесіне біріктіріледі. Мәлімдеме, Лемпель-Зива (Ziv және Lempel, 1977) алгоритмін пайдаланатын ZIP программасының көмегімен тығыздалады. Бұл қадам нәтижесін *P1.Z* деп атаймыз.

Сонан кейін PGP программасы Алисаға кездейсоқ мәтіндік жолды енгізуді сұрайды. K_M мәлімдемесінің 128-разрядты кілтін құрастыру кезінде IDEA алгоритмі үшін оның мазмұны да, енгізу жылдамдығы да есепке алынады. (PGP жайлы әдебиеттерде бұл кілт сеанстық деп аталған, бұл терминді дұрыс пайдаланбау, себебі ешқандай сеанс жоқ.) Содан кейін *P1.Z* K_M кілтінің көмегімен IDEA алгоритмімен, шифрланған кері байланыс режимінде шифрланады. Бұдан басқа, K_M кілті E_B, Бобтың ашық кілтімен шифрланады. Бұл екі компонент біріктіріліп, *7-тарауда* MIME стандарты жайлы сөз қозғаған кезде айтылған, base64 кодына түрлендіріледі. Нәтижесінде алынған мәлімдемде тек әріптер, цифрлар және +, *, = символдары болады. Бұл мәлімдемені RFC

822 стандартындағы хат денесіне енгізуге болатынын білдіреді және қабылдаушыға өзгеріссіз жететініне сенуге болады.

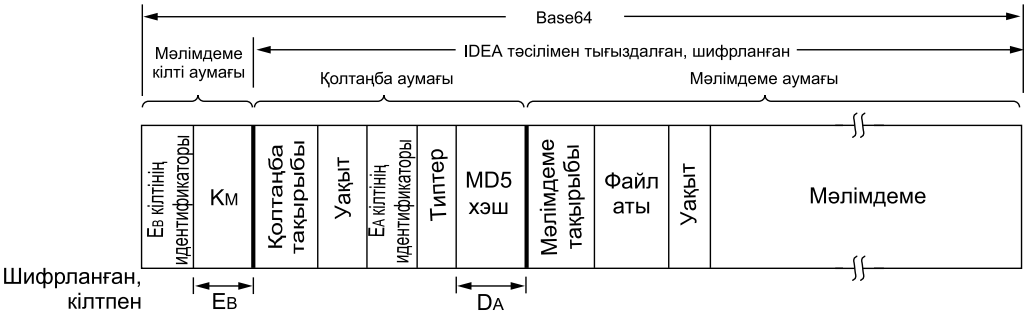
Боб мәлімдемені алып, base64 кері түрлендіруін орындайды және IDEA-кілтті өзінің жабық RSA-кілтімен кері шифрлайды. IDEA-кілтінің көмегімен ол мәлімдемені кері шифрлап, *P1.Z* алады. Zip-файлды ашып, Боб шифрланған хэш-кодты ашық мәтіннен ажыратады және оны Алисаның ашық кілтімен кері шифрлайды. Егер ашық мәтінді MD5 алгоритмімен өңдеу нәтижесінде дәл сондай хэш алынса, бұл P мәлімдемесінің Алисадан келгендігін білдіреді.

Мұнда RSA алгоритмінің тек екі жерде қолданылатынын айта кету керек: 128-разрядты MD5-хэшті және 128-разрядты IDEA-кілтті шифрлау үшін RSA алгоритмі баяу алгоритм, бірақ оған тек 256 битті шифрлау қажет, бұл тіпті көп емес. Бұдан бетер, бұл 256 бит жоғары дәрежеде кездейсоқ, сондықтан кілтті табу үшін зиянкеске біраз жұмыс істеуге тура келеді. Негізгі шифрлау IDEA алгоритмімен орындалады, ол RSA қарағанда әлдеқайда жылдам. Сонымен, PGP жүйесі құпиялықты, тығыздауды және сандық қолтаңбаны қолдайды және мұның бәрін *8.16-суреттегі* схемаға қарағанда тиімді орындайды.

PGP жүйесі төрт ұзындықтағы RSA кілтін қолдайды. Қажет ұзындықты тұтынушы өз қалауынша таңдай алады. Ұзындықтың келесі нұсқалары ұсынылады:

1. Шалағай (384 бит): бюджеті үлкен ұйымдар шифрды бір күнде сындыруы мүмкін.
2. Коммерциялық (512 бит): шифрды үш әріпті ұйымдар сындыруы мүмкін.
3. Әскери (1024 бит): бұл шифрды Жер бетінде ешкім сындыра алмайды.
4. Ғаламшар аралық (2048 бит): бұл шифрды бүкіл әлемде ешкім сындыра алмайды.

RSA алгоритмі тек екі кішігірім есептеуде қолданылатын болғандықтан, әрқашанда барлығыңыздың ұзындығы 2048 бит, Ғаламшар аралық кілтті қолданғандарыңыз дұрыс.



8.41-сурет. PGP-мәлімдеме

Дәстүрлі PGP-мәлімдеме форматы *8.41-суретте* көрсетілген. Сонымен бірге, көптеген басқа форматтарда қолданылады. Мәлімдеме үш бөліктен тұрады: кілт аумағы, қолтаңба аумағы және мәлімдеме аумағы. Кілт аумағында IDEA-кілттен басқа ашық кілт идентификаторы орналасады, себебі тұтынушыға бірнеше ашық кілт ұстауға рұқсат етілген.

Қолтаңба аумағында қазір бізді қызықтырмайтын тақырып орналасқан. Тақырыптан кейін уақыт штампы, жөнелтушінің ашық кілті орналасады. Бұл кілттің көмегімен қабылдаушы қолтаңба ретінде пайдаланылатын хэш-кодты кері шифрлайды. Одан кейін пайдаланылған шифрлау және хэштеу алгоритмдер идентификаторы орналасады (кейін пайдалану үшін, мысалы, MD6 немесе RSA2 құрастырылған кезде). Қолтаңба аумағының соңында шифрланған хэш-кодтың өзі орналасады.

Мәлімдеме аумағында да тақырып, үнсіз келісім бойынша файл аты, егер қабылдаушы қабылданған мәлімдемені дискте сақтаса, мәлімдеме құрастырылған уақыт штампы және соңында мәлімдеменің өзі орналасады.

PGP жүйесінде кілтпен жұмыс істеуге аса көңіл бөлінген, себебі бұл қорғау жүйелерінің осал жері. Әр тұтынушыда жергілікті екі деректер құрылымы бар: жабық кілттер жиынтығы және ашық кілттер жиынтығы (бұл жиынтықтарды кеде буда деп атайды). **Жабық кілттер будасында (private key ring)** жабық және ашық кілттен тұратын, бір немесе бірнеше жеке кілттер жұбы бар. Бірнеше кілттер жұбы, қандай да бір кілт беделі түсіп, қауіп төнген кезде тұтынушыға оларды дүркін-дүркін ауыстырып отыру үшін қолданылады. Кілтті өзгертуде, жаңа кілтті тасымалдау үшін қандай да бір шұғыл әрекет орындаудың қажеті жоқ. Әр кілт жұбының онымен байланысқан идентификаторы бар, сондықтан жөнелтуші қабылдаушыға мәлімдеме кілті қандай кілтпен шифрланғанын хабарлау жеткілікті. Мәлімдеме идентификаторы ашық кілттің кіші 64-разрядынан тұрады. Кілттер идентификаторлары арасында қайшылық туындамауына тұтынушының өзі жауап береді. Жабық кілттер, дискте оларды ұрлаудан сақтайтын арнайы құпия сөзбен (еркін ұзындықтағы) шифрланған.

Ашық кілттер будасы (public key ring) тұтынушы корреспонденттерінің ашық кілтінен тұрады. Олар әр мәлімдемемен байланысты, мәлімдеме кілтін шифрлау үшін қажет. Ашық кілттер жиынтығының әр жазбасы кілттен басқа, оның 64-разрядты идентификаторын және тұтынушының бұл кілтке деген сенімділік дәрежесін көрсететін белгіден тұрады.

Кілтке сенімділік дәрежесі, мысалы, оны алу жолынан тұрады. Айталық, ашық кілттер электронды хабарландыру тақтасында орналасқан (BBS). Труди хабаландыру тақтасын шабуылдап, онда орналасқан Бобтың ашық кілтін өз кілтіне ауыстырып қоюы мүмкін. Алиса ауыстырылған кілтті пайдаланбақ болған кезде, Труди Бобқа «ортадағы адам» шабуылын қолдануы мүмкін.

Мұндай шабуылдарды болдырмау үшін немесе, кем дегенде олар келтіретін шығынды азайту үшін Алисаға ашық кілттер жиынтығындағы Бобтың ашық кілтіне қаншалықты сенуге болатындығын білу керек. Егер Боб оған кілт жазылған дискті өз қолымен берсе, онда ол бұл кілтке ең жоғары

сенімділік белгісін қояр еді. PGP-дің PKI орталықтандырылған схемасынан айырмашылығы ашық кілттерді басқарудың тұтынушы бақылайтын, орталықтандырылмаған жолы, міне осыда.

Алайда, іс жүзінде ашық кілттер сенімді кілттер серверін сұрау арқылы алынады. Осы себептен, X.509 стандартталғаннан кейін PGP жүйесі дәстүрлі ашық кілттер будасы механизмімен бірге сертификаттарды қолдай бастады. PGP-дің қазіргі версияларының барлығында X.509 қолдауы бар.

8.8.2. S/MIME

Электронды пошта құпиялығын қамтамасыз етудегі IETF-тің келесі өнертабысы – **S/MIME (Secure/MIME – қорғалған MIME)** жүйесі. Ол RFC 2632-ден RFC 2643 құжаттарының барлығында сипатталған. Ол сәйкестендіруді, деректер тұтастығын және ақпараттың тұпнұсқалығын қамтамасыз етеді. Жақсы иілгіштігі бар, әртүрлі криптографиялық алгоритмдерді қолдайды. Атына қарап, кез келген типтегі мәлімдемелерді қорғайтындығы жағынан S/MIME-нің MIME-мен тығыз байланыста екенін байқауға болады. Көптеген жаңа MIME тақырыптары анықталған, мысалы, сандық қолтаңбалар үшін.

S/MIME-де сертификаттардың қатаң иерархиясы, бірегей басқару орталығы жоқ, бұл ертедегі PEM (Privacy Enhanced Mail – құпиялығы жоғары пошта) жүйесі үшін мәселе туындатады. Оның орнына тұтынушылар сенімді якорьлер жиынтығымен жұмыс істей алады. Сертификатын сенімді якорьмен тексерілгенше дұрыс деп саналады. S/MIME жүйесі біз қарастырған стандартты алгоритмдер және хаттамаларды пайдаланады, сондықтан біз оларға тоқталмаймыз. Толық ақпаратты RFC таба аласыздар.

8.9. ДҮНИЕЖҮЗІЛІК ӨРМЕКТЕ АҚПАРАТТЫ ҚОРҒАУ

Біз ақпаратты қорғауды қажет ететін екі маңызды аумақты – байланыс және электронды поштаны талқылауды аяқтадық. Аперитив және сорпа болды деуге болады. Енді ең басты тағамға келедік: Дүниежүзілік өрмекте ақпаратты қорғау. Зиянкестердің көпшілігі, өз арам пиғылдарын жүзеге асыру үшін WWW-те жұмыс істейді. Келесі бөлімдерде біз Дүниежүзілік өрмек қауіпсіздігіне қатысты кейбір мәселелерді қарастырамыз.

Бұл тақырыпты үш бөліке бөлуге болады. Біріншісі объектілер және ресурстарға қауіпсіз атау берумен байланысты. Екіншісі – сәйкестендірілген байланыс орнатумен. Үшіншісі – веб-сайт клиентке орындалатын код жөнелткенде не болатынымен байланысты. Мүмкін деген қауіптердің барлығын тізгеннен кейін біз осы сұрақтарды қарастыруға көшеміз.

8.9.1. Мүмкін қауіптер

Әр апта сайын газеттер Дүниежүзілік өрмектегі қауіпсіздік жайлы мақалаларды жариялайды. Жағдай шын мәнінде түңілерліктей. Іс жүзінде орын алған кейбір мысалдарды қарастырайық. Біріншіден, әртүрлі көлемдегі көптеген ұйымдардың үй парақтары хакерлер шабуылына душар болып, жалған парақтармен алмастырылғанын білеміз. (Компьютер әлемінде түсінігі аз, бірақ программистер жаргонын пайдаланғысы келген журналистер арқасында «хакер» (hacker) термині «сындырушы» деген мағынаға ие болды. Іс жүзінде «хакер» деп ұлы программистерді атаған. Программаны бұзушыларды сындырушылар (cracker) деп атаймыз.) Сындырылған сайттардың ішінде Yahoo!, АҚШ әскери күштерінің сайты, ОББ (ЦРУ), NASA, сонымен бірге *New York Times* газетінің сайттары болды. Көптеген жағдайларда сындырушылар сайттың тұпнұсқасын, қандай да бір күлкілі мәтін (әдетте, келемеж) жазылған өз парақтарына ауыстырған, бірер сағатан кейін сайтты қалыпқа келтіруге мүмкіндік туындаған.

Алайда, бұдан да күрделі шабуылдар болған. Көптеген сайттар, сервер игере алмайтын, жасанды асыра жүктелу салдарынан істен шыққан («қызмет көрсетуден бас тарту» түріндегі шабуыл, DoS). Көбінде мұндай шабуылдар, сындырушылар шабуылдап, еріксіз қылмыс жасауға мәжбүрлеген, бірнеше машинадан жасалып отырған («таратылған DoS», DDoS). Мұндай шабуылдардың кең таралғандығы соншалық, олар жаңалық болудан қалды. Осылай бола тұрса да, олар келтірген шығындар мыңдаған доллармен саналады.

1999 жылы швед сындырушысы Hotmail сайтына (Microsoft корпорациясы) кіріп, айна құрастырып, осының нәтижесінде тілек білдіргендердің барлығы осы сайттың кез келген тұтынушысының аты-жөнін енгізіп, оның барлық ағымдағы және мұрағаттағы поштасын оқуға мүмкіндік алған.

Ал Максим атты 19-жасар орыс сындырушысы электронды коммерциямен айналысатын сайттан 300 000 кредиттік карта нөмірін ұрлап алған. Сонан кейін ол карта иелеріне, егер 100 000 доллар бермесеңдер, кредиттік карталарының нөмірін Интернетте жариялаймын деген. Олар оның айтқанына көнбеген, сол кезде ол шынымен кредиттік карта нөмірлерін Интернетте жариялаған, бұл кінәсіз адамдарға айтарлықтай шығын әкелген.

Калифорния университетінің жиырма үш жасар студенті электронды пошта арқылы жаңалықтар агенттігіне, Emulex корпоррациясындағы үлкен шығындар және бас директорының жұмыстан кеткендігі жайлы жалған пресс-релиз жөнелткен. Бірнеше сағаттан кейін Emulex корпорациясы акцияларының биржалық құны 60% төмендеген, нәтижесінде акция ұстаушылары $2 миллиардтан қағылған. Зиянкес жалған мәлімдеме бермес бұрын өз акцияларын сатып, ширек миллион доллар пайда тапқан. Бұл жағдайда сындыру Дүниежүзілік өрмекте орын алған жоқ, бірақ компания сайтында орналасқан осындай хабарландырудың да дәл осындай әсері болар еді.

Өкінішке орай, мұндай мысалдарды тізбектеу бірнеше парақты алар еді. Біз енді істің техникалық жағын қарастыруға көшеміз. Барлық түрдегі қауіпсіздік мәселелеріне қатысты ақпаратты Anderson (2008a), Stuttard және Pinto (2007), Schneier (2000) басылымдарынан қараңыз. Интернеттен іздеуде жақсы нәтиже береді.

8.9.2. Ресурстарға қауіпсіз атау беру

Қарапайым заттан бастайық. Айталық, Алиса Бобтың веб-сайтына кіргісі келеді. Ол браузерде URL тергеннен кейін, бірнеше секунд ішінде парақ пайда болады. Бұл парақты, шын мәнінде Боб құрастырды ма? Мүмкін иә, мүмкін жоқ. Трудидің өз әрекеттерін қайта бастамағына кім кепіл. Мысалы, ол Алисадан шыққан дестелерді ұстап алып, зерттеуі мүмкін. Боб парағын алуға GET сұранысын тауып алып, оны өзгертіп Алисаға жөнелтуі мүмкін. Алиса ештеңені сезбес те еді. Одан бетер, Труди Бобтың электронды дүкеніндегі бағаларды да төмен бағаға өзгертіп, оның ұсыныстарын жағымды етіп көрсетуі де мүмкін. Енді Алисаның «Бобқа» өз несие картасының нөмірін жөнелту (жағымды бағамен бір нәрсе сатып алу үшін) ықтималдығы күрт өседі.

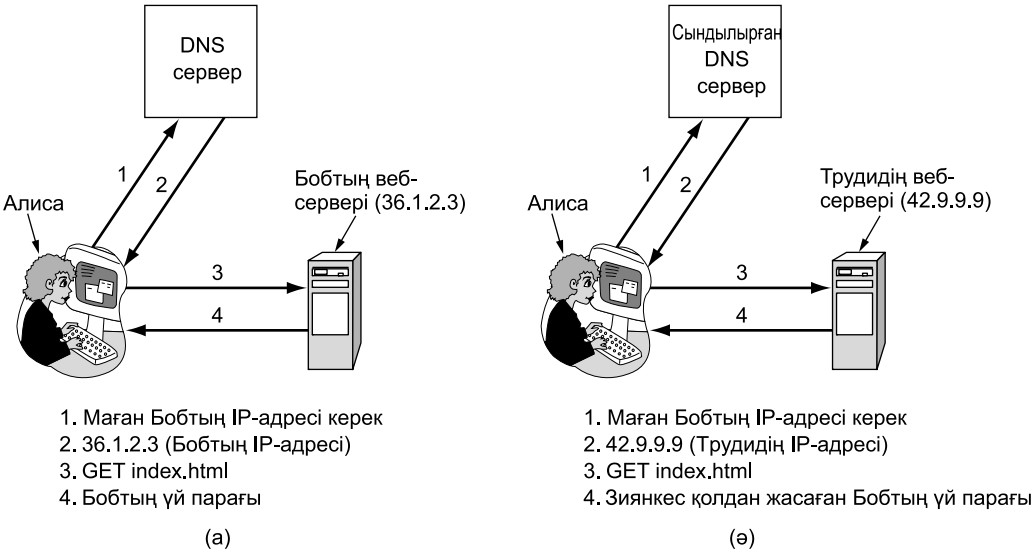
«Ортадағы адам» схемасының бір кемшілігі – Труди Алисаның шығыс трафигін ұстап алып, өз шығыс трафигін қолдан жасап аларлықтай жағдайда болуы тиіс. Іс жүзінде ол не Бобтың, не Алисаның телефон торабын (оптоталшықты кабельді тыңдау оңай есеп емес) тыңдауы тиіс. Бұл, әрине мүмкін, бірақ Труди тек ақылды және айлакер ғана емес, сонымен бірге жалқау. Ол Алисаны алдаудың жеңіл жолын біледі.

DNS-ті алдау

Айталық, Труди DNS жүйесін сындырып (мысалы, оның Алиса интернет-провайдерінде орналасқан DNS кэшіндегі бөлігін), Бобтың IP-адресін (мысалы, 36.1.2.3) өз IP-адресіне (мысалы, 42.9.9.9) өзгерте алады. Онда шабуыл жасауға болады. Қалыпты жағдайда барлығының қалай жұмыс істеу керектігі *8.42 а-суретінде* көрсетілген: (1) Алиса DNS қызметінен Бобтың IP-адресін сұрайды және (2) оны алады. (3) ол Бобтың үй парағын сұрайды және (4) оны да алады. Труди Бобтың IP-адресін өз IP-адресіне өзгерткеннен кейін біз *8.42 ә-суретінде* көрсетілген жағдайды аламыз. Алиса Бобтың IP-адресін іздейді, бірақ оның орнына зиянкес Трудидің IP-адресін алады, сондықтан Алисаның Бобқа арналған трафигінің барлығы Трудиға кетеді. Труди Алиса телефон торабында «қолтырауын» орнатып әуреленбей, «ортадағы адам» шабуылын ұйымдастыра алады. Оның орнына ол DNS аттар серверінде тек бір жазбаны өзгертеді. Бұның әлдеқайда жеңіл екеніне келісетін шығарсыздар.

Труди DNS-ты қалай алдады? Ал бұл қиын емес болып шықты. Егер егжей-тегжейіне тоқталмасақ бұл үдерісті келесідей сипаттауға болады: Тру-

ди Алисаның интернет-провайдері DNS-серверін Боб адресін іздеуге сұраныс жөнелтуге мәжбүрлейді. Өкінішке орай, DNS UDP пайдаланатын болғандықтан, сервер жауапты нақты жіберуші кім екенін біле алмайды. Труди осы қасиетті пайдаланып, күтілген жауапты қолдан жасайды, осылайша Бобтың IP-адресі жайлы DNS-сервер кэшіне жалған жазба енгізеді. Қарапайымдылық үшін біз Алиса провайдері бастапқыдан *bob.com* Бобтың веб-сайты жайлы ешнәрсе білмейді деп жорамалдаймыз. Егер мұндай жазба бар болса, онда зиянкес жазбаның жарамдылық мерзімінің өткенін күтіп, қайта шабуылдауы мүмкін (немесе басқа айла қолдануы). Труди өз шабуылын Алиса провайдеріне *bob.com* IP-адресін іздеуге сұраныс жөнелтуден бастайды. Сәйкес жазба жоқ болғандықтан, сервер өз кезегінде жоғарғы домен серверін сұрайды (*.com*). Бірақ, Труди бұл серверден бұрын, *bob.com* IP-адресі 42.9.9. деген жалған жауап жөнелтіп үлгереді. Біз білетіндей іс жүзінде бұл Трудидің адресі. Бұл жауап бірінші болып келгендіктен, деректер провайдер серверінің кэшіне енгізіледі, ал кейін келген дұрыс жауап есепке алынбайды. Жалған IP-адресті орнату **DNS-ті алдау (DNS spoofing)** деп аталады. Ал алдын ала жалған IP-адрес сақталған кэш **уланған кэш (poisoned cache)** деп аталады.



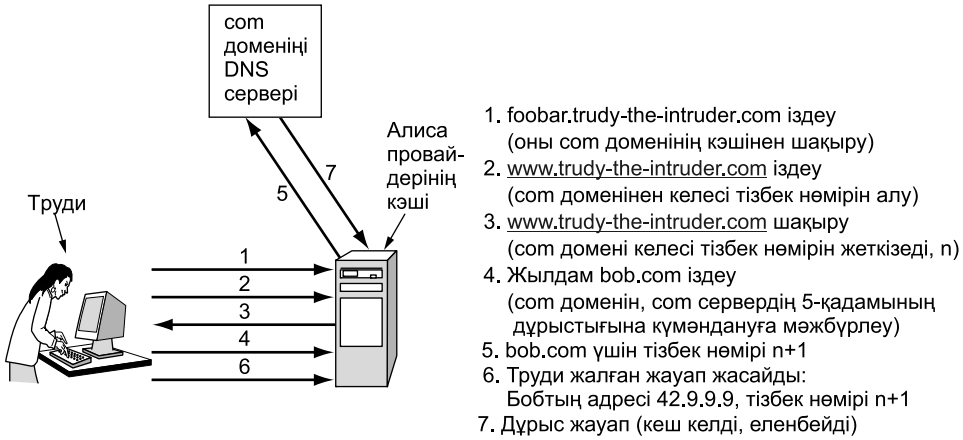
8.42-сурет. Қалыпты жағдай (а); DNS-ты сындырып, Бобқа қатысты жазбаны өзгерту шабуылы (ә)

Іс жүзінде барлығының осыншалықты қарапайым емес екендігін айта кету керек. Біріншіден, Алиса провайдері жауаптағы жоғары деңгей адресінің дұрыстығын тексереді. Бірақ Труди сәйкес өріске басқа ақпарат жазып, бұл кедергіден өтіп кете алады. Жоғары деңгей серверлерінің аты жалпыға қолжетімді екенін ескеріп, мұны жасау қиынға түспейді.

Екіншіден, DNS-сервер жауаптың қандай сұранысқа сәйкес екендігін түсіну үшін барлық сұраныстарға реттік нөмір қосылады. Алиса провайдерін алдау үшін Труди ағымдағы реттік нөмірді білуі тиіс. Оны білудің ең қарапайым жолы – бұл жекеменшік доменді тіркеу, мысалы, *trudy-the-intruder.com*.

Айталық, бұл доменнің IP-адресі де 42.9.9.9. Труди осы домен үшін DNS-сервер құрастырады: *dns.trudy-the-intruder.com*. Екі домен де бір компьютерде орналасқандықтан, оның IP-адресі де сол (42.9.9.9). Енді Алиса провайдерін Труди DNS-серверіне қызушылық тудыруға мәжбүрлеу керек. Бұны орындау қиын емес. Тек, мысалы, *foobar.trudy-the-intruder.com* сұрау жеткілікті, Алиса провайдеріне *.com* жоғары деңгей доменіне сұраныс жөнелтіп, Трудидің жаңа доменіне кім қызмет көрсететінін білуге тура келеді.

Міне, енді *dns.trudy-the-intruder.com* жазбасы провайдер кэшіне енген кезде шабуылды бастауға болады. Труди Алиса провайдерінен *www.trudy-the-intruder.com* сұраныс береді, ал ол жауап ретінде Труди DNS-серверіне сұраныс жөнелтеді. Осы сұраныста зиянкеске керекті реттік нөмір орналасады. Енді Труди кідіріссіз әрекет етуі керек: ол Алиса провайдерінің көмегімен Бобты іздейді және өз сұранысына жауап ретінде жалған жауап жөнелтеді «Адрес *bob.com*: 42.9.9.9». Бұл жалған жауаптың реттік нөмірі, алдында алынғаннан бірге жоғары. Шабуыл кезінде реттік нөмірі екі санға жоғары тағы бір жалған жауап жөнелтуге болады, сонымен бірге реттік нөмірі өсіп отыратын, бірнеше осындай «жауаптар» жөнелтуге болады. Олардың біреуінің мақсаты бізге белгілі. Алиса сұранысына жалған жауап келгеннен кейін ол кэшке орналастырылады, дұрыс жауап келген уақытта ол ескерілмейді, себебі сервер оны күтпейді.



