Универзитет „Св. Кирил и Методиј“ во Скопје,
Република Македонија

ТОНИ ЈАНЕВСКИ

ИНТЕРНЕТ ТЕХНОЛОГИИ

Скопје, 2015

Содржина

Глава 1

Вовед во Интернет технологии

Развојот на комуникациите и комуникациските технологии од првата деценија во 21-от век има недвосмислен правец кон една цел, а тоа е Интернет како единствена платформа за сите сервиси преку една глобална мрежа. Уште повеќе, почетниот концепт на телекомуникациите каде што имаше размена на говор меѓу двајца корисници преку телефон или пак дифузија на видео и аудио (т.е. телевизија и радио) се замени со комплетно нов пристап, каде што информациите стануваат постојано достапни и корисникот пристапува до нив по сопствена потреба или желба во било кое време и било каде да се наоѓаат. Така, хетерогеноста на мрежи за различни сервиси (телефонија, пренос на податоци, на слика или видео, аудио, мултимедија, пораки итн.) се сведе на една мрежа (Интернет) која ги обединува сите можни хетерогености на сервиси, терминални уреди (телефони, компјутери, мобилни апарати, телевизори итн.), медиуми за пренос (бакарни парици или кабли, светловоди или безжичен медиум). Всушност, Интернет ја овозможува истата работа за комуникациите и пристапот и размената на информациите меѓу луѓето или машините (компјутерите) што ја овозможува електро-дистрибутивната мрежа за електричните апарати (тостери, машини за перење, компјутери, телевизори, итн.) каде што приклучокот во мрежата (штекерот) е ист за сите различни апарати. Во случајот со Интернет, самиот пристап до Интернет е “штекер” за сите видови на комуникациски сервиси кои овозможуваат пристап до одредени информации (на пример, до одредени содржини на веб) или размена на информации (на пример, телефонија преку Интернет).

Што е Интернет?

Интернет е мрежа од мрежи кои се поврзани меѓусебе и кои го користат Интернет протоколот (IP – Internet Protocol) за меѓумрежно поврзување и размена на информациите (без разлика на типот) меѓу одделните мрежи и корисниците кои се закачени на нив.

Сите технологии кои се користат во Интернет за обезбедување на одделните сервиси се нарекуваат Интернет технологии. Бидејќи Интернет комуникацијата е независна од транспортната мрежа или од пристапната мрежа (Етернет мрежа, модемска врска, безжичен Интернет пристап и сл.), Интернет технологиите ги опфаќаат т.н. протоколни нивоа на мрежното ниво (тука е Интернет протоколот, IP, како базичен протокол во Интернет) и погорните протоколни нивоа (нивовскиот концепт во телекомуникациските системи ќе биде објаснет во продолжение).

Оваа книга навлегува во детали на сите значајни Интернет технологии кои постојат денес. Самата содржина на книгава е поделена во 10 глави.

Во првата глава се дава вовед во Интернет и Интернет технологиите.

Втората глава ги опфаќа најзначајните Интернет протоколи. Притоа, акцентот е ставен на IP протоколот во неговите две верзии, верзија 4 и верзија 6, како и на најчесто користените транспортни протоколи – TCP (Transmission Control Protocol) и UDP (User Datagram Protocol).

Во третата глава е опфатен TCP/IP клиент-серверот моделот на комуникација во Интернет и се дадени основите на мрежното програмирање.

Основните Интернет технологии на апликациско ниво (сервисите), како што се DNS (Domain Name Server), симнувањето на фајлови (File Transfer Protocol - FTP), електронската пошта (e-mail), се тема во глава четири.

Во глава пет е опфатена веб технологијата (www – world wide web) која е заснована на HTTP/TCP/IP протоколниот стек, технологија која доминира во Интернет мрежата денеска.

Мултимедиските комуникации кои стануваат се поинтересни и поактуелни заради се поголемите битски брзини за пристап на крајните корисници се опфатени во глава шест.

Во иднина телефонијата, која денес сеуште е базирана во најголем дел од системите на комутација на кола (класични телефонски мрежи), во иднина ќе биде

базирана целосно на Интернет протоколот преку користење на т.н. говор преку IP (Voice over IP - VoIP). Преносот на говор преку Интернет е тема на глава 7.

За управување со различните системи кои служат за рутирање на IP пакетите или за овозможување на одреден тип сервис за корисниците, неопходно е да се управуваат различните системи во рамките на една мрежа или во повеќе мрежи во Интернет на организиран начин и по стандардизирани протоколи, што е предмет во глава 8 од оваа книга.

Во глава 9 акцентот е ставен на сигурносните аспекти при Интернет комуникациите, што е суштинско за да може да се зачува интегритетот на информациите (на пример, на содржините на даден веб сајт), доверливоста при комуникацијата (да се спречи неовластено набљудување или изменување на информациите при нивниот пренос преку Интернет), како и да се контролира пристапот до одредени мрежи, сервиси и системи преку механизми на авторизација и автентикација на корисниците.

Во последната глава од оваа книга, глава 10, ќе биде опфатен процесот на интеграција на Интернет и класичните телекомуникациски мрежи (за пренос на говор, како и за дистрибуција на телевизија и радио), процес кој интензивно се случува во рамките на комуникациските технологии на почетокот на 21-от век, кога сите информации стануваат дигитални (говорот е дигитализиран за пренос, а до 2015 година целосно ќе "изумре" и аналогната телевизија и радио и ќе продолжат да постојат само како дигитални сервиси). Кога сервисите се дигитални, тогаш имаме пренос на дигити (т.е. бројки), најчесто бити (бидејќи тогаш е наједноставен преносот на сигналите), па во таков случај секој тип на информации може да се пренесува преку Интернет што е и главна тенденција и неспорна иднина во следните децении.

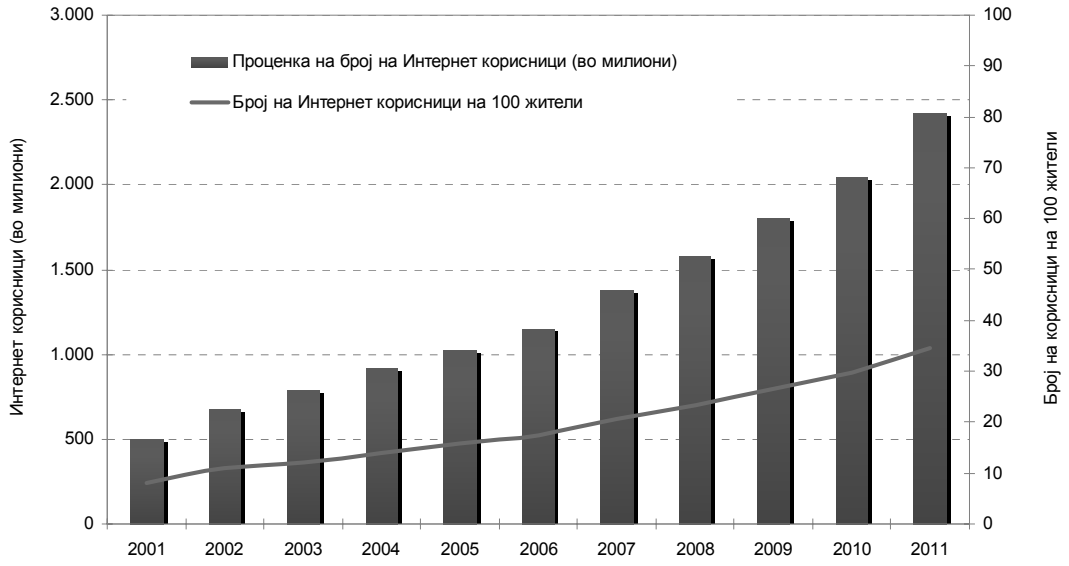
Оваа книга е наменета за студентите кои го слушаат предметот Интернет технологии на Факултетот за електротехника и информациски технологии при Универзитетот „Св. Кирил и Методиј", Скопје, но е така конципирана да може да биде искористена и од вработените во фирмите кои се занимаваат со комуникациско-информациските технологии, студентите, професионалците, како и од сите други кои се интересираат на Интернет технологиите и начините за нивното креирање (engineering) и функционирање.

Скопје, 2012 год.

Проф. д-р Тони Јаневски

1.1 Појавата и развојот на Интернет

Денес се соочуваме со невиден развој на две технологии ширум светот. Од друга страна развојот на Интернет технологијата и нејзиното раширување во светот се случува слично како порастот на мобилните комуникации. Имено, од самите почетоци на Интернет кон крајот на 60-те години, Интернет мрежата експоненцијално го зголемува бројот на корисниците и хост компјутерите секоја година. Ако се погледнат показателите за порастот на Интернет прикажан на слика 1.1, може да се забележи експоненцијалниот пораст на бројот на Интернет хост компјутери.



Слика 1.1 Растот на Интернет во период 2001-2011 година (според податоците од ITU – International Telecommunication Union)

Овој пораст се должи и на појавата на веб сообраќајот www (world wide web), кој овозможува лесно пребарување и добивање на различни типови на информации од корисниците, така што го запали фитилот за експлозијата на Интернет во годините потоа. Меѓутоа, вкупниот сообраќај по Интернет мрежата расте несомнено многу побрзо од бројот на хостовите на мрежата. Интернет е глобален и овозможува низа на мултимедиски сервиси, што допринесува Интернет мрежата да навлезе во речиси сите

аспекти на човековото постоење: во научниот свет од каде што произлезе во технолошки поглед, во општествениот живот, забавата, информативните медиуми, администрацијата, со тенденција да продолжи со понатомошно навлегување во сите пори на општеството, како што е електронска деловност (остварување на деловни трансакции преку Интернет), електронска администрација и електронска влада (обезбедување јавни сервиси на граѓаните), интерактивна ТВ и радио итн. Крајните граници не можат да се согледаат, ако воопшто и постојат. Се разбира, развојот на Интернет и неговата глобализација беа овозможени со развојот на персоналните компјутери и податочните мрежи за нивно поврзување. Како технологија Интернет се засновува на Интернет протоколот - IP (Internet Protocol), кој е доволно робустен и доволно флексибилен за да овозможи пренос на информациите сместени во IP пакети, и тоа при ниска цена за нивна контрола при преносот од крај до крај. Подетален опис на IP, како и на начините за обезбедување на доверлив пренос на IP пакетите ќе биде изнесен во продолжение на оваа книга.

Интернет доживува невиден пораст во последнава деценија. Развојот на секоја комуникациска технологија денес, безжичните комуникации или оптичките комуникации или традиционалните фиксни комуникации, мора да го имаат во предвид порастот на Интернет и неговото влијание на корисниците и на телекомуникациските мрежи. Главна причина за ваквиот брз пораст на Интернет мрежата и бројот на нејзините корисници е појавата на www и неговото брзо ширење, како и можноста да се пристапи преку www до големи количества на информации од различни области. Денес, Интернет мрежата е заснована на еден тип на сервис за сите, сервис најдобар обид (best effort). Денешната Интернет мрежа не обезбедува гаранции во однос на опсегот, загубите или доцнењето на пакетите. Втора карактеристика на денешната Интернет мрежа е хетерогеноста. Хетерогеност постои како во крајните точки, серверите и клиентите, кои што можат да бидат персонални компјутери, комуникатори, мобилни терминали, моќни сервери, така и во мрежните врски, кои што можат да бидат со протоци од неколку kbit/s до протоци кои се мерат со Gbit/s. Потоа, постои разлика и во однос на протоколите кои се наоѓаат над Интернет протоколот за контрола на транспортот, кои што можат да бидат неконекциски ориентирани (на пример UDP- User Datagram Protocol) или конекциски ориентирани доверливи протоколи (на пример, TCP- Transmission Control Protocol). На крај, постои и хетерогеност во типовите на апликации што го користат Интернет, кои што може да бидат апликации кои немаат побарувања во однос на перформансите (на пример, пренос на фајлови) до апликации

кои имаат строги побарувања во однос на загубите (packet losses) или доцнењето на пакетите (packet delay), како што е комуникација во реално време или дистрибуција на аудио и/или видео информации во реално време. Интернет е заснован на Интернет протоколот, кој е протокол на ниво 3 од хиерархијата на OSI (Open Systems Interconnection) моделот. За контрола на транспортот на пакетите и осигурување на доверлив пренос потребно е постоење на протоколи над IP. Најчесто користен протокол во стекот над IP е TCP, протокол кој обезбедува доверлив пренос на IP пакетите на Интернет. Двата протоколи се подетално објаснети во глава 2 од оваа книга.

1.1.1 Историјата на Интернет

Во продолжение фактографски се прикажани неколку значајни датуми и информации во развојот на Интернет и Интернет технологиите, [1]:

- Рани 1960-ти - претходница на Интернет е проектот на ARPA (Advanced Research Projects Agency), подоцна повремено преименувана и во DARPA (Defense Advanced Research Projects Agency), истражувачки проект воден од Licklider што започнал во октомври 1962.
- 1960-ти – појава на ARPANET и истражувањата за пакетска комутација од страна на Leonard Kleinrock (теоретски пристап) и Larry Roberts (практичен пристап).
- Септември 1969 - првиот Интернет јазол бил поставен во UCLA (University of California Los Angeles).
- 29 октомври 1969 година - првите карактери разменети преку два јазли поврзани со линк (телефонска линија) се случила на, меѓу јазлите сместени на UCLA (во лабораторијата на Kleinrock) и SRI (Stanford Research Institute).
- 1969 - RFCs (Requests For Comments) стартувани од S. Crocker, при што се започнало со RFC0001 и се продолжило во тој стил се до денешен ден, при што може да се каже дека се што е опфатено во RFC од аспект на технологии можеме да ги наречеме по дефиниција Интернет технологии (веб адресата на сајтот каде сто се поставени сите RFC-а е www.ietf.org, каде што IETF доаѓа од Internet Engineering Task Force).

- 1970-те - Robert Kahn ја започнува пионерски работата на комуникациски принципи за оперативните системи што води до промена на ARPANET кон отворена мрежна архитектура (што подоцна ќе резултира во TCP/IP протоколниот модел).
- 1972 - Email апликацијата креирана од Ray Tomlinson и Larry Roberts
- Декември 1974 год. - Vint Cerf и Robert Kahn го креираат Transmission Control Program (не протоколот TCP кој го имаме денес), како RFC 675, [2], на IETF (Specification of Internet Transmission Control Program), каде што IP бил неконекциски ориентиран датаграм сервис во рамките на програмот. Заради тоа, протоколниот модел на Интернет се нарекувал и TCP/IP (подоцна овие два програми биле раздвоени како посебни протоколи, Интернет протокол - IP и TCP како протокол над IP).
- Септември 1981 год. - стандардизиран е Интернет протокол верзија 4 (IPv4), како RFC 791 од IETF, [3], кој што денес го препознаваме под акронимот IP (Internet Protocol) и е примарен протокол кој го овозможува глобалниот Интернет.
- Септември 1981 год. - стандардизиран е TCP (Transmission Control Protocol), како RFC 793 од IETF, [4].
- 1982 - креиран е DNS (Domain Name System) како дистрибуиран и скалабилен механизам за мапирање (resolving) на хост-имиња (host names) во IP адреси.
- 1 јануари 1983 год. (flag day) - на сите хостови во Интернет во еден ден е заменет NCP (Network Control Program) со TCP/IP протоколниот стек (стандардизиран во септември 1981 год.) кој бил многу пофлексибилен и помоќен од NCP. Од тој ден наваму тоа е основниот протоколен стек кој го дефинира Интернет и кој го имаат сите хостови закачени за глобалната Интернет мрежа. Тогаш Интернет имал само неколку стотини хостови (главно на универзитети и во истражувачки лаборатории), така што таа промена била можна да се направи. Промената е направена со инсталација на хостовите на оперативниот систем Unix BSD (Berkeley Software Distribution - од Berkeley универзитетот во САД) во кој за првпат биле имплементирани IP и TCP протоколите во TCP/IP протоколен стек и за првпат е направен сокет (socket) како апстракција на оперативниот систем за апликациите кои работат на него.
- 1985 год. – NSF (National Science Foundation) во САД започнал програм за воспоставување на пристапни мрежи, централизирани околу шест

суперкомпјутерски центри (NSFNET) преку кои се поврзувале со ARPANET со користење на TCP/IP протоколниот стек.

- 1989 год. - Tim Berners-Lee во CERN ги поставува основите на најзначајниот Интернет сервис - вебот (WWW – World Wide Web), каде што апликациски протокол за комуникација е HTTP (Hyper Text Transfer Protocol):
  - Предлог за WWW во 1990 (со име „WorldWideWeb“)
  - Прва веб страница на 13 ноември 1990
- Мај 1996 год. - се стандардизира HTTP верзија 1.0 од IETF (RFC 1945), [5].
- Јануари 1997 год. - се стандардизира HTTP верзија 1.1 од IETF (првично со RFC 2068, а подоцна подобрена со RFC 2616 во јуни 1999 год., [6]).
- 1990-те - се создава W3C (World Wide Web Consortium, www.w3.org), тело кое се грижи за развојот на веб технологиите преку соодветни спецификации, препораки и софтверски алатки.
- Декември 1998 год. - стандардизирана е верзија 6 на Интернет протоколот - IPv6, [7], која долгорочно треба да ја замени иницијалната верзија (IPv4) од 1981.
- 2000 год. - околу 50 милиони хостови поврзани на Интернет
- 2012 год. - околу 2,5 милијарди хостови поврзани на Интернет
- 2024 год. (предвидување) - бројот на уреди поврзани на Интернет ќе биде повеќе пати поголем од бројот на луѓе корисници поврзани на Интернет, при што целата човечка популација на Земјата ќе биде поврзана на Интернет (секој поединец во просек ќе има Интернет пристап преку повеќе типа на хостови или уреди, фиксни и мобилни, било каде и во било кое време). Груба проценка на бројот на хостови (вклучувајќи и различни уреди): околу 100 милијарди хостови приклучени на Интернет.

Еден од визионерите на Интернет и неговите творци Licklider има изјавено:

> “On-line interactive communities... will be communities not of common location, but *of common interest*.... the total number of users...will be large enough to support extensive general purpose [computers]. All of these will be interconnected by telecommunications channels... [to] constitute a labile network of networks--ever changing in both content and configuration.”
>
> J. C. R. Licklider

1.1.2 Интернет – револуција во развојот на комуникациите и комуникациските технологии

Од аспект на развојот на телекомуникациите може да се издвојат неколку значајни настани (револуции):

- Воведувањето на автоматската телефонска централа (кон крајот на 19-от век);
- Дигитализацијата на телекомуникациските системи во 1970-те, 80-те и 90-те години;
- Интеграција на конекциски ориентираните телекомуникации со комутација на канали и пакетски-базираните неконекциски Интернет комуникации во 1990-те и 2000-те години и премин кон all-IP (се преку Интернет);
- Целосно IP базирани мрежи и сервиси, конвергенција на сите мрежи, сервиси, уреди, кон Интернет технологиите, 2010-те години и понатаму.

Во телекомуникациите, основната филозофија е секогаш да се прави баланс меѓу трошоците и квалитетот на дадена услуга, т.е. мрежните оператори и давателите на услуги настојуваат да дадат поголем квалитет на сервисот (Quality of Service) со пониски трошоци така да крајните корисници можат да ги купуваат сервисите.

Имено, не е секогаш во прашање дали нешто може да се направи, туку најчесто е важно по која цена тоа може да се направи, мислејќи притоа на комерцијален развој и ширење на одредена технологија. Додека во одредени случаи (за војска или полиција во дадена земја), цената може да не е во прв план, туку функционалноста, кога се работи за широко распространети сервиси суштински се трошоците за давање на конкретната услуга т.е. сервис.

Во телекомуникациските системи и мрежи има два главни дела: комутација/рутирање и пренос.

Од друга страна, има два главни трошоци за мрежните оператори:

- Трошоци за опрема и инсталација, и
- Трошоци за одржување на системите.

Која е причината за успехот на Интернет?

Врз основа на претходната дискусија, една од причините секако е ниската цена на мрежната опрема и воопшто на сервисите до крајните корисници. Имено, мрежните елементи кои се користат во Интернет (комутатори, рутери) вообичаено имаат многу пати пониска цена од соодветната опрема во класичните телекомуникации каде што доминантен сервис е телефонија (на пример: централите со комутација на кола). Во балансот на квалитет и цена, кај Интернет квалитетот на сервисот при иницијалниот дизајн на Интернет бил ставен во втор план, така да тоа овозможило со доста поевтина опрема и на поедноставен начин да се имплементира некоја мрежа и да се поврзе со глобалниот Интернет. Така, денес имаме т.н. best-effort тип на сервис во Интернет, што значи дека расположливиот капацитет на даден линк во Интернет подеднакво се дели меѓу сите конекции кои го користат тој линк во даден момент.

Втората причина е тоа што Интернет е креиран така да претставува единствена мрежа преку која може да се обезбедуваат различни типови сервиси (веб, телефонија, видео стриминг, пораки итн.), што била дамнешна идеја во телекомуникациите – наместо да се има различни мрежи за различни телекомуникациски сервиси (на пример, телефонска мрежа за говор, ТВ дистрибутивна мрежа за телевизија, посебна мрежа за радио, посебни мрежи за пренос на податоци итн.), сите сервиси да се нудат преку една мрежа, преку еден приклучок кон "мрежата". Тоа го овозможува Интернет на многу ефикасен начин, каде што сервисите се всушност апликации во хостовите (компјутерите, мобилните телефони и сите уреди со Интернет конективност), а рутирањето на пакетите од крај до крај е овозможено со Интернет протоколот (IP – Internet Protocol) кој е имплементиран во сите Интернет хостови (кориснички уреди и сервери приклучени на Интернет).

1.2 Основна Интернет архитектура

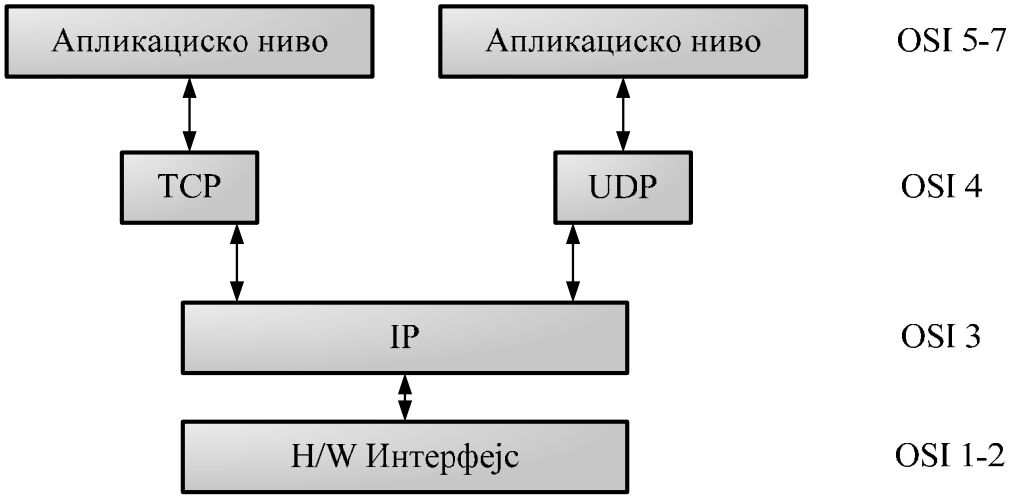
Телекомуникациските мрежи се класифицираат во различни категории во зависност од географската распространетост, намената и имплементацијата. Секоја специфицирана мрежа има усвоена архитектура и протоколна организација. За пренос на податоци се развиени различни мрежни архитектури.

Под телекомуникациска мрежа подразбираме множество од терминали (телефони, компјутери, PDA итн.) и комутациски елементи поврзани меѓусебе. Начинот на поврзување на елементите во мрежата ја одредува нејзината топологија, која може да биде прстен, ѕвезда, или да има глобална топологија (како Интернет кој претставува множество од различни пристапни локални, метро, скелетни, национални, транзитни мрежи).

За полесно градење на софтверот за комуникација меѓу процесите кои се наоѓаат на иста или на различни машини се увидело дека е потребна поделба и групирање на задачите кои треба да се извршат, во нивоа со точно дефинирани интерфејси меѓу нив. Притоа, при креирањето на архитектурата за комуникација меѓу процесите било потребно таа да не биде ограничена на одреден систем туку да може да се применува глобално. Како стандард е прифатен т.н. OSI (Open Systems Interconnection) модел, кој содржи седум нивоа (наброени одејќи од горе кон долу во протоколниот модел):

- Апликациско ниво (application layer)
- Презентациско ниво (presentation layer)
- Сесиско ниво (session layer)
- Транспортно ниво (transport layer)
- Мрежно ниво (network layer)
- Податочно ниво (data-link layer)
- Физичко ниво (physical layer)

Мрежната архитектура за комуникација која ги користи TCP/IP и UDP/IP протоколите се состои од 4 нивоа, при што OSI нивоата од 5 до 7 се обединети во едно ниво (5-то ниво), кое е апликациско ниво во Интернет. Мрежното ниво во Интернет е IP, а на транспортното ниво се користат најчесто TCP и UDP протоколите. Најдоле се наоѓа хардверскиот интерфејс кој ги содржи OSI нивоата 1 и 2. На слика 1.2 е дадена споредба меѓу OSI и TCP/IP моделите.

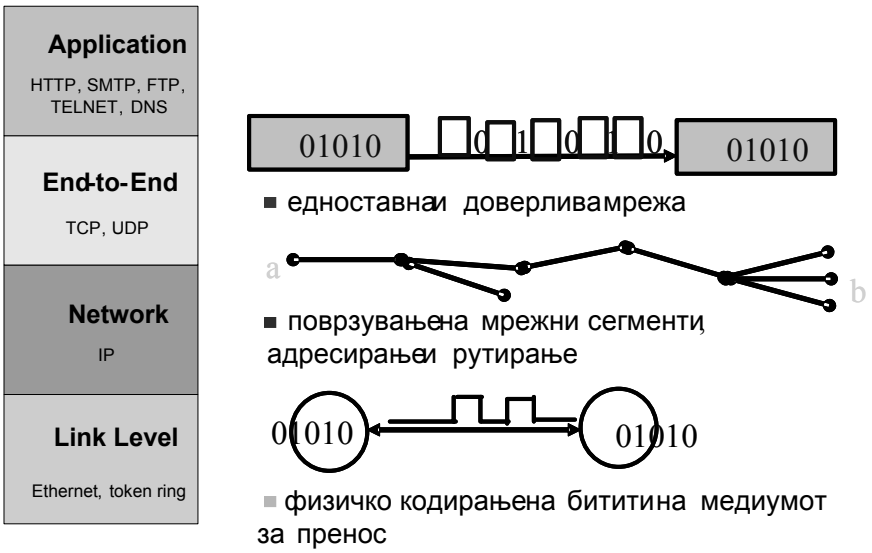


Слика 1.2 Споредба на Интернет мрежниот модел со OSI моделот

Интернет е составен од многу т.н. IP мрежи. Секоја IP мрежа е одделена од други IP мрежи преку рутери. Елементите кои можат да постојат во една IP мрежа во Интернет можат да бидат:

- Хостови (hosts), компјутери т.е. Интернет терминали кои содржат апликации кои се активни на мрежата.
- Рипитер (repeater), мрежен уред кој се користи за регенерирање како на аналогни така и на дигитални сигнали кои при преносот претрпеле слабеење. Овој уред не носи интелигентна одлука при проследувањето на пакетите како рутерот или мостот. Функционира на прво (физичко) ниво.
- Хаб (hub), служи за концентрација на физичките врски. Со него се овозможува група од хостови мрежата да ги гледа како единствена единица. Најчесто се пасивни уреди без друг ефект при преносот на податоците. Постојат и активни Hub-ови кои покрај концентрирањето на хостовите вршат и регенерирање на сигналите. Функционира на прво (физичко) ниво.
- Мостови (bridges), вршат конвертирање на мрежните податочни форми како и управување со трансмисијата. Овозможуваат комуникација меѓу локалните мрежи (LAN - Local Area Networks), како и поделба на еден колизионен домен на два. Вршат проверка на податоците дали треба да поминат низ мостот или не. Со нив се овозможува поголема функционалност на секој дел од мрежата. Функционираат на второ ниво. За проследувањето на податоците ги користат MAC адресите.

- Комутатор (switch), мрежен уред за кој може да се каже дека претставува повеќе-портен мост кој функционира побрзо. Најчесто функционираат на второ (податочно) ниво, но има и такви кои имаат улога на рутери и функционираат на трето (мрежно) ниво. Најчесто се користат денес во градењето на Етернет (Ethernet) локални мрежи како најтипичен претставник на IP мрежа.
- Рутери (routers), компјутери кои содржат активни апликации за рутирање на пакетите во мрежата зависно од нивната дестинација. За рутирањето на податоците ја користат мрежната адреса на праќачот и примачот (не користат кориснички адреса како што се телефонските броеви во класичните телекомуникациски мрежи). Функционираат на трето (мрежно) ниво (гледано според OSI моделот).



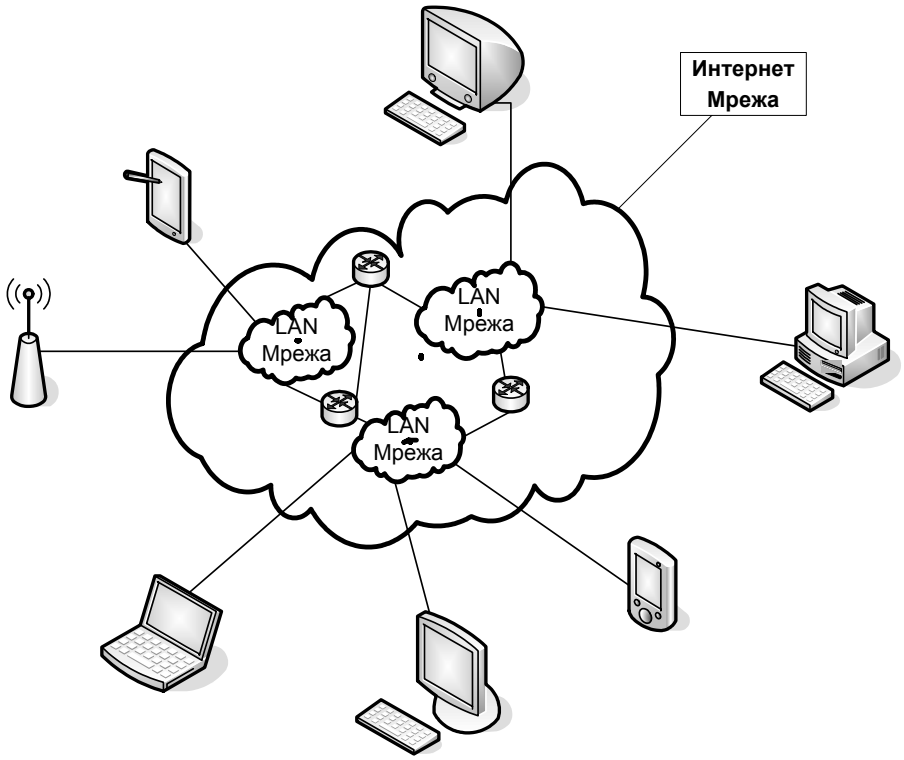
Слика 1.3 Интернет протоколи и функционалности во слоеви

За да можат мрежните елементи (апликации, хостови, рутери) да комуницираат меѓусебе потребно е да постојат одредени правила. Токму тие правила се нарекуваат протоколи. Протоколите (пример: TCP, UDP, IP и други) го одредуваат форматот и редоследот на пораките кои се разменуваат, како и акциите кои се преземаат при примањето на пораките. Најтипичните протоколи во соодветните OSI нивоа, кои се користат во Интернет, се прикажани на слика 1.3.

Значи една глобална мрежа (како што е Интернет) се состои од географски дистрибуирани хардверски и софтверски компоненти. При комуникација на машините

преку мрежи логичката комуникација на процесите од одредено ниво на едната машина е со процесите (ентитети) од истото ниво во протоколниот модел на другата машина. На пример, апликацијата која е на апликациско ниво на една машина комуницира со апликацијата на друга машина во мрежата.

Комуникацијата меѓу ентитетите се остварува преку размена на количества на податоци наречени пакети. Податоците се сместуваат во пакети од страна на апликацијата и се упатуваат кон подолните нивоа (транспортно ниво, мрежно ниво, податочно ниво) кои им додаваат соодветни заглавја, а при прием на пакетите се отстрануваат заглавјата на пакетите од страна на истите нивоа на приемната страна.



Слика 1.4 Основна архитектура на Интернет

Во Интернет постојат различни типови мрежи, кои зависно од географската област која ја опфаќаат се делат на: локални мрежи - LAN (Local Area Network), метрополитен мрежи - MAN (Metropolitan Area Network), мрежи на поголеми растојанија -WAN (Wide Area Network), како и мрежи со специјална намена.

Интернет претставува колекција од различни видови на мрежни архитектури кои се меѓусебно поврзани (слика 1.4). Функционирањето на Интернет укажува дека интерконектираните мрежни системи прифаќаат соодветни протоколи кои

овозможуваат сите компјутери кои се приклучени на оваа мрежа (се мисли на глобалниот Интернет) меѓусебно да комуницираат.

Меѓуповрзувањето на мрежите со цел да се создаде Интернет мрежа се врши со сметачки машини наречени IP рутери кои се приклучуваат меѓу две или повеќе мрежи. Секој од овие два уреди мора да знае како да изврши проследување на пакетите кои пристигнуваат до нив (до рутерите) до дестинацијата на истите (на пакетите).

Секоја IP мрежа е уникатна и самостојна. Може да функционира нормално и без да биде приклучена на друга мрежа. Поврзувањето на дадена IP мрежа (т.н. мрежен сегмент) со една или повеќе други мрежи се врши со рутери и се нарекува internetworking т.е. меѓумрежно работење. Комуникацијата во Интернет се одвива по best-effort принципот кој е имплементиран во сите рутери (одделни рутери може да имаат и дополнителни функционалности, на пример од аспект на обезбедување на одреден квалитет на сервисите). Рутерите се во функција на меѓумрежното поврзување. Притоа, мрежите не се подложни на централизирана контрола.

Овие принципи како и флексибилноста на мрежата одат во прилог на различните апликации и податоци кои се пренесуваат низ мрежата со цел да се поддржат сите видови сервиси, оние кои постојат денес, но и оние што ќе се појават во иднина.

1.3 Карактеризација и класификација на IP сообраќајот

За да може да се пристапи кон креирање на IP мрежа потребно е на почеток прво да се анализира IP сообраќајот и да се одредат неговите специфичности.

Меѓутоа, за да може да се овозможи поддршка на различни класи на сообраќај кои функционираат во реално време или не, и кои побаруваат различно ниво на квалитет на сервисот, потребно е да се одреди природата на IP сообраќајот и да се согледаат неговите карактеристики. Традиционалните техники во телекомуникациите за поддршка на QoS (Quality of Service) се засновани на говорниот сообраќај, каде што распределбата на ресурсите е детерминистичка (доделување и комутација на канали). За разлика од традиционалните телекомуникациски мрежи IP сообраќајот има две фундаментални разлики: 1) завземањето на ресурсите е динамичко (на пакетско ниво, а

не на ниво на врска), и 2) нема експлицитна поддршка за обезбедување на одреден квантитет на мрежни ресурси. Но, од друга страна ваквата едноставност на IP мрежите овозможува различните типови на сообраќај: говор, видео, аудио и податоци, да се пренесуваат преку иста мрежа, без потреба да се гради паралелна мрежна инфраструктура за секој тип на сообраќај. Ваквата карактеристика на IP мрежите, во услови на забрзан развој и понуда на мултимедиски сервиси, допринесе IP да се појави како главен новитет во концептите за сегашните и идните телекомуникациски мрежи.

1.3.1 Карактеристики на IP сообраќајот

Интернет сообраќајот, заснован на IP протоколот, опфаќа низа на мултимедиски сервиси, кои што имаат различни карактеристики во однос на нивните сообраќајни специфичности: битска брзина, избувливост, времетраење на врската; и нивните побарувања во однос на перформансите на сообраќајот: доцнењето на пакетите, загубите и протокот.

Исто така, при анализирањето на IP сообраќајот од интерес се и карактеристиките на агрегатниот Интернет сообраќај, односно мрежните карактеристики. Осознавањето на агрегатниот Интернет сообраќај е од големо значење за негово класифицирање и карактеризирање со статистички показатели.

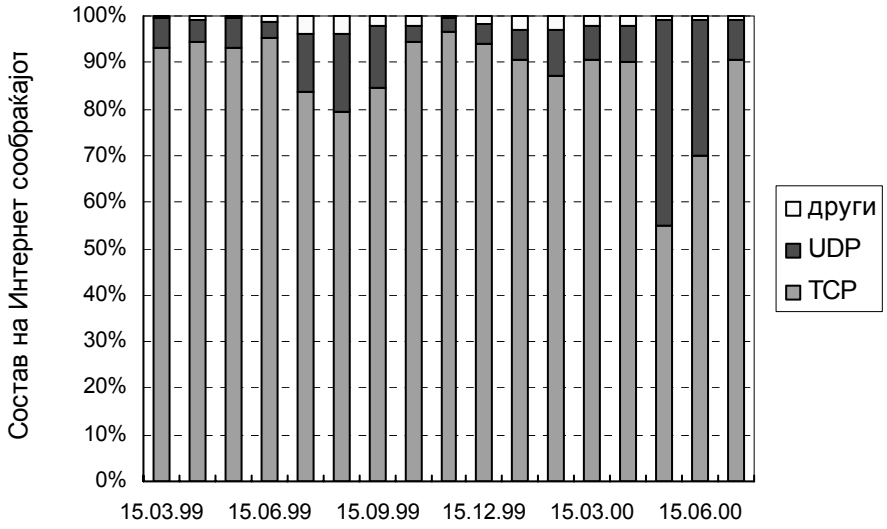
Во овој дел ќе биде разгледан агрегатниот сообраќај во Интернет со користење на траги од реални мерења (traces), а потоа ќе биде филтриран сообраќајот на одделните сообраќајни компоненти според апликациите кои учествуваат во неговото генерирање.

Агрегатен Интернет сообраќај

Доминантен сервис во денешниот Интернет е best-effort сервисот, односно ист сервис за сите без разлика на крајните апликации. Целата контрола на транспортот е препуштена на крајните точки. Од друга страна, сообраќајот потекнува од комуникацијата клиент-сервер (за ова ќе стане повеќе збор во глава 3) на чија што основа е и изграден Интернет. Имено, постојат јазли во мрежата кои што овозможуваат одредени сервиси за Интернет корисниците, наречени сервери, како и апликации во крајните кориснички терминали (персонални компјутери, мобилни апарати и сл.) кои

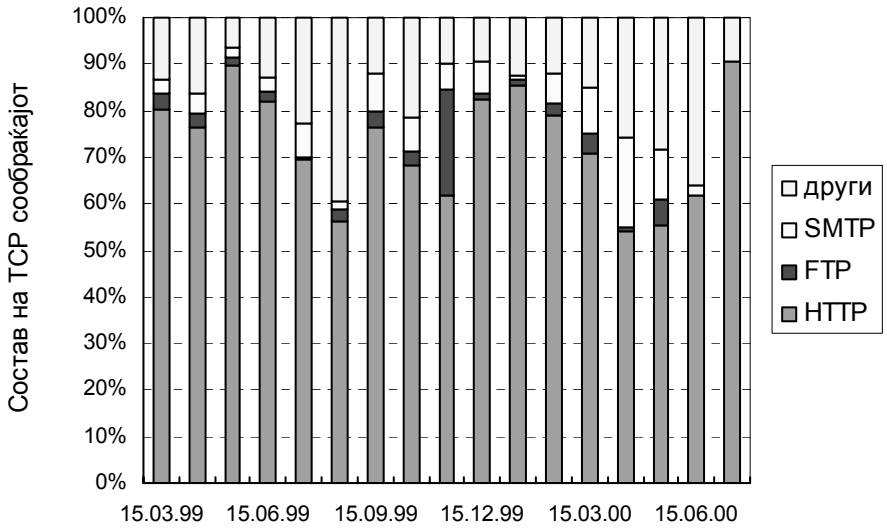
што ги побаруваат сервисите. Притоа, за да се овозможи транспортот на пакетите, информациите се делат на пакети на кои што потоа им се додаваат заглавија на патот од апликацијата до транспортниот медиум, односно истите се отфрлаат во обратна насока, од физичкото ниво кон апликацијата. Различни апликации користат различни транспортни протоколи зависно од нивните сообраќајни побарувања (на пример: TCP, UDP). Овие протоколи комуницираат со крајните апликации преку т.н. сокети (sockets). Од друга страна, меѓу транспортните протоколи и физичкиот медиум во Интернет секогаш се наоѓа IP протоколот. Од таа причина, агрегатниот сообраќај во Интернет ќе го референцираме како IP (или Интернет) сообраќај.

Во продолжение ќе разгледаме сообраќајна анализа со обработка на Интернет траги во период од 24 часа и 7 дена од приватна Интернет скелетна мрежа. Анализите се направени на агрегатниот Интернет сообраќај и на сообраќајот по протоколи. Се потврдува дека најголем дел од Интернет сообраќајот потекнува од TCP сообраќај (95% од сите бајти, 85-95% од сите пакети и 75-85% од сите потоци), а најголем дел од TCP се должи на www сообраќајот (65-80% од сите бајти, 55-75% од сите пакети и 65-75% од сите потоци). Притоа, во времето пред појавата на www, TCP сообраќајот се должел на преносот на фајлови (FTP), електронската пошта, интерактивни апликации итн. По појавата на www, заснован на користење на HTTP (Hyper Text Transport Protocol) протоколот на апликативно ниво и TCP на транспортно ниво, овој сообраќај заземал доминантна улога во поделбата на агрегатниот Интернет сообраќај. Сите анализи досега покажуваат дека TCP сообраќајот зазема доминантно место во Интернет. Тоа е и очекувано, имајќи го во предвид карактерот на денешниот Интернет, кој обезбедува еден сервис за сите и не дава поддршка за квалитет на сервисот. Во такви услови на постоење само на best-effort сервис очекувано е да се користи најмногу протокол кој ќе обезбедува доверлив пренос.



Слика 1.5 Состав на Интернет сообраќајот (според број на октети) по протоколи

На слика 1.5 се прикажани мерењата на сообраќајот на еден од Интернет линковите меѓу Јапонија и САД. Мерењата ја покажуваат процентуалната застапеност на сообраќајот според транспортни протоколи, Очигледно е дека TCP сообраќајот е доминантен во Интернет, на ист начин како што тоа го покажуваат и други анализи во текот на претходните децении, [8].



Слика 1.6 Распределба на TCP сообраќајот по тип на апликација

Компоненти на Интернет сообраќајот

Агрегатниот Интернет сообраќај може да се подели на сообраќај по транспортни протоколи и сообраќај по тип на апликација. Понатаму, секој од овие сообраќајни делови е составен од повеќе мултиплексирани потоци. Притоа, една или повеќе врски можат да бидат иницирани од ист корисник и да бидат активни паралелно (на пример: повеќе активни отворени сесии од еден персонален компјутер).

Покажавме дека TCP сообраќајот е доминантен во Интернет денес. На слика 1.6 е прикажано распределбата на агрегатниот TCP сообраќај по апликации. Согласно овие податоци 55-90% од вкупниот TCP сообраќај е www сообраќај. Помал дел од сообраќајот произлегува од FTP, SMTP (Simple Mail Transfer Protocol) и други апликативни протоколи. Но, оваа статистика е од пред една деценија. Во последнава деценија се појави и се рашири и peer-to-peer сообраќајот (од различните peer-to-peer апликација од 2000-та година наваму), кој што зазема денес значаен процент од вкупниот TCP сообраќај, а делумно има удел и во UDP сообраќајот.

Табела 1.1 Споредба на UDP и TCP сообраќајот во однос на број на пакети , бајти и потоци (flows), според CAIDA (http://www.caida.org)

Период	**UDP/TCP однос**			**Вкупен измерен IP сообраќај како референца (пакети/бајти/потоци)**
	пакети	**бајти**	**потоци**	
08-2002	0.11	0.03	0.11	(1371M/838GB/79M)
01-2003	0.12	0.05	0.27	(463M/267GB/26M)
06-2008	0.14	0.05	1.43	(4427M/2279GB/197M)
02-2009	0.19	0.07	2.34	(1922M/1410GB/110M)

Иако TCP е доминантен транспортен протокол во Интернет, еден незанемарлив дел од сообраќајот произлегува од UDP ориентираните врски. UDP најчесто се користел за комуникација сервер-сервер и за DNS (Domain Name Server) сообраќај, и сеуште се користи за таа намена. UDP е погоден и се користи за сервисите во реално време (на пример, во комбинација со RTP - Real Time Protocol), но за таа цел е потребно обезбедување на т.н. QoS (Quality of Service) поддршка во Интернет (на пример, ограничено доцнење на пакетите, гарантиран битски проток, итн.). Со премин на сервисите во реално време од посебни мрежи за таа намена (на пример, телефонска мрежа, ТВ дистрибутивна мрежа) во Интернет се поголем дел зазема и UDP

сообраќајот кој се користи за пренос на корисничките информации од говор, видео или мултимедија во реално време, што може да се забележи и од табела 1.1 (дадена е споредба на UDP и TCP сообраќајот).

Секако, ваквите анализи може да варираат од регион до регион во светот, а меѓу другото зависат и од битските брзини со кои Интернет корисниците пристапуваат до мрежата, како и од сервисите кои се нудат на корисниците од мрежните провајдери (на пример, операторите што нудат Интернет пристап) и од сервис провајдерите.

Глава 2

Основни протоколи во Интернет

2.1 Вовед

Како што спомнавме Интернет мрежата е WAN (World Area Network) мрежа, глобална и најголема постоечка мрежа во компјутерските комуникации. Иако во почетокот се појави како мрежа исклучиво наменета за пренос на податоци, поради вртоглавиот развој на мултимедиските апликации, Интернет денес нуди многу повеќе услуги од првобитните “Telnet” и “FTP-File Transfer Protocol" апликации. Основните протоколи на кои се базира Интернет претрпеле само мали измени во изминатите 30 години и покрај тоа што во меѓувреме се појавија други мрежни технологии и концепти (ISDN – Integrated Services Digital Networks, Frame relay, ATM – Asynchronous Transfer Mode, и други). Неколку децении стариот TCP/IP протоколен стек преживеа и стана основа на една инфраструктура со планетарни димензии, која ќе постои и ќе се развива и понатаму со исто или слично темпо. Она што го одржа Интернет досега се нарекува TCP/IP протоколен стек, кој генерално се состои од 5 нивоа. На врвот се наоѓа апликациското ниво односно некоја од повеќето апликации со кои корисникот непосредно комуницира, но она што го чини Интернет една единствена мрежа е протоколот на мрежно ниво - IP (Internet Protocol). За разлика од некои други мрежни протоколи, тој е замислен и дизајниран да располага со особините на "internetworking”, односно со способност успешно да ги обезбедува своите сервиси користејќи ги услугите на различни мрежи, кои се базираат на различни технологии. На границата меѓу тие мрежи се наоѓаат рутерите, кои всушност ја обезбедуваат интерконекцијата

меѓу тие различни мрежи на трето ниво-мрежно ниво, токму преку IP протоколот. Сепак сите овие функции IP протоколот сам не може да ги извршува. Затоа, покрај IP на мрежно ниво постојат и некои други контролни протоколи. Комуникацијата во Интернет се одвива на следниов начин. Податочниот поток кој доаѓа од апликацијата, транспортното ниво (TCP или UDP протоколот) го организира во парчиња кои ги предава на IP ентитетот, кој пак од нив формира датаграми. Тие се со максимална должина од 64kB (65535 бајти), но најчесто во реалноста се со должина до 1500 бајти. Секој датаграм независно се пренесува низ Интернет. Притоа, секој од нив вдолж транспортот до дестинацијата, може да биде фрагментиран во помали датаграми, кои во дестинацискиот IP ентитет ќе бидат реасемблирани во изворниот датаграм. Таквиот датаграм потоа се предава на транспортното ниво кој понатаму го проследува до корисничката апликација.

Значи, задачата на IP е да обезбеди најбрз и најекономичен (best effort) можен транспорт на датаграмите од изворот до дестинацијата, без разлика дали тие две машини се на иста или различна мрежа, и без разлика дали помеѓу нив постои некоја трета мрежа. Значи IP протоколот мора да го почитуваат и да го имаат имплементирано и хостовите и рутерите. Тој е неконекциски ориентиран што значи дека патеката т.е рутата од изворот до дестинацијата не треба да се воспостави пред податоците да влезат во мрежата. Можно е секој пакет да си има своја различна независна рута од рутата на претходниот пакет кој има иста изворна и дестинациона адреса. IP не гарантира дека пакетите ќе пристигнат во оригиналната секвенца ниту пак дека воопшто ќе пристигнат до нивната дестинација.

2.2 Интернет протокол

Во Интернет на мрежно ниво секогаш се наоѓа Интернет протоколот (www.ietf.org, RFC 791). Постојат (денес) две верзии на Интернет протоколот:

- Верзија 4 (IP version 4 – IPv4), RFC 791 (септември 1981 год.), [3],
- Верзија 6 (IP version 6 – IPv6), RFC 2460 (декември 1998 год.), [4].

Описот на IP ги содржи следниве многу значајни елементи:

- IP ги дефинира основните податочни единици кои можат да се испратат преку Интернет т.е IP дефинира формат на податочни единици (датаграми) кои се испраќаат.
- IP софтверот ги извршува рутирачките функции засновани на IP адресите.
- IP содржи сет на правила за тоа како хостовите (крајните корисници) и рутерите да се справат со добиените датаграми, како и кога да се генерираат пораки за грешка и за тоа кога може датаграмите да се отстранат од мрежата.

Слика 2.1 Формат на IPv4 заглавие

Интернет мрежата е изградена врз Интернет протоколот IP, верзија 4. Протоколот овозможува рутирање на IP пакетите во пакетски комутирани мрежи со користење на т.н. IP адреси. Имено, секоја пристапна точка на Интернет мрежата има своја IP адреса, која што во верзијата 4 на IP (IPv4) содржи 4 бајти. Со порастот на Интернет мрежата, можно е да се очекува адресниот простор со 4 бајти на IPv4 да не ги задоволува потребите, па за таа цел е стандардизирана и нова верзија на IP, верзија 6 (IPv6). Во оваа верзија на IP е доделен поголем адресен простор во заглавието на пакетите, односно просторот од 32 бити во IPv4 е проширен на 128 бити. Меѓутоа, сеуште е актуелна верзијата 4, заради што во рамките на оваа книга доколку не е поинаку нагласено под IP ќе се подразбира IPv4.

Две основни функции на IP се адресирањето и фрагментацијата. Адресирањето е потребно за да може секој елемент (јазол) на патот на Интернет мрежата да го рутира IP пакетот кон дестинацијата. Фрагментацијата е потребна за да може да се излезе во пресрет на различните ограничувања на должината на IP пакетите во различните

мрежи. Интернет протоколот го третира секој пакет независно од другите пакети во мрежата.

Овој протокол постои во секој елемент на Интернет (рутер, хост) за да може да се извршува рутирањето на IP пакетите на патот од изворот до дестинацијата. За извршување на сервисите Интернет протоколот ги користи механизмите: тип на сервис (Type of Service – ToS), преостанато време на живот на IP пакетот (Time To Live – TTL), опциите и контролата на грешка во заглавието на пакетот. Подетален опис на IP заглавието може да се најде во RFC791. Ќе се задржиме на типот на сервисот каде што единствено се специфицирани можности за контрола на QoS. Ова поле во IP заглавието има должина 1 бајт. Во IPv6 дефинирано е поле со име DS (Differentiated Services) наместо ToS полето во IPv4, со иста функција. Притоа, првите три бита се наменети за контрола, додека 4-6 бит се наменети за специфицирање на побарувањата во однос на доцнењето на пакетите, протокот (throughput) и доверливоста на преносот (reliability). Останатите два бити се оставени за идна употреба, а во скоро време е предложено нивно користење за реализација на концептот на диференцирани сервиси по класи.

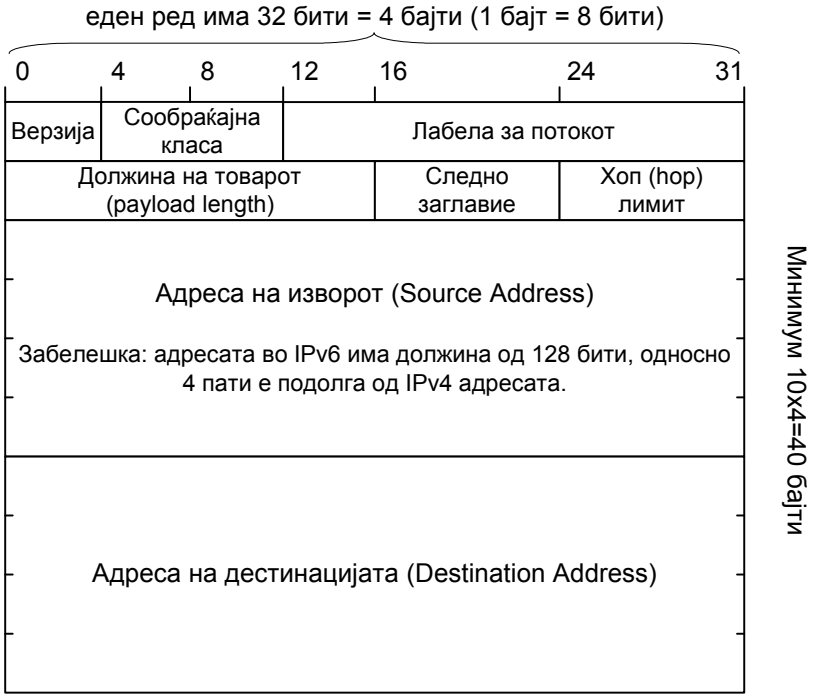
Меѓутоа, Интернет протоколот не обезбедува доверлив пренос на информациите. За да се овозможи тоа потребно е да се постави протокол за контрола на транспортот над IP (на транспортните протоколи ќе се задржиме во повеќе детали понатаму во оваа глава).

Развојот пак на IPv6 главно бил мотивиран од малиот адресен простор во IPv4. Експанзијата на користењето на Интернет услугите довела до потреба да се редефинира заглавјето со цел да се овозможи поефикасно рутирање.

2.2.1 IPv6 – замена за IPv4 ?

IPv6 не е директно компатибилна со IPv4. Адресната шема на IPv6 е нова и е базирана да им служи на демографски и модерни мрежи. Адресниот простор на IPv6 е долг 128 бити со кој е овозможено постоење на огромен број на адреси. Во пракса големината на адресите е помалку важна отколку структурата. IPv6 има три вида на unicast адреси, нуди нов формат на unicast адреси и најавува нови anycast адреси. Multicast форматот во IPv6 овозможува огромен број на можни групни кодови (2^{119}) секој идентификувајќи два или повеќе примачи. Anycast адресата е единечно доделена на повеќе од еден интерфејс, кои обично припаѓаат на различни компјутери. Пакетот

испратен со оваа адреса се рутира до најблискиот интерфејс кој ја има истата. Како и да е IPv6 не е развиена со цел да го разреши само проблемот со адресите бидејќи истиот може да биде решен со воведување на NAT (Network Address Translation) на тој начин дозволувајќи луѓето и организациите да користат неуникатни т.н. приватни IPv4 адреси кои мора да бидат мапирани во уникатни јавни IP адреси, бидејќи само јавните IP адреси се видливи во Интернет глобално додека приватните IP адреси се видливи во рамките на мрежата (сегментот) кој се наоѓа од едната страна на машината која врши NAT (таа машина најчесто се нарекува и портен јазел – gateway).



Слика 2.2 IPv6 заглавие, [7]

- Верзија (Version): ја специфицира верзијата на IP.
- Сообраќајна класа (Traffic Class): т.е. DiffServ DS-byte за IPv6 пакети за дефинирање на QoS сообраќајна класа за даден тип сообраќај во мрежата.
- Лабела за потокот (Flow Label): има ненулта вредност доделена од изворот која ги идентификува пакетите од соодветниот податочен поток кој е наменет за некои посебни услуги (QoS или нестандардни сервисни услуги). Ова поле е долго 20 бити.
- Должина на товарот (Payload Length): 16-bit вредност која ги дефинира должините на октетите на товарот како и дополнителните заглавија кои го следат основното IPv6 заглавие.

- Следно заглавие (Next Header): го дефинира типот на заглавието кое е следно и притоа тоа може да биде дополнително, транспортно заглавие или податочен товар. Има должина од еден октет.
- Хоп лимит (Hop Limit): 8-bit вредност која ги нуди истите функции како и TTL (Time To Live) полето во IPv4.
- Source Address: 128-bit изворна IPv6 адреса.
- Destination Address: 128-bit дестинациона IPv6 адреса.

Како новитет во однос на IPv4, IPv6 ги подржува QoS на мрежно ниво. Ова е овозможено индиректно со користење на flow labels и приоритетна индикација но притоа нема гаранција за реалните end-to-end QoS (Quality of Service) како што нема и резервирање на мрежни ресурси. При користење заедно со протоколи за резервирање на мрежните ресурси (како на пример RSVP – Resource reSerVation Protocol) вистинските end-to-end QoS можат да се овозможат со користење на IPv6 функционалностите.

Во поглед на безбедноста IPv6 подржува автентификација и приватност. Овозможува основни функционалности на наплаќање на услугите (тарифирање) како и наплаќање за идните видови сообраќај. Се разбира, тоа е можно и кај IPv4 со користење на одредени протоколи (на пример: RADIUS протоколот, кој ќе биде опфатен во глава 9 од оваа книга).

Со цел да се подобри рутирањето форматот на заглавјето е фиксен што овозможува хардверско процесирање на истото со што се овозможува побрзо рутирање во споредба со рутирањето со софтверско процесирање на заглавјето. Позначајни промени се направени во поглед на фрагментирањето на податоците. Кај IPv6 фрагментирањето на податоците се прави кај изворот а не како кај IPv4 каде истото се врши во рутерот кој има ограничени можности во поглед на големината на пакетите. Целата информација за фрагментацијата е преместена од IP заглавието во IP продолженото заглавие (extension header) со цел да се поедностави протоколот и да се забрза процесирањето на датаграмите во рутерите.

Проверката на грешка на IP ниво е испуштена во IPv6 со цел да се намали процесирањето и да се подобри рутирањето. Проверката на грешка само би одземала време и бити од заглавието а воедно ќе воведе и редундантност бидејќи и даталинк и транспортното ниво вршат проверки на грешки (со заштитно кодирање на заглавијата што ги вметнуваат, соодветно секое од нивоата).

IPv6 е следна генерација на IP верзија која има многу подобрувања во однос на верзијата 4, меѓу другите е и поголемиот адресен простор од 128 бити во однос на 32 битното адресно поле на IPv4.

Зошто IPv6?