8.43-сурет. Алиса провайдерін алдау

Сонымен, Алиса *bob.com* IP-адресін іздейді және оның 425.9.9.9 екенін біледі. Біз білетіндей, бұл өз бөлмесінен шықпай, «ортадағы адам» шабуылын ойдағыдай орындаған, Трудидің адресі. Осы нұсқаға сәйкес Трудидің орын-

даған қадамдар тізбегі *8.43-суретте* көрсетілген. DNS-серверлерді өз сұраныстарында реттік нөмірлерді инкрементеу орнына кездейсоқ идентификаторларды пайдалану арқылы бұл шабуылды болдырмауға болады. Өкінішке орай, бір «тесікті» жауып, біз басқасын ашамыз. Мысалы, ID бар жоғы он алты биттік, сондықтан компьютер орын ауыстырулар орындаған кезде олардың барлығымен жұмыс істеу оңай.

DNS-ті қорғау

Мәселенің бастысы, DNS, Интернет бірнеше жүздеген университеттерде жұмыс істеген, таза зерттеулік желі болған кезде құрастырылған болатын. Өмірдің бұл мерекесіне Алиса да, Боб та, Трудиде шақырылған емес. Ол кезде ақпаратты қорғау мәселесі болған емес, басты міндет Интернетті жұмыс істету болатын. Уақыт өте Интернет өмір сүрген орта қатты өзгерістерге ие болды, сондықтан 1994 жылы IETF, басты мақсаты – DNS қорғау жұмысшы тобын құрастырды. Бұл (жалғасып келе жатқан) жоба **DNSsec (DNS Security – DNS қорғау)** деген атпен белгілі. Жұмысшы топ жұмысының алғашқы нәтижесі RFC 2535 құжатында жарияланған. Өкінішке орай, DNSsec-ті әлі де үлкен масштабта жүзеге асыруға мүмкіндік болмай отыр, сондықтан көптеген DNS-серверлер зиянкестер шабуылына душар болып отыр.

DNSsec тұжырымдамасы өте қарапайым. Ол ашық кілтпен шифрлауға негізделген. DNS-тің әр аумағының (*7.2-суреттегі* терминмен) ашық және жабық кілттер жұбы бар. DNS-сервер жөнелтетін бүкіл ақпаратқа жөнелтуші аумағы жабық кілт көмегімен қолтаңба қойылады, сондықтан қабылдаушы жақ сәйкестікті тексере алады.

DNSsec үш негізгі қызмет түрін көрсетеді:

1. Деректер жөнелтілу орнын растау.
2. Ашық кілттерді тарату.
3. Транзакциялар және сұраныстарды сәйкестендіру.

Ең негізгісі – бірінші қызмет түрі. Оның көмегімен келіп жеткен деректер жөнелтушімен расталғандығы тексеріледі. Екінші қызмет түрі ашық кілттерді қауіпсіз сақтап, тарату үшін пайдалы. Үшінші қызмет қайта қалыпқа келтіру және серверді алдау шабуылдарынан қорғануға мүмкіндік береді. Назар аударыңыздар: мұнда құпиялық қамтамасыз етілмейді, мұндай мақсат жоқ, себебі DNS-тегі бүкіл ақпарат ашық деп саналады. DNSsec қатарға енгізу үдерісі бірнеше жылдар бойы жүргізілетін болғандықтан, қауіпсіздік жүйесімен жабдықталған және жабдықталмаған серверлерге өзара әрекеттесуге мүмкіндік беру маңызды. Бұл хаттаманы өзгертуге болмайды дегенді білдіреді. Енді бұл жүйені егжей-тегжейлі қарастырайық.

DNS жазбалары **RRSet (Resource Record Set – ресурстар жазбасының жиынтығы)** деп аталатын жиындарға топталады. Жиынға аттары, кластары және типтері бірдей жазбалар кіреді. Айталық, егер DNS аты бірінші және екінші IP-адреске сәйкес келсе, жиында бірнеше A жазбасы бола алады. Жиындар кейбір жаңа типтегі жазбалар есебінен кеңейтіледі (төменде талқыланады). Әрбір RRSet хэштеледі (мысалы, SHA-1 пайдаланып). Хэшке аймақ жабық кілті көмегімен қолтаңба қойылады (мысалы, RSA алгоритмі бойынша). Клиентке жөнелтілетін ақпарат бірлігі RRSet қолтаңбасы қойылған ақпарат болып саналады. Клиент ақпаратты қабылдап, оған қолтаңба қою үшін шын мәнінде жөнелтуші аймақ жабық кілт пайдаланылғанын тексере алады. Егер қолтаңба дұрыс болса, деректер қабылданады. Әр RRSet-тің жеке қолтаңбасы болғандықтан, жиындарды кез келген жерде кэштеуге болады, тіпті аса сенімді емес серверлерде олардың тағдыры үшін қам жемеуге болады.

DNSsec жүйесі бірнеше жаңа жазбалар типін енгізеді. Олардың біріншісі – бұл *KEY* жазбасы. Онда тұтынушы, хост немесе басқа да принципиал аймағының ашық кілті, қолтаңбаны генерациялау криптографиялық алгоритмі, тасымалдау хаттамасына ат беру және тағы бірнеше бит сақталады. Ашық кілт қорғалмаған түрде сақталады. X.509 сертификаттары үлкен болғандықтан, қолданылмайды. Алгоритм өрісіндегі MD5/RSA сәйкес келетін ұсынылатын мән 1-ге тең, басқа комбинациялар үшін басқа мәндер пайдаланылады. Хаттама өрісі IPsec немесе басқа байланысты қорғау хаттамасын пайдалануды нұсқауы мүмкін (егер ол қолданылса).

Екінші жаңа жазба түрі – *SIG*. Мұндай жазбада *KEY* көрсетілген алгоритмге сәйкес құрастырылған, қол қойылған хэш жазылады. Қолтаңба барлық RRSet жазбаларын, барлық *KEY* жазбасымен қоса қамтиды, бірақ өзін-өзі қамтымайды. Мұнда сонымен бірге, қолтаңбаның әрекет ету уақытының басы және соңы, қолтаңба иесінің аты және біршама қосымша ақпарат сақталады.

DNSsec жүйесі аймақ жабық кілті автономды режимде сақталатындай етіп құрастырылған. Күніне бір немесе екі рет деректер базасының мазмұнын автономды режимде жұмыс істейтін, жабық кілтті сақтайтын машинаға қолмен (мысалы, компакт-дискке жазып) ауыстырып отыруға болады. Ол жерде барлық жиындар үшін қолтаңба генерациялауға және осылайша алынған *SIG* жазбасын тағы да компакт-дискке жазып, бас серверге ауыстыруға болады. Осылайша, жабық кілтті компакт-дискте, сейфке жауып сақтап, әр тәулік сайын автономды машинадағы RRSet типтері жиынтығы жаңартуларына қол қою үшін шығаруға болады. Қолтаңба генерациялау аяқталғаннан кейін кілттің барлық көшірмелері жадыдан өшіріледі, ал компакт-диск сейфке оралады. Бұл процедура ақпаратты электронды қорғауды тұтынушыларға әлдеқайда түсінікті физикалық түрге айландырады.

Жиындарға алдын ала қол қою тәсілі сұраныстарды өңдеу үдерісін жылдамдатады, себебі «ұшуда» шифрлау қажеттілігі болмайды. Бұның құны – DNS деректер базасының барлық кілттері және қолтаңбасын сақтауға қажет үлкен көлемдегі диск кеңістігі. Осының салдарынан кейбір жазбалар көлемі ондаған есе ұлғаяды.

Қол қойылған RRSet алып, хэшті кері шифрлау үшін клиент аймақтың ашық кілтін қолданады, содан кейін хэшті өз бетінше есептейді және екі мәнді салыстырады. Егер олар сәйкес келсе деректер дұрыс деп саналады. Осылай бола тұрса да, бұл процедура клиенттің аймақ ашық кілтін алу сұрағын шешпейді. Мұнда қолданылатын бір әдіс кілтті сенімді серверден сұрау және оны қорғалған байланыс арқылы тасымалдау (мысалы, IPsec көмегімен).

Алайда, іс жүзінде клиентте барлық жоғары деңгей домендерінің ашық кілті бар деп болжанады. Егер Алиса Бобтың сайтына кіргісі келсе, ол DNS қызметінен *bob.com* үшін RRSet жиынтығын сұрайды. Бұл жиынтықта IP-адрес және Бобтың ашық кілті бар *KEY* жазбасы жазылады. RRSet-ке жоғары деңгей доменімен (*.com*) қол қойылады, сондықтан Алиса жиынның тұпнұсқа екенін тексере алады. RRSet жиынының мысалы *8.4-кестеде* келтірілген.

8.4-кесте. *bob.com* үшін RRSet жиынтығының мысалы. *KEY* жазбасында Бобтың ашық кілті жазылған. *SIG* жазбасы – бұл жоғарғы деңгей домен сервері (*com*) қол қойған *A* және *KEY* хэші, олардың сәйкестігін тексеру үшін.

Домен аты	Өмір уақыты	Класс	Тип	Мәні
bob.com	86400	IN	A	36.1.2.3
bob.com	86400	IN	KEY	3682793A7B73F731029CE2737D…
bob.com	86400	IN	SIG	86947503A8B848F5272E53930C…

Енді Бобтың расталған ашық кілт көшірмесін алып, Алиса Бобтың DNS-серверінен *www.bob.com* IP-адресін біле алады. Бұл RRSet-ке Бобтың жабық кілтімен қол қойылады, сондықтан Алиса Боб қайтарған жиын қолтаңбасының түпнұсқа екенін тексере алады. Егер зиянкес қандай да бір жолмен кэштердің біріне жалған RRSet енгізе алса, Алиса бұны сезеді, себебі *SIG* жазбасы дұрыс болмайды.

Дегенмен, DNSsec те жауаптарды сәйкес сұраныстармен байланыстыру үшін криптографиялық механизмді ұсынады. Бұл *8.43-суреттке* көрсетілгендей шабуылды болдырмауға көмектеседі. Бұл (міндетті емес) шара, механизмді сұраушының жабық кілтімен қол қойылған, сұраныс хэшінің жауабына қосумен жүзеге асырылады. Трудиге жоғары деңгей серверінің (*com* доменінің) жабық кілті белгісіз болғандықтан, ол Алиса провайдерінің осы серверге жөнелткен сұранысқа жауабын өзгерте алмайды. Ол, әрине, нағыз жауаптың алдын орап кете алады, бірақ жасандылық хэштелген сұранысқа дұрыс қолтаңба қосылмағандығымен бірден сезіледі.

DNSsec кейбір басқа жазбаларды да қолдайды. Мәселен, сертификаттарды сақтау үшін (мысалы, X.509 стандартының) *CERT* жазбасын пайдалануға болады. Мұндай жазба не үшін керек? Себебі, DNS-ті PKI инфроқұрылымына айналдырғысы келетінде бар. Бұл іс жүзінде орын ала ма, жоқ па, әлі белгісіз. Осымен біз DNSsec талқылауды аяқтаймыз. Толық ақпаратты RFC 2535 табасыздар.

8.9.3. SSL – қорғалған сокеттер хаттамасы

Әрине, ресурстар атын қорғау – жаман бастама емес, алайда бұнымен Дүниежүзілік өрмектегі қауіпсіздік толығымен қамтамасыз етілмейді. Келесі қадам – қауіпсіз байланыс орнату. Қазір біз оның қалай орындалатынын қарастырамыз. Қауіпсіздікпен байланыстының барлығы күрделі және бұл тақырыпта ерекше емес.

Веб-технологиялар алғаш рет көпшілікке ұсынылған кезде, олар стратегиялық парақтарды тарату үшін пайдаланылды. Алайда, кейбір компаниялар сол кезден бастап-ақ Дүниежүзілік өрмекті, несиелік карта бойынша тауар сатып алу, онлайн банк операциялары, құнды қағаздармен электронды сауда жүргізу тәрізді қаржылық транзакцияларды орындау үшін пайдалануды ойлады. Мұндай қосымшалар үшін қорғалған байланысты ұйымдастыру қажет болды. 1995 жылы сол кездегі браузер құрастырушылар ішіндегі көшбасшы, Netscape Communications осы мәселеге жауап ретінде **SSL (Secure Sockets Layer – қорғалған сокеттер хаттамасы)** атты қауіпсіздік жүйесін ұсынды. Сәйкес программалық жабдықтама хаттама тәрізді бүгінде кеңінен қолданылуда (соның ішінде Firefox, Safari, Internet Explorer), сондықтан SSL-ді егжей-тегжейлі қарастыруға болады.

Сонымен, SSL екі сокет арасында қорғалған байланыс ұйымдастырады. Бұл сокеттер:

1. клиент және серверге қолданылатын параметрлер жайлы келісуге;
2. клиенттің серверді сәйкестендіруіне;
3. құпия сөйлесуді ұйымдастыруына;
4. деректер тұтастығын қамтамасыз етуге мүмкіндік береді.

Аталған пункттердің барлығы бізге таныс, сондықтан біз оларға түсініктеме бермейміз.

SSL-дің әдеттегі хаттамалар стегінде орналасуы *8.44-суретте* көрсетілген. Іс жүзінде қолданбалы және транспорттық деңгейлер арасында браузерден сұранысты қабылдап, оларды TCP арқылы серверге жөнелтетін жаңа деңгей пайда болады. Қорғалған байланыс орнатылғаннан кейін SSL-дің негізгі мақсаты тығыздау және шифрлауды қолдау. Егер SSL үстінде HTTP қолданылатын болса, бұл нұсқа әдеттегі HTTP хаттамасы болғанына қарамастан, **HTTPS (Secure HTTP – қорғалған HTTP)** деп аталады. Кейде айырмашылық болуы да мүмкін, айталық, қолжеткізу стандартты порттың (80) орнына жаңа порт (443) арқылы жүзеге асырылады. Сонымен бірге, SSL пайдалану тек веб-браузерлермен шектелмейді, бірақ бұл кең таралған қолданылу. Ол өзара сәйкестендіруді де қамтамасыз ете алады.

SSL хаттамасының бірнеше нұсқасы бар. Төменде біз тек 3 нұсқаны қарастырамыз, себебі ол кеңінен таралған. SSL әртүрлі қосымша функциялары

бар, олардың ішінде тығыздаудың болуы немесе болмауы, шифрлаудың белгілі бір алгоритмі, сонымен бірге экспортты шектеу және криптографиямен байланысты мәселелер болуы мүмкін. Соңғы, криптография мәселесі, тек байланыстың екі басы да АҚШ шеңберінде екендігіне көз жеткізген кезде ғана күрделі криптография қолданылады. Басқа жағдайларда кілттің ұзындығы 40 битпен шектеледі, мұны криптографтар қалжың ретінде қабылдайды. Алайда, АҚШ үкіметінен экспортқа лицензия алу үшін Netscape шектеу енгізу керек болды.

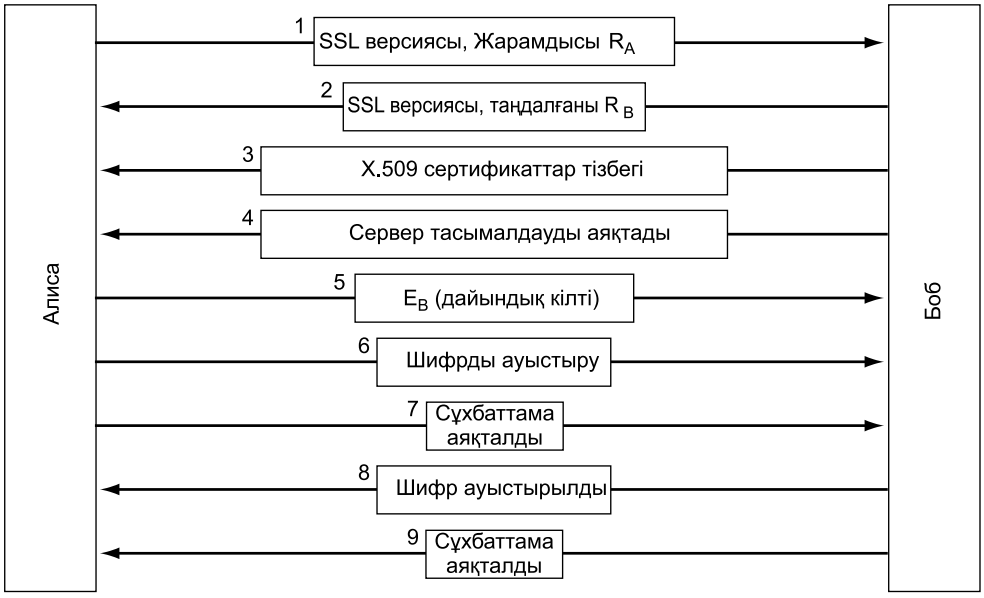
SSL екі сұхбаттамадан тұрады. Олардың бірі қорғалған байланысты орнатудан, ал екіншісі – осы байланысты пайдаланудан тұрады. Әңгімені байланыс орнатудан бастайық. Осы мәселемен айналысатын сұхбаттама жұмысы *8.44-суретте* көрсетілген. Барлығы, Алисаның Бобқа байланыс орнатуға сұранысы ретінде жөнелтетін 1-мәлімдемеден басталады. Бұл мәлімдемеде SSL версиясы, Алисаға ұнамды тығыздау және шифрлау алгоритмі, сонымен бірге кейіннен қолданылатын R_A нонсы көрсетіледі.

8.44-сурет. SSL бар әдеттегі үй браузері пайдаланатын деңгейлер (және хаттамалар)

Енді Бобтың кезегі келді. Ол 2 мәлімдемеде, Алиса қолдайтын бір алгоритмді таңдап, өзінің жеке R_B нонсы жөнелтеді. Үшінші мәлімдемеде өз ашық кілті сертификатын жөнелтеді. Егер сертификатта қандай да бір сыйлы ұйым қолы қойылмаған болса, ол да Алиса Бобтың сертификатына сенуге болатындығын тексере алатындай сертификаттар тізбегін жөнелтеді. Барлық браузерлер, соның ішінде Алиса пайдалданатын браузер де, о бастан шамамен жүздеген ашық кілттермен қамтылған, сондықтан егер олардың арасында Боб жөнелткен сертификаттың кілті кездессе, Алиса осы кілттің көмегімен Боб кілтін қайта қалыпқа келтіріп тексере алады. Осы сәтте Боб басқа мәлімдемелер де жөнелтуі мүмкін (мысалы, Алисаның ашық кілті сертификатын алуға сұраныс). Боб хаттаманың өзіне тиесілі бөлігін орындағаннан кейін, Алисаның кезегі келді деген 4-мәлімдемені жөнелтеді.

Алиса жауап ретінде 384-разрядты **дайындық кілтін (premaster key)**

таңдап, оны өз ашық кілтімен шифрлап, Бобқа жөнелтеді (5-мәлімдеме). Нақты сеанс кілті дайындық кілті және екі жақ нонстарының көмегімен есептеледі. Бұл өте күрделі процедура. Бесінші мәлімдемені алғаннан кейін Алиса да, Боб та сеанс кілтін есептей алады. Ол үшін Алиса Бобтың жаңа шифрға көшуін сұрайды (6-мәлімдеме), сонымен бірге байланысты орнату сұхбаттамасын аяқталды деп есептейді (7-мәлімдеме). Боб Алисамен келіседі (8 және 9-мәлімдеме).

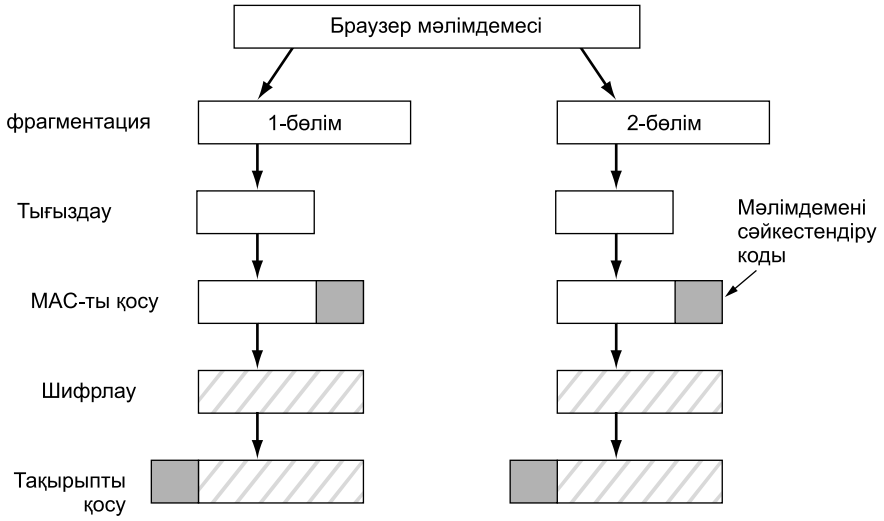


8.45-сурет. SSL байланыс орнату сұхбаттамасының қарапайым нұсқасы

Алайда, Алиса Бобтың кім екендігін білгенімен, Боб оның кім екенін білмейді (тек егер ашық кілт және оның сертификаты жоқ, бұл әдеттегі жеке тұлға үшін ережеден тыс). Сондықтан Алиса үшін бірінші мәлімдеме, ертеде алынған тұтынушы аты және құпия сөзді пайдаланып, тіркеуге тұру болуы мүмкін. Жүйедегі тіркеу хаттамасы SSL құзыретінен тыс. Кез келген жағдайда, осы сұраныс-растаулар сериясы аяқталғаннан кейін деректер тасымалдау басталады.

Жоғарыда айтылғандай, SSL әртүрлі криптографиялық алгоритмдерді қолдайды. Олардың ішіндегі ең мықтысы үш жеке кілті бар үштік DES-ті шифрлау үшін және SHA-1 деректер тұтастығын қамтамасыз еті үшін қолданылады. Алгоритмдердің мұндай комбинациясы өте баяу жұмыс істейді, сондықтан көбіне банк операцияларын орындау кезінде және жоғары дәрежедегі қауіпсіздікті қажет ететін қосымшаларда қолданылады. Электронды коммерцияның әдеттегі қосымшаларында шифрлау үшін 128-разрядты кілті бар RC4, ал сәйкестендіру үшін - MD5 қолданылады. RC4 бастапқы мән-

дері ретінде, алгоритммен жұмыс істеу кезінде бірнеше есе өсетін 128-разрядты кілт беріледі. Бұл ішкі сан кілттік ағынды құрастыру үшін падаланылады. Соңғысы ашық мәтінмен модуль-2 бойынша қосылады, нәтижесінде *8.12-суретте* көрсетілгендей әдеттегі ағындық шифр алынады. Алгоритмнің экспорттық нұсқалары да RC4 алгоритмімен, 128-разрядты кілтпен жұмыс істейді, бірақ бұл разрядтардың 88-і ашық етіп құрастырылады, бұл шифрды тез сындырылдатындай етеді.



8.46-сурет. SSL қолданған кездегі деректерді тасымалдау

Нақты деректерді тасымалдау үшін *8.46-суретте* көрсетілген екінші сұхбаттама пайдаланылады. Браузерден келетін мәлімдемелер көлемі 16 Кбайт деректер бірлігіне бөлінеді. Сонан кейін екі нонсы және дайындық кілті бойынша жабық кілт есептеледі. Жабық кілт тығыздалған мәтінмен біріктіріліп, келісілген алгоритм көмегімен хэштеледі (жиі MD5). Хэш әр фрагментке MAC (Message Authetication Code – мәлімдемені сәйкестендіру коды) түрінде қосылады. Осы тығыздалған фрагммент MAC бірге симметриялы кілтті келісілген алгоритм көмегімен шифрланады (әдетте бұл RC4 кілтті ағынммен модуль-2 бойынша қосу). Соңында, фрагмент тақырыбын қосып, барлығы TCP-байланыс арқылы жөнелтіледі.

Келесідей күтпеген жағдайдан абай болу керек: біз RC4-тің осал кілттері бар екендігін айтқан болатынбыз, сондықтан RC4 бар SSL – бұл өте осал негіз (Fluhrer және басқалар, 2001). Тұтынушыға шифрды таңдауды ұсынатын браузерлерді үнемі 168-разрядты кілті бар үштік DES және SHA-1 пайдаланатындай етіп баптаған дұрыс. Мұндай комбинация RC4+MD5 қарағанда баяу жұмыс істейді, бірақ қауіпсіздік маңыздырақ. Немесе SSL қабылдауышын қолдайтын браузерлерді пайдалану керек, біз оны қысқаша сипаттаймыз.

SSL байланысты тағы бір мәселе бар: принципалдардың сертификаттары болмауы мүмкін, тіпті ол болған жағдайдың өзінде де, кілт және сертификаттың сәйкестігін тексеру әрқашан жүргізілмейді.

1996 жылы Netscape Communications корпорациясы SSL-ді IETF стандарттауға жөнелтті. Нәтижесінде, **TLS (Transport Layer Security – транспорттық деңгейді қорғау)** стандарты болды. Ол RFC 5246 сипатталған.

TLS SSL үшінші нұсқасында құрастырылған. TLS стандартын құрастыру кезінде SSL-ге біраз өзгертулер енгізілген болатын, алайда осы өзгертулер салдарынан SSL 3 нұсқасы және TLS үйлесімсіз болды. Мысалы, кілтті күшейту үшін дайындық кілті және нонс бойынша сеанс кілтін есептеу әдісі өзгертілді. Осы үйлесімсіздік салдарынан көптеген браузерлер екі хаттаманы да пайдаланады және қажет болған жағдайда TLS кері SSL-ке түрлендіріледі. Бұл SSL/TLS деп аталады. TLS алғаш рет 1999 жылы қолданысқа енді, ал 1.2-версиясы 2008 жылдың тамыз айында пайда болды. Ол әлдеқайда мықты шифрлар жиынтығын қолдайды (соның ішінде AES). SSL әлі де мықты нарықтық позицияны ұстауда, алайда TLS біртіндеп орын алмастыруда.

8.9.4. Тасымалданатын программалар қауіпсіздігі

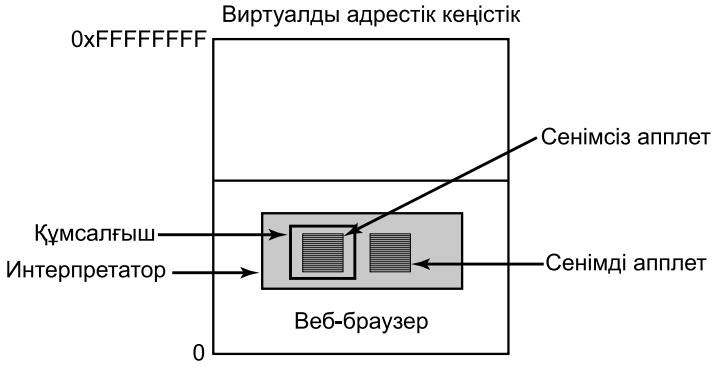
Ресурстарға ат беру және байланыс – бұл Дүниежүзілік өрмектегі ақпаратты қорғаумен тығыз байланысқан аумақтар екені күмәнсіз. Алайда, осы тақырыппен байланысты, басқа да маңызды мәселелер бар. Бастапқыда веб-парақтар толығымен статистикалық HTML-файл болып келетін және мұнда орындалатын код жоқ болатын. Қазір веб-парақтарда жиі кішігірім программалар кездеседі: Java-апплеттер, ActiveX басқару элементтері, JavaScript скрипттері. Осындай **тасымалданатын программаларды (mobile code)** жүктеу және орындау қауіпсіздік үшін өте үлкен тәуекелмен байланысты. Осындау тәуекел қаупін азайтудың әртүрлі тәсілдері құрастырылған болатын. Төменде біз тасымалданатын программалар салдарынан орын алатын кейбір сұрақтарды қарастырамыз.

Java-апплеттер қауіпсіздігі

Java-апплеттер – бұл **JVM (Java Virtual Machine – Java витуал машинасы)** деп аталатын стектік құрылымы бар машиналық тілге компиляцияланған Java тіліндегі кішігірім программалар. Осындай программалар веб-парақта орналастырылып, парақпен бірге жүктеледі. Парақ жүктелгеннен кейін апплеттер *8.47-суретте* көрсетілгендей, браузердегі JVM интерпретаторымен өңделеді.

Интерпретацияланатын кодтың компиляцияланатын кодтан артықшылығы – орындалар алдында әр нұсқау зерттеледі. Бұдан бетер, жүйелік шақырулар да ұсталып, интерпретацияланады. Мысалы, егер апплет сенімді (мысалы, жергілікті дискте құрастырылған) болса, оның жүйелік шақырулары қосымша

тексерусіз орындалуы мүмкін. Егер апплет сенімді деп саналмаса (мысалы, ол Интернеттен жүктелген), онда оның жүйелік ресурстарды пайдалануға талпыныс жасайтын әрекетін реттейтін **құмсалғышқа (sandbox)** орналастыруға болады.



8.47-сурет. Апплеттерді веб-браузер интерпретациялай алады

Егер жүйелік ресурсты жаулап алғысы келсе, онда бұл әрекетке не рұқсат беретін, не тыйым салатын қауіпсіздік мониторына шақыру беріледі. Монитор шақыруды ақпаратты қорғаудың жергілікті саясат тұрғысынан зерттейді және содан кейін қажет шешімді қабылдайды. Осылайша, апплеттерге кейбір ресурстрға (барлығына емес) қолжеткізуге мүмкіндік беріледі. Өкінішке орай, іс жүзінде мұндай модель нашар жұмыс істейді, онда үнемі қателіктер орын алады.

ActiveX

ActiveX басқару элементтері – бұл веб-парақтарға енгізуге болатын, x86 процессорына есептелген екілік программалар. Парақта осындай программа кездескен кезде оны орындау қажеттілігі тексеріледі және оң жауап кезінде ол іске қосылады. Бұл программалар интерпретацияланбайды және құмсалғышқа орналастырылмайды, сондықттан әдеттегі тұтынушы программасындай қасиеттерге ие және үлкен зиян келтіруі мүмкін. Осылайша, бұл жағдайда ақпаратты қорғау осындай басқарушы элементті іске қосуға бола ма деген сұраққа тіреледі. Осы құрылымның барлығы қауіпсіздік жүйесіндегі үлкен жырық деген шешім жасауға болады.

Осындай шешім қабылдау үшін Microsoft корпорациясы **код қолтаңбасына (code signing)** негізделген тәсілді таңдады. Бұның мағынасы, ActiveX әр элементі сандық қолтаңбамен қамтамасыз етіледі, нақты, оны құрастырушының ашық кілтімен қол қойылған хэш кодпен. Браузер басқарушы элементті кездестірген кезде, ол жол бойында кодтың ауыстырылмағанына көз жеткізу үшін алдымен қолтаңбаның дұрыстығын тексереді. Егер қолтаңба дұрыс бол-

са, браузер өзінің ішкі кестелері бойынша программаны құрастырушыға сенуге болатындығын тексереді. Нақты программа құрастырушы жайлы ешнәрсе белгісіз болуы мүмкін, бірақ өзінің сенімділігімен танымал қандай да бір құрастырушыға әкелетін растамалар тізбегі болады. Егер құрастырушы сенімді болса, программа орындалады, кері жағдайда еленбейді. ActiveX басқарушы элементтерін тексеру үшін Microsoft құрастырған жүйе **Authenticode** деп аталады.

Java және ActiveX әдістерін бір-біріне қарсы қойған пайдалы. Бірінші жағдайда апплет авторлығын орнатуға ешқандай талпыныс жасалмайды.

Оның орнына, апплетке қандай да бір жағымсыз әрекетті орындауға тыйым салатын интерпретатор пайдаланылады. Кодқа қол қою тәсіліне келетін болсақ, бұл жағдайда керісінше, программаның орындалу барысындағы әрекеті ешбір қадағаланбайды. Егер ол сенімді жерден алынған, бірақ жолда өзгертілген болса ол іске қосылмайды. Кодтың өзі тексерілмейді. Егер программист қасақана қатты дискті форматтайтын және компьютердің флэш-жадысын программа жазса, бірақ ол сенімді, тексерілген деп саналса, онда код орындалып, компьютер істен шығады (тек, егер ActiveX басқарушы элементтері өшірілген болса).

Көптеген адамдар, белгісіз программалық жабдықтамалар шығарушыларға сену жеңілтектік деп санайды. Осыны дәлелдеу үшін Сиэтлдегі бір программист өз компаниясын құрастырып, сенімділік сертификатына қолжеткізді. Бұл қиын шаруа емес. Осыдан кейін ол бар-жоғы компьютерді өшіретін ActiveX басқарушы элементін жазды. Ол бұл программаны кеңінен таратты және ол бір мың компьютерді өшірген. Әрине, компьютерді қайта іске қосуға болады, сондықтан мұндай басқарушы элементтің ешқандай зияны жоқ. Жобаның мақсаты – әлемге мәселенің бар екендігін көрсету. Ресми әрекет нақты осы басқарушы элемент сертификатына айтылды, сонымен іс жабылды. Бірақ мәселе шешілген жоқ және арам ниет программистер қорғанысстағы осы жырықты пайдалана берер еді (Garfinkel және Spafford, 2002). Тасымалданатын программалар жазушы барлық компаниялардың әрекеттерін бақылау мүмкін емес болғандықтан, кодқа қол қою тәсілі жақын арада үлкен қауіп төндіруі мүмкін.

JavaScript

JavaScript-те қандай да бір ресми ақпаратты қорғау моделі жоқ, оның орнына оны енгізудің сәтсіз талпыныстарының ұзын сонар тарихы бар. Әр өнім шығарушы өзінің бір затын құрастырғысы келеді. Мысалы, Netscape Navigator 2-версиясында Java-модельге ұқсас зат құрастырылды, ал төртінші версиясында код қолтаңбасы моделінің ізі байқалады.

Мәселенің мағынасы – бейтаныс программаға қандай да бір әрекет орындауға мүмкіндік беріледі. Бұл болжап болмайтын нәтижеге әкелуі мүмкін. Қауіпсіздік тұрғысынан, бұл ұрыны қонаққа шақырып, оның ас үйден қонақжайға кіріп кетпеуін мұқият бақылаумен тең. Егер күтпеген жағдай орын

алса, ал осы сәтте сіздің назарыңыз басқа затқа ауып кетсе, не болса да орын алуы мүмкін. Тасымалданатын програмаларды қорғау үшін айтылатын аргумент – олардың көмегімен флэш-график жеңіл жүзеге асырылады және тұтынушымен әрекеттесу жылдам орындалады. Веб-сайт құрастырушылар бұл ақпаратты қорғаудан әлдеқайда маңызды деп есептейі, әсіресе іс бөтен біреудің компьютеріне байланысты болса.

Браузерлер кеңейтілімі және қосымшалар

Веб-парақтарға код қосуға болатын тәрізді, браузерлерге де қосымша жасауға болады **(browser extension – браузер кеңейтілімі; add-on – браузерге қосымша, «аддон»; plug-in – іске қосылушы модуль, «плагин»)** және бұл қосымшалар нарығы өсуде. Бұл қосымшалар, веб-браузер мүмкіндігін кеңейтетін компьютерлік программалар. Іске қосылатын модульдер жиі қандай да бір нақты интерпретация функциясын немесе нақты контентті көрсетеді, мысалы, PDF немесе флэш-анимацияны көрсетуді ұсынады. Кеңейтілімдер және қосымшалар, құпия сөздерді жақсы басқару, парақтады басқарудың жаңа тәсілдері (мысалы, оларды белгілеу арқылы) немесе ыңғайлы интернет-сауда тәрізді браузердің жаңа функциясын қамтамасыз етеді.

Қосымшаны, кеңейтілімді немесе іске қосылушы модульді орнату өте қарапайым: желіден өзіңізге қажет программаны тауып, орнату үшін сілтеме арқылы өтіп отырсаңыз жеткілікті. Бұл программалардың барлығы, қандай браузерде қолданылатындығына байланысты, әртүрлі ортада жазылған. Алайда, бірінші жақындауда олар браузердің сенімді есептеу базасы бола алады. Яғни, егер қателігі бар код орнатылып жатса, браузердің өзі қателікпен жұмыс жасауды бастауы мүмкін.

Мұнда екі қақпан болуы мүмкін. Біріншісі – программа зиян келтіре бастауы мүмкін, мысалы жеке ақпараты жинап, оны қашықтықтағы серверге жөнелте бастау. Тұтынушының кеңейтуді осы мақсатта орнатқанын тек браузер білетін болады. Екінші мәселе – кеңейтілу модульдері браузерге жаңа типтегі ақпаратты оқуға мүмкіндік береді. Бұл ақпарақ дербес программалау тілдерінде жазылуы мүмкін. PDF және Flash бұған жақсы мысал бола алады. Тұтынушы PDF және Flash бар парақтарды қараған кезде, олардың браузерлеріндегі кеңейтулер модулі PDF және Flash кодтарды орындайды. Бұл кодтардың қауіпсіз болғаны жақсы болар еді, бірақ кейбір сезімтал кодтар бар. Осы себептердің барлығымен қосымшалар және кеңейтулер модулін сенімді сатушылардан алып орнатқан жақсы.

Вирустар

Вирустар – тасымалданатын кодтың өзінше бір формасы. Тек жоғарыда келтірілген мысалдардан ерекшелігі – бұндай программаларды ешкім іске қосқысы келмейді. Вирустардың әдеттегі тасымалданатын програмалардан

айырмашылығы – олар өз-өздерін қалыпқа келтіре алады. Жүйеге вирус енген кезде (веб-парақтан, электронды поштада қыстырылған файлдардан немесе тағы басқа жолдармен), алдымен ол дискте сақталған, орындалушы программаларға жұғады. Осы программалардың бірі іске қосылған кезде басқару вирусқа беріледі, ал ол әдетте өз әрекетін басқа да машиналарға таратпақ болады, мысалы, өз-өзін электронды пошта адрестік кітабындағы барлық адресаттарға жөнелту арқылы. Кейбір вирустар қатты дисктің жүктеу секторына жұғады, содан кейін машина іске қосылған кезде активтеледі.

Қандай да бір сәтте вирустар Интернет үшін ірі масштабтағы мәселе бола бастады және миллиардтаған шығындар әкелді. Мәселенің қандай да бір қарапайым шешімі жоқ. Мүмкін қорғалған микроядроға негізделген операциялық жүйелердің жаңа буыны құрастыру және тұтынушылар, үдерістер, ресурстар ынтымақтастығы жағдайды түзете алар.

8.10. ӘЛЕУМЕТТІК АСПЕКТ

Интернет және ақпаратты қорғау технологиясы – бұл әлеуметтік сұрақтар, мемлекеттік саясат және технологиялар тығыз байланысқан аумақтар. Төменде біз үш мәселені қысқаша қарастырамыз: құпиялық, сөз еркіндігі және авторлық құқық. Осы кітап көлемінде біз бұл сұрақтарға тек үстіртін сипаттама бере аламыз. Толық ақпаратты Anderson (2008a), Garfinkel және Spafford (2002), Schneier (2004) басылымдарынан алуға болады. Көптеген материалдарды Интернеттен оқуға болады. Іздеу машинасында «құпиялық» (privacy – ақпаратты ағылшын тілінде алу үшін), «цензура» (censorship) немесе «авторлық құқық» (copyright) сөздерін терсеңіз жеткілікті. Сонымен бірге, осы кітап сайтын қарауларыңызға болады: *www.pearsonhighered.com/tannenbaum.*

8.10.1. Құпиялық

Адамдардың құпиясы болуға құқығы бар ма? Жақсы сұрақ. АҚШ конституциясына енгізілген төртінші түзетуде үкіметтік ұйымдарға аса қажет болмаған жағдайда азаматтардың ниеті, тұрғын үйі және қағаздарына қызығушылық тудырмау керек делінген. Бұл шектеуді бұзуға болатын жағдайлар тізімі келтірілген. Осылайша, құпиялық сұрағы күн тәртібінде 200 жылдан бері тұр, жеке алғанда АҚШ-та.

Соңғы он жылда не өзгерді? Үкімет бұрын-соңды болмаған, жеңіл жолмен азаматтар артынан тыңшылыққа түсуге мүмкіндік алды және азаматтар да, осы тыңшылықтың жолын кесуге жеңіл мүмкіндіктер алып отыр. XVIII ғасырда азаматтың жеке қағаздарына қол жеткізу үшін оның мекен жайына, қолайсыз ауа райына, жол ауыртпалығына қарамастан, мініс атпен полиция қызметкерін жөнелту керек болатын – бұның бәрі бөтен біреудің бір парақ қағазын

оқу үшін. Қазіргі күнде телефон компаниялары және интернет-провайдерлер сәйкес ордер алып келгендердің барлығын жасырын тыңдаушы құрылғылармен қамтамасыз етеді. Бұл құрылғылардың көмегімен полиция қызметкерінің жұмысы әлдеқайда жеңілдейді және оған ер-тоқымнан түсіп қалу немесе жолда ұйықтап қалу қаупі төнбейді.

Осылай бола тұрса да, криптографияны пайдалану жағдайды біршама өзгертеді. Кез келген адам, жақсы қорғалған, сенімді кілті бар PGP жүктеп, орнатуға қам жасауы тиіс, нәтижесінде ол ордері бар жоғына қарамастан, Дүниежүзінде ешкім оның электронды поштасын оқи алмайтынына сенімді бола алады. Үкімет бұны жақсы түсінеді және бұл оларға ұнамайды. Іс жүзінде – құпиялық, өкілетті ұйымдарға әртүрлі қылмыскерлерді қадағалау қиындатады дегенді білдіреді, әсіресе журналистер және саяси оппонеттерді. Көптеген үкіметтердің криптографияны пайдалану және экспорттауға тыйым салатыны ғажап емес. Мәселен, Францияда 1999 жылға дейін кез келген үкіметтік емес криптографияға тыйым салынған болатын, тек үкіметке қолданыстағы барлық кілттерді берген жағдайда ғана рұқсат етілетін.

Бұл істе Франция жалғыз емес. 1993 жылдғы сәуір айында АҚШ өзінің **аппараттық криптопроцессор (clipper chip)** құрастырып, оны кез келген желілік коммуникацияларда пайдалану стандарты еткісі келетіні жайлы хабарлады. Осылайша, азаматтар кепілденген құпиялыққа ие болады. Сонымен бірге, үкіметтің осындай криптопроцессорлардың бүкіл трафигін, үкіметке барлық кілттерге қолжеткізуге мүмкіндік беретін, арнайы технология көмегімен (**key escrow**) кері шифрлауға мүмкіндігі болатыны жайлы хабарланды. Алайда, бұл мүмкіндікті сәйкес санкция болған жағдайда ғана пайлануға уәде берген. Мұндай хабарландырудың үлкен әсер туғызғаны түсінікті: құпиялықты қолдаушылар жоспарды бастан-аяқ айыптады, ал осы бастаманы қолдаушы атқамінерлер үкімет ұсынысын мадақтаумен болды. Осылай бола тұрса да, үкімет кейіннен өз позициясынан беріп, идеясынан бас тартты.

Сандық ақпараттың құпиялығына қатысты көптеген материалдар Electronic Frontier Foundation (*www.eff.org*) веб-сайтында қолжетімді.

Анонимді таратулар

PGP, SSL және басқа да технологиялар екі жақ арасында қорғалған, сәйкестендірілген, үшінші жақтың кірісуіне шалдықпаған байланыс орнатуға мүмкіндік береді. Алайда, кейде құпиялық сәйкестендірудің жоқтығымен жақсы қамтамасыз етіледі, яғни іс жүзінде анонимді байланыстың орнатылуымен. Анонимділік екі тұтынушы арасында мәлімдеме тасымалдағанда да, сонымен бірге желілік телеконференцияларда да талап етілген.

Кейбір мысалдарды қарастырайық. Біріншіден, авторитарлық режимде өмір сүретін саяси диссиденттер репрессиядан құтылу үшін анонимді хабарласуы мүмкін. Екіншіден, көптеген коммерциялық, білім, үкімет және басқа да

ұйымдардағы әртүрлі ережені бұзушылықтар жиі сөз тасушылардың көмегімен анықталатыны рас. Үшіншіден, дәстүрлі емес (яғни, көбіне кінәлау) әлеуметтік, саяси немесе діни көзқарасты ұстанушылар телеконференцияларды (немесе электронды поштаны), нақты өз аттарын жасыра алатын қатынасу мүмкіндігі деп біледі. Төртіншіден, көптеген адамдар алкоголизімді, жан ауыруын, сексуалды мәселелерді, балаларды жәбірлеу мәселесін немесе қудаланушы азшылықтарды, анонимді болып қалатын телеконференцияларда талқылауды жақсы көреді. Бұдан басқа да көптеген мысалдар бар.

Нақты бір мысалды қарастырайық. 1990 жылдары бір дәстүрлі емес діни секта сыншылары өз пікірлерін USENET конференциясында **анонимді тарату (anonymous remailer)** арқылы жариялаған. Сервер тұтынушыларға жасырын ат жасап, одан электронды хаттар жөнелтуге мүмкіндік берген. Содан кейін бұл хаттар таңдап алынған жасырын аттан таратылады. Соңында хаттың нақты авторы кім екенін анықтау мүмкін болмады. Бұл мақалалардың көбі, секта мүшелерінің пікірінше, коммерциялық құпиялар және авторлық құқықпен қорғалған құжаттардан тұратын әшкерелеу болды. Осы әшкерелеуге жауап ретінде секта, коммерциялық құпияны ашқандығын және авторлық құқық заңының бұзылғандығын тиек етіп, сотқа жүгінді. Бұл екі мәселеде, сервер орналасқан аумақта қылмыс болып саналатын. Сот, сервер иелерін жалған атты бүркеніп, әшкерелеулерді таратқан нақты аттарын атауды талап етті (бұл, шіркеудің құпияның ашылғандығына наразылық білдіріп, сотқа жүгінуінің орын алып отырғаны бірінші рет емес еді: Уильям Тиндэйл (William Tyndale) 1536 жылы Библияны ағылшын тіліне аударғаны үшін тірідей өртелген болатын.).

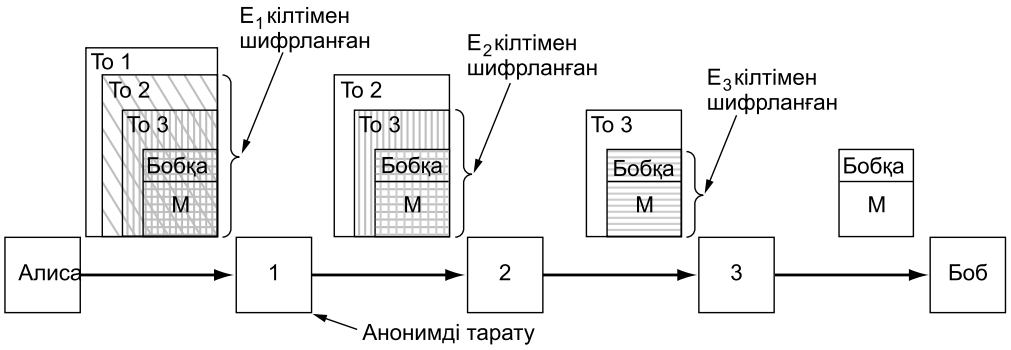
Интернет-қауымдастықтың үлкен бөлігі құпиялық қағидасының бұзылғандығына наразылық білдірді. Нағыз электронды адрестер мен жалған аттар сәйкестігі кестесін сақтаған (бұл бірінші типтегі анонимдік тарату деп аталды) аноним тарату иесінің дұрыс істемегенімен келісті. Бұл жағдай, сот тарапынан осындай шабуылға төтеп бере алатын, аноним таратуды дамуға ынталандырды.

Бірінші типтегі таратулар, жиі **шифрланған панктік таратулар (cypherpunk remailer)** деп аталады. Олар келесідей жұмыс істейді: тұтынушы әдеттегі RFC 822 тақырыбымен (әрине, *From* жоқ) электронды хат құрастырады, оны тарату ашық кілтімен шифрлап, серверге жөнелтеді. Ол жерде оның RFC 822 тақырыбын алып тасталынып, мазмұны кері шифрланады және мәлімдеме жазылушыларға таратылады. Таратуда ешқандай есепке алу жазбасы жоқ, ешқандай журналға тіркелмейді, сондықтан, тіпті серверді тәркілеу жасап, алып алғанның өзінде ол арқылы өткен хаттың ізін таба алмайсыз.

Өз жеке басының анонимдігі мәселесі үшін қатты күйзелетін тұтынушылар, өз мәлімдемелерін *8.48-суретте* көрсетілгендей, анонимді мәлімдемелер тізбегі арқылы өткізеді. Бұл мысалда, Алиса Бобқа Әулие Валентин күнімен құттықтап, шын мәнінде анонимді мәлімдеме жөнелткісі келеді. Ол үшін Алиса үш анонимді таратуды пайдаланады. Ол М хатын құрастырып, Боб электронды поштасы көрсетілген тақырыпты қояды. Содан кейін бұл мәлімдеме, 3-та-

рату ашық кілтімен шифрланады (көлденең штрихтармен көрсетілген). Осы мәлімдемеге 3-тарату электронды адресінің тақырыбы қосылады (ашық мәтінмен беріледі). Нәтижесінде суреттегі 2 және 3-таратулар арасында көрсетілген мәлімдеме алынады.

Мұнымен мәлімдеменің тарихы аяқталмайды. Ол , 2-тарату кілтімен шифрланады (тік штрихтармен көрсетілген) және 2-тарату электронды адресі көрсетілген тақырыппен біріктіріледі. Нәтижесінде алынған мәлімдеме, суретте 1 және 2-таратулар арасында көрсетілген. Сонан кейін, Алиса өз мәлімдемесін , 1-тарату ашық кілтімен шифрлайды және 1-тарату тақырыбы адресі қосылып, жөнелтіледі. Соңғы мәлімдеме суретте Алисаның оң жағында көрсетілген.



8.48-сурет. Алисаның хатын Бобқа жөнелту үшін үш анонимдік таратуды пайдалану

Алиса хаты 1-таратуға жеткен кезде, одан сыртқы тақырыбы алынып тасталынады. Мәлімдеме денесі кері шифрланып, 2-тарату арқылы жөнелтіледі. Сәйкес әрекеттер келесі екі серверде орындалады.

Алиса соңғы мәлімдемесінің жолын қалыпқа келтіру күрделігіне қарамастан, көптеген таратулар қосымша қауіпсіздік шараларын пайдаланады. Мысалы, олар мәлімдемені кездейсоқ уақыт аралығында кідіртіп отырады, мәлімдеменің соңындағы қоқысты өшіреді немесе керісінше қосады, мәлімдемелердің орнын ауыстырады, жалпы трафикті қадағалап, анонимдік тарату арқылы өткен мәлімдеменің авторы кім екенін түсінгісі келгендерді шатыстырады. Осы айтылған идеяның жүзеге асырылу жүйесінің сипаттамасын Mazieres және Kaashoek (1998) басылымынан қарауға болады.

Анонимдік қағидаларын пайдалану шеңбері бір электронды поштамен шектелмейді. Дәл осы, бір түйін тізбектегі келесі түйінді ғана білетін, қатпарланған жолдар формасын пайдаланып, интернет-материалдарды анонимді қарауға мүмкідік беретін қызмет түрлері де бар. Мұндай тәсіл **жуалық маршруттау (onion routing)** деп аталады, осылайша дестені әрі қарай жөнелту үшін, әр түйін жуаның бір қабатын аршиды. Tor – осындай жүйенің жақсы мысалы бола алады (Dingledine және басқалар, 2004). Тұтынушы өз браузерін осындай қызметті пайдалану үшін прокси ретінде баптай алады. Осыдан кейін,

тұтынушы орнына парақты сұрайтын HTTP-сұраныстар осы қызметке тиесілі адрестер бойынша жөнелтіледі. Егер анонимдікті қамтамасыз ететін серверде активтілік журналы сақталмаса, шын мәнінде парақты кімнің сұрағандығын анықтау мүмкін емес.

8.10.2. Сөз еркіндігі

Құпиялық, бөтен адамдардан жариялауға болмайтын ақпаратты жасыру мәселесімен байланысты. Екінші, кілтті әлеуметтік аспект – бұл сөз еркіндігі және оның қарсы бері – цензура. Бұл жағдайда басқарушы ұйымдар, азаматтар оқып, жариялай алатын ақпарат спекторын шектеуге тырысады. Миллиондаған парақтары бар Дүниежүзілік өрмек цензура үшін – пейіш. Режимнің түрі және идеологиясына байланысты көруге тыйым салынған материалдар тізіміне төменде аталғандар кіруі мүмкін:

1. Балалар және жасөспірімдерге көрсетуге болмайтын материалдар.
2. Қандай да бір этникалық, діни, сексуалды және басқа да топтарға өшпенділікті насихаттайтын материалдар.
3. Демократия және демократия құндылықтары жайлы материалдар.
4. Ресми нұсқаға сәйкес келмейтін тарихи оқиғалар сипаттамасы.
5. Мәлімдемені шифрлау, қару жасау, әртүрлі құлыптарды сындыруға басшылық және т.б.

Нашар, жарамсыз сайтты қарауға тыйым салу ең жеңілі.