- Моменталниот адресен простор на IPv4 е многу мал. Потребен е поголем адресен простор.
- Големина на рутирачките табели. Бројот на мрежи во Интернет како и соодветните рутирачки табели расте многу брзо. Поголемиот адресен простор овозможува поголема флексибилност за да се создадат хиерархии со цел да се акумулираат повеќе ниско нивоски пристапи во еден повисоко нивоски пристап.
- Едноставност. Многу работи околу IPv4 изгледаат комплексно. Фрагментирањето, процесирањето се само два примера.
- Перформанси. Големите рутирачки табели, фрагментацијата како и опциоините полиња придонесуваат за поголема латентност и доцнења во мрежата.
- Flow Label. Изворот може да ги лабелира пакетите кои побаруваат посебни услуги а истите се разгледуваат во intermediate рутерите. Ова го поедноставува процесот на класификација на пакетите во секој рутер.
- Безбедност – повеќе опции во самиот стандард за IPv6.
- Екстензибилност. Ова е особина на додавање на нови функции и карактеристики без да се бара корегирање на постоечкиот протокол.
- Коегзистирање и преведување на IPv4. Може со сигурност да се каже дека IPv4 уредите и апликациите ќе бидат со нас засекогаш (т.е. уште доста години). Тоа значи дека секоја нова верзија на IP (значи, и IPv6) мора да коегзистира со IPv4.

Значи IPv6 е нова IP верзија која не е радикално различна од моменталната IPv4. На мрежите се уште им се доделуваат мрежни адреси или префикси, IPv6 ги препраќа пакетите на истиот начин, неконекциски ориентирано по принципот делница по делница (делница е линк кој поврзува два соседни рутери или хост и рутер), IPv6 рутерите се уште ги користат рутирачките протоколи, мрежниот уред мора да биде исконфигуриран со валидна IPv6 адреса итн. На IPv6 треба да се гледа како на поедноставна, поскалабилна и поефикасна верзија на IP.

2.2.2 Избор помеѓу IPv6 или IPv4

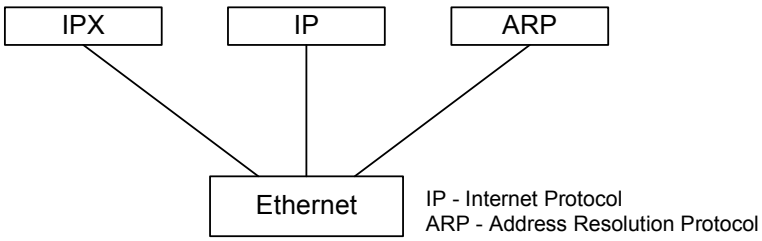
IPv6 нема преку ноќ да го замени IPv4 протоколот. Двете верзии ќе мора да коегзистираат подолго време. Во принцип IPv6 може да биде имплементирана како софтверска надоградба на постоечката IPv4 опрема овозможувајќи подолга транзициона фаза (Simple Internet Transition - SIT) а со тоа и намалување на трошоците за нова опрема штитејќи ги на тој начин претходните вложувања. Не е можно да се каже дали сите Интернет оператори ќе преминат на IPv6 технологија, тоа практично зависи од бенифициите кои би ги добил операторот. Сега засега повеќето оператори се опкружени со IPv4 рутери и најголемиот дел од сообраќајот треба де се адаптира на IPv4 мрежите поради што интересот за промена е мал. Новите plug-and-play карактеристики кои ги прават IPv6 мрежите многу поедноставни за конфигурирање и одржување од IPv4 мрежите можат да бидат атрактивни за некои оператори чија инфраструктура брзо еволуира. Со цел да се поедностави преодот кон IPv6 важно е постоечките IPv4 апликации да бидат способни да работат и со IPv6 апликации (пр. производителот на Интернет пребарувачи треба да им овозможи на клиентите да комуницираат со двете IP верзии). Битен предуслов за кооперативност е хостовите кои користат IPv6 протокол да користат дуални протоколни стекови (stacks), еден за IPv4 протоколниот stack и еден за IPv6 протоколниот stack. Краткорочно гледано можеме да заклучиме дека воведувањето на IPv6 го прави изборот на IP протокол посложен, а бенифициите во поглед на поефикасното рутирање ќе зависат од тоа дали операторите ќе извршат ориентирање кон IPv6. Долгорочно гледано не постои никаква дилема. IPv6 е покомплетен и поефикасен протокол отколку IPv4 кој може да го истисне штом апликациите ќе се развиваат така да ги искористуваат предностите на IPv6 како што е пример QoS поддршката. Преминот од IPv4 кон IPv6 станува посебно интересен во втората деценија на 21-от век бидејќи IPv4 адресниот простор е веќе "потрошен", така што транзицијата од IPv4 кон IPv6 е неминовна и зема се посилен замав. Но, реално е да се очекува дека во период од деценија-две ќе коегзистираат IPv4 и IPv6 така што учеството на IPv6 адресниот простор се повеќе ќе се зголемува и ќе го надмине IPv4 адресниот простор по број на IP адреси, за на крајот целосно и да го истисне.

2.3 Мултиплексирање во IP протоколниот стек

Има повеќе нивоа на мултиплексирање на протоколи во IP мрежите, [9]. Има неколку причини за тоа, т.е.:

- Протоколите на ниво линк се поврзани со повеќе мрежни протоколи
- Мрежните протоколи се поврзани со повеќе транспортни протоколи
- Транспортните протоколи се поврзани со повеќе апликации

Ќе се задржиме на IP мултиплексирањето, бидејќи сите IP пакетски мрежи мора да го користат IP протоколот, додека линк протоколите зависат од типот на линкот (на пример, Ethernet, ATM, SDH итн.), а исто така има и повеќе транспортни протоколи (на пример: TCP, UDP, RTP) и повеќе типови на апликации кои можат да се користат во Интернет (World Wide Web – WWW, e-mail, FTP, streaming итн.). Се разбира, има и повеќе протоколи на мрежно ниво, но денес доминантен е IP протоколот (го има во секој компјутер или терминал закачен за Интернет – жично или безжично, и во секој рутер или пакетски комутатор во Интернет мрежата).



Слика 2.3 Ethernet мултиплексирање

Ethernet мултиплексирање

Под Ethernet мултиплексирање подразбираме мултиплексирање на сообраќајот од мрежното ниво (т.е. различните мрежни протоколи) кон Ethernet мрежата, и во обратна насока демултиплексирање на сообраќајот. Еден пример на Ethernet мултиплексирање/демултиплексирање е прикажан на Слика 2.3.

Како Ethernet одредува дали пакетот што ќе пристигне до хостот (компјутерот што е закачен на Ethernet мрежата) е наменет за IP или за ARP протоколот, или за некој друг? Тоа го прави со користење на 2 бајта, наречени Ethertype Field, кои се дел од Ethernet рамката. На пример, вредноста за ARP во тоа поле од два бајта е 2054 (во декаден броен систем), а за IP е 2048 декадно (или 0800 хексадецимално).

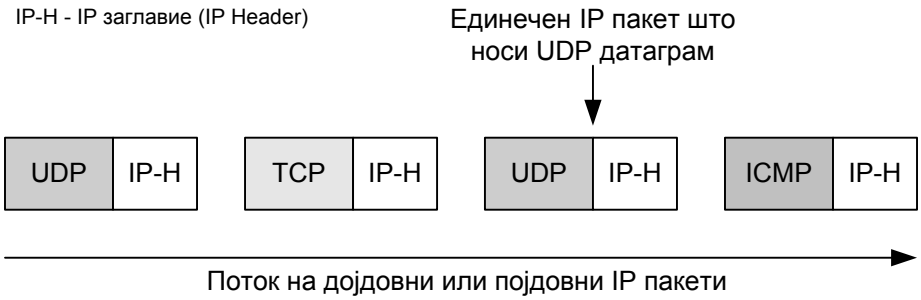
IP мултиплексирање

IP пакетите можат да пренесуваат различни содржини во нивните податочни полиња, како на пример:

- TCP (Transmission Control Protocol) сегменти
- UDP (User Datagram Protocol) сегменти
- ICMP (Internet Control Message Protocol) пораки итн.

Сите содржини кои се пренесуваат со IP пакети се сместуваат во податочниот дел на пакетот, како што е прикажано на Слика 2.4.

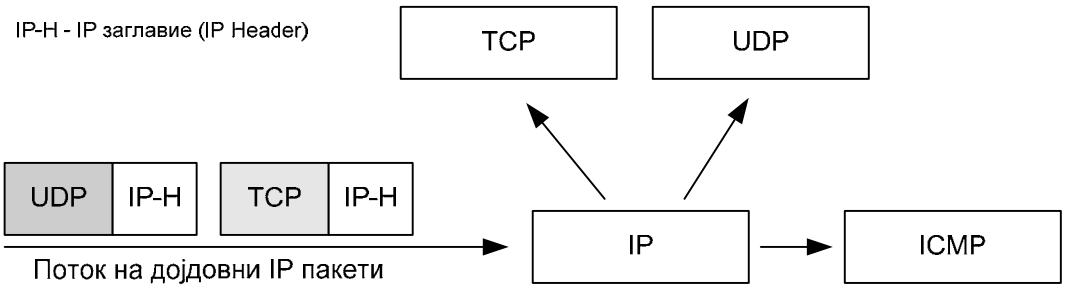
Слика 2.4 IP пакет



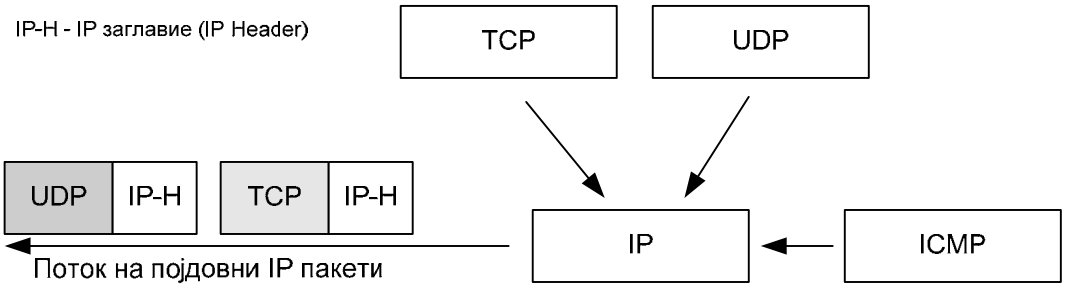
Слика 2.5 Поток на IP пакети

На пакетот секогаш му се додава IP заглавие при мултиплексирањето кон подолното ниво на линк (на пример: Ethernet), како што е покажано на Слика 2.5. На тој начин, содржините кои доаѓаат од различни апликации и транспортни протоколи се сместуваат во единствени IP пакети, кои понатаму се упатуваат на ист излезен линк и се пренесуваат по исти правила низ Интернет мрежата.

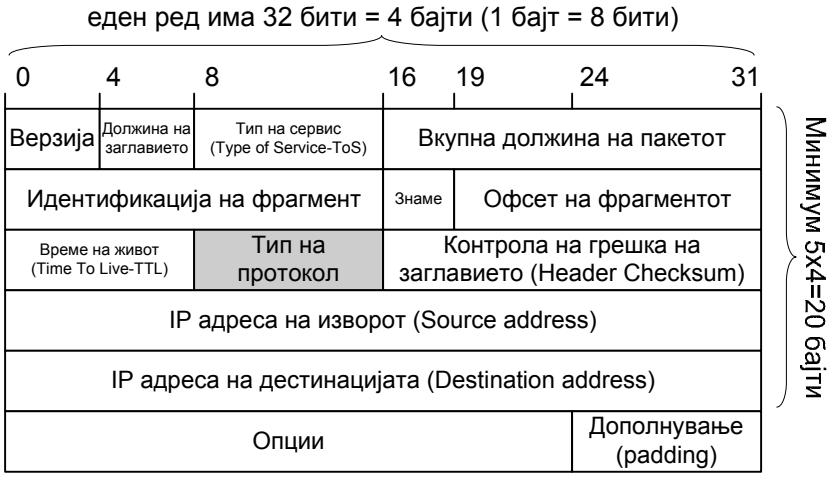
Демултиплексирањето се врши во насока од IP нивото кон транспортните протоколи (на пример: TCP, UDP, ICMP итн). При демултиплексирањето IP процесот мора да ја испрати содржината на дојдовните IP пакети до соодветниот процес за таа содржина (Слика 2.6). При мултиплексирањето IP процесот мора да ги прифати пораките од повеќе процеси и да ги мултиплексира во излезниот поток (Слика 2.7).



Слика 2.6 Демултиплексирање на дојдовен IP поток



Слика 2.7 Мултиплексирање на IP пакети во појдовен поток



Слика 2.8 Изглед на заглавие на IP пакет (IP верзија 4 – IPv4)

Се поставува прашањето како IP процесот одредува до кој транспортен протокол треба да ја достави содржината од податочниот дел на IP пакетот (по отстранување на IP заглавието)? За таа цел се користи посебно поле во заглавието на IP пакетите кое се нарекува “Протокол” и има должина од 8 бити (Слика 2.8). Полето “Протокол” служи за идентификација на содржината на податочниот дел на IP пакетите. На пример, ако

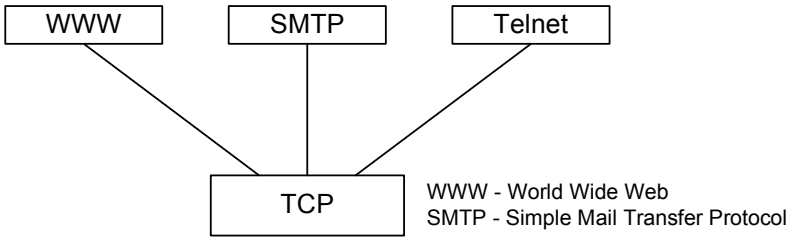
содржината на IP пакетот е наменета за ICMP протоколот, тогаш во полето “Протокол” од IP заглавието на тој пакет ќе биде запишана вредноста 1; ако содржината е наменета за TCP, тогаш вредноста во полето ќе биде 6; ако содржината е наменета за UDP, тогаш вредноста ќе биде 17 итн. (Табела 2.1).

Табела 2.1 Поважни вредности на полето “Протокол” во IP заглавието

Декадна вредност на “протокол” полето	**Протокол од транспортно ниво**
0	Резервирано
1	Internet Control Message Protocol (ICMP)
2	Internet Group Management Protocol (IGMP)
4	IP in IP encapsulation
5	Stream IP
6	Transmission Control Protocol (TCP)
17	User Datagram Protocol (UDP)

Мултиплексирање на транспортно протоколно ниво

Кога TCP или UDP (или некој трет) протоколен модул ќе прими пакет од IP нивото (како стигнуваат до нив видовме претходно при опис на IP мултиплексирањето), тој треба да ги раздели пакетите кои треба да бидат процесирани од различни апликации (на пример: FTP, HTTP или web-апликацијата, SMTP, SNTP итн.). TCP и UDP протоколните модули го прават тоа со помош на поле во нивните заглавија во кое се содржи 16-битен број на порта за дадените пакети (16-bit port number). Бидејќи, TCP и UDP протоколите се два различни протоколи, следствено и нивните броеви на порти се наоѓаат во различни адресни простори (во овој случај, под адресен простор на порти ги подразбираме сите броеви на порти кои можат да бидат доделени на даден протокол). Меѓутоа, некои апликации имаат исти порти и за TCP и за UDP. Тоа е во случај кога дадените апликации (сервиси) можат да се користат и преку едниот и преку другиот протокол (т.е. и преку TCP и преку UDP).



Слика 2.9 TCP мултиплексирање/демултиплексирање

Број на изворна порта - Source Port Number (16 бити)			Број на дестинациска порта - Destination Port Number (16 бити)
Број на секвенца - Sequence Number (32 бити)			
Број на потврда- Acknowledgement Number (32 бити)			
Должина (4 бити)	Резервирани (6 бити)	Знаменца (6 бити)	Големина на прозорецот - Window Size (16 бити)

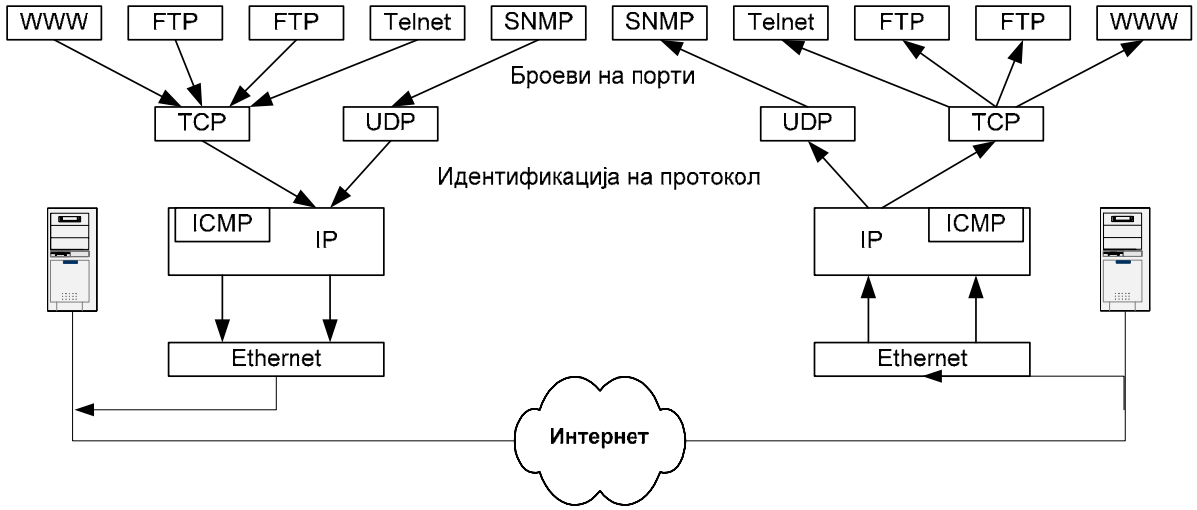
Слика 2.10 Изглед на заглавие на TCP пакет

Ќе го разгледаме подетално TCP мултиплексирањето/демултиплексирањето (Слика 2.9). На Слика 2.10 е прикажан изгледот на заглавието на еден TCP пакет. Може да се забележи дека има два броја на порти, изворна и дестинациска порта. Под изворна порта се подразбира бројот на портата преку која се врши TCP мултиплексирањето во насока од апликацијата кон TCP протоколот, а под дестинациска порта се подразбира бројот на портата преку која се врши TCP демултиплексирањето на приемната страна (т.е. во приемниот хост).

Што е “број на порта”? Имено, транспортното ниво врши мултиплексирање во кое повеќе софтверски елементи (по OSI терминологијата, софтверските елементи се нарекуваат протоколи) делат иста мрежна адреса (во IP мрежите, т.е. Интернет, тоа е IP адресата). За да може на единствен начин да се идентификуваат тие софтверски елементи (т.е. апликации), потребен е начин за нивно адресирање во транспортното ниво. Овие адреси се нарекуваат и “транспортни адреси”, а во TCP/IP мрежите за транспортните адреси се користи називот “броеви на порти” (port numbers). Во Табела 2.2 се дадени некои поважни броеви на порти кои ги користи TCP протоколот.

Табела 2.2 Поважни броеви на порти кои се користат за TCP мултиплексирање/демултиплексирање

Број на порта	Апликација
20	FTP data port
21	FTP control port
23	Telnet
25	SMTP
80	World Wide Web (HTTP)



Слика 2.11 Мултиплексирање/демултиплексирање на протоколи

На слика 2.11 е прикажан целиот пат од апликацијата на предавателната страна (лево на сликата), процесот на мултиплексирање на транспортно ниво, IP мултиплексирање и Ethernet мултиплексирање, преносот на информациите низ Интернет, и демултиплексирањето во приемниот хост (десно на сликата) кое се врши по обратен редослед од мултиплексирањето.

2.4 IP адресирање (IPv4)

Секој хост на Интернет има своја IP адреса, која во себе го содржи бројот на мрежата (network number) и бројот на хостот (host number) во таа мрежа, и е со вкупна должина од 32 бита (за IPv4, која ќе се подразбира понатаму под поимот IP). Нивната комбинација на еднозначен начин врши идентификација на дадениот компјутер. Рутерите се компјутери кои се наоѓаат на границите меѓу повеќе мрежи, па затоа располагаат со повеќе IP адреси: по една за секоја од тие мрежи.

Постојат два начини на IP адресирање:

- адресирање со класи (Classfull), и
- адресирање без класи (Classless).

При IP адресирање со класи се дефинирани 5 различни класи на IP адреси, прикажани на слика 2.12. IP адресата од класа A дозволува постоење на 126 различни мрежи со 16 милиони хостови; класата B- 16382 мрежи со 65534 хостови; класата C- 2 милиони мрежи со по 254 хостови.

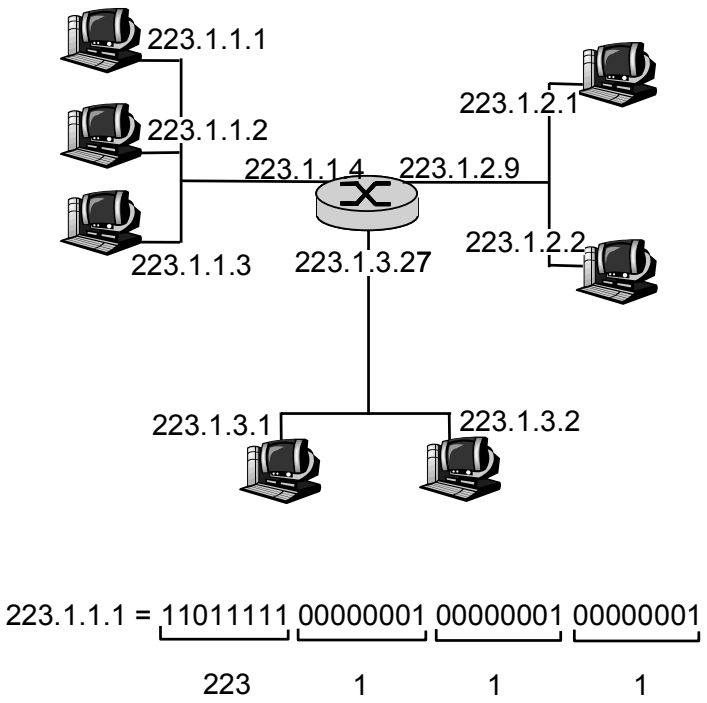
Слика 2.12 Класи на IP адреси

Класата D е наменета за мултикаст адреси. Покрај единствената IP адреса, некој хост може да располага и со една или повеќе мултикаст адреси од класата D. Притоа

една мултикаст адреса се доделува на различни хостови. Секој датаграм што содржи некоја мултикаст дестинациона адреса, истовремено се испорачува до сите хостови на кои им е доделена таа мултикаст адреса. Адресите од класа Е се резервирани за идна употреба.

IP адресите ги доделува NIC (Network Information Center). На секоја нова мрежа која бара пристап на Интернет, NIC доделува по еден мрежен број, а тоа всушност се група од IP адреси со која операторот на мрежата може слободно да располага.

IP адресите се 4 бајтни, па затоа обично IP адресата се претставува во вид на 4 децимални броеви раздвоени со точка како на пример мрежниот број 192.168.1.1. Пример на адресирање во Интернет е даден на слика 2.13, каде што три IP мрежи се поврзани меѓусебе со еден рутер.



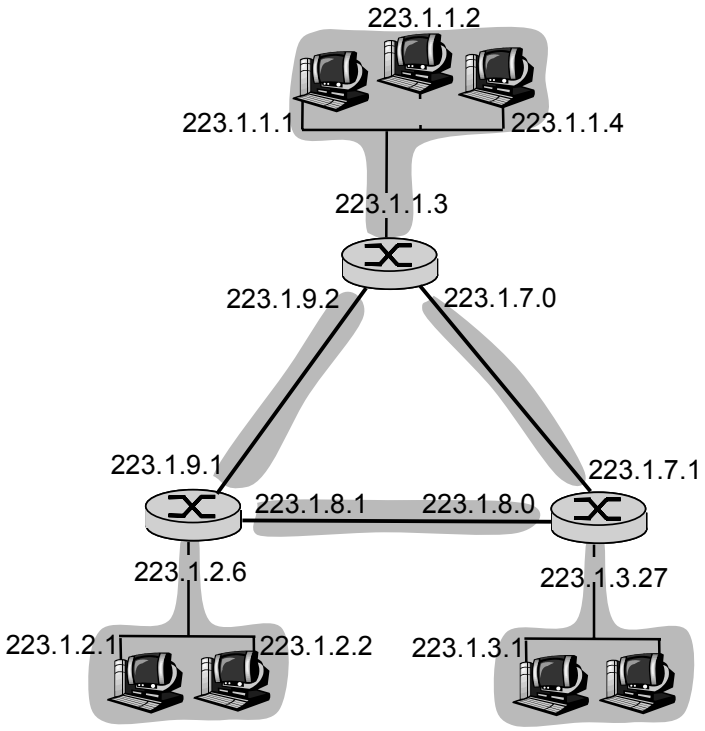
Слика 2.13 Доделување на IP адреси на хостови приклучени на различни IP мрежи (сегменти) кои се поврзани преку еден рутер

Адресирањето без класи се врши со користење на т.н. маска која ги има сетирано на вредност “1” сите бити кои се однесуваат на мрежниот дел на IP адресите во дадената IP мрежа на која се однесува таа маска. На пример, IP мрежата 192.168.1.0/255.255.255.252 дефинира IP мрежа која има само две IP адреси 192.168.1.253 и 192.168.1.254 (IP адресата кај која последниот бајт е 255 т.е. сите “1”-ци

е наменета за broadcast и не може да се доделува на мрежни интерфејси на хостови или рутери). Каде би ни користела ваква мрежа со две IP адреси? Тоа би можело да се искористи ако имаме линк кој поврзува два рутера, бидејќи секој од двата краја (секој од интерфејсите на секој од рутерите на кој завршува по еден крај од линкот) има потреба од IP адреса, а два мрежни интерфејси на двата рутери кон линкот што ги поврзува ќе формираат една мрежа. Секако, ова е специфичен случај, обично пристапна IP мрежа (т.е. сегмент) има и хостови, па бројот на IP адреси во мрежата е поголем. Постои уште еден начин на прикажување на IP адресите без класи каде што маската е заменета со бројка која го означува бројот на бити "1"-ци во маската од лево на десно, или во разгледуваниот пример IP мрежата може да се запише дека има адресен простор 192.168.1.0/30, што значи дека првите 30 бити од IP адресата се однесуваат на мрежата, а преостанатите на делот за хостот.

IP рутирање и подмрежување

Мрежите кои припаѓаат на иста мрежа се со IP адреси со ист мрежен број т.е иста класа (пример, слика 2.14). Ако некоја организација од NIC добие една група IP адреси од класата C тоа значи дека вкупниот број на компјутери кои можат да се приклучат на Интернет е 254. Таквата локална мрежа може и да порасне над 254 компјутери па тие нема да можат да имаат пристап. Тогаш се јавува потреба од друга група на IP адреси. Тоа значи дека на крајот може да се јават повеќе LAN мрежи секоја со по еден рутер и со свој мрежен број. Но оваа ситуација е прилично сложена од аспект на нивно одржување и администрирање, а набавувањето на нов број од NIC бара и негово дистрибуирање по околните рутери, затоа решението е во постоење на една поголема мрежа која ќе биде поделена на повеќе помали т.н. подмрежи за внатрешна употреба а сепак за надворешниот свет тоа ќе биде единствена мрежа. За таа цел оваа организација наместо C треба да обезбеди мрежен број од класата B. Притоа 16-битниот host number внатре во мрежата може да биде произволно организиран. Надвор од мрежата, подмрежите се невидливи така што формирањето на нова подмрежа не бара контакт со NIC или менување на рутирачките табели во околните рутери.



Слика 2.14 Пример IP адресирање во случај кога имаме повеќе пристапни мрежи и скелетна мрежа од рутери поврзани со линкови (на сликата има 6 IP мрежи)

Како хостот може да добие IP адреса?

Секој хост мора има IP адреса на секој интерфејс преку кој комуницира. Постојат два начина за доделување на IP адреси:

- Статички IP адреси (hard-coded) доделени од страна на администраторот на системот во фајл:
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: динамички се доделува адресата од сервер (“plug-and-play”), ќе го анализираме подетално во продолжение.

Пример за распределба на блокови на адреси кои се доделени на даден ISP:

ISP блок адреси	**11001000 00010111 00010000**	**00000000**	**200.23.16.0/20**
Организација 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Организација 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Организација 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	…..	….	**. . . .**
Организација 7	11001000 00010111 00011110	00000000	200.23.30.0/23

Како мрежата го добива мрежниот дел од IP адресата?

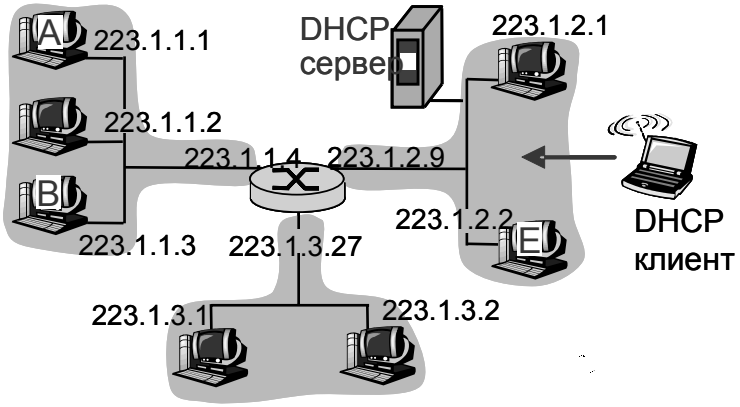
Го добива од доделените адреси од адресниот простор на ISP (Internet Service Provider).

Како ISP добива блок од IP адреси?

Преку ICANN (Internet Corporation for Assigned Names and Numbers), www.icann.org. Притоа, не сите ISP-а контактираат со ICANN директно. Имено, најчесто помалите ISP-а добиваат блокови адреси од поголемите ISP-а. Само најголемите ISP-а, кои најчесто се интернационални, комуницираат директно со ICANN за доделување на IP адреси.

2.5 DHCP (Dynamic Host Configuration Protocol)

Dynamic Host Configuration Protocol (DHCP), споецифициран со RFC 2131, [10], обезбедува дистрибуција на конфигурациски параметри наменети за Интернет корисниците (hosts). DHCP се состои од две компоненти: протокол кој се грижи за доставување на специфичните кориснички конфигурациски параметри од DHCP серверот до корисникот, и механизам за алоцирање на мрежни адреси на корисниците.



Слика 2.15 Имплементација на DHCP сервер во ISP мрежа

Како модел на работа на DHCP е избран клиент-сервер моделот (слика 2.15). Со тој модел DHCP серверот на динамички конфигурабилните корисници (клиенти) им доставува конфигурациски параметри кои претходно ги алоцира во својата база. DHCP поддржува три механизми за алокација на параметри и тоа: автоматска алокација, динамичка и мануелна алокација.

Автоматската алокација на адреси се применува за автоматско доделување на перманентни (трајни) адреси на корисниците кои се приклучуваат на мрежата. При динамичко доделување, адресата на корисникот која му се доделува е со ограничено време на користење (lease time), односно корисникот може да ја користи во временскиот интервал додека не истече доделеното време или додека експлицитно клиентот не се откаже од нејзино користење. Кај мануелната алокација на адреси, адресите всушност се доделуваат од страна на администраторот на мрежата , а DHCP се користи замо за нивно дистрибуирање и одржување во рамките на мрежата. Динамичката алокација на адреси преставува единствен од трите механизми кој овозможува реискористување на адресниот простор. Односно адресите кои не се користат на дадениот мрежен сегмент од ниеден клиент или се ослободени од клиентот после истекот на дозволеното време се сметаат за слободни и се доделуваат на некој клиент кој од DHCP серверот ќе побара адреса.

Форматот на DHCP пораките се базира на BOOTP пораките, со цел да се симулира однесувањето на BOOTP relay agent (агентите за премостување на мрежните сегменти) за да се овозможи интероперативност помеѓу постоечките BOOTP клиенти со DHCP серверите. Користењето на BOOTP relay agent (кој во верзијата на DHCP е дефиниран како DHCP relay agent) ја елиминра потребата од постоење на DHCP сервер на сите сегменти од мрежата.

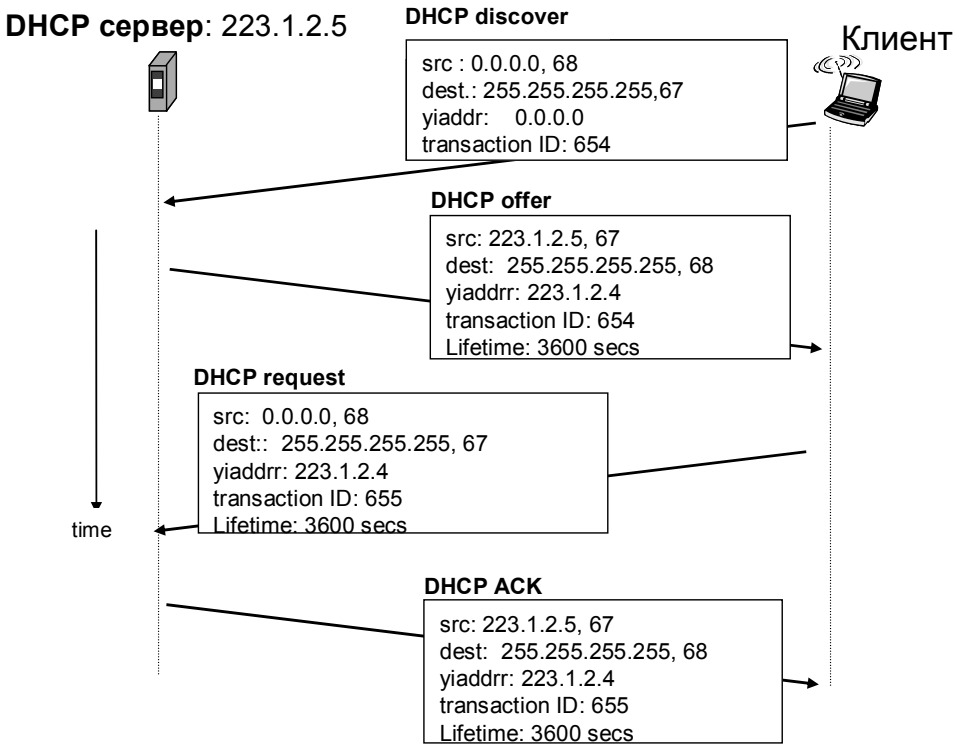
Од аспект на клиентот DHCP преставува екстензија на BOOTP механизмот. Ова однесување овозможува да постоечките BOOTP клиенти колаборираат со DHCP серверите без дополнителни промени во клиентскиот софтвер. Начинот на нивната колаборација е опишан во RFC 1542.

<u>Цел на DHCP:</u> да се овозможи динамичко доделување на IP адресите од мрежен сервер кога клиентот се приклучува кон мрежата.

Има четири типа на DHCP пораки:

- Хостот емитира “DHCP discover” порака
- DHCP серверот одговара со “DHCP offer” порака
- Хостот бара IP адреса со “DHCP request” порака
- DHCP серверот ја праќа IP адресата со “DHCP ack” порака

Користењето на одделните DHCP пораки е прикажано на слика 2.16.



Слика 2.16 Пример на клиент-сервер комуникација со DHCP

op (1)	htype (1)	hlen (1)	Hops (1)
xid (4)			
secs (2)		flags (2)	
ciaddr (4)			
yiaddr (4)			
siaddr (4)			
giaddr (4)			
chaddr (16)			
sname (64)			
file (128)			
options (variable)			

Слика 2.17 Изглед на DHCP порака

Изгледот на DHCP порака е прикажана на слика 2.17. Полињата во рамките на пакетот се дефинирани на следниов начин:

- op - го дефинира типот на пораката (1–BOOTREQUEST, 2–BOOTREPLAY)
- htype – тип на хардверска адреса пр. “1” за Ethernet
- hops – должина на хардверската адреса пр. “6” за Ethernet
- xid - Идентификациски број на трансакцијата (број кој се користи меѓу клиентот и серверот за подредување на пораките при меѓусебната комуникација)
- secs – го пополнува клиентот и означува колку секунди се поминати од почетокот на барањето за обновување или добивање на адреса.
- flags – ознаки за дедефинирање на пораката.
- ciaddr – клиентска IP адреса - ова поле е пополнето само доколку клиентот е во осостојба на повторно врзување или обновување и е во состојба да одговара на ARP повици.
- yiaddr – сопствената IP аддреса
- siaddr – IP адреса на серверот кој треба да се користи , се добива како резултат на доделување на IP адреса (DHCPOFFER , односно DHCPACK од DHCP серверот).
- giaddr – IP адреса на Relay agent – от доколку се користи.
- chaddr – клиентска хардверска адреса.
- cname- опционо клиентско име на серверот, нул терминиран стринг.
- file- Boot file name, нул терминиран стринг, генеричко име или null во DHCPDISCOVER, или целосна патека за пристап во DHCPOFFER пакетот.
- options – поле со опциони параметри

DHCP како свој транспортен протокол го користи UDP и пораките од клиентот кон серверот се праќат преку порта 67, а пораките од серверот кон клиентот се праќаат преку порта 68. Постои дефинирано поле во DHCP пораките т.н. ‘server identifier’ кое се користи во два случаи и тоа: за да се идентификува DHCP серверот во DHCP пораката и како дестинациона адреса од клиентите кон серверите. DHCP серверот треба да утврди од кој сегмент во мрежата му доаѓа барањето за доделување на IP адреса и во зависност од тоа да му додели адреса и параметри кои се предвидени за користење на тој сегмент.

DHCP broadcast пораката од клиентот која се праќа пред тој да добие IP адреса, за изворна адреса треба да стави вредност еднаква на 0. Доколку полето 'giaddr' добиено

од клиентот не е еднакво на 0, DHCP серверот ги праќа своите повратни пораки на DHCP relay agent–от чија адреса е внесена во истото поле. Доколку тоа е еднакво на 0, а полето 'ciaddr' не е еднакво на 0, тогаш DHCP серверот праќа уникаст DHCPOFFER и DHCPACK пораки кон адресата означена во истото поле. И крајно доколку и во 'giaddr' и во 'ciaddr' вредностите се еднакви на 0 и broadcast битот во ‘flags’ полето е поставен на 0, тогаш DHCP серверот дифузно ги испраќа DHCPOFFER и DHCPACK пораките кон broadcast IP адресата 0xffffffff. Доколку broadcast битот е различен од нула, а останатите се еднакви на нула, тогаш серверот прави уникаст на DHCPOFFER и DHCPACK пораките кон клиентската хардверска адреса и 'yiaddr'. Во било кој случај кога 'giaddr' е еднаков на 0 серверот дифузно ги праќа DHCPNAK пораките на адреса 0xffffffff.

Да резимираме, DHCP може да се користи и преку рутер со користење на т.н. DHCP relay. На тој начин се овозможува да има еден DHCP сервер за повеќе мрежни сегменти (IP мрежи) што е чест случај во мрежите на Интернет сервис провајдерите.

2.6 Транспортни протоколи во интернет

Двата основни транспортни протоколи во рамките на Интернет мрежата се: TCP кој е конекциски ориентиран протокол и UDP кој е неконекциски ориентиран протокол.

TCP претставува конекциски ориентиран протокол бидејќи пред самиот пренос на корисни податоци претходи процедура на воспоставување на врска меѓу двата крајни корисници. Притоа повиканата страна може да го прифати или одбие повикот, а врз основа на бараните параметри за преносот кои ги поставил повикувачот. Кога преносот на корисните податоци ќе биде завршен следува постапка на раскинување на врската било од едната или од другата страна.

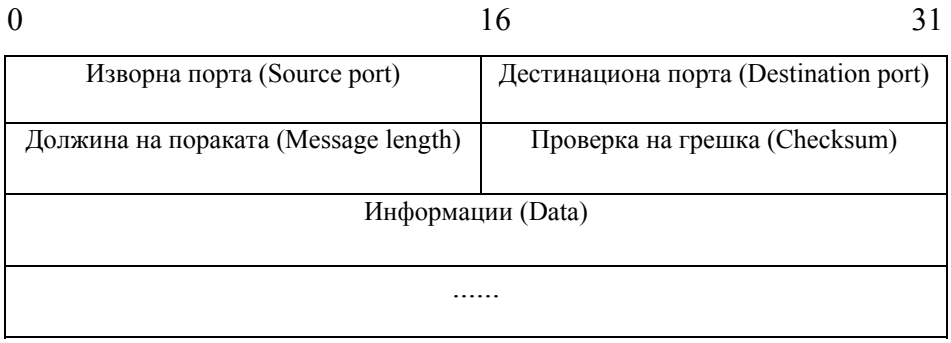
За разлика од TCP, UDP претставува неконекциски ориентиран протокол бидејќи корисникот веднаш започнува со праќање на корисната информација без да знае дали спротивната страна е на мрежа и дали воопшто ги прима податоците што другиот ги праќа. UDP се користи кога сакаме брзо и експедитивно да пренесеме податоци. Овој протокол е едноставен и додава кусо заглавие пред корисната информација која ја добива од својата апликација по што ја проследува до IP нивото.

2.6.1 Карактеристики на UDP (User Datagram Protocol)

UDP претставува неконекциски транспортен протокол (од четвртото ниво според OSI референтниот модел). Тој е интерфејс помеѓу IP и протоколите од повисоките нивоа. За разлика од TCP, UDP не обезбедува контрола на протокот, надежност и контрола на грешка на испратените податоци преку IP. Поради неговата едноставност UDP заглавјето се состои од помалку бајти во споредба со TCP.

UDP е корисен во случаи кога механизмот на доверливост од TCP не е неопходен како и во случаи кога протоколи од повисоките нивоа можат да обезбедат контрола на проток и грешка. Тој се користи како транспортен протокол за неколку добро познати апликациски протоколи (седмо ниво според OSI референтниот модел), како што се NFS, SNMP, DNS (Domain Name System) и TFTP (Trivial File Transfer Protocol), [9]. UDP овозможува:

- Недоверлив пренос на податоци, податоците може да бидат изгубени или дуплирани (нема повратна потврда кај овој протокол).
- Нема индикација за загушување на приемната страна. Имено, праќачот на пораките може да ги испраќа со поголема брзина од онаа што може да ја поднесе примачот на пораките.
- Нема сигнализација меѓу изворот и дестинацијата, нема претходно поврзување на крајните точки при комуникацијата како што е случај кај TCP.



Слика 2.18 Формат на UDP заглавието

UDP, всушност го користи IP, мрежниот протокол, обезбедувајќи го истиот сервис како и IP за пренос на пораки од еден хост до друг, но со една важна разлика, а тоа е можноста на UDP да разликува повеќе различни процеси на една иста машина благодарејќи на воведувањето на портите за протоколи (имено, секоја UDP порака ги содржи двете порти, изворната и дестинационата). На сликата 2.18 е прикажан форматот на UDP заглавието.

Полињата во рамките на UDP пакетот се дефинирани како:

- Source port и Destination port - содржат 16 битни броеви со кои се означува UDP портата, која се користи за мултиплексирање/демултиплексирање на датаграмите од процесите од апликациско ниво.
- Length - ја дефинира должината на UDP пакетот (заглавие + податоци).
- Checksum - овозможува проверка на грешка на UDP заглавјето и податоците.

2.6.2 TCP протокол (Transmission Control Protocol)

За доверлив пренос на информациите стандарден протокол во Интернет е TCP (Transmission Control Protocol), [3]. Најголем број од популарните сервиси денес се засноваат на TCP, [11], на пример: www, FTP (File Transfer Protocol), Telnet итн. Мерењата на Интернет сообраќајот покажуваат дека најголем дел од сообраќајот се засновува на TCP протоколот, се разбира поставен над IP (види глава 1). Заради овие причини, во оваа глава подетално е објаснет TCP протоколот.

TCP е заснован на принципот на потврдување на успешно примените пакети од дестинацијата кон изворот на сообраќај. Во тој поглед, TCP припаѓа на групата ARQ (Acknowledgement Repeat reQuest) протоколи, каде што потврдите (ACK–Acknowledgement) се испраќаат по секој еден или повеќе успешно примени TCP сегменти во дестинацискиот јазол. Притоа броевите на секвенците кои се испраќаат во потврдите се засновани на бројот на примени бајти. TCP дава потврда која го содржи секвенцијалниот број на последниот успешно примен бајт по редослед (in-order). Основниот елемент во функционирањето на TCP е TCP сегментот (типична големина на еден TCP сегмент е 512 бајти). Сегментот претставува секвенца на бајти која се идентификува со 32-битни секвенцијални броеви за почетниот и последниот бајт на секвенцата. На овој начин дефиниран TCP овозможува доверлив пренос на IP пакетите.

Во продолжение се објаснети механизмите на TCP во услови на појава на загуби на пакети.

TCP протоколот е специјално дизајниран за да обезбеди надежен податочен проток меѓу два крајни корисници преку мрежа која не дава никакви гаранции за точноста на пренесуваните податоци. Интернет е составен од многу различни видови на мрежи од кои поголемиот дел се ненадежни.

Како функционира TCP?

TCP го прифаќа податочниот поток од апликацијата преку познат интерфејс и го дели на парчиња-сегменти кои потоа ги проследува до IP кој ги сместува во датаграми и ги проследува во мрежата.

IP нивото знаеме дека не дава никакви гаранции за тоа дали датаграмот ќе биде точно испорачан до дестинацијата, тоа е функција на TCP. Тој треба да обезбеди функции за потврди и тајмаути т.е да врши ретрансмисии ако е потребно. На пример, датаграмите на дестинацијата можат да пристигнуваат по погрешен редослед, па тогаш задачата на TCP ќе биде да ги реасемблира истите во правилната секвенца. TCP треба да обезбеди гаранции за точноста на пренесените податоци меѓу два крајни корисници.

Праќачкиот и приемниот TCP ентитет при една конекција разменуваат податоци во вид на сегменти. Еден сегмент се состои од фиксно 20 бајтно заглавие по кое следи корисната информација. Самиот TCP тековно одлучува за должината на сегментите која не смее да надмине дадена вредност. Постојат две ограничувања на големината на сегментот. Едното ограничување е вкупната должина на IP датаграмот која не смее да биде поголема од 64 KB (2^{16} бајти, бидејќи за големината на датаграмот има поле со 16 бити во заглавието на IP пакетот). Второто ограничување доаѓа од мрежната технологија под IP нивото, односно секоја мрежа си има ограничувања на максималната должина на даталинк рамката. Во пракса најчесто MTU (Maximum Transfer Unit) е со должина од неколку илјади бајти. Долж својата патека датаграмот може да наиде на мрежа која е способна да пренесува само помали податочни парчиња, па рутерот на влезот во таа мрежа врши сегментација на датаграмот. Бидејќи сегментацијата бара на секој сегментиран дел да му се додели ново IP заглавие истата не е баш погодна бидејќи влијае на зголемување на редундансата во однос на пренесената корисна информација која пак предизвикува намалување на искористеноста на пропусниот опсег.

TCP протоколот е протокол со лизгачки прозорец

Во моментот кога праќачот ќе емитира еден сегмент TCP ентитетот врши доделување на реден број и стартува тајмер. Кога сегментот ќе стигне до дестинацијата приемниот TCP ентитет веднаш одговара со свој сегмент независно дали во тој момент има да прати корисни информации или не. Овој сегмент носи потврда во вид на acknowledgement number еднаков на редниот број на следниот бајт кој приемната страна очекува да го прими, кој се праќа преку т.н. ACK датаграм (т.е. пакет). Ако тајмерот на страната на праќачот истече се врши ретрансмисија на истиот сегмент.

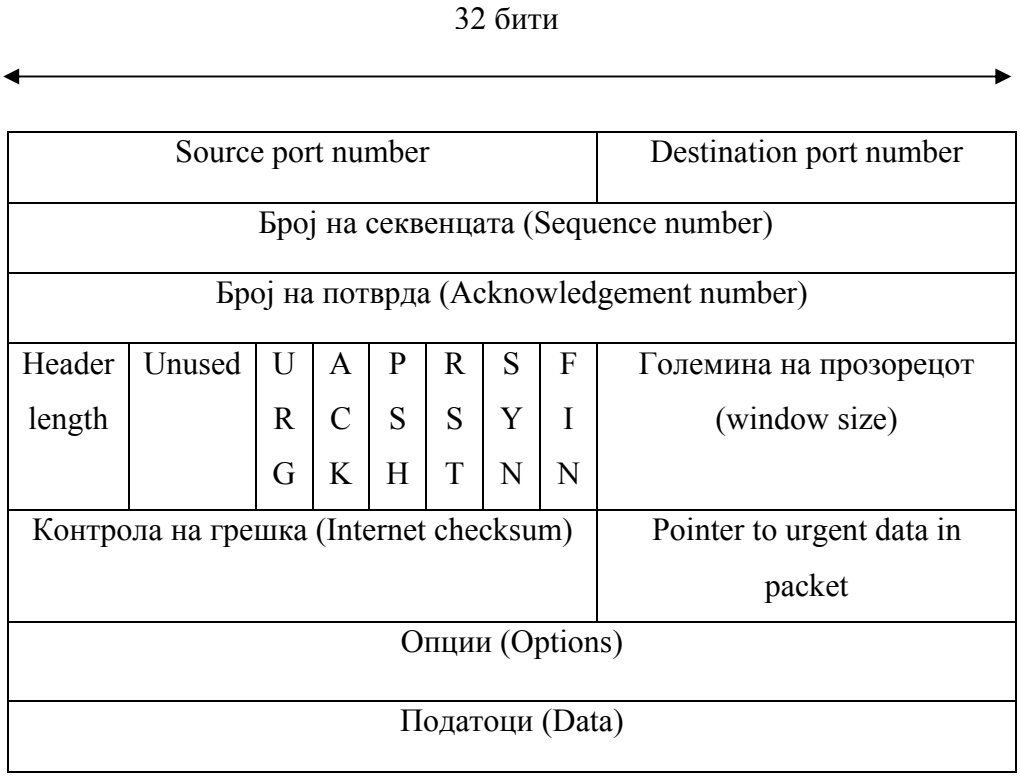
Бидејќи сегментите можат да бидат фрагментирани можно е еден дел од првобитниот сегмент да стаса до дестинацијата а остатокот да се загуби. Во овој случај оригиналниот датаграм нема да може да биде реасемблиран, приемникот не праќа позитивна потврда, па изворот мора да изврши препраќање на целиот сегмент.

Сегментите може да не пристигнуваат по правилен редослед, па една група на бајти нема да може да се потврди се додека не стаса и онаа која припаѓа на интервалот помеѓу.

Поради натрупување во некоја точка долж рутата (на пример: во баферите во попатните рутери) може да се случи да истече тајмаутот за испратениот сегмент во предавателот, па повторно да се испрати истиот што би можело да резултира во прием на дуплициран сегмент во дестинацијата бидејќи првиот сегмент иако со задоцнување можно е да пристигне до дестинацијата. За да може TCP да се справи и со овие проблеми дополнително е оптимизиран во одделни свои верзии.

Формат на заглавието на TCP сегментот

Секој TCP сегмент започнува со фиксно 20 бајтно заглавие. На него опционо можат да му се доделат повеќе опциони полиња. По можното опционо поле следуваат корисните податоци кои можат да бидат со максимална големина од 65495 бајти. Од 65535 се одземаат 20 бајти кои се од IP заглавието, потоа се одземаат 20 бајти од TCP сегментот и се добиваат 65495 бајти. Се користат и сегментите кои имаат нулти должини на дата полето како и контролни пораки и потврди.



Слика 2.19 Детален формат на TCP сегмент

Source port и Destination port полињата ги идентификуваат крајните точки (порти) меѓу кои е воспоставена TCP конекцијата (слика 2.19).

Sequence number и Acknowledgement number се 32 битни полиња со кои се врши подредување на испратената информација и пренос на потврди, соодветно. Со овие броеви се врши нумерирање на испратените бајти, а не на редниот број на испратениот сегмент. Поради ова броевите на секвенца на два соседни сегменти не се разликуваат за еден туку за бројот на бајти во инфо полето на првиот од двата соседни сегменти.

TCP header length полето кажува колку 32 битни зборови се содржани во TCP заглавието. Оваа информација е потребна бидејќи Options полето е со променлива големина.

Потоа следува 6 битно поле кое не се користи. По ова поле следат шест еднобитни полиња. URG е сетиран кога се користи Urgent поинтерот со кој се индицира бајт офсетот од тековниот реден број во дадениот сегмент каде се сместени ургентните податоци.

ACK битот се сетира на 1 за да покаже дека Acknowledgement полето е валидно и дека носи потврда. Ако е 0 кај дестинационата машина полето се игнорира.

PSH битот укажува на тоа дека со дадениот сегмент се пренесуваат PUSH податоци. Со него од приемникот се бара да ги испорача податоците на апликацијата со самото нивно пристигнување, без да врши нивно баферирање.

Со RST битот се врши ресетирање на некоја проблематична конекција. Приемот на овој бит може да значи дека спротивната страна одбива да го прифати барањето за воспоставување на конекција. Кога овој бит е сетиран тоа значи дека има некаков проблем.

SYN битот се користи за воспоставување на конекции заедно со ACK битот. Кога некој хост бара воспоставување на врска кон спротивната страна испраќа сегмент со SYN=1 и ACK=0. Доколку хостот ја прифати сесијата како одговор ќе испрати сегмент со SYN=1 и ACK=1.

FIN битот служи за раскинување на TCP конекцијата. Тој се сетира кога едната страна веќе нема податоци за праќање па така ја информира спротивната страна за тоа. Конекцијата е целосно терминирана кога и спротивната страна ќе испрати FIN=1.

Контролата на проток во TCP се остварува со користење на лизгачки прозорец со променлива должина. Со Window size полето приемната страна му укажува на праќачот колку бајти може да испрати почнувајќи од оние кои се потврдени со Acknowledgement number полето. Доколку предавателот прими сегмент со вредност на window size полето 0, тогаш привремено запира со праќањето на податоци се до моментот кога ќе прими сегмент со Window size ≠ 0. Тоа би значело дозвола од приемникот да се продолжи со испраќање на податоците, почнувајќи од бајтот идентификуван со Acknowledgement number во истиот сегмент.

Checksum полето обезбедува дополнителна сигурност при откривање на грешки меѓутоа за разлика од IP checksum врши заштитно кодирање на целиот сегмент.

Со Options полето се обезбедуваат дополнителни функции. Со него се специфицира максималната должина на инфо полето во сегментот. Користењето на поголеми сегменти е подобро отколку користењето на помали. Поради можноста од грешки. Големината на сегментите се договара за време на воспоставувањето на врска при што секоја страна може да ја најави својата максимална должина на инфо полето. Секогаш се усвојува помалата од двете вредности. Доколку не се користи оваа можност тогаш големината на инфо полето има вообичаена вредност од 536 бајти.

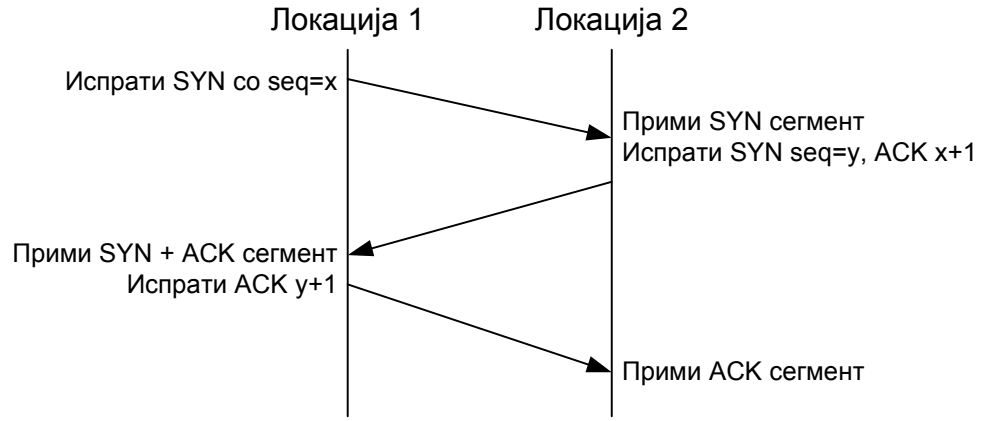
Воспоставување и раскинување на TCP конекција

Пред воспоставувањето на врска серверот очекува дојдовна врска. Тој се сетирал во таа состојба откако се извршила LISTEN процедурата на неговата страна. Клиентот извршувајќи ја CONNECT процедурата кон серверот испраќа еден TCP сегмент со SYN=1 и ACK=0. Со овој прв сегмент може да се пренесе некоја корисна информација како на пример лозинка (password) за пристап до серверот. На Sequence number полето притоа му се доделува некоја произволна вредност, на пример X.

Кога испратениот сегмент ќе стигне до дестинацијата се испорачува до соодветниот сокет (за сокетите ќе стане збор во глава 3 од оваа книга). Барањето за конекција од приемната страна може или да се прифати или да се отфрли. Доколку серверот ја прифати конекцијата испраќа во спротивна насока сегмент за потврда со вредности SYN=1 и ACK=1. Acknwledgement number полето тогаш добива вредност X+1 бидејќи така се потврдува примениот SYN сегмент. Sequence number полето добива некоја произволна вредност Y.

Кога испратениот од серверот сегмент ќе пристигне кај клиентот се потврдува со испраќање на сегмент со Sequence number = X+1 и Ackowledgement number = Y+1.

Вака се одвива сценариото на воспоставување на една нормална TCP конекција и се нарекува three-way handshake (слика 2.20).



Слика 2.20 Воспоставување на TCP конекција

Можно е двата хоста истовремено еден кон друг да се обидуваат да воспостават врска. До ова не може да дојде, т.е. не може да дојде до воспоставување на две конекции бидејќи секоја конекција се идентификува со нејзините крајни порти и мрежните адреси. Во случајов и двете конекции ќе имаат ист пар на порти.