Кейде мұндай саясат нәтижесі күткендей болмайды. Мысалы, кейбір көпшілікке ашық кітапханалар өздеріне порнографиялық сайттарды кіргізуге рұқсат бермейтін және осылайша Өрмек мазмұнын жасөспірімдер қарауға қауіпсіз ететін веб-сүзбе орнатқан. Сүзбе парақты шығарар алдында өзінің «қара тізімімен» салыстырады және оларда әдепсіз сөздердің болуын тексереді. Бірде, Вирджиния штатының, Лаудаун округінде сүзбе сүт безі рагы жайлы ақпарат іздеуді оқшаулаған, себебі сұраныста «breast» (әйел көкірегі, сүт безі) сөзі болған. Клиент округ үкіметіне қарсы іс ашқан. Дәл осы кезде басқа да жағдай болған: Калифорния штатының Ливермор қаласында бір ата-ана көпшілікке ашық кітапхананы сүзбе қоймағандары үшін сотқа берген. Клиент өзінің 12 жасар баласын осы кітапханада порнографиялық сайтты қарап отырған жерінде ұстап алған. Кітапханалар не істеуі керек?

Көпшілігі Дүниежүзілік өрмектің – шынында да дүниежүзілік екенін түсіне бермейді. Ол бүкіл жер шарын қамтыған. Әр елде Желіде не болуы керек, не болмауы керек деген сұраққа өз көзқарасы бар. Мысалы, 2000 жылдың қараша айында француз соты Калифорниядағы Yahoo! корпорациясы ұлтшылдар естелік заттары аукционына француз тұтынушыларына қолжеткізуге тыйым

салуы керек деді, себебі мұндай ақпаратты білу француз заңына қайшы келеді. Yahoo! істі корпорация пайдасына шешкен АҚШ сотына шағым түсірді, алайда, жалпы Интернетте қай елдің заңын пайдалану керектігі өзекті мәселе болып қалмақ.

Өздеріңізге Юта штатының қандай да бір соты Франция шараптарға арналған сайтты оқшаулауы керек, себебі мұндай арпарақ штаттың алкоголь тауарына қатысты қатаң заңын бұзады, деген шешім қабылдаған жағдайды елестетіп көріңіздерші? Дәл осылайша Қытай демократия жайлы әңгімелейтін сайттарға шек қоюы мүмкін, себебі бұл Аспан асты елінің қызығушылығында емес. Иранның дін жайлы заңдары Швецияда қолданылуы тиіс пе? Сауд Аравиясы әйелдер құқығын қорғайтын сайттарды оқшаулауы мүмкін бе? Мәселенің Пандора жәшігі тәрізді және көптеген сұрақтарды туындататыны түсінікті.

Осы мәселе жайлы Джон Гилмор (John Gilmore) бағалы пікір айтқан: «Желі цензураны жолдың қираған бөлігі ретінде қабылдап, айналып өтуде». Бұл ойдың нақты жүзеге асырылуы **мәңгілік қызметі (eternity service)** деп аталады (Anderson, 1996). Оның мақсаты – бір кезде жарияланған материалдар жоғалып кетпейді және Сталин кезінде кеңес өкіметінде болғандай қайта жазылмайтындығына кепілдік беру. Мәңгілік қызмет тұтынушысы тек ақпараттың қандай уақыт аралығында сақталуы тиіс екендігін көрсетуі тиіс, ақпарат көлеміне және мерзіміне сәйкес ақыны төлеп, деректерді серверге жүктеуі тиіс. Бұдан кейін ешкім, тіпті тұтынушының өзін қоса алғанда, мәңгілік қызмет серверінде орналасқан материалдарды өзгертіп, өшіре алмайды.

Мұндай қызметті іс жүзінде қалай жүзеге асыруға болады? Ең жеңілі, құжаттар әрқайсысы өз сыйақы үлесін алатын, жоба мүшелерінің ондаған серверлерде орналасатын, тең құқықты (пирингілік) жүйені ұйымдастыру, бұл олардың жүйеге енуін ынталандырады. Серверлер әртүрлі орындарда және заң құзіретінде орналасуы тиіс, бұл жүйенің тұрақтылығын қамтамасыз етеді. Бір сервер сәтсіздікке ұшырағанда басқасы аман қалатындай, кездейсоқ таңдап алынған 10 серверлер тізімін құпия және әртүрлі орны да сақтау керек. Ұнамсыз ақпаратты жойып жіберуге тым әуестенген кез келген үкіметтік ұйымдар, барлық көшірмелерді тапқандықтарына ешуақытта сенімді бола алмайды. Бұдан басқа, құжаттың қандайда бір нұсқасы өшірілгендігі жайлы мәлімдеме келген кезде, жүйе өзін-өзі қалыпқа келтіре алатындай етіп құрастыруға болады. Басқа көшірмелерді сақтағандар істен шыққандардың орнына басқа сақтау орындарын табады.

Мәңгілік қызмет Желідегі цензураға қарсы тұрудың бірінші талпынысы болатын. Одан бері осы тақырыпта әртүрлі көптеген ойлар айтылды және олардың кейбіреулері жүзеге де асырылды. Шифрлау, анонимдік, істен шығуға қарсы тұру тәрізді жаңа мүмкіндіктер қосылды. Көбіне құжаттар бірнеше фрагменттерге бөлініп, бірнеше серверде сақталады. Осындай жүйелердің ішінде Freenet (Clarke және басқалар, 2002), PASIS (Wylie және басқалар, 2000) және Publius (Waldman және басқалар, 2000) жүйелерін айта кетуге болады. Тағы бір құрастырылым Serjantov (2002) басылымында сипатталған.

Веб-сайт, програмалық жабдықтама, ғылыми құжаттар, электронды пошта, көмек көрсету телефон қызметі және осы тәрізді басқа да сезілмейтін заттар экспортын қадағалағысы келетін елдер көбейіп келеді. Тіпті, өзінің ғасырлар бойы сөз еркіндігін қолдау дәстүрімен дәріптелетін Ұлыбританияда, мысалы, Британия профессоры және Кембридж университеті студенті арасындағы мемлекеттік лицензиялауға жататын экспорт жайлы техникалық пікірталасты анықтайтын қатаң заңдар пайда бола бастады (Anderson, 2002). Мұндай саясат қарама қайшылыққа толы деп санайтындар көп болуы керек.

Стеганография

Цензура кеңінен қолданылатын елдерде, әр уақытта цензураны айналып өтудің өз әдісі бар диссидентер болған. Әрине, криптография (әркез заңды түрде емес) мағынасын ешкім түсіне алмайтындай мәлімдемелер жөнелтуге мүмкіндік береді. Алайда, егер үкімет Алисаны өз жауы деп санаса, онда оның Бобпен қарым-қатынаста болуы, Бобты да үкімет жауы жағдайына әкелуі мүмкін. Осылайша, математиканы жақсы білетін саясатшылар транзитивтік қағидасын түсінеді және пайдаланады. Анонимді мәлімдемелер көмекке келе алады, бірақ үкімет оған тыйым салуы мүмкін, осындай кезде шекарадан әрі мәлімдеме жөнелту үшін экспорттық лицензия қажет болады. Осылайша, анонимді таратулар – бұл да мәселенің шешімі бола алмайды. Алайда, Дүние-жүзілік өрмек әдеттегідей тығырықтан шығудың жолын табады.

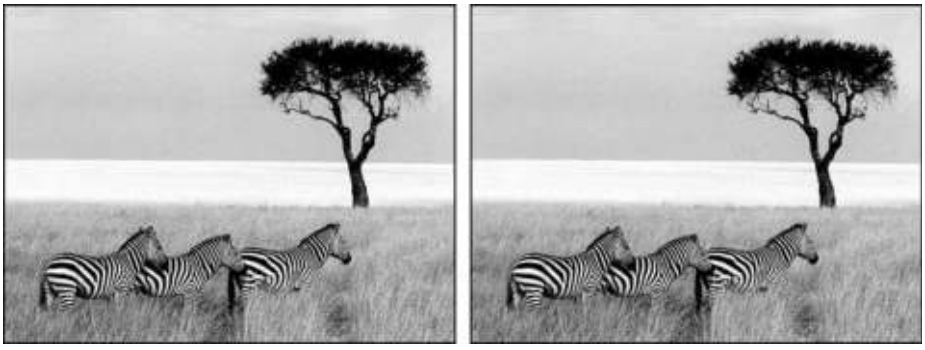
Құпия қарым-қатынас қажет адамдар, жиі қарым-қатынастың өзін жасырғылары келеді. Мәлімдемені жасырумен айналысатын ғылым **стеганография** (**steganography**, стенографиямен шатастырмаңыз!) деп аталады. Бұл гректің «қорғалған хат» деп аударуға болатын сөзінен шыққан. Ертедегі гректердің өздері бірінші болып осы әдісті пайдаланған. Геродот әскери басшылар құпия әрекеттесуінің өзіндік тәсілін сипаттаған: шабарманның шашын тықырлап алып, желкесіне құпия мәлімдеме бар татуировака салып, шашының өсуін күткен. Шашы өскеннен кейін жолға шығарған. Тек өткізгіштік қабілеттілік жоғары, ал кідіріс төмен болғандығы болмаса, қазіргі заманғы технологияларда осы тұжырымдамаға негізделген.

Мысал ретінде *8.49 а-суретін* қарастырайық. Кения авторларының бірі жасаған бұл фото суретте үш ала құлан және мамыргүл сипатталған. Алайда бұл сурет тек эстетикалық жағынан ғана ұнамды емес. Мәселе, *8.49 ә-суретіне* Шекспирдің бес ең танымал пьессаларының толық мәтіні енгізілген: *Гамлет, Лир патшасы, Макбет, Венециан көпесі* және *Юлий Цезарь* – барлық мәтін 700 Кбайттан артық орын алады.

Бұл қалай жасалған? Стеганографикалық арна келесідей жұмыс істейді. Бастапқы суреттің көлемі 1024×768 пиксель. Әр пиксель 8-биттік санмен берілген, олардың әрқайсысы өз түс құраушысына жауап береді (түс қоюлығының қызыл, жасыл және көк арнасы бар). Нәтижелік түс үш түс суперпозициясының сызықтық қоюлығынан алынады. Стеганографиялық

шифрлау тәсілінде көрсетілген үш санның әрбір кіші биті құпия арнаның қажет мәнімен алмастырылады. Осылайша, әр пиксельде құпия ақпараттың үш биті орналасады: бірі қызыл түс қоюлық арнасында, бірі – жасыл және бірі – көк. Біздің суретімізге 1024×768×3 бит немесе 294 912 байт құпия ақпарат орналасады.

Шекспирдің бес пьесасының толық мәтіні кішігірім алғы сөзімен қоса 734 891 байтты алады. Мәтінді кез келген тығыздағыш программа көмегімен 274 Кбайтқа дейін тығыздауға болады. Содан кейін осы тығыздалған мәтінді IDEA алгоритімінің көмегімен шифрланып, түстер қоюлығының кіші байттарына орналастырылады. Өзіңіз көргендей (дұрысы, көрінбейтіндей болып қалады) мәтіндік ақпараттың бар екендігі мүлдем байқалмайды. Үлкен, толық түрлі-түсті фото суретте де бұл байқалмайды. Адам көзі 21-разрядты санмен, 24-разрядты түспен өрнектелген түстерді ажырата алмайды.



а *ә*

8.49-сурет. Үш ала құлан және мамыргүл (а); Үш ала құлан, мамыргүл және Вильям Шекспирдің бес пьесасы (ә)

8.49-суретте келтірілген ақ-қара түсті фотосуреттер стеганография мүмкіндігі жайлы толық ақпарат бере алмайды. Суреттің тиімдірек нұсқасы, *8.49 ә-суреті*, осы кітап веб-сайтынан табуға болатын, толық түрлі-түсті версияда берілген.

Білдірмей ақпарат алмасу қажеттілігі үшін диссиденттер, саяси тұрғыдан жарамды фотосуреттерге толы (мысалы, Ұлы жетекші, жергілікті спорт командасы, теле және кино жұлдыздары және т.б.) веб-сайт құрастыра алады. Мұндай суреттер стеганографиялық мәлімдемелерден тұруы мүмкін. Мәлімдемелерді графикалық файлдарға орналастыру алдында шифрланып, тығыздалатынын ескерсек, тіпті бұл суреттер «ішінде» мәлімдеме бар екенін білгеннің өзінде, оларды ақ шудан ажырату, одан бетер оның мағынасын кері шифрлау, тіпті қиын. Әрине, стеганографиялық өңдеуге арналған фотосуреттер сканерленген немесе сандық камераға жеке түсірілген болуы керек. Егер суретті Интернеттен алып, оған құпия мәлімдеме жазсақ, оның «екінші түбін» қарапайым биттік салыстыру арқылы анықтау әлдеқайда жеңіл болады.

Стеганографиялық мәлімдемені графикалық файлдардан басқа файлдарға да жазуға болатындығы түсінікті. Бұл мақсатта дыбыстық файлдар өте қолайлы. Десте кідірістерін, дыбыстың бұрмалануын немесе тіпті десте тақырыптар өрісін басқару арқасында жасырын ақпаратты VoIP-қоңырауға орналастыруға болады (Lubacz және басқалар, 2010). Тіпті, HTML-файлдағы мәтіннің форматталуы және тәгтердің орналасуы белгілі бір жасырын ақпаратты тасымалдауы мүмкін екендігін айта кету керек.

Біз стеганографияны сөз еркіндігі мәселесі тұрғысынан қарастырдық, алайда бұл технологияның басқа да көптеген қолданыстары бар. Жиі электронды бейнелер авторлары файлдарға, қажет болған жағдайда олардың авторлық құқығын дәлелдеп беретін құпия мәлімдемелер енгізеді. Егер кімдекім сәтті суретті ұрлап алып авторын көрсетпей өз сайтында орналастырса, заңды иесі сотта өз авторлығын дәлелдей алады. Мұндай қолтаңбаларды кейде **су белгілері** деп атайды. Бұл тақырып Piva және басқалар (2002) басылымда талқыланған.

Стеганография мәселесінің толық сипаттамасын Wayner (2008) басылымынан табуға болады.

8.10.3. Авторлық құқықты қорғау

Құпиялық және цензура – бұл технологиялық аспектілер және қоғам мүддесі қақтығысатын аумақтар. Осындай үшінші аумақ – авторлық құқықты қорғау. **Авторлық құқық (copyright)** оны құрастырушылар, мысалы, жазушылар, ақындар, суретшілер, композиторлар, сазгерлер, фотосуретшілер, кинорежиссерлер, хореографтар және т.б. **интелектуалдық жекеменшікті (Intellectual Property – IP)** иелену еркіндігіне кепілдік береді. Авторлық құқық белгілі бір мерзімге беріледі, әдетте, ол автор өмірі және оған 50 жыл қосылады (75 жыл – корпоративтің авторлық құқық жағдайында). Осы мерзім аяқталғаннан кейін интеллектуалдық жекеменшік қоғам игілігіне беріледі және әркім өз қалауынша иеленуге мүмкіндігі бар. Мәселен, Gutenberg (*www.promo.net/pg*) ертеде қоғам игілігіне айналған мыңдаған шығармаларды Желіде орналастырды (Шекспир, Диккенс, Марк Твен жұмыстары). Голливуд өтініші бойынша 1998 жылы АҚШ Конгресі авторлық құқық мерзімін тағы 20 жылға созбақ болды, себебі бұл шараны қолданбасақ бұдан былай ешкім ешнәрсе құрастырмайды деп дәлел айтты. Ал осы тұста, өнертабысқа берілетін патенттер авторлық құқығының мерзімі бар-жоғы 20 жыл, ешкім шағымданбайды – адамдар жаңалықтар ашып өнертабыстар жасауда.

Әуендермен алмасу Napster қызметінің тұтынушылар саны кенеттен 50 млн жеткен кезде, авторлық құқықты қорғау сұрағы бірінші қатарға шықты. Napster жүйесінің жазбасы ешқайда көшірмегеніне қарамастан, сот тұтынушылардың қайсысында қандай жазба бар ақпарат сақталған орталық деректер базасын сақтады деген айып тақты. Осылайша, жүйе тұтынушыларды авторлық

құқық заңын бұзуға итермеледі. Әрине, ешкім авторлық құқық идеясы нашар деп айтпайды (бұл жерде азаматтарға қарағанда компаниялар жеңілдіктері көп), алайда әуенде жазбасын таратудың келесі технологиялары этикалық сипаттағы сұрақтарды көтеруде.

Мысалы, заңды файл алмасумен айналысатын (қоғам иелігі болып саналады, үй бейне жазбалары, діни трактаттар (шіркеудің коммерциялық құпиясына жатпайтын) және т.б.) тең рангілі желіні қарастырайық. Мүмкін, осы файлдардың кішкене бір бөлігі заңмен қорғалған шығар. Айталық, барлық тұтынушылар тұрақты байланыста отырарды (ADSL немесе кабельді Интернет). Әр машинада қатты дискте жазылған файлдар, сонымен бірге желідегі барлық машиналар тізімі бар. Қандай да бір файлды іздеу кезінде еркін таңдап алынған машинаға жүгініп, ондағы материалдар тізімін сұрауға болады. Егер қажет ақпарат жоқ болса, онда тізімдегі қалған басқа машиналарды, сонымен бірге басқа машиналар тізімінде сақталған машиналардан сұрауға болады. Бұл жерде компьютер сирек кездесетін материалдарды іздеу жұмысын әлдеқайда жеңілдетеді. Қажет файл табылған кезде тұтынушы оны тек өзіне көшіріп алады.

Егер табылған жұмыс авторлық құқық заңымен қорғалған болса, тұтынушы, өзі білместен осы заңды бұзушы болып саналады (мұнда істің қай елде орын алып отырғаны да және әрбір нақты жағдайда қандай заңды қолдану керектігі соңғы рөл ойнамайды, себебі кейбір жағдайда файлды жүктеу - заңсыз, ал көшіріп алу – заңды). Ал ақпаратты жеткізуші кінәлі ме? Өз қатты дискісінде, дискке басқа адамдар да қолжеткізе алатын шарттар бар болса, ақша төлеп сатып алынған және заңды жолмен Интернеттен көшіріп алған жазбаларды сақтау, қылмыс болып санала ма? Егер сіздің, ешқашан құлпы ілінбеген кепе үйіңізге ноутбугі және сканері бар ұры кіріп, авторлық құқық заңымен қорғалған кітаптың көшірмесін жасап алып кетіп қалса, сіз бұл қылмысқа кінәлісіз бе, сіз бөтен біреудің авторлық құқығын қорғауыңыз керек пе?

Авторлық құқықпен байланысты тағы бір мәселе бар. Голливуд және компьютерлік индустрия арасында қиян-кескі тартыс жүріп келеді. Голливуд интеллектуалдық жеке меншікті қорғауды күшейтуді талап етеді, ал компьютерлік индустрия голливуд құндылығын олардың өздері қорғау керектіні айтады. 1998 жылдың қазан айында Конгресс **Сандық мыңжылдықтағы автолық құқық жайлы акт (DCMA – Digital millennium Copyright Act)** қабылдады. Мұнда, авторлық құқық заңымен қорғалған жұмыстағы қорғау механизмін сындыру, сонымен бірге сындыру технологиясын басқа біреуге хабарлау қылмыс болып саналады делінген. Сәйкес заңдылық акт Еуроодақта да қабылданған. Бір жағынан барлығы Қиыр Шығыс пираттары үшін мұндай акт өкім болып саналмайтынын ұмытып кеткен тәрізді, ал екінші жағынан көбі жаңа заң авторлық құқық иелері және қоғам мүддесі арасындағы тепе-теңдікті бұзды деп санайды.

Мына бір мысалды алайық. 2000 жылдық қыркүйек айында, аудиожазбаларды саудалаудың сенімді онлайн жүйесін құрастырумен абыржыған,

әуен индустриясымен байланысты консорциум, жүйені сындыруға тілек білдіргендердің барлығын шақырып жарыс ұйымдастырды (бұл жүйе құрастырудағы, шын мәнінде маңызды саты). Әртүрлі университеттердің ғалымдар тобы, ақпаратты қорғау саласының маманы, Пристоннан келген профессор Эдвард Фельтен, (Edward Felten) басшылығымен, шақыруды қабылдап, жүйені сындырды. Содан кейін зерттеу барысындағы нәтижелерді сипаттайтын мақала жазылды. Мақала, ақпаратты қорғау мәселелеріне арналған USENIX конференция жөнелтілді. Баяндама сәйкес деңгейде қарастырылып, қабылданды. Алайда, конференциядан аз уақыт бұрын Фельтен, егер мақала жарияланса ұйым авторларды DCMA Актін бұзғандары үшін сотқа беретіндігі жазылған, Америка дыбыс жазушы индустрия Қауымдастығынан (RIAA – Recording Industry Association of America) ескерту алды.

Ғалымдардың, ақпаратты қорғауға қатысты ғылыми мақалаларды жариялау заңды ма деген сауалмен, федералды сот ұйымдарына сұраныс жөнелтуден басқа шаралары қалмады. Істі ары қарай дамуы дыбыс жазушы индустриясы пайдасына шешілмейтініне күмәнданған ұйым өкілдері өздерінің Фельтенге деген наразылықтарын қайтарып алды, осымен іс жабылды. Әрине, дыбыс жазушы индустрия наразылығының себебі ұсынылған жүйенің әлсіздігі. Өздері алдымен жүйенің қорғанысын сындыруға адамдар шақырды, ал шақырған адамдар жүйені нақты сындырғаннан кейін оларды сотқа берді. Барлық шиеленістер шешілгеннен кейін мақала жарық көрді (Craver және басқалар, 2001). Мұндай текетірестердің алдағы уақытта да көптеп кездесетіндігі айқын.

Ал пират әуендер және пираттық фильмдер файл алмасу пирингілік желілерінің өсуіне әкелді. Әрине, бұл қарсы әрекет ретінде DCMA пайдаланған құқық иелерін қуантпайды. Сонымен бірге, файл алмасу желілері арқылы іздеу жүргізетін автоматты жүйелер де бар. Бұл жүйелер файл табылған кезде, авторлық құқықты бұзғандығына күмән келтіретін желі операторлары және тұтынушыларға ескертулер жөнелтеді. Құрама Штаттарда бұл ескертулерді **DMCA takedown notices – өшіру жайлы DMCA ескертпесі** деп атайды. Бұл іздеу жаппай қаруланууды еске салады, себебі авторлық құқықты бұзушыларды өте қиын. Тіпті, сіздің принтеріңізді де қылмыскер ретінде қабылдауға болады (Piatek және басқалар, 2008).

Талқыланып отырған тақырыппен, көптеген елдердегі сот шешімдерінің нәтижесі болған, **заңды пайдалану доктринасы (fair use doctrine)** тығыз байланысты. Бұл доктрина, авторлық құқық заңымен қорғалған тауарды сатып алушылардың осы тауардың көшірмесін жасауға, тіпті оның бөліктерін ғылыми мақсатта (мысалы, мектептер мен колледждерде оқытушы материал ретінде) пайдалануға және тығыздалған мұрағаттық көшірмесін жасауға шектеулі құқығы бар дейді. Тауарды пайдаланудың заңды екендігін қалай тексеруге болады? Көрсеткіштер мынадай: (1) коммерциялық пайдалану, (2) көшіріліп алынған деректер саны, (3) көшірмелердің сату көлеміне әсері. DCMA және Еуроодақта қабылданған сәйкес заңдар көшіруден қорғау жүйесін сындыруға тыйым салатын болғандықтан, сонымен бірге мұндай заңдар заң-

ды пайдалануға да тыйым салады. Іс жүзінде, DCMA тұтынушыларды тарихи қалыптасқан, тауарын сатып алған саудагерлерді қолдау құқығынан айырып отыр. Бұл идеяның жүзеге аспайтыны белгілі.

Өзінің деңгейі бойынша, авторлық құқық иелері және тіпті DCMA тұтынушылары арасындағы теп-теңдіктің ауытқуына әкеліп отырған оқиға – бұл **TGC (Trusted Computing Group – сенім жүктелген есептеулер тобы)** индустриясы өкілдері қолдайтын сенім **жүктелген есептеу (trusted computing).** Бұл жоба Intel және Microsoft бірлестігінің арқасында құрастырылуда. Бұндағы ой – операциялық жүйе және одан төмен деңгейде тұтынушы әрекеттерін бақылауды қолдауға көмек көрсету (мысалы, заңсыз жолмен көшіріліп алынған дыбыс жазбасын қайта қалыпқа келтіру) және заңсыз әрекеттерді орындауға тыйым салу. Жүйе кішігірім чиппен **TPM (Trusted Platform Module – сенім артқан платформа модулы)** толықтырылады. Чиптің жұмысына араласу өте күрделі. Бүгінгі күнде саудадағы көптеген дербес компьютерлер TPM-мен бірге сатылады. Мұндай жүйе авторлық құқығы бар авторлар жазған программаларға да тұтынушы дербес компьютерлерін іс-әрекеттерін басқаруға мүмкіндік берер еді, ал тұтынушы ештеңені ауыстыра алмайды. Осының салдары ретінде сұрақ туындайды, сенім жүктелген есептеулердегі сенімді тұлға кім? Тұтынушы емес екендігі айқын. Бұның үлкен қоғамдық толқыныс тудырары анық. Әрине, индустрияның ақпаратты қорғау мәселесіне назар аударғаны өте қуанышты жағдай, алайда күштің үлкен бөлігінің көптеген тұтынушыларды толғандыратын вирустармен, сындырушылармен, зиянкестермен және басқа да мәселелермен күресуге емес, авторлық құқық заңын күшейтуге бағытталғаны өкінішті-ақ.

Қорыта келгенде, заң авторлары және заңгерлерге көптеген жылдар бойы авторлық құқық иелері және тұтынушылар арасындағы қарым-қатынасты реттеуге тура келеді. Кибер кеңістіктің социумнан еш айырмашылығы жоқ: бұнда да, онда да әртүрлі топ мүдделері үнемі қақтығысып отырады, бұл қатаң күреске және сот отырыстарына әкеледі, осының нәтижесінде ерте ме, кеш пе ымыраға келеді. Осылай болады деп үміттенеміз және қарама-қайшылыққа толы жаңа технология пайда болғанша, кем дегенде, салыстырмалы тыныштық болады деп ойлаймыз.

8.11. ТҮЙІНДЕМЕ

Криптография – ақпарат құпиялығын, оның тұтастығы және сәйкестігін қамтамасыз ету үшін қолданылатын құрал. Қазіргі заманғы барлық криптографиялық жүйелер, алгоритмдер тілек білдіргендердің барлығына қолжетімді, ал кілттер – құпия сақталуы керек, дейтін Керкгоф жүйесіне негізделген. Көптеген алгоритмдер мәтінді шифрлау кезінде алмастыру және орын ауыстырудан тұратын, күрделі түрлендірулерді орындайды. Осылай бола

тұрса да, егер кванттық криптография қағидаларын жүзеге асыруға мүмкіндік туса, онда бір реттік блокноттар көмегімен, шын мәнінде сенімді криптожүйе құрастыруға болады.

Барлық криптография алгоритмдерін екі түрге бөлуге болады: симметриялы және ашық кілтті. Симметриялы кілтті алгоритмдер шифрлау кезінде, кілтпен параметрленген, тізбектік итерациядағы биттер мәнін бұзады. Осы типтегі ең танымал алгоритмдер – үштік DES және AES (Rijndael). Симметриялы кілтті алгоритмдер электронды шифроблокнот, шифр блоктарын іліндіру, ағындық шифр және т.б. режимдерде жұмыс істей алады.

Ашық кілтті алгоритмдердің ерекшелігі – шифрлау және кері шифрлауда әртүрлі кілттер қолданылады, мұнда кері шифрлау кілтін шифрлау кілтінің көмегімен есептеуге болады. Бұл қасиеттер кілтті ашық етуге мүмкіндік береді. Үлкен сандарды қарапайым көбейткіштерге жіктеу күрделілігіне негізделген RSA алгоритмі жиі қолданылады.

Ресми, коммерциялық және басқа да құжаттарға қол қою қажет. Симметриялы және ашық кілттерге негізделген, сандық қолтаңба генерациялаудың түрлі тәсілдері бар. Әдетте, MD5 немесе SHA-1 алгоритмдерінің көмегімен, қол қоюды қажет ететін мәлімдеме хэш-функциясы есептелінеді және қолтаңба мәлімдеменің өзіне емес, хэш-функцияның мәніне қойылады.

Ашық кілттерді басқаруды, ашық кілттер және олардың иесі болып келетін принципалдарды байланыстыратын құжат ретінде саналатын, сертификаттар көмегімен жүзеге асырылады. Сертификаттарға сенімді ұйымдар немесе соңында сенімді ұйымға әкелетін, сертификаттар тізбегін ұсына алатын кімдекім қол қоя алады. Бұл тізбектің бастапқы бөлігі (орталық басқару) алдын ала белгілі болуы тиіс. Ол үшін браузерлерге орталық басқарулардың көптеген сертификаттары енгізіледі.

Бұл криптографиялық тәсілдер желілік трафикті қорғауға мүмкіндік береді. Бұл үшін желілік деңгейде, хосттар арасында тасымалданатын дестелерді шифрлаумен айналысатын IPsec жүйесі жұмыс істейді. Желі аралық экрандар, пайдаланатын хаттама және портты сараптап, кіріс трафикті де, сонымен қатар шығыс трафикті де сүзбеден өткізе алады. Виртуалды жеке желілер, қорғаныстың сәйкес жоғарылатылған қасиетін ұсыну үшін бөлінген тораб негізіндегі желілерді иммитациялай алады. Соңында, сымсыз желілерде мәлімдемелерді барлығы бірдей оқымас үшін ақпаратты қорғау күрделі түрлері қажет. 802.11i хаттамалары осы қорғанысты қамтамасыз ете алады.

Екі жақ арасында байланыс сеансын орнатқан кезде, олар бір-бірін сәйкестендіруі және ортақ сеанс кілтін анықтаулары тиіс. Ол үшін әртүрлі сәйкестендіру хаттамалары қолданылады, олардың ішінде сенімді үшінші жақтың бар екендігін болжайтын хаттамалар да бар. Сонымен бірге, Диффи-Хеллман, Kerberos және ашық кілтті пайдаланатын хаттамалар да қолданылады.

Электронды поштадағы ақпаратты қорғауға, біз осы тарауда қарастырған тәсілдердің түрлі комбинацияларын пайдалану арқылы қол жеткізуге болады.

Мысалы, PGP мәлімдемені тығыздап, содан кейін оны құпия кілт көмегімен шифрлайды. Құпия кілт қабылдаушы ашық кілт көмегімен шифрланады. Бұдан басқа, хэш-функция мәлімдемесі шифрланады. Жөнелту алдында хэшке қолтаңба қою арқылы тұтастықты тексеруді орындауға болады.

Дүниежүзілік өрмектегі қауіпсіздік – ресурстарға ат беру қауіпсіздігінен басталатын маңызды мәселе. DNSsec DNS-серверлерді алдаудың жолын кеседі. Электронды коммерцияға арналған көптеген сайттар, клиент және сервер арасында сенімді, сәйкестендірілген байланыс сеансын орнату үшін SSL/TLS пайдаланады. Тасымалданатын программалар мәселесімен күресу үшін түрлі тәсілдер құрастырылған. Олардың ішінде жүктелетін кодты оқшауландырған ортаға орналастыру («құмсалғышқа») және кодтарға қол қою.

Интернетте, технологиялық мәселелер мемлекет саясатымен сабақтасқан көптеген сұрақтар туындайды. Біз олардың тек кейбіреуін ғана қарастырдық: құпиялық, сөз еркіндігі және авторлық құқық.

СҰРАҚТАР

1. Моноәліппелік орны ауыстыру шифрының көмегімен шифрланған келесі мәлімдемені кері шифрлаңыз. Тек әріптерден тұратын ашық мәтін Льюис Кэрролл танымал поэмасының үзіндісі.

mvyy bek mnyx n yvjjyr snijrh invq n muvjvdt je n idnvy
jurhri n fehfevir pyeir oruvdq ki ndq uri jhrnqvdt ed zb jnvy
Irr uem rntrhyb jur yeoijrhi ndq jur jkhjyri nyy nqlndpr
Jurb nhr mnvjvdt ed jur iuvdtyr mvyy bek pezr ndq wevd jur qndpr
mvyy bek, medj bek, mvyy bek, medj bek, mvyy bek wevd jur qndpr
mvyy bek, medj bek, mvyy bek, medj bek, medj bek wevd jur qndpr

2. Гомогендік шифр моноәліппелік орны ауыстыру шифрының бір нұсқасы болып табылады. Мұнда әліппедегі өлшемі *m* әріптер алдымен *0* және *m-1* арасындағы бүтін санға бейнеленеді. Содан кейін мәтіннің бүтін сандық бейнесі бүтінсандық шифро-мәтінге түрлендіріледі. Жеке әріп үшін түрлендіру функциясы $E(x)=(ax+b) \bmod m$, мұндағы *m* – әліппе өлшемі, ал *a* және *b* – маңыздылығы бірдей шифр кілттері. Зиянкес Бобтың гомогендік шифрды пайдаланып, шифрланған мәтін құрастырғанын білді. Ол шифрланған мәтіннің көшірмесін алып, ондағы жиі кездесетін әріп *«R»* содан кейін жиілігі бойынша *«K»* екенін анықтады. Зиянкестің кодты қалай сындырып, мәтінді алатынын анықтаңыз.

3. Келесі бағаналық орын ауыстыру шифрын сындырыңыз. Ашық мәтін компьютер жайлы танымал кітаптан алынған, сондықтан «computer» сөзінің кездесу ықтималдығы жоғары. Ашық мәтін тек әріптерден тұрады (бос орынсыз). Оқуға ыңғайлы болу үшін шифрланған мәтін бес әріптен тұратын блоктарға бөлінген.

aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc
nntii imiha oofpa gsivt tpsit lbolr otoex

4. Алиса өзінің Бобқа арналған мәлімдемесін шифрлау үшін орын ауыстыру шифрын қолданды. Жоғары қауіпсіздік үшін ол орын ауыстыру шифрын алмастыру кілтімен шифрлап, шифрды өз компьютерінде сақтады. Труди алмастыру кілтіне қолжеткізді. Труди Алисаның мәлімдемесін кері шифрлай ала ма? Неліктен иә және неліктен жоқ?

5. Жетпіс жеті биттік бір реттік блокнот құрастырыңыз. Осы блокноттың көмегімен *8.3-суретте* көрсетілген шифрдан «Hello World» сөзін алуға бола ма?

6. Сіз тыңшысы және сіздің бақытыңызға қарай қоластыңызда, сансыз көп кітаптары бар кітапхана бар. Сіздің операторыңыздың да осындай кітапханасы бар. Сіздер *«Сақина әміршісі»* кітабын бір реттік блокнот ретінде пайдалануға келістіңіздер. Қолда бардың барлығын шексіз бір реттік блокнот құрастыру үшін қалай қолданатыныңызды түсіндіріңіз.

7. Кванттық криптографияны пайдаланған кезде, қажет болған жағдайда бір битке сәйкес келетін бір реттік фотон шығаратын фотон зеңбірегі болу керек. Бір биттің, өткізгіштік қабілеттілігі 250 Гбит/с оптыталшық торап бойымен неше фотон тасымалдайтынын есептеңіз. Фотон ұзындығы толқын ұзындығына тең деп санаймыз, яғни 1 мкм (осы есеп қажеттілігі үшін). Оптыталшықтағы сәуле жылдамдығы 20 см/нс тең.

8. Егер зиянкес кванттық криптографияда қолданылған фотонды ұстап алып, кері генерациялай алса, кейбір мәндер дұрыс болмайды, демек Боб қабылдайтын бір реттік блокнотта да қателіктер пайда болады. Боб қабылдаған блокноттың қандай бөлігінде (орта есеппен) қателіктер болады?

9. Криптографияның іргетасты қағидасы барлық мәлімдемелер молшылықты болуы керек дейді. Алайда, біз молшылықтың зиянкеске құпия кілтті дұрыс шешкенін түсінуге мүмкіндік беретінін білеміз. Молшылықтың екі түрін қарастырыңыз. Бірінші түрінде ашық мәтіннің алғашқы *n* битінде белгілі тізбек бар. Екінші түрінде мәлімдеменің соңғы *n* битінде хэш бар. Осы екі молшылық түрлері қауіпсіздік тұрғысынан эквивалентті ме? Жауабыңызды түсіндіріңіз.

10. *8.5-суреттегі* S-блоктар және P-блоктар бір-бірін алмастырады. Бұл эстетикалық тұрғыдан жасалған ба немесе шифратордың осылай ұйымдастырылуы, алдымен барлық P-блоктар, содан кейін барлық S-блоктар орналасқан жағдайға қарағанда кодты сенімді ете ме? Өз жауабыңызды талқылаңыз.

11. Ашық мәтін тек ASCII-символдарының кіші әріптерінен, бос орыннан, үтір, қоснүкте, нүктелі үтір символы, жаңа жолға көшу және каретканы қайтару символдарынан тұрады деген ақпаратқа сүйеніп, DES шифрынан шабуыл құрастырыңыз. Ашық мәтіндегі жұптық биттері жайлы ешнәрсе белгісіз.

12. Жоғарыдағы мәтінде, 1 кілтті 1 нс өңдей алатын, миллион процессормен жабдықталған машинаға 128-биттік AES версиясының шифрын

сындыру үшін 10^{16} жыл қажет делінген. Осы уақытты 1 жылға дейін қалай қысқартуға болатынын санайық. Ол үшін компьютерлер 10^{16} есе жылдам жұмыс істеуі керек. Егер Мур заңдылығы компьютерлердің есептеу қуаты әр 18 ай сайын екі еселеніп отырады десе, онда параллель компьютер шифрды сындыру уақытын 1 жылға дейін қысқарту үшін неше жыл өтеді?

13. AES кілттерінің ұзындығы 256 бит болуы мүмкін. Осы режимде кілттің неше нұсқасы қолжетімді? Сіз қандай да бір ғылымда, мысалы, физика, химия немесе астрономияда осындай шамаларға сүйенетін түсініктерді таба аласыз ба? Интернеттен үлкен сандарға арналған материалдарды табыңыз. Осы зерттеуден қорытынды шығарыңыз.

14. Мәлімдеме, блоктарды іліндіру режиміндегі DES шифрымен шифрланған делік. Тасымалдау кезінде, шифрланған мәтін болктарының бірінде бір бит 0-ден 1-ге өзгерді. Мәтінің қандай бөлігі бұзылады?

15. Тағы да блоктарды іліндіру арқылы шифрлауды қарастырайық. Бұл жолы шифрланған мәтін блоктарының арасында тасымалдау кезінде бір нөлдік бит қойылып кетті. Бұл жағдайда мәтіннің қандай бөлігі бұзылады?

16. Блоктарды іліндіріп шифрлауды кері байланыс режимінде шифрлаумен, үлкен файлды тасымалдау үшін қажет шифрлау операциялары саны тұрғысынан салыстырыңыз. Қандай режим тиімді?

17. Код символдары a=1, b=2 және т.с.с., y=25, z=26 болғанда, ашық кілтті RSA алгоритімін пайдаланып орындаңыз.

а) Егер p=5 және q=13 болса, d үшін қолжетімді бес мәнді табыңыз.
ә) Егер p=5, q=31 және d=37 болса, e табыңыз.
б) Егер p=3, q=11 және d=9 болғандағы e тауып, «hello» сөзін шифрлаңыз.

18. Алиса және Боб қарым-қатынас кезінде шифрлау үшін RSA ашық кілттерін пайдаланады. Труди олардың, ашық кілттер n жұбын анықтау үшін қарапайым сан қолданатындарын анықтады. Басқа сөзбен айтқанда, Труди $n_a=p_a\times q$ және $n_b=p_b\times q$ екенін анықтайды. Труди бұл ақпаратты Алиса кодын сындыру үшін пайдалана ала ма?

19. *8.13-суретте* көрсетілген санауыш режимін қарастырыңыз, бірақ инициализациялау векторының мәнін нөлге тең деп алыңыз. Нөлді пайдалану жалпы шифр сенімділігіне қауіп төндіре ме?

20. *8.17-суретте* біз Алисаның Бобқа қол қойылған мәлімдемені қалай жөнелтетінін көреміз. Егер Труди *P* ауыстырып қойса, Боб оны байқайды ма? Егер Труди *P*-ны да қолтаңбаны да ауыстырса не болады?

21. Сандық қолтаңбалардың бір кемшілігі бар: оларды жалқаулар пайдалана алады. Электронды коммерцияда келісімшарт жасағаннан кейін, әдетте, клиент оған өз SHA-1 хэшімен қол қоюы тиіс. Егер тұтынушы хэш және контракт сәйкестігін тексермесе, ол кездейсоқ басқа келісімшартқа қол қою қаупі бар. Айталық, кезбе және өлмес мафия осыдан ақша жасауды ұйғарды делік. Бандиттер ақылы веб-сайт құрастырды (мысалы, порнография, азарт ойындар және т.б. тұратын) және жаңа клиенттердің несие карталарының нөмірін сұрайды делік. Содан кейін тұтынушыға, қызметті пайдаланып, оның ақысын несие картасымен төлегісі келетіндігі жазылған келісімшарт жөнелтіледі. Келісімшартқа қол қою керек, бірақ сайт иесі клиенттердің контрактты хэшпен сәйкестендіре қоймасын біледі. Зиянкестердің бриллиантты заңды зергерлік интернет-дүкенде, клиенттер есебінен қалай сатып алатынын көрсетіңіз.

22. Математикалық топта 25 студент бар. Айталық студенттердің барлығы жылдың бірінші жартысында туған – бірінші қаңтар және отызыншы маусым аралығында. Кем дегенде, екі студенттің туған күні сәйкес келу ықтималдығы қандай? Ешқайсының туған күні 29 ақпанға келмейді деп жорамалдайық, сондықтан қарастырылатын туған күндер нұсқаларының жалпы саны 181-ге тең.

23. Элен Мэрилинге өзінің Томға лауазым беру жайлы алдағандығын айтқаннан кейін, Мэрилин мәселені шешу үшін өз мәлімдемелерінің мазмұнын диктофонға жазуды ұйғарды. Жаңа хатшы сәйкес мәтінді тек теруі керек. Содан кейін Мэрилин терілген мәлімдеменің нақты өзі айтқан сөздерден тұратындығын тексеру үшін өз терминалында оқиды. Жаңа хатшы, туған күндер жайлы есепте жалған мәлімдеме құрастыру үшін шабуылды пайдалана ала ма? Қолдана алса, қалай? *Көмек:* пайдалана алады.

24. Алисаның Бобтың ашық кілтін сәтті алған мысалды қарастырыңыз (8.20-сурет). Айталық, олар ортақ жабық кілтті орнатты, алайда Алиса енді Бобтың ашық кілтін білгісі келеді. Осы жағдайда оны еш қауіпсіз алудың жолы бар ма? Бар болса, қандай?

25. Алиса Бобпен ашық кілтпен шифрлауды пайдаланып сөйлескісі келеді. Ол болжам бойынша, Боб деп саналатын адаммен байланысады. Алиса оның ашық кілтін сұрайды және ол кілтті Орталық басқару қол қойған X.509 сертификатымен қоса жөнелтеді. Алисада ОБ ашық кілті бар. Енді Алиса, шын мәнінде Бобпен хабарласып отырғанына көз жеткізу үшін не істеуі керек? Айталық, Бобқа кіммен хабарласып отырғаны маңызды емес (яғни, бұл жерде Боб деген атпен біз, мысалы, жалпыға қолжетімді қызметті айтамыз).

26. Айталық, жүйе сертификатты басқарудың ағаш тәріздес иерархиясына негізделген PKI қолданады. Алисаның Бобпен хабарласқысы келеді және байланыс орнатылғаннан кейін одан X сертификатты басқарумен қол қойылған сертификат алады. Алиса X мүлдем білмейді және ол жайлы ешнәрсе естіген жоқ делік. Байланыстың екінші басында, шынымен Боб екенін тексеру үшін Алиса не істеуі керек?

27. Егер машиналардың бірі NAT-блоктан кейін орналасса, сәйкестендіру тақырыбы бар IPsec транспорттық режимде қолдануға бола ма? Жауабыңызды түсіндіріңіз.

28. Алиса SHA-1 хэштерін пайдаланып, Бобқа мәлімдеме жөнелткісі келеді. Ол қолайлы алгоритмді пайдалану жайлы сізден кеңес сұрайды. Сіз қандай кеңес берер едіңіз?

29. Желіаралық экрандарды кіріс трафикті сараптау режимінде баптау керектігінің бір себебін атаңыз. Енді желіаралық экрандарды шығыс трафикті сараптау режимінде баптау керектігінің бір себебін атаңыз. Осындай сараптау қаншалықты сәтті деп ойлайсыз?

30. Интернетте өз сайттарының арасында қауіпсіз байланысты қамтамасыз ету үшін, ұйым виртуалды жеке желі орнатты. Осы ұйымның қызметкері, Джим, виртуалды жеке желіні өз бастығы Мэримен әрекеттесу үшін пайдаланады. Джим және Мэри арасындағы шифрлауды немесе басқа қауіпсіздік механизмін қажет етпейтін коммуникация типін сипаттаңыз және шифрлау немесе қауіпсіздік механизмін қажет ететін коммуникация типін келтіріңіз. Өз жауабыңызды түсіндіріңіз.

31. *8.30-суретте* бейнеленген хаттамадағы бір мәлімдемені хаттама айналық шабуылға тұрақты болатындай етіп өзгертіңіз. Енгізген өзгертулеріңіздің мағынасын түсіндіріңіз.

32. Алиса және Боб арасында құпия кілтті орнату үшін Диффи-Хеллман алгоритмі қолданылады. Алиса Бобқа (227, 5, 82) жөнелтеді. Боб (125) жауап қайтарады. Алисаның құпия саны $x=12$, Бобтың құпия саны $y=3$. Алиса мен Бобтың құпия кілтті қалай есептейтінін көрсетіңіз.

33. Егер екі тұтынушы ортақ кілттерді пайдаланбаса және сертификаттары да болмаса да олар Диффи-Хеллман алгоритмінің көмегімен ортақ жабық кілт орната алады.

а) Осы алгоритмнің «ортадағы адам» шабуылына қалай ұшырайтынын түсіндіріңіз.
ә) Егер n немесе g құпия болса шабуылға ұшырау қалай өзгереді?

34. Неліктен *8.35-суретте* бейнеленген хаттамада A шифрланған сеанс кілтімен бірге ашық мәтінмен жөнелтіледі?

35. Нидхэм-Шредер (Needham-Schroeder) хаттамасында Алиса R_A және R_{A2} екі шақыру құрастырады. Бір ғана шақыруды пайдалану жеткілікті емес пе?

36. Айталық, ұйым сәйкестендіру үшін Kerberos тәсілін пайдаланады. Бұл жағдайда қауіпсіздік термині және жұмыс қабілеттілік сәйкестендіру серверінің немесе билеттер беру серверінің істен шығу жүйесіне қалай әсер етеді?

37. Алиса Бобпен коммуникацияны сәйкестендіру үшін *8.39-суретте* көрсетілген ашық кілтті сәйкестендіру хаттаманы пайдаланады. Алайда, 7-мәлімдемені жөнелте отырып, Алиса R_B шифрлауды ұмытып кетті. Осылайша, Труди R_B мәнін біледі. Алиса мен Бобқа коммуникацияның қауіпсіз екендігіне көз жеткізу үшін сәйкестендіру процедурасын жаңа параметрлермен қайталау қажет пе? Жауабыңызды түсіндіріңіз.

38. *8.39-суретте* көрсетілген ашық кілтпен сәйкестендіру хаттамасында 7-мәлімдемеде кездейсоқ R_B саны K_S кілтімен шифрланған. Осы шифрлау қажет пе немесе санды кері ашық кілтпен жөнелтуге бола ма? Жауабыңызды түсіндіріңіз.

39. Магниттік карталар және PIN-кодты пайдаланатын кассалық аппараттардың бір кемшілігі бар: кассир-зиянкес, болашақта қосымша (жалған) транзакциялар жөнелту үшін картадағы барлық ақпаратты оқып, PIN-кодпен қоса өзіне сақтайтындай етіп, өз кассалық

аппаратының санауыш құрылғысын түрлендіре алады. Кассалық терминалдардың келесі буынында толыққанды процессорлары, пернетақтасы және кішігірім дисплейі бар карталар пайдаланылатын болады. Осы жүйе үшін кассир-зиянкес сындыра алмайтын хаттама құрастырыңыз.

40. PGP мәлімдемені көп абоненттік тарату арқылы жөнелтуге бола ма? Қандай шектеулер қолданылатын болады?

41. Интернетте жаппай PGP қолданылады делік. Осы жағдайда еркін интернет адрес бойынша PGP-мәлімдеме жөнелтіп, мәлімдеменің келесі бетте дұрыс кері шифрланатынына толық сенімді болуға бола ма? Өз жауабыңызды талқылаңыз.

42. *8.43-суретте* көрсетілген шабуылда бір қадам қалып қойған. Шабуылдың сәттілігі үшін ол қадам міндетті емес, алайда оның бар болуы мүмкін деген күмәнді азайтар еді. Сонымен, не қалып қойды?

43. Деректерді тасымалдау SSL хаттамасы екі нонстың және дайындық кілтінің болуын талап етеді. Нонстар қандай рөл ойнайды (егер олар қандай да бір рөл ойнайтын болса)?

44. Өзіңізге өлшемі 2048-де 512 пиксель суретті елестетіңіз. Сіздің көлемі 2,5 Мбайт ақпаратты шифрлағыңыз келеді. Файлдың қандай бөлігін сіз осы суретте шифрлай аласыз? Егер файл нақты өлшемінің төрттен бір бөлігіне дейін тығыздалса, өз файлыңыздың қандай бөлігін шифрлай алар едіңіз? Өз есептеулеріңізді көрсетіңіз.

45. *8.49 ә-суреттегі* бейнеде Шекспирдің бес пьесасының ASCII-мәтіні бар (бейнеге қарап олай демейсің). Алаңдар арасына мәтін емес, әуенді жасыруға бола ма? Болмайтын болса, неліктен?

46. Сізге көлемі 60 Мбайт мәтіндік файлды, сурет файлының кіші биттерінде стеганографияны пайдаланып шифрлауға берді. Файлды толығымен шифрлау үшін қандай көлемдегі сурет қажет болады? Егер файл алдын ала өзінің бастапқы көлемінің төрттен біріне дейін тығыздалса қандай көлем қажет? Жауапты пиксельмен беріп, есептеулеріңізді көрсетіңіз. Айталық, сурет көлемінің қатынасы 3:2, мысалы, 3000x2000 пиксель.

47. Алиса бірінші типтегі аноним таратудың тұрақты тұтынушысы болатын. Ол өзінің сүйікті конференциясына *alt.fanclub.alice* қалағанынша

мәлімдеме жөнелте алатын, бұл мәлімдемелердің авторы Алисадан келетінін барлығы білетін, себебі мәлімдемелердің барлығына бір жасырын атпен қол қойылған болатын. Егер тарату дұрыс жұмыс істейді, Трудидің Алиса атынан мәлімдеме жазуға мүмкіндігі жоқ деп жорамалдасақ. Бірінші типтегі аноним таратулардың барлығы жабылғаннан кейін, Алиса шифропанктік таратуларға көшті және өз конференциясында жаңа тақырыпты талқылай бастады делік. Жаңа жағдайда бөтен біреу атынан хат жазбақ болған Труди әрекетінен қорғаудың жолын ұсыныңыз.

48. Интернеттен құпиялыққа байланысты қызықты бір оқиғаны іздеп тауып, осы тақырып бойынша есеп жазыңыз.

49. Интернеттен тауарды адал пайдаланушы тұтынушы және авторлық құқық заңы арасындағы текетіресті сипаттайтын кезекті жағдайды іздеп тауып, өзіңіздің осы жағдайдан білгендеріңіз жайлы есеп жазыңыз.

50. Кіріс деректерді кілтті ағынмен модуль-2 бойынша қосып шифрлайтын программа жазыңыз. Кілтті ағынды құрастыру үшін жақсы кездейсоқ сандар генераторын табыңыз немесе өзіңіз жазыңыз. Программа сүзбе тәрізді жұмыс істеуі тиіс, кірісте ашық мәтінді қабылдап, шығыстағы стандартты құрылғыға шифрды беруі керек (және керісінше). Программаның бір ғана параметрі болу керек: кездейсоқ сандар генераторын іске қосатын кілт.

51. Деректер блогының SHA-1 хэшін есептейтін процедура жазыңыз. Процедураның екі параметрі болу керек: кіріс буферіне көрсеткіш және 20-байттық шығыс буфер көрсеткіші. SHA-1 спецификациясын табу үшін Интернеттен FIPS 180-1 іздеңіз, осы құжатта толық сипаттама бар.