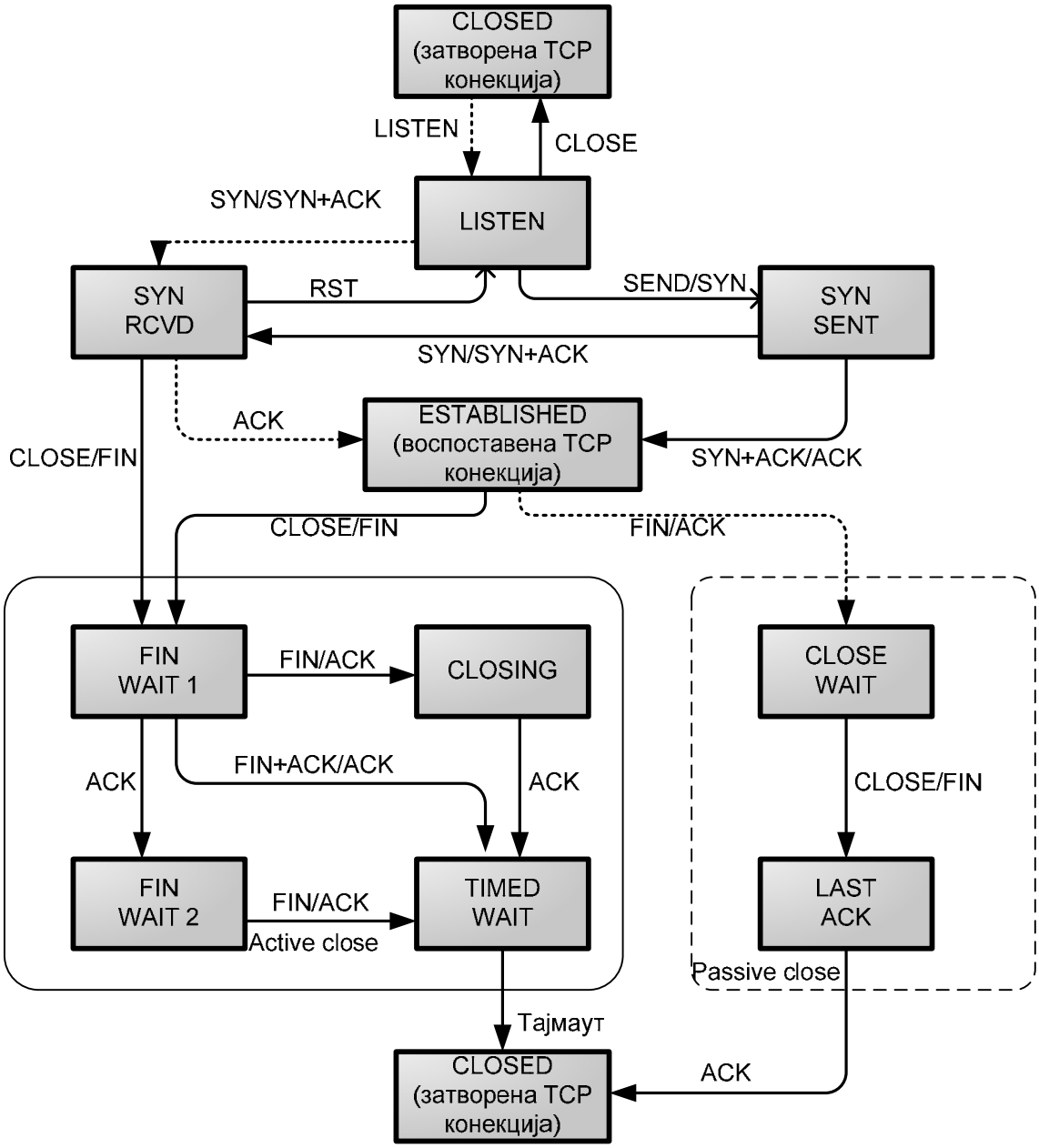
TCP конекцијата е full-duplex но при нејзиното раскинување се однесува како две еднонасочни simplex врски. Споменавме дека доколку едната страна нема податоци за испраќање дека ќе испрати FIN сегмент со вредности FIN=1 и ACK=0. Откако ќе биде потврден на спротивната страна со FIN=1 и ACK=1 врската во таа насока ќе се смета за терминирана но спротивната страна може да продолжи со испраќање на податоци кои другата ќе ги прими. Кога врската ќе се раскине и во другата насока конекцијата ќе се смета за дефинитивно терминирана. Доколку двете страни истовремено иницираат раскинување тоа ќе се одвива независно во двете насоки на начин кој беше споменат.

При испраќањето на FIN сегментот се активира посебен тајмер за доколку дојде до губење на овој сегмент долж неговата патека врската по истекувањето на тајмерот да се терминира. Кога другата страна ќе забележи дека спротивната страна е подолго време неактивна самата ќе го терминира сокетот во приемна насока.

Пред секоја конекција сокетот се наоѓа во состојба CLOSED (дијаграмот на состојби за TCP е прикажан на слика 2.21). Кога серверот ќе ја прими примитивата LISTEN ја напушта состојбата CLOSED. Кога клиентот ќе ја иницира CONNECT примитивата, преминува во состојба SYN RCVD и како одговор го праќа SYN+ACK сегментот. Откако овој сегмент ќе биде коректно примен клиент сокетот преминува во состојба ESTABLISHED. Се додека клиентот и серверот разменуваат податоци и меѓусебно комуницираат тие остануваат во оваа состојба.

Кога клиентот ќе сака да ја прекине комуникацијата со серверот ја иницира CLOSE примитивата по која сокетот го праќа FIN сегментот и влегува во Active close секвенцата. Првата состојба во која преминува е FIN WAIT 1. Кога спротивната страна ќе го прими FIN сегментот ја започнува Passive close секвенцата т.е преминува во состојбата Close Wait и праќа потврда. Кога потврдата ќе стигне до клиентот тој преминува во Fin Wait 2 по што конекцијата од клиентот кон серверот се смета за затворена што не е случај во спротивна насока сервер-клиент.

Кога серверот ќе ја иницира CLOSE примитивата ќе го испрати FIN сегментот и ќе премине во состојба LAST ACK. По приемот на FIN сегментот клиентот праќа потврда ACK и преминува во TIMED WAIT состојбата. На ACK серверот преминува во CLOSED состојбата. Во оваа состојба преминува и клиентот по истекот на тајмаутот.



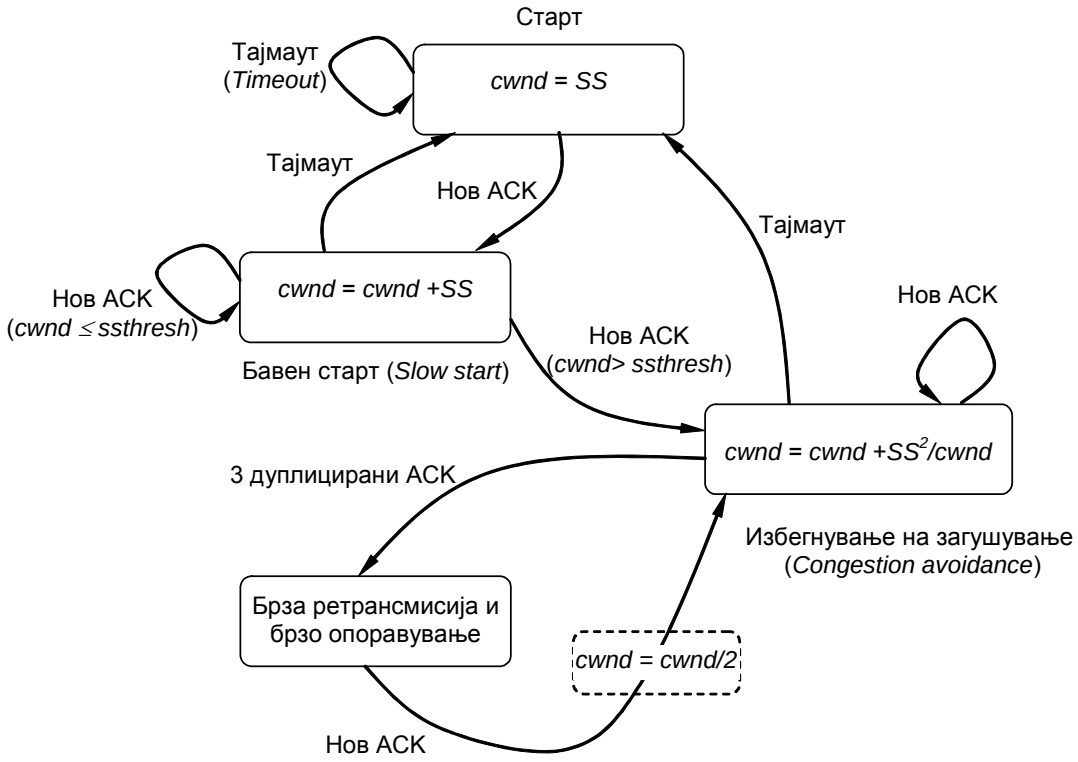
Слика 2.21 Дијаграм на состојби при воспоставување на TCP конекција

Доколку двете страни истовремено побараат раскинување на конекцијата, клиентот пред да дојде во CLOSED состојбата поминува низ CLOSING и TIME WAIT состојбите. Серверот поминува низ претходно опишаните состојби.

Механизми на TCP

TCP содржи дефинирани механизми на реакција во услови на појава на загуби во мрежата. Во услови на Интернет, имајќи го во предвид и принципот една класа на

сообраќај за сите, загубите се појавуваат во јазлите на мрежата (рутери, комутатори) кога ќе се појави загушување. Во таков случај, дел од пакетите се загубени. TCP протоколот на страната на изворот ќе открие дека пакет е загубен кога ќе прими пакет со ист секвенцијален број како последниот позитивно потврден пакет или ако воопшто не прими пакет од секвенцата подолго време.



Слика 2.22 Механизам на TCP за избегнување на загушување и оправување од загуби

Бидејќи TCP иницијално е дефиниран за фиксни пакетски мрежи засновани на IP протоколот, секогаш претпоставува дека загубите се резултат на загушување на попатните јазли во мрежата, па реагира со намалување на протокот за избегнување на загушувањето (congestion avoidance). За контрола на загушувањето TCP користи механизам на прозорци (windows), [12]. За функционирањето на механизмот на прозорци, TCP користи две променливи со кои се регулира големината на прозорецот на загушување: cwnd (congestion window) која ја содржи моменталната вредност на прозорецот на загушување, и праг на бавниот старт ssthresh (slow start threshold). Секоја TCP врска започнува со бавен старт (slow start), [13], слика 2.22, при што cwnd се сетира на големина на еден TCP сегмент. Исто така, по тајмаут врската продолжува од

фазата на бавен старт. TCP обезбедува контрола на потокот (flow control) така што не дозволува прозорецот на загушување cwnd (congestion window) да ја надмине максимална големина на cwnd, договорена со TCP протоколот на приемната страна пред почеток на врската (rcvwnd - receiver advertised window size). Во овој труд претпоставуваме дека секогаш е задоволено cwnd < rcvwnd. Во фаза на бавен старт (slow start), секоја примена позитивна потврда влијае TCP изворот да го зголеми cwnd за еден сегмент. Кога големината на прозорецот ќе го достигне прагот на бавниот старт ssthresh, тогаш навлегува во друга фаза, фаза на избегнување на загушувањето CA (Congestion Avoidance), [13]. Во оваа фаза промените на cwnd не се тригерираат со доаѓање на нови ACK, туку cwnd се зголемува приближно за еден сегмент за време на кружниот пат на пакетите меѓу изворот и дестинацијата на TCP врската.

Кога TCP ќе открие загуба на сегмент, реагира на тој начин што го препратува загубениот сегмент. Постојат два механизми за детектирање на загубен сегмент и ретрансмисија: временски (timer-driven) и податочно (data-driven) тригерирани ретрансмисии. Временски тригерирана ретрансмисија настанува кога TCP на изворот на пакети нема да добие потврда од дестинацијата. За да се тригерира ретрансмисија потребно е да истече одредено време тн. тајмаут (timeout). За одредување на овој временски интервал потребно е да се одреди прво просечното време на кружниот пат, кое што се одредува по EWMA (Exponential Weighted Moving Average) формулата согласно спецификацијата:

$$srtt = \alpha * rtt + (1 - \alpha) * srtt \tag{2.1}$$

Во (2.1) со srtt (smoothed round trip time) е означена средната вредност на времето на кружниот пат на пакетите меѓу изворот и дестинацијата (извор-дестинација-извор), rtt (round trip time) е моменталната вредност на времето потребно пакет да отиде од изворот до дестинацијата и да се врати назад, додека α е константа во формулата, која што вообичаено е поставена на вредност 0,125. Тајмаут (rto – round trip timeout) се појавува ако до изворот не пристигне ACK за временски период:

$$rto = srtt + 4 * rtt\,\mathrm{var} \tag{2.2}$$

Во (2.2) rttvar (round trip time variance) ја означува варијансата на кружното време, која се пресметува на ист начин како и средното кружно време, само со α=0,25. Повеќето TCP реализации користат грануларност од 500msec за тајмаут. Меѓутоа, тајмаутот вообичаено е последното решение бидејќи е најмалку флексибилно.

Напредните механизми на TCP за реагирање во услови на загубен пакет(и) и механизмите за избегнување на загушувањето се дефинирани во верзии на TCP. Податочно тригерираните ретрансмисии користат прием на дуплицирани потврди како информација за изгубен пакет. Сите сегменти по загубениот генерираат дуплицирани потврди до TCP изворот. Меѓутоа, при преносот на IP пакетите може да се случи да се добие дуплициран ACK без да се загуби пакет (тоа е случај на прием на пакетите од дестинацијата по различен редослед од тој на испраќање). Затоа, за да се избегне препраќање на сегмент кој не е всушност изгубен (не е примен по редослед на испраќањето), се чека да се добијат 3 последователни дуплицирани потврди (fast retransmission), за потоа да се изврши ретрансмисија на пакетот. Тоа е проследено со брзо опоравување на TCP потокот (fast recovery), фаза во која се продолжува со испраќање на пакети ако изворот продолжи да прима дуплицирани потврди. Тоа значи дека пристигнуваат пакетите кои по редослед се по загубениот пакет, што значи дека потокот продолжува да "протекува" во мрежата, па според тоа нема потреба да се премине на фазата бавен старт. При постоење на брзо опоравување, TCP, откога ќе биде сигурен дека повеќе од половината од пакетите во актуелниот прозорец cwnd пристигнале до дестинацијата врз основа на бројот на примени дуплицирани потврди, продолжува со испраќање на пакети. По пристигнување на позитивна потврда за препратениот пакет, TCP излегува од фазата на брза ретрансмисија и брзо опоравување, го намалува cwnd за половина и продолжува да работи во состојба на CA, (слика 2.22).

Постојат повеќе верзии на TCP, зависно од тоа кои механизми се имплементирани и на кој начин.

Имплементации на TCP

Дефинирани се поголем број на имплементации на TCP протоколот, варијации на иницијалната дефиниција на TCP. Најмногу распространети верзии на TCP се верзиите TCP Tahoe и TCP Reno.

Во TCP Tahoe верзијата се имплементирани бавниот старт, избегнувањето на загушување и брзата ретрансмисија, која што во оваа верзија е проследена со враќање во бавен старт по детекција на загушување на потокот во мрежата. Ако на механизмите на TCP Tahoe го додадеме механизмот на брзо опоравување (fast recovery), тогаш ја добиваме верзијата на TCP Reno. Погоре опишаните механизми на TCP се однесуваат во потполност на TCP Reno. Оваа верзија денес е и најраспространета во TCP имплементациите. Ако не е поинаку нагласено, во понатамошниот текст на овој труд под TCP ќе се подразбира TCP Reno верзијата.

TCP Tahoe функционира добро при поединечни загуби на пакети во еден прозорец на загушување, но започнува со бавен старт по загуби. TCP Reno ги подобрува перформансите на TCP потокот при единечни загуби, но може да има проблеми при повеќекратни загуби на пакети во еден прозорец. За надминување на различни проблеми кои се појавуваат во TCP транспортот дефинирани се повеќе различни верзии на TCP, како што се: TCP NewReno, TCP SACK (Selective ACKnowledgements), итн.

TCP NewReno верзијата воведува подобрување на популарната TCP Reno верзија во однос на контролата на загушување. Подобрувањето се однесува при повеќекратни загуби во еден прозорец. За да се избегне навлегувањето во тајмаути кај TCP Reno, во TCP NewReno изворот го запомнува најголемиот секвенцијален број испратен до дестинацијата. Кога ќе навлезе TCP изворот во фаза на брза ретрансмисија и опоравување, TCP ги испраќа загубените сегменти се додека не добие потврда дека го TCP приемникот го примил и сегментот со најголем секвенцијален број испратен од изворот. Се дотогаш TCP NewReno останува во фазата на брза ретрансмисија и опоравување. На ваков начин, TCP NewReno овозможува TCP изворот да се опорави од X загуби на сегменти за временски период од X кружни временски интервали rtt.

На пример, за подобрување на перформансите на TCP во услови на повеќе последователни загуби на пакети во еден прозорец, предложена е и TCP SACK верзијата, [14]. При единечни загуби TCP SACK реагира на идентичен начин како TCP Reno. Имено, секој дуплициран ACK ја има активирано SACK опцијата. Во оваа верзија приемникот при загуби испраќа информација до изворот за најмногу 3 најдолги примени континуелни блокови податоци (ограничувањето доаѓа од расположивиот простор во заглавието на TCP протоколот), како додаток кон дуплицираните ACK. На тој начин, TCP изворот располага со точни информации за тоа кои сегменти се успешно примени во приемникот. Имајќи ја таа информација TCP во изворот ќе ги препрати

само навистина загубените пакети, подобрувајќи ја контролата на потокот. Имено, кога прозорецот на TCP изворот овозможува да се испрати нов сегмент како одговор на дуплициран ACK, доколку се наоѓа во фаза на брза ретрансмисија TCP ја користи информацијата од селективните потврди за да ги препрати загубените сегменти пред да испрати нов сегмент кон дестинацијата. Функционирајќи на ваков начин, TCP SACK овозможува да се реставрира потокот по повеќекратни загуби во еден прозорец во еден интервал на кружното време rtt, што не е случај со верзиите Tahoe и Reno. Бидејќи загубите на пакети во серии се очекувани да се појават во услови на безжичен линк, кој се одликува со променлива и локациски зависна битска грешка, TCP SACK овозможува побрзо опоравување на потокот од повеќекратните загуби во еден прозорец.

2.7 Дискусија

Во оваа глава ги разгледавме најважните протоколи во Интернет денес, но и во Интернет на самите почетоци пред неколку децении. Овде ќе направиме само кратко резиме. Имено, најважниот Интернет протокол го носи токму името Интернет протокол – IP. Тој се наоѓа на мрежното ниво на сите хостови и рутери приклучени на Интернет и е сржта на "мрежата". Направен е така да биде флексибилен кон сите типови на апликации кои постојат денес, но и целосно отворен и кон оние што ќе доаѓаат во иднина. Тоа произлегува од извонредната идеја при креирањето на IP, а таа се содржи во тоа да се води сметка само за да стигне дадена порака (наречена датаграм или IP пакет) од еден хост извор до друг хост дестинација исклучиво врз база на IP адресите кои ги имаат мрежните интерфејси (кон Интернет) на двата хоста кои комуницираат меѓусебе. Сите останати работи како доверливоста на пораките, типот на информациите (говор, видео, мултимедија, пораки, податоци, итн.), се оставени на други протоколи над него или под него, зависно од типот на мрежата, како и на апликациите за конкретниот сервис.

Покрај IP, во оваа глава ги изнесовме UDP и TCP како најзначајни транспортни протоколи, преку кои оди најголемиот дел од Интернет сообраќајот денес. Притоа, доминантна улога има TCP како протокол кој ја дава доверливоста на преносот на податоци преку Интернет и кој е најзастапен транспортен протокол денес, заради што целиот протоколен стек на Интернет често се рефернцира и како TCP/IP. Сржта на best-

effort принципот на Интернет што резултира во подеднакво делење на расположивиот капацитет меѓу сите конекции кои делат ист линк во исто време, директно произлегува од механизмите на кои е изграден TCP.

Глава 3

TCP/IP клиент-сервер комуникација

Основна цел на оваа глава е опис и имплементација на API (Aplication Program Interface) во комуникациските протоколи, делот кој се наоѓа меѓу корисничката апликација и TCP/IP протоколниот стек, [15], [16]. Овој интерфејс е достапен за програмерот зависно до оперативниот систем кој се користи. На пример, API за работа во UNIX оперативна средина се Berkley sockets и System V transport layer interface.

Стандардите кои постојат за мрежна комуникација не специфицираат точно на кој начин треба да се остварува врската меѓу апликациските програми и софтверот на протоколите, така да интерфејсната архитектура во овој дел не е стандардизирана. Втора причина заради која се одделени интерфејсите од протоколите за мрежни апликации е зависноста на интерфејсот од конкретниот оперативен систем кој се користи. Меѓутоа, и покрај што нема детален стандард, најшироко е прифатен и се применува при креирањето на API на 4BSD sockets. За да преминеме на подетален опис на API за мрежни апликации, да се навратиме на историјата.

UNIX оперативниот систем е креиран кон крајот на 60-те и почетокот на 70-те години. Тој беше замислен како еднопроцесорски систем со делење на процесорското време меѓу апликациите кои истовремено се извршуваат (timesharing scheme). Основата во UNIX е процесот. Апликациските програми се извршуваат како процеси. Притоа, комуникацијата на апликацијата со оперативниот систем е преку тн. системски повици (system calls) кои од гледна точка на програмерот се однесуваат како процедури. Имено, при системскиот повик се специфицираат одредени аргументи (целобројни

вредности или покажувачи кон некој објект во апликацијата), а по извршувањето на истиот се добиваат вредност(и) како резултат.

Кога се работи за UNIX-овите влезни/излезни карактеристики, тогаш важи филозофијата отвори-прочитај-запиши-затвори (open-read-write-close). Всушност, кај UNIX оперативниот систем сета работа за влезни/излезни операции се сведува на читање и запишување во датотеки, било да се работи за обична датотека, уред (device) или пак мрежна комуникација. Кориснички процес може да започне I/O (Input/Output) операции со повикот open, при што повикот враќа како одговор цел број наречен фајл дескриптор (file descriptor) кој е поврзан со повиканиот фајл или уред. Кога објектот кон кој се пристапува за I/O операции ќе биде отворен, може да се чита од него или да се запишува во него користејќи ги повиците read и write (и двата повици користат по три аргументи при повикување: фајл дескрипторот, адресата на баферот и бројот на бајти кои треба да бидат пренесени), соодветно. Кога ќе се заврши со I/O операциите, процесот го завршува комуницирањето со објектот со повикување на close (ако процесот кој повикал open заврши, оперативниот систем автоматски ги затвара сите I/O операции поврзани со тој процес и без повикување на close).

На почетокот од имплементацијата на TCP/IP во раните 80-те години, за мрежна комуникација на UNIX оперативниот систем со други системи врзани во мрежа, се користел истиот пристап како за I/O операции за фајлови. Оригиналната TCP/IP имплементација за BSD UNIX била развиена од Bolt Beranek и Newman за DARPA во 1981 година. Меѓутоа, бидејќи мрежните протоколи се покомплексни од обичните I/O уреди, следувало дека комуникацијата меѓу корисничките процеси и мрежните протоколи мора да биде посложена од комуникацијата меѓу корисничките процеси и обичните I/O операции локално во системот. За таа цел, при креирањето на UNIX 4BSD се вовеле повеќе нови системски повици во оперативниот систем, а согласно и нови рутини во библиотеките на системот (library routines). Комплексноста произлегува и од тежнеењето интерфејсот за UNIX протоколите да овозможува поддршка на повеќе протоколи, а кои можат истовремено да постојат во системот (на пример, TCP/IP и Xenox). Подоле се наведени некои карактеристики кои е потребно да се имаат во предвид при мрежното програмирање за разлика од I/O операциите со фајлови:

- Односот клиент/сервер не е симетричен. Процесот треба да знае дали е клиент или сервер.

- Мрежната комуникација може да биде конекциски ориентирана или неконекциски ориентирана (понатаму ќе биде објаснето значењето на секој од овие термини).
- Во мрежните I/O операции имињата се многу позначајни од I/O операциите со фајлови. Имено, при мрежна комуникација, апликацијата мора да го знае името на оддалечениот примитивен процес (primitive) за да може да комуницира со него.
- При секоја мрежна комуникација мора да се специфицираат 5 параметри (во I/O операциите со фајлови беа потребни само 3 параметри): протокол, локална адреса, локален процес, надворешна адреса, надворешен процес.
- Некои комуникациони протоколи имаат ограничување во однос на големината на пораките кои се пренесуваат. Unix-от I/O систем е проточно (stream) ориентиран, а не е базиран на пораки (messages).
- Мрежниот интерфејс би требало да поддржува повеќе комуникациони протоколи кои истовремено постојат во кернелот на оперативниот систем, така што било потребно да се употребуваат генерални техники за да се контролираат својствата како што се адресите.

Основите за I/O операциите за мрежна комуникации се тн. сокети (sockets).

3.1 Што е сокет?

Првиот сокет интерфејс е направен во рамките на системот 4.1cBSD за VAX во 1982 година. Сокет интерфејсот кој денес е најраспространет е од оперативниот систем Unix верзија 4.3BSD од 1986 година. Оваа верзија ги опфаќа следните комуникациони протоколи:

- Unix domain
- Internet domain
- Xenox NS domain

За мрежно програмирање од интерес е Internet domain протоколот, па според тоа и сокет интерфејсот (Internet sockets) кон овој комуникационен протокол. Покрај погоре наведените постојат и други комуникациони протоколи кои не се од интерес во оваа глава, како на пример CCITT X.25 Sockets.

За мрежни интерфејси се користат во основа два типа на Интернет сокети, [15], и тоа:

- **TCP/IP sockets (stream sockets)**, за конекциски ориентирана (connection oriented) комуникација (при комуникацијата прво се воспоставува врската и се одредува патот по кој ќе се пренесуваат пакетите, а потоа се остварува преносот по утврдената патека).
- **UDP sockets (datagram sockets)**, за неконекциски (connectionless) ориентирана комуникација (при тоа не е потребно да се воспостави претходно врската меѓу крајните точки на преносот, туку адресата на дестинацијата се содржи во заглавјето на секој пакет и потоа секој пакет одделно го пронаоѓа патот до дестинацијата).

Покрај овие два типа на сокети, во Интернет сокети припаѓаат и се употребуваат и Raw sockets, кои се користат за дозволување на привилегирани програми да пристапат до протоколите во пониските нивоа од мрежниот интерфејс (овие сокети се спомнати само заради целосност во излагањето).

3.1.1 TCP sockets

Stream sockets се двонасочни комуникациони потоци со доверлив пренос на пакетите. Да ги разгледаме подетално нивните карактеристики:

- Пот поток (stream) се подразбира пренос на големо количество на податоци меѓу две апликациски програми (кориснички процеси), при што податоците се поделени во 8 битни октети тн. бајти (bytes).
- Кај овој тип на интерфејс врската е виртуелен круг (круг бидејќи е двонасочна комуникација). Овозможувањето на пренос на податоци е аналогно на воспоставување на телефонска врска. Всушност, едната машина го генерира повикот за воспоставување на врската, а другата го прифаќа. При воспоставувањето

на повикот двата модули на протоколите во двете машини мораат да ги информираат локалните оперативни системи дека сакаат да воспостават врска за stream пренос на податоци. Врската е виртуелна, бидејќи апликациските програми ја гледаат врската како доделен хардвер, но всушност тоа е илузија за програмите од сервисите за контрола на стреам преносот. Доделените ресурси при една комуникација по завршувањето на истата се ослободуваат и можат да се доделат на друга врска. Следното воспоставување на комуникацијата меѓу двете точки не значи дека ќе биде воспоставена истата виртуелна патека, бидејќи тоа зависи од расположливите ресурси на патеките меѓу двете крајни точки и алгоритмите за нивно доделување.

- Трансферот е бафериран. Апликациските програми ги праќаат stream податоците во виртуелниот пат, при што големината на пакетите слободно ја одредува апликацискиот програм (цел број бајти, значи најмалата големина на пакет може да биде еден бајт), но протоколите од подолните нивоа во мрежниот модел можат да ги прегрупираат октетите во нови пакети за да обезбедат контрола на протокот за ефикасно искористување на мрежните ресурси. Притоа, пакетите пристигнуваат на приемната страна по истиот редослед по кој се испратени. И на двете страни постојат бафери во кој се сместуваат октетите кои треба да се пренесат/кои се примени. Ако баферот има слободен простор постои тн. push функција која ја форсира апликацијата побрзо да генерира пакети, односно на приемната страна обезбедува TCP да ги доставува податоците на приемната апликација без доцнење.
- Потокот податоци не е структуиран така да може на поминување низ виртуелниот пат да биде прочитан од трета страна. Имено, пред започнувањето на преносот двете страни се договараат за форматот на потокот на податоци кој ќе се користи при преносот.
- Врската е двонасочна (full duplex). Комуникацијата на TCP/IP потокот овозможува истовремена комуникација во двете насоки. Комуникацијата во едната насока може да се заврши без да се наруши другата насока. Двонасочниот пренос го редуцира мрежното сообраќајно оптоварување, бидејќи овозможува повратната контролна информација да се пренесува во датаграмите кои патуваат во спротивната насока.

3.1.2 UDP sockets

UDP (User Datagram Protocol) се еден дел од TCP/IP протоколот, обезбедувајќи го примарниот механизам за да можат апликациските протоколи да испраќаат датаграми до други апликациски програми. Имено, TCP/IP овозможува пренос на IP датаграми преку компјутери хостови (hosts), при што секој датаграм се рутира зависно од неговата IP адреса на дестинацијата. Во нивото на Интернет протоколот (IP-Internet Protocol) IP адресата го одредува само дестинациониот хост, но ништо повеќе во однос на корисникот или апликацискиот програм кој треба да го прими датаграмот. Во оперативните системи на повеќето компјутери е овозможено симултано извршување на повеќе програми кои се реферираат како процеси (processes), задачи (tasks), апликациски програми (application programs) или кориснички процеси (user level processes). Ваквите системи се нарекуваат мултипроцесорски системи. Меѓутоа, да се специфицира одреден процес како крајна дестинација не е погодно, затоа што процесите постојано се создаваат и завршуваат. На пример, ние би сакале да можеме да ги промениме процесите кои ги примаат датаграмите без да го информираме испраќачот на датаграми (таков случај е reboot-ање на машината). Потоа, цел е да може да се воспостави комуникација со дестинацијата врз основа на функциите кои ги обезбедува, но без да се знае процесот кој ги обезбедува тие функции (на пример, сакаме да му се дозволи на испраќачот на датаграмот да воспостави врска со некој сервер без да знае кој процес на дестинационата машина го имплементира тој сервер). Заклучок е дека наместо да се користи процесот како крајна дестинација при комуникацијата, ќе замислиме дека секоја машина содржи множество на апстрактни дестинациони точки наречени порти за протоколи (protocol ports). Секоја порта за протокол се идентификува преку одреден (предефиниран) цел број. Оперативниот систем во машината содржи интерфејс кој го одредува доделувањето на портите и пристапот до нив зависно од карактерот на апликацијата. Притоа, за време на пристапот до портата се запира работата на процесот (пример, процесот не може да ги извлече податоците од портата пред да пристигне целиот блок на податоци кој процесот сака да го прочита. За да се комуницира со надворешна порта (порта која не се наоѓа на локалниот компјутер), потребно е да се специфицира во пораките до оддалечената порта IP адресата на машината до која се пристапува и портата на таа машина преку која ќе се пристапи. Покрај дестинационата порта, секоја порака содржи

податок за изворната порта на машината од која се праќа таа порака, а кон која треба да бидат адресирани одговорите или пораките од оддалечената машина (секој процес кој ќе ја прими пораката ќе може да пристапи до изворната машина, бидејќи ќе ја прочита изворната порта од пораката).

Како карактеристики на UDP протоколот може да се наведат следните:

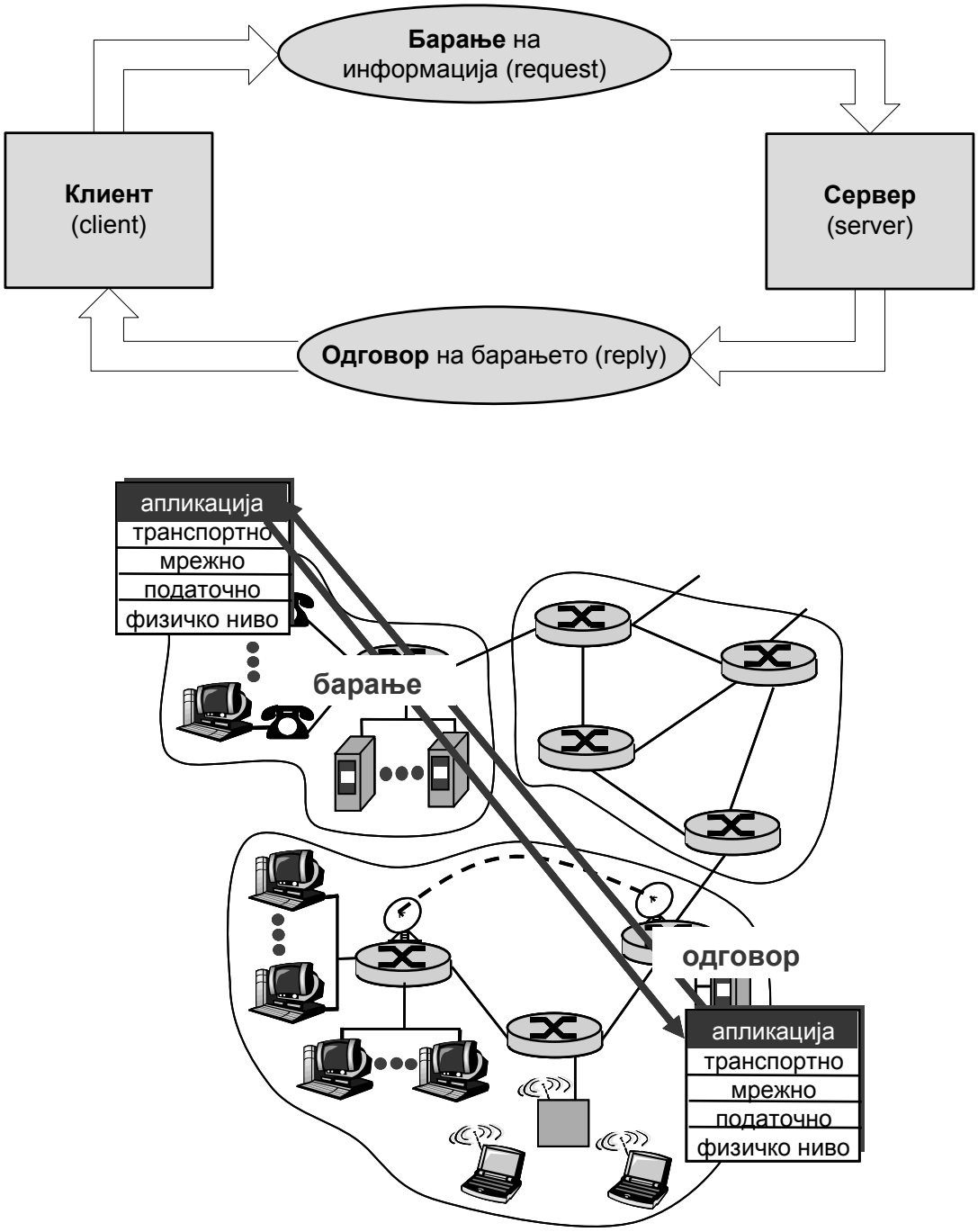
- Недоверлив пренос на податоци, податоците може да бидат изгубени или дуплицирани (нема повратна потврда кај овој протокол).
- Нема индикација за загушување на приемната страна. Имено, праќачот на пораките може да ги испраќа со поголема брзина од онаа што може да ја поднесе примачот на пораките.
- Нема сигнализација меѓу изворот и дестинацијата, нема претходно поврзување на крајните точки при комуникацијата како што е случај кај TCP.

UDP, всушност го користи IP, мрежниот протокол, обезбедувајќи го истиот сервис како и IP за пренос на пораки од еден хост до друг, но со една важна разлика, а тоа е можноста на UDP да разликува повеќе различни процеси на една иста машина благодарејќи на воведувањето на портите за протоколи (имено, секоја UDP порака ги содржи двете порти, изворната и дестинационата).

3.2 Модел клиент-сервер

Во комуникацијата преку мрежа најчесто се користи моделот клиент-сервер. Тоа значи дека некои од машините кои се приклучени на мрежата се подготвени да примат барање од друга машина за одреден сервис. Моделот клиент-сервер е асиметричен:

- Серверот нуди сервиси (услуги) преку добро дефиниран интерфејс.
- Клиентот бара сервиси преку интерфејсот на серверот.
- Клиент: како да се побара сервис.
- Сервер: како да се понуди сервис.



Слика 3.1 Клиент-сервер комуникација

TCP/IP комуникацијата е заснована на моделот клиент-сервер (слика 3.1). Серверот, кој е всушност програм кој ослушнува (наречен daemon) на одредена TCP порта на хостот. Клиентот е програм кој ја иницира врската кон апликацијата (сервер) на другата машина. Општо, при комуникацијата меѓу клиентот и серверот може да се употребува TCP, UDP или некој друг протокол, меѓутоа и на двете страни потребно е да се користи ист тип на протокол и соодветно сокет интерфејси.

Пример за клиент-сервер модел е telnet апликацијата: серверот ги очекува повикувањата на неговата порта 23. Клиентот кој бара telnet сервис, при повикувањето се поврзува на порта 23 на серверот. Друг пример се ftp и http протоколите. Ftp (а истото важи и за http) е протокол за пренос на фајлови. Тоа значи дека програм наречен daemon е активен на машината која ги нуди фајловите. Друг програм, кој го поддржува ftp воспоставува врска со серверот и се претставува. Откога врската еднаш ќе биде воспоставена, можат да се пренесуваат команди и фајлови во двете насоки.

Во случај на неконекциски ориентирана комуникација (datagram service) не е потребна експлицитна идентификација за тоа кој е сервер, а кој е клиент. При иницирање на повик повикувачката апликација потребно е да ги наведе сопствената IP адреса и бројот на портата на локалната машина преку која ќе се остварува комуникацијата од нејзина страна. Доколку се чека на поврзување однадвор, апликацијата треба да ја декларира портата на која на локалната машина преку која сака да се остварува поврзувањето по иницирачки повици од други машини во мрежата.

Во случај на конекциски ориентирана комуникација (TCP) сликата е поинаква:

- Клиентот мора да се поврзе со серверот пред да прима или испраќа податоци од/кон него.
- Серверот мора да ја прифати (accept) комуникацијата со клиентот пред да праќа или прима податоци.
- Серверот може да го прифати клиентот (системски повик accept()) кога ќе прими барање за поврзување преку повикот connect() иницран од клиентот.

Овие системски повици за поврзување, а подоцна и комуникација на крајните точки (клиентот и серверот) го сочинуваат сокет интерфејсот. Системските повици за сокет интерфејсот се детално опишани во делот 3.4 од оваа глава.

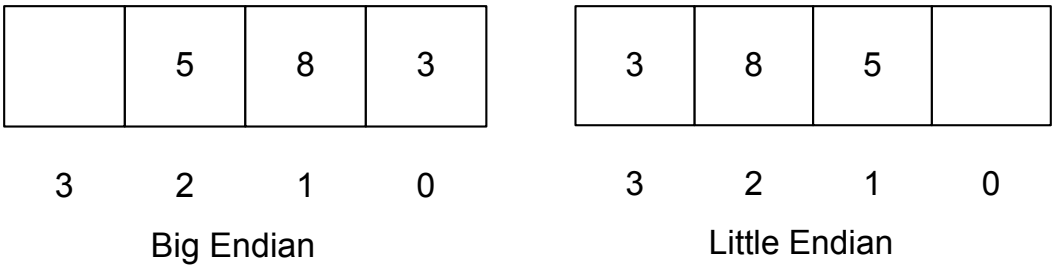
3.3 Мрежно адресирање

За да може да се оствари одредена комуникација потребно е постојат одредени адреси кои ќе бидат уникатни и различни за секоја машина на мрежата. Секој Интернет

хост има една или повеќе глобално-единствени адреси (тн. IP адреси), кои претставуваат 32 битни броеви. Бројот на адреси кои му се доделуваат на одреден хост зависи од бројот на интерфејсни мрежни картички кои ги поседува (за секоја картичка се доделува една уникатна IP адреса). Адресите се запишуваат во форма на 4 цели броеви разделени меѓусебе со точки (dotted decimal notation), при што секој одговара на еден бајт од Интернет адресата. На пример 206.99.216.1 е една Интернет адреса. Истата IP адреса е глобална, што значи дека се користи при секоја комуникација со хостот на кој му е доделена таа адреса. Концептуално, секоја IP адреса се состои од два дела: идентификатор за мрежата на која припаѓа хостот (netid), и идентификатор за самиот хост во рамките на таа мрежа (hostid). Токму поради тоа, IP адресите не ги специфицираат поединечните машини, туку точките за поврзување во мрежата (оттаму може една машина да има повеќе адреси како што спомнавме претходно во текстот), но пак при префрлување на машината од една мрежа во друга таа ќе добие друга IP адреса.

Адресите може да се однесуваат и на мрежи и на хостови. Вообичаено, кај адресите на мрежи сите битови кои се определени за адреса на хостот се нули.

При рутирањето на пакетите низ мрежите рутерите го користат делот од IP адресата специфицирана во пакетите која се однесува на мрежата на која припаѓа хостот (netid).



Слика 3.2 Big Endian и Little Endian начин на претставување на податоците

Меѓутоа, за да се креира Интернет кој е независен од архитектурата на поединечните машини во мрежите, а кои во основа се разликуваат една од друга помалку или повеќе, било потребно да се дефинира стандардно претставување на податоците, на пример, кога една машина испраќа 32 битен цел број до друга, што е случај кога се испраќа Интернет адреса. Податоците се пренесуваат низ физичкиот медиум без да им се промени редоследот. Но, сите машини не ги сместуваат 32-битните цели броеви на еден ист начин. На некои машини најниската мемориска адреса го

содржи најмалку значајниот бајт од одреден податок, и тие се нарекуваат Little Endian. На други машини, наречени Big Endian, најниската мемориска адреса го содржи најзначајниот бајт од одреден податок (слика 3.2).

Воведувањето на стандард за редоследот на бајтите при преносот од една машина кон друга преку мрежа е од суштинско значење бидејќи во пакетите се содржат информации како што се IP адресите, должината на пакетите, кои мора да бидат протолкувани на ист начин од машината која ги прима и од машината која ги праќа. За таа цел е воведен стандард по кој редослед да се испраќаат бајтите низ мрежата во однос на нивното значење (MSB-Most Significant Byte прв да се испрати или пак тоа да биде LSB-Least Significant Byte). При тоа, прифатено е стандард при пренесувањето на податоци преку мрежа (network standard byte order) да биде стилот Big Endian. Тоа значи дека при испраќањето пакет (кој секогаш содржи повеќе бајти), прво се испраќа најзначајниот бајт (MSB) од пакетот и на крај се испраќа кон другата машина најмалку значајниот бајт од пакетот (LSB). Значи секоја машина е слободна да има сопствен локален начин на претставување на податоците, но пред да се испратат податоци на мрежа потребно е да се претворат во network byte order. Обратно, на приемната страна хостот ги претвора податоците од network byte order во host byte order, односно во начин на претставување на податоците локално во машината.

За конвертирање на податоците од локално ниво на мрежно ниво во оперативниот систем Unix воведени се системски повици за таа цел.

3.3.1 Адресни структури за сокети

Од гледна точка на апликациското програмирање единствена разлика меѓу мрежните протоколи се шемите за адресирање на мрежите и хостовите. Ако тоа се постави на една страна, сето што останува на програмерот при создавањето на мрежна апликација е употреба на операциите како што се connect, send, receive и disconnect. За TCP/IP, идеален интерфејс API би бил тој што би ги "разбирал" IP адресите и броевите на портите (port numbers). Бидејќи библиотеката за сокети во оперативниот систем обично се користи за повеќе протоколи (не е врзана со еден), адресите се сместуваат во поопшти структури кои би можеле да бидат употребени од повеќето протоколи кои се раширени во употреба. Таа општа структура на адресите изгледа вака:

```
struct sockaddr {
unsigned short sa_family;    /* address family, AF_XXX */
char sa_data[14];            /* protocol specific data */
};
```

Во оваа структура, sa_family полето го одредува протоколот. За TCP/IP, ова поле е поставено да биде AF_INET, додека на пример за други протоколи се користи AF_UNIX за Unix domain, AF_NS за Xenox итн. Останатите 14 бајти од оваа структура зависат од протоколот кој се употребува. Од наш интерес се TCP/IP адресните структури, па само на нив ќе се задржиме. За Интернет фамилијата во 14 преостанати бајти се сместуваат Интернет (IP) адресите и броевите на портите. За да се овозможи подобро користење на овие полиња од оперативниот систем се користи посебен тип на адресна структура наместо погоре спомнатата:

```
struct sockaddr_in {
short sin_family;            /* AF_INET za Internet*/
unsigned short sin_port;     /* 16-bit port number, network byte ordered */
struct in_addr sin_addr;     /* 32-bit netid/hostid, network byte ordered */
char sin_zero[8];            /* neiskoristeni 8 bajti */
};
```

Двете структури кои се прикажани погоре се компатибилни меѓусебе. Двете имаат по 16 бајти. И во двете структури првите два бајти се поле за типот на фамилијата (Internet, Unix, Xenox итн.). Така, struct sockaddr_in секогаш може да се преслика во struct sockaddr. Може да се забележи дека специфичната структура за Интернет фамилијата (за TCP/IP) содржи во својата дефиниција структура in_addr како поле, а која ја има следнава форма:

```
struct in_addr {
unsigned long s_addr;        /* 32-bit netid/hostid, network byte order */
};
```

Структурата sockaddr_in е комплицирана структура создадена да може да ја содржи IP адресата на различни начини. Во структурата in_addr можат да се сместат точно 4 бајти колку што е и долга една Интернет адреса. Во sockaddr_in полето sin_port има големина од 16 бити и служи за сместување на бројот на портата. Како што и претходно спомнавме, сите овие полиња мора да бидат во формат на мрежата (network byte order). Дефинициите на овие структури се наоѓаат во <sys/socket.h>. Да разгледаме еден пример на употреба на адресна структура за Интернет адреса:

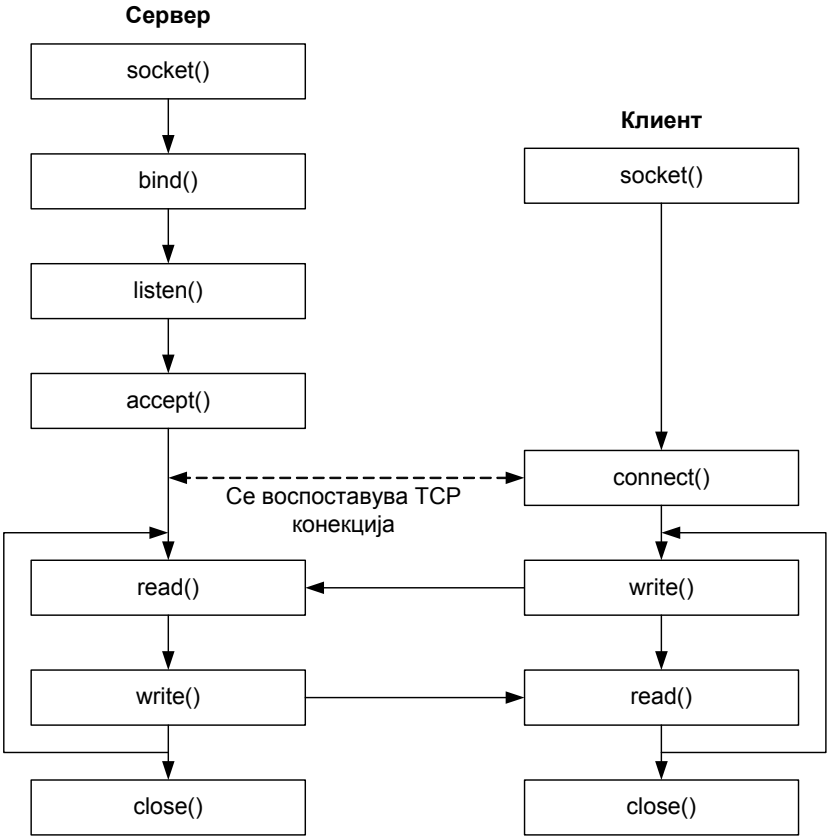
```
struct sockaddr_in sin;
sin.sin_family=AF_INET;
sin.sin_port=htons(6666)
sin.sin_addr.s_addr=inet_addr("206.99.216.1");
```

Во овој пример, структурата sin ја содржи Интернет адресата 206.99.216.1, и бројот на порта 6666. Притоа, за сетирање на овие вредности се употребени две библиотечни функции (библиотечни бидејќи се содржат во самиот оперативен систем), htons за претворување на бројот на портата од локален редослед на бајтите во мрежен редослед на бајти, и функцијата inet_addr за конвертирање на стрингот кој ја претставува Интернет адресата во децимален облик во 32-битен цел број со мрежен редослед на бајтите (network byte order).

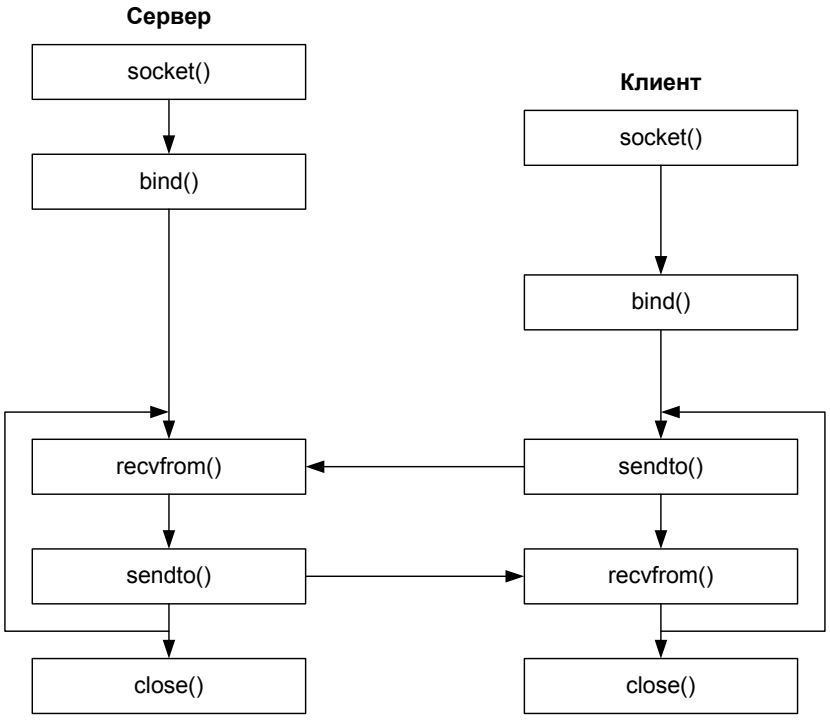
Структурата за интернет адреса sockaddr_in служи за да се одреди хостот (преку Интернет адресата) и апликацијата (преку бројот на портата) кон која треба да се испрати датаграмот.

3.4 Системски повици и рутини за сокети

Во овој дел ќе бидат разгледани елементарните системски повици, за подоцна да се опфатат и некои од посуптилните системски повици за сокети во оперативниот систем Unix 4.3BSD.



а) конекциски ориентиран протокол (TCP)



б) неконекциски ориентиран протокол (UDP)

Слика 3.3 Системските повици за сокети за конекциски(а) и неконекциски(б) ориентирани протоколи

На почеток ќе ги прикажеме (слика 3.3) двете сценарија кои се од интерес: конекциски ориентирани протоколи (connection oriented) и неконекциски ориентирана протоколи (connectionless).

Од наш интерес се Интернет протоколите, а тоа се TCP за конекциски ориентирана комуникација и UDP за неконекциски ориентирана комуникација. Во однос на овие два протоколи ќе бидат разгледани и системските повици за сокети во продолжение на оваа глава.

3.4.1 Системски повик socket

За да може еден процес да извршува мрежни влезно/излезни операции, прва работа што треба да ја направи е да го повика системскиот повик socket, со што ќе го специфицира протоколот кој ќе го користи при мрежната комуникација (Internet TCP, Internet UDP итн.). Системскиот повик socket е дефиниран вака:

int socket (int family, int type, int protocol);

Накратко, овој системски повик ја креира крајната точка за комуникација. Овој системски повик враќа вредност што е цел број кој го референцира сокетот што се креира. Овој број како параметар се пренесува до речиси сите други системски повици. За Интернет фамилијата на протоколи, параметарот family треба да биде поставен на AF_INET. Параметарот за тип на протокол во тој случај може да биде SOCK_STREAM (за TCP) или SOCK_DGRAM (за UDP). Последното поле во дефиницијата за socket се користи за специфицирање на протокол кога мрежниот модел поддржува различни поточни или датаграм модели. Како и да е, TCP/IP има само по еден протокол за двата типа на комуникација, така да ова поле треба да биде сетирано на 0.

Примери:

1) креирање на TCP socket:

```
int s;
s=socket(AF_INET, SOCK_STREAM, 0);
```

2) креирање на UDP socket:

```
int s;
s=socket(AF_INET, SOCK_DGRAM, 0);
```

3.4.2 Системски повик bind

Пред да се започне со праќање и примање на податоци преку сокетот потребно е прво да се поврзе сокетот со локална порта на машината и адресата на мрежниот интерфејс. Пресликувањето на сокетот на TCP/UDP порта на машината и на Интернет адресата е наречено поврзување (“binding”).

Поврзувањето најчесто се случува кога сервер треба да ја ослушнува мрежата и да чека повици од клиенти за поврзување. Но, исто така и клиент може да го употреби овој системски повик, на пример кога на клиентот не му е потребна специфична изворна порта на локалната машина, туку единствено што сака е да разменува пораки со одредена порта на оддалечен хост. Компликации настануваат кога постојат повеќе од еден мрежен интерфејс на хостот, па тогаш треба да се специфицира која мрежа треба да се користи. Системскиот повик bind се користи за поврзување на сокетот кој е отворен за мрежна комуникација, TCP/UDP мрежната порта и мрежниот интерфејс. Дефиницијата на системскиот повик bind е:

int bind(int socket, struct sockaddr *myaddr, int addrlength);

Првиот параметар во bind е идентификаторот на сокетот, целиот број кој се добива од системскиот повик socket. Вториот параметар е адресната структура на сокетот. Кога се работи за TCP/IP, како во овој случај, тоа е всушност структурата sockaddr_in. Од оваа структура полето sin_port ја содржи локалната изворна порта (source port) која се поврзува со сокетот. Тоа значи дека при секоја операција на испраќање на датаграм преку сокетот, во TCP/UDP заглавјето на пакетот се сместува оваа вредност за локалната порта. Ако на пример не е потребно точно специфицирање на изворна порта при комуникацијата, може да се постави вредноста на портата да биде INADDR_ANY, со што му се овозможува на оперативниот систем да избере било кој број на порта кој е на располагање во тој момент. Полето sin_addr од таа структура специфицира кој мрежен интерфејс да се користи. Најголем број од хостовите имаат

само еден мрежен интерфејс (една мрежна картичка), што значи дека имаат една Интернет адреса, па во полето sin_addr се сместува таа Интернет адреса. Меѓутоа, може ова поле исто така да се постави на вредноста INADDR_ANY, со што му се овозможува на оперативниот систем да избере било која од Интернет адресите на хостот (ако има повеќе од една) и соодветните мрежни интерфејси. Третиот параметра во bind е полето addrlength, и ја содржи големината на адресната структура изразена во бајти. Овај системски повик се користи во три случаи:

1. Серверот ја регистрира својата адреса со системот за да се овозможи пораките кои ќе бидат упатени на адресата на серверот, да стигнат до него.
2. Клиентот може да регистрира адреса за себе.
3. Неконекциски ориентираниот клиент го користи повикот во случај кога очекува да добие потврда од серверот дека некоја порака е примена, а со цел да му достави валидна адреса кон која серверот може да го испрати одговорот.

Пример:
Нека е отворен сокет со повикот во претходниот пример, тогаш поврзувањето на тој сокет со локална порта и глобално уникатна адреса на машината се остварува со следниов системски повик:

bind(s, (struct sockaddr *) &sin, sizeof(sin));

Тука можеме да спомнеме кои порти може да се доделуваат на корисничките процеси при повикот bind, а кој се резервирани порти.

Број на порта:
1 - 1023 само за привилегирани корисници
1024 - 4999 се користат од страна на системот и од корисничките процеси
5000 - се користат само од корисничките процеси

Ако портата која се специфицира да се користи од сокетот е веќе во употреба, bind ќе врати вредност -1. Кога bind е повикан за UDP сокет, сокетот е спремен за примање и испраќање датаграми. За TCP сокети, по bind сокетот е спремен за следните системски повици connect и accept.

3.4.3 Поврзување на сокетот со дестинационата адреса, системски повик connect

Системскиот повик connect се користи од страна на клиентите за поврзување со определен сервер при конекциски ориентирана комуникација (TCP). Сокетот се креира неповрзан со дестинација. За да може да се пренесуваат податоци потребно е тој да се поврзе со саканата дестинација. UDP сокетите немаат потреба да се поврзуваат пред да се користат, но од друга страна со поврзувањето се овозможува да не се специфицира дестинацијата за секој испратен датаграм. Системскиот повик за поврзување е connect:

int connect(int socket, struct sockaddr *destinationaddr, int addrlength);

Овој системски повик е дефиниран во <sys/types.h> и <sys/socket.h>. Аргументот socket е дескрипторот на сокет кој се референцира на сокетот Вториот аргумент е покажувач кон адресната структура што ја содржи дестинационата адреса, а третиот аргумент ја специфицира големината на адресната структура изразена во бајти.

Овој системски повик различно функционира зависно од типот на комуникацијата (TCP или UDP). За конекциски ориентираните протоколи (за Интернет фамилијата тоа е TCP) овој повик всушност ја воспоставува врската меѓу клиентот и серверот, или јавува грешка во случај да не може да го стори тоа. Клиент процесот не мора да се поврзе со локалната адреса и порта преку употреба на bind пред да го повика системскиот повик connect, бидејќи поврзувањето на клиентот со серверот обично предизвикува одредување на локалната адреса, локалниот процес, дестинационата адреса и дестинациониот процес. Најчесто во практиката по креирањето на сокет за клиент за доверлив конекциски ориентиран пренос, поврзувањето се остварува преку connect (не се користи bind).

За UDP клиент може исто така да се употреби системскиот повик connect. Но, во овој случај тоа значи само локално складирање на дестинационата адреса, ништо повеќе. Тоа се користи понатаму во комуникацијата за да може да знае системот каде да ги испрати податоците кои се запишуваат во сокетот. Предноста на овој случај е во тоа што не мора да се специфицира дестинационата адреса за секој датаграм.

3.4.4 Испраќање и примање на податоци преку сокет

Системски повици за читање/запишување кои користат еден бафер

Прво ќе ги разгледаме системските повици за читање/запишување од/во сокети кои користат еден бафер (податоците при читање/запишување се сместуваат во последователни мемориски локации).

За испраќање на податоци преку сокет кој е поврзан со дестинацијата со системскиот повик connect, се користи системскиот повик send, а за примање на податоци при исти услови се употребува системскиот повик recv. За примање и испраќање на датаграми без претходно поврзување на изворната со дестинационата адреса се користат системските повици recvfrom и sendto, соодветно. Овие четири системски повици се дефинирани во <sys/types.h> и <sys/socket.h>.

int send(int socket, char *buffer, int bufferlength, int flags); /* TCP/UDP */

int recv(int socket, char *buffer, int bufferlength, int flags); /* TCP/UDP */

int sendto(int socket, char *buffer, int bufferlength, int flags, struct sockaddr *destination_address, int address_length); /* UDP */

int recvfrom(int socket, char *buffer, int bufferlength, int flags, struct sockaddr *sender_address, int *address_length); /* UDP */

Овие системски повици се всушност стандардни повици за читање и запишување во фајлови, но со некои дополнителни аргументи. Во сите 4 повици погоре наведени првиот аргумент е сокет дескрипторот, вториот аргумент buffer е покажувач кон поле (бафер) од бајтови во кои се сместуваат податоците кои треба да се испратат односно кои треба да се примат, а третиот аргумент bufferlength е големината на полето во бајти. Аргументот flags обично е поставен на 0.

Системските повици send и recv се употребуваат кога претходно е воспоставена врската на клиентот со серверот преку повикување на системскиот повик connect.

Меѓутоа, при UDP комуникација нема потреба да се поврзе клиентот со серверот пред да започне комуникацијата. Во тој случај се користат системските повици sendto и recvfrom, соодветно за испраќање и примање на датаграми. За таа цел, кај овие два системски повици се специфицира адресата на дестинацијата (кај sendto), односно адресата на испраќачот на датаграмот (кај recvfrom). При тоа се користи адресната структура sockaddr (може да се употреби и sockaddr_in на ова поле во системските повици). Се користи sin_addr за специфицирање на дестинационата/изворната IP адреса и sin_port за дестинационата/изворната порта. На крај, во системските повици е специфицирана и големината на адресната структура во бајти (address_length).

Сите 4 повици враќаат вредност која е број на бајти кои се прочитани или запишани во полето на баферот.

Кај recvfrom, при користење на UDP, вредноста што ја враќа системскиот повик е должината на датаграмот. Ако просторот во баферот одреден со bufferlength аргументот е помал од големината на датаграмот кој се прима, тогаш само првите bufferlength бајти од датаграмот ќе се запишат, а другите се загубени. Со овој повик се прима адресата на испраќачот и се запишува во адресната структура и должината на оваа адреса се запишува во address_length, кој е покажувач кон цел број (истиот аргумент за должината на адресата кај системскиот повик sendto е цел број, не е покажувач). Кај системскиот повик sendto повратен код ОК означува дека пакетот е успешно испратен, но тоа не значи дека е и успешно примен од дестинацијата.

За секој отворен сокет кој е успешно врзан со порта на локалната машина, апликацијата која го користи може да ги повика системските повици sendto и recvfrom толку пати колку што има потреба.

За комуникација со сокети кои се претходно поврзани со дестинацијата, можат да се користат и системските повици read и write, кои се користат за I/O операции со фајлови.

Системски повици за читање/запишување кои користат повеќе бафери

Системот 4BSD обезбедува системски повици за читање и запишување во сокети кои користат повеќе бафери кои не се континуелни.

int writev(int fd, struct iovec iov[], int iovcount);

int readv(int fd, struct iovec iov[], int iovcount);

int sendmsg(int socket, struct msghdr msg[], int flags);

int recvmsg(int socket, struct msghdr msg[], int flags);

Првите два системски повици writev и readv се наречени gather write и scatter read, и се користат и за фајл и за сокет дескриптори (за сокети кои се веќе поврзани со дестинацијата) за I/O операции со користење неконтинуирани бафери. Баферите кои се користат се референцираат како iov[0], iov[1], итн. Структурата iovec се користи за сместување на почетната адреса на баферот и на неговата големина и е дефинирана во <sys/uio.h> на следниов начин:

```
struct iovec {
        caddr_t iov_base;     /* pocetna adresa na baferot */
        int       iov_len;       /* dolzina na baferot vo bajti */
}
```

Системските повици writev и readv се воведени за да се избегне повеќекратно повикување на write и read во одредени случаи.

Системските повици sendmsg и recvmsg се најопшти повици за I/O операции со сокети. Како аргумент во овие повици се јавува структурата msghdr која е дефинирана во <sys/socket.h>.

```
struct msghdr {
        caddr_t       msg_name;          /* opcionalna adresa */
        int             msg_namelen;       /* golemina na adresata */
        struct iovec  *msg_iov;          /* scatter/gather pole */
        int             msg_iovlen;        /* broj na elementi vo msg_iov */
        caddr_t       msg_accrights;     /* dozvola za prakanje/primanje */
}
```

Првите два аргументи во структурата msghdr се адресата каде што се сместува пораката што се чита/запишува и нејзината големина. Третиот аргумент е структурата која ги дефинира неконтинуелните бафери, а четвртиот аргумент го специфицира

бројот на баферите кои се користат. Последниот аргумент од msghdr служи за предавање и примање на правата за пристап кон објектите управувани од системот (уреди, фајлови, сокети) меѓу процесите во системот (4.3BSD поддржува само пренесување на права за пристап кон фајл од еден процес на друг преку пренесување на фајл дескрипторите преку Unix доменот).

3.4.5 Системски повик listen

Овој системски повик се користи од страна на сервер кој е конекциски ориентиран (TCP), а кој е спремен да прими повици од клиенти. Дефиницијата на повикот е:

int listen(int socket, int maxwaiting);

Овој системски повик обично се извршува по системските повици socket и bind, а пред системскиот повик accept. Првиот параметар е сокет дескрипторот кој се однесува на веќе креиран сокет, додека вториот аргумент maxwaiting го специфицира максималниот број на барања за поврзувања од клиенти кои можат да се баферираат (да не бидат одбиени) додека се чека серверот да го изврши повикот accept. Типична вредност (а воедно и максимална, барем за 4.3BSD) за maxwaiting е 5.

3.4.6 Системски повик accept

По извршување на системскиот повик listen, погоре опишан, TCP серверот го повикува системскиот повик accept, дефиниран во <sys/types.h> и <sys/socket.h>.

int accept(int socket, struct sockaddr *fromaddrptr, int *address_length);

Овој системски повик го прифаќа првото барање од некој клиент кој чека и креира нов сокет со исти особини како сокетот кој е отворен и кој го прифатил

барањето за поврзување. Вториот аргумент во системскиот повик е покажувач кон адресна структура во која се сместува адресата на повикувачот клиент. Третиот аргумент во accept е покажувач кон цел број кој ја покажува големината на адресната структура во број на бајти.

Кога е повикан повикот accept, ако нема барања кои чекаат да бидат опслужени, тој го блокира повикувачот (серверот) се додека не се појави барање за поврзување.

Двата аргументи fromaddrptr и address_length служат за да ја добијат адресата на клиентот кој бара да се поврзе со серверот. Имено, серверот кој го повикува повикот во address_length ја сместува вредноста за големината на адресната структура кон покажува покажувачот fromaddrptr, а по извршување на системскиот повик accept се заменува оваа вредност со вистинскиот број на бајтови сместени во адресната структура. Бидејќи адресните структури за Интернет се со фиксна големина од 16 бајти, тогаш вредноста која ја сместува повикувачот на accept во address_length е 16 и тоа е исто така и бројот на бајти кои ќе ги врати системскиот повик во fromaddrptr. Значи, овој системски повик враќа три вредности: цел број за индикација на грешка или нов сокет дескриптор ако е се во ред, адресата на процесот на клиентот (fromaddrptr) и големината на адресата во бајти (address_length).

По прифаќањето на барање на клиент за поврзување почетниот сокет дескриптор креира дете процес (child) идентификуван преку ново креиран сокет дескриптор кој ќе продолжи во комуникацијата на accept со клиентот што се поврзува со серверот, а почетниот сокет дескриптор кој го рефенренцира татко процесот (parent) ќе продолжи да чека барања за поврзувања од нови клиенти.

3.4.7 Терминирање на комуникацијата преку сокет, системски повици close и shutdown

Кога сесијата на пренос на податоци преку некој сокет е завршена, тој едноставно може да се затвори со системскиот повик close.

int close(int socket);

За UDP сокетите овој системски повик ја ослободува портата за која тие биле врзани претходно.

Кога се работи за TCP сокети, тогаш со повикување на close се затвара комуникацијата во двете насоки, пред да се ослободи портата за која е врзан сокетот. Во случај на TCP, по повикување на системскиот повик close, recv повиците од други хостови кон хостот кој штотуку го затворил сокетот за комуникација ќе резултира во враќање на вредност 0, што индицира дека прекин на комуникацијата кој настанал на другиот крај од врската меѓу хостовите. Ако се испрати send кон хост кој претходно ја затворил комуникацијата, send ќе врати вредност -1. Препорачливо е при завршување на комуникацијата апликацијата да го повика close за да го затвори сокетот со цел TCP врската правилно да терминира на двата краја.

TCP сокетите можат да се затворат и само во една насока на комуникационата врска со користење на системскиот повик shutdown.

int shutdown(int socket, int how);

Аргументот how служи за специфицирање на насоката на комуникација која треба да се затвори. Кога се специфицира how да биде 0, тогаш се оневозможуваат повиците recv, односно оневозможено е читањето од сокетот. Ако how има вредност 1, тогаш запишувањето во сокетот е оневозможено (оневозможен е повикот send кон тој сокет). Специфицирање вредност 2 за how ќе ја оневозможи комуникацијата во двете насоки. Меѓутоа, за да се затвори сокетот мора да се повика системскиот повик close.

3.4.8 Рутини кои се користат при програмирање на сокети

Има библиотечни рутини кои се користат при програмирањето на сокети, а не се составен дел од библиотеката за сокети. Имено, рутините се разликуваат од системските повици по тоа што тие се однесуваат како процедурите кои ги креира програмерот во програмот, додека системските повици комуницираат директно со кернелот на оперативниот систем. Во овој дел ќе се задржиме на рутините кои најчесто се употребуваат во програмирањето на сокети.

Рутини за претварање на Интернет адресите од децимален формат во бинарен и обратно

Интернет адресите обично се запишуваат во децимална форма како четири цели броеви со вредности меѓу 0 и 255, разделени меѓусебе со точки. За претварање на адресата во бинарен облик се користи рутината inet_addr.

unsigned long inet_addr(char *string);

Аргументот string е покажувач кон стринг кој ја претставува Интернет адресата во децимална нотација. Рутината inet_addr враќа 32-битна вредност со мрежен редослед на бајтите. Вака добиената вредност може да се користи во аргументот sin_addr.s_addr од структурата sockaddr_in. Во случај на грешка функцијата враќа вредност -1 (на пример, ако не може стрингот да се интерпретира како 4 цели броеви одделени со точки).

Обратната функција, која што конвертира 32 битна вредност (што претставува Интернет адреса со мрежен редослед на бајти) во стринг е inet_ntoa.

char *inet_ntoa(struct in_addr inaddr);

Оваа функција враќа покажувач кон стринг во кој се сместува Интернет адресата во децимална форма со точки. Меѓутоа, при секое повикување на inet_addr, таа го враќа истиот покажувач, така што препорачливо е да се копира стрингот кој ја содржи Интернет адресата во друг бафер пред да се повика повторно функцијата. Дефиницијата на овие две рутини е во <sys/socket.h>, <netinet/in.h> и <arpa/inet.h>.

Рутини за менување на редоследот на бајтите

Различни компјутери може да ги сместуваат бајтите со различен редослед во меморијата на машината. Да се потсетиме, мрежниот редослед на бајтите е Big Endian. За приспособување на редоследот на бајтите во пораките кон редоследот на бајти усвоен при комуникација преку мрежа, како и обратно, за конвертирање од мрежен редослед на бајтите во редослед на бајтите кој го користи хостот, постојат неколку рутини дефинирани во <sys/types.h> и <netinet/in.h>.

unsigned long htonl(unsigned long ul); 32-битна вредност од hbo во nbo
unsigned long ntohl(unsigned long ul); 32-битна вредност од nbo во hbo
unsigned short htonl(unsigned short ul); 16-битна вредност од hbo во nbo
unsigned short ntohs(unsigned short us); 16-битна вредност од nbo во hbo

Целобројните вредности (како што се адресните информации) мора експлицитно да бидат конвертирани од/во мрежен редослед на бајтите. Токму тука овие функции наоѓаат примена. На машините кои користат Big endian редослед на бајти, како што се Sun Sparcs и Motorola процесорите, овие рутини едноставно ја враќаат вредноста која е специфицирана како аргумент. Кај други машини, како што се процесорите Intel x86 и било кој систем кој работи во околина на Windows NT, овие рутини вршат прераспределба на бајтите. На повеќето машини денес, функцијата htons е еквивалентна на функцијата ntohs. Меѓутоа, за идните 64-битни системи оваа еквивалентност не мора да важи.

Рутини за операции со бајти

Во адресните структури кои се користат има повеќебајтни полиња со кои треба да се работи. Некои од овие полиња не се C компатибилни (имено C стринговите секогаш завршуваат со NULL бајт, а во овој случај стринговите може да имаат NULL бајти во својот состав кои немаат значење - крај на стринг), па за таа цел се употребуваат одредени рутини. Ќе разгледаме три такви рутини кои се дефинирани за 4.3BSD.

bcopy (char *source, char *destination, int length);

bzero(char *destination, int length);

int bcmp(char *ptr1, char *ptr2, int length);

Функцијата bcopy копира length бајти од еден бафер (source) во друг (destination). Функцијата bzero запишува одреден број (length бајти) null бајти во

специфицираната дестинација. Функцијата bcmp споредува два стринга, при што враќа вредност 0 кога двата стринга се идентични или ненулта вредност во спротивен случај.

3.4.9 Други корисни системски повици при работа со сокети

Опции на сокетот, системски повици getsocketopt и setsocketopt

За да се овозможи контрола врз сокетот од страна на апликацијата која го креирала (на пример, за поставување на интервалите на чекање кај сокетите со ретрансмисија, за одредување на големината на баферите и сл.) се создадени два системски повици:

int getsockopt(int socket, int level, int optionname, char *optval, int *optlength);

int setsockopt(int socket, int level, int optionname, char *optval, int optlength);

Аргументот socket го специфицира сокетот за кој е потребно да се добие информацијата. Вториот аргумент level кажува кој треба да ја интерпретира опцијата: самиот сокет или протоколите од подолните нивоа во мрежната архитектура (пр. TCP/IP). Аргументот optionname ја специфицира опцијата на која се однесува барањето. Четвртиот параметар е покажувач кон променлива која ја содржи вредноста на опцијата која треба да се постави за сокетот со системскиот повик setsockopt, односно во неа се запишува вредноста за опцијата која е резултат на повикот getsockopt. Последниот аргумент во овие системски повици ја специфицира големината на променливата за повикот setsockopt, додека за getsockopt тој пред повикот е сетиран на големина на променливата, а по неговото повикување во него се сместува големината на податоците сместени во променливата optval.

Постојат два типа на опции: бинарни опции (овозможуваат или оневозможуваат одредени особини на сокетите) и опции кои се идентифицираат со одредена вредност која што може да се постави или да се анализира (се користат за пренесување на одредена вредност помеѓу корисничкиот програм и системот). Поединечните опции нема да бидат разгледани овде.

Одредување на локалната и оддалечената адреса при комуникацијата

Новокреираниот процес понекогаш има потреба да ја дознае дестинационата адреса со која се поврзува сокетот или да ја одреди локалната адреса на сокетот. За таа цел постојат два системски повици, дефинирани во <sys/types.h> и <sys/socket.h>.

int getpeername(int socket, struct sockaddr *peer, int *addrlength);

Системскиот повик getpeername ја враќа адресата на ентитетот на оддалечениот крај со кој се поврзува процесот, односно ја враќа оддалечената адреса и оддалечениот процес кои ги сместува во адресната структура кон која покажува peer. Аргументот addrlength е покажувач кон цел број кој ќе ја содржи големината на адресната структура.

int getsockname(int socket, struct sockaddr *peer, int *addrlength);

Системскиот повик getsockname ја сместува во адресната структура кон која покажува peer локалната адреса поврзана со сокетот. Дискусијата за останатите параметри е иста како за getpeername.

Добивање на името на хостот и на неговиот домен

Системскиот повик gethostname го копира името (до nlength бајти) за идентификација на хост од локалниот компјутер во поле од карактери одредено со покажувачот length.

int gethostname(char *name, int length);

Следниот системски повик gethostbyname врз основа на името на доменот (domain) враќа покажувач кон структура hostent која содржи информации за тој хост. Аргумент во повикот е покажувач кон стринг во кој е сместено името на доменот. Во hostent структурата се сместуваат информации како што се: официјалното име на хостот,

лист на алиаси, типот на адресата на хостот, нејзината должина и листа на една или повеќе адреси за хостот.

struct hostent *gethostbyname(char *stringHost);

Форматот на структурата hostent е следниот:

```
struct host
        char *h_name;                   /* oficijalno ime na hostot */
        char *h_aliases;                /* lista na aliasi */
        short h_addrtype;               /* tip na adresata na hostot */
        short h_length;                 /* golemina na adresata */
        char **h_addr_list;             /* lista na adresi */
}

#define h_addr h_addr_list[0]       /* prvata adresa vo listata */
```

Првата Интернет адресата се содржи во првите 4 бајти на h_addr_list. Понатаму може да се користи h_addr како референца кон адресите во листата. За Интернет адреси, полето на покажувачи h_addr_list[i] за i=0, i=1, итн, не се покажувачи кон карактери туку покажувачи кон структури од типот in_addr.

Покрај тоа, освен по име, даден хост може да се референцира и по адресата со користење на системскиот повик gethostbyaddr, каде што наместо името се специфицира адресата на хостот.

Системски повик select за истовремена работа со повеќе сокети

Системскиот повик select е дефиниран со:

int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, const struct timeval *tineout);

Кога апликацијата ќе го повика системскиот повик recv или recvfrom тогаш апликацијата се блокира додека не пристигнат податоци за сокетот. Апликацијата може

да работи нешто корисно додека дојдовниот поток на податоци е празен. Друг проблем е ако апликацијата прими податоци од повеќе сокети. Од друга страна со повикување на системските повици recv или recvfrom на сокетот кој нема податоци на својот влез се спречува примањето на податоци од другите сокети. Овој проблем на постоење на повеќе сокети од кои може да чита апликацијата се решава со воведувањето на системскиот повик select, кој што овозможува програмот да ги гледа сите сокети и да види дали се тие достапни за неблокирачки операции на читање и запишување на податоци. Аргументите во select се со следните значења:

- nfds е бројот на сокети кои треба да се разгледуваат. Дескрипторите кои се проверуваат се означуваат од 0 до nfds-1.
- readfds, writefds се покажувачи кон множество на сокет или фајл дескриптори кои можат да бидат иницирани за неблокирачко читање/запишување. Вредност NULL индицира празно множество.
- exceptfds е покажувач кон множество на дескриптори кои треба да бидат проверени за посебни услови.
- timeout е покажувач кон временска структура timeval која ја содржи информацијата колку долго select треба да ги надгледува сокетите кои се специфицирани во множеството сокети за select за влезни/излезни операции. Ако вредноста на timeout е 0, овој системски повик веднаш ќе одговори по негово повикување од програмот. Ако timeout аргументот е NULL тогаш тој ќе ја блокира апликацијата се додека барем еден од сокетите не е достапен за операции на читање/запишување. Во спротивен случај, кога е селектирана вредност различна од претходните две, тогаш select ќе одговори кога ќе помине времето кое е специфицирано со структурата чија адреса е timeout или кога барем еден дескриптор ќе биде спремен за влезно/излезни операции.

Вредноста што ја враќа select кон апликацијата е број на сокети кои се спремни за I/O операции или ако поминало времето специфицирано со timeout без да се појави сокет спремен за читање/запишување ја враќа вредноста 0. За манипулација на дескрипторите се користат неколку макроа:

FD_SET(s, *set) Го отстранува дескрипторот s од множеството set.

FD_ISSET(s, *set)	Има ненулта вредност ако s е член на множеството дескриптори set, 0 во спротивно.
FD_SET(s, *set)	Додава дескриптор s во множеството set.
FD_ZERO(*set)	Го поставува множеството set да биде NULL множество.

Процесот прво ги креира сите сокети кои му се потребни, а потоа го користи select за да специфицира кои од нив први да бидат спремни за I/O операции.

Асинхрони I/O операции со сокети

Накратко ќе ги разгледаме асинхроните влезни/излезни операции со сокети. Асинхроните I/O операции му овозможуваат на процесот да му каже на кернелот на системот кога некој дескриптор е спремен за I/O операции. При асинхрон I/O кернелот комуницира со корисничкиот процес преку сигналот SIGIO. За асинхрон I/O, процесот мора да ги изврши следните три чекори:

1. Процесот мора да добие идентификатор (handler) за SIGIO сигналот, а тоа се постигнува со системскиот повик signal.
2. Процесот мора да го постави идентитетот (ID) на процесот или идентитетот (group ID) на групата на која припаѓа процесот за да може да се прими SIGIO сигналот, што се постигнува со користење на системски повик fcntl (служи за поставување на опции на отворен фајл или сокет) со F_SETOWN командата.
3. Процесот мора да го овозможи асинхрониот I/O преку системскиот повик fctnl, со користење на командата F_SETFL со аргумент FASYNC.

Веќе го спомнавме системскиот повик fcntl, кој заедно со системскиот повик ioctl се користат за сетирање на особините на сокет или фајл. Овде нема да се задржуваме на овие системски повици, бидејќи се надвор од доменот на интерес на оваа книга за Интернет технологии.

3.5 Дискусија

Во оваа глава се запознавме со поимот за сокети, како и со начините за нивно програмирање со користење на системски повици. Бидејќи оригиналниот сокет интерфејс бил развиен на UNIX оперативен систем, кој бил напишан во "c" програмскиот јазик, и изложените програмски структури кои се користат за сокети во оваа глава се прикажани преку синтаксата на овој програмски јазик. Самото креирање на API, што е всушност интерфејс на дадена апликација кон Интернет протоколниот стек во дадена машина, се нарекува и мрежно програмирање. Секако, програмирањето на API може да се изврши и со друг програмски јазик (на пример, C++, Java итн.)

Во Интернет, сите сервиси (вклучувајќи го и говорот преку IP) претставуваат апликации во даден оперативен систем (Windows, Unix, Symbian итн.). Во сите оперативни системи во комуникациските терминали (на пример: десктоп компјутери со мрежни картички – Ethernet или Wi-Fi или WiMAX, мобилни терминали, PDA – Personal Digital Assistants итн.) за да се оствари комуникација преку Интернет неопходно е да бидат имплементира Интернет протоколите – IP на мрежно ниво, TCP и UDP на транспортно ниво. Самиот сервис, дали тоа ќе биде веб, или видео преку IP, или Интернет телефонија, електронска пошта или Telnet, или некој друг тип сервис, е работа само на апликациите кои се наоѓаат на апликациското ниво на страна на клиентите и серверите.

Имајќи во предвид дека развојот на телекомуникациите оди во насока сите сервиси, вклучувајќи ја и телефонијата (говорот) како фундаментален телекомуникациски сервис, како и дигиталната телевизија и радио, да се пренесуваат преку Интернет (секако, во таков случај е потребна и одредена имплементација на поддршка за обезбедување на квалитет на одделните сервиси од аспект на бараните протоци во бити/сек, дозволените загуби на пакети и доцнења на пакетите), да се пренесуваат преку Интернет (т.е. преку IP), тогаш може да се согледа големото значење на познавањето на интерфејсот меѓу апликациите и TCP/IP протоколите во Интернет хостовите, без разлика дали се тоа фиксни или мобилни уреди.

Глава 4

Основни Интернет технологии и сервиси

Во овој дел од оваа книга ќе се запознаеме со основните Интернет технологии и сервиси. Во таа група ќе ги вклучиме DNS (Domain Name Server), Telnet, FTP (File Transfer Protocol) и електронската пошта (e-mail). За најзастапената Интернет апликација денес, вебот (WWW – World Wide Web), ќе посветиме посебно поглавје (следната глава) за да се истакне значењето на оваа апликација за развојот на Интернет во глобални рамки.

4.1 DNS

Протоколите на мрежно ниво (Интернет протоколот) користат 32 битни цели броеви (IP адреси). Овие адреси овозможуваат целосно дефинирање на изворот и дестинацијата во пакетите кои се испраќаат преку Интернет. Но, за луѓето како корисници поприфатливо е на машините да им доделуваат имиња кои лесно би се изговарале и памтеле (наместо бројки, кои всушност се користат од страна на машините, клиентите и серверите). Бидејќи корисниците пристапуваат до Интернет преку користење на одредени апликации, природно се наметнува дека корисниците ќе ги впишуваат имињата на машините во апликациите кои ги користат (на пример, во веб клиентот – browser). Бидејќи апликациите се наоѓаат на највисокото ниво според

протоколниот модел, ваквите имиња на машините на Интернет (составени од зборови и букви) ќе ги наречеме високо-нивовски имиња. Понатаму, потребно е да се изврши мапирање на високо-нивовското име на машината со нејзината IP адреса и обратно мапирање на IP адресата со високо-нивовско име. Оваа шема на доделување на имиња е интересна од два аспекта. Прво, бидејќи се користи за доделување на имиња на компјутерите во глобални рамки на ниво на Интернет. Второ, бидејќи користи географски дистрибуиран сет на сервери кои вршат мапирање на имињата во IP адреси. Во случајов (на DNS) како ниско-нивовски адреси се сметаат IP адресите.

Како е можно еден систем за именување да задоволи еден голем и брзорастечки сет на имиња без притоа да има некоја централизирана страна која ќе го администрира? Одговорот е многу едноставен. Станува збор за децентрализирање на системот за именување и дистрибуираната одговорност за мапирање на имињата и адресите.

Партиционирањето т.е. делењето на просторот за именување мора да биде така дефиниран да овозможува ефикасно мапирање на имињата а воедно и да гарантира автономна контрола при доделување на имињата.

Со цел полесно да се сфати поделбата и организирањето на просторот за именување би било згодно да го разгледаме следниов пример. Таа би наликувала како една интерна структура на голема компанија. Значи на врвот би бил главниот извршител кој би имал целосна одговорност. Бидејќи тој како единка не е во можност да врши преглед на сите работи, организацијата мора да се подели на оддели кои ќе ги управуваат одделни менаџери. Главниот менаџер им дозволува ограничена автономност на одделите. Менаџерите во нив имаат право да вработуваат нови кадри да отпуштат, да доделуваат канцеларии итн., без притоа да бараат дозвола од главниот менаџер. Вака се овозможува автономна оперативност на компанијата. Притоа протокот на информации се движи хиерархиски. Пример, доколку на менаџерот му треба телефонски број на некој од работниците, а тој не е во моментот таму, ќе се обрати најпрво до неговите колеги за да го добие истиот. А потоа хиерархиски низ соодветните нивоа. Принципот на организација на системот за доделување на имиња во рамки на Интернет е слично организиран хиерархиски така да наликува на една голема хиерархиска структура на компанија.

Механизмот во TCP/IP кој овозможува хиерархиско именување на компјутерите се нарекува Domain Name System (DNS). DNS се карактеризира со два концептуално независни аспекти. Првиот е апстрактен и ја дефинира синтаксата на имињата како и на правилата за делегирање на авторитет за имињата. Вториот е конкретен и ја

специфицира имплементацијата на дистрибуиран компјутерски систем кој ефикасно ќе врши мапирање на имињата со адресите. Овој систем користи хиерархиска шема на именување позната како имиња на домени (domain names). Името на доменот (domain name) се состои од секвенца на подимиња меѓусебно одделени со точка. Индивидуалните делови од името можат да претставуваат сајтови или групи. Но DNS системот секоја секција од името ја нарекува лабела.

Пример за име на домен е fakultet.univerzitet.edu. Можеме да забележиме дека истото се состои од три лабели (fakultet, univerzitet и edu) одвоени меѓусебно со точки. Секој суфикс на лабела во домен името се нарекува домен.

Во случајов доменот на најниско ниво е fakultet.univerzitet.edu и претставува домен име за веб сајтот на факултетот fakultet на универзитетот univerzitet. Второстепениот домен е univerzitet.edu и претставува домен име за универзитетот univerzitet. Највисокото хиерархиско ниво во овој пример е edu, кој претставува домен име за едукативни организации.

Од примерот можеме да заклучиме како се врши лабелирањето во имињата на домените кај DNS. Најдесната лабела во дадено име го означува доменот на највисоко хиерархиско ниво (tld - top level domen), како што се gov, edu, com, net, org, итн. Така, во примерот, tld на името на домен fakultet.univerzitet.edu е edu. Хиерархијата на домените опаѓа одејќи од десно на лево во дадено име на домен.

Во однос на домените на највисоко ниво (top level domains) разликуваме неколку типови денес, од кои ќе ги издвоиме двата најважни: генеричките TLD и TLD по код на држава. Во генеричките TLD (generic Top Level Domains - gTLD) спаѓаат домените com, edu, gov, info, int, mil, net, org, итн. Во случајот на gTLD има неколку домени што се неограничени на тип на организација (т.н. unrestricted gTLD), а тоа се com, net, org и info. Други се ограничени на типови на организации (на пример, edu се користи исклучиво за образовни организации на сите нивоа, mil се користи исклучиво само за воени организации, итн.). TLD по код на држава (country code TLD - ccTLD) се домени кои се резервирани според државите (на пример, mk за Македонија, uk за Велика Британија, jp за Јапонија, итн.). Првиот Интернет домен се викал arpa, име кое потекнува од времето на ARPANET. Иако првично било замислено да постои само привремено за да се реализира миграцијата од имињата на хостовите во ARPANET кон DNS системот кој денес се користи во Интернет, доменот останал да постои како т.н. инфраструктурен највисок домен кој се користи исклучиво за инфраструктурни цели во Интернет, како што е реверзибилен DNS (reverse DNS). Додека класичното DNS

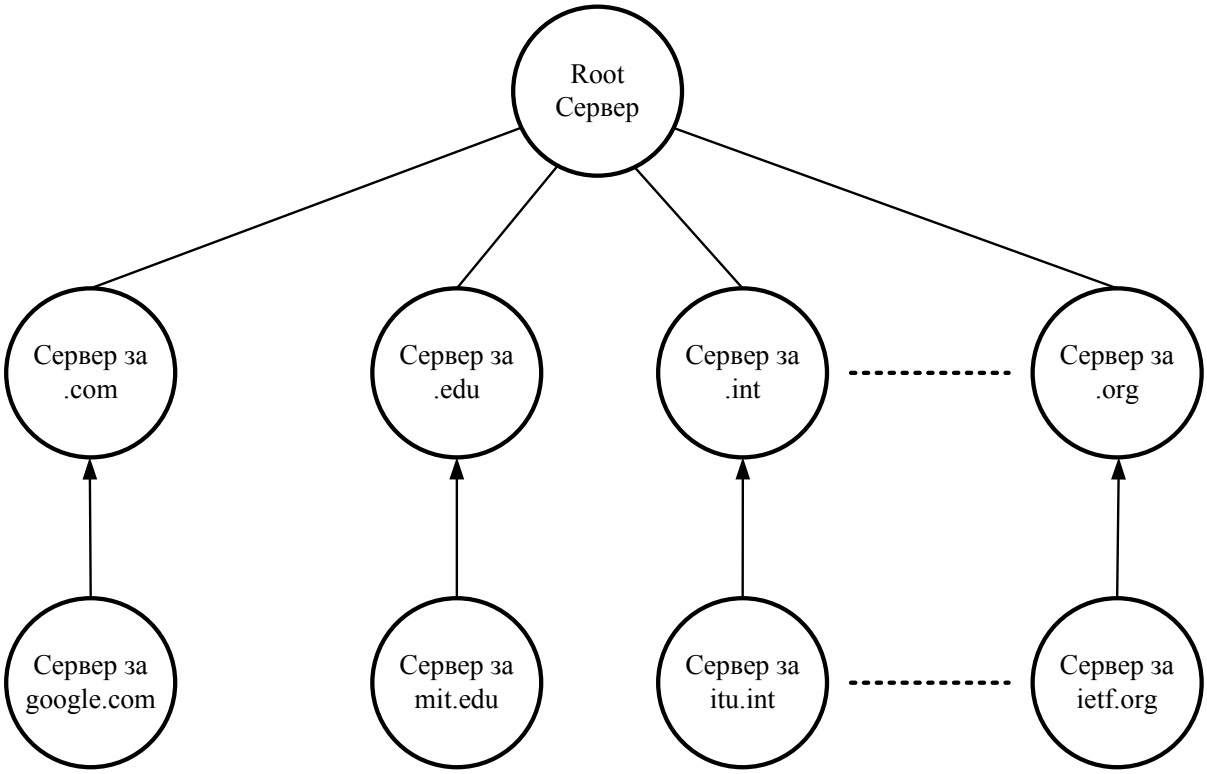
разрешување (т.н. lookup) значи да се одреди IP адресата на даден хост преку дадено име на доменот на тој хост, реверзибилниот DNS значи да се одреди името на даден домен врз основа на позната IP адреса (на пример, ако DNS се користи за телефонија во Интернет, тогаш треба врз основа на телефонскиот број на корисникот треба да се најде IP адресата на уредот кој го користи корисникот за телефонија во Интернет, што е еден пример каде што се користи реверзибилниот DNS). .

4.1.1 Мапирање на домен имињата со адресите

Дистрибуираниот систем кој служи за мапирање на имињата со адресите се состои од сет на сервери кои меѓусебно соработуваат и ги решаваат проблемите поврзани со мапирањето. Овој систем е доста ефикасен бидејќи повеќето од имињата се мапираат локално. Системот не е ограничен, а е и доверлив така да испад на една машина нема да доведе до пад на системот или негова дисфункционалност.

Системот за мапирање на имиња на домени (Domain Name System – DNS) се состои од независни кооперативни системи кои се нарекуваат name сервери (сервери за име). Name server претставува серверска програма која овозможува преведување на името во адреса т.е овозможува, мапирање на името на доменот со IP адреса. Вообичаено серверскиот програм се извршува на посебен процесор па така целата машина се нарекува name server. Клиентскиот софтвер се нарекува name resolver и користи еден или повеќе name server-и кога врши преведување на името. Најлесен начин да се разбере работата на domain серверите е да се замисли дека истите се аранжирани во топологија на стебло која кореспондира со хиерархијата на именување, што е илустрирано на слика 4.1.

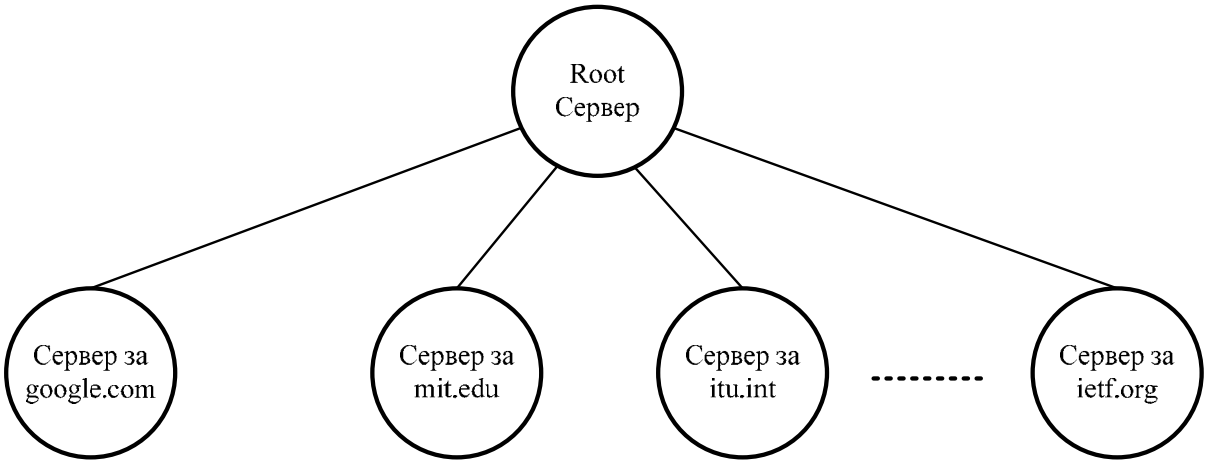
Коренот на стеблото го сочинува сервер кој ги препознава највисоките нивоа на домени т.е. топ нивоа на домени (на пример, edu). За дадено име root server-от може да го одбере правилниот сервер наменет за тоа име. Серверите од следното ниво ги знаат одговорите за еден домен од највисоко ниво. Серверите на ова ниво знаат кои сервери можат да ги разрешат секои од поддомените кои припаѓаат под нивниот домен. На третото ниво од стеблото се наоѓаат name серверите кои ги обезбедуваат одговорите за поддомените. Концептуално ова стебло продолжува да се разгранува со по еден сервер на секое следно ниво за кое е дефиниран претходно поддомен.



Слика 4.1 Приказ на Domain name сервери во топологија на стебло

Линковите прикажани на концептуалното стебло не укажуваат на физичките мрежни врски. Тие укажуват на тоа кој друг name server го знае дадениот сервер и контактира со него. Серверите може да бидат поставени на избрани локации во Интернет мрежата. Според ова стеблото со сервери претставува абстракција која го користи Интернет за комуникација.

Доколку серверите во Domain системот функционират на поедноставен начин каков што разгледавме тогаш врската помеѓу конектирањето и автентикацијата ќе биде многу едноставна. Кога ќе се одобри некој поддомен тогаш потребно е да се постави domain name сервер за тој поддомен и соодветно да се поврзе во стеблото. Во пракса врската помеѓу хиерархијата на именувањето и стеблото не е толку едноставна како што е прикажана во моделов. Обично стеблото со сервери има неколку нивоа бидејќи еден физички сервер може да ги содржи сите информации за поголеми делови од хиерархијата за именување. Воглавно кога станува збор за поголеми организации тие најчесто собираат информации од сите нивни поддомени во единствен сервер. Пореална организација на сервери кои се вклучени во хиерархијата за именување е прикажана на сликата 4.2.



Слика 4.2. Пореална организација на сервери

Root серверот содржи информација за root како и за top-level домените. Секоја организација си користи сервер за своите именувања.

4.1.2 Преведување на имињата на домени (Domain Name Resolution)

Концептуалното стебло прави врските меѓу серверите да се разберат поедноставно, но сепак не разоткрива некои пофини детали. Тие може да се објаснат со name resolution алгоритмот. Концептуално Domain name resolution се извршува од врвот кон дното на стеблото. Започнува со root name серверот и се проследува се до листовите на дрвото. Постојат два начина на користење на Domain Name System-от, со контактирање на серверите еден по еден или пак со барање да name сервер системот ја изврши целата транслација. Во друг случај клиентскиот софтвер формира domain name барања кои треба да се разрешат. Барањето содржи информации за: декларација на класата на името, типот на одговор кој е посакуван, како и кодот кој специфицира дали name серверот треба да го преведе името комплетно. DNS клиентот (на корисничка страна) го испраќа барањето до name серверот за да го разреши. Кога domain name серверот ќе добие барање, проверува со цел да види дали името припаѓа во неговиот поддомен. Доколку е ова случај тогаш врши трансформирање на името во адреса зависно од неговата база на податоци и го додава одговорот на барањето пред да го испрати назад до корисникот. Доколку name серверот неможе да го реши името комплетно, проверува да види каков тип на интеракција специфицирал клиентот.

Доколку клиентот побарал целосно преведување серверот контактира со domain name сервер кој може да го разреши проблемот и да го врати одговорот на клиентот (рекурзивно решение). Доколку корисникот бара нерекурзивно решение name серверот нема да може да обезбеди одговор. Генерира одговор кој го специфицира name серверот кој корисникот треба да го контактира со цел да го реши името.

Како клиентот го наоѓа името на серверот кој треба да го отпочне барањето?

Како name серверот ги наоѓа name серверите кои го знаат одговорот на барањето на кое тој не може да одговори?

Клиентот мора да знае како да исконтактира барем со еден name сервер. DNS серверот (или серверите) се дефинираат мануелно од страна на корисникот во делот за конфигурација на TCP/IP протоколите во оперативниот систем или се дефинираат и поставуваат автоматски преку динамичкото доделување на IP адреса на корисничкиот хост со користење на Dynamic Host Configuration Protocol - DHCP (опишан во глава 2). Со цел да се осигура дека domain name серверот може да контактира со други, DNS системот бара секој сервер да ја знае адресата на најмалку еден root сервер. DNS сервер може да ја знае адресата на сервер кој е домен над него.

4.2 FTP File Transfer Protocol

Преносот на податоци и датотеки претставува една од најексплоатираните TCP/IP апликации. Стандардните протоколи за пренос на податоци постоеле уште при ARPANET пред да стане оперативен TCP/IP. Овие рани верзии на софтвер за пренос на податоци подоцна еволуирале во стандардот денес познат како FTP.

4.2.1 Карактеристики на FTP

Авторизацијата, именувањето меѓу хетерогени машини кога се остварува меѓусебна комуникација го прави овој протокол сложен. Самиот протокол нуди повеќе можности кои се затскриени од неговата главна функција пренос на податоци:

- Интерактивен пристап

FTP е дизајниран за да се користи од програми при што повеќето нивни имплементации на корисниците им овозможуваат лесно да комуницираат со оддалечените сервери.

- Format Specification

  FTP му дозволува на клиентот да го дефинира форматот на податоците кои се зачувуваат.

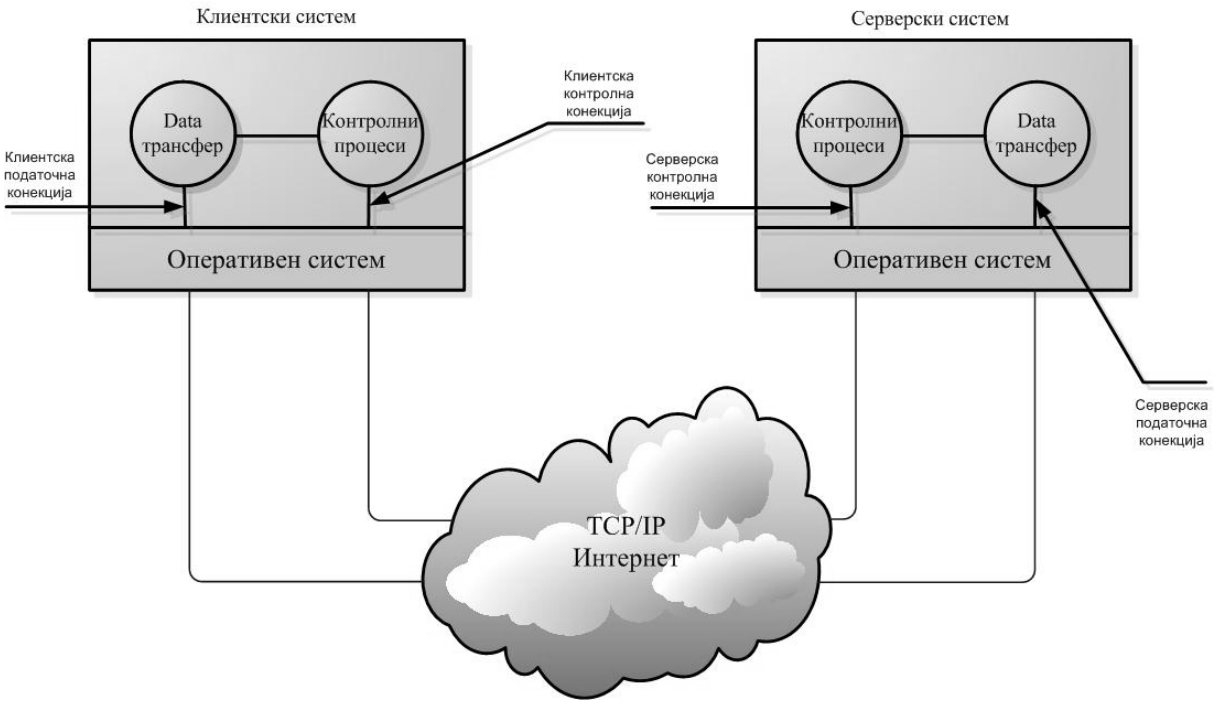
- Контрола при автентикација

  FTP бара клиентите да извршат нивно авторизирање со испраќање на корисничко име (login name) и лозинка (password) до серверот пред да отпочне процесот на размена на податоци. Во случај да побараните податоци не се валидни нема да му биде дозволен пристапот на истиот.

4.2.2 FTP модел

Повеќето FTP сервери поддржуваат истовремен пристап на повеќе корисници. Клиентите користат TCP протокол за поврзување со серверот. Главниот сервер ги процесира барањата и креира slave процеси со цел да се опслужи секоја конекција. Овие slave процеси не ги вршат сите неопходни пресметки. Slave процесите ги прифаќаат и опслужуваат контролните конекции. Контролните конекции служат за пренос на команди кои му укажуваат на серверот кој фајл треба да го пренесе. Конекциите кои го користат TCP како транспортен протокол за комуникација служат исклучиво за пренос на податоци. Обично и клиентот и серверот креираат посебни процеси со цел да се справат со преносот на податоци.

Обично деталите за тоа како се одвива процесот на пренос на податоци зависат од оперативниот систем. Основите на концептот се дадени на слика 4.3.



Слика 4.3 FTP клиент и сервер помеѓу кои има воспоставено TCP контролна конекција како и одделна TCP конекција која служи за трансфер на податоци помеѓу нив

Како што е прикажано на сликата клиентскиот контролен процес се конектира со серверскиот контролен процес користејќи TCP конекција. Процесите за пренос на податоците си користат своја посебна TCP конекција. Контролниот процес и контролната конекција се активни се додека корисникот ја држи активна TCP конекцијата. Како и да е, FTP воспоставува нова врска за пренос на податоци за секој пренос кој треба да се оствари. Повеќето имплементации креираат нов пар на процеси за пренос на податоци како и воспоставување на нова TCP конекција кога серверот треба да испрати податоци до клиентот.

Конекциите за пренос на податоци како и процесите за пренос на податоци кои тие ги користат можат да се креираат динамички по потреба, за разлика од контролните конекции кои се перманентни и се доделуваат на ниво на сесија. Штом ќе исчезне контролната конекција тоа значи дека сесијата е терминирана, а софтверот на двата краја на терминалните уреди ги прекинува сите процеси поврзани со пренос на податоци.

Клиентските имплементации кои се извршуваат на компјутер без оперативен систем поддржуваат повеќекратни процеси кои може да имаат помалку комплексна структура.

Доделување на броеви за TCP порти

Кога корисникот воспоставува почетна комуникација со серверот тој користи случаен локално доделен број на порт, но со серверот комуницира со well-known port 21. Сервер кој користи еден протоколен порт може да прифати повеќе конекции од повеќе клиенти, бидејќи TCP ги користи двата краја за да изврши идентификација на конекцијата.

Се поставува прашање кога контролниот процес креира нова TCP конекција за даден пренос на податоци? Кои протоколни порт броеви ќе ги користи? Очигледно е дека нема да може да се искористат истите порт броеви кои се користат за контролната конекција. Клиентот си побарува неискористен број на порт кој ќе се искористи за TCP конекција при процесот на пренос на податоци на серверската машина. Процесот на пренос на податоци кај серверската машина ги користи т.н well-known портови кои се резервирани за FTP комуникација (порт 20).

Како корисникот го гледа FTP

Корисникот го гледа TCP како интерактивен систем. Корисникот обично ги извршува следниве команди периодично: чита линија за влез и ја извршува командата со соодветните аргументи.

Листа на команди кои корисникот може да ги користи се:

!	**cr**	**macdef**	**proxy**	**send**
$	**delete**	**mdelete**	**sendport**	**status**
account	**debug**	**mdir**	**put**	**struct**
append	**dir**	**mget**	**pwd**	**sunique**
ascii	**disconnect**	**mkdir**	**quit**	**tenex**
bell	**fom**	**mls**	**quote**	**trace**
binary	**get**	**mode**	**recv**	**type**
Bye	**glob**	**mput**	**remotehelp**	**user**
case	**hash**	**nmap**	**rename**	**verbose**
cd	**help**	**ntrans**	**reset**	**?**
cdup	**lcd**	**open**	**rmdir**	
close	**Is**	**prompt**	**runique**	

За да добие подетални информации за некоја команда корисникот во cmd може да отчука help откако претходно ќе отчука ftp. Од името на секоја команда може интуитивно да се согледа нејзината употреба кај ftp.

Споменавме дека со цел да се направи FTP посигурен без да се помине процедурата на автентикација не може да се пристапи до одредени датотеки. Со цел да се овозможи пристап до јавни фајлови многу сајтови овозможуваат анонимни FTP сесии. Анонимниот FTP пристап значи дека на корисникот не му се потребни посебно доделено корисничко име и лозинка. Корисникот за корисничко име внесува anonymous, а за лозинка guest. Серверот во случајов ќе го дозволи пристапот, но истиот ќе биде ограничен на оние фаилови кои се јавни. Корисниците обично користат само неколку FTP команди за да воспостават конекција и да преземат фајл. Пример за FTP комуникација во случај кога некој корисник ставил online копија на текст во фајл tcpbook.tar во подфолдерот pub/comer кој пак е сместен на компјутерот ftp.fakultet.univerzitet.edu Корисник кој е логиран на друг сајт може да добие копија од истиот извршувајќи ги следниве команди:

```
% ftp ftp.fakultet.univerzitet.edu
Connected to makedon1.fakultet.univerzitet.edu.
220 makedon1.fakultet.univerzitet.edu FTP server ready.
Name (ftp.fakultet.univerzitet.edu:usera): anonymous
331 Guest login ok, send e-mail address as password.
Password: guest
230 Guest login ok, access restrictions apply.
ftp> get pub/janevski/Internet_tehnologii_kniga.pdf bookfile
200 PORT cortunand okay.
150 Opening ASCII mode data connection for tcpbook-tar (3754682 bytes)
226 Transfer complete.
3754682 bytes received in 9.53 seconds (3.9e+02 Kbytes/s)
ftp> close
221 Goodbye.
ftp> quit
```

Во овој пример корисникот ја дефинира машината ftp.fakultet.univerzitet.edu како аргумент во FTP командата така да клиентот автоматски отвара конекција и го

отпочнува процесот на авторизација. Бидејќи станува збор за анонимна FTP сесија ги внесува општите username и password. Потоа следи барањето за соодветниот фајл со користење на командата get. Во случајов get командата е следена од два аргументи кои го дефинираат далечното име на фајлот како и име на локалната копија. Оддалеченото име на фајлот е pub/comer/tcpbook.tar. а локалната копија е сместена во bookfile. Штом ќе заврши преносот на податоци корисникот испишува close со цел да ја терминира конекцијата со серверот, а quit за да го напушти клиентот.

Контролните и пораките за грешка меѓу FTP клиентот и серверот започнуваат со трицифрен број кој е следен од текст. Бројот е наменет за софтверот, а текстот за луѓето. Во примеров е илустрирана една карактеристика на FTP: креирање на нова TCP конекција за трансфер на податоци. Треба да се забележи Port командата во излезниот извештај. Клиентската Port команда укажува на тоа дека нов TCP порт број е побаран за да се искористи при пренос на податоците. Клиентот ја испраќа информацијата за портот до серверот преку контролните конекции, тогаш процесите за трансфер на податоците на двата краја користат нови порт броеви при формирање на конекции. Откако трансферот ќе заврши процесите за пренос на податоци на секој крај ја терминираат конекцијата.

4.3 Телнет

Доверливи, приточни (стрим) протоколи, како TCP, овозможуваат интерактивно користење на оддалечени машини. На пример, замислете конструкција на сервер кој што овозможува сервис за работа со текст од далечина. За да се имплементира управувачки сервис потребен е сервер кој што прифаќа барања за управување со некој фајл и клиент кој што ќе го генерира барањето. За да се стартува сервисот за далечинско управување, корисникот го иницира клиентскиот програм. Корисникот (клиентот воспоставува TCP врска со серверот, а потоа започнува со испраќање на клучни пораки кон серверот. Серверот во обратен правец испраќа одговор кон клиентот.

Како може да се генерализира сервисот за интерактивно далечинско управување? Проблемот во користењето на еден сервер за секој процесирачки сервис е во брзото натрупување на машините со процеси од серверите. Може да се елиминираат

најспецијализираните сервери и да се обезбеди воопштување. Ова се прави преку овозможување на корисникот да воспостави сесија и да се логира на оддалечената машина, а потоа да задава наредби. Со помош на remote login facility, корисниците имаат пристап до сите команди кои што се достапни на оддалечениот систем, а дизајнерите на системот немора да обезбедат посебни сервери.

Се разбира, овозможувањето на логирање од далечина не е едноставно. Компјутерските системи кои што се дизајнирани без да се земе во предвид умрежувањето, очекуваат сесии за логирање исклучиво од директно поврзаната тастатура. На ваков компјутер (машина), за додавање на оддалечен сервер за логирање потребно е да се изврши модификација на нејзиниот оперативен систем. На пример, да разгледаме систем кој што има доделено специјална функција за одредена комбинација од тастери. Ако локалниот систем ја смета комбинацијата: Control+C дека има значење "престани со извршување на моментниот процес", тогаш скоро е невозможно да се пренесе оваа наредба на оддалечениот систем. Ако клиентот ја пренесе Control+C наредбата, ќе настане проблем за прекинување на процесот кај локалната машина.

Покрај техничките потешкотии, систем-програмерите успеале да изградат софтвер за логирање на оддалечен сервер, за повеќето оперативни системи како и апликациски програми кои што се однесуваат како клиенти. Најчесто, клиентскиот софтвер ги препишува значењата на сите копчиња, освен на едено, со што на корисникот му овозможува да комуницира со оддалечената машина, исто како да комуницира преку директно поврзан терминал. Единствениот тастер кој што е исклучок од ова правило, е наредбата за прекинување на врската со локалната машина. Дополнително, некои протоколи за логирање имаат листа на привилегирани хостови кои што можат да се логираат на оддалечената машина без лозинка (password), други протоколи постигнуваат безбедност преку шифрирање на сите трансмисии.

4.3.1 Телнет протокол

TCP/IP протоколниот стек вклучува едноставен терминален далечински протокол, наречен Телнет (Telnet). Телнет му овозможува на корисникот да се логира на оддалечен компјутер преку Интернет. Овој протокол воспоставува TCP конекција, а потоа го пренесува значењето на тастерите од локалниот компјутер на оддалечениот

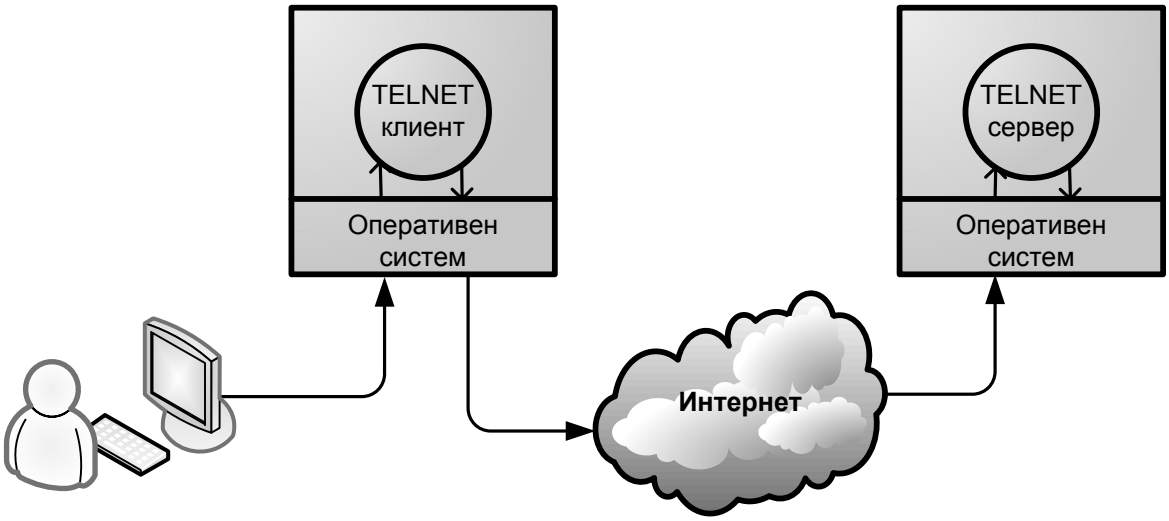
компјутер. Сервисот се нарекува транспарентен бидејќи се добива ефект како наредбите да се внесуваат директно од тастатуртата на оддалечениот компјутер. Телнет исто така пренесува повратна информација од оддалечената машина кон локалната машина.

Иако Телнет не е доволно развиен протокол во споредба со некои други протоколи, сепак е многу користен. Телнет клиентскиот софтвер му овзможува на корисникот да ја идентификува оддалечената машина преку нејзината IP адреса или со назначување на нејзиното име (domain name). Поради користењето на IP адресата, Телнет може да се користи дури и кога не е овозможено идентификувње на името на машината.

Телнет нуди три основни сервиси:

- дефинира виртуелен мрежен терминал, кој што нуди стандарден интерфејс за оддалечени системи. Клиентскиот програм немора да ги познава деталите за сите можни оддалечени системи; тие се конструирани да користат стандарден интерфејс.
- вклучува механизам кој што на корисникот и на серверот им овозможува опции за меѓусебно преговарање, а нуди и множество од стандардни опции (пр: дали информацијата да се пренесува со помош на 7-битни ACII карактери или 8-битно множество).
- двата краја на конекцијата ги третира симетрично. Било кој од нив може да преговара за опциите. Телнет не бара стриктно од влезниот клиент да комуницира преку тастатура, ниту пак го принудува корисникот да ја прикаже излезната информација на монитор.

Како што покажува сликата 4.4 кога корисникот ќе го активира ТЕЛНЕТ, апликацискиот програм на корисничката машина станува клиент. Клиентот воспоставува TCP конекција со сервер, преку кој што ќе комуницира. Штом се воспостави конекција, клиентот ги презема функциите на тастерите од корисничкиот компјутер и ги пренесува кон серверот. Истовремено, клиентот ги презема карактерите кои што се пратени од серверотот и ги прикажува на корисничкиот монитор. Серверот мора да ја прифати TCP конекцијата од клиентот и потоа да размени податоци помеѓу TCP конекцијата и локалниот оперативен систем.



Слика 4.4 ТЕЛНЕТ имплементација клиент-сервер

Во пракса, серверот е покомплексен одколку што е тоа прикажано на сликата, бидејќи мора да управува со повеќе истовремени конекции. Најчесто, главен (master) сервер чека нови конекции и креира нов подреден (slave) сервер за управување со секоја конекција. Затоа, "Телнет-сервер" прикажан на сликата, претставува подреден сервер кој што управува со една одредена конекција. На сликата не е прикажан мастер серверот кој што чека нови барања за воспоставување на конекции. На сликата не се прикажани и останатите подредени сервери кои што управуваат со други конекции.

Го користиме зборот псевдо-терминал за да ја опишеме влезната точка од оперативниот систем која што дозволува старување на програм како Телнет серверот, кој врши пренос на карактери кон оперативниот систем како да доаѓаат директно од тастатура. Доколку оперативниот систем не подржува имплементирање на Телнет, тогаш не е возможно да се изгради Телнет сервер. Ако системот подржува псевдо-терминално разграничување, Телнет серверот може да се имплементира со помош на апликациски програми. Секој подреден сервер поврзува TCP притока од еден клиент до конкретниот псевдо-терминал.

Подесувањето на Телнет серверот да биде програм од апликациско ниво има предности и недостатоци. Најголема предност е тоа што прави модификацијата и контролата да бидат поедноставни. Најголем недостаток е неефикасноста. Секоја тастер-функција патува од корисничката тастатура, преку оперативниот систем, до клиент-програмата на корисникот, од клиент програмата назад преку оперативниот систем и низ Интернет до серверот. Откако ќе стигне до дестинацијата, информацијата

мора да оди преку оперативниот систем на серверот, до апликацискатa програма на серверот, а преку апликацискиот програм на серверот назад до оперативниот систем на серверот кон псевдо-терминалната влезна точка. Конечно, оддалечениот оперативен систем ги пренесува карактерите до апликацискатa програма којашто ја користи клиентот. Во меѓувреме, излезот (вклучувајќи го ехото од оддалечениот карактер кој што е селектиран) патува назад од серверот кон корисникот преку истата патека. Кај повеќето системи, потребна е дополнителна промена на содржината бидејќи оперативниот систем на серверот мора да пренесува карактери од псевдо-терминалот, назад кон друг апликациски програм (на пример: command interpreter). Сепак шемата е практична бидејќи корисниците не пишуваат со голема брзина.