52. ASCII-символдар ағынын қабылдап, оны блоктарды іліндіру режимінде шифрлайтын функция жазыңыз. Блок көлемі 8 байт болу керек. Программа енгізудің стандартты құрылғысынан ашық мәтін қабылдауы және шифрланған мәтінді стандартты шығару құрылғысында басып шығаруы тиіс. Бұл есепті шешу үшін сіз кіріс ағынының соңына жеткендікті және/немесе блокты аяқтау үшін кірістіруді қашан орындау керектігін анықтауға кез келген қолайлы жүйені пайдалануыңызға болады. Сіз деректерді шығару форматын да таңдай аласыз, тек ол екі ойлы болмауы тиіс. Программа тек параметрді қабылдауы тиіс:

1. Инициализациялау векторына көрсеткіш.
2. Орын ауыстыру шифрының ығысуын көрсететін *k* нөмірі. Сондықтан

әріптің ASCII мәні әліппеде алдында тұрған k-шы мәнмен шифрланады.

Мысалы, егер x=3 болса, онда A әрпі D ретінде шифрланады, B E ретінде және т.с.с. ASCII жиынтығындағы соңғы мән бойынша дұрыс жорамал жасаңыз. Өз кодыңызда кіріс ағынға және шифрлау алгоритміне қатысты жорамалдардың барлығын тізіп отырғаныңызға көз жеткізіңіз.

53. Бұл есептің мақсаты – сізге RSA жұмыс механизмі жайлы дұрыс түсінік беру. Параметр ретінде қарапайым *p* және *q* сандарын алатын функция жазыңыз. Фунцкия осы параметрлерді пайдаланып ашық және құпия RSA кілттерін есептеп, шығыс деректер ретінде *n, z, d* және *e* шығаруы тиіс. Сонымен бірге, функция ASCII ағындарының мәнін қабылдап, осы кіріс ағынды есептелген RSA кілттерінің көмегімен шифрлауы керек. Программа ашық мәтінді стандартты енгізу құрылғысынан қабылдауы және шифрланған мәтінді стандартты шығыс құрылғысына басып шығаруы керек. Шифрлау әр мән бойынша орындалуы тиіс, яғни кіріс деректердің әр мәні басқа мәндерге тәуелсіз шифрлануы тиіс. Бұл есепті шешуде кіріс ағынның соңына жеткендікті анықтау үшін кез келген қолайлы жүйені пайдалануға болады. Сіз деректерді шығару форматында таңдай аласыз, тек ол екі ойлы болмауы тиіс. Өз кодыңызда кіріс ағынға және шифрлау алгоритміне қатысты жорамалдардың барлығын тізіп отырғаныңызға көз жеткізіңіз.

9-ТАРАУ
ОҚУҒА АРНАЛҒАН НҰСҚАУЛАР ЖӘНЕ ӘДЕБИЕТТЕР

Біз компьютерлік желілерді оқуды аяқтадық, алайда бұл – тек басы ғана. Көптеген қызықты тақырыптар егжей-тегжейсіз үстіртін қарастырылды, ал кейбір сұрақтар орын жетпегендіктен мүлдем қалып кетті. Бұл тарауда компьютерлік желіні оқуды әрі жалғастырғысы келетін оқырмандарға арналған қосымша әдебиеттер тізімі келтірілген.

9.1. ӘРІ ҚАРАЙ ОҚУҒА АРНАЛҒАН ӘДЕБИЕТТЕР

Компьютерлік желілер және таратылған жүйелердің барлық аспектілеріне қатысты кітаптар өте көп. Осы саладағы мақалаларды жариялайтын журналдар ішінен *IEEE/ACM Transactions on Networking* және *IEEE Journal on Selected Areas in Communications* ерекше атап өтуге болады.

ACM Special Interest Groups on Data Communications (SIGCOMM) және Mobility of Systems, Users, Data, and Computing (SIGMO-BILE) мерзімді басылымдарында, әсіресе, дамып келе жатқан тақырыптар бойынша көптеген қызықты мақалалар жарияланады. Бұл *Computer Communication Review* and *Mobile Computing and Communications Review* басылымдары.

Бұдан басқа, Электротехника және электроника инженерлер институты (IEEE) тағы үш журнал шығарады — *IEEE Internet Computing, IEEE Network*

Magazine және *IEEE Communications Magazine.* Бұл журналдарда компьютерлік желілермен байланысты зерттеулер жайлы ақпараттар, оқу мақалалары және шолулар жарияланады. Алғашқы екі журнал көбіне стандарттар, құрылым және программалық жабдықтамаларға арналған. Ал *IEEE Communications Magazine* журналы коммуникациялық технологияларды (оптыталшықты, спутниктік байланыс және т.с.с.) түсіндіруге бағытталған.

Жыл сайын немесе екі жылда бір желілерге арналған бірнеше конференциялар өткізіледі. Жеке алғанда, *SIGCOMM conference, NSDI (Symposium on Networked Systems Design and Implementation), MobiSys (Conference on Mobile Systems, Applications, and Services), SOSP (Symposium on Operating Systems Principles)* және *OSDI (Symposium on Operating Systems Design and Implementation)* конференцияларына назар аударыңыздар.

Төменде біз тараулар бойынша топтастырылған әдебиеттерді келтіреміз. Ұсынылатын әдебиеттердің немесе жеке тараулардың басым бөлігі үйрету кітаптары немесе шолулар болып келеді. Толық сілтеме *9.2-бөлімде* берілген.

9.1.1. Кіріспе және арнайы емес әдебиеттер

Comer, *The Internet Book*, 4-басылым.

Интернеттің қарапайым және түсінікті сипаттамасын іздегендердің барлығының осы басылымды қарағаны дұрыс. Бұл кітапта Интернет қызметтері, хаттамаларының және технологияларының дамуы мен тарихы, тіпті, осы салаға жаңа келген оқырманға да түсінікті тілмен сипатталған.

Computer Communication Review, 25th Anniversary Issue, Jan. 1995

Бұл арнайы басылым 1995 жылдан бастап Интернеттің қалай дамып келе жатқандығы жайлы маңызды мақалалардың барлығын жинақтап келеді. Осы басылымда жарияланған мақалалар TCP, тасымалдау, DNS, Ethernet және толық құрылым жайлы ақпараттарды қамтиды.

Crovella and Krishnamurthy, *Internet Measurement*

Интернеттің қаншалықты жақсы жұмыс істейтіндігін қайдан білуге болады? Бұл сұрақтың жауабы жоқ, себебі, Интернетке ешкім жауап бермейді. Аталмыш кітапта Интернет жұмысын: желілік инфроқұрылымнан бастап қосымшаға дейін өлшеу үшін құрастырылған, дамытылған тәсілдер сипатталған.

IEEE Internet Computing, Jan.-Feb. 2000

Жаңа мыңжылдықтың алғашқы басылымында *IEEE Internet Computing* журналы, барлығы күткендей, Интернетті құрастыруға атсалысушылардың жаңа ғасырда Интернет қандай болатындығы жайлы ойларын жариялады. Пікірталасқа Поль Бэрэн (Paul Baran), Лоуренс Робертс (Lawrence Roberts),

Леонард Кляйнрок (Leonard Kleinrock), Стэфан Крокер (Stephen Crocker), Дэнни Коэн (Danny Cohen), Боб Мэткалф (Bob Metcalfe), Бил Гейтс (Bill Gates), Билли Джой (Bill Joy) және тағы басқа эксперттер қатысқан. Он жылдан соң олардың болжамдарының қаншалықты жүзеге асқандығын қараңыз.

Kipnis, "Beating the System: Abuses of the Standards Adoption Process"

Стандарттау комитеті мүмкіндігінше адал және құрастырушыларға тәуелсіз жұмыс істеуде, бірақ өкінішке орай, кейбір компаниялар бұл жүйені бұзуға тырысады. Мысалы, компаниялар алдымен стандарттарды құрастыруға көмектеседі, ал стандарт бекітілгеннен кейін оның осы компанияға тиесілі патенттерге негізделгенін алға тартып, баға мен лицензия беру сұрақтарын өзіміз шешеміз деуде. Бұл материал стандарттаудың жағымсыз тұстарын білгісі келгендерге пайдалы болады.

Hafner and Lyon, *Where Wizards Stay Up Late*
Naughton, *A Brief History of the Future*

Дегенмен, Интернетті құрастырған кім? Көптеген адамдар өздерін құрастырушылар қатарына қосқанды қалайды. Кейбіреулердің еңбегі бұған заңды түрде лайық: дестелер коммутациясы жайлы есеп жазған Пол Барн (Paul Baran), ARPANET құрылымын жасаған, әртүрлі университеттерде жұмыс істейтін адамдар, алғашқы IMP-ті программалаған BBN қызметкерлері, TCP/IP құрастырған Боб Кан (Bob Kahn) және Винт Серф (Vint Cerf) және тағы басқалар. Бұл кітаптар 2000 жылға дейінгі Интернет тарихын сипаттады және көптеген күлкілі әңгімелерден тұрды.

9.1.2. Физикалық деңгей

Bellamy*, Digital Telephony*, 3-басылым.

Бұл көрнекті басылымда, телефон жүйесі жайлы сіз білгіңіз келетін егжей-тегжейдің барлығы жазылған. Деректерді тасымалдау және мультиплекстеу, сандық коммутация, талшықты оптика, мобильді телефон және DSL-ге арналған тараулар өте қызықты.

Hu and Li, "Satellite-Based Internet: A Tutorial"

Интернетке спутник арқылы қолжеткізу, жөнелтуші тораптарды пайдаланудан өзгеше. Мұнда тек кідіріс қана емес, сонымен бірге маршруттау және коммутация да есепке алынуы тиіс. Авторлар Интернетке қолжеткізу үшін спутниктік жүйелерді пайдалану мәселелерін қарастырған.

Joel, "Telecommunications and the IEEE Communications Society"

Бұл басылымда телекоммуникация тарихы телеграфтан басталып, 802.11 желі стандарттарына дейін, қысқа және өте түсінікті формада сипатталған.

Сіздер бұл басылымнан радио, телефон, аналогтық және сандық коммутация, суасты кабельдер, деректерді сандық тасымалдау, теледидарлық хабарлау, спутник, кабельдік теледидар, оптикалық байланыс тораптары, мобильді телефон, дестелерді коммутациялау, ARPANET, әрине Интернет жайлы бөлімдерді табасыздар.

Palais, *Fiber Optic Communication*, 5-басылым.

Әдетте, оптыталшықты технологияларға арналған басылымдар мамандарға арналып жазылады, алайда бұл кітап жалпыға түсінікті тілмен жазылған. Бұл кітапта су толқындары, жарық көзі, жарық детекторы, байланыстырушы муфталар, модуляция, шу және басқа да тақырыптар қамтылған.

Su, *The UMTS Air Interface in RF Engineering*

Бұл кітапта ұялы жүйенің 3G буынына толық шолу жасалған. Негізінен радиоинтерфейске, яғни мобильді телефондар және желілік инфроқұрылым арасында қолданылатын сымсыз хаттамаларға назар аударылған.

Want, *RFID Explained*

Бұл кітап – физикалық деңгейдің RFID технологиясының қалай жұмыс істейтіндігін білгісі келетіндерге арналған және оқуға жеңіл жазылған оқулық. Кітапта RFID-тің барлық аспектілері қамтылған. Сонымен бірге, RFID пайдалану және осы бағытта алынған тәжірибелер де келтірілген.

9.1.3. Арналық деңгей

Kasim, *Delivering Carrier Ethernet*

Қазіргі кезде Ethernet тек жергілікті байланыс технологиясы ғана емес. Ethernet-ті пайдаланудың жаңа формасы – деректерді алыс қашықтыққа Ethernet сенімділігімен тасымалдау арнасы ретінде қолдану. Кітапта осы тақырыпты егжей-тегжейлі қамтитын эссе келтірілген.

Lin and Costello, *Error Control Coding*, 2-басылым.

Қателіктерді анықтап, түзетуші кодтар – компьютерлік желілер сенімділігінің негізі. Бұл танымал оқулық осы кодтардың ішіндегі маңыздыларын: қарапайым сызықтық Хэмминг кодынан бастап, аз ғана тығыздығы бар, жұптықты тексеруге арналған ең күрделі кодтарға дейін сипаттайды. Авторлар алгебраны аз қолдануға тырысқанымен, ол өте көп.

Stallings, Data and Computer Communications, 9-басылым.

Кітаптың екінші бөлімінде деректерді сандық тасымалдау және әртүрлі арналар, сонымен бірге қателіктерді анықтау, қайта тасымалдаумен қателерді бақылау және ағынды басқару жайлы айтылған.

9.1.4. Ортағы қолжеткізу ішкі деңгейін басқару

Andrews et al., *Fundamentals ofWiMAX*

Бұл WiMAX технологиясы жайлы жан-жақты жазылған кітап: сымсыз кеңжолақты тасымалдаудан бастап, мульти қолжеткізу, OFDM және бірнеше антеннаны қолданатын сымсыз тәсілдерді қоса қамтыған. Кітаптың оқулық стилі, осы күрделі материалды жеңіл қабылдауға мүмкіндік береді.

Gast, *802.11 Wireless Networks*, 2-басылым.

Бұл кітап – 802.11 хаттамасы және технологияларымен танысуға жақсы кіріспе. Ол MAC деңгейінен бастап, содан кейін басқа физикалық деңгейлер және қауіпсіздік жайлы материалдарды ұсынады. Алайда, екінші басылымда 802.11n жайлы айтарлықтай жаңа материалдар аз.

Perlman, *Interconnections*, 2-басылым.

Бұл көпірлер, маршруттауыштар және жалпы маршруттау жайлы жазылған қызықты, сенім артарлық кітап. Кітап авторы IEEE 802 стандарты желілерінде қолданылатын байланыстырушы ағаш алгоритмін құрастыруға қатысқан және желілік технологиялардың түрлі аспектілері бойынша алдыңғы қатарлы әлемдік эксперттерінің бірі болып саналады.

9.1.5. Желілік деңгей

Comer, *Internetworking with TCP/IP*, 1-том, 5-басылым.

TCP/IP хаттамалары жиынтығы жайлы анағұрлым толық еңбектің бесінші басылымы. Кітіптың бірінші жартысы толығымен IP және онымен байланысқан желілік деңгей хаттамаларына арналған. Басқа бөлімдерде, негізінен жоғары деңгейлер сипатталған.

Grayson et al., *IP Design for Mobile Networks*

Дәстүрлі телефон желілері мен Интернет, IP негізінде жүзеге асырылатын мобильді желілермен қарама-қарсы позицияларда орналасқан. Бұл кітап IP-хаттаманы қолдананып, мобильді телефон қызметін қолдайтын желіні қалай құрастыру керектігін сипаттайды.

Huitema, *Routing in the Internet*, 2-басылым.

Егер сіз маршруттау хаттамалары жайлы терең мағлұмат алғыңыз келсе, онда бұл кітап сізге арналған. Мұнда аттары белгілі алгоритмдермен (мысалы, RIP және CIDR) бірге, белгісіз (мысалы, OSPF, IGRP және BGP) алгоритмдер де егжей-тегжейлі сипатталған. Кітап ескі болғандықтан, онда жаңа құрылымдар жайлы айтылмаған, бірақ кітаптың жазылу тілі өте жақсы.

Koodli and Perkins, *Mobile Inter-networking with IPv6*

Бұл кітапта желілік деңгейдің екі маңызды құрастырылымы: IPv6 және Mobile IP ұсынылған. Кезінде Перкинс Mobile IP-ді қозғаушы күштің бірі болған.

Nucci and Papagiannaki, *Design, Measurement and Management of Large-Scale IP Networks*

Біз желінің қалай жұмыс істейтіндігі жайлы көп айттық, бірақ егер сіз интернет-провайдер болсаңыз оны қалай құрастыру, орнату және басқару керектігі жайлы ешнәрсе айтылмады. Кітап осы аралықты толықтырады. Мұнда трафикті ұйымдастырудың қазіргі заманғы тәсілдері және интернет-провайдерлердің желіні пайдаланып, қалай қызмет көрсететіндіктері жазылған.

Perlman, *Interconnections*, 2-басылым.

Кітаптың 12-ден 15-тарауына дейін автор ауқымды және жергілікті желілер үшін бірадрестік және көпадрестік тарату алгоритмдерін құрастыруды және оларды әртүрлі хаттамаларда жүзеге асыруды сипаттайды. Алайда, ең қызықтысы 18-тарау, мұнда автор өзінің желілік хаттамалармен жұмысы жайлы, көпжылдық тәжірибесімен бөліседі. Бұл тарау қызықты, сонымен бірге көп ақпарат алуға болады, оны хаттама құрастырушылардың барлығы оқығаны дұрыс.

Stevens, *TCP/IP Illustrated*, 1-том.

3-тен 10-тарауға дейін IP және онымен байланысты (ARP, RARP және ICMP) хаттамалар түсінікті тілмен сипатталып, мысалдармен толықтырылған.

Varghese, *Network Algorithmics*

Біз маршруттауыштар және басқа да желі элементтерінің бір-бірімен қалай әрекеттесетіндігі жайлы көп айттық. Бұл кітап басқаша – мұнда өте жоғары жылдамдықпен дестелерді жөнелтетін маршруттауыштардың нақты қалай жасалғандығы жайлы айтылған. Кітапты маршруттауыштардың ішкі құрылымын және онымен байланысты сұрақтарды білу үшін оқыған дұрыс. Автор – іс жүзіндегі програмалық және аппараттық жабдықтамаларда жоғары жылдамдықты желілік элементтерді құрастыруда қолданылатын «айласы» бар алгоритмдердің танымал, беделді адамдарының бірі.

9.1.6. Транспорттық деңгей

Comer, *Internetworking with TCP/IP,* 1-том, 5-басылым.

Жоғарыда айтылғандай автор TCP/IP хаттамалары жайлы толық еңбек жазған. Кітаптың екінші бөлігі UDP және TCP хаттамаларына арналған.

Farrell and Cahill, *Delay- and Disruption-Tolerant Networking*

Бұл байланыстың қатаң шарттары жағдайында жұмыс істейтін «толық қанды емес желілер» құрылымы, хаттамалары және қосымшалары жайлы терең түсінік алу үшін оқуға тұрарлық кітап. Кітап авторлары IETF DTN Research Group құрамында DTN-нің дамуына қатысқан.

Stevens, *TCP/IP Illustrated,* 1-том.

Кітаптың 17-24-тарауларында TCP хаттамасы сипатталып, мысалдармен толықтырылған.

9.1.7. Қолданбалы деңгей

Berners-Lee et al., "The World Wide Web"

Дүниежүзілік өрмектің өткеніне, дамуына құрастырушылар және CERN бойынша әріптестері тұрғысынан көз жүгірту. Мақала Дүниежүзілік өрмек құрылымына, бірыңғай көрсеткіштерге (URL), HTTP хаттамасына және HTML тіліне, сонымен бірге оның келешегіне арналған. Басқа таратылған ақпараттық жүйелермен салыстырулар келтірілген.

Held, *A Practical Guide to Content Delivery Networks*, 2-басылым.

Бұл кітап CDN-нің іс жүзінде қалай жұмыс істейтіндігі жайлы мағлұмат береді. Сонымен бірге, мұнда CDN-нің жақсы жұмыс істейтін бөліктерінің құрастырылуына аса көңіл бөлінген.

Hunter et al., *Beginning XML*, 4-басылым.

HTML, XML және веб-қызметтер жайлы кітаптар көп. Бұл 100-парақтық кітапта сіз білгіңіз келген ақпараттың басым бөлігі жазылған. Мұнда HTML және XML қалай жазу керектігі ғана емес, сонымен бірге Ajax, SOAP және басқа да қолданыстағы тәсілдерді пайдаланып, XML құрастырып, оны басқаратын веб-қызметтер жайлы жазылған.

Krishnamurthy and Rexford, *Web Protocols and Practice*

Дүниежүзілік өрмектің барлық аспектілерін қамтыған және түсінікті тілмен жазылған кітапты табу өте қиын. Мұнда клиенттер, серверлер, прокси және кэштеу жайлы жазылған. Алайда, сіз мұнда трафик және оны Дүниежүзілік өрмекте өлшеуге, сонымен бірге Web-тің дамуы жайлы ағымдағы зерттеулерге арналған бөлімдер бар деп болжамаған боларсыз.

Simpson, *Video Over IP,* 2-басылым.

Автор кітабында бейнені Интернет және осы мақсатта құрастырылған жекеменшік желілердегідей, жалпы желі арқылы тасымалдау үшін IP-технологияны қалай қолдану керек екендігіне көз жүгірткен. Бұл кітаптың бейне саласындағы, желімен қызыққыш мамандарға арналғандығы қуантады.

Wittenburg, *Understanding Voice Over IP Technology*

Бұл кітапта IP-телефония жұмысы: IP хаттамаларды пайдаланып аудио деректерді тасымалдау және қызмет көрсету сапасынан (QoS) бастап, SIP және H.323 хаттамалар жұмысына дейін сипатталған. Материалдар егжей-тегжейлі және қабылдауға жеңіл блоктарға бөлініп келтірілген.

9.1.8. Желілердегі қауіпсіздік

Anderson, *Security Engineering,* 2-басылым.

Бұл кітап адамдардың қауіпсіздік тәсілдерін қалай қолданылатыны (немесе дұрыс қолданбайтындығы) жайлы жазылған. Мұнда, *Secrets and Lies* басылымына қарағанда, техникалық аспектілер көбірек қамтылған, бірақ *Network Security*-мен (төменде қараңыз) салыстырғанда аздау. Алдымен ақпаратты қорғаудың негізгі тәсілдері, содан кейін толығымен банк және атомдық энергетиканы басқару жүйесі, қауіпсіз баспа, биомертия, физикалық қорғаныс, электронды соғыстар, телекоммуникациядағы қорғаныс, электронды коммерция және авторлық құқықты қорғау тәрізді әртүрлі қосымшаларға арналған тараулар берілген.

Ferguson et al., *Cryptography Engineering*

Танымал шифрлау алгоритмдерінің қалай жұмыс істейтіндігіне арналған кітаптар өте көп. Бұл кітап криптографияны қалай қолдану керектігі жайлы – шифрлау алгоритмдері неліктен осылай құрастырылған және оларды сіздің қауіпсіздік мақсатыңызды қанағаттандыратын жүйеге қалай біріктіруі керектігі сипатталған. Бұл шағын кітап, оны криптографияға тәуелді желі құрастырушылардың барлығы оқығаны дұрыс.

Fridrich, *Steganography in Digital Media*

Стеганография ежелгі Грекияда қолданылған болатын, онда бос таяқшалардың балауызы алынып, құпия мәлімдеме жазылып, таяқша қайтадан балауызбен жабылған. Қазіргі кезде Интернеттегі бейне, аудио және басқа да контенттер көмегімен, құпия мәлімдемелер үшін түрлі арналар құрастыруға болады. Бұл кітапта бейнедегі құпия мәлімдемені анықтау және жасырудың қазіргі заманғы тәсілдері талқыланады.

Kaufman et al., *Network Security,* 2-басылым.

Егер сізді желі қауіпсіздігін қамтамасыз ететін хаттамалар және алгоритмдердің техникалық аспектілері жайлы қосымша ақпарат қызықтыратын болса, онда осы сенім артылған және өткір тілді кітапты оқыңыз. Мұнда құпия және ашық кілтті алгоритмдер және хаттамалар, мәлімдемені хэштеу, сәйкестендіру, Kerberos, PKI, IPsec, SSL/TLS және электронды пошта қауіпсіздігін қамтамасыз ету алгоритмдері мен хаттамалары толық сипатталған.

Барлық тараулар мысалдармен толықтырылған. 26-тарау ақпаратты қорғау тақырыбының фольклорына арналған, бұл – нағыз жауһар дүние. Қауіпсіздікті қамтамасыз ету ісінде барлығы маңызды. Егер сіз, шын мәнінде пайдалы қорғаныс жүйесін құрастырғыңыз келсе, онда бұл тарау сізге өмірден алынған көптеген қызықты мысалдарды айтып береді.

Schneier, *Secrets and Lies*

Cryptography Engineering басылымын бастан-аяқ оқып, сіз криптография алгоритмдерінің білгірі боласыз. Егер сіз бұдан кейін *Secrets and Lies* басылымында мұқият оқып шықсаңыз, онда сіз бір алгоритммен істің аяқталмайтындығын түсінесіз. Көптеген қорғаныс жүйелерінің осалдығы – нашар алгоритм немесе тым қысқа кілтпен емес, осы жүйені қоршаған орта кемістігімен байланысты екенін білесіз. Бұл кітап қорғаныс жүйесі мәселелерін кеңінен қарастыратын, техникалық емес басылым болып саналады.

Skoudis and Liston, *Counter Hack Reloaded*, 2-басылым.

Сындырушыны қалай тоқтатуға болады? Дәл сындырушы тәрізді ойлау керек! Бұл кітапта желі сындырушы көзқарасы тұрғысынан сипаттаған. Автор ақпаратты қорғау бүкіл желілік жүйенің бір функциясы болуы тиіс дейді, сонымен бірге ол жеке, арнайы технология ретінде қолданыстағы желілерге біріктірілмеуі керек. Кең таралған шабуылдардың барлығы, тіпті, тұтынушының электронды қауіпсіздік жүйесін білмейтіндігіне негізделген «әлеуметтік инженерия» шабуылы қоса қарастырылған.

9.2. ӘДЕБИЕТТЕРДІҢ ӘЛІППЕЛІК ТІЗІМІ

ABRAMSON, N.: "Internet Access Using VSATs", *IEEE Commun. Magazine,* vol. 38, pp. 68, July 2000.

AHMADI, S.: "An Overview of Next-Generation Mobile WiMAX Technology", *IEEE Commun. Magazine,* vol. 47, pp. 84-88, June 2009.

ALLMAN, M., and PAXSON, V.: "On Estimating End-to-End Network Path Properties", *Proc. SIGCOMM '99 Conf.,* ACM, pp. 263-274, 1999.

ANDERSON, C.: The Long Tail: Why the Future of Business is Selling Less of More, *rev. upd. ed., New York: Hyperion, 2008a.*

ANDERSON, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems, *2nd ed., New York: John Wiley & Sons, 2008b.*

ANDERSON, R.J.: "Free Speech Online and Offline", *IEEE Computer,* vol. 25, pp. 28-30, June 2002.

ANDERSON, R.J.: "The Eternity Service", *Proc. Pragocrypt Conf.,* CTU Publishing House, pp. 242-252, 1996.