4.4 Електронска пошта (e-mail)

Една од најмногу користените TCP/IP апликации е електронската пошта или е-маил. Електронската пошта се користи повеќе од 20 години. Првиот состав за испраќање на електронска пошта бил едноставен состав за пренос на датотеки во кој првата линија од пораката (или датотеката) ја содржела адресата на примачот.

Брзо биле воочени недостатоците на ваквиот пристап:

- пораките немале своја внатрешна структура, па нивната обработка била тешка;
- било комплицирано испраќањето на пораки на поголема група,
- испраќачот не можел да знае дали неговата порака е испратена или не;
- не било можно да се состави и прате порака која ќе содржи текст, графика, звук итн.

Во 1982 година се објавени передлози од ARPANET за испраќање на електронска пошта. тие предлози се објавено под заедничко име RFC 821 (стандард за размена на пораки помеѓу два компјутери), [17], и RFC 822 (стандард за структурата на пораката), [18]. Од тогаш RFC 821 и RFC 822 се користат како Интернет стандарди, како стандард 10 и 11, т.е. STD 10 и STD 11, соодветно. Подоцна во 2001 год. се заменети RFC 821 и RFC 822 со RFC 2821 и RFC 2822 соодветно, но во литературата и понатаму се користат како референци иницијалните стандарди за електронска пошта, [1].

Основен протокол кој е дефиниран за комуникација меѓу mail серверите eSMTP – Simple Mail Transter Protocol, за брзо и ефикасно испраќање на електронска пошта меѓу TCP/IP хостовите.

SMTP протоколот се однесува на сет од протоколи кои се интерно поврзани меѓусебе. Трите стандарди се следните:

- Стандард за размена на пошта помеѓу два компјутера (STD 10/RFC 821), кој го специфицира протоколот кој се користи за праќање пошта помеѓу TCP/IP хостовите. Тој протокол е SMTP.
- Стандард (STD 11) за обликот и структурата на пораките. Овој стандард се содржи во RFC 822 и RFC 1049. RFC 822 ја опишува синтаксата на заглавјето на поштата и значењето на mail header (заглавје). RFC 1049 опишува каков тип на документ освен ASCII текст може да се користи во телото на mail-от. Името на овој протокол е MAIL.
- Стандард за рутирање (упатување) на поштата користејќи го Domain Name System. Официјалното име на овој протокол е DNS-MX.

4.4.1 SMTP – Simple Mail Transfer Protocol

SMTP е базиран на принцип познат како до крај до крај испорака (end-to-end delivery), бидејќи SMTP серверот се обидува да воспостави контакт со клиентот (примачот) за да ја достави поштата и ја задржува поштата се додека не ја достави.

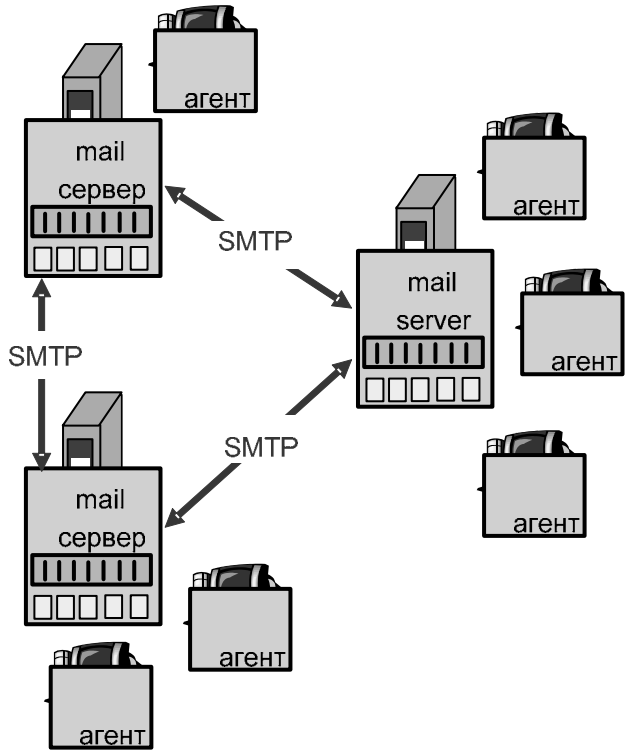
Во различни имплементации, постои можност за размена на пошта помеѓу TCP/IP SMTP mail системот и локално користени mailing системи.

Овие апликации се наречени mail gateways (mail порти) или mail bridges (mail мостови). Испраќајќи маил преку mail gateway се менува end-to-end испораката и притоа SMTP ќе гарантира испорака до mail-gateway хост, а не до одредишниот хост. Кога се користи оваа апликација mail gateway, SMTP end-to-end преносот е host-to-gateway, gateway-to-host или gateway-to gateway. Однесувањето надвор од gateway не е дефиниран со SMTP.

Mail gateway-от ги избира корисниците во регуларно време, испорачувајќи ја поштата која е адресирана до нив и ја одбира поштата што треба да се испрати. Иако ова не е end-to-end испорака, овој систем се покажал како доста корисен.

Секоја порака содржи:

- **Заглавје**, чија структура е дефинирана во RFC 822. Заглавјето на поштата (mail header) е означена со null линија. Null линијата претставува празна (бела) линија.
- **Содржина**, - содржината претставува се после null линијата и се вика тело на пораката. Телото на пораката содржи ASCII карактери (секој карактер означува број од 0 до 127).



Слика 4.5 SMTP модел

Телото од заглавјето е одвоено со null линијата, т.е. линијата која нема ништо пред CRLF (Carriage-Return/Line-Feed). Полето на заглавјето може да се набљудува како една линија ASCII знакови која се состои од field name (име на поле) по кое следи две точки " : " и field-value (вредност на телото) позади кое следи CRLF.
field name: field-value

Телото на полето претставува низа од лексички симболи:

- индивидуални специјални знаци
- стрингови во наводници
- литерали

- коментари во загради

Многу важни вредности на полето се mailboxes т.е. сандачиња. Тие можат да ги имаат следните форми:

- tonij@feit.ukim.edu.mk
- Тони Јаневски< tonij@feit.ukim.edu.mk >
- "Тони Јаневски"< tonij@feit.ukim.edu.mk >

Стрингот Тони Јаневски претставува име на сопственикот на сандачето, tonij е корисничкото име на корисникот на сандачето, feit.ukim.edu.mk е доменот на mail серверот, а целиот стринг tonij@feit.ukim.edu.mk ја претставува email адресата на сандачето. Оваа фаза на email адресирање е блиску поврзана со концептот на Domain Name System (DNS), што е неопходно за mail клиентот да ја одреди IP адресата на mail серверот со цел да воспостави клиент-сервер TCP/IP конекција.

Полиња во заглавието на email порака се следните:

- **To: e-mail** адреса на примачот
- **Cc: e-mail** адреса на корисниците на кои на кои и е пратена копија од пораката.
- **Bcc: e-mail** адреса на корсници на кои и е пратена копија, а останатите примачи не можат да ја видат.
- **From**: корисникот кој ја составил пораката
- **Sender: e-mail** адреса на испраќачот
- **Received**: како пораката стигнала до одредената адреса
- **Return-Path**: се бележи патот до испраќачот

Некои додатни полиња се:

- **Date**: датум и време на испраќањето
- **Reply-To: e-mail** адресата на која ќе се си прати одговор
- **Message-Id**: единствен број на пораката за нејзино референцирање
- In-Reply-To: Message-Id на порака на која се одговорило
- **References**: останати важни Message-Id
- **Subject**: наслов на пораката

Mail командите и одговорите, за разлика од името на корисникот може да се пишуваат со мали и големи букви. SMTP инструкциите се стрингови кои завршуваат со <CRLF> (Carriage-Return/Line-Feed).

```
S: 220 fakultet.edu.mk
C: HELO gmail.com
S: 250 Hello gmail.com, pleased to meet you
C: MAIL FROM: <antonio@xyzmail.com>
S: 250 antonio@xyzmail.com... Sender ok
C: RCPT TO: <dario@fakultet.edu.mk>
S: 250 dario@fakultet.edu.mk ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Zdravo na site,
C: Denes predavanjata ke pocnat vo 14 casot.
C: Pozdrav
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 fakultet.edu.mk closing connection
```

Слика 4.6 Пример на SMTP интеракција

Пренос на електронска пошта

Има три фази на преносот на пораките кај SMTP:

- handshaking (greeting)
- пренос на пораки
- closure

Преносот започнува со инструкцијата MAIL која го идентификува испраќачот. Следува една или повеќе инструкции RCPT која дава информација за примачот. Потоа следува DATA инструкција со која се праќа текстот на поштата (mail data). На крај, индикаторот за крај на mail data го потврдува преносот. Пример за SMTP интеракција е прикажан на слика 4.6. SMTP стандардно ја користи портата 25.

4.4.2 Multipurpose Internet Mail Extensions (MIME)

MIME е стандарден протокол кој се користи при пренос на електронската пошта (e-mail), дефиниран е со RFC 2045, [19]. E-mail е најмногу користена TCP/IP апликација. SMTP е ограничен на 7-бит ASCII текст со максимална големина од 1000

карактери во една линија. Повеќенаменските проширувања за електронската пошта (MIME) се дефинирани за да може со електронската пошта да се пренесуваат податоци кои не се во ASCII формат.

SMTP не може да пренесува исполнети фајлови или други бинарни податоци. MIME не ги заменува ни SMTP, ни POP3 другите протоколи за електронска пошта, туку овозможува произволни податоци да се кодираат во ASCII облик и да се пренесат како стандардна email порака. За да се опфатат произволните типови на податоци и начините за нивно претставување,секоја MIME порака содржи информации кои го известуваат примачот за типот на податокот и типот на кодирањето на содржината. Овие информации се наоѓаат во заглавјето на пораката на стандардниот формат 822, во оние редови на заглавјето кои се однесуваат на MIME се наведува верзијата на MIME која се користи, типот на податоците кои се испраќаат и кодирањето кое се користи за конверзија на податокот во ASCII. формат. Ограничувањата дефинирани во RFC 821 и RFC 822 кои се однесуваат на системите за електронска пошта покажуваат дека навистина ја лимитираат содржината на пораката до многу кратки линии од ASCII карактери. Ова ги наведува корисниците да го конвертираат секој нетекстуален податок (што сакаат да го испратат) во 7 битни бајтови од ASCII карактери. Откако корисникот ќе ја изврши оваа конверзија, го повикува корисничкиот агент за да ја испрати пораката.

SMTP не може да пренесува текстуални податоци кои ги вклучуваат карактерите од националниот јазик, доколку не се претставени како кодови со вредност 128 или поголема, базирани на ASCII формат.

SMTP серверите можат да ги одбијат mail пораките ако ја преминат вистинската големина. Некои сервери си имаат перманенти и транзиентни ограничувања на максимална големина на mail податокот и ќе може да ја прифати од клиентот во било кое време.

MIME е стандард кој вклучува механизми за да ги реши проблемите кои се јавуваат. Овој стандард е високо компатибилен со постоечките RFC 822 стандарди. Бидејќи mail пораките се препраќаат (forward/reset) преку mail портите, SMTP клиентот не може да направи разлика помеѓу серверот кој управува со одредишното сандаче (destination mailbox) и оној кој се однесува како gateway за другата мрежа.

MIME не бара mail деловите да бидат кодирани. Одлуката за тоа е оставена на корисникот или на mail програмот. За бинарен податок кој се пренесува преку SMTP,

потребно е кодирање, но за податок кој се состои од текст може да се изостави кодирањето.

RFC 2045 ги категоризира елементите од телото на mail-от во седум content-type полиња, кои имаат подтипови. Content-type полето или подтиповите имаат параметри со кои го обликуваат објектот што не интересира. Во ова RFC е дефиниран механизам за регистрирање на нови вредности за овие или други MIME полиња со помош на Internet Assigned Numbers.

Бидејќи RFC 822 ја дефинира синтаксата од заглавјето на пораките, но не и композицијата на телото на пораките, MIME стандардот во голема мера е компатибилен со RFC 822, посебно со RFC 2045 кој ја дефинира структурата на телото, како и множество од заглавја со чија помош ја опишува структурата на пораката.

Принцип на работа на MIME

MIME дефинира нови пет заглавја (headers) кои можат да се додадат на оригиналното e-mail заглавие:

- **MIME-Version:** Со ова се дефинира верзијата на MIME протоколот со чија помош е направена пораката. Верзии на MIME се 1.0 и 1.1 (тековно се користи верзијата 1.1, [1]).
- **Content-Type:** Ова поле ја содржи информацијата за типот на податокот. Ако е тоа GIF слика тогаш полето ќе ја има следната вредност: Content-Type: image/gif. Полето Content-Type мора да содржи два идентификатори: тип на содржина и подтип, а меѓу нив треба да стои коса црта. Во нашиот пример, типот на содржината е image, а подтипот е gif. Default вредноста е text/plain, charset = us-ascii, која индицира 7-бит ASCII текст податок (кој е тело на пораката според RFC 822 дефиницијата).
- **Content-Transfer-Encoding:** Во овој ред од заглавјето стои информацијата за начинот на конвертирање на податокот во ASCII формат, т.е кое кодирање се користи при конверзијата. На пример: Content-Transfer-Encoding: base64. За да може сликата да се види, e-mail апликацијата на примачот мора најпрво да ја конвертира сликата од base64 назад во бинарен податок и потоа да избере апликација која ќе ја прикаже сликата на екранот на корисникот.

- **Content-Description:** Овој дел од заглавјето дава опис за содржината на пораката.
- **Content ID:** Претставува глобално уникатна вредност која ја карактеризира содржината (составот) на овој дел од пораката.

Полето "Content type"

Телото на пораката е опишано со ова поле Content-Type во следнава форма:
Content-Type: type/subtype; parameter=value; parameter=value

Параметрите кои треба да бидат доделени, зависат од типот и подтипот. Некои тип/подтип парови немаат параметри, некои имаат како опција, некои имаат повремено, а некои ги имаат и двете. Постојат седум стандардни полиња од овој тип кои се елаборирани во продолжение, тоа се: 1) текст (text), 2) повеќе делови (multipart), 3) слика (image), 4) видео (video), 5) аудио (audio), 6) апликација (application), и 7) порака (message).

1) Текст

Кај текстот подтипот е дефиниран и може да биде:

- plain: Неформатиран текст. Овој текст може да се карактеризира со charset параметар. Следните вредности се дозволени:
  - us-ascii: Текстот содржи ASCII карактери во опсег од 0 до 127.
  - iso-8859-x: Овде x претставува опсег од 1 до 9 за различни делови од ISO-8859 стандардот. Текстот се состои од ISO карактери во опсег од 0-255.
- HTML: текст форматиран според html правилата (види глава 5 за повеќе детали).

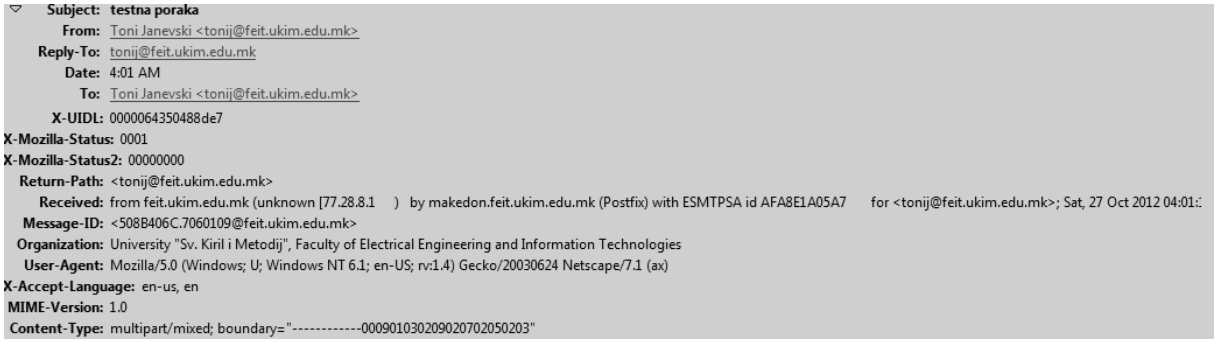
Сите карактерни сетови од ISO-8859 се базирани на ASCII во опсг од 128 до 255. Доколку текстот не содржи не содржи карактери со вредност над 127, тогаш сетот од карактери се специфицира како us-ascii,бидејќи текстот може да биде претставен само во тој сет.

2) Multipart

MIME порака може да содржи повеќе (multiple) независни делови (parts). Multipart заглавието обезбедува разграничување меѓу различните делови на пораката.

MIME пораката се состои од различни делови кои се независни меѓу себе и се енкапсулирани во пораката (attachments). Тоа се дефинира со параметар во content-type полето, како на пример:

Content-Type: multipart/mixed; boundary="199502130905517"



Слика 4.7 Пример за multipart порака

Полињата на заглавјето ја дефинираат содржината на енкапсулираната порака. Постојат 4 подтипови на текст кои се дефинирани:

- **Mixed** (мешан): Овој подтип дозволува една порака да содржи повеке делови кои се независни помеѓу себе, но се пренесуваат заедно. Различните делови можат да имаат своја содржина и свој тип на кодирање. Тие треба да бидат претставени на примачот бидејќи постојат во mail пораката. Мешаните пораки овозможуваат во истата порака да се вметнат текст, графика и звук или да се пренесуваат забелешки за различни сегменти.
- **Parallel** (паралелен): Овој подтип се разликува од претходниот по тоа што не постои ред во деловите. Паралелниот подтип дозволува една порака да се прати на делови кои треба да се прикажат заедно (пример: видео и звучниот запис кои треба истовремено да се пуштат).
- **Alternative** (алтернативен): Овој подтип дозволува една порака да содржи повеќе претставувања на истата содржина. Алтернативните пораки од повеќе делови се корисни кога истата порака се испраќа на повеќе корисници кои користат различни хардверски и софтверски системи. На пример, некој документ може да се испрати како обичен ASCII текст и во форматиран облик, така да корисниците кои имаат системи со графичка можност можат да го изберат да го читаат форматираниот облик.

- **Digest**: Овој подтип овозможува една порака да содржи збир од други пораки (пример: колекција од други пораки од електронската пошта). Се користи во заеднички случај кога multipart RFC 822 или MIME порака се пренесени заедно.

Пример за составена multipart/mixed порака е прикажана на слика 4.7.

3) Image - слика

Телото содржи податок слика која бара графички дисплеј или некој друг уред кој може да го прикаже ваквиот податок. Дефинирани се два подтипови:

- JPEG (Joint Picture Experts Group)
- GIF (Generic Image Format)

4) Видео

Телото содржи податок со подвижна слика (можно е синхронизија со аудио звук). Потребно е терминал или мултимедијална работна станица за да се види. Пример за подтип:

- mpeg: MPEG формат

5) Аудио

Телото содржи податок за звук за кој е потребен микрофон и звучна картица (или сличен хардвер) за да биде прикажан. Пример за подтип:

- Basic: канал на кодиран говор со земање примероци со фреквенција 8 kHz

6) Application - апликација

Овој тип е наменет за типови кои не се вклопуваат во другите категории, како и обработка на податокот со апликациски програм пред да биде прикажан (претставен) на корисникот, како на пример: MS-Word, MS-Excel, Adobe Acrobat итн. Наменет е и за апликациски програми кои треба да бидат обработени како дел од процесот за читање на mail-от.

7) Message (порака)

Овој тип се однесува на целосна mail порака (на пример: порака која е проследена кон следен корисник – forwarded).

MIME типовите опишани погоре се листа од сет на различни типови од податоци кои можат да бидат вклучени во mail пораките. За да можат да бидат пренесени преку електронската пошта, неопходно е нивно кодирање во форма во која ќе можат да се пренесат, како и нивно декодирање на приемната страна.

Полето "Content-Transfer-Encoding"

Content-Transfer-Encoding полето го дефинира методот за кодирање на пораките во единици и нули (т.е. во бити). Ова поле може да прими 5 различни вредности. Три од овие вредности покажуваат дека не е извршено кодирање. Петте типови на кодирања се следниве:

- **7 – битно**: Телото се состои од линии ASCII текст со должина помала од 1000 симболи.
- **8 – битно**: Кај 8-битното кодирање не мора да бидат ASCII симболи, но линијата е и тука ограничена на 1000 симболи. Притоа, во SMTP имплементациите треба најважниот бит да се намести на нула. Во друг случај, 8-битното кодирање би било невалидно.
- **Binary (бинарно)**: Индицира презентација на не-ASCII симболи и линиите можат да бидат и подолги од 1000 симболи, премногу долги за SMTP транспорт. Не постојат стандарди за пренос на некодирани бинарни податоци преку електронска пошта базирана на TCP/IP протоколниот стек. Во овој случај MIME не се користи, така што останува на SMTP да ги испраќа податоците. Заради тоа и не се користи во практика овој тип, туку следните два во продолжение, [1].
- **Quoted–Printable**: Целта на ова кодирање е да ги остави text фајловите и да бидат читливи во нивната кодирана форма, што значи дека не ги кодира ASCII симболите (т.е. текстот). Quoted-Printable го заменува секој нетекстуален карактер (кој не е ASCII симбол) со 3 бајтна секвенца, каде што првиот бајт е знакот еднакво (=), а вторите два бајти се хексадецимална презентација на бајтот што се кодира. Ова кодирање се применува кога поголемиот дел од пораката е од ASCII симболи со помал дел кој не е составен од ASCII симболи (т.е. не е текст).
- **Base64**: Оваа кодирање е создадено за податоци кои главно не содржат текстуални знаци. Base64 кодирањето работи на тој начин што влезниот

поток (stream) го третира како битски поток (bit stream), прегрупирајќи ги битите во пократки бајтови, надополнувајќи ги кратките бајтови до 8 бита и потоа да се пренесат овие бајтови како ASCII карактери (независно од тоа дали потекнуваат од видео фајл, аудио фајл, апликација, слика, итн.). Максималната должина која се користи е 6 бити, која може да претстави 64 уникатни карактери (оттука и името Base64). На сликата 4.8 е прикажано мапирање меѓу Base64 и ASCII карактерите кое се користи при ова кодирање.

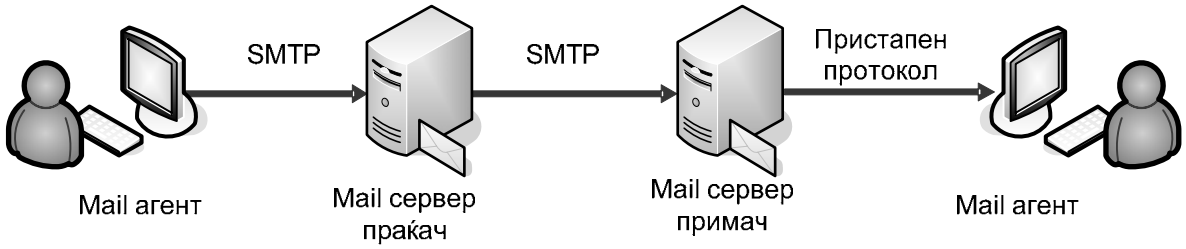
Base64 вредности	ASCII карактери	Base64 вредности	ASCII карактери	Base64 вредности	ASCII карактери
0	A	26	a	52	0
1	B	27	b	53	1
2	C	28	c	54	2
3	D	29	d	55	3
4	E	30	e	56	4
5	F	31	f	57	5
6	G	32	g	...	...
7	H	33	h	61	9
...	...	...	...	62	+
25	Z	51	z	63	/

Слика 4.8 MIME Base64 алфабет

4.4.3 Post Office Protocol (POP)

Протоколите кои се користат за пристап од email клиентот (на пример Outlook, Netscape mail, итн.) до mail серверот се нарекуваат пристапни протоколи (слика 4.9). Генерално, тие можат да се поделат на три групи email пристапни протоколи:

- POP3: Post Office Protocol version 3 (RFC 1939, [20])
  - авторизација (агент <--> сервер) и download
- IMAP: Internet Mail Access Protocol (RFC 1730, [21])
  - Повеќе својства (покомплексен)
  - Манипулација со складираните пораки на серверот
- HTTP (webmail): Hotmail, Yahoo! Mail, Google mail (Gmail), итн.



Слика 4.9 Протоколи за пристап до mail

Најпопуларниот протокол кој се користи за трансфер на e-mail пораки од траен mail box до локален компјутер е познат како верзија 3 на Post Office Protocol (POP3), иако во 21 век подеднакво се користи и HTTP пристапот до електронска пошта.

Кога се користи POP3, корисникот го применува POP3 клиентот, кој создава TCP конекција до POP3 серверот на mailbox компјутерот. Корисникот најпрво испраќа корисничко име (username) и лозинка (password) со цел да изврши автентикација на сесијата. Штом ќе ја изврши автентикација корисничкиот клиент, корисникот испраќа команди да прими копии од една или повеќе пораки, како и опционално да изврши нивно бришење од поштенските сандачиња. Пораките се зачувуваат и пренесуваат како текст фајлови во RFC 822 стандардниот формат.

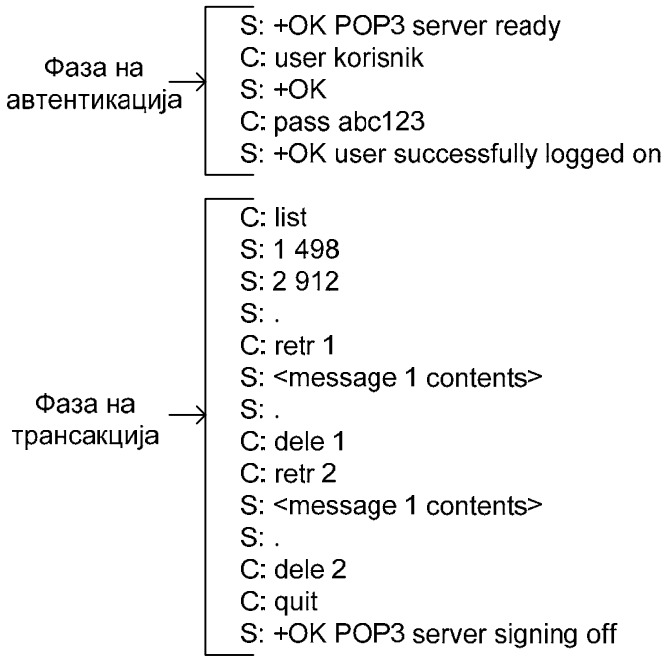
Во случај кога се користи POP3 како пристапен протокол, тогаш e-mail серверот кој го опслужува дадениот корисник треба да има два активни сервери: 1) SMTP сервер за дојдовни пораки од други SMTP сервери или за пораки испратени од корисниците преку нивните e-mail клиентски апликации (на пример, Outlook, Thunderbird, итн.); и 2) POP3 сервер за комуникација со POP3 корисничките клиенти (исто така интегрирани во корисничката e-mail апликација) кога побаруваат излистување и симнување на e-mail пораките (од серверот кон клиентот). SMTP серверот ја прифаќа пораката испратена до корисникот и ја додава секоја нова порака во трајното т.н. поштенско сандаче на корисникот. POP3 серверот дозволува корисникот да екстрахира пораки од поштенското сандаче и да ги брише. Со цел да се осигура точната операција двата сервери мора да го координираат користењето на поштенското сандаче на страна на e-mail серверот. Ова значи дека пораките пристигнуваат преку SMTP, а корисникот ги симнува од серверот користејќи POP3.

POP3 клиентите остваруваат TCP конекција со серверот користејќи ја портата 110. Кога конекцијата е остварена, POP3 серверот праќа порака за потврда на клиентот.

Сесијата тогаш влегува во состојба на автентикација (authentication state). Клиентот мора да му испрати идентификација на серверот додека сесијата се наоѓа во оваа состојба.

Доколку серверот успешно ја потврди идентификацијата, сесијата влегува во т.н. "состојба на трансакција" (transaction state). Во оваа состојба клиентот има пристап до поштенското сандаче.

Кога клиентот испраќа QUIT команда, тогаш сесијата влегува во "update состојба" и конекцијата се прекинува.



Слика 4.10 Фази на POP3 протоколот

POP3 команди и одговори

POP3 командите содржат клучни зборови (keywords). Клучните зборови се долги три или четири карактери и се одделени со по еден празен (space) карактер од аргументите. Пример на клучни зборови во фаза на автентикација се "user" и "pass" (слика 4.10). За секоја команда од клиентот во оваа фаза, изразена преку клучен збор проследен со аргументи, серверот праќа одговор. Одговорот мора да започнува со статус индикатор кој покажува дали одговорот е позитивен или негативен. Овие индикатори се или позитивен одговор (+OK) или негативен одговор (-ERR).

Во фазата на трансакција (слика 4.10), клучни зборови се: "list" за листање на пораките, "retr" за преземање на порака од серверот, "dele" за бришење на порака на

серверот (обично по преземањето на пораката, ако така е поставен e-mail клиентот), "quit" за терминирање на POP3 конекцијата меѓу POP3 клиентот и POP3 серверот.

4.4.4 Internet Message Access Protocol version 4 (IMAP 4)

IMAP4 е стандарден протокол за пораки со функции и на клиент и на сервер. Слично на POP3, IMAP4 серверите ги зачувуваат пораките за повеќе корисници кои треба да бидат вратени по барање на клиентот, IMAP4 клиентите имаат повеќе способности во тоа отколку POP клиентите.

IMAP4 им дозволува на клиентите да имаат повеќе сандачиња, да креираат нови поштенски сандачиња на email серверот, или да избришат постоечки поштенски сандачиња, што не постои како можност со POP3. Со IMAP4 може да се креираат директориуми на email серверот што не е можно со POP3 (со него е можно да се креираат директориуми и да се класифицираат пораките само локално, на корисничката страна).

IMAP4 клиентите можат да посочат критериум за симнување на пораки, да симнат само делови од дадена порака што е погодно при трансфер на големи пораки преку побавни линкови (со помалку бити/сек). IMAP4 секогаш ги чува пораките на серверот, како и нивни копии. Тековната верзија на IMAP4 (rev1) е дефинирана со RFC 3501.

IMAP4 основни модели на електронска пошта

IMAP4 ги подржува сите три главни модели за електронска пошта (RFC 1733). Овие модели се offline, online и disconnected кориснички модели. Во моделот offline, клиентот периодично се конектира со серверот и ги симнува mail пораките. Mail пораките се бришат на серверот. Пример за mail протокол кој го користи ваквиот модел е POP3. Во моделот online, клиентите прават промена на серверот.

Моделот disconnect претставува комбинација од online и offline моделот. Во овој модел клиентот симнува податоци, прави измени во самите податоци, за да подоцна изврши upload на серверот. Пример за mail програм кој може да го користи овој модел е Lotus Notes mail.

Овие три модели имаат некои предности и недостатоци. Се додека IMAP4 ги подржува сите овие модели, клиентот е способен да се префрла во друг модел. На пример, ако конекцијата е премногу бавна и постои голема порака во сандачето IMAP4 клиентот може да врати мал дел од пораката, поконкретно може да се пристапи со IMAP до секој MIME дел на email пораката.

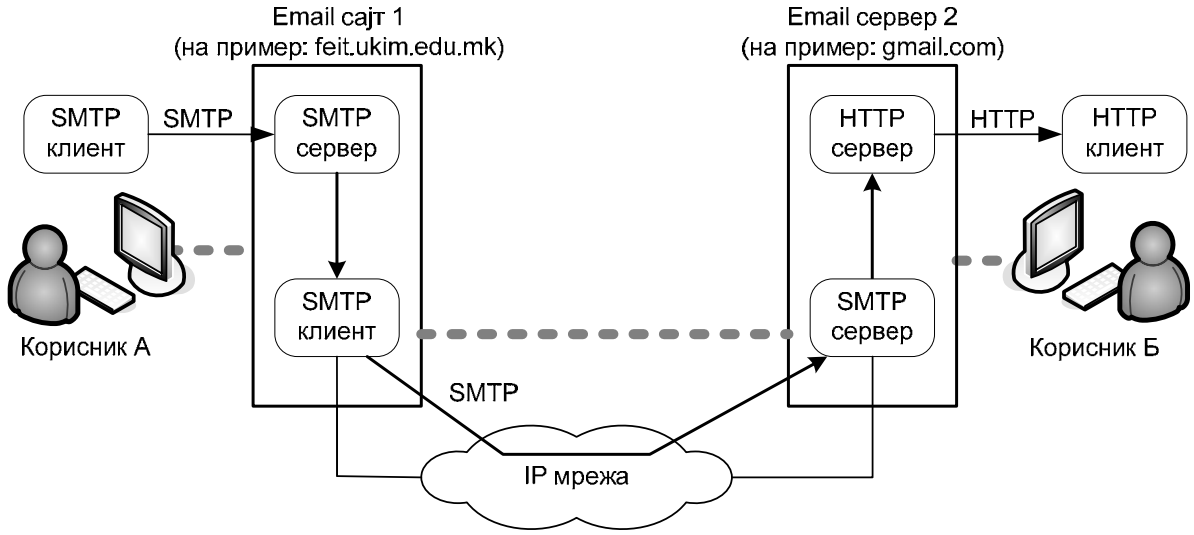
Генерално, IMAP4 има предности во однос на POP3 во однос на повеќе функционалности кои ги нуди во однос на управување со поштенското сандаче (mailbox) на даден корисник сместено на email сервер (поштенското сандаче е апстракција за даден директориум или фајл во кој се сместуваат пораките за даден корисник во email серверот).

4.4.5 Веб-базирана електронска пошта (webmail)

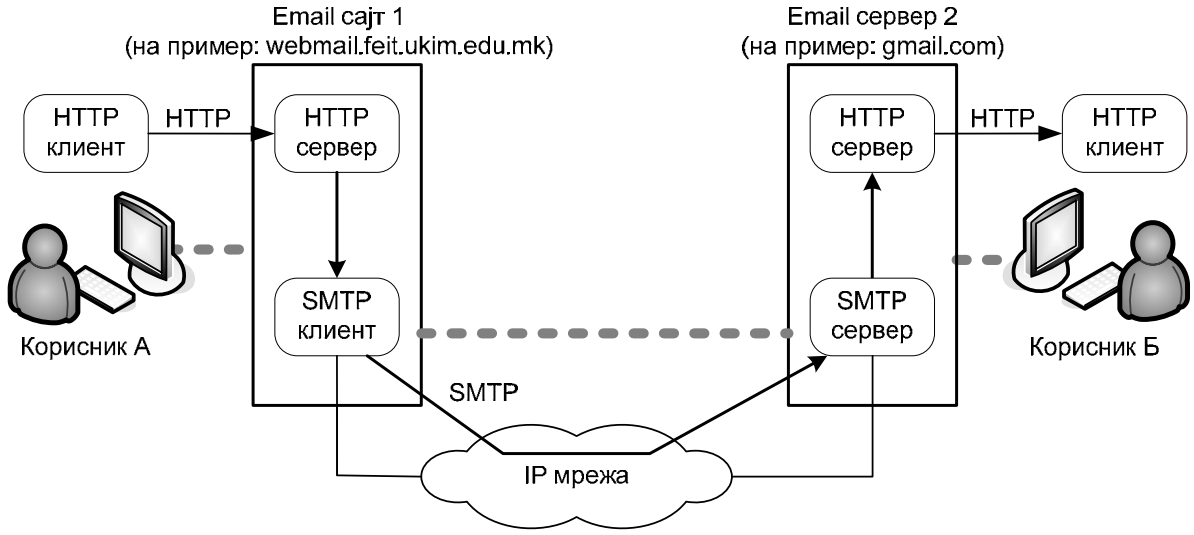
Со раширувањето на веб сајтовите кон крајот на 1990-те години, од кои поголем дел го засновуваат бизнис моделот на маркетинг преку мултимедиски објекти на нивните веб страници, се создаде подлога за понуда на бесплатни e-mail адреси (т.е. поштенски сандачиња) со пристап преку даден веб сајт. Така, од почетокот на 21-от век веб-базираната електронска пошта брзо се рашири што овозможи секој поединец кој го користи Интернет да може да има своја e-mail адреса (најчесто и повеќе e-mail адреси по корисник, од ист или од различни домени). Типични глобални примери на веб-базирана пошта денес, кои се и најчесто користени, се Yahoo mail, Google mail (gmail), Hotmail (на Microsoft), како и многу други. Покрај нив, денес секоја поголема и помала компанија или институција исто така овозможува и веб-базирана електронска пошта, покрај традиционалниот начин преку кориснички агент кој користи SMTP за праќање на e-mail, а POP3 за повлекување на пораките од e-mail серверот.

Користење на e-mail преку веб пребарувач всушност е користење на HTTP протоколот (види глава 5 за HTTP) за интеракција меѓу клиентот (веб пребарувачот, како што е Internet Explorer, Firefox, Chrome, Opera, итн.) и веб серверот кој е интегриран со e-mail серверот (SMTP серверот) во ваков случај. Притоа, можеме да разграничиме две сценарија при користење на веб-базиран e-mail, [1], што е илустрирано на слика 4.11:

- сценарио 1: корисникот А праќа e-mail порака до корисникот Б, при што корисникот А користи email агент со SMTP клиент (на пример: Outlook, Thunderbird, итн.), а корисникот Б пристапува до електронската пошта преку користење на HTTP, односно преку неговиот веб пребарувач;
- сценарио 2: корисникот А праќа e-mail порака до корисникот Б, при што и двата корисника користат веб-базирана електронска пошта, и во двата случаи: праќање на e-mail пораки или читање на пристигнати e-mail пораки во поштенското сандаче.



а) Сценарио 1



б) Сценарио 2

Слика 4.11 Веб-базирана електронска пошта (webmail)

4.5 Дискусија

Во оваа глава фокусот беше насочен кон основните Интернет технологии и сервиси DNS, FTP, Telnet и e-mail. Без DNS не би можел да се замисли денешниот Интернет, кој е заснован на користење на имиња на сајтови кои се преведуваат во IP адреси за да може да се пристапи до нив преку користење на механизмите на рутирање на IP пакетите од крај до крај низ Интернет. Според тоа, DNS е една од најфундаменталните Интернет технологии, која е суштинска за функционирање на сервисите како што се FTP (преведување на името на FTP сајтот, на пример ft.ukim.edu.mk, во соодветна IP адреса на која ослушнува FTP серверот), електронската пошта (преведување на домен имињата на e-mail серверите во IP адреси кои ги имаат на нивните интерфејси), како и веб сервисите (кои ќе ги опфатиме во следната глава од оваа книга), но исто така и низа други апликации/сервиси кои се користат во Интернет денес или ќе се користат во иднина.

Понатаму оваа дискусија ќе ја продолжиме со електронската пошта, прескокнувајќи ги FTP (кој се користи и денес, но речиси е целосно заменет од HTTP, т.е. вебот) и Telnet (кој е важен од историска гледна точка, но денес речиси да не се користи повеќе). Електронската пошта (e-mail) е еден од најстарите Интернет сервиси, стар повеќе од три децении, кој продолжува да егзистира и да го зголемува своето учество во однос на број на e-mail сервери и e-mail корисници. Имено, e-mail сервисот стана незаменлив во изминативе деценија-две и во секојдневната комуникација по електронски пат меѓу луѓето и во деловното (бизнис) работење. На пример, традиционалниот факс во бизнис секторот заминува во историја токму заради e-mail сервисите кои овозможуваат закачува на мултимедиски документи притоа кодирајќи ги и пренесувајќи ги како ASCII симболи (како текст). Според ова, електронската пошта е еден од најважните Интернет сервиси кој постои денес и кој сигурно ќе постои да егзистира и во блиската иднина во оваа форма.

Глава 5

Веб технологија (World Wide Web)

Во оваа глава ќе се запознаеме со WWW (World Wide Web), апликација која имала најголемо влијание во развојот на Интернет досега. World Wide Web е глобален систем чиј развој започнал во 1989 година од Tim Berners Lee во Европската лабораторија за практична физика CERN во Швајцарија. Тој креирал протокол за дистрибуција на документи, каде што во еден текстуален документ можело да се постават линкови до други документи или објекти (видео фајлови, аудио фајлови и сл.) кои биле сместени на истиот или на некој друг сервер приклучен на Интернет мрежата. Протоколот кој овозможувал пристап до такви страни на принципот клиент-сервер (каде што страниците се сместени на страната на серверот) бил наречен HTTP (HyperText Transfer Protocol), односно протокол за пренос на страници со хипертекст.

Зошто бил потребен HTTP?

Според зборовите на Tim Berners Lee од 1991 година, HTTP мора да обезбеди:

- Функционалности за пренос на фајлови
- Можност да се пребарува по дадени клучни зборови (по индекс)
- Автоматски да се преговара за форматот за приказ на веб документот меѓу серверот и клиентската апликација
- Можност да се редиректира клиентот кон друг сервер на Интернет

Актуелниот стандард за веб (www) т.е. HTTP е RFC 2616. Постојат две верзии (досега) на HTTP: HTTP 1.0 (RFC 1945), и HTTP 1.1 (RFC 2068).

Вебот е составен од голем број документи наречени **веб страници**. Секоја веб страница е т.н. **hypermedia** документ (**hyper** затоа што содржи линкови до други веб страници, а **media** бидејќи може да содржи и други објекти покрај текст). Вебот е клиент-сервер апликација заснована на TCP/IP, серверот слуша на порта 80.

- Веб **клиентот** е **пребарувач** (**browser**), пример: Internet Explorer, Netscape, Opera, Firefox
- Веб **серверот** содржи веб страници, пример на веб сервер: IIS (Internet Information Services), Apache.

Јазикот кој бил креиран за пишување на такви страници со линкови до други страници или објекти е наречен HTML (HyperText Markup Language).

Web browser (веб пребарувач) е апликација која овозможува пристап до Web сервер на кој се сместени веб страници, која што го "разбира" HyperText Markup Language (HTML) и содржи HTTP клиент кој се користи за обновување на веб страни.

Веб пребарувачите се одговорни за форматирање и прикажување на информација, како и за интеракција со корисникот.

5.1 HTTP (Hypertext Transfer Protocol)

Постојат две верзии на HTTP протоколот: HTTP 1.1 кој е опишан во RFC 2616; и постариот HTTP 1.0 - протокол кој е опишан во RFC 1945, [5]. HTTP протокол е дизајниран за да овозможи трансфер на HTML (Hypertext Markup Language) документите. HTML е tag јазик кој се користи за креирање на хипертекстуални документи. Hypertext документи вклучуваат линкови до други документи кои содржат дополнителни информации за бараниот термин или објект. Овие документи освен текст, можат да содржат графички слики, аудио и видео клипови, JAVA applets и сл.

Една HTTP клиент-сервер комуникација е поделена во 4 чекори:

- Клиентот иницира TCP конекција (креира сокет) кон серверот, порта 80
- Серверот ја прифаќа TCP конекцијата од клиентот
- HTTP пораки (на апликациско ниво) се разменуваат меѓу веб пребарувачот (HTTP клиентот) и веб серверот (HTTP сервер)
- TCP конекцијата се затвара

HTTP комуникацијата се остварува преку TCP/IP конекции преку портот TCP/IP број 80, но покрај овој порт можат да се користат и други порти.

На пример, за да се симне една веб страна која вклучува две графички слики, тогаш пребарувачот ќе отвори една конекција за страната и две конекции за секоја од графичките слики, доколку се работи за HTTP 1.0 (првичната верзија на HTTP).

HTTP 1.1 е дефиниран во RFC 2616, [6], го олеснува овој проблем со тоа што една TCP конекција се воспоставува за веб страната и за сите објекти кои се линкувани на таа страна, на пример за страната и двете слики од претходниот пример. Ваквата конекција е постојана (persistant). Верзијата на HTTP пораките се праќа со HTTP полето во првата линија на пораката.

Секоја веб страница има уникатен идентификатор - URI (Uniform Resource Identifier). URI се однесуваат генерално на web адресите и на комбинацијата од URL (Uniform Resourse Locator) и URN (Uniform Resourse Name).

Пример за URL: **http://www.lokacija.mk/tk/**

- Првиот дел го специфицира протоколот: **http**
- Хостот-сервер е: **www.lokacija.mk**
- Патот до веб страницата на тој сервер е: **/tk/strana.html**

Генерален пример за http URL:

http: // *imenahost* [:porta] / *pateka* [; *parametri*] [? *query*]

Како што може да се забележи во генералниот пример за URL постои можност (опционално) да се специфицира друга порта за веб комуникација (различна од 80) која следува одвоена со две точки (“:”) по името на хостот (т.е. веб серверот). По локацијата на страната на веб страната може да се специфицираат параметри со конкретни вредности во т.н. query кое е одделено со прашалник (“?”) од локацијата на веб страницата на дадениот веб сервер. На пример:

http://www.webserver.mk/login.html?username=toni&password=123&language=mk

5.2 HTML

Програмскиот јазик Hypertext Mark-Up Language (HTML) се користи за креирање на хипертекст (текст кој не мора да биде секвенцијален) фајлови. Сфатен како семантички јазик за означување на логичката структура (mark-up), HTML им дава на корисниците можност за идентификација на составните делови од документот. Корисниците на HTML креираат датотеки од означен текст, аналогно како кај програмски јазик: авторите ги запишуваат информациите користејќи одредена структура со цел "компјутерот" (во овој случај веб пребарувачот) да го разбере.

Иако HTML не е комплициран како некои компјутерски програмски јазици, пишувањето на HTML бара од авторите да се придржуваат кон одредени правила за означување на деловите од документот. Притоа, авторите внимателно ја дефинираат структурата на документот така што секој постоечки веб пребарувач може да го прочита и да го прикаже на начин најдобар за тој веб пребарувач. Ова овозможува прикажување на информации во HTML без креирање на посебна верзија за секој тип веб пребарувачи одделно.

Секој HTML документ се состои од текст и знаци (tags) кои се користат за претставување на податоците од документот и означување на структурата. Едноставен HTML документ е прикажан на сликата 5.1.

```
<HTML>
<HEAD>
<TITLE>Пример на HTML документ</TITLE>
</HEAD>
<BODY>
    Ова е пример на HTML документ
</BODY>
</HTML>
```

Слика 5.1 HTML документ (веб страница)

Симболите <знак> (т.е. <tag>) се користат за означување и разделување на елементите на документот. Елементи кои можат да бидат во HTML документот се:

заглавие (head) или тело (body) и во зависност од тоа како функционираат коментари и елементи за графика.

HTML елементот ги заокружува HTML елементите во датотека. Неговата почетна ознака е <HTML>, а крајната е </HTML>.

Коментар во HTML документ може да се постави со почетна ознака <!-- и крајна -->. На пример:

<!-- Ова е коментар -->

Овие елементи се користат за идентификација на својствата на целиот документ, како што се насловот (title), врските (links) за укажување на други сродни документи и основниот URL na документот.

5.2.1 Дизајнирање на HTML документ

За да се креира еден HTML документ, авторите ги поставуваат елементите од заглавјето (информации за документот) и елементите од телото (содржина на документот) во датотека т.е. фајл. Добра идеја е овие делови да се одделат со знаци за почеток и крај на заглавјето и телото.

Бидејќи има знаци кои авторите ги користат во сите HTML документи, добро е да се направи маска (template) која ке ги содржи основните елементи дадени на сликата. Еден таков пример е даден на слика 5.2.

```
<HTML>
    <HEAD>
        <TITLE>Наслов на документ</TITLE>
    </HEAD>
    <BODY>
        <ADDRESS>Име презиме, улица, број (e-mail)</ADDRESS>
    </BODY>
</HTML>
```

Слика 5.2 Пример за template.html

Наслов на документот (title)

Најчесто се користи како спецификатор на HTML документот во многу содржини на веб. Насловот се поставува меѓу знаците <title> и </title> во заглавјето на документот. Пример:

<TITLE> Факултет за електротехника и информациски технологии </TITLE>

Заглавја (headers)

Шесте нивоа на заглавја дават можности за креирање на хиерархија на информациите во документот. Како такви се елементите од семантичката хиерархија. Авторите треба да ги користат овие заглавја последователно, почнувајќи од <H1> до <H6> со чекор 1. Нивната цел е подетално опишување на содржината на документот и која е неговата намена.

<H1> Интернет технологии </H1>

Врски

Врските се основата на хипертекстот и се креираат со помош на знакот котва “A”. Имено, една веб страница напишана во HTML вклучува врски (хиперлинкови) до страни, најчесто сродни по содржина на дадената страница од веб. Основниот облик за поставување на врски е следниов:

<A 'Href=http://www.feit.ukim.edu.mk’> ФЕИТ </A>

каде href е URL за документот посочен со врската, а ФЕИТ (во примеров) е текстот што ќе се појави при прикажувањето на веб страницата, кој најчесто е осветлен или подвлечен (или и двете) при прикажување на страницата во веб пребарувачот.

Постојат и многу други детали кои се однесуваат на дизајн на веб страниците, но тоа не е цел во оваа книга. Главната цел е начинот на функционирање на HTTP протоколот, односно HTTP комуникацијата.

5.3 HTTP комуникација

Во повеќето случаи, HTTP комуникацијата е иницирана од корисник-агент кој што бара ресурси од веб сервер. Во наједноставен случај, конекцијата е овозможена преку единечна конекција меѓу корисникот и серверот. Притоа се користи TCP, а постапката тече на следниов начин**:**

- Клиентот иницира TCP конекција (креира сокет) кон серверот, порта 80
- Серверот ја прифаќа TCP конекцијата од клиентот
- HTTP пораки (на апликациско ниво) се разменуваат меѓу веб пребарувачот (HTTP клиентот) и веб серверот (HTTP сервер)
- TCP конекцијата се затвара

Притоа, HTTP е “stateless”, што значи дека серверот не води евиденција за поминатите барања (request-и) од клиентот. Таква евиденција вообичаено секој од клиентите (веб пребарувачи) ја сместува привремено во меморијата на локалниот хост на кој работи тој веб пребарувач (на пример: историја на сурфањето по веб, или содржините на страните кои се симнале се сместуваат и чуваат одредено време во кеш меморијата која е исто така сместена на локалниот хард диск на клиентскиот хост).

Во некои случаи не постои директна врска меѓу корисникот и серверот. Постои една или повеќе интермедијаторни врски меѓу клиентот и серверот како што се: прокси сервер, gateway или тунел.

Прокси претставува портал на апликациско ниво и тој може да се справи со содржината на информацијата која ја испраќа клиентот и истата може да ја модифицира. Кога некое барање ќе дојде до прокси, тој ја копира целосно или делумно пораката и ја препраќа до следната дестинација. Gateway-от, пак, ја добива пораката и ја испраќа кон протоколите во соодветен формат. Тунелот не ја менува содржината на пораката, туку ја испраќа истата како што е (промените се прават на мрежното ниво со енкапсулирање/декапсулирање на IP пакетите, содржината на апликациско ниво е непроменета). Прокси серверите и порталите можат да се справат со прифаќање на HTTP пораките. Ова може драстично да го намали времето на реакција и да го редуцира IP сообраќајот во мрежата (понатаму во оваа глава, глава 5, ќе се навратиме на кеширањето при HTTP комуникацијата). Бидејќи тунелите не ја разбираат

содржината на пораките, тие не можат да ја меморираат информацијата од HTTP пораките.

5.3.1 Основни карактеристики на HTTP

Протоколот кој се користи за комуникација помеѓу пребарувачот и Web серверот или помеѓу клиентските машини и Web серверите е познат како HyperText Transfer Protocol (HTTP). HTTP ги има следниве карактеристики:

- **Апликациско ниво**. HTTP функционира на апликациско ниво. Претставува доверлив конекционо ориентиран протокол, кој не овозможува доверливост во поглед на ретрансмитирање.
- **Request/Response**. Штом ќе се воспостави транспортна сесија, едната страна (најчесто пребарувачот) мора да испрати HTTP барање на кое другата страна треба да одговори.
- **Stateless**. Секое HTTP барање е самостојно серверот не чува историја на претходни барања или за претходни сесии.
- **Бидирекционен пренос**. Најчесто пребарувачот побарува Web страни а серверот ги проследува копиите до пребарувачот. HTTP дозволува трансфер од пребарувач до сервер.
- **Способност за преговарање**. HTTP им дозволува на пребарувачите и серверите да ги договорат деталите. Испраќачот може да ги дефинира можностите кои ги нуди а примачот да ги дефинира можностите кои ги прифаќа.
- **Поддршка за кеширање**. Со цел да се подобри времето на одговор, пребарувачот ја кешира секоја копија од веб страна. Доколку корисникот ја побара повторно истата, HTTP му дозволува на пребарувачот да комуницира со серверот за да се утврди дали содржината на страната се изменила откако била направена последната кеширана копија на истата.
- **Поддршка при посредување**. HTTP дозволува постоење на машина помеѓу пребарувачот и серверот која игра улога на proxy server, а која што ќе кешира веб страни и ќе ги опслужува барањата на пребарувачите.

5.3.2 Типови HTTP конекции

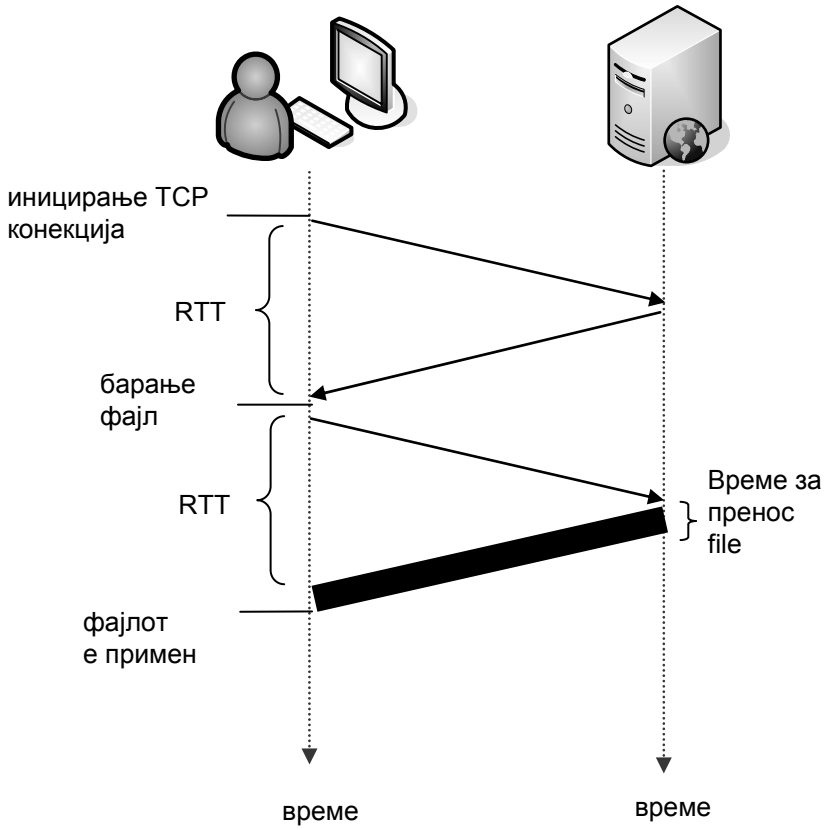
Раните верзии на HTTP ја користеа истата синтакса како FTP, односно за секој трансфер на податоци формираа нова TCP конекција. Фундаментална промена се случи со верзијата 1.1 која се појави во јуни 1999 година. Кај оваа верзија конекцијата со серверот откако еднаш ќе се воспостави е континуирана и се користи да одговори на секое барање на клиентот. Откако ќе се заврши со сесијата, конекцијата се затвора по претходно информирање на обете страни. Главните предности на оваа перзистентна конекција лежат во помалото време на воспоставување, помала меморија искористена за баферите, како и помало процесорско време. Меѓутоа при континуираната конекција потребно е на одреден начин да се изврши разграничување, односно да го дефинира почетокот и крајот на трансмисијата, што и не е така едноставно. Кај HTTP овој проблем се решава со праќање на информација за податочната должина. Меѓутоа не секогаш серверот ја знае тривијално оваа вредност, посебно при користење на Common Gateway Interface (CGI) кој што дозволува динамично креирање на Web страни. Во овој случај серверот го информира клиентот и по завршувањето на трансферот конекцијата се прекинува. Значи, HTTP конекциите можат да бидат:

- **Постојани (persistent)**
  - Повеќе објекти може да се пратат преку една TCP конекција (клиент - сервер).
  - HTTP/1.1 користи постојани конекции во default мод
- **Непостојани (non-persistent)**
  - Најмногу еден објект може да се прати преку TCP конекција.
  - HTTP/1.0 користи nonpersistent HTTP

5.3.3 Моделирање на времето на одговор – RTT (Round Trip time)

Специфичностите на HTTP протоколот влијаат и на брзината со која може да се одвива HTTP комуникацијата или со други зборови, да се сурфа. Принципот на HTTP каде што клиентот праќа request за секој објект (веб страна, слика, видео или аудио фајл, фајл од некоја друга апликација – на пример: Word фајл) и добива соодветен

response од веб серверот, влијае на брзината со која може да пристигнуваат фајловите. Соодветно моделирање на RTT (Round Trip Time) за HTTP е прикажан на слика 5.3.



Слика 5.3 RTT пресметка за HTTP комуникација

Дефиниција на RTT: време да се прати мал пакет од клиентот до серверот и назад.

Пресметка на времето на одговор за HTTP (**response time**):

- Потребно е едно RTT за да се иницира TCP конекција
- Потребно е едно RTT за HTTP request-от и првите бајти од HTTP response да се вратат
- Време за праќање на фајлот

Вкупно време = 2*RTT + (времето за праќање)

Вообичаено, доминантна компонента во горниот израз е RTT кое вообичаено изнесува неколку стотини милисекунди во денешниот Интернет. Времето на праќање зависи од брзината на линкот кој е врска со Интернет. На пример: ако имаме линија со проток од 500 kbit/s, тогаш за пакет со големина 5 kB времето за пренос ќе биде:

(5 KB * 8 bit/Byte) / (500 kbit/s) = 16 usec

5.4 HTTP пораки

HTTP претставува протокол наменет за комуникација помеѓу пребарувачот и Web серверот или одредена преодна машина и Web серверот. Оперира на апликациско ниво, наменет е да користи конекциски-ориентиран транспортен протокол (TCP). Откако транспортната сесија ќе биде воспоставена, едната страна (вообичаено пребарувачот), испраќа HTTP барање на кое другата страна мора да одговори, при што серверот не чува историски податоци за претходни барања. Трансферот на податоците може да се изведува во обете насоки од северот кон пребарувачот и обратно. Исто така има можност за зачувување на претходно посетените страни, што овозможува доколку е потребно директно да се пристапи до нив.

Во наједноставен случај пристапувањето до одредена страна на серверот од страна на пребарувачот е директно. Пребарувачот започнува со соодветниот URL, го користи DNS-от за да изврши соодветно мапирање на името од hostname во соодветна IP адреса, која што ја користи за формирање на TCP конекција со серверот. Откако конекцијата ќе биде воспоставена, пребарувачот и Web серверот го користат HTTP за комуникација.

Пребарувачот може да ја користи командата HTTP GET за да се воспостави конекција со серверот, така што по воспоставувањето на конекцијата не е неопходно внесување на апсолутниот URL. Синтаксата е прикажана во следниот пример. Во вториот случај при воспоставена конекција со серверот ќе се добие истата страна како во првиот случај.

Барањето се состои од линија текст која започнува со клучниот збор GET и по него следува URL-то и бројот на HTTP верзијата. На пример:

GET http://www.feit.ukim.edu.mk/test/ HTTP/1.1

Откако е воспоставена TCP врската, нема потреба потоа да се праќа целата URL, на пример:

GET /test/ HTTP/1.0

Генерално, HTTP има два типа пораки: **request** (**барање**) и **response** (**одговор**). Барањата ги праќа веб клиентот кон веб серверот, а одговорите ги праќа веб серверот кон веб клиентот.

HTTP се користи меѓу веб пребарувач и веб сервер. Пребарувачот праќа барање (request) по што серверот враќа одговор (response).

5.4.1 Состав и должина на HTTP пораките

Составот на HTTP пораките (т.е. нивната градба) многу наликува на email пораките, што е и очекувано, бидејќи email механизмите за размена на мултимедиски пораки на принципот клиент-сервер биле креирани и стандардизирани порано од веб протоколот. Така, HTTP пораката се состои од следните полиња:

- Тип на порака: HTTP пораката може да биде или барање на клиентот (request) или одговор на серверот (response).
- Заглавје на порака: Заглавјето на HTTP пораката може да биде едно од следниве:
  -главно заглавје (general header);
  -заглавје на барање (request header);
  -заглавје на одговор (response header);
  -заглавје на ентитет (entity header).
- Тело на пораката: Телото на пораката може да се однесува на име на ентитет ,доколку не е применето некое кодирање. Телото на пораката го пренесува телото на ентитетот на соодветно барање или соодветен одговор.

5.4.2 Дефиниции на методи

За дефинирање на типот на барањето и одговорот при HTTP комуникацијата постојат дефинирани повеќе методи во HTTP 1.0 и во HTTP 1.1. Еден од нив е и GET кој го анализиравме претходно во оваа глава. Поважните методи дефинирани за HTTP се следните:

- GET: Овој метод му овозможува на клиентот да си ја добие информацијата (на пример: веб страна или некој друг објект) определена со барањето со специфициран URI.
- HEAD: Овој метод му овозможува на клиентот на да си обезбеди метаинформација за ентитетот (веб страната) и не бара да се врши трансфер на целиот ентитет.
- POST: Со оваа функција клиентот (веб пребарувачот) му праќа параметри со одредени вредности на веб серверот (на пример: за веб автентикација со корисничко име и лозинка).
- PUT: Овој метод е сличен со post методот со тоа што во овој случај се специфицира URI на ентитетот (на пример: фајл на веб сервер).
- DELETE: Овие методи бараат серверот да ги избрише изворите од барањата на URI. Барањето не значи дека ќе биде извршено од страна на веб серверот, бидејќи самото извршување на вакво барање зависи од привилегиите (за бришење) што ги има дадениот клиент на веб серверот.
- TRACE: Овој метод му овозможува на клиентот да види како пораката е примена од другата страна (серверската) за тестирања и слични цели.

5.4.3 Пораки на одговор и грешка

Пораките за response од веб серверот враќаат код (трицифрен број) со придружен коментар за објаснување. Самите кодови се класифицирани во групи кои се детерминирани со првата цифра од трицифрениот број.

Дефиниции за статусот на кодовите

- Информативен (1xx): Информативните статус кодови индицираат привремен одговор. Пример за кодови:

  100 продолжи

  101 промена на протоколи

- Успешно (2xx): Оваа класа на кодови покажува дека одредено барање е успешно добиено, разбрано и прифатено. Пример за кодови:

  200 ОК

  202 прифатено

- Пренасочување (Redirection, 3xx): Оваа класа на кодови индицира дека се бара активност од корисникот со цел да се оствари барањето. Пример за кодови:

  302 привремено преместено

  305 користи прокси

- Грешка кај клиентот (4xx). Пример за кодови:

  400 лошо барање

  404 не е најдено

- Грешка кај серверот (5xx). Пример за кодови:

  500 внатрешна грешка

  503 недостапен сервис

Како да одговори веб серверот во случај на неисправно барање. Најчесто ваквите барања се испраќаат од страна на пребарувачот кој потоа ќе се обиде да прикаже се што серверот ќе му врати како одговор. Серверите обично генерираат пораки на грешка во HTML формат.

Пребарувачот ја користи "head" на документот интерно а на корисникот му се прикажува само телото. Парот на тагови <H1> и </H1> предизвикуваат пребарувачот да прикаже Bad Request како заглавие, што на корисникот ќе му се прикаже во два реда како на пример:

Bad Request

Your browser sent a request that this server could not understand.

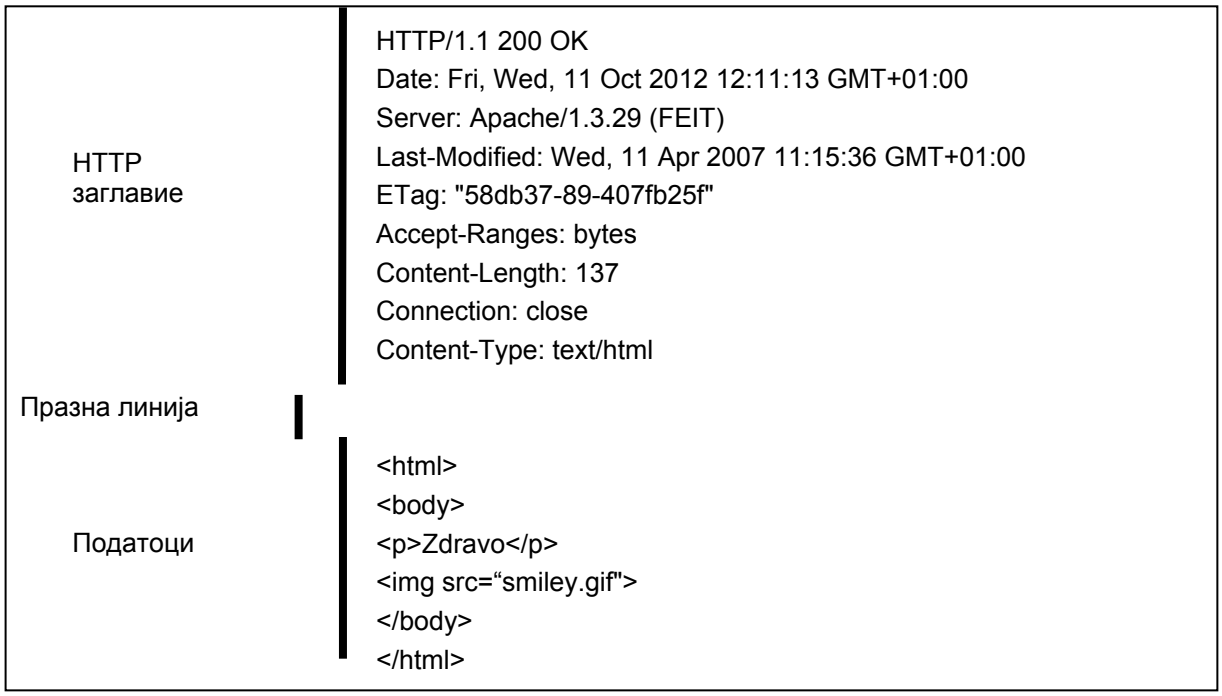
5.4.4 Постојаните конекции и должините на HTTP пораките

За да се испратат информациите за должината, HTTP ја користи истата репрезентација, односно истиот формат на рамка како кај E-mail (стандард RFC 822). Секоја HTTP трансмисија е составена од заглавие, празна линија и податочно поле. Во табелата на слика 5.4 се претставени неколку од најчесто употребуваните заглавија и нивното значење соодветно. Притоа, дел од нив се преземени од MIME кај email, т.е.

Content-Type и Content-Encoding ги имаат истите можни вредности како кај MIME, како и истиот формат кога се специфицираат во заглавието на пораката.

Заглавие (Header)	**Значење**
Content-Length	Големина во октети
Content-Type	Тип на содржината
Content-Encoding	Кодирање
Content-Language	Јазик

Слика 5.4 Примери на содржини кои можат да се појават во заглавието



Слика 5.5 Пример на HTTP response порака

На слика 5.5 е прикажан конкретен пример за една HTTP response порака, каде што може да се видат составните делови на една HTTP порака кои беа претходно опишани. Бидејќи во овој пример се работи за перзистентна конекција (HTTP 1.1), со оваа порака се затвара и конекцијата, имено поставено е "Connection: close". Дополнително HTTP може да содржи голем број на заглавија кои служат за размена на информации помеѓу клиентот и серверот.

HTTP овозможува и условни барања, кои што најпрво во заглавието ги дефинираат условите кои треба да бидат исполнети. Доколку одговорот е потврден,

серверот не одговара на барањето со што се избегнува непотребниот сообраќај. На ваков начин може да се избегне потребата од трансфер доколку не постои потреба.

5.4.5 Преговарање

HTTP протоколот освен за информирање на деталите за содржината,го користи заглавјето и за преговарање за можностите помеѓу клиентот и серверот, [22]. Можностите кои постојат и може да се преговара за нив содржат низа од карактеристични врски (дали се проверува право на пристап), претставување (дали се прифаќаат слики во jpeg формат или кои типови на слики можат да се прифатат), содржината (дали текстуалните датотеки мораат да бидат на англиски јазик) и контрола (време за кое ќе важи веб страницата). Постојат два вида на преговарања:

- Преговарање под контрола на серверот;
- Преговарање под контрола на клиентот (пребарувачот).

Преговарањето под контрола на серверот започнува со барање на пребарувачот. Барањето содржи листа од преференци. Серверот меѓу можните опции избира некоја која ги задоволува преференците на пребарувачот. На пример, ако некој документ се чува на повеќе јазици, а во барањето приоритет се дава на англискиот јазик, тогаш серверот ќе ја испрати англиската верзија.

Преговарањето под контрола на клиентот значи дека пребарувачот избира опција и два чекори. Пребарувачот, најпрво испраќа барање со прашање што е расположливо на страна на серверот. Серверот враќа список со можности. Пребарувачот избира една можност и праќа друго барање за да го добие саканиот документ. Недостатокот кај овој вид на преговарање е тоа што се потребни две интеракции со серверот, а предност е тоа што пребарувачот има потполна контрола над бирањето.

Пребарувачот го користи заглавјето Accept за да се договори за прифатените документи, како тие ќе бидат претставени и кодирањето кое се користи. Заглавјето користи листа од имиња на формати, на кои се доделени вредности за приоритет. На пример:

Accept: text/html, text/plain, q=0.5, text/x-dxi, q=0.8

Пребарувачот го прифаќа првиот вид на медиум text/html, но во случај ова да не постои може да го прифати text/plain. Вредностите кои се доделени во вториот и третиот дел се земаат како нивоа на приоритет.

Доколку не е наведено нивото на приоритет се зема q=1, а вредноста q=0 значи дека тој тип на документ не е прифатлив. Постојат низа од заглавја Accept кои одговараат на опишаните Content заглавја во MIME стандардот. На пример, пребарувачот може да ги испрати следните:

Accept-Encoding:

Accept-Charset:

Accept-Language:

Со овие заглавја се наведува кое кодирање, кои знаци и кои јазици пребарувачот е подготвен да ги прими.

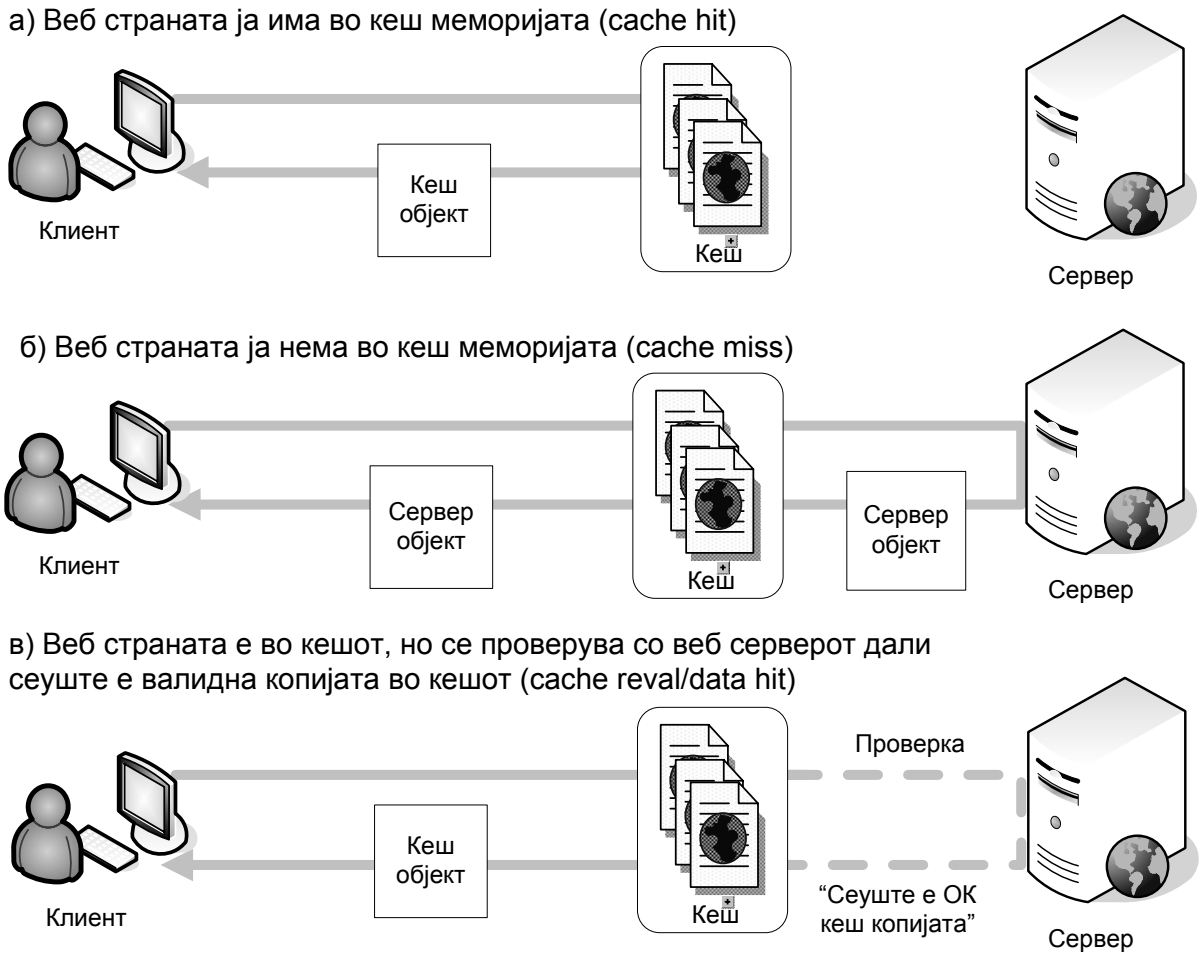
5.5 Кеширање (caching)

Целта на кеширањето е да се зголеми ефикасноста: елиминирање на непотребни преноси, намалување на времето на чекање, како и на волуменот на мрежниот сообраќај, [23]. Најочигледен аспект е чувањето: кога на веб страницата се пристапува прв пат, нејзината копија се сместува на диск на пребарувачот, или на прокси, а може и на двата. Подоцна, ако сакаме пристап до истата страна може да се скрати процесот на барање, бидејќи копијата на страницата се зема од кешот наместо од веб серверот.

Главното прашање кое се поставува во сите шеми на кеширање се однесува на времето-колку долго може да се чува документот во кешот? Од една страна, ако копијата предолго се чува во кешот таа застарува што значи дека сите промени кои се случуваат во оригиналот не се спроведуваат на копијата. Од друга страна, ако копијата не се чува доволно долго доаѓа до неефикасност затоа што при следниот пристап до истата страница мора повторно да се оди до серверот.

HTTP му дозволува на серверот да го контролира кеширањето на два начина. Првиот начин е серверот да може во одговорот на барањето да ги наведе деталите кои се однесуваат на кеширањето (дали може таа страница воопшто да се кешира, дали

прокси смее да ја кешира таа страница, кој може да пристапи до кешираната копија, време на чување на кешираната копија и ограничување во дозволените трансформации на копиите).



Слика 5.6 Начини на кеширање при HTTP комуникација

HTTP му дозволува на пребарувачот да наметне повторно потврдување на страницата. Ова за да се постигне пребарувачот испраќа барање за страницата и користи заглавје во кое бара максимално време на “старост” (времето од кога се чува копијата на страницата) да не биде поголемо од нула. Ниедна копија од страницата во кешот не може да го задоволи тој услов бидејќи староста на секоја копија е поголема од нула.

Поради ова, само првобитниот сервер може да одговори на барањето. Во текот на патувањето прокси серверот ќе добие свеж примерок за својот кеш, како и пребарувачот кој бара таков пристап.

Прокси серверите се важен дел од веб архитектурата затоа што овозможува оптимизација со која се намалува времето на чекање, како и оптовареноста на серверот. Меѓутоа, прокси серверите не се транспарентни - пребарувачот мора да се конфигурира така да се овозможи да се воспостави врска со локалниот прокси сервер, а не со вистинскиот извор, а прокси серверот мора да се конфигурира за да ги кешира копиите од веб страниците.

На пример, во некоја фирма во која голем дел од вработените користат Интернет и сакаат да постават прокси сервер. Фирмата, тогаш мора да ги конфигурира пребарувачот и прокси серверот така да пребарувачот го испраќа барањето на прокси серверот. Кога првиот корисник пристапи кон одредена веб страница, тогаш прокси серверот мора да ја има копијата од серверот кој управува со таа страница. Прокси остава копија во својот кеш и ја враќа бараната страница како одговор на барањето од клиентот. Кога следниот пат некој друг корисник ја побара истата страница, прокси серверот ги зема податоците од кешот и не праќа барање преку Интернет. Со ова може значително да се намали сообраќајот за тој сајт на Интернет.

HTTP содржи експлицитна поддршка за прокси серверите. Имено, протоколот HTTP точно го одредува начинот на кој прокси го обработува секое барање, како прокси треба да го сфати барањето, како пребарувачот треба да преговара со прокси и како прокси преговара со серверот.

На пример, едно заглавје овозможува прокси да ја потврди својата автентикација на серверот, друго заглавје овозможува секој прокси кој обработува баран документ да ја забележи идентификацијата, така да крајниот примач ќе добие листа од сите посредни прокси сервери. На крајот, HTTP му дозволува на веб серверот да го контролира прокси серверот во врска со начинот на кој ја обработува секоја веб страница. Серверот во одговорот може да го вклучи заглавјето Max-Forwards и на тој начин да го ограничи бројот на прокси сервери кои го обработуваат документот пред да се испорача на пребарувачот. Ако веб серверот одреди само еден, тогаш

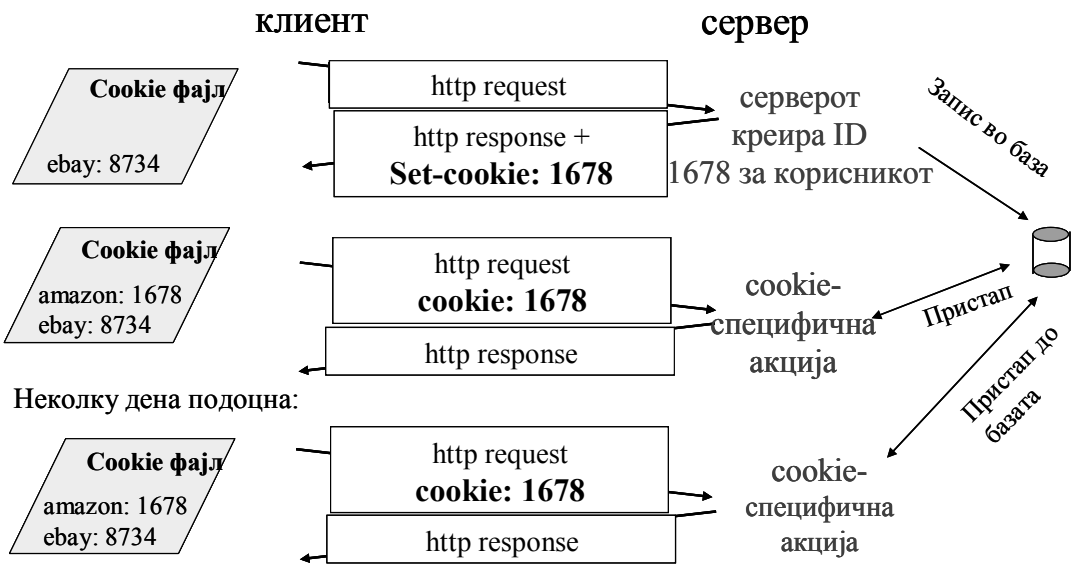
Max-Forwards:1

на патот од серверот до пребарувачот дозволен е само еден прокси. Вредност нула во ова поле значи дека му е забрането на прокси сервер да го обработува документот.

5.6 Веб колаче (cookie)

Што е веб колаче (cookie)? **Веб колаче** (анг. **cookie, web cookie,** или **HTTP cookie,** што се синоними) е мала информација испратена од сервлет (servlet) кон веб пребарувач. Притоа, сервлет е јава програм кој работи во рамките на веб серверот, додека аплет (applet) е јава програм кој работи во рамките на веб пребарувачот.

Денес сите поголеми веб сајтови користат веб колачиња. Веб колачето автоматски се поставува од страна на веб серверот во клиентскиот хост (преку веб пребарувачот) при праќање на response порака кон клиентот (како одговор на неговото барање - request порака). Ако веќе има поставено веб колаче од даден веб сајт во клиентот, тогаш тоа веб колаче се испраќа при даден request кон тој сајт (секако, доколку се посети веб сајтот од страна на клиентот). Сите веб клиенти денес, т.е. веб пребарувачи (browsers), овозможуваат да се контролира користењето на веб колачиња.



Слика 5.7 Пример за веб колаче (cookie)

Веб колачето (cookie) има **четири компоненти**:

- cookie header линија во HTTP response пораката
- cookie header линија во HTTP request пораката
- cookie фајлот се чува на корисничкиот хост и е управуван од веб пребарувачот

– Има соодветен запис во одредена база на податоци на веб сајтот кој го поставил на клиентскиот хост

За што се користат веб колачињата? Постојат повеќе примени на веб колачињата, а некој може да смисли и дополнителна примена за некоја конкретна цел. Еве неколку примери за примена на веб колачиња:

– Авторизација
– Препораки
– Следење на кориснички сесии (на пример: кај webmail)

Веб колачето му овозможува на даден веб сајт "да научи нешто" за корисникот доколку корисникот тоа го дозволи тоа преку сетирањата на неговиот веб пребарувач за заштита на приватните информации од аспект на веб колачиња (со тоа што ќе дозволи да се складираат веб колачиња од веб сервери локално на неговиот уред, со посредство на веб пребарувачот). На пример, може да се дозволи да се дава името или email адресата на веб серверот преку веб колачето. Друга примена на веб колачето е кај машините за пребарување на Интернет (search) за да "научат" повеќе.

Еден конкретен пример на поставување на веб колаче кај клиентот преку HTTP response од страна на веб серверот на даден HTTP request од клиентот е прикажан на слика 5.7. На истата слика е прикажано и праќањето на веќе складиран веб колаче од страна на веб клиентот кон веб серверот преку HTTP request порака испратена од клиентот.

5.7 Дискусија

Во оваа глава го опфативме најзначајниот Интернет сервис (апликација) кој изворно е создаден како Интернет технологија. Всушност, заедно со електронската пошта, веб технологиите се најважните Интернет технологии кои воопшто се појавиле до денес. Под веб (WWW - World Wide Web) го подразбираме HTTP како протокол на апликациско ниво кој функционира на принципот клиент - сервер, каде што веб клиентот праќа барања во веб серверот и добива одговори или податоци, т.е. веб страни со различни мултимедиски објекти на нив.

Појавата на вебот (т.е. HTTP) на почетокот на 1990-те години ја овозможи експанзијата на Интернет во глобални рамки до сите типови корисници, излегувајќи од претходните простори на истражувачката академска средина каде што е создаден и ограничената бизнис средина во која почнал прво да навлегува кон крајот на 1980-те години. Вебот е причината на успехот на Интернет, бидејќи овозможи поставување и пристап до мултимедиска содржина на едноставен начин (преку веб страници, поставени на дадени веб сајтови кои се сместени на веб сервери), како и поврзување на содржините од еден сајт со содржините (на веб страници) од други сајтови преку користење на хиперлинкови. На тој начин набрзо по појавата на вебот целиот Интернет стана испоповрзан преку хиперлинкови поставени на веб страниците кои се испреплетуваат и го водат корисникот низ Интернет мрежата според неговите желби и интереси. Во тој поглед, вебот и неговата појава може слободно да се сметаат за револуција во Интернет мрежата, а појавата на вебот всушност го промовираше Интернет во она што претставува денес, единствена пакетска мрежа за сите постоечки и идни сервиси. Така, голем број на обични корисници ги поистоветуваат вебот и Интернет мрежата, па често пати се смета во таквите размислувања дека Интернет = WWW, што укажува на огромното влијание кое го има направено вебот врз Интернет, а преку него и на општеството денес. Имено, денес кога го користиме Интернет за работа (преку веб сајтови на институции или компании), за забава (преку веб сајтови со забавни или интересни мултимедиски содржини составени од текст, слики, аудио, видео, податоци), за информации (преку веб сајтови за информации за вести, временска прогноза, производи, цени, итн. итн.), за комуницирање со други луѓе (преку веб сајтови на социјалните мрежи), итн., ја користиме веб технологијата. Дури и графичките кориснички интерфејси на различни уреди (дури и кога тие не се поврзани на Интернет мрежата) се базирани на веб технологијата каде што различните информации, менија и опции се претставени преку веб страници. И кога се користи некоја друга Интернет апликација (т.е. сервис) во речиси сите случаи корисничкиот интерфејс преку кој се пристапува до апликацијата (т.е. сервисот) е базиран на веб, како што ќе видиме како пример и во следната глава за мултимедиски комуникации преку Интернет.

На крајот, да резимираме, вебот (WWW) во многу елементи е најважниот сервис кој изворно се појавил во Интернет досега и кој ги креира облиците и формите преку кои се користи Интернет мрежата денес со различните апликации и сервиси во неа.

Глава 6

Мултимедиски комуникации преку Интернет

Пред да започнеме со прегледот на конкретните протоколи за пренос на мултимедиски содржини во реално време преку IP (Internet Protocol), да се задржиме на некои општи информации за тоа што се протоколи и кое е нивното значење.

Под мултимедиска комуникација преку Интернет подразбираме пренос на говор, видео и/или аудио во реално време. Притоа, имајќи ја во предвид специфичноста на говорот (телефонијата) како конверзациски двонасочен сервис, во оваа глава акцентот ќе биде на преносот на видео и аудио мултимедиски информации преку Интернет.

За мрежен пренос на дигиталните видео сигнали (streams) многу повеќе се користи UDP протоколот, бидејќи природата на видеото како медиум преставува течение во реално време и најчесто многу е поважно да се зачува реалниот тек на информациите отколку да се препраќаат сите загубени пакети (како кај TCP), посебно имајќи ја во предвид чувствителноста на преносот во реално време во однос на доцнењето на пакетите. Затоа за пренос се користи UDP протоколот на транспортно, досега сигнализациските и контролните информации често е погодно да се праќаат преку TCP на транспортно ниво, имајќи ја во предвид важноста на контролните информации од крај до крај (меѓу серверот и клиентот). Од аспект на стабилност и квалитет на преносот преку IP треба да се погрижиме да обезбедиме квалитетна, стабилна и сигурна мрежа, што подразбира да има ограничено доцнење од крај до крај и да има загарантиран битски проток.

Над овие мрежни протоколи доаѓаат апликациските протоколи, кои се користат за интеркомуникација меѓу мрежно свесните апликации, и служат за пренос на соодветно форматирани информации разбирливи за апликациите што ги користат. Во овие протоколи спаѓаат и специфичните протоколи за пренос на видео сигнали (video streams) како и контролните протоколи за истите.

Листата на ваквите протоколи не е крајна и конкретна, а ние ќе се задржиме на најкористените и најквалитетните. Некои од протоколите за транспорт на мултимедиска содржина (на пример: видео) кои активно се користат се:

- HTTP – Hyper Text Transport Protocol, [6],
- RTP – Real-time Transport Protocol, [24],
- RTCP – Real-time Transport Control Protocol, [24],
- RTSP – Real-time Streaming Protocol, [25],
- MMS – Microsoft Media Server.

Пред да почнеме со деталната анализа, да се задржиме малку на посебностите и специфичностите при преносот на дигитален видео сигнал, како и за проблемите што настануваат и со кои се соочуваме при користење на IP базирана мрежа за пренос на дигитално видео.

Најпрво, важно е да споменеме дека видео сигналот во дигитален облик секогаш се кодира со соодветен кодек (codec), чија што примарна намена е да го компресира видео сигналот, за да зафаќа што е можно помал пропусен мрежен опсег (bandwidth). Ваквиот кодиран видео сигнал потоа се пакува во битски потоци (bitstreams) од IP пакети, кои потоа користејќи одреден протокол (на пример RTP) се пренесуваат преку одреден транспортен протокол (TCP или UDP).

За ваквиот тип на пренос да се одвива квалитетно и стабилно, клучни се неколку параметри:

- Големина на пропусен опсег (bandwidth) – параметар што задолжително мора да го исполнува секоја мрежа од почетниот до последниот јазел, за да може видео сигналот да се пренесе во полна големина; големината на пропусниот опсег мора да биде соодветна на компресираниот видео сигнал плус контролните пакети од самите видео протоколи и overhead-от што секоја транспортна технологија го внесува при обезбедување на преносот.
- Губење на пакети (packet loss) – можеби најважниот параметар за квалитетен видео пренос; најчесто се користат одредени механизми за исправка на грешки

за да се спречи загубата на пакети.

- Мрежно доцнење (network delay) – потребно е доцнењето на пакетите низ мрежата да е што е можно помало за да имаме течение на видео сигналот во реално време; се користат разни механизми за да се избегнат мрежните доцнења, на мрежно ниво се користи обликување на мрежниот сообраќај, а на апликациско ниво се воведуваат баферирања (buffering) на видео сигналот неколку секунди однапред.
- Синхронизација (synchronization) – поради фактот дека видеото се одвива во реално време потребен е точен временски извор за сите системи кои се вклучени во преносот на видео сигналот генерално станува збор за синхронизација меѓу сите рутери вклучени во преносот, како и на системите/серверите што се одговорни за генерирање и испорака на видео сигналот.

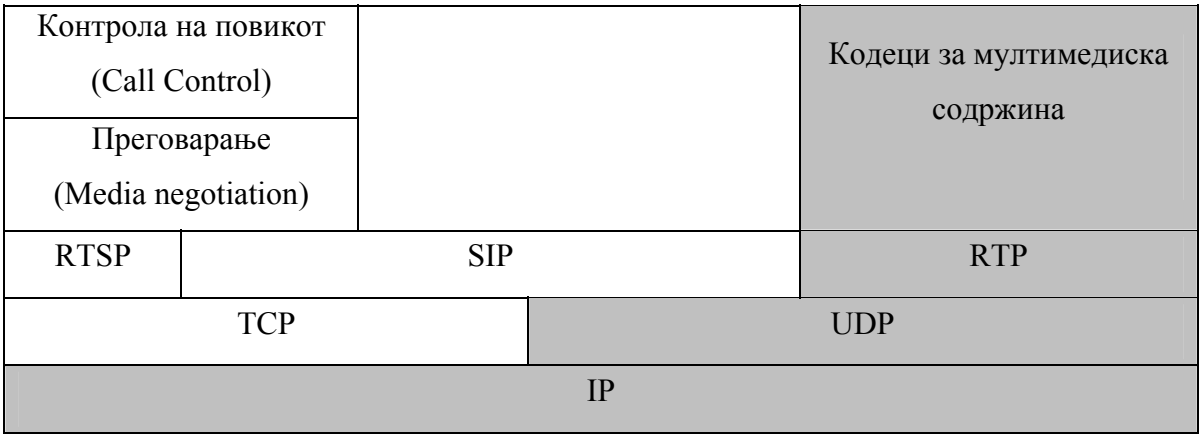
Според квалитетот и исполнувањето на сите овие параметри се дефинира и квалитетот на преносот на видео сигналот преку IP.

Како што веќе споменавме, постојат повеќе стандардизирани протоколи на апликациско ниво за пренос на видео преку IP, од кои некои се отворени и може да се користат без лиценцирање, додека некои се затворени и е потребно користење на лиценца за нивното користење. Во оваа глава, посебен осврт ќе направиме на најупотребуваниот, а воедно и најдобриот протокол за дигитален пренос на видео сигнали преку IP, а тоа е RTP (Real-time Transport Protocol), со неговиот контролен протокол RTCP (Real-time Transport Control Protocol), [24].

6.1 RTP - Real-time Transport Protocol

Real-time Transport Protocol (RTP) е клучен стандард за транспорт на аудио и видео преку IP мрежите и се стреми да обезбеди сервиси за таа намена. Овие сервиси вклучуваат: временско усогласување, детекција и корекција на загуба, идентификација на капацитетот и изворот, повратна информација за квалитетот на прием, синхронизација на содржината и управување. Во почетокот RTP бил дизајниран за употреба при multicast конференции, но од тогаш се покажал корисен во широк спектар

на апликации: во H.323 видео конференции, webcasting и ТВ дистрибуција, во фиксната и мобилната телефонија, итн.



6.1 IETF протоколен стек за мултимедиски комуникации

6.2 ITU протоколи за телеконференции

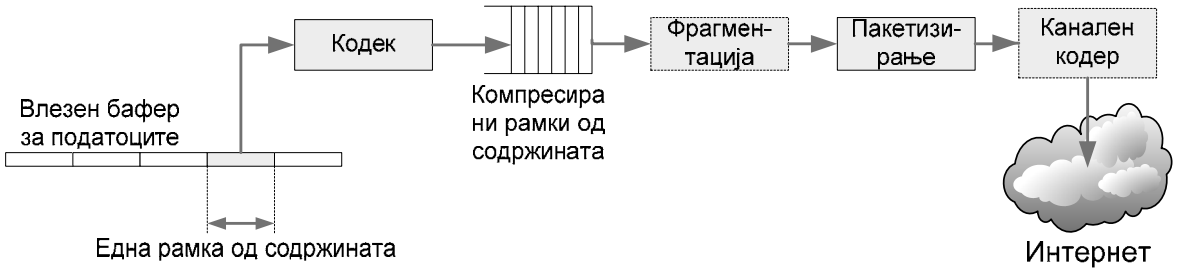
Протоколот се употребува во точка-до-точка (point-to-point) комуникација, како и во multicast сесии со илјадници корисници, во low-bandwith апликации за мобилна телефонија, а во последно време и за испорака на некомпресиран сигнал за HDTV (High-Definition Television).

Покрај RTP, за функционирање на еден комплетен систем потребни се и други протоколи и стандарди, пред се протоколите и стандардите за најава, иницирање и контрола на сесија, компресија на содржината и мрежен транспорт.

На слика 6.1 и слика 6.2 прикажан е односот, според IETF и ITU, меѓу протоколите за договарање и контрола, транспортниот слој (RTP), алгоритмите за компресија и декомпресија и основната мрежа.

6.1.1 RTP извор (sender)

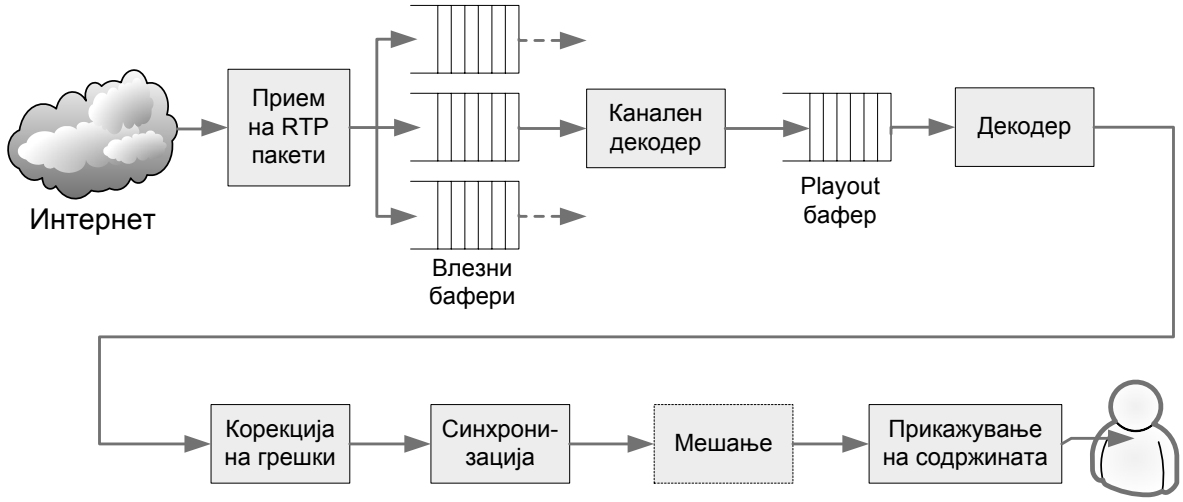
Изворот е одговорен за прифаќање и трансформација на аудио-визуелните податоци за пренос, како и за генерирање на RTP пакети. Исто така може да учествува во корекција на грешките и контрола на натрупувањето приспособувајќи ја содржината која се пренесува во однос на повратните информации од примачот. На слика 6.3 е прикажан процесот на испраќање.



Слика 6.3 Преглед на постапката на кодирање на мултимедиската содржина од изворот на информации (во примеров, видео информации) до влезот во Интернет

6.1.2 RTP примач (receiver)

Примачот е одговорен за примање на RTP пакетите од мрежата, корекција на загуби, временско усогласување, декомпресија на содржината и приказ на резултатот на корисникот. Исто така, на изворот му дава информација за квалитетот на прием овозможувајќи му да го прилагоди преносот за примачот и одржува база на учесници во сесијата. На слика 6.4 е прикажан еден модел на примач на мултимедиска содржина.



Слика 6.4 Процесирање на RTP пакетите со дадена мултимедиска содржина на приемната страна

6.1.3 Основни филозофии на дизајнот

Предизвикот со кој се соочија дизајнерите на RTP беше да се изгради механизам за пренос на робусни содржини во реално време преку несигурно транспортно ниво, [26].

Тие ја постигнаа целта со користење на две филозофии: обликување на апликациско ниво (application-level framing) и принцип од крај до крај (end-to-end principle).

Обликување на апликативно ниво

Основната теза на оваа филозофија е дека само апликацијата поседува доволно знаење и информации за своите податоци за да одлучи како тие податоци треба да бидат транспортирани. Тоа значи дека транспортниот протокол треба да ги прифати податоците и да даде информација за завршениот пренос, за да во случај на појавување на некоја грешка апликацијата преземе мерки за нејзино отстранување.

Апликацијата има многу начини да се поврати од грешките на мрежно ниво и тие зависат од апликацијата и од сценариото во кое се употребува. Во некои случаи ќе биде потребен повторен транспорт на истиот пакет, во други може да се транспортира

копија со послаб квалитет од оригиналот а можно е и занемарување на загубата. Овие опции се возможни само ако апликативното ниво блиску соработува со мрежното ниво.

Целта на application-level framing е во спротивност со TCP, кој ја крие особината на IP мрежата да губи пакети овозможувајќи сигурна испорака за сметка на навременоста. Затоа пак добро се вклопува со UDP базираниот транспорт и карактеристиките на содржините во реално време.

Принцип од крај до крај (end-to-end principle)

Другата филозофија прифатена од RTP е end-to-end principle. Тоа е еден од пристапите при дизајнирање на систем кој треба доверливо да комуницира преку мрежа. Едниот пристап е пренесување на одговорноста за доверлив пренос од точка до точка. Вториот пристап е пренесување на одговорноста за преносот на крајните точки овозможувајќи end-to-end доверлив пренос без оглед на доверливоста на точките (јазлите) во средината.

Главната последица од end-to-end principle е тоа што интелигенцијата во системот се стреми да се искачи на врвот од протоколниот стек.

Ако системите кои ја сочинуваат мрежната патека никогаш не превземаат одговорност за податоците тогаш тие можат да бидат едноставни и не мора да се робусни. Тие можат да ги занемарат податоците кои не можат да ги пренесат бидејќи крајните точки ќе ги повратат без нивна помош.

End-to-end principle имплицира дека интелигентноста е на крајните точки, а не во мрежата. Резултатот од овој дизајн се паметни, свесни за мрежата крајни точки и проста мрежа.

6.1.4 RTP спецификација

RTP работи преку UDP/IP, истовремено подобрувајќи го со детекција на загуба и известување за квалитетот на прием, временско опоравување и синхронизација, идентификација на капацитетот и изворот, како и означување на значајните настани во течението на мултимедијалните информации (media streams). Најголем дел од

имплементациите на RTP се дел од апликација или библиотека која работи врз UDP/IP обезбеден од оперативниот систем.

RTP се состои од два дела: протокол за трансфер на податоци (data transfer protocol) и контролен протокол (control protocol).

RTP data transfer protocol управува со испорака на податоците во реално време меѓу крајните системи.

RTP control protocol (RTCP) обезбедува повратна информација за квалитетот на прием, идентификација на учесникот и синхронизација на течението на мултимедијалните информации. RTCP работи паралелно со RTP и периодично ги обезбедува овие информации. Иако податочните пакети се испраќаат на неколку милисекунди контролниот протокол работи на неколку секунди.

RTP профили

Важно е да се знаат можностите на RTP протоколот затоа што тој не е комплетен во две карактеристики.

Прво, стандардот не специфицира алгоритми за playout и временска регенерација, синхронизација помеѓу течението на мултимедиските информации (media streams), прикривање на грешките и нивна корекција, или контрола на застој. Ова се должности на дизајнерот на мултимедиската апликација, затоа што различни апликации имаат различни протоколи и не би било добро стандардот да ги одредува овие однесувања.

Второ, некои детали од транспортот се оставени отворени за модификација. Можни се модификации на профилите како и на дефинирањето на форматот на носивоста на пакетот (payload).

RTP Payload формати

Последен дел од RTP се payload форматите кои дефинираат како одреден тип на медија се пренесува со RTP. Payload форматите се референцирани од RTP профилите и можат да дефинираат одредени параметри на RTP data transfer протоколот.

RTP payload форматите и профилите со релациите меѓу нив креираат една уникатна целина (namespace). Во оваа целина, релацијата на идентификаторот на типот на носивоста на пакетот со неговиот формат, овозможува тесна релација на носивоста

(payload) со одреден мултимедијален кодек, со цел оптимизација на преносот на информациите во зависност од типот на мултимедијалната информација што се пренесува. Во некои случаи можеме да имаме статично врзување помеѓу типот и форматот на носивоста, додека во други случаи возможно е и динамичко врзување преку некој надворешен контролен протокол.

RTP сесии

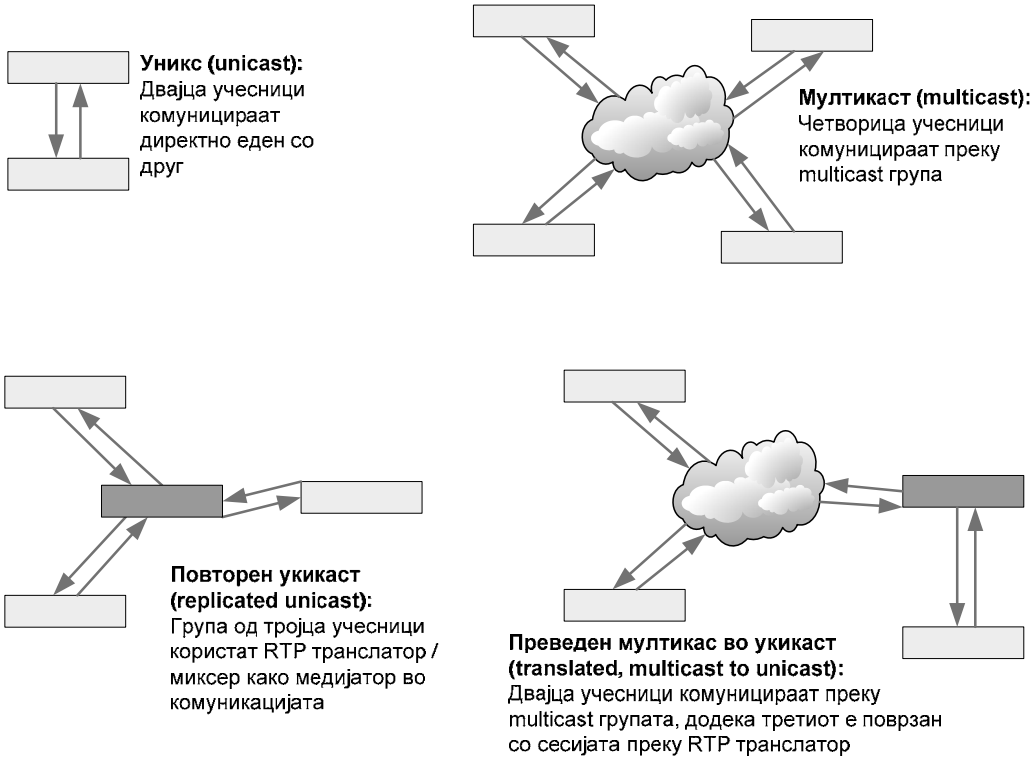
RTP сесија се состои од група на учесници кои комуницираат користејќи RTP. Учесникот може да биде активен во една или повеќе сесии. Пример, во една сесија може да разменува аудио податоци, а во друга сесија видео податоци. За секој учесник сесијата е идентификувана со мрежната адреса и по еден пар порти за праќање и примање на податоци. Портите за праќање и примање можат да бидат исти. Секој пар на порти се состои од порта со парен број за RTP податочни пакети и порта со непарен број за RTCP. Стандардни порти се 5004 и 5005 за UDP/IP, но многу апликации динамички ги одредуваат портите при иницијализација на сесијата. RTP сесиите се дизајнирани за транспорт на еден тип медија и за одредена мултимедијална комуникација секој тип на медија користи посебна RTP сесија.

Сесијата може да биде unicast, директно помеѓу два учесници (point-to-point сесија) или до централен сервер кој понатаму ги редистрибуира податоците, или multicast кон група на учесници. Сесијата исто така треба да биде ограничена на единствен транспортен адресен простор. RTP транслаторите можат да се употребат за врска меѓу unicast и multicast, или помеѓу IP и некој друг транспорт како IPv6 или ATM.

RTP Data Transfer пакет

Пакетот се состои од 4 дела:

1. RTP header
2. Header extension (optional)
3. Payload header (optional)
4. Payload data



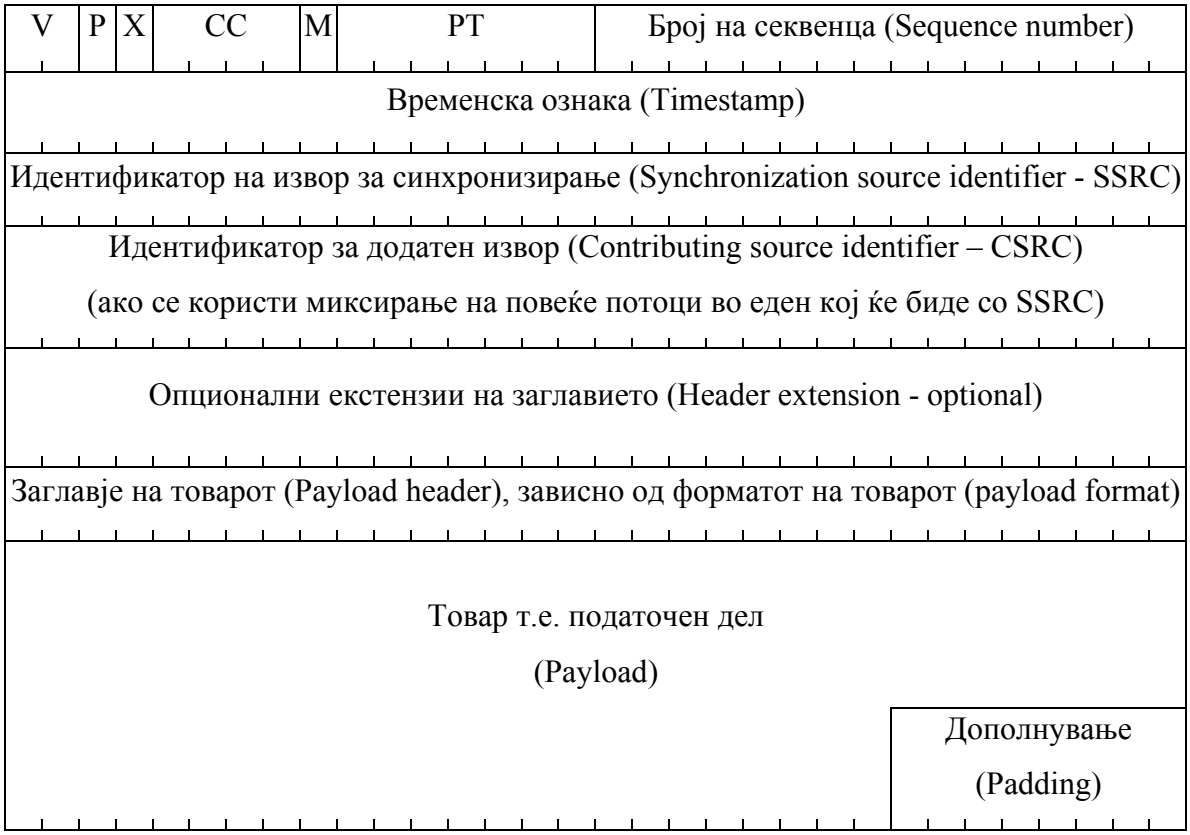
6.5 Приказ на различни типови на RTP сесии

Валидација на пакетите

Бидејќи RTP сесиите користат динамички доделени порти од особена важност е проверката дали примените пакети се навистина RTP пакети.

Постојат два вида на тестови:

1. Per-packet checking – проверка на секој пакет базирана на фиксна вредност на некое од полињата во заглавието. Пример: пакетите кај кои вредноста на верзијата не е 2 – се невалидни.
2. Per-flow checking – базирано на одреден патерн во полињата на заглавието. Пример: Ако SSRC е константно и бројот на секвенцата се зголемува за еден со секој примен пакет и timestamp интервалите се соодветни на payload типот, тогаш тоа е скоро сигурно RTP flow.



V= Version number; P = Padding; X = extensions; CC = број на извори кои се миксираат (мултиплексираат); M = Marker; PT = Payload Type.

Слика 6.6 RTP пакет

6.2 RTCP – Real-time Transport Control Protocol

RTCP се состои од три компоненти: структура на пакети, правила на тајминг и база на учесници.

Постојат неколку типови на RTCP пакети од кои пет се стандардни. Заедно со нив се и правилата според кои тие се спојуваат во збирни пакети и кои понатака ќе бидат транспортирани.

Интервалот помеѓу пакетите е познат како интервал на извештај (reporting interval). Сите RTCP активности се случуваат во правилни последователни временски дистанци одредени со интервалот на извештајот. Покрај времето помеѓу пакетите, постои и време за кое се пресметува статистика за квалитетот на прием како и време за обнова на описот на изворот и синхронизација на аудиото и видеото. Интервалот зависи од форматот на мултумедиските информации и од големината на сесијата.

Типична вредност е 5 секунди за мали сесии но за големи групи може да се наголеми и на неколку минути.

Секоја имплементација би требало да содржи и база на учесници во која би се сместувале информациите добиени од RTCP пакетите. Оваа база се користи за пополнување на пакетите за извештај на прием кои мора да се испраќаат периодично, но и за синхронизација на аудио и видео делот од мултимедијалните информации како и управување со описот на изворот.

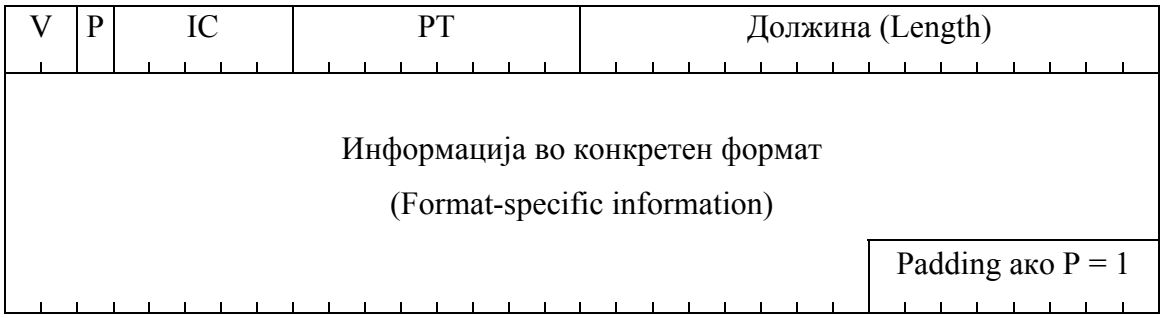
Транспорт на RTCP пакети

Секоја RTP сесија се состои од мрежна адреса и пар порти, една за RTP податоци и една за RTCP податоци. RTP податоците треба да се на парна порта, а RTCP на една порта погоре. На пример, ако медија податоците се праќаат преку UDP на порта 5004, тогаш контролниот протокол ќе биде на истата адреса на UDP порта 5005.

Секој учесник во сесија треба да праќа RTCP пакети и за возврат добива RTCP пакети од сите учесници. Peer-to-peer природата на RTCP им овозможува на учесниците да ги имаат информациите за сите други учесници: нивно присуство, квалитет на прием, а опционално и лични детали како име, e-mail адреса, локација, телефонски број.

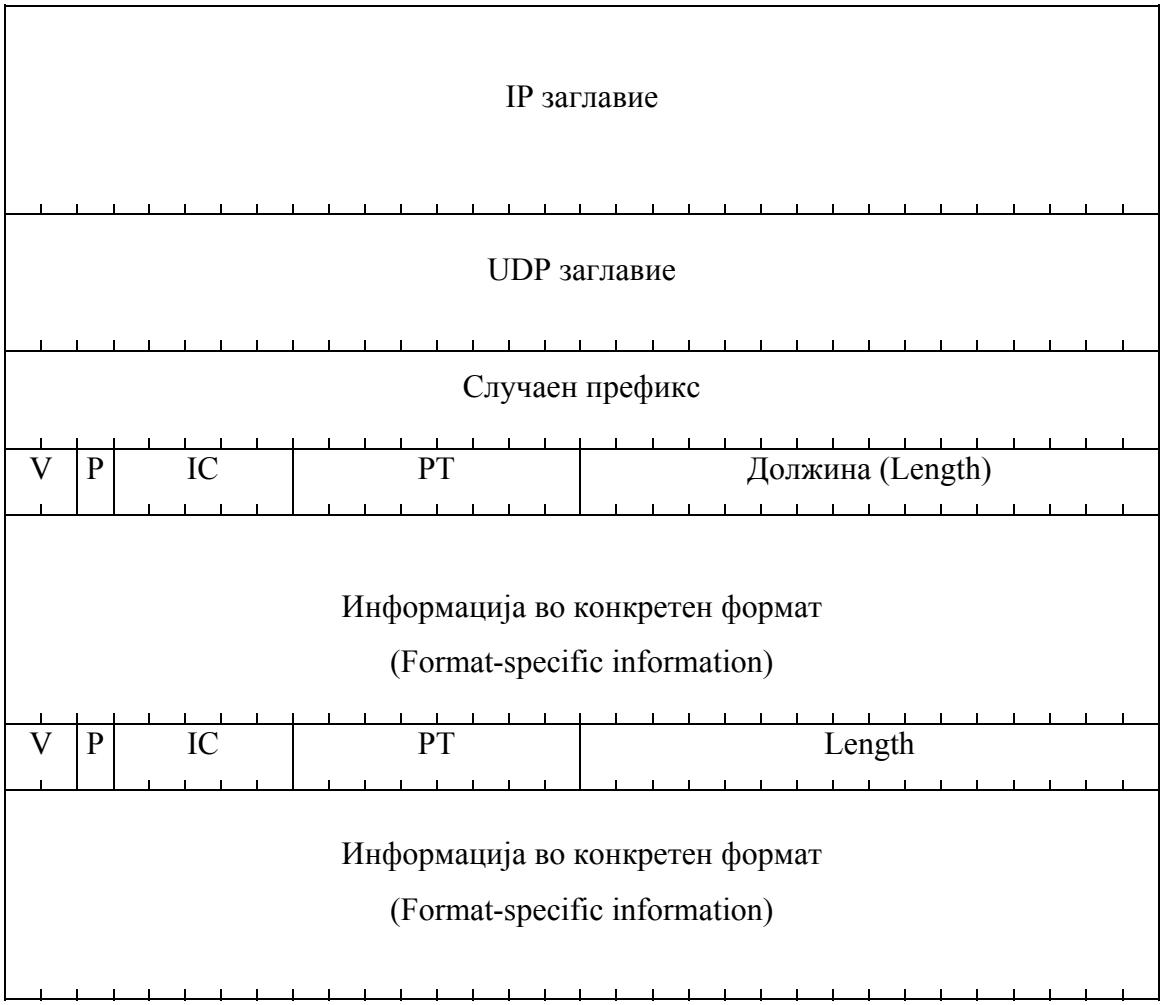
Формат на RTCP пакети

Со RTP спецификацијата се дефинирани пет типови на пакети: извештај на примачот (receiver report - RR), извештај на праќачот (sender report - SR), опис на изворот (source description - SDES), управување со членството (membership management - BYE) и пакети дефинирани од апликацијата (application-defined - APP). Сите типови ја задржуваат основната структура на пакетот прикажана на следната слика со тоа што format-specific делот содржи информации во зависност од типот на пакетот:



V = version number; P = padding; IC = item count; PT = packet type

Слика 6.7 RTCP пакет



Слика 6.8 RTCP пакет

Заглавието (header) кое е исто кај сите пет типови е долго четири октети и ги содржи следниве полиња:

1. Version number (V) – За сегашната верзија на RTP бројот на верзијата секогаш

има вредност 2.

2. Padding (P) – Битот за полнење означува дали пакетот е наполнет повеќе од неговата вообичаена големина. Ако овој бит е поставен значи дека еден или повеќе октети се додадени на пакетот. Последниот октет содржи информација за тоа колку октети се додадени.
3. Item Count (IC) – Некои типови на пакети содржат и листа на елементи. Ова поле кај овие типови се користи за да се знае бројот на елементи во пакетот. Во секој RTCP пакет може да има максимум 31 елемент. Ако се потребни повеќе од 31 елемент апликацијата треба да генерира повеќе RTCP пакети.
4. Packet Type (PT) – Го означува типот на информација содржана во пакетот. Пет стандардни типови се дефинирани во RTP (во иднина можат да се дефинираат уште):

   200 Sender report

   201 Receiver report

   202 Source description message

   203 Bye message

   204 Application specific message

5. Length – Ја означува должината на пакетот после стандардното заглавие. Се мери во единици од 32 битни зборови бидејќи сите RTCP пакети се мултипли од 32 бита должина.