ANDREWS, J., GHOSH, A., and MUHAMED, R.: Fundamentals of WiMAX: Understanding Broadband Wireless Networking, *Upper Saddle River, NJ: Pearson Education, 2007.*

ASTELY, D., DAHLMAN, E., FURUSKAR, A., JADING, Y., LINDSTROM, M., and PARKVALL, S.: "**LTE: The Evolution of Mobile Broadband", *IEEE Commun. Magazine,* vol. 47, pp. 44-51, Apr. 2009.**

BALLARDIE, T., FRANCIS, P., and CROWCROFT, J.: "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.,* ACM, pp. 85-95, 1993.

BARAN, P.: "On Distributed Communications: I. Introduction to Distributed Communication Networks", *Memorandum RM-420-PR,* Rand Corporation, Aug. 1964.

BELLAMY, J.: *Digital Telephony,* 3rd ed., New York: John Wiley & Sons, 2000.

BELLMAN, R.E.: *Dynamic Programming,* Princeton, NJ: Princeton University Press, 1957.

BELLOVIN, S.: "The Security Flag in the IPv4 Header", RFC 3514, Apr. 2003.

BELSNES, D.: "Flow Control in the Packet Switching Networks", *Communications Networks,* Uxbridge, England: Online, pp. 349-361, 1975.

BENNET, C.H., and BRASSARD, G.: "Quantum Cryptography: Public Key Distribution and Coin Tossing," *Int'l Conf. on Computer Systems and Signal Processing,* pp. 175-179, 1984.

BERESFORD, A., and STAJANO, F.: "Location Privacy in Pervasive Computing", *IEEE Pervasive Computing,* vol. 2, pp. 46-55, Jan. 2003.

BERGHEL, H.L.: "Cyber Privacy in the New Millennium", *IEEE Computer,* vol. 34, pp. 132-134, Jan. 2001.

BERNERS-LEE, T., CAILLIAU, A., LOUTONEN, A., NIELSEN, H.F., and SECRET, A.: "**The World Wide Web", *Commun. of the ACM,* vol. 37, pp. 76-82, Aug. 1994.**

BERTSEKAS, D., and GALLAGER, R.: *Data Networks,* 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1992.

BHATTI, S.N., and CROWCROFT, J.: "QoS Sensitive Flows: Issues in IP Packet Handling", *IEEE Internet Computing,* vol. 4, pp. 48-57, July-Aug. 2000.

BIHAM, E., and SHAMIR, A.: "Differential Fault Analysis of Secret Key Cryptosystems", *Proc. 17th Ann. Int'l Cryptology Conf.,* Berlin: Springer-Verlag LNCS 1294, pp. 513-525, 1997.

BIRD, R., GOPAL, I., HERZBERG, A., JANSON, P.A., KUTTEN, S., MOLVA, R., and YUNG, M.: "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas in Commun.,* vol. 11, pp. 679-693, June 1993.

BIRRELL, A.D., and NELSON, B.J.: "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems,* vol. 2, pp. 39-59, Feb. 1984.

BIRYUKOV, A., SHAMIR, A., and WAGNER, D.: "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption,* Berlin: Springer- Verlag LNCS 1978, pp. 1-8, 2000.

BLAZE, M., and BELLOVIN, S.: "Tapping on My Network Door", *Commun. of the ACM,* vol. 43, p. 136, Oct. 2000.

BOGGS, D., MOGUL, J., and KENT, C.: "Measured Capacity of an Ethernet: Myths and Reality", *Proc. SIGCOMM '88 Conf.,* ACM, pp. 222-234, 1988.

BORISOV, N., GOLDBERG, I., and WAGNER, D.: "Intercepting Mobile Communications: The Insecurity of 802.11", *Seventh Int'l Conf. on Mobile Computing and Networking,* ACM, pp. 180-188, 2001.

BRADEN, R.: "Requirements for Internet Hosts—Communication Layers", RFC 1122, Oct. 1989.

BRADEN, R., BORMAN, D., and PARTRIDGE, C.: "Computing the Internet Checksum", RFC 1071, Sept. 1988.

BRANDENBURG, K.: "MP3 and AAC Explained", *Proc. 17th Intl. Conf.: High-Quality Audio Coding,* Audio Engineering Society, pp. 99-110, Aug. 1999.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C., MALER, E., YERGEAU,

F., and COWAN, J.: "Extensible Markup Language (XML) 1.1 (Second Edition)", W3C Recommendation, Sept. 2006.

BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G., and SHENKER, S.: "Web Caching and Zipf-like Distributions: Evidence and Implications", *Proc. INFOCOM Conf.,* IEEE, pp. 126-134, 1999.

BURLEIGH, S., HOOKE, A., TORGERSON, L., FALL, K., CERF, V., DURST, B., SCOTT, K., and WEISS, H.: "Delay-Tolerant Networking: An Approach to Interplanetary Internet", *IEEE Commun. Magazine,* vol. 41, pp. 128-136, June 2003.

BURNETT, S., and PAINE, S.: RSA Security's Official Guide to Cryptography, *Berkeley, CA: Osborne/McGraw-Hill, 2001.*

BUSH, V.: "As We May Think", *Atlantic Monthly,* vol. 176, pp. 101-108, July 1945.

CAPETANAKIS, J.I.: "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory,* vol. IT-5, pp. 505-515, Sept. 1979.

CASTAGNOLI, G., BRAUER, S., and HERRMANN, M.: "Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits", *IEEE Trans. on Commun.,* vol. 41, pp. 883-892, June 1993.

CERF, V., and KAHN, R.: "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun.,* vol. COM-2, pp. 637-648, May 1974.

CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W., WALLACH, D., BURROWS, M., CHANDRA, T., FIKES, A., and GRUBER, R.: **"Bigtable: A Distributed Storage System for Structured Data", *Proc. OSDI 2006 Symp.,* USENIX, pp. 15-29, 2006.**

CHASE, J.S., GALLATIN, A.J., and YOCUM, K.G.: "End System Optimizations for HighSpeed TCP", *IEEE Commun. Magazine,* vol. 39, pp. 68-75, Apr. 2001.

CHEN, S., and NAHRSTEDT, K.: "An Overview of QoS Routing for Next-Generation Networks", *IEEE Network Magazine,* vol. 12, pp. 64-69, Nov./Dec. 1998.

CHIU, D., and JAIN, R.: "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Comput. Netw. ISDN Syst.,* vol. 17, pp. 1-4, June 1989.

CISCO: "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014", Cisco Systems Inc., June 2010.

CLARK, D.D.: "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.,* ACM, pp. 106-114,1988.

CLARK, D.D.: "Window and Acknowledgement Strategy in TCP", RFC 813, July 1982.

CLARK, D.D., JACOBSON, V., ROMKEY, J., and SALWEN, H.: "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine,* vol. 27, pp. 23-29, June 1989.

CLARK, D.D., SHENKER, S., and ZHANG, L.: "Supporting Real-Time

Applications in an Integrated Services Packet Network", *Proc. SIGCOMM '92 Conf.,* ACm, pp. 14-26, 1992.

CLARKE, A.C.: "Extra-Terrestrial Relays", *Wireless World,* 1945.

CLARKE, I., MILLER, S.G., HONG, T.W., SANDBERG, O., and WILEY, B.: "Protecting Free Expression Online with Freenet", *IEEE Internet Computing,* vol. 6, pp. 40-49, Jan.-Feb. 2002.

COHEN, B.: "Incentives Build Robustness in BitTorrent", *Proc. First Workshop on Economics of Peer-to-Peer Systems,* June 2003.

COMER, D.E.: *The Internet Book,* 4th ed., Englewood Cliffs, NJ: Prentice Hall, 2007.

COMER, D.E.: *Internetworking with TCP/IP,* vol. 1, 5th ed., Englewood Cliffs, NJ: Prentice Hall, 2005.

CRAVER, S.A., WU, M., LIU, B., STUBBLEFIELD, A., SWARTZLANDER, B., WALLACH, D.W., DEAN, D., and FELTEN, E.W.: **"Reading Between the Lines: Lessons from the SDMI Challenge",** ***Proc. 10th USENIX Security Symp,*** **USENIX, 2001.**

CROVELLA, M., and KRISHNAMURTHY, B.: ***Internet Measurement,*** **New York: John Wiley & Sons, 2006.**

DAEMEN, J., and RIJMEN, V.: *The Design ofRijndael,* Berlin: Springer-Verlag, 2002.

DALAL, Y., and METCLFE, R.: "Reverse Path Forwarding of Broadcast Packets", *Commun. of the ACM,* vol. 21, pp. 1040-1048, Dec. 1978.

DAVIE, B., and FARREL, A.: *MPLS: Next Steps,* San Francisco: Morgan Kaufmann, 2008.

DAVIE, B., and REKHTER, Y.: *MPLS Technology and Applications,* San Francisco: Morgan Kaufmann, 2000.

DAVIES, J.: *Understanding IPv6,* 2nd ed., Redmond, WA: Microsoft Press, 2008.

DAY, J.D.: "The (Un) Revised OSI Reference Model", *Computer Commun. Rev.,* vol. 25, pp. 39-55, Oct. 1995.

DAY, J.D., and ZIMMERMANN, H.: "The OSI Reference Model", *Proc. of the IEEE,* vol. 71, pp. 1334-1340, Dec. 1983.

DECANDIA, G., HASTORIN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PIL- CHIN, A., SIVASUBRAMANIAN, S., VOSSHALL, P., and VOGELS, W.: **"Dynamo: Amazon's Highly Available Key-value Store",** ***Proc. 19th Symp. on Operating Systems Prin.,*** **ACM, pp. 205-220, Dec. 2007.**

DEERING, S.E.: "SIP: Simple Internet Protocol", *IEEE Network Magazine,* vol. 7, pp. 16-28, May/June 1993.

DEERING, S., and CHERITON, D.: "Multicast Routing in Datagram Networks and Extended LANs", *ACM Trans. on Computer Systems,* vol. 8, pp. 85-110, May 1990.

DEMERS, A., KESHAV, S., and SHENKER, S.: "Analysis and Simulation

of a Fair Queueing Algorithm", *Internetwork: Research and Experience,* vol. 1, pp. 3-26, Sept. 1990.

DENNING, D.E., and SACCO, G.M.: "Timestamps in Key Distribution Protocols", *Commun. of the ACM,* vol. 24, pp. 533-536, Aug. 1981.

DEVARAPALLI, V., WAKIKAWA, R., PETRESCU, A., and THUBERT, P.: "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, Jan. 2005.

DIFFIE, W., and HELLMAN, M.E.: "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *IEEE Computer,* vol. 10, pp. 74-84, June 1977.

DIFFIE, W., and HELLMAN, M.E.: "New Directions in Cryptography", *IEEE Trans. on Information Theory,* vol. IT-2, pp. 644-654, Nov. 1976.

DIJKSTRA, E.W.: "A Note on Two Problems in Connexion with Graphs", *Numer. Math.,* vol. 1, pp. 269-271, Oct. 1959.

DILLEY, J., MAGGS, B., PARIKH, J., PROKOP, H., SITARAMAN, R., and WHEIL, B.: "Globally Distributed Content Delivery", *IEEE Internet Computing,* vol. 6, pp. 50-58, 2002.

DINGLEDINE, R., MATHEWSON, N., SYVERSON, P.: "Tor: The Second-Generation Onion Router", *Proc. 13th USENIX Security Symp.,* USENIX, pp. 303-320, Aug. 2004.

DONAHOO, M., and CALVERT, K.: *TCP/IP Sockets in C,* 2nd ed., San Francisco: Morgan Kaufmann, 2009.

DONAHOO, M., and CALVERT, K.: *TCP/IP Sockets in Java,* 2nd ed., San Francisco: Morgan Kaufmann, 2008.

DONALDSON, G., and JONES, D.: "Cable Television Broadband Network Architectures", *IEEE Commun. Magazine,* vol. 39, pp. 122-126, June 2001.

DORFMAN, R.: "Detection of Defective Members of a Large Population", *Annals Math. Statistics,* vol. 14, pp. 436-440, 1943.

DUTCHER, B.: *The NAT Handbook,* New York: John Wiley & Sons, 2001.

DUTTA-ROY, A.: "An Overview of Cable Modem Technology and Market Perspectives", *IEEE Commun. Magazine,* vol. 39, pp. 81-88, June 2001.

EDELMAN, B., OSTROVSKY, M., and SCHWARZ, M.: "Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords", *American Economic Review,* vol. 97, pp. 242-259, Mar. 2007.

EL GAMAL, T.: "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. on Information Theory,* vol. IT-1, pp. 469-472, July 1985.

EPCGLOBAL: EPC Radio-Frequency Identity Protocols Class- Generation-UHF RFID Protocol for Communication at 860-MHz to 960-MHz Version 1.2.0, *Brussels: EPCglobal Inc., Oct. 2008.*

FALL, K.: "A Delay-Tolerant Network Architecture for Challenged Internets", *Proc. SIGCOMM 2003 Conf.,* ACM, pp. 27-34, Aug. 2003.

FALOUTSOS, M., FALOUTSOS, P., and FALOUTSOS, C.: "On Power-Law Relationships of the Internet Topology", *Proc. SIGCOMM '99 Conf.,* ACM, pp. 251-262, 1999.

FARRELL, S., and CAHILL, V.: Delay- and Disruption-Tolerant Networking, *London: Artech House, 2007.*

FELLOWS, D., and JONES, D.: "DOCSIS Cable Modem Technology", *IEEE Commun. Magazine,* vol. 39, pp. 202-209, Mar. 2001.

FENNER, B., HANDLEY, M., HOLBROOK, H., and KOUVELAS, I.: "Protocol Independent Multicast-Sparse Mode (PIM-SM)", RFC 4601, Aug. 2006.

FERGUSON, N., SCHNEIER, B., and KOHNO, T.: Cryptography Engineering: Design Principles and Practical Applications, *New York: John Wiley & Sons, 2010.*

FLANAGAN, D.: *JavaScript: The Definitive Guide,* 6th ed., Sebastopol, CA: O'Reilly, 2010.

FLETCHER, J.: "An Arithmetic Checksum for Serial Transmissions", *IEEE Trans. on Commun.,* vol. COM-0, pp. 247-252, Jan. 1982.

FLOYD, S., HANDLEY, M., PADHYE, J., and WIDMER, J.: "Equation-Based Congestion Control for Unicast Applications", *Proc. SIGCOMM 2000 Conf.,* ACM, pp. 43-56, Aug. 2000.

FLOYD, S., and JACOBSON, V.: "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking,* vol. 1, pp. 397-413, Aug. 1993.

FLUHRER, S., MANTIN, I., and SHAMIR, A.: "Weakness in the Key Scheduling Algorithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography,* Berlin: Springer-Verlag LNCS 2259, pp. 1-24, 2001.

FORD, B.: "Structured Streams: A New Transport Abstraction", *Proc. SIGCOMM 2007 Conf.,* ACM, pp. 361-372, 2007.

FORD, L.R., Jr., and FULKERSON, D.R.: *Flows in Networks,* Princeton, NJ: Princeton University Press, 1962.

FORD, W., and BAUM, M.S.: *Secure Electronic Commerce,* Upper Saddle River, NJ: Prentice Hall, 2000.

FORNEY, G.D.: "The Viterbi Algorithm", *Proc. of the IEEE,* vol. 61, pp. 268-278, Mar. 1973.

FOULI, K., and MAIER, M.: "The Road to Carrier-Grade Ethernet", *IEEE Commun. Magazine,* vol. 47, pp. S30-S38, Mar. 2009.

FOX, A., GRIBBLE, S., BREWER, E., and AMIR, E.: "Adapting to Network and Client Variability via On-Demand Dynamic Distillation", *SIGOPS Oper. Syst. Rev.,* vol. 30, pp. 160-170, Dec. 1996.

FRANCIS, P.: "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine,* vol. 7, pp. 30-37, May/June 1993.

FRASER, A.G.: "Towards a Universal Data Transport System", *IEEE J. on Selected Areas in Commun.,* vol. 5, pp. 803-816, Nov. 1983.

FRIDRICH, J.: Steganography in Digital Media: Principles, Algorithms, and Applications, *Cambridge: Cambridge University Press, 2009.*

FULLER, V., and LI, T.: "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, Aug. 2006.

GALLAGHER, R.G.: "A Minimum Delay Routing Algorithm Using Distributed Computation", *IEEE Trans. on Commun.,* vol. COM-5, pp. 73-85, Jan. 1977.

GALLAGHER, R.G.: "Low-Density Parity Check Codes", *IRE Trans. on Information Theory,* vol. 8, pp. 21-28, Jan. 1962.

GARFINKEL, S., with SPAFFORD, G.: Web Security, Privacy, and Commerce, *Sebastopol, CA: O'Reilly, 2002.*

GAST, M.: 802.11 Wireless Networks: The Definitive Guide, *2nd ed., Sebastopol, CA: O'Reilly, 2005.*

GERSHENFELD, N., and KRIKORIAN, R., and COHEN, D.: "The Internet of Things", *Scientific American,* vol. 291, pp. 76-81, Oct. 2004.

GILDER, G.: "Metcalfe's Law and Legacy", *Forbes ASAP,* Sepy. 13, 1993.

GOODE, B.: "Voice over Internet Protocol", *Proc. of the IEEE,* vol. 90, pp. 1495-1517, Sept. 2002.

GORALSKI, W.J.: *SONET,* 2nd ed., New York: McGraw-Hill, 2002.

GRAYSON, M., SHATZKAMER, K., and WAINNER, S.: ***IP Design for Mobile Networks,*** **Indianapolis, IN: Cisco Press, 2009.**

GROBE, K., and ELBERS, J.: "PON in Adolescence: From TDMA to WDM-PON", *IEEE Commun. Magazine,* vol. 46, pp. 26-34, Jan. 2008.

GROSS, G., KAYCEE, M., LIN, A., MALIS, A., and STEPHENS, J.: **' The PPP Over AAL5," RFC 2364, July 1998.**

HA, S., RHEE, I., and LISONG, X.: "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *SIGOPS Oper. Syst. Rev.,* vol. 42, pp. 64-74, June 2008.

HAFNER, K., and LYON, M.: *Where Wizards Stay Up Late,* New York: Simon & Schuster,

1998.

HALPERIN, D., HEYDT-BENJAMIN, T., RANSFORD, B., CLARK, S., DEFEND, B., MORGAN, W., FU, K., KOHNO, T., and MAISEL, W.: **"Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses",** *IEEE Symp. on Security and Privacy,* pp. 129-142, May 2008.

HALPERIN, D., HU, W., SHETH, A., and WETHERALL, D.: "802.11 with Multiple Antennas for Dummies", *Computer Commun. Rev.,* vol. 40, pp. 19-25, Jan. 2010.

HAMMING, R.W.: "Error Detecting and Error Correcting Codes", *Bell System Tech. J.,* vol. 29, pp. 147-160, Apr. 1950.

HARTE, L., KELLOGG, S., DREHER, R., and SCHAFFNIT, T.: ***The Comprehensive Guide to Wireless Technology,*** **Fuquay-Varina, NC: APDG Publishing, 2000.**

HAWLEY, G.T.: "Historical Perspectives on the U.S. Telephone Loop", *IEEE Commun. Magazine,* vol. 29, pp. 24-28, Mar. 1991.

HECHT, J.: *Understanding Fiber Optics,* Upper Saddle River, NJ: Prentice Hall, 2005.

HELD, G.: A Practical Guide to Content Delivery Networks, *2nd ed., Boca Raton, FL: CRC Press, 2010.*

HEUSSE, M., ROUSSEAU, F., BERGER-SABBATEL, G., DUDA, A.: "Performance Anomaly of 802.11b", *Proc. INFOCOM Conf.,* IEEE, pp. 836-843, 2003.

HIERTZ, G., DENTENEER, D., STIBOR, L., ZANG, Y., COSTA, X., and WALKE, B.: **"The** IEEE 802.11 Universe", *IEEE Commun. Magazine,* vol. 48, pp. 62-70, Jan. 2010.

HOE, J.: "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", *Proc. SIGCOMM '96 Conf.,* ACM, pp. 270-280, 1996.

HU, Y., and LI, V.O.K.: "Satellite-Based Internet: A Tutorial", *IEEE Commun. Magazine,* vol. 30, pp. 154-162, Mar. 2001.

HUITEMA, C.: *Routing in the Internet,* 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1999.

HULL, B., BYCHKOVSKY, V., CHEN, K., GORACZKO, M., MIU, A., SHIH, E., ZHANG, Y., BALAKRISHNAN, H., and MADDEN, S.: **"CarTel: A Distributed Mobile Sensor Computing System",** ***Proc. Sensys 2006 Conf.,*** **ACM, pp. 125-138, Nov. 2006.**

HUNTER, D., RAFTER, J., FAWCETT, J., VAN DER LIST, E., AYERS, D., DUCKETT, J., WATT, A., and MCKINNON, L.: ***Beginning XML,*** **4th ed., New Jersey: Wrox, 2007.**

IRMER, T.: "Shaping Future Telecommunications: The Challenge of Global Standardization", *IEEE Commun. Magazine,* vol. 32, pp. 20-28, Jan. 1994.

ITU (INTERNATIONAL TELECOMMUNICATION UNION): ITU Internet Reports 2005: The Internet of Things, *Geneva: ITU, Nov. 2005.*

ITU (INTERNATIONAL TELECOMMUNICATION UNION): Measuring the Information Society: The ICTDevelopment Index, *Geneva: ITU, Mar. 2009.*

JACOBSON, V.: "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, Feb. 1990.

JACOBSON, V.: "Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf.,* ACM, pp. 314-329, 1988.

JAIN, R., and ROUTHIER, S.: "Packet Trains—Measurements and a New Model for Computer Network Traffic", *IEEE J. on Selected Areas in Commun.,* vol. 6, pp. 986-995, Sept. 1986.

JAKOBSSON, M., and WETZEL, S.: "Security Weaknesses in Bluetooth", *Topics in Cryptology: CT-RSA 2001,* Berlin: Springer-Verlag LNCS 2020, pp. 176-191, 2001.

JOEL, A.: "Telecommunications and the IEEE Communications Society", *IEEE Commun. Magazine,* 50th Anniversary Issue, pp. 6-14 and 162-167, May 2002.

JOHNSON, D., PERKINS, C., and ARKKO, J.: "Mobility Support in IPv6", RFC 3775, June 2004.

JOHNSON, D.B., MALTZ, D., and BROCH, J.: "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks", *Ad Hoc Networking,* Boston: Addison-Wesley, pp. 139-172, 2001.

JUANG, P., OKI, H., WANG, Y., MARTONOSI, M., PEH, L., and RUBENSTEIN, D.: "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet", *SIGOPS Oper. Syst. Rev.,* vol. 36, pp. 96-107, Oct. 2002.

KAHN, D.: *The Codebreakers,* 2nd ed., New York: Macmillan, 1995.

KAMOUN, F., and KLEINROCK, L.: "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks", *Computer Networks,* vol. 3, pp. 337-353, Nov. 1979.

KARN, P.: "MACA—A New Channel Access Protocol for Packet Radio", *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.,* pp. 134-140, 1990.

KARN, P., and PARTRIDGE, C.: "Improving Round-Trip Estimates in Reliable Transport Protocols", *Proc. SIGCOMM '87 Conf.,* ACM, pp. 2-7, 1987.

KARP, B., and KUNG, H.T.: "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", *Proc. MOBICOM 2000 Conf.,* ACM, pp. 243-254, 2000.

KASIM, A.: *Delivering Carrier Ethernet,* New York: McGraw-Hill, 2007.

KATABI, D., HANDLEY, M., and ROHRS, C.: "Internet Congestion Control for Future High Bandwidth-Delay Product Environments", *Proc. SIGCOMM 2002 Conf.,* ACM, pp. 89-102, 2002.

KATZ, D., and FORD, P.S.: "TUBA: Replacing IP with CLNP", *IEEE Network Magazine,* vol. 7, pp. 38-47, May/June 1993.

KAUFMAN, C., PERLMAN, R., and SPECINER, M.: *Network Security,* 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 2002.

KENT, C., and MOGUL, J.: "Fragmentation Considered Harmful", *Proc. SIGCOMM '87 Conf.,* ACM, pp. 390-401, 1987.

KERCKHOFF, A.: "La Cryptographie Militaire", *J. des Sciences Militaires,* vol. 9, pp. 5-38, Jan. 1883 and pp. 161-191, Feb. 1883.

KHANNA, A., and ZINKY, J.: "The Revised ARPANET Routing Metric", *Proc. SIGCOMM '89 Conf.,* ACM, pp. 45-56, 1989.

KIPNIS, J.: "Beating the System: Abuses of the Standards Adoption Process", *IEEE Commun. Magazine,* vol. 38, pp. 102-105, July 2000.

KLEINROCK, L.: "Power and Other Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications", *Proc. Intl. Conf. on Commun.,* pp. 43.1.1-43.1.10, June 1979.

KLEINROCK, L., and TOBAGI, F.: "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels", *Proc. Nat. Computer Conf.,* pp. 187-201, 1975.

KOHLER, E., HANDLEY, H., and FLOYD, S.: "Designing DCCP: Congestion Control without Reliability", *Proc. SIGCOMM 2006 Conf.,* ACM, pp. 27-38, 2006.

KOODLI, R., and PERKINS, C.E.: *Mobile Inter-networking with IPv6,* New York: John Wiley & Sons, 2007.

KOOPMAN, P.: "32-Bit Cyclic Redundancy Codes for Internet Applications", *Proc. Intl. Conf. on Dependable Systems and Networks.,* IEEE, pp. 459-472, 2002.

KRISHNAMURTHY, B., and REXFORD, J.: ***Web Protocols and Practice,*** **Boston: Addison-Wesley, 2001.**

KUMAR, S., PAAR, C., PELZL, J., PFEIFFER, G., and SCHIMMLER, M.: "Breaking Ciphers with COPACOBANA: A Cost-Optimized Parallel Code Breaker", *Proc. 8th Cryptographic Hardware and Embedded Systems Wksp.,* IACR, pp. 101-118, Oct. 2006.

LABOVITZ, C., AHUJA, A., BOSE, A., and JAHANIAN, F.: **"Delayed Internet Routing** Convergence", *IEEE/ACM Trans. on Networking,* vol. 9, pp. 293-306, June 2001.

LAM, C.K.M., and TAN, B.C.Y.: "The Internet Is Changing the Music Industry", *Commun. of the ACM,* vol. 44, pp. 62-66, Aug. 2001.

LAOUTARIS, N., SMARAGDAKIS, G., RODRIGUEZ, P., and SUNDARAM, R.: "Delay Tolerant Bulk Data Transfers on the Internet", *Proc. SIGMETRICS 2009 Conf.,* ACM, pp. 229-238, June 2009.