RTCP пакетите никогаш не се транспортираат индивидуално туку се групираат во збирни пакети. Структурата на збирниот пакет е прикажана на слика 6.8.

Постојат 5 типа RTCP пораки како што е елаборирано погоре во текстот. Овде накратко ќе дадеме објаснување за намената за секој од овие типови пакети:

- **Приемниците** периодично испраќаат **Receiver Report** за да го информираат испраќачот за состојбата при приемот:
  - Најголем примен секвенцијален број
  - Кумулативни и процентуални загуби
  - Информации за џитерот
  - Време на последниот RTCP Sender Report
- **Испраќачите** периодично испраќаат **Senders Report** кој обезбедува:

  - Апсолутна временска ознака (timestamp) што е неопходно за синхронизација на одделените медиски потоци (на пример: видео поток со придружено аудио).
- Испраќачот праќа и **Source description message** кој дава опис на испраќачот
  - Задолжително се праќа: user@host
  - Опционално се праќа: e-mail адреса, телефонски број, географска локација, тип на апликацијата и сл.
- Испраќачот праќа **bye** порака кога го затвара потокот
- **Application specific message** служи да овозможи на апликацијата да дефинира message type (на пример: праќање на Word фајл).

База на учесници

Секоја апликација во RTP сесија има база на податоци со информации за учесниците и за самата сесија. Информацијата за сесијата може да ги содржи следниве променливи:

- RTP bandwidth – типичниот bandwidth конфигуриран при стартување на апликацијата;
- RTCP bandwidth fraction – колкав дел од RTP опсегот е резервиран за RTCP. Типично е 5% но може да биде и 0% што значи дека RTCP не се праќа;
- Просечната големина на сите RTCP пакети испратени и примени од учесникот;
- Број на учесниците во сесијата, број на учесници кога последен пат учесникот испратил RTCP пакет, и дел од оние кои испратиле RTP податочен пакет во претходниот извештаен период;
- Време кога последно е испратен RTCP пакет и време кога повторно треба да се испрати;
- Индикатор за тоа дали имплементацијата има испратено RTP податоци откако се испратени последните два RTCP пакети;
- Индикатор за тоа дали имплементацијата воопшто има испратено RTCP пакет.

Имплементацијата треба да има и променливи кои се вклучуваат во RTCP пакетите:

- Број на RTP пакети и октети кои ги има испратено;
- Последниот секвенцијален број;
- Однос помеѓу RTP часовникот и RTP-format timestamp.

Timing правила

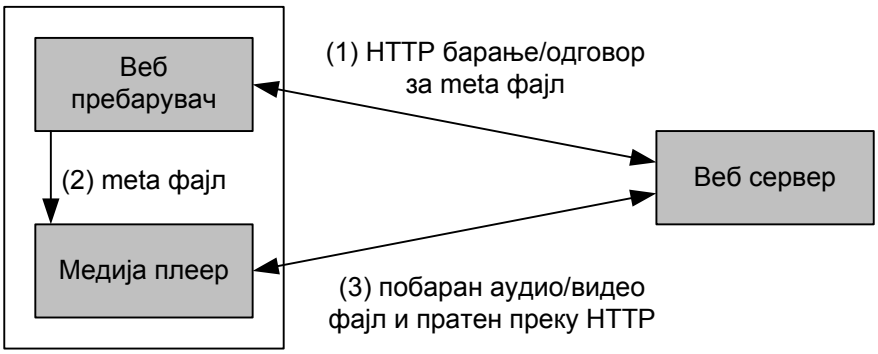
Брзината со која секој учесник праќа RTCP пакети не е фиксна туку зависи од големината на сесијата и форматот на мултимедијалните информации што се пренесуваат. Целта е да се ограничи RTCP сообраќајот на 5% од расположливиот капацитет (bandwidth). Ова се постигнува со намалување на брзината и зачестеноста со која се праќаат RTCP пакетите. Меѓу контролните пакети , 25% се алоцирани на Sender Reports и 75% Receiver Reports.

Просечното време кое поминува помеѓу две праќања на RTCP пакети се вика интервал на извештај (reporting interval) кој зависи од: капацитетот кој е алоциран за RTCP, просечната големина на RTCP пакетите и вкупниот број на учесници со делот на оние кои се испраќачи.

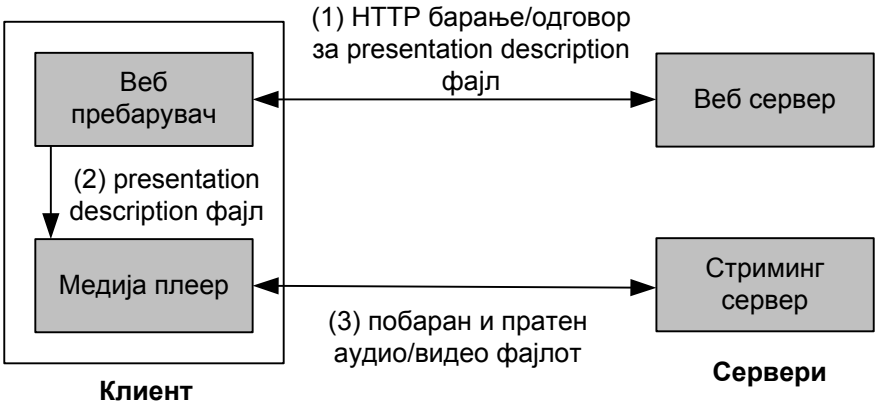
6.3 Real-Time Streaming Protocol (RTSP)

Real-Time Streaming Protocol (RTSP), [25], е сесиски протокол за стриминг медија преку Интернет. Една од главните функционалности на RTSP е поддршката за VCR (видео плеер) контроли како стоп, пауза/продолжи , брзо нанапред и брзо наназад. Како додаток, RTSP исто така нуди опции за избор на канал за пренос (UDP, multicasting UDP или TCP) и доставувачки механизми базирани на RTP. RTSP работи и во мултикаст како и во уникаст.

Кај RTSP, наместо да се симнуваат цели мултимедиски фајлови и да се стартуваат, мултимедиските податоци се праќаат во потоци (streams). RTSP е клиент-сервер протокол за контролиран пренос на мултимедиски содржини преку Интернет. Тој е апликациски протокол (RFC 2326), со default порта 554. Протоколот е развиен од RealNetworks, Netscape Communications и универзитет Columbia.



а) RTSP со веб сервер (без стриминг сервер)



б) RTSP со веб сервер и со streaming сервер

Слика 6.9 Комуникација клиент-сервер кај RTSP

Друга битна функција на RTSP е создадат и контролираат потоци на аудио и видео медија помеѓу медија сервери и медија клиенти. Специфично, RTSP ги овозможува следниве операции:

- **Media retrieval:** клиентот може да побара од медија серверот да се воспостави сесија за да се достави бараната медиска содржина.
- **Inviting a media server to a conference:** медија серверот може да биде поканет на конференција за да презентира медија.
- **Adding media to an existing session:** серверот или клиентот може да се известуваат меѓусебно за секоја додатна медија која станала достапна за воспоставената сесија.

RTSP е наменет да ги овозможи истите сервиси за стриминг аудио и видео како шт HTTP ги прави за текст и графика (HTTP: HyperText Transfer Protocol). Овој

протокол е дизајниран да има слична синтакса и операции така што сите екстензивни механизми на HTTP може да се имплементираат на RTSP.

Kaj RTSP секоја презентација и медија поток (stream) е идентификувана преку RTSP URL (URL: Universal Resource Locator). Целосната презентација и својствата на медијата се дефинирани во описен фајл кој може да ги вклучи кодирањето, RTSP дестинацискатa URL локација, портата и други параметри.

Презентацискиот описен фајл може да се добие од клиентот користејќи HTTP, e-mail и други средства.

Како заклучок, RTSP е дизајниран да иницира и директно достави стриминг медија содржина од медија серверите до клиентите.

6.4 Мултиплексирање на мултимедиски содржини

Во пакетските мрежи е карактеристично да имаме различни типови на информации сместени во различни содржини. Мултимедиските содржини вклучуваат повеќе различни типови информации во себе (аудио, видео, податоци), [26]. Под пренос на мултимедиска содржина подразбираме пренос на различните типови информации мултиплексирани во заеднички поток, или во повеќе одделни потоци кои се по потреба синхронизирани меѓусебе, [27]. Еден од најраспространетите типови на мултимедиски содржина се MPEG (Moving Pictures Experts Group) стандардите. Во продолжение ќе видиме како се врши мултиплексирањето/демултиплексирањето во најраспространетите MPEG стандарди денес, т.е. MPEG-1, MPEG-2 и MPEG-4.

6.4.1 Мултиплексирање во MPEG-1

Стандардот MPEG-1 е креиран со цел кодирање на видео со придружено аудио за брзини до 1,5 Mbit/s. Главна цел при креирањето на овој стандард е запис на видео и аудио информации за пристап до нив локално на компјутерот. Ваков тип на апликации се снимени видео информации, интерактивни CD записи и игри. Притоа, стандардот е оптимизиран за одредена резолуција SIF (Standard Image Format), 352 × 240 за фреквенција на рамките 30 Hz (овој формат е наменет за ТВ согласно стандардот NTSC

при 60 полурамки/сек.), односно 352 × 288 за 25 Hz (формат наменет за стандардите за телевизија PAL и SECAM, при 50 полурамки/сек.) за т.н. луминентни компоненти. Резолуцијата за колор-диферентните компоненти е двапати помала по димензии.

Во стандардот се користат три типа на слики (рамки), I, P и B. Луминантните компоненти на секој макроблок се 16 × 16 пиксели, додека колор диферентните компоненти во еден макроблок се со димензии 8 × 8. Трите типа на слики се групираат во групи на слики (GOP – Group of Picture), кој е дефиниран со два параметри: бројот на P слики меѓу две последователни I слики и бројот на B слики меѓу две последователни I или P слики.

Аудио делот на MPEG-1 овозможува семплирање со фреквенции до 48kHz, и 4 типа на работа: моно, стерео, независни два излези и споено стерео.

Во стандардот MPEG-1 е дефиниран системскиот дел за мултиплексирање на податоците од различни потоци во единствен поток на излез од кодерот. Елементарните потоци се референцираат како елементарни потоци (elementary streams). За синхронизација на потоците се користат временски ознаки (time stamps). Овие временски ознаки се користат од страна на декодерот за декодирање и прикажување на видео и аудио информациите.

Синхронизација

Овде накратко ќе се задржиме на синхронизацијата кај MPEG кодираните сигнали. MPEG обезбедува механизам за синхронизација меѓу видеото и придруженото аудио. Притоа се користат два параметри: референца на системскиот часовник (System Clock Reference – SCR) и временски ознаки за прикажување (Presentation TimeStamp – PTS). Системскиот часовник специфициран со MPEG работи на 90 kHz. SCR и PTS се кодираат со 33 бити со што може да се претстави било кој временски циклус во тек на 24 часа ($2^{33} = 8589934592 > 60 \times 60 \times 24 \times 90000 = 7776000000 > 2^{32} = 4294967296$).

SCR е копија во даден момент од системскиот часовник на кодерот кој се сместува во потокот бити во системското ниво. При декодирањето, овие вредности се користат за освежување на системскиот часовник.

PTS се отсечоци од системскиот часовник на кодерот, а се поврзани со делови од кодиран видео или аудио сигнал. PTS го специфицираат времето релативно во однос на

системскиот часовник кога треба да биде прикажана видео слика или пак за аудио, кога треба да започне аудио секвенца.

Декодерот може да прескокне или пак да повтори одредена слика при прикажување на видео секвенца, зависно од тоа дали PTS во рамките на сликата е во рамките на поместување помало од (1/90000) секунди во однос на SCR. Доколку PTS е порано од моменталната вредност на SCR при прикажување на сликата, тогаш декодерот ја отфрла сликата. Обратно, ако PTS е подоцна од SCR (има поголема вредност), тогаш декодерот го повторува прикажувањето на сликата.

6.4.2 Мултиплексирање во MPEG-2

По стандардот MPEG-1 се појавила потреба и за компресија на ТВ сигналите за поефикасна дистрибуција на истите во кабелските мрежи и сателитските системи. Подоцна со стандардот била опфатена и телевизијата со висока резолуција (HDTV), која претходно била планирана да биде во посебен стандард MPEG-3.

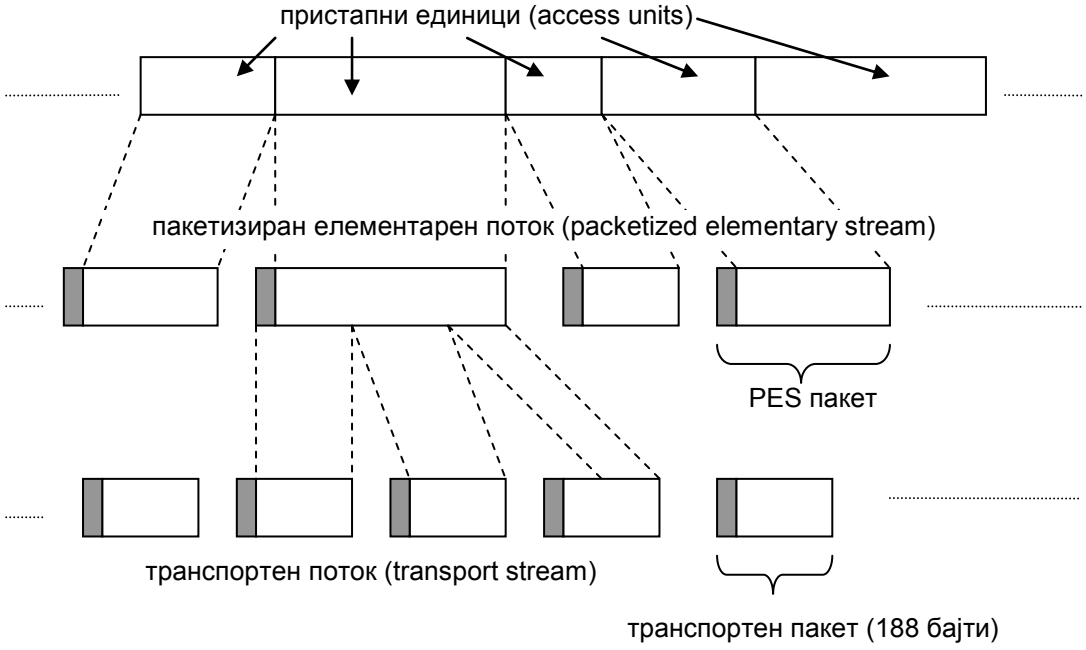
MPEG-2 внесува повеќе новитети во однос на неговиот претходник MPEG-1. MPEG-2 овозможува пренос и во средини со одредена веројатност на загуби на пакетите или битски грешки што не е случај со MPEG-1. Во MPEG-2 се внесуваат повеќе резолуции на слики кои се поддржуваат, како и повеќе формати на слики. Се појавува и скалестото кодирање за првпат во серијата на MPEG стандарди. Во MPEG-2 се воведува интерактивност во мултимедија сервисите, како што е интерактивна телевизија, функции на далечинска контрола (VCR functions), како и паралелен пренос на повеќе видео или аудио потоци. Се воведува и дефинира транспортен поток.

Имено, MPEG-2 има две шеми за мултиплексирање, и тоа:

1 **програмски поток (PS – Program stream)**, кој овозможува синхронизација и мултиплексирање на повеќе видео и аудио потоци. Програмскиот поток е компатибилен со MPEG-1 стандардот и е наменет за средини без грешки при преносот (наменет е за податоци кои локално се складираат на компјутерот). Пакетите се долги и имаат променлива должина, а со цел да се намали учеството на заглавието во потокот. Разликата на програмскиот поток на MPEG-2 во однос на MPEG-1 е постоењето на сигнализација во овој стандард;

2 **транспортен поток (transport stream)**, ги комбинира еден или повеќе

програмски потоци во еден поток. Притоа, програмите може, но не мора да имаат иста временска база (на пример, при дистрибуција на ТВ треба да има синхронизација меѓу елементарните потоци, што не е случај со пренос на податоци како што се игрите и сл.). Овој поток се употребува во средини со поголема веројатност на грешка. Притоа, транспортниот поток не обезбедува техники за заштита од грешки при преносот, туку тоа го препушта на мрежните нивоа под него. Тој е всушност влез во транспортно ниво согласно дефинициите од OSI референтниот модел со 7 нивоа, а не е вистински транспортно ниво.



Слика 6.10 Пакување на MPEG-2 компресираните податоци во програмскиот и транспортниот поток на MPEG-2

И за двете нивоа на мултиплексирање постои заедничка структура на податоците, а тоа се т.н. елементарни програмски потоци (PES – Program Elementary Stream). Тоа е првото ниво при мултиплексирањето, а второто е зависно од комуникацискиот медиум. Притоа, програмскиот поток се генерира преку едноставно поврзување на PES пакетите со неопходните податоци за да се создаде единствен поток. Понатаму транспортниот поток се создава со пакување на програмскиот поток во пакети со фиксна големина од 188 бајти (TS – Transport Packet), како што е прикажано на слика 6.10. Големината од 188 бајти произлегла од тежнеењето MPEG-2 да се користи за пренос преку ATM мрежи со користење на CBR сервисите за кои што е дефинирано ATM адаптациското

ниво 1 (ATM Adaptation Layer 1 – AAL1). AAL1, согласно спецификацијата од ITU-T, има должина на податочниот дел од 47 бајти, а 4*47=188 бајти, колку што е големината на транспортниот пакет во MPEG-2.

Во MPEG-2 се користат два типа на временски ознаки, и тоа: 1) референтни временски ознаки, односно SCR во програмскиот поток, а PCR во транспортниот поток, и 2) временски ознаки за моментите на декодирање и прикажување на видео и аудио рамките, DTS (Decoding Time Stamp) и PTS (Presentation Time Stamp) соодветно. Временските ознаки се користат кога има потреба за синхронизација (на пример, при дистрибуција на дигитална телевизија), но не е задолжително да се користат во MPEG-2 видео потоците (на пример, нема потреба од временски ознаки при симнување на MPEG-2 фајлови од мрежа).

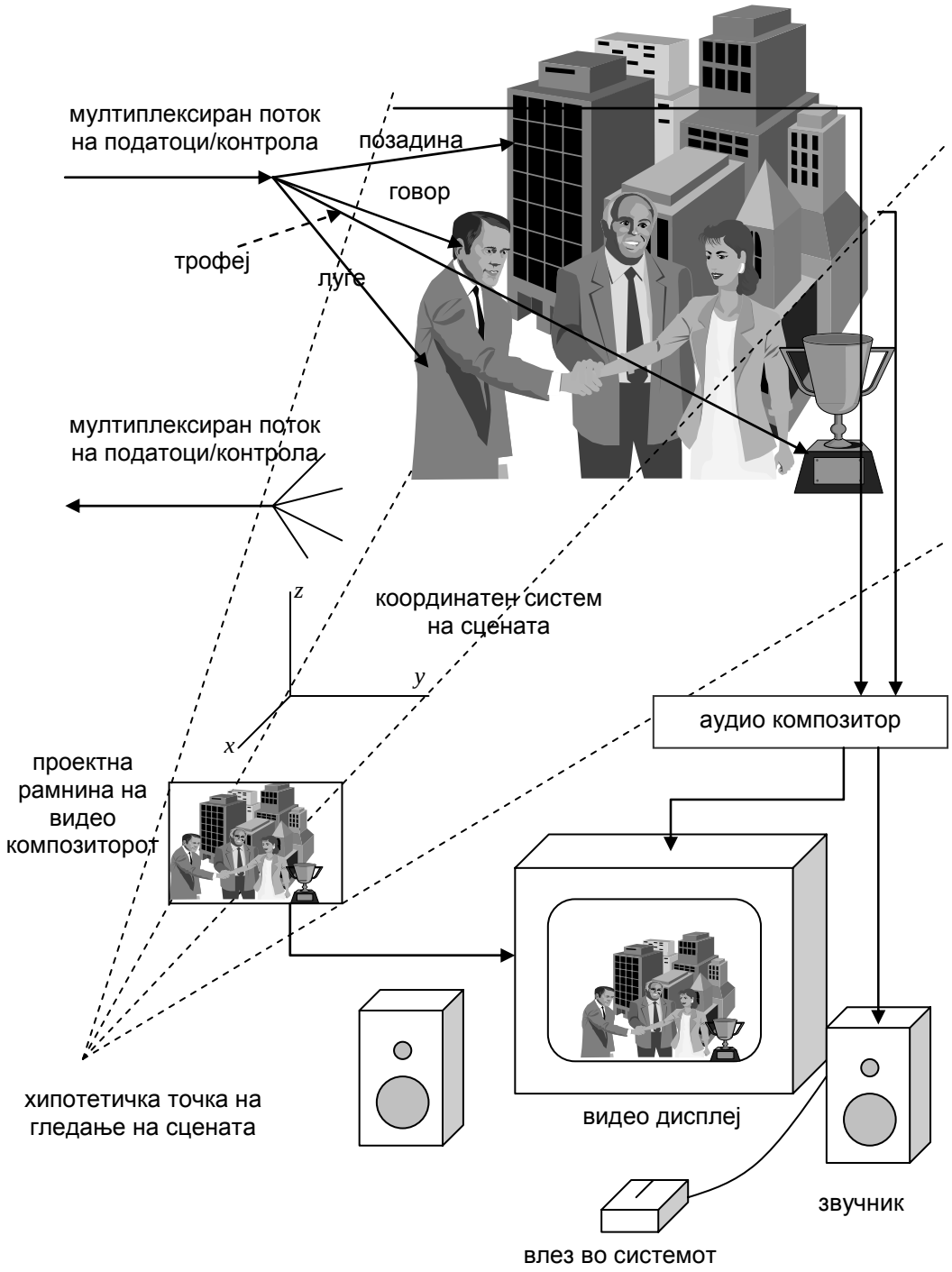
6.4.3 Мултиплексирање во MPEG-4

MPEG-4 стандардот е направен врз база на три успешни подрачја во светот на комуникациите и компјутерските системи: дигиталната телевизија, интерактивната графика и WWW (World Wide Web). Целта на MPEG-4 е да овозможи интеграција на овие три полиња во однос на продукцијата, дистрибуцијата и пристапот до содржината.

Дефиниции на основните поими во MPEG-4 стандардот

Бидејќи една од основните карактеристики на MPEG-4 како стандард е интеракцијата со содржината на одредена сцена во однос на манипулација на објектите кои се наоѓаат на неа, дефинирани се AVO (Audio Visual Object). Под AVO се подразбира аудио визуелен објект од одредена сцена до кој може да се пристапи. Тоа е основниот чекор надвор од стандардното кодирање кое не овозможуваше интерактивност во однос на содржина на една сцена. Генерално, еден AVO може да има:

- само компонента видео објект (VOC);
- само компонента аудио објект (AOC);
- двете компоненти.



Слика 6.11 MPEG-4 аудиовизуелна сцена

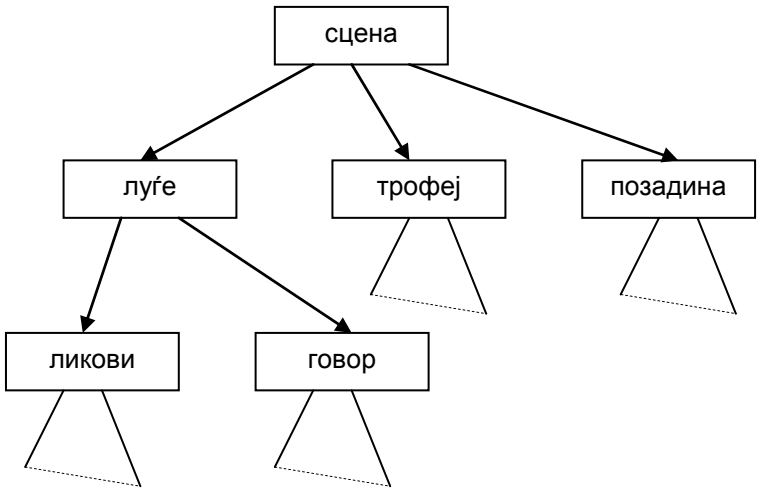
Аудио-визуелните сцени се добиваат со комбинирање на повеќе AV објекти, организирани во хиерархиски редослед. На завршетоците од една таква разгранета хиерархија се наоѓаат примитивни AVO, како што се слика на лице кое зборува, гласот придружен на тоа лице и слично. Аудио компонентите можат да бидат моно, стерео

или повеќеканални. Видео објектите можат да бидат 2D или 3D, природни или вештачки генерирани, моно или виртуелни. Пример на една аудио-визуелна сцена составена од повеќе AVO е дадена на Слика 6.11.

Повеќе примитивни аудиовизуелни објекти можат да се комбинираат за да се создадат нови посложени аудиовизуелни објекти. Стандардот MPEG-4 овозможува да се сместуваат аудио-визуелните објекти било каде во даден координатен систем и да се групираат примитивните AVO за да се добијат сложени AVO. Исто така, може да се промени интерактивно точката на гледање или слушање било каде на сцената, да се отстранат или преместат одредени објекти, да се вметнат нови објекти на сцената.

Заради овозможувањето на манипулација со содржината на сцената кај MPEG-4, податочната структура се разликува од претходните два стандарди, MPEG-1 и MPEG-2.

За да се овозможи индивидуален пристап до секој објект на сцената, потребно е сцената да биде претставена како композиција од различните објекти кои на приемната страна се компонираат според "сценариото" за да се реконструира сцената. Тука можеме да го дефинираме поимот виртуелна рамнина на објект VOP (VOP – Virtuel Object Plane). VOP претставува дводимензионален VOC со произволен облик (на пример, трофејот на Слика 6.11 е еден VOP, позадината со згради е друг итн.).



Слика 6.12 Опис на видео сцена

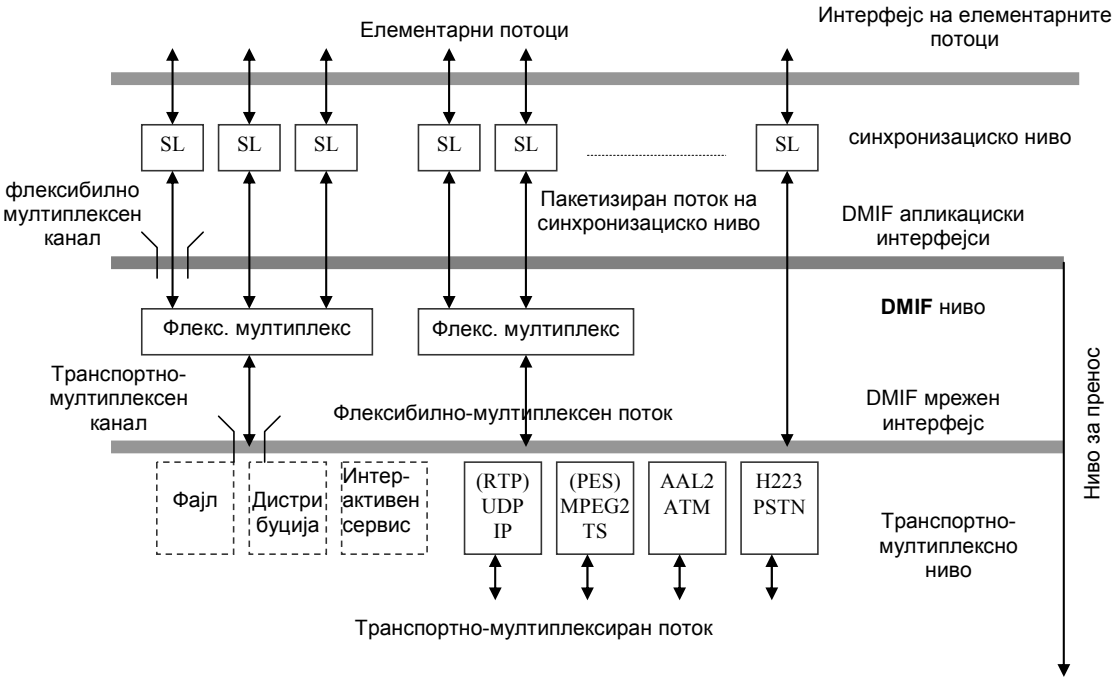
За да може да се реконструира сцената на приемната страна постои опис на сцена (scene description), кој му е придружен на аудио-визуелниот објект, се кодира и се испраќа од кодерот истовремено со AVO. Описите на сцените се кодираат независно од

потокот (stream) на AVO, а со цел да се обезбеди полесна авторизација во однос на правото за пристап или манипулација, полесна интерактивност и манипулација со параметрите на објектите. Параметрите на аудио-визуелните објекти кои можат да се менуваат се содржат во описот на сцената, со што се овозможува тие да се менуваат без да се декодира аудио-визуелниот објект. На Слика 6.12 е даден шематски приказ на описот на сцената од Слика 6.11.

Мултиплексирање на потоци во MPEG-4

Кодираните аудиовизуелни објекти се сместуваат во еден или повеќе елементарни потоци (elementary streams). Секој од овие потоци е дефиниран преку параметрите за одредување на QoS (Quality of Service) кој треба да биде обезбеден при преносот (на пример: максимална битска брзина, веројатност на грешка при преносот итн.).

Системскиот модел за MPEG-4 кодираните податоци е прикажан на Слика 6.13. Во основа може да се подели на нивоа и интерфејси кои се ориентирани повеќе кон апликацијата и нивоа и интерфејси ориентирани кон преносниот систем во подолните нивоа. Генерално, разликуваме 3 нивоа.



Слика 6.13 Системски модел на MPEG-4

Нивото за пристап (AccessUnit Layer – AL) или **синхронизациското ниво (Synchronization Layer – SL)** овозможува идентификација на различните единици до кои може да пристапи, како на пример видео или аудио секвенци, опис на сцена итн., потоа има задача обновување на нарушена временска база на описот на сцената и синхронизација.

Синхронизациското ниво го дефинира начинот (синтаксата) за пакување на елементарните потоци во пристапни единици (access units). Секвенца од SL пакети создава SL-пакетски поток (SL-packetized stream – SPS). Пристапните единици се основни единици за синхронизација.

Еден SL пакет е составен од заглавие на пакетот и податоци (payload). Заглавието на SL пакетот овозможува континуирана проверка во случај на загуби и ги содржи во кодирана форма временските ознаки (timestamps) и придружената информација. Меѓутоа, SL пакетот не содржи информација за својата должина, па затоа има потреба од соодветен протокол од пониско системско ниво кој ќе ги внесе SL пакетите во рамки (frames). Тоа ниво е нивото за флексибилно мултиплексирање (FlexMux Layer).

Синхронизациските (SL) пакети не ја носат информацијата за идентитетот на елементарниот поток во своето заглавие. Ова пресликување меѓу елементарните потоци и синхронизациските пакети се обезбедува со табели на мапирање на потоците кои се пренесуваат со користење на соодветна сигнализација во зависност од транспортниот механизам (ATM, IP итн.).

Интерфејсот на елементарните потоци кон SL (Elementary Stream Interface – ESI) е интерфејс кој специфицира кои податоци треба да се разменуваат меѓу нивото за кодирање и синхронизациското ниво под него. Комуникацијата меѓу овие две нивоа не се заснова само на пренос на компресирани информации (аудио, видео), туку и дополнителни информации како што се временските ознаки, должината на пристапните единици итн.

Интерфејсот на поточен мултиплекс (Stream Multiplex Interface – SMI) е интерфејс меѓу синхронизациското ниво и транспортниот механизам (Слика 6.13). Овој интерфејс не ги вклучува само SL-пакетските податоци, туку и дополнителни информации во однос на должината на секој SL пакет.

Флексибилен мултиплекс (FlexMux Layer) е мултиплексно ниво, целосно специфицирано со MPEG-4 стандардот како и сите нивоа над него. Ова ниво содржи

алатка за мултиплексирање која овозможува групирање на елементарните потоци со додавање на дополнително мало заглавие (overhead). Управувањето со сесиите (отворањето на транспортните потоци) е задача на т.н. DMIF (Delivery Multimedia Integration Framework). Ова значи дека интерфејсот меѓу DMIF и системот се наоѓа меѓу двете мултиплексни нивоа на MPEG-4. Оттаму, флексибилно мултиплексираното ниво се референцира и како DMIF ниво. Нивото за флексибилно мултиплексирање е опционално и може да биде избегнато доколку подолното транспортно ниво еквивалентно ги обезбеди неговите функции.

Транспортен мултиплекс (Transport Multiplex Layer) е првото мултиплексно ниво кое треба да понуди транспортни сервиси соодветни на барањата за QoS. За да се овозможи максимална флексибилност при креирање на сервисите и апликациите, ова ниво не е специфицирано во MPEG-4 стандардот. Само интерфејсот меѓу транспортниот мултиплекс и погорното ниво е специфициран со MPEG-4. Во ова ниво може да се употреби било која хиерархија на транспортни нивоа која што е погодна: (RTP)/UDP/IP, (AAL5)/ATM или MPEG-2 транспортното ниво преку соодветно физичко ниво. Изборот им е препуштен на производителите на опремата и крајните корисници.

Мултиплексирање на елементарните потоци

Во овој дел ќе го разгледаме флексибилното мултиплексирање на пакетизираните потоци кои доаѓаат од синхронизациското ниво SL.

Во нивото на флексибилниот мултиплекс се користи алатка која е флексибилен мултиплексер што врши разместување (interleaving) на SL-пакетските потоци со моментални промени во битската брзина. Основна податочна единка на ова ниво е флексибилно мултиплексираниот пакет (FlexMux packet), кој има променлива должина. Тука MPEG-4 стандардот се разликува од MPEG-2 (претходниот MPEG стандард) каде што постоеше во стандардот специфицирано транспортно ниво (кое овде не е специфицирано и е оставено да се избере некое од постоечките кои одговараат, како што се IP, ATM итн.).

За да се разликуваат синхронизацискиите пакети кои потекнуваат од различни елементарни потоци, алатката за флексибилно мултиплексирање доделува тн. број на флексибилен мултиплексен канал (FlexMux Channel). Секој SL-пакетски поток е

пресликан во еден флексибилен мултиплексен канал. Со тоа, може без грижа да се разместуваат флексибилно мултиплексираните пакети од различни SL потоци. Секвенца од вака разместени FlexMux пакети кои се наоѓаат во еден поток се вика флексибилно мултиплексиран поток (FlexMux Stream).

Нивото за флексибилно мултиплексирање не обезбедува случаен пристап (random access) или санација на грешки (error recovery), што е препуштено на подолните протоколи кои треба за таа цел да ги сместуваат флексибилно мултиплексираните пакети во рамки. Исто така, од подолните транспортни нивоа се бара да имаат и доволно добра детекција на грешка.

6.5 Дискусија

Мултимедиските комуникации преку Интернет се првиот тип на комуникации во реално време преку Интернет што ги обработуваме во оваа книга. Кога имаме пренос на мултимедија во реално време, тоа најчесто претставува стриминг на видео со придружено аудио, при што може да има и повеќе видео или аудио потоци во рамките на една мултимедиска конекција. При пренос на мултимедиски информации во реално време, потребно е да има синхронизација меѓу видео и аудио потоците, како и повратна врска меѓу примачот (клиентот) и праќачот (серверот) на мултимедиска содржина за прилагодување на условите на патеката од крај до крај меѓу клиентот и серверот (на пример, прилагодување со резолуцијата на сликата, бројот на потоци и слично кон расположливиот битски проток кој е променлива големина). За таа намена, за поддршка на сервисите во реално време стандардизиран е RTP (Real-time Transport Protocol), кој го опфативме во детали во оваа глава. Самиот протокол се користи врз UDP протоколот на транспортно ниво, како апликација/RTP/UDP/IP протоколен стек. Ваквиот протоколен стек се користи за пренос на корисничките податоци (од видео, аудио итн.), додека сигнализацијата за воспоставување на мултимедиска комуникација најчесто се реализира преку HTTP (кога имаме видео стриминг преку веб страници) или со користење на SIP (Session Initiation Protocol), кој е опфатен во глава 7 во оваа книга.

За креирање на мултимедиска содржина најчесто се користат MPEG стандардите, од кои што денес најмногу раширен во Интернет мрежата за видео мултимедиските содржини е MPEG-4 кој постојано се надополнува со амандмани

(parts) на ISO стандардизациското тело, од 1999 година до денес. Така, MPEG-4 е модуларен стандард кој е во постојан развој, при што речиси не постои имплементација која ги вклучува сите негови можности, односно тоа е оставено на индивидуалните развивачи на кодеци и мултимедиски апликации. Така, на MPEG-4 се базирани DivX, Xvid, видеото со висока резолуција како Blu-ray дисковите, AVC/H.264, Adobe Flash видео кодеците, итн.

Глава 7

Говор преку Интернет

Интернет телефонијата, позната и како VoIP (Voice over IP) или IP телефонија, е испорака на говор во реално време (и по можност на други мултимедиски податочни типови) меѓу два или повеќе ентитети преку мрежа користејќи Интернет протоколи. Интернет телефонијата нуди можност да се дизајнира глобален мултимедиски комуникациски систем кој би ја заменил постоечката телефонска инфраструктура. Меѓутоа, таа мора да ги понуди и стандардните телефонски сервиси. Во секој случај, преминот кон Интернет - базираните телефонски сервиси нуди можност за креирање на нови сервиси кои се пофлексибилни и се со помала комплексност отколку кај постоечката PSTN (Public Switsched Telephone Network) мрежа, [28].

Иако ќе го користиме терминот "Интернет телефонија", сепак треба да се нагласи дека во Интернет средина моќно е и комбинирање на телефонијата со друг тип на медија, како што се видео или деливи (shared) апликации. На пример, може да постои желба за заедничко следење на некои филмови или следење на директен пренос на некој настан преку Интернет (спортски натпревар, модна ревија, итн.), при што пријатели поврзани на различни локации може да даваат разни коментари и истовремено да водат разговори.

За разлика од традиционалните телекомуникациски мрежи каде што дистрибутивните сервиси (радио, ТВ) и традиционалните комуникациските сервиси (телефон, факс) целосно се опишани (во стандарди) во однос на технологијата, корисничките интерфејси, комуникациските уреди итн., ова не е случај со Интернет мрежата каде што има можност за поголема иновативност.

Интернет телефонијата се разликува од Интернет - мултимедискиот стриминг во контролата и потврдата на сесијата, т.е. сигнализацијата. Персоналната мобилност, достапноста и терминалните уреди за комуницирање (преку Интернет) влијаат на комплексноста на процесот на сигнализација.

Иако самиот поим Интернет телефонија се поврзува главно со пренос на говор меѓу два корисника (во двете насоки истовремено), ниеден од протоколите кои ќе бидат опишани овде не е ограничени само на еден тип на медија (на пример, можно е комбинирање на говорот со видео, како и со пренос на текст и други податоци или фајлови) или само на уникаст пренос (на пример, можни се конференциски врски). Всушност и најголемата предност на Интернет телефонијата во однос на традиционалните телекомуникациски мрежи, т.н. POTS (Plain Old Telephone System), е транспарентноста на мрежата кон сервисите и различните типови на информации, така што додавањето на нов сервис или промена на даден сервис не иницира промени во мрежната инфраструктура (иако во дадени случаи тоа е потребно, посебно кога е неопходно да се обезбеди одреден гарантиран квалитет за даден тип на сервис).

Генерално, IP мрежите се податочни мрежи чија основна карактеристика е динамичко користење на ресурсите со користење на best-effort принципот. Таквата карактеристика е погодна за пренос на податоци, меѓутоа заради сообраќајната избувливост која е одлика на податочните мрежи би се појавиле сериозни проблеми при пренос на говор кои би се манифестирале како променливи доцнења од крај до крај, појава на ехо, џитер (т.е. варијација во доцнењето на пакетите) итн.

7.1 Разлики меѓу Интернет Телефонијата и PSTN

Интернет телефонијата и обичната телефонија се разликуваат, [29]. Интернет телефонијата го претвора говорот во пакети со говорни информации и ги пушта преку релативно поедноставна мрежа - Интернет. Од друга страна, IP пакетите ги примаат интелигентни уреди како компјутери и IP телефони. Традиционалните телефонските оператори пак го пренесуваат говорот кон/од корисникот како аналоген сигнал низ систем од жици и кабли кои се поврзани со интелигентни централни компјутери наречени комутатори (централи). Таму говорот се дигитализира (се претвора од аналогна во дигитална форма и обратно, со исклучок на фиксните ISDN телефонски

линии и кај мобилните системи каде што дигитализацијата се врши на страната на корисникот и неговата терминална опрема или телефон) и се препраќа преку други комутатори се додека не пристигне повикот до бараната дестинација (според телефонскиот број што го завртел повикувачот). Класичните телефонски апарати се аналогни телефони кои се многу поедноставни од компјутер или IP телефон, а со самото тоа се и поевтини. Правејќи споредба меѓу PSTN и Интернет телефонијата можеме да издвоиме повеќе предности на едната или на другата страна.

Така, на пример, основните предности на PSTN мрежата се:

- константен и гарантиран битски проток, т.е. дедицирани канали (G.711 PCM: 8kHz*8bit = 64 kbit/s), што пак од друга страна може да се согледува и како недостаток заради отсуство на статистичко мултиплексирање на говорот од повеќе корисници на даден линк;
- малата латентност (т.е. доцнење);
- отсуството на џитер (варијација на доцнењето);
- отсуство на загуби на говорни податоци.

Од друга страна, недостатоците на PSTN мрежата можат да се сублимираат на следниот начин:

- потребна е голема количина на скапа опрема (на пример, централи);
- постои голема комплексност во изработката и потешкотии во имплементацијата на нови промени во комутаторите, или во воведувањето на нови сервиси со комбинирање на различни типови на медија (видео, податоци);
- ресурсите (по еден 64 kbit/s канал во секоја од двете насоки) се зафатени цело време независно дали има корисна информација т.е. говор или не (нема динамичко користење на ресурсите);
- потребни се различни мрежи за пренос на говор (PSTN), за пренос на податоци, за пренос на ТВ, за пренос на радио итн., за разлика од Интернет каде што истата пристапна, скелетна и транспортна мрежа се користи за пренос на различни сервиси (говор, телевизија, best-effort Интернет сервиси).

IP телефонијата ја олеснува од-крај-до-крај (end-to-end) парадигмата за испорачување на сервиси. Сигнализациските пораки се меѓу крајните системи

инволвирани во повикот при што мрежните рутери ги третираат овие сигнализациски пакети како и било кој друг податок, игнорирајќи ја семантиката што постои во нив.

Ако телефонска компанија сака да обезбеди нов сервис, како на пример идентификација на повик, мора да ги репрограмира комплексните комутатори што не е лесна работа. VoIP (Voice over IP) провајдерите лесно можат да понудат нови услуги поради фактот што крајните уреди се доволно интелигентни (тоа се најчесто компјутери или телефони со целосен оперативен систем на кој работат различни апликации меѓу кои и апликација за VoIP) и можат да се надградат со нов софтвер. Заради тоа VoIP провајдерите постојано нудат нови услуги кои што телефонските провајдери тешко или не можат да ги остварат. Сепак, за да се овозможи IP телефонија со квалитет како што е телефонијата во PSTN, потребно е градење на доверлива сигнализациска мрежа (што ја нема во класичниот best-effort Интернет), а тоа ја зголемува комплексноста на мрежата и нејзината цена и ја доближува до цената на имплементација на PSTN.

Меѓутоа и кај Интернет телефонијата постојат одредени проблеми. Телефонските компании можат да понудат гарантиран QoS (Quality of Service) или GoS (Grade of Service) од 99.999 проценти годишно. Тоа значи дека во просек само 5 минути годишно нема да можат да понудат услуги. За разлика од нив, VoIP провајдерите не можат да понудат ниту тој процент на годишно ниво т.е. статистички гледано. За да се замени класичната телефоинија со VoIP (т.е. Интернет телефонија) потребно е да се обезбедат решенија за поддршка на квалитетот на сервисите во пристапните и транспортните мрежи со користење на решенија за QoS поддршка во Интернет како што се диференцирани сервиси, интегрирани сервиси, и/или MPLS/VPN (Multi-Protocol Label Switching/ Virtual Private Networks), [30].

Понатаму, за разлика од фиксната телефонија каде што напојувањето доаѓа преку корисничката парица од страна на централата (комутаторот) на кој е приклучен корисникот и каде што задолжително има поддршка за непрекинато напојување од барем 24 часа во случај на прекин на напојувањето од електродистрибутивната мрежа, кај VoIP напојувањето (од електричната мрежа или преку батерии) е оставено на страната на корисниците. Но, примерот со користење на батериите во мобилните телефони (што е неопходност) укажува на фактот дека тоа повеќе не е проблем.

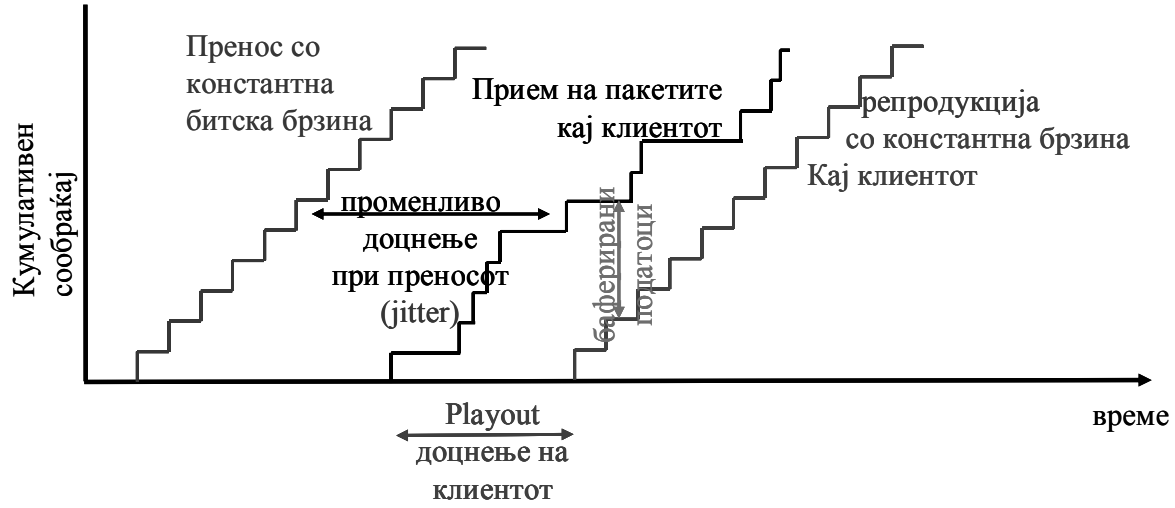
7.2 Квалитет на сервисот кај VoIP

Од аспект на квалитетот на сервисот (Quality of Service - QoS), [30], сервис провајдерот треба да се грижи за неколку работи кога реализира VoIP повик:

- **Латентноста (доцнењето)** се случува кога пакетите се пренесуваат за подолго време заради појава на тесно грло во мрежата (на пример: загушување во мрежните јазли – рутери вдолж патот).
- **Цитерот** е разлика во доцнењата на пакетите. Некои пакети пристигнуваат со помало доцнење, а некои со поголемо доцнење, што зависи од условите во попатните јазли и од капацитетот на линковите по кои се пренесуваат. Притоа, треба да се има во предвид дека во Интернет различни пакети можат да пристигнат од ист извор до иста дестинација по различни патеки, иако треба да се нагласи дека за VoIP понуден од мрежни (телеком) оператори се избегнува тоа.
- **Загуба на пакет** се случува кога пакетот ќе наиде на преполнети бафери поради појава на тесно грло, па мрежата го отфрла, или кога доцнењето на пакетот е преголемо за да биде искористен на приемната страна.

Латентноста и загубата на пакети создаваат чудно чувство на тишина во телефонскиот разговор или пак прават да изгледа како едниот корисник да го прекинува другиот при зборувањето. Доцнењата можат да предизвикаат ехо (појавување на порано кажан дел од говорот после дел од говорот кој бил подоцна кажан) и други звучни ефекти (на пример: промена на бојата на гласот).

Но, за секој потенцијален проблем за VoIP постојат соодветни решенија. Во принцип, на VoIP може да му се додели доволно широк опсег, како на пример во телефонската мрежа на некоја компанија која поседува оптика како скелетна (backbone) мрежа. На VoIP пакетите им се доделува највисока приоритетност што овозможува тие да не се отфрлат ако дојде до појава на тесно грло. Исто така провајдерите, заради сигурност, обично доделуваат повеќе опсег одошто е потребно за VoIP. Индустрискиот термин за ова е предимензионирање (overprovisioning). Последниот проблем се всушност последните неколку километри во пристапната корисничка мрежа, односно врската помеѓу провајдерот и корисникот (локалната јамка, local-loop).



Слика 7.1 Анализа на доцнењето на VoIP пакети

На слика 7.1 е прикажана анализа на доцнењето на пакетите од праќачот (изворот) до примачот (дестинацијата). Приемникот се обидува да го репродуцира секој дел (chunk) точно на интервали од q msec од моментот кога бил генериран тој дел (chunk):

- chunk има временска ознака - timestamp t: ќе се репродуцира во момент t+q
- chunk пристигнува по t+q: податоците пристигнуваат премногу доцна за репродукција, податоците се изгубени (data "lost")

Притоа, имаме мала "трговија" за должината на интервалот q т.е.:

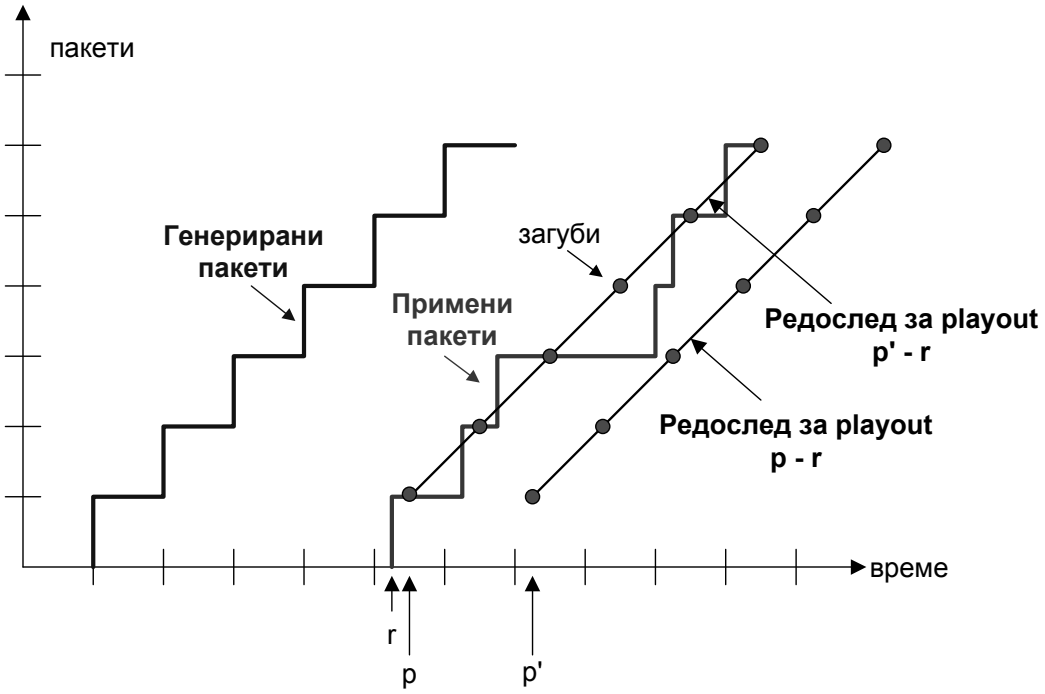
- Долг интервал q: помалку загубени пакети
- Мал интервал q: подобар квалитет, бидејќи ќе има помало вкупно доцнење на пакетите од моментот на генерирање во изворот до моментот на репродукција на приемната страна.

За анализа на загубите на пакети кај VoIP да ја погледнеме слика 7.2:

- Праќачот (sender) генерира пакети на секои 20 msec за време на talk spurt
- Првиот пакет пристигнува во моментот r
- Првата репродукцијата почнува во моментот p
- Втората репродукција: почнува во моментот p'

Од сликата може да се забележи дека загубите на пакети настануваат и тогаш кога пакетот ќе пристигне на приемната страна, но подоцна од времето кога треба да биде искористен за репродукција на говорот. При почеста појава на вакви загуби решението е да се зголеми доцнењето на пакетите (преку подолго баферирање на приемната страна), при што максималното време на доцнење од крај до крај не треба да

биде поголемо од 400 ms, а препорачливо за телефонијата е да биде до 150 ms. Секако, врз вкупниот буџет на доцнењето на пакетите влијаат оптовареноста на линковите (загушувањето во мрежата по пат), битските протоци на линковите по кои се пренесува VoIP сообраќајот, како и процесирањето на VoIP пакетите на предавателната и на приемната страна од конекцијата.



Слика 7.2 Анализа на загубите на пакети кај VoIP

За да се излезе во пресрет на доцнењето на пакетите кај VoIP кое е најкритично во телефонијата (имено, VoIP телефонијата може да трпи загуби на пакети и до 10% без посериозно нарушување на субјективниот квалитет на говорот што го слуша човекот, но доцнењето не смее да биде поголемо од 400 ms за да се има чувството на конверзација во реално време, односно кога зборуваме по телефон да имаме чувство дека корисникот се наоѓа во наша непосредна близина иако може реално да биде многу оддалечена локација, на пример - на друг континент на земјата). За таа цел, за да се оптимизира доцнењето и загубите на пакети кај VoIP постои начин на користење на адаптивно playout време во Интернет телефонот (т.е. во VoIP апликацијата). Притоа, главна цел е минимизирање на доцнењето при playout, со цел помали загуби. Пристап на адаптивно прилагодување на доцнењето при playout е следен:

- Проценка на мрежното доцнење, прилагодување на playout доцнењето на почеток на секој talk spurt

- Компресија на silent периодите
- Деловите и понатаму се репродуцираат на секои 20 msec за време на говорниот период

За да покажеме на кој начин може да се имплементира адаптивното прилагодување на доцнењето, нека ги усвоиме следниве ознаки:

t_i = timestamp за i - от пакет
r_i = пакетот i е примен во применикот
p_i = пакетот i е репродуциран во приемникот
$r_i - t_i$ = доцнење низ мрежата за i - от пакет
d_i = проценка на просечното мрежно доцнење по приемот на i - от пакет

Тогаш, динамична проценка ма просечното доцнење кај приемникот:

$$d_i = (1-u)d_{i-1} + u(r_i - t_i) \tag{7.1}$$

каде што u е фиксна константа (на пример: $u = 0.01$).
Проценката на просечната девијација на доцнењето v_i е дадена со:

$$v_i = (1-u)v_{i-1} + u\,|\,r_i - t_i - d_i\,| \tag{7.2}$$

Проценките d_i и v_i се пресметуваат за секој примен пакет, иако тие се користат само на почетокот на говорниот период (talk spurt). За првиот пакет во говорниот период (talk spurt), playout времето е:

$$p_i = t_i + d_i + Kv_i \tag{7.3}$$

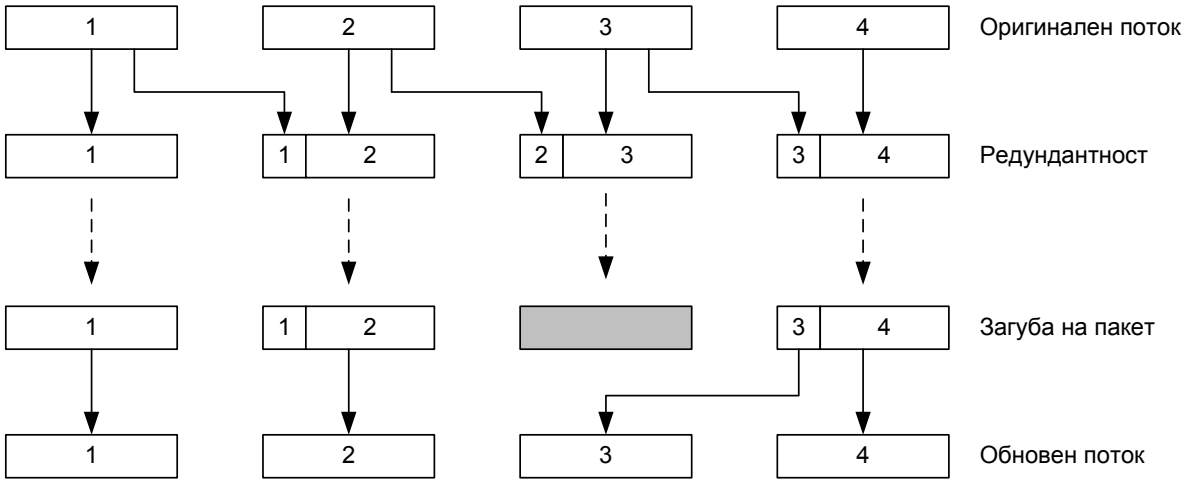
каде што K е позитивна константа. Останатите пакети во говорниот период се репродуцираат периодично.

Како приемникот одредува кој пакет е прв во говорниот период?

- Ако нема загуби, приемникот ги гледа временските ознаки (timestamps):
  - кога разликата меѓу последователните ознаки (stamps) > 20 msec --> почнува talk spurt (говорниот дел).

- Ако има можни загуби, приемникот мора истовремено да ги гледа и временските ознаки (timestamps) и секвенцијалните броеви (sequence numbers):
    - кога разликата меѓу последователните ознаки (stamps) > 20 msec и секвенцијалните броеви се сите на број --> почнува talk spurt.

За заштита од загубите на пакети постојат Forward Error Correction (FEC) методи на заштитно кодирање кои овозможуваат на приемната страна да се детектира и по можност да се корегира грешка при преносот или загубени пакети. Друг начин кој се применува за да се дистрибуира влијанието на загубите на различни пакети (да не биде концентрирано на пократок временски интервал) е методот на мешање (interleaving).