LARMO, A., LINDSTROM, M., MEYER, M., PELLETIER, G., TORSNER, J., and WIEMANN, H.: **"The LTE Link-Layer Design",** ***IEEE Commun. Magazine,*** **vol. 47, pp. 52-59, Apr. 2009.**

LEE, J.S., and MILLER, L.E.: CDMA Systems Engineering Handbook, *London: Artech House, 1998.*

LELAND, W., TAQQU, M., WILLINGER, W., and WILSON, D.: "On the Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Trans. on Networking,* vol. 2, pp. 1-15, Feb. 1994.

LEMON, J.: "Resisting SYN Flood DOS Attacks with a SYN Cache", *Proc. BSDCon Conf.,* USENIX, pp. 88-98, 2002.

LEVY, S.: "Crypto Rebels", *Wired,* pp. 54-61, May/June 1993.

LEWIS, M.: Comparing, Designing, and Deploying VPNs, *Indianapolis, IN: Cisco Press, 2006.*

LI, M., AGRAWAL, D., GANESAN, D., and VENKATARAMANI, A.: "Block-Switched Networks: A New Paradigm for Wireless Transport", *Proc. NSDI 2009 Conf.,* USENIX, pp. 423-436, 2009.

LIN, S., and COSTELLO, D.: *Error Control Coding,* 2nd ed., Upper Saddle River, NJ: Pearson Education, 2004.

LUBACZ, J., MAZURCZYK, W., and SZCZYPIORSKI, K.: **"Vice over IP",** ***IEEE Spectrum,*** **pp. 42-47, Feb. 2010.**

MACEDONIA, M.R.: "Distributed File Sharing", *IEEE Computer,* vol. 33, pp. 99-101, 2000.

MADHAVAN, J., KO, D., LOT, L., GANGPATHY, V., RASMUSSEN, A., and HALEVY, A.: "Google's Deep Web Crawl", *Proc. VLDB 2008 Conf.,* VLDB Endowment, pp. 1241-1252, 2008.

MAHAJAN, R., RODRIG, M., WETHERALL, D., and ZAHORJAN, J.: "Analyzing the MAC-Level Behavior of Wireless Networks in the Wild", *Proc. SIGCOMM 2006 Conf.,* ACM, pp. 75-86, 2006.

MALIS, A., and SIMPSON, W.: "PPP over SONET/SDH", RFC 2615, June 1999.

MASSEY, J.L.: "Shift-Register Synthesis and BCH Decoding", *IEEE Trans. on Information Theory,* vol. IT-5, pp. 122-127, Jan. 1969.

MATSUI, M.: "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology— Eurocrypt 1993 Proceedings,* Berlin: Springer-Verlag LNCS 765, pp. 386-397, 1994.

MAUFER, T.A.: *IP Fundamentals,* Upper Saddle River, NJ: Prentice Hall, 1999.

MAYMOUNKOV, P., and MAZIERES, D.: "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric", *Proc. First Intl. Wksp. on Peer-to-Peer Systems,* Berlin: Springer-Verlag LNCS 2429, pp. 53-65, 2002.

MAZIERES, D., and KAASHOEK, M.F.: "The Design, Implementation, and Operation of an Email Pseudonym Server", *Proc. Fifth Conf. on Computer and Commun. Security,* ACM, pp. 27-36, 1998.

MCAFEE LABS: McAfee Threat Reports: First Quarter 2010, *McAfee Inc., 2010.*

MENEZES, A.J., and VANSTONE, S.A.: "Elliptic Curve Cryptosystems and Their Implementation", *Journal of Cryptology,* vol. 6, pp. 209-224, 1993.

MERKLE, R.C., and HELLMAN, M.: "Hiding and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Information Theory,* vol. IT-4, pp. 525-530, Sept. 1978.

METCALFE, R.M.: "Computer/Network Interface Design: Lessons from Arpanet and Ethernet", *IEEE J. on Selected Areas in Commun.,* vol. 11, pp. 173-179, Feb. 1993.

METCALFE, R.M., and BOGGS, D.R.: "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM,* vol. 19, pp. 395-404, July 1976.

METZ, C: "Interconnecting ISP Networks", *IEEE Internet Computing,* vol. 5, pp. 74-80, Mar.-Apr. 2001.

MISHRA, P.P., KANAKIA, H., and TRIPATHI, S.: "On Hop by Hop Rate-Based Congestion Control", *IEEE/ACM Trans. on Networking,* vol. 4, pp. 224-239, Apr. 1996.

MOGUL, J.C.: "IP Network Performance", in *Internet System Handbook,* D.C. Lynch and M.Y. Rose (eds.), Boston: Addison-Wesley, pp. 575-575,1993.

MOGUL, J., and DEERING, S.: "Path MTU Discovery", RFC 1191, Nov. 1990.

MOGUL, J., and MINSHALL, G.: "Rethinking the Nagle Algorithm", *Comput. Commun. Rev.,* vol. 31, pp. 6-20, Jan. 2001.

MOY, J.: "Multicast Routing Extensions for OSPF", *Commun. of the ACM,* vol. 37, pp. 66, Aug. 1994.

MULLINS, J.: "Making Unbreakable Code", *IEEE Spectrum,* pp. 40-45, May 2002.

NAGLE, J.: "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun.,* vol. COM-5, pp. 435-438, Apr. 1987.

NAGLE, J.: "Congestion Control in TCP/IP Internetworks", *Computer Commun. Rev.,* vol. 14, pp. 11-17, Oct. 1984.

NAUGHTON, J.: *A Brief History of the Future,* Woodstock, NY: Overlook Press, 2000.

NEEDHAM, R.M., and SCHROEDER, M.D.: "Using Encryption for Authentication in Large Networks of Computers", *Commun. of the ACM,* vol. 21, pp. 993-999, Dec. 1978.

NEEDHAM, R.M., and SCHROEDER, M.D.: "Authentication Revisited", *Operating Systems Rev.,* vol. 21, p. 7, Jan. 1987.

NELAKUDITI, S., and ZHANG, Z.-L.: "A Localized Adaptive Proportioning Approach to QoS Routing", *IEEE Commun. Magazine* vol. 40, pp. 66-71, June 2002.

NEUMAN, C., and TS'O, T.: "Kerberos: An Authentication Service for Computer Networks", *IEEE Commun. Mag.,* vol. 32, pp. 33-38, Sept. 1994.

NICHOLS, R.K., and LEKKAS, P.C.: ***Wireless Security,*** **New York: McGraw-Hill, 2002.**

NIST: "Secure Hash Algorithm", U.S. Government Federal Information Processing Standard 180, 1993.

NONNENMACHER, J., BIERSACK, E., and TOWSLEY, D.: "Parity-Based Loss Recovery for Reliable Multicast Transmission", *Proc. SIGCOMM '97 Conf.,* ACM, pp. 289-300, 1997.

NUCCI, A., and PAPAGIANNAKI, D.: Design, Measurement and Management of Large- Scale IP Networks, *Cambridge: Cambridge University Press, 2008.*

NUGENT, R., MUNAKANA, R., CHIN, A., COELHO, R., and PUIG-SUARI, J.: **"The** CubeSat: The PicoSatellite Standard for Research and Education", *Proc. SPACE 2008 Conf.,* AIAA, 2008.

ORAN, D.: "OSIIS-IS Intra-domain Routing Protocol", RFC 1142, Feb. 1990.

OTWAY, D., and REES, O.: "Efficient and Timely Mutual Authentication", *Operating Systems Rev.,* pp. 8-10, Jan. 1987.

PADHYE, J., FIROIU, V., TOWSLEY, D., and KUROSE, J.: "Modeling TCP Throughput: A Simple Model and Its Empirical Validation", *Proc. SIGCOMM '98 Conf.,* aCm, pp. 303-314, 1998.

PALAIS, J.C.: *Fiber Optic Commun.,* 5th ed., Englewood Cliffs, NJ: Prentice Hall, 2004.

PARAMESWARAN, M., SUSARLA, A., and WHINSTON, A.B.: "P2P Networking: An Information-Sharing Alternative", *IEEE Computer,* vol. 34, pp. 31-38, July 2001.

PAREKH, A., and GALLAGHER, R.: "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case", *IEEE/ACM Trans. on Networking,* vol. 2, pp. 137-150, Apr. 1994.

PAREKH, A., and GALLAGHER, R.: "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", *IEEE/ACM Trans. on Networking,* vol. 1, pp. 344-357, June 1993.

PARTRIDGE, C., HUGHES, J., and STONE, J.: "Performance of Checksums and CRCs over Real Data", *Proc. SIGCOMM '95 Conf.,* ACM, pp. 68-76, 1995.

PARTRIDGE, C., MENDEZ, T., and MILLIKEN, W.: "Host Anycasting Service", RFC 1546, Nov. 1993.

PAXSON, V., and FLOYD, S.: "Wide-Area Traffic: The Failure of Poisson Modeling", *IEEE/ACM Trans. on Networking,* vol. 3, pp. 226-244, June 1995.

PERKINS, C.: "IP Mobility Support for IPv4", RFC 3344, Aug. 2002.

PERKINS, C.E.: RTP: Audio and Video for the Internet, *Boston: Addison-Wesley, 2003.*

PERKINS, C.E. (ed.): *Ad Hoc Networking,* Boston: Addison-Wesley, 2001.

PERKINS, C.E.: *Mobile IP Design Principles and Practices,* Upper Saddle River, NJ: Prentice Hall, 1998.

PERKINS, C.E., and ROYER, E.: "The Ad Hoc On-Demand Distance-Vector Protocol", in *Ad Hoc Networking,* edited by C. Perkins, Boston: Addison-Wesley, 2001.

PERLMAN, R.: *Interconnections,* 2nd ed., Boston: Addison-Wesley, 2000.

PERLMAN, R.: Network Layer Protocols with Byzantine Robustness, *Ph.D. thesis, M.I.T., 1988.*

PERLMAN, R.: "An Algorithm for the Distributed Computation of a Spanning Tree in an Extended LAN", *Proc. SIGCOMM '85 Conf.,* ACM, pp. 44-53, 1985.

PERLMAN, R., and KAUFMAN, C.: "Key Exchange in IPsec", *IEEE Internet Computing,* vol. 4, pp. 50-56, Nov.-Dec. 2000.

PETERSON, W.W., and BROWN, D.T.: "Cyclic Codes for Error Detection", *Proc. IRE,* vol. 49, pp. 228-235, Jan. 1961.

PIATEK, M., KOHNO, T., and KRISHNAMURTHY, A.: "Challenges and Directions for Monitoring P2P File Sharing Networks - or Why My Printer Received a DMCA Takedown Notice", *3rd Workshop on Hot Topics in Security,* USENIX, July 2008.

PIATEK, M., ISDAL, T., ANDERSON, T., KRISHNAMURTHY, A., and VENKA- TARAMANI, V.: **"Do Incentives Build Robustness in BitTorrent?", *Proc. NSDI2007 Conf.,* USENIX, pp. 1-14, 2007.**

PISCITELLO, D.M., and CHAPIN, A.L.: Open Systems Networking: TCP/IP and OSI, *Boston: Addison-Wesley, 1993.*

PIVA, A., BARTOLINI, F., and BARNI, M.: "Managing Copyrights in Open Networks", *IEEE Internet Computing,* vol. 6, pp. 18-26, May- 2002.

POSTEL, J.: "Internet Control Message Protocols", RFC 792, Sept. 1981.

RABIN, J., and McCATHIENEVILE, C.: "Mobile Web Best Practices 1.0", W3C Recommendation, July 2008.

RAMAKRISHNAM, K.K., FLOYD, S., and BLACK, D.: "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, Sept. 2001.

RAMAKRISHNAN, K.K., and JAIN, R.: "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer", *Proc. SIGCOMM '88 Conf.,* ACM, pp. 303-313,1988.

RAMASWAMI, R., KUMAR, S., and SASAKI, G.: *Optical Networks: A Practical Perspective,* 3rd ed., San Francisco: Morgan Kaufmann, 2009.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., and SHENKER, S.: "A Scalable Content-Addressable Network", *Proc. SIGCOMM 2001 Conf.,* ACM, pp. 161-172, 2001.

RIEBACK, M., CRISPO, B., and TANENBAUM, A.: "Is Your Cat Infected with a Computer Virus?", *Proc. IEEE Percom,* pp. 169-179, Mar. 2006.

RIVEST, R.L.: "The MD5 Message-Digest Algorithm", RFC 1320, Apr. 1992.

RIVEST, R.L., SHAMIR, A., and ADLEMAN, L.: "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. of the ACM,* vol. 21, pp. 120-126, Feb. 1978.

ROBERTS, L.G.: "Extensions of Packet Communication Technology to a Hand Held Personal Terminal", *Proc. Spring Joint Computer Conf.,* AFIPS, pp. 295-298,1972.

ROBERTS, L.G.: "Multiple Computer Networks and Intercomputer Communication", *Proc. First Symp. on Operating Systems Prin.,* ACM, pp. 3.1-3.6, 1967.

ROSE, M.T.: *The Simple Book,* Englewood Cliffs, NJ: Prentice Hall, 1994.

ROSE, M.T.: *The Internet Message,* Englewood Cliffs, NJ: Prentice Hall, 1993.

ROWSTRON, A., and DRUSCHEL, P.: "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18th Int'l Conf. on Distributed Systems Platforms,* London: Springer-Verlag LNCS 2218, pp. 329-350, 2001.

RUIZ-SANCHEZ, M.A., BIERSACK, E.W., and DABBOUS, W.: "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine,* vol. 15, pp. 8-23, Mar.-Apr. 2001.

SALTZER, J.H., REED, D.P., and CLARK, D.D.: "End-to-End Arguments in System Design", *ACM Trans. on Computer Systems,* vol. 2, pp. 277-288, Nov. 1984.

SAMPLE, A., YEAGER, D., POWLEDGE, P., MAMISHEV, A., and SMITH, J.: "Design of an RFID-Based Battery-Free Programmable Sensing Platform", *IEEE Trans. on Instrumentation and Measurement,* vol. 57, pp. 2608-2615, Nov. 2008.

SAROIU, S., GUMMADI, K., and GRIBBLE, S.: "Measuring and Analyzing the Characteristics of Napster & Gnutella Hosts", *Multim. Syst.,* vol. 9,, pp. 170-184, Aug. 2003.

SCHALLER, R.: "Moore's Law: Past, Present and Future", *IEEE Spectrum,* vol. 34, pp. 52-59, June 1997.

SCHNEIER, B.: *Secrets and Lies,* New York: John Wiley & Sons, 2004.

SCHNEIER, B.: *E-Mail Security,* New York: John Wiley & Sons, 1995.

SCHNORR, C.P.: "Efficient Signature Generation for Smart Cards", *Journal of Cryptology,* vol. 4, pp. 161-174, 1991.

SCHOLTZ, R.A.: "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun.,* vol. COM-0, pp. 822-854, May 1982.

SCHWARTZ, M., and ABRAMSON, N.: "The AlohaNet: Surfing for Wireless Data", *IEEE Commun. Magazine,* vol. 47, pp. 21-25, Dec. 2009.

SEIFERT, R., and EDWARDS, J.: ***The All-New Switch Book,*** **NY: John Wiley, 2008.**

SENN, J.A.: "The Emergence of M-Commerce", *IEEE Computer,* vol. 33, pp. 148-150, Dec. 2000.

SERJANTOV, A.: "Anonymizing Censorship Resistant Systems", *Proc. First*

Int'l Workshop on Peer-to-Peer Systems, London: Springer-Verlag LNCS 2429, pp. 111-120,2002.

SHACHAM, N., and MCKENNY, P.: "Packet Recovery in High-Speed Networks Using Coding and Buffer Management", *Proc. INFOCOM Conf.,* IeEe, pp. 124-131, June 1990.

SHAIKH, A., REXFORD, J., and SHIN, K.: "Load-Sensitive Routing of Long-Lived IP Flows", *Proc. SIGCOMM '99 Conf.,* ACM, pp. 215-226, Sept. 1999.

SHALUNOV, S., and CARLSON, R.: "Detecting Duplex Mismatch on Ethernet", *Passive and Active Network Measurement,* Berlin: Springer-Verlag LNCS 3431, pp. 3135-3148, 2005.

SHANNON, C.: "A Mathematical Theory of Communication", *Bell System Tech. J.,* vol. 27, pp. 379-423, July 1948; and pp. 623-656, Oct. 1948.

SHEPARD, S.: *SONET/SDH Demystified,* New York: McGraw-Hill, 2001.

SHREEDHAR, M., and VARGHESE, G.: "Efficient Fair Queueing Using Deficit Round Robin", *Proc. SIGCOMM '95 Conf.,* ACM, pp. 231-243, 1995.

SIMPSON, W.: *Video Over IP,* 2nd ed., Burlington, MA: Focal Press, 2008.

SIMPSON, W.: "PPP in HDLC-like Framing", RFC 1662, July 1994b.

SIMPSON, W.: "The Point-to-Point Protocol (PPP)", RFC 1661, July 1994a.

SIU, K., and JAIN, R.: "A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic", *ACM Computer Communications Review,* vol. 25, pp. 6-20, Apr. 1995.

SKOUDIS, E., and LISTON, T.: *Counter Hack Reloaded,* 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2006.

SMITH, D.K., and ALEXANDER, R.C.: *Fumbling the Future,* New York: William Morrow, 1988.

SNOEREN, A.C., and BALAKRISHNAN, H.: "An End-to-End Approach to Host Mobility", *Int'l Conf. on Mobile Computing and Networking* , ACM, pp. 155-166, 2000.

SOBEL, D.L.: "Will Carnivore Devour Online Privacy", *IEEE Computer,* vol. 34, pp. 87-88, May 2001.

SOTIROV, A., STEVENS, M., APPELBAUM, J., LENSTRA, A., MOLNAR, D., OSVIK, D., and DE WEGER, B.: **"MD5 Considered Harmful Today", *Proc. 25th Chaos Communication Congress,* Verlag Art d'Ameublement, 2008.**

SOUTHEY, R.: *The Doctors,* London: Longman, Brown, Green and Longmans, 1848.

SPURGEON, C.E.: *Ethernet: The Definitive Guide,* Sebastopol, CA: O'Reilly, 2000.

STALLINGS, W.: *Data and Computer Communications,* 9th ed., Upper Saddle River, NJ: Pearson Education, 2010.

STARR, T., SORBARA, M., COIFFI, J., and SILVERMAN, P.: "DSL Advances", Upper Saddle River, NJ: Prentice Hall, 2003.

STEVENS, W.R.: TCP/IP Illustrated: The Protocols, *Boston: Addison Wesley, 1994.*

STINSON, D.R.: *Cryptography Theory and Practice,* 2nd ed., Boca Raton, FL: CRC Press, 2002.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M.F., and BALAKRISHNAN, H.: "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. SIGCOMM 2001 Conf.,* ACM, pp. 149-160, 2001.

STUBBLEFIELD, A., IOANNIDIS, J., and RUBIN, A.D.: "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", *Proc. Network and Distributed Systems Security Symp.,* ISOC, pp. 1-11,2002.

STUTTARD, D., and PINTO, M.: The Web Application Hacker's Handbook, *New York: John Wiley & Sons, 2007.*

SU, S.: The UMTS Air Interface in RF Engineering, *New York: McGraw-Hill, 2007.*

SULLIVAN, G., and WIEGAND, T.: "Tree Algorithms for Packet Broadcast Channels", *Proc. of the IEEE,* vol. 93, pp. 18-31, Jan. 2005.

SUNSHINE, C.A., and DALAL, Y.K.: "Connection Management in Transport Protocols", *Computer Networks,* vol. 2, pp. 454-473, 1978.

TAN, K., SONG, J., ZHANG, Q., and SRIDHARN, M.: "A Compound TCP Approach for High-Speed and Long Distance Networks", *Proc. INFOCOM Conf.,* IEEE, pp. 1-12, 2006.

TANENBAUM, A.S.: *Modern Operating Systems,* 3rd ed., Upper Saddle River, NJ: Prentice Hall, 2007.

TANENBAUM, A.S., and VAN STEEN, M.: Distributed Systems: Principles and Paradigms, *Upper Saddle River, NJ: Prentice Hall, 2007.*

TOMLINSON, R.S.: "Selecting Sequence Numbers", *Proc. IGCOMM/SIGOPS* Interprocess Commun. Workshop, *ACM, pp. 11-23, 1975.*

TUCHMAN, W.: "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum,* vol. 16, pp. 40-41, July 1979.

TURNER, J.S.: "New Directions in Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine,* vol. 24, pp. 8-15, Oct. 1986.

UNGERBOECK, G.: "Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction", *IEEE Commun. Magazine,* vol. 25, pp. 5-11, Feb. 1987.

VALADE, J.: *PHP & MySQL for Dummies,* 5th ed., New York: John Wiley & Sons, 2009.

VARGHESE, G.: *Network Algorithmics,* San Francisco: Morgan Kaufmann, 2004.

VARGHESE, G., and LAUCK, T.: "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.,* ACM, pp. 25-38, 1987.

VERIZON BUSINESS: 2009 Data Breach Investigations Report, *Verizon, 2009.*

VITERBI, A.: CDMA: Principles of Spread Spectrum Communication, *Englewood Cliffs, NJ: Prentice Hall, 1995.*

VON AHN, L., BLUM, B., and LANGFORD, J.: "Telling Humans and Computers Apart Automatically", *Commun. of the ACM,* vol. 47, pp. 56-60, Feb. 2004.

WAITZMAN, D., PARTRIDGE, C., and DEERING, S.: "Distance Vector Multicast Routing Protocol", RFC 1075, Nov. 1988.

WALDMAN, M., RUBIN, A.D., and CRANOR, L.F.: "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System", *Proc. Ninth USENIX Security Symp.,* USENIX, pp. 59-72, 2000.

WANG, Z., and CROWCROFT, J.: "SEAL Detects Cell Misordering", *IEEE Network Magazine,* vol. 6, pp. 8-9, July 1992.

WANT, R.: *RFID Explained,* San Rafael, CA: Morgan Claypool, 2006.

WARNEKE, B., LAST, M., LIEBOWITZ, B., and PISTER, K.S.J.: "Smart Dust: Communicating with a Cubic Millimeter Computer", *IEEE Computer,* vol. 34, pp. 44-51, Jan. 2001.

WAYNER, P.: Disappearing Cryptography: Information Hiding, Steganography, and Watermarking, *3rd ed., San Francisco: Morgan Kaufmann, 2008.*

WEI, D., CHENG, J., LOW, S., and HEGDE, S.: "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Trans. on Networking,* vol. 14, pp. 1246-1259, Dec. 2006.

WEISER, M.: "The Computer for the Twenty-First Century", *Scientific American,* vol. 265, pp. 94-104, Sept. 1991.

WELBOURNE, E., BATTLE, L., COLE, G., GOULD, K., RECTOR, K., RAYMER, S., BALAZINSKA, M., and BORRIELLO, G.: **"Building the Internet of Things Using RFID", *IEEE Internet Computing,* vol. 13, pp. 48-55, May 2009.**

WITTENBURG, N.: *Understanding Voice Over IP Technology,* Clifton Park, NY: Delmar Cengage Learning, 2009.

WOLMAN, A., VOELKER, G., SHARMA, N., CARDWELL, N., KARLIN, A., and LEVY, **H.:** "On the Scale and Performance of Cooperative Web Proxy Caching", *Proc. 17th Symp. on Operating Systems Prin.,* ACM, pp. 16-31, 1999.

WOOD, L., IVANCIC, W., EDDY, W., STEWART, D., NORTHAM, J., JACKSON, C., and DA SILVA CURIEL, A.: "Use of the Delay-Tolerant Networking Bundle Protocol from Space", *Proc. 59th Int'l Astronautical Congress,* Int'l Astronautical Federation, pp. 3123-3133, 2008.

WU, T.: "Network Neutrality, Broadband Discrimination", *Journal on Telecom. and High-Tech. Law,* vol. 2, pp. 141-179, 2003.

WYLIE, J., BIGRIGG, M.W., STRUNK, J.D., GANGER, G.R., KILICCOTE, H., and KHOSLA, P.K.: **"Survivable Information Storage Systems", *IEEE Computer,* vol. 33, pp. 61-68, Aug. 2000.**

YU, T., HARTMAN, S., and RAEBURN, K.: "The Perils of Unauthenticated Encryption: Kerberos Version 4", *Proc. NDSS Symposium,* Internet Society, Feb. 2004.

YUVAL, G.: "How to Swindle Rabin", *Cryptologia,* vol. 3, pp. 187-190, July 1979.

ZACKS, M.: "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing,* vol. 5, pp. 8-9, Nov.-Dec. 2001.

ZHANG, Y., BRESLAU, L., PAXSON, V., and SHENKER, S.: "On the

Characteristics and Origins of Internet Flow Rates", *Proc. SIGCOMM 2002 Conf.,* ACM, pp. 309-322, 2002.

ZHAO, B., LING, H., STRIBLING, J., RHEA, S., JOSEPH, A., and KUBIATOWICZ, J.: "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", *IEEE J. on Selected Areas in Commun.,* vol. 22, pp. 41-53, Jan. 2004.

ZIMMERMANN, P.R.: *The Official PGP User's Guide,* Cambridge, MA: M.I.T. Press, 1995a.

ZIMMERMANN, P.R.: *PGP: Source Code and Internals,* Cambridge, MA: M.I.T. Press, 1995b.

ZIPF, G.K.: Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology, *Boston: Addison-Wesley, 1949.*

ZIV, J., and LEMPEL, Z.: "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory,* vol. IT-3, pp. 337-343, May 1977.

ӘЛІППЕЛІК КӨРСЕТКІШ

ӘЛІППЕЛІК КӨРСЕТКІШ

Нөмірлер

A

N

O

P

Q

R

S

T

U

V

Ж

Қ

Л

М

Ө

П

Р

Т

Ц

Ч

Ш

Ы

Қазақ тіліне аударған
А. М. Махметова

Таненбаум Э. С., Уэзеролл Д. Дж

КОМПЬЮТЕРЛІК ЖЕЛІЛЕР
2-БӨЛІМ
Оқулық

Басуға 03.09.14 қол қойылды. Қағазы офсеттік. Қаріп түрі «Times». Пішіні 70x100$^{1/16}$. Баспа табағы 33,25. Таралымы: Мемлекеттік тапсырыс бойынша – 1000 дана. Тапсырыс № 1012.

Тапсырыс берушінің дайын файлдарынан басылып шықты.

ЖШС РПБК «Дәуір», 050009,
Алматы қаласы, Гагарин д-лы, 93а.
E-mail: rpik-dauir81@mail.ru, zakaz@dauir.kz