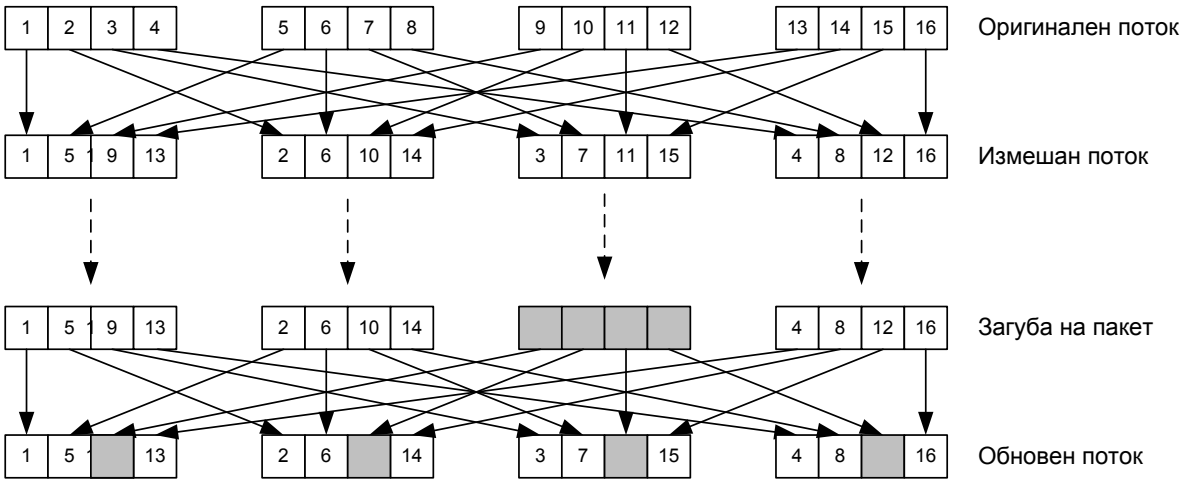
Слика 7.3 FEC заштита за VoIP

Еден Forward Error Correction (FEC) пример со едноставна шема е следниот:

- За секоја група на n делови (chunks) се креира редундантен chunk со вршење на ексклузивно или врз n-те оригинални делови (chunks)
- Се праќаат n+1 делови, со што се зголемува потребниот проток за 1/n.
- Може да се реконструираат оригиналните n делови ако е загубен максимум еден од n+1 делови (chunks)
- Playout доцнењето треба да биде фиксирано на такво време така што да може да се примат сите n+1 пакети
- Трговија: поголем n, помалку опсег се троши, но и подолг playout доцнење, како и поголема веројатност да бидат загубени 2 или повеќе дела од n-те

Друг пример на FEC шема е истовремено праќање на редундантен (идентичен) поток со помал квалитет покрај главниот поток. На пример: покрај 64 kbit/s PCM, се праќа истовремено GSM поток со 13 kbit/s кој е редундантен (ја носи истата информација како потокот од 64 kbit/s), но кој се користи за да се надополнат загубите на информации од 64 kbit/s поток.

Покрај FEC, може да се применува и мешање (interleaving) за да се дистрибуира влијанието на загубите на пакети на поголем број пакети, а со тоа да се намали влијанието на загубите во помалите временски интервали што е почувствително од страна на човекот.



Слика 7.4 Мешање (interleaving) за заштита од загуби кај VoIP

Пример на користење на мешање (interleaving) е прикажан на слика 7.4. Принципот на мешање (interleaving) може да се опише преку следните чекори:

- Деловите (chunks) се разделени во помали единици
- На пример, 4 единици по 5 msec единици по еден дел (chunk)
- Еден пакет содржи помали единици од различни делови (chunks)
- Ако се загуби некој пакет, сепак останува поголемиот дел од секој chunk
- Нема редундантно заглавие, но се додава playout доцнење

Отворената и мултисервисна природа на Интернет мрежата овозможува многу различни компоненти на телефонските сервиси да бидат обезбедени од комплетно различни сервис провајдери. На пример, еден може да обезбеди говорна пошта (voice

mail), друг може да обезбеди мобилни сервиси, додека некој друг веќе обезбедил конференциски сервиси. При тоа, единствен услов кој мора да се задоволи е компатибилност на протоколите. Уште повеќе, од крај до крај (end-to-end) природата на Интернет значи дека секој со Интернет конекција може да оперира како сервис. Ова води кон мошне конкурентен телекомуникациски пазар за сите Интернет сервиси, [31].

7.3 Карактеристики на Интернет Телефонија

Архитектурните разлики кои беа опишани во претходниот дел водат кон голем број предности на Интернет телефонијата, а не само кон "евтин телефонски повик".

Прилагоден квалитет: Иако на почетокот Интернет (или говорот преку IP) телефонијата имаше лоша репутација за нејзиниот квалитетот (заради немањето експлицитна гаранција за квалитетот на сервисот од крај до крај и стандардизирана сигнализација), не постои ниедна причина (освен недостаток на обезбеден квалитет на сервисот, како што се гарантиран битски проток од крај до крај и ограничено доцнење не поголемо од 200-300 msec од крај до крај) поради која истата технологија не би можела да обезбеди квалитетен аудио сигнал или музика. Бидејќи Интернет не е дедицирана мрежа за еден тип на сервис, реално е да се очекува дека телефонскиот сообраќај ќе ги дели истите линкови со други типови на сообраќај (на пример, со веб, со e-mail, со видео сообраќај, итн.) и следствено се неопходни решенија за соодветно третирање на потребите од обезбеден квалитет на сервисот за Интернет телефонијата од крај до крај (меѓу било кои два или повеќе корисници кои комуницираат).

Сигурност: SIP може да врши енкрипција и автентикација на сигнализациски пораки; RTP поддржува енкрипција на информациите што ги пренесува, следствено тоа важи и во случаите кога RTP се користи за пренос на говор преку IP. Според тоа, SIP за сигнализацијата и RTP за говорните податоци обезбедуваат доверлива и сигурна комуникација.

Идентификација на корисник: Стандардните POTS и ISDN телефонски сервиси нудат ефикасна идентификација на повикувачот, но во тек на конференција со повеќе учесници не постои можност за индикација со кого се разговара. RTP (Real Time Protocol) протоколот кој се користи за Интернет телефонија лесно подржува индикација

на повикувачот и во мултикаст и во уникаст конфигурација, а може да пренесе и подетални информации ако посака повикувачот.

Кориснички интерфејс: Многу POTS и ISDN телефони имаат прилично ограничен кориснички интерфејс. Напредните PSTN карактеристики, како што е препраќање на повик, ретко се користат. Ова влијае на лимитирање на сигнализациски способности на крајните системи и на поимот "мрежна интелигенција" во однос на интелигенцијата на крајните системи. Иако крајните системи кај IP телефонијата имаат многу побогати сигнализациски способности, графичкиот кориснички интерфејс понуден со Интернет телефонијата е ориентиран кон корисникот и нуди индикации за нови карактеристики, процеси и прогреси.

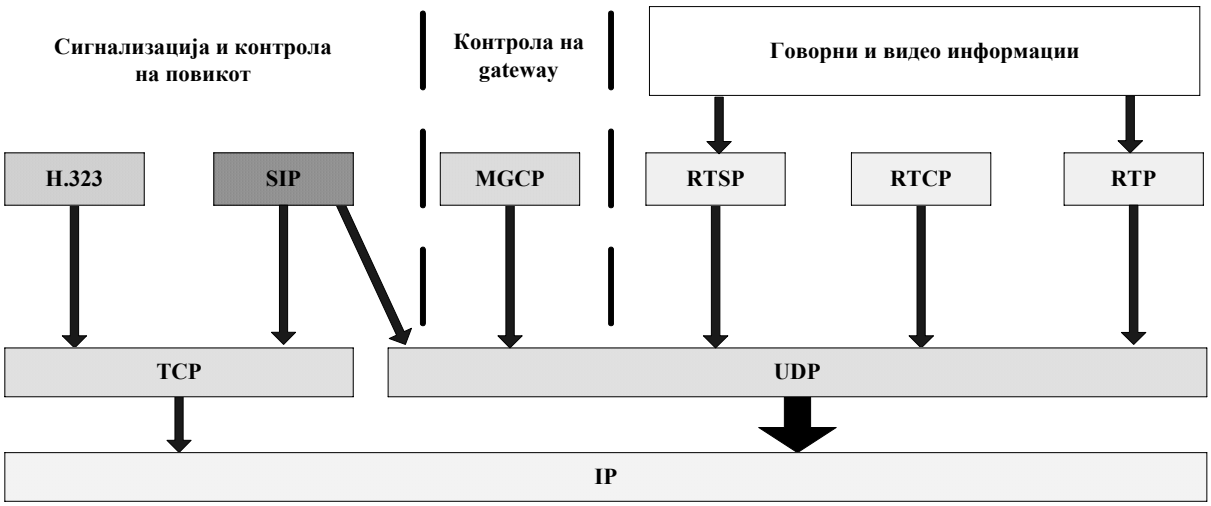
Интеграција компјутер - телефон: Поради потполната разделеност на податочните и контролните патеки и разделбата на телефонската опрема од компјутерите кои ја контролираат, интеграцијата меѓу компјутерите и телефоните е многу комплексна. Многу од манипулативните (оперативни) функционалности на повикот можат еднаш да се извршат при премин на податочната и контролната патека низ интелигентни, мрежно-поврзани крајни системи.

Мултимедија: Додавањето на различни медиски типови, како што се видео или делење на апликации (sharing), е многу полесно во Интернет околина отколку во POTS и ISDN.

Компресија и спречување на тишината: Со праќањето на аудио податоците како пакети лесно се спречуваат периодите на тишина и прилично се намалува трошењето на опсегот. За разлика од PSTN, кај која спречувањето на тишината се врши на трансконтиненталните линкови, кај IP телефонијата спречувањето на тишината се врши во крајните точки. Ова води кон намалување на цената за спречување на тишината. Компресијата може да се искористи на крајните системи за намалување на потрошувачката на опсег. Постојат кодери кои можат да компресираат широкопојасен (wideband) говор до 16 kbit/s и кои даваат одличен квалитет на говорот и истовремено редуциран битски опсег во споредба со PSTN.

Напредни сервиси: Веднаш по првото искуство со протоколите, се појавило мислењето дека е многу полесно да се развијат и инсталираат напредните телефонски сервиси во пакетски комутирана околина отколку во PSTN. Интернет протоколите, како што е SIP, кои подржуваат стандардни CLASS (Custom Local Area Signaling Services) карактеристики може да извршуваат функции и на user-to-network сигнализациските протоколи како што е Q.931 (стандард на ITU), и на протоколите

наменети за мрежна сигнализација (на пример: ISDN User Part - ISUP од Signaling System 7 - SS7).



Слика 7.5 Протоколен стек на Интернет телефонија

7.4 Протоколи за Интернет телефонија

Во Интернет, RTSP (Real Time Streaming Protocol) е стандарден протокол за контрола на медиските потоци, а SIP (Session Initiation Protocol) е протокол кој се користи за сигнализација кај Интернет телефонските сервиси. И RTSP и SIP се дел од протоколен стек прикажан на слика 7.5. Овој протоколен стек ги вклучува VoIP сервисите и сервисите за складирање на медија во еден интегрален дел. За разлика од обичната телефонијата базирана на комутирање на кола, Интернет телефонските сервиси се изградени во домен на пакетско-комутирачки протоколи. Имено, функционалноста на телефонските сигнализациски протоколи SS7 ISUP и TCAP вклучуваат рутирање, резервирање на ресурси, пристап на повик, потврда на повик, транслација на адреса, управување со повик и наплата на повик. Кај Интернет, рутирањето меѓу автономните системи, се реализира со BGP (Border Gateway Protocol), а резервирањето на ресурсите по конекција може да се реализира со RSVP (Resource Reservation Protocol) или друг протокол наменет за резервирање на ресурси, [30]. SIP протоколот, кој подоцна ќе биде опишан, е сигнализациски протокол кој креира, модифицира и терминира конекции меѓу крајните Интернет системи, вклучувајќи конференциски и точка - точка (point-to-point) повици. Не постои стандардизиран протокол за наплата на услугите на Интернет телефонијата (што е случај и во PSTN

претходно). Но, постојат протоколи како што се RADIUS или DIAMETER (види глава 9 од оваа книга за RADIUS), кои во комбинација со SIP автентикација можат да послужат за таа цел за VoIP повиците, преку користење централизирани портни јазли (gateway).

Примарната употреба на UDP (User Datagram Protocol) протоколот е за пренос на примероците од говорот (говорниот сообраќај), а во себе содржи можност за идентификација на пакетите што доаѓаат, но и на тие што треба да се испратат. Овој протокол не е доволно комплетен за да ги поддржи самостојно говорните комуникации. Тековен стандарден протокол на транспортно ниво за пренос на говор е RTP (Real-time Transport Protocol), [24]. RTP претставува подобрување на UDP и се користи во протоколен стек RTP/UDP. Доколку е употребен TCP (Transmission Control Protocol) се овозможува подобра контрола и обнова на протокот, бидејќи TCP е доверлив протокол кој овозможува сите загубени пакети да бидат препратени. Затоа TCP главно се користи за сигнализацијата за VoIP (надвор од опсег), а за пренос на говорните информации се користи UDP. Имено, кај говорот нема потреба од потврди (ACK), како што имаме кај TCP, бидејќи нема потреба од препраќање на загубените пакети заради осетливоста на говорот кон доцнењето и џитерот (препраќањето на пакетите на говорот би довело до дополнително доцнење и џитер кое е неприфатливо за конверзациски сервис како VoIP сервисот).

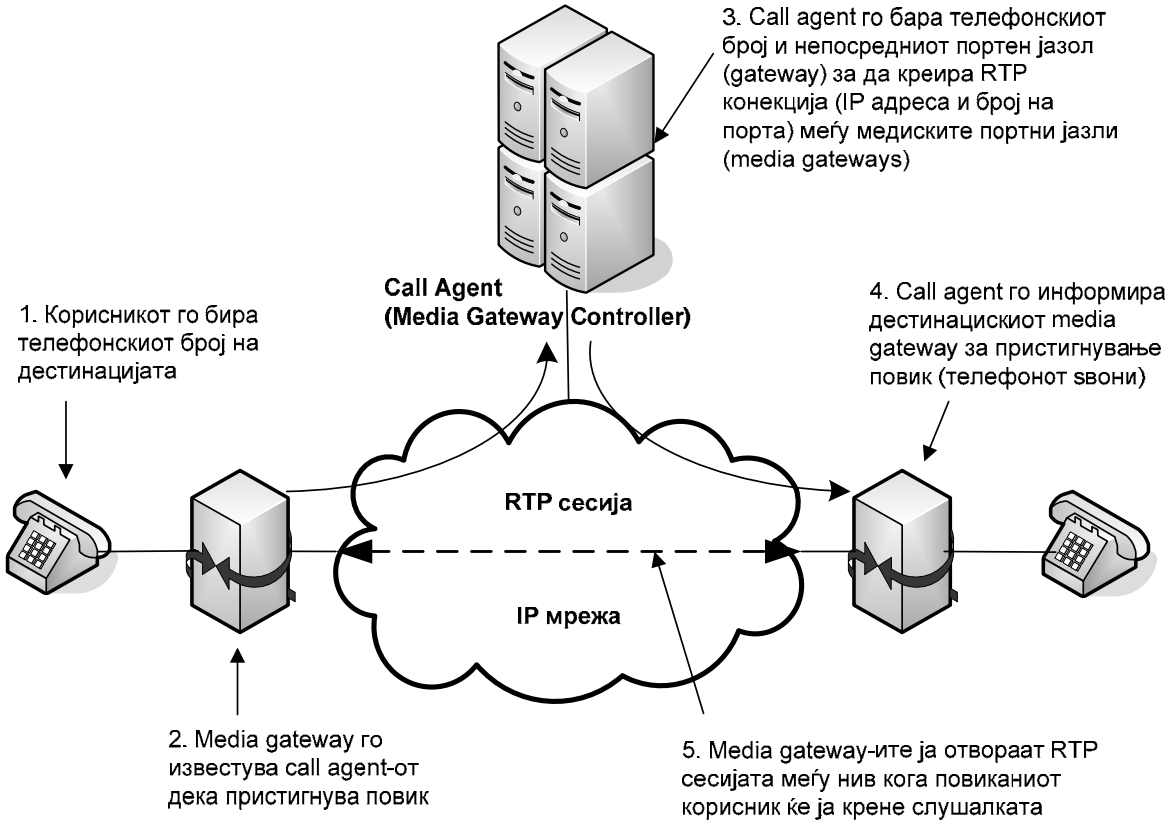
Протоколи кои се користат во IP телефонијата за воспоставување и прекин на говорните повици во IP мрежата се H.323 (од ITU), MGCP (или H.248 од ITU) и SIP (Session Initiation Protocol) како протокол од IETF. Тие потекнуваат од мултимедиските конференциско-мрежни апликации и се специјализирани за говорни комуникации, но сепак ја имаат задржано способноста за мултимедиски комуникации.

Во продолжение ќе ги изложиме основните карактеристики на некои од протоколите.

7.4.1 MGCP (Media Gateway Control Protocol)

Media Gateway Control Protocol (MGCP) ја вметнува улогата на традиционалните говорни комутатори во компоненти на т.н. медија портен јазол (media gateway), медија портен контролер (media gateway controller) и функционалните компоненти на сигнален портен јазол (signaling gateway).

MGCP е протокол со контрола надреден-подреден (master-slave) што ги координира активностите на media gateway-овите (слика 7.6). Media Gateway контролерот во номенклатурата на MGCP е познато како повикувачки агент (call agent). Повикувачкиот агент (call agent) ја менаџира логичката контрола за управување на сигнализацијата поврзана со повикот, додека медија портниот јазол (media gateway) го информира повикувачкиот агент (call agent) за настаните во сервисите. Повикувачкиот агент (call agent) му дава инструкции на медија портниот јазол (media gateway) за воспоставување и раскинување на врски кога се генерира VoIP повик. Најчесто повикувачкиот агент (call agent) го информира медија портниот јазол (media gateway) да започне RTP сесија помеѓу две крајни точки.



Слика 7.6 MGCP функции

Сигнализацијата помеѓу повикувачкиот агент и медија портниот јазол е во форма на пораки кои се сместени во UDP пакети. Повикувачкиот агент и медија портниот јазол овозможуваат ретрансмисија на овие пораки.

7.4.2 H.323 сигнализација за Интернет Телефонија

Интернет телефонијата побарува широк опсег на протоколи, почнувајќи од тие потребни за транспорт на податоци во реално време, па се до протоколи за резервација на ресурси, интелигентни рутирачки протоколи, протоколи за менаџирање на мрежата и протоколи за наплата на услугите. Покрај тоа, Интернет телефонијата, дефинирана овде како пренос на синхронизиран говор или мултимедиска комуникација меѓу два или повеќе ентитети, бара средства и начини на кој начин еден корисник ќе му сигнализира на друг дека сака да комуницира со него (класичниот пристап е со користење на телефонскиот број на бараниот корисник од страна на корисникот кој го иницира повикот). Оваа функционалност ќе ја означиме како сигнализација во Интернет телефонија. Токму потребата од ваква сигнализациска функционалност ја разликува Интернет телефонијата од останатите Интернет мултимедиски сервиси, како што се дифузни сервиси (broadcast) или медија на барање (media-on-demand).

Еден IP телефонски протокол извршува голем број на функции. IP телефонскиот сигнализациски протокол вообичаено поседува опција која овозможува договор за карактеристиките. Оваа функција дозволува крајни системи да се договорат за нивните параметри, на пример каков типот на кодирање ќе користат и каква медија ќе разменуваат. Бидејќи различни точка-точка (point-to-point) конекции вклучуваат различна медија и различни медиски параметри, не е потребно сетот и типот на медија да бидат униформни долж повикот. Постојат софтверски кодеци кои се способни да примат различни формати на кодирање во рамките на еден повик, но се ограничени на испраќање на само еден тип на медија за секој поток (или обратно).

Секој учесник во повикот може да повика други да учествуваат во тековен повик или да ги прекине конекциите со некои учесници од повикот. Оваа функционалност ја нарекуваме менаџирање на повикот од страна на учесник.

Промените на карактеристиките овозможуваат прилагодување на составот на медиските сесии во текот на одвивање на повикот затоа што некои учесници побарале зголемување или намалување на нивните функционалности поради додавање или отстранување на учесници во повикот.

H.323 протокол

H.323 е заснован на H.225 протоколот за повикувачка сигнализација кој е сличен на протоколот применет во ISDN H.245 протоколот кој го контролира податочниот проток помеѓу терминалите. H.323 го обезбедува интерфејсот помеѓу IP говорните и видео уредите и IP или IPX (Internetwork Packet Exchange) мрежите. Конкретните аудио/видео уреди и IP/IPX мрежи се надвор од интересот на H.323. Gateway сервисите му овозможуваат на H.323 протоколот интерфејс со класични телефонски мрежи како PSTN. Овој протокол нуди можност за поширок избор на стандарди за кодирање на говор, вклучувајќи ги стандардите: G.711 (импулсно кодна модулација, 64 Kbit/s), G.726 (адаптивна диференцијална импулсно кодна модулација, 32 Kbit/s) и G.723.1 (линеарно предиктивно кодирање, 5.3 Kbit/s). Кодирањето на видео сигнали ги вклучува H.261 и H.263 стандардите кои оперираат на околу 28.8 kbit/s.

H.225 (повикувачка сигнализација) и H.245 (контрола на преносот) го употребуваат TCP како нивни протокол на транспортно ниво за да обезбедат доверлив пренос на пораките. Повеќето од H.323 имплементациите ги пренесуваат податочните, говорните и видео пакетите со користење RTP врз UDP (RTP/UDP протоколен стек).

Меѓутоа, H.323 страда од неколку сериозни недостатоци:

Комплексност: H.323 спецификацијата опфаќа повеќе стотици страни и постојано се зголемува. Неговата имплементација во C++ опфаќа програмски код од близу стотина илјади линии.

Скалабилност: H.323 бил дизајниран за конференции преку локални мрежи. Бидејќи сите функции на овој протокол се базирани на централен конференциски сервер се појавуваат проблеми со скалабилноста при големи конференции.

Проширливост: Додавањето на нови елементи, карактеристики и заглавија, како и управување со компатибилноста преку различни верзии не е лесно да се реализира. H.323 исто така работи само со мал број на стандардизирани ITU кодеци за говор и видео.

Овие недостатоците кои се јавуваат кај H.323 протоколот се анулирани помалку или повеќе кај SIP протоколот, со што тој стана моќен, флексибилен, едноставен и скалабилен протокол кој може да послужи како добра основа за широка употреба на Интернет телефонијата. Од тие причини тука само информативно го спомнавме H.323, а во продолжение ќе го опфатиме SIP како de-facto глобален и единствен прифатен стандард за сигнализација за Интернет телефонија, кој наскоро и целосно ќе ја замени

сигнализацијата SS7 која се користи за телефонија и во фиксните и во мобилните мрежи.

7.4.3 SIP (Session Initiation Protocol)

SIP (Session Initiation Protocol) е сигнализациски протокол кој ги креира, модифицира и терминира врските меѓу крајните Интернет системи, вклучувајќи конференции и точка-точка повици, [32]. Тој подржува уникаст, меш и мултикаст конференции, како и комбинации од овие модови. SIP имплементира сервиси како што се трансфер и препраќање на повик, ставање на повици во состојба на чекање итн. Имплементацијата на SIP овозможува реискористување на делови од други Интернет сервис протоколи, како што се HTTP и RTSP. Во понатамошниот дел од текстот ќе го опишеме SIP и ќе покажеме како се користат неговите основни примитиви за да се конструира широк опсег на телефонски сервиси.

SIP е клиент-сервер протокол. Ова значи дека барањата се генерираат од еден ентитет (клиентот) и се праќаат на приемниот ентитет (серверот) кој ги обработува и враќа одговори на клиентот. Во случај на IP телефонијата, повикувачот се однесува како клиент, а повикуваниот како сервер. Еден повик може да вклучи неколку сервери и клиенти, но и еден единствен хост може да се однесува и како клиент и како сервер за еден повик.

Барањата и одговорите се текстуални и содржат полиња со заглавија кои ги пренесуваат својствата на повикот и информациите за сервисот. SIP реискористува многу од полињата кои се користат во HTTP, како што се заглавието за ентитетот (Content-type) и заглавието за автентикација. Ова овозможува лесна интеграција на SIP серверите со веб и mail серверите.

Повиците во SIP се идентификуваат со т.н. идентификувач на повик (Call Identifier - Call-ID), сместен во Call-ID полето на SIP пораката. Call-ID е создаден од креаторот на повикот и го користат сите учесници во повикот.

SIP дефинира неколку методи, од кои првите три ги подготвуваат и управуваат со повиците и врските: INVITE поканува корисник на конференција, BYE раскинува конекција меѓу два корисника во конференција, OPTIONS бара информации за можностите, но не го воспоставува повикот. STATUS информира друг сервер за

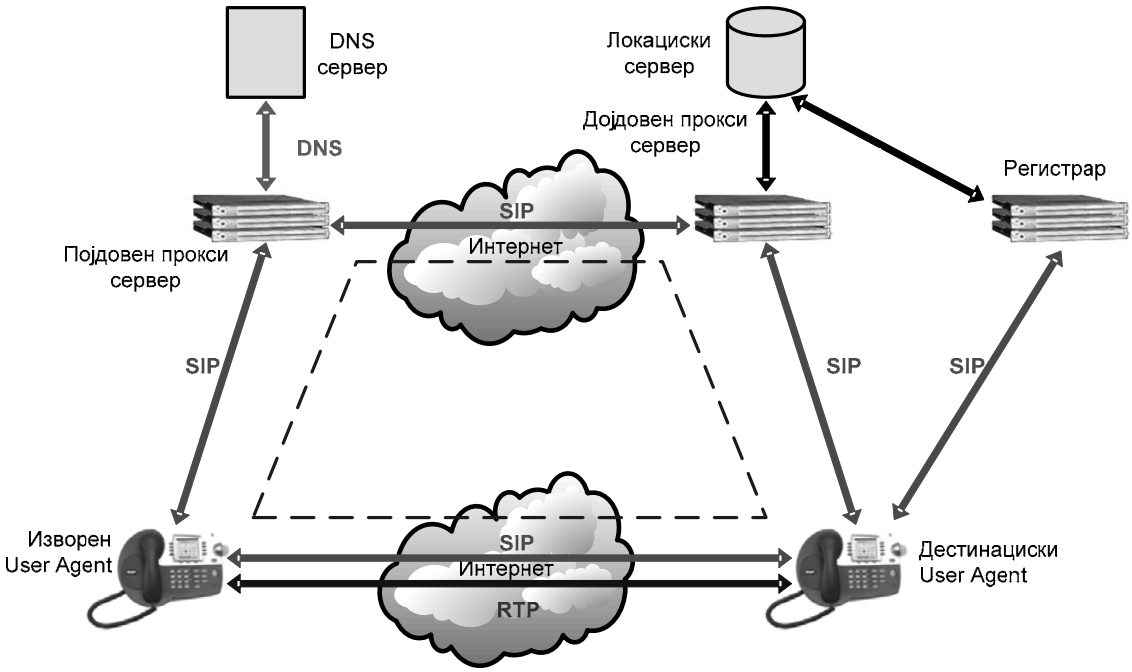
прогресот на сигнализацискатa акција која била побарана. ACK се користи за сигурна размена на порака за поканување на повик. CANCEL ја завршува потрагата по корисник. И конечно, REGISTER ја пренесува информацијата за локацијата на SIP серверот.

Адресирање и именување

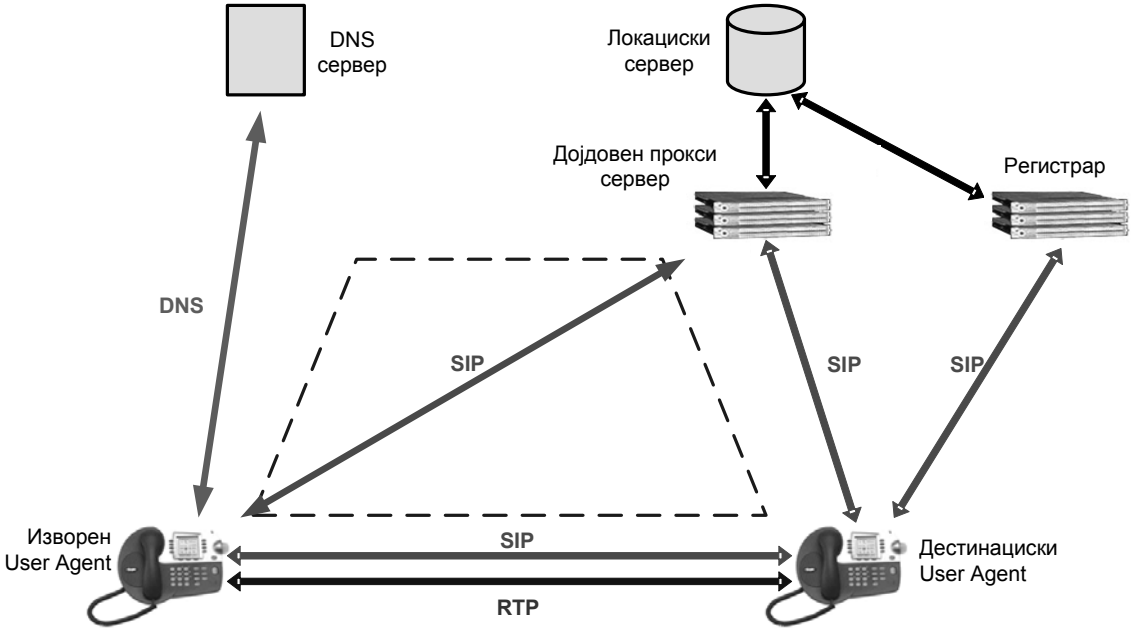
За да се покани и идентификува ентитетот кој се повикува мора да е именуван. Поради најчесто сретнуваната форма на адресирање на корисниците во Интернет, SIP избира адреса од типот на e-mail адреса: “user@domain", "user@host", "user@IP_address" или "phone-number@gateway". Името на доменот на првиот тип на адреса може да е име на хостот на кој е логиран корисникот или некое специфично име на самиот домен доколку врши, на пример, транслација на некој сервис. Адресите од типот “phone-number@gateway” ги именуваат PSTN телефонските броеви со што тие стануваат достапни преку именуваната порта.

Се претпоставува дека повеќето од корисниците ќе бидат во можност да ги користат e-mail адресите како нивни SIP адреси. E-mail адресите веќе понудија основна локациско-независна форма на адресирање.

За e-mail, наоѓањето на хостот за размена на пораки е доволно за испорачување на пораката, бидејќи корисникот или е логиран на хостот за размена на пораки или користи протоколи како што се IMAP или POP за да ги земе своите пораки. Меѓутоа, за интерактивна аудио и видео комуникација учесниците испраќаат и примаат податоци на работна станица, PC или Интернет алатка во нивна непосредна физичка близина. Поради тоа, SIP мора да воспостави врска меѓу “name@domain” и “user@host”. Една надворешно видлива адреса може да води до различни хостови во зависност од тоа во кој период од денот се воспоставува конекција или каков тип на медија ќе се пренесува. Исто така, хостовите кои се конектираат преку dial-up модеми можат да добиваат (преку DHCP) различни IP адреси во различно време (на пример: различни dial-up сесии).



а) SIP трапез (VoIP корисниците кои комуницираат се во два различни домени)



б) SIP триаголник (VoIP корисниците кои комуницираат се во еден ист домен)

в) SIP peer-to-peer комуникација

Слика 7.7 SIP сценарија за VoIP

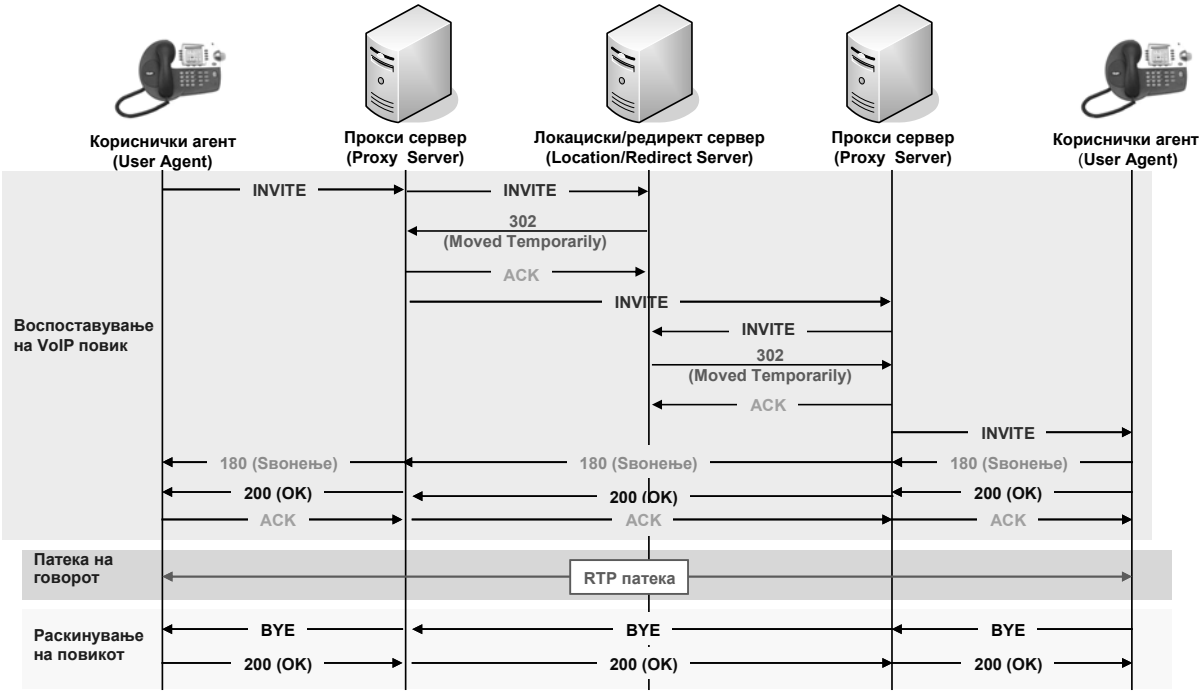
Функционирање на SIP

Поканувањето на нови учесници на повик е најважната операција на SIP протоколот. SIP клиентот прво добива една адреса која покажува каде се новите учесници со кои треба да контактира. Таа адреса е од типот “name@domain”. Тогаш клиентот се обидува да го преведе овој домен во IP адреса на која може да се најде серверот (на пример: со DNS). Кога ќе се пронајде IP адресата, клиентот им испраќа INVITE порака користејќи или UDP или TCP.

Бидејќи серверот кој ќе ја прими пораката обично не е хостот на кој корисникот е логиран, постојат три различни типа на мрежни јазли при SIP комуникацијата: прокси сервер (proxy), сервер за пренасочување заедно со локациски сервер и регистрат, и како трет тип - кориснички агент (user agent), [33].

Proxy серверот го прима барањето од корисникот повикувач и го препраќа кон моменталната локација на оној кој се повикува. На пример, серверот одговорен за primer.com.mk може да го препрати повикот за trpe.trpkovski@primer.com.mk кон trpe@prodazba.primer.com.mk. На Сликата 7.7 се прикажани три сценарија за користење на SIP за сигнализација за VoIP конекции. Слика 7.7а покажува генерална слика на употреба на SIP за сигнализација меѓу два VoIP корисника (т.е. два SIP кориснички агенти) кога истите се наоѓаат во два различни домени (најчесто тоа значи дека се опслужуваат од два различни VoIP сервис провајдери, каде што секој VoIP сервис провајдер има свој прокси сервер за својот домен). На слика 7.7.б е прикажан SIP триаголник, што е сценарио кога двата VoIP корисника (SIP кориснички агенти) се наоѓаат во ист домен, опслужуван од ист прокси сервер. Третиот случај, илустрира на слика 7.7в е случајот на користење на SIP за peer-to-peer телефонија, каде што ги нема мрежните елементи (кои ги има во сценаријата на слика 7.7а и 7.7б). Вообичаено прокси серверите се користат за да ги поврзат различните мрежни домени меѓу себе (во дистрибутивниот дел, меѓу различните домени), или пак да ги поврзат корисниците со мрежните SIP јазли (од страна на пристапната мрежа каде што се закачени VoIP корисниците).

Корисничкиот агент (user agent) се наоѓа на хостот (на пример: компјутерот, IP телефонот, IP домашниот портен јазол, мобилниот уред, итн.) кој го користи корисникот за иницирање или добивање (терминирање) на VoIP повици. Корисничкиот агент иницира SIP пораки (на пример, при појдовен телефонски повик) или терминира и одговара на SIP пораки (на пример, одговара со потврдна SIP порака ACK), [34].



Слика 7.8 SIP дијаграм за VoIP

Принципот на пораки и одговори кај SIP е многу сличен како кај HTTP, што може да се забележи од SIP дијаграмот на пораки за воспоставување на VoIP повик, како што е прикажано на слика 7.8. Така, секој SIP елемент на дадена порака испраќа одговор назад. Одговорот содржи код и причина поради која се праќа тој код. Кодовите спаѓаат во една од класите 100 до 600, слично како кај HTTP. На пример, одговорите за успешност содржат само една порака, а тоа е пораката "200 OK", а пренасочувањето (redirect) од даден сервер се реализира преку испраќање на пораката "302 Moved Temporarily".

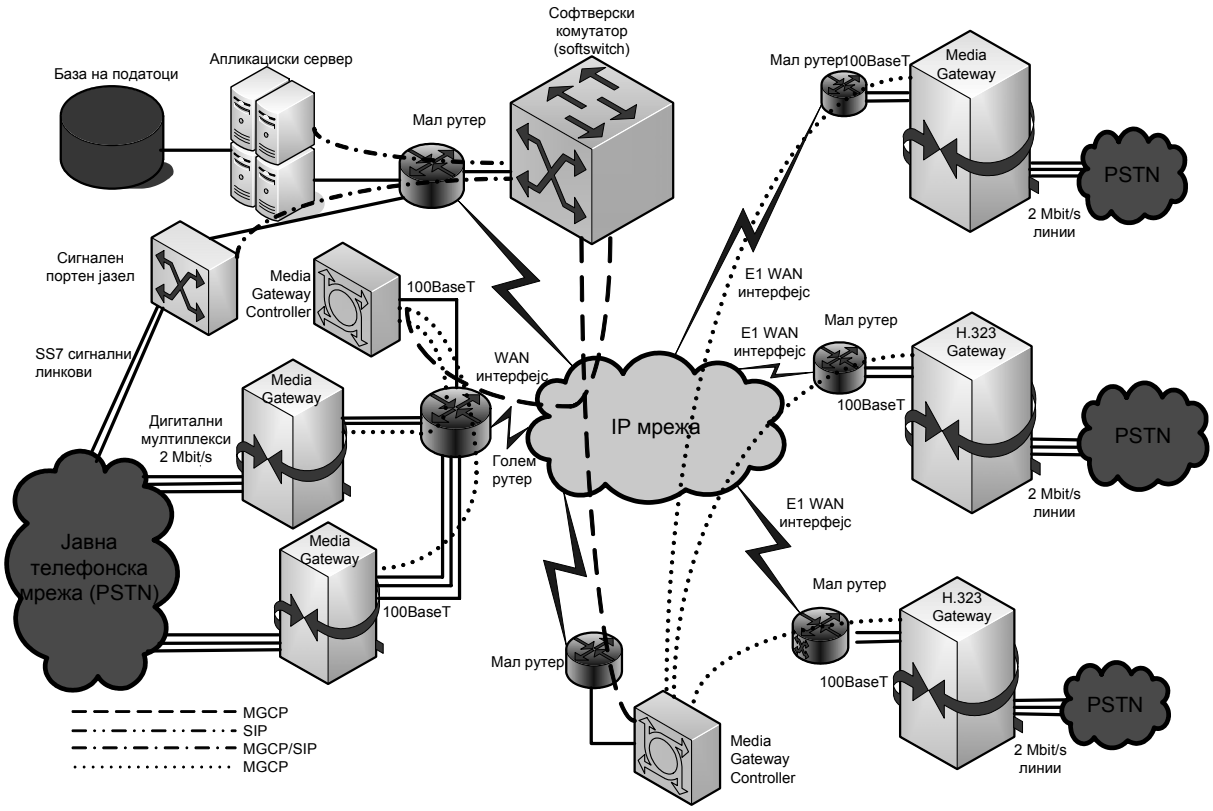
За разлика од другите барања, на поканата за повик не може веднаш да се одговори бидејќи лоцирањето на оној кој го повикуваме и чекањето истиот да одговори одзема неколку секунди. Повикот може да се стави во ред на чекање ако оној што го повикуваме е зафатен. Одговорите од класата 100 (означени како 1xx) го назначуваат прогресот на повикот (информативни се), така да тие секогаш се придружени со други одговори формирајќи го на тој начин крајниот исход од барањето.

Додека одговорите од класата 100 се привремени, одговорите од другите класи го назначуваат финалниот статус на барањето: 2xx за успех, 3xx за пренасочување или препраќање, 4xx, 5xx и 6xx за неуспеси на клиентот, серверот и глобални неуспеси, соодветно. За да се подигне сигурноста на повисоко ниво серверот врши препраќање на

финалниот одговор се додека клиентот не го потврди приемот со ACK барање кон серверот.

7.5 Архитектура за VoIP

Во овој дел е презентиран VoIP мрежни модели во кои се имплементираат претходно споменатите протоколи.



Слика 7.9 Мрежен модел за VoIP

Мрежниот модел за VoIP, прикажан на слика 7.9, ги опфаќа сите решение со претходно изложените протоколи за VoIP сигнализација. Ја спомнуваме сигнализацијата, бидејќи таа е клучниот момент кој влијае на архитектурата т.е. има пресудно значење за да се воспостави врската меѓу два корисника на Интернет. Откако ќе се воспостави врската меѓу корисниците, потоа говорот се пренесува со користење на RTP со контрола преку RTCP.

За да може да се повикуваат броеви од PSTN мрежата од страна на VoIP телефон или обратно, потребно е да постои во мрежата на местото каде што се остварува врската меѓу PSTN и Интернет т.н. сигнализациски gateway (signaling gateway – SG) како интерфејс меѓу Интернет и PSTN со SS7 (SS7 - Signaling System 7 се користи за сигнализација при воспоставување и раскинување на повици во класичните телефонски мрежи денес), а контролата врз media gateway-ите (media gateways – MG) ја врши преку media gateway контролерите (Media Gateway Controllers – MGC).

Апликациите (на страната на корисниците и нивните компјутери) се поддржани од т.н. апликациските сервери (application servers) кои преку соодветен интерфејс комуницираат со softswhitch-от (слика 7.9), а исто така комуницираат и со своите бази на податоци (databases) каде што ги складираат сите потребни информации за корисниците и нивните сервиси.

Мрежниот модел се одликува со добра функционална декомпозиција бидејќи секоја функција се извршува од посебен систем. Во поглед на архитектурата, се забележува концептот клиент-сервер (client-server) што е и очекувано бидејќи овој модел потекнува од организацијата која е задолжена за Интернет стандардите - IETF. На пример, softswhitch-от може да се разгледува како сервер и како таков ги користи MGCP (Media Gateway Control Protocol) или Megaco/H.248 протоколите за да ги контролира media gateway контролерите, кои, се разбира, се негови клиенти. Softswhitch-от го користи MGCP за да комуницира со сигнализацискиот gateway; овој вториот, може да функционира како сервер и/или како клиент, а како таков го користи SIP протоколот за да комуницира со апликацискиот сервер.

Ваквиот пристап обезбедува добра раздвоеност на различните компоненти во мрежата. Но сеуште не постојат стандардни протоколи за комуникација меѓу апликацискиот сервер и софтверскиот комутатор (softswhitch), ниту пак протоколи за комуникација меѓу различни софтверски комутатори (softswitch) и покрај тоа што SIP протоколот е предложен за употреба во споменатите комуникации. Како недостаток може да се спомене и тоа што мрежниот модел не дефинира протоколи за комуникација меѓу апликацискиот сервер и самите апликации.

Треба да се спомене дека мрежниот модел има недостатоци. Еден softswhitch може да поддржи онолку повици, media gateway-и, media gateway контролери се додека не стане тесно грло и се додека не предизвика отежнување на работата на целата мрежа. Исто така овој моделот има недостаток заради проблеми во интерконекцијата меѓу различни провајдери.

Моделот може да поддржи SS7 сигнализација од крај до крај (end-to-end) ако и самите softswhitch-ови ја поддржуваат и ја користат од крај до крај. Тоа би значело дека ниту еден од H.323, MGCP, Megaco/H.248 протоколите не се користат ексклузивно меѓу системите.

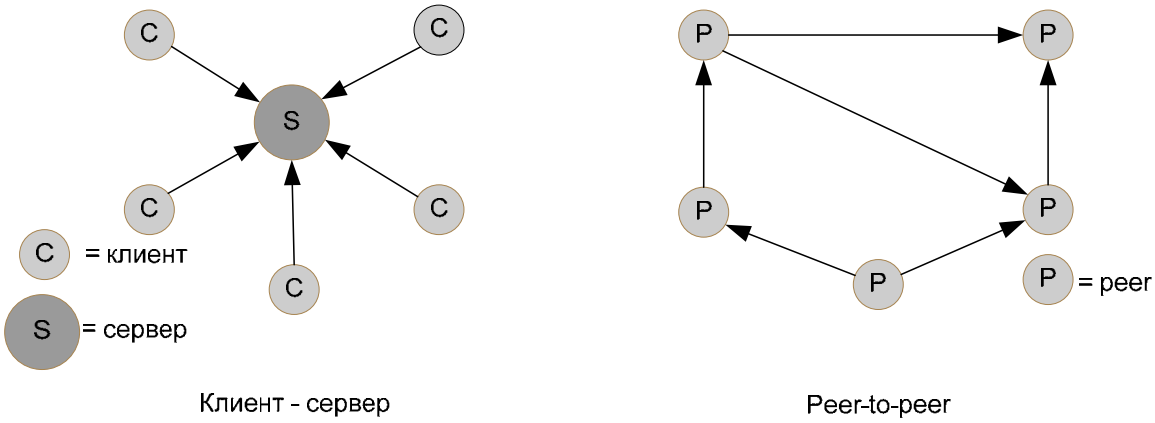
7.6 Peer-to-Peer Интернет Телефонија

P2P (Peer-to-peer) системите имаат висока скалабилност и робустност бидејќи се самоорганизирачки и немаат централен сервер. Но овие карактеристики влијаат на зголемувањето на времето потребно за лоцирање на ресурсите во P2P мрежата. Интернет телефонијата може да се гледа како една апликација на P2P архитектура во која учесниците формираат самоорганизирачка P2P мрежа за да комуницираат со други учесници. Во овој дел ќе биде презентирана чиста P2P архитектура за SIP-базирани IP телефонски системи. Оваа P2P-SIP архитектура покрај регистрација на корисник и воспоставување на повик подржува и некои напредни сервиси како што се испорака на offline пораки, говорна и видео пошта и multi-party конференции. Ќе бидат разгледани и некои предизвици за P2P-SIP како што се firewall, Network Address Translator (NAT) и безбедност.

7.6.1 Карактеристики на P2P системите

Постоечката клиент-сервер архитектура на Интернет телефонијата базирана на SIP или ITU-T H.323 препораката употребува сервер за регистрација во секој домен. Корисничките агенти (или IP телефоните) ги регистрираат нивните IP адреси на серверот во доменот за да можат други корисници да контактираат со нив.

Од друга страна, P2P системите се скалабилни и сигурни бидејќи не содржат единствената точка во која може да се случи испад. Како што е покажано на Слика 7.10, P2P системите немаат концепт на сервери. Секој учесник кој е приклучен на мрежата претставува јазол со рамноправен пристап до мрежните ресурси и до сите другите корисници.



Слика 7.10 Споредба на клиент-сервер и P2P системи

P2P системите може да се поделат на неструктурни, како што е Kazaa, кои не поседуваат посебна структура за складирање на фајлови, и структурни, како тие што користат DHT. Неструктурните системи се концентрирани на практичните проблеми, како што се NAT и firewall, но потрагата по соодветен корисник вклучува испраќање на дифузни пакети до сите соседни јазли, метод познат како поплавување. Од друга страна, структурните системи се фокусираат на оптимизација на времето потребно за пребарување наместо користење на неефикасниот слеп начин со поплавување.

7.6.2 Skype

Skype е бесплатна P2P апликација базирана на архитектурата на Kazaa која овозможува реализирање на повик преку Интернет со било кој друг Skype корисник. Skype ги има следниве проблеми:

1. За разлика од отвореноста на SIP каде секој може да пристапи, интелектуалната сопственост на Skype е заштитена.
2. Тој обезбедува едноставни сервиси, како што се реализирање на говорни повици и инстант пораки.
3. Не обезбедува архитектура за нови напредни сервиси.
4. Има централизирани елементи за автентикација при логирање што значи дека ако овој елемент испадне од работа, системот нема да работи.

Skype архитектурата не се разликува многу од класичната SIP телефонска архитектура, освен што т.н. Global Index Server назначува супер јазол за новиот јазол кој се придружува, [35]. Супер јазолот ја потврдува информацијата за присутноста на овој јазол и лоцира други корисници комуницирајќи со други супер јазли. Секој јазол кој има доволно капацитет и способност може да стане супер јазол. Пребарувањето кај Skype е базирано на некоја варијација на преплавување (flooding), наместо употреба на поефикасно дистрибуирано пребарување.

Цели при дизајнирање на P2P-SIP телефонска архитектура

Врз основа на прегледот на постоечките P2P системи (Skype), P2P-SIP телефонската архитектура треба да ги поседува следниве карактеристики:

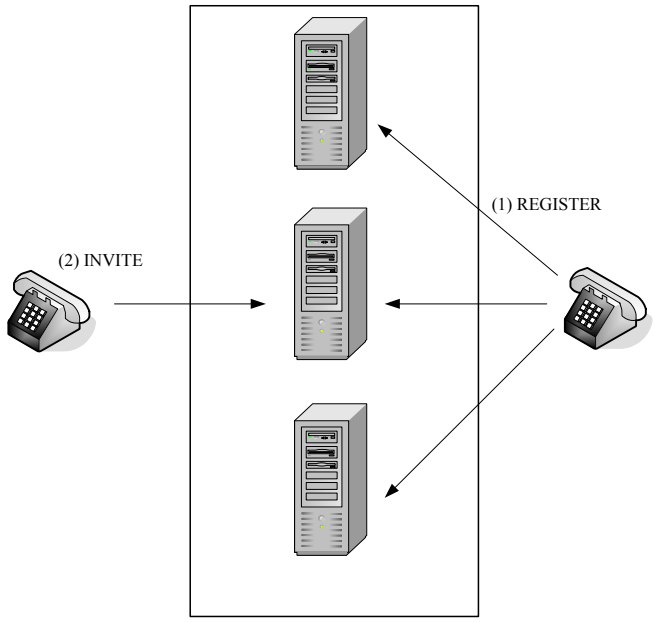
- **Zero конфигурација:** Системот треба да е способен автоматски да се конфигурира со детектирање на NAT и поставување на firewall, откривање на соседни јазли и извршување на почетна регистрација.
- **Хетерогени јазли:** Треба да е во состојба да ги прилагоди своите расположливи ресурси и да ги разликува јазлите по нивниот капацитет кон Интернет (во bit/s) и нивните способности.
- **Ефикасно пребарување:** "Слепиот" начин на пребарување базиран на поплавување е неефикасен. Системот треба да користи подлога за дистрибуирано пребарување - DHT (Distributed Hash Tables), за да се оптимизира пребарувањето.
- **Напредни сервиси:** P2P Интернет телефонијата треба да подржува напредни телефонски сервиси како што се offline говорни пораки, multi-party конференции, пренасочување на повик, трансфер на повик и инстант пораки.
- **Интероперабилност:** P2P телефонската архитектура треба лесно да се интегрира со постоечките протоколи и IP телефонската инфраструктура.

Дизајн алтернативи

Во понатамошниот дел од текстот ќе изложиме неколку дизајн алтернативи кои овозможуваат реализирање на горенаведените цели. Постојат генерално неколку различни дизајн алтернативи за P2P мрежи:

- Копирање на информацијата за локацијата на корисникот на сите сервери;
- Потрага по коректниот сервер кој ја поседува информацијата за локацијата на корисникот;
- Комбинација на претходните дизајн алтернативи и употреба на DHT при што регистрацијата се врши на logN сервери.

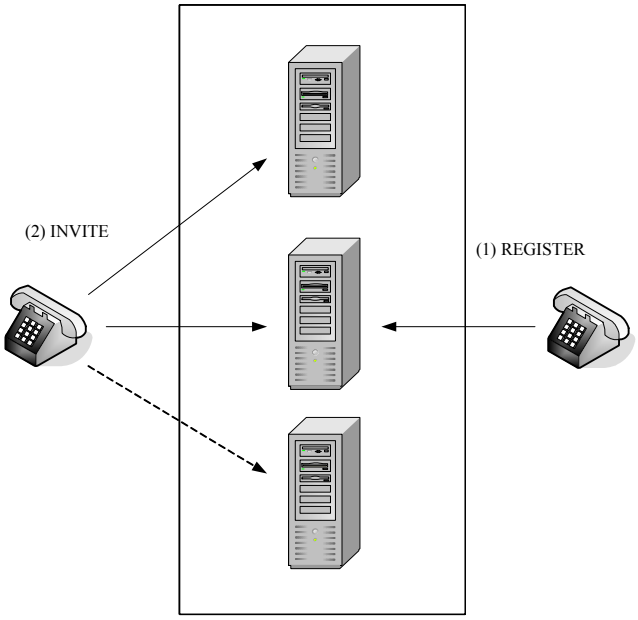
На сликата 7.11 е прикажана повеќекратна регистрација, но всушност станува збор наизменично копирање на базата на податоци од сервер во сервер во рамките на еден кластер. Во вториот случај прикажан на сликата 7.12, или повикувачот за да го пронајде корисникот поединечно пристапува кон секој сервер или првиот сервер на кој се конектирал го продолжува барањето по корисникот.



Слика 7.11 Дизајн: копирање на информацијата за локацијата на корисникот во сите сервери

Проблемот со првиот случај е што тој повлекува синхронизација за секоја регистрација. Постои опасност од баналeн запис на локацијата на корисникот на некои сервери во краток временски интервал откако е извршено едно корегирање на регистрацијата, бидејќи додека другите сервери да ја добијат таа информација можно е да се случи уште едно корегирање на регистрацијата. Ако регистрацијата се корегира на секој час по корисник, тогаш оваа архитектура го ограничува вкупниот број на

корисници подржани од системот, бидејќи синхронизациониот сообраќај ќе овозможи појава на ефектот на “тесно грло”.



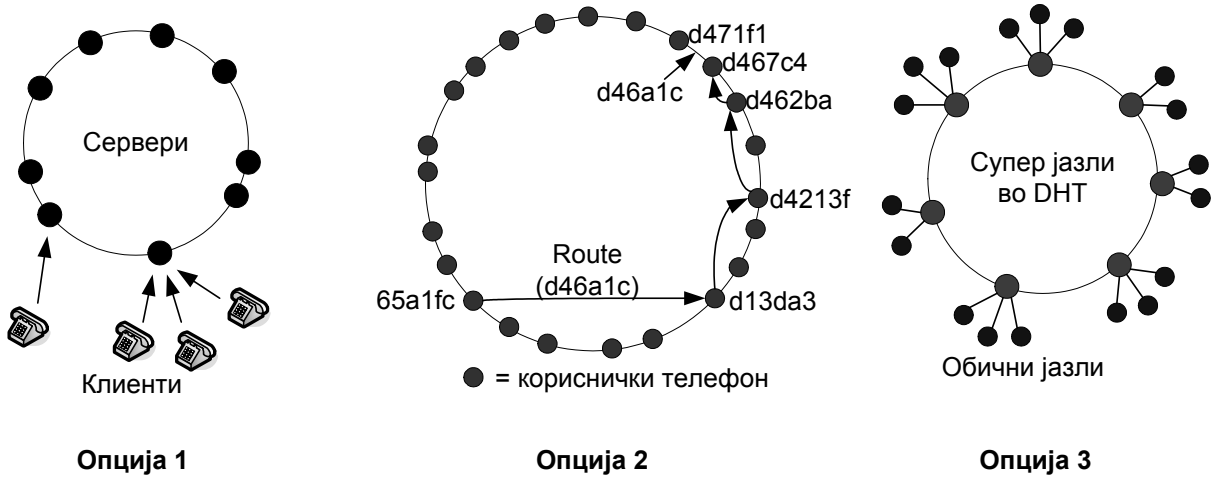
Слика 7.12 Дизајн: барање на вистинскиот серверот кој ја поседува информацијата за локацијата на корисникот

Во вториот случај, времето потребно за воспоставување на повик се зголемува со секое наредно барање. Ова навидум може да се реши со истовремено барање кон сите сервери во кластерот, но тоа ќе имплицира зголемување на барањата за опсег. И двете дизајн алтернативи појавуваат големи проблеми кога бројот на сервери е многу голем. Во понатамошниот дел од текстот ќе се фокусираме на вториот случај.

Формирање на DHT

Претходно разгледаните дизајн алтернативи може да се комбинираат со употреба DHT, така што регистрацијата ќе се врши на LogN сервери наместо на сите N сервери. Ова директно влијае на намалување на времето потребно за воспоставување на повик, бидејќи во овој случај барањето на корисникот ќе се врши на LogN, наместо на N сервери. Постојат три дизајн алтернативи кои овозможуваат употреба на DHT. Во првиот случај на Слика 7.13, DHT може да го ограничиме на “серверската фарма” (група компјутери - сервери). Во овој случај секој корисник е конектиран на еден од серверите. Серверите го применуваат DHT за да го лоцираат кориснички запис. При тоа

претпоставуваме прстенеста топологија како подлога за DHT. Оваа архитектура се уште е клиент-сервер ориентирана. Имено, корисникот треба да открие барем еден сервер, кој е пожелно да биде послабо оптоварен, и да се конектира на истиот. Во вториот случај клиентот се однесува како сервер и го применува чистиот P2P концепт со сите други клиенти. Првата опција не бара модифицирање на клиентите, а обезбедува скалабилна и сигурна архитектура на серверската фарма. Но, се уште постојат одредени проблеми со одржувањето на серверот и конфигурацијата за разлика од втората опција.



Слика 7.13 Формирање на DHT

Еден од главните проблеми со P2P концептот е фактот што сите јазли немаат еднаков капацитет и употребливост (т.е. време на исправно функционирање). Имено, јазлите со слаба конекција (мал опсег во bit/s) кон Интернет или оние јазли кои се зад firewall и NAT не се способни за потполна функција во DHT бидејќи може да има потреба од in-band конекции, што значи дека е потребен дополнителен опсег што некои од овие јазли не се во можност да го обезбедат. На пример, потребен е значаен опсег за препраќање на P2P пораки или значителна меморија или CPU за одржување на DHT состојбата. Овој проблем може да се реши со примена на третата опција од Сликата 7.13. Некои од јазлите со голем капацитет (опсег, CPU, меморија) и употребливост стануваат супер јазли. Само овие јазлите го формираат DHT. Обичните јазли се конектираат на еден од достапните супер јазли. Ова е слично со првата опција, освен што не постои разлика меѓу клиенти и сервери и секој јазол може да биде супер или обичен јазол, зависно од неговиот капацитет и употребливост.

Одлуката дали еден јазол ќе биде обичен или супер е локална. Кога јазолот ќе почне да функционира станува обичен јазол. Кога обичниот јазол ќе детектира доволно капацитет и употребливост (долго време на исправно функционирање) ќе премине во супер јазол. Јазолот со доволно капацитет и употребливост може да биде присилен да стане супер јазол ако постоечкиот супер јазол испаднал од работа или стасал до лимитот на капацитетот. Од друга страна, некои јазли кои знаат дека поседуваат доволно капацитет и употребливост може да станат супер јазли веднаш по нивното стартување. Исто така, некои јазли имаат моќ да влијаат на соседите да станат супер јазли.

Постоењето на двонивовската структура, супер јазли и обични јазли, не го намалува времето потребно за пребарување. Сепак оваа архитектура влијае на подобрување на перформансите во практиката, бидејќи потребата за одржување на DHT сообраќајот се намалува ако јазлите во DHT се постабилни, т.е. ако долго време опстојуваат како супер, односно обични јазли.

7.6.3 Архитектура и функционирање на P2P VoIP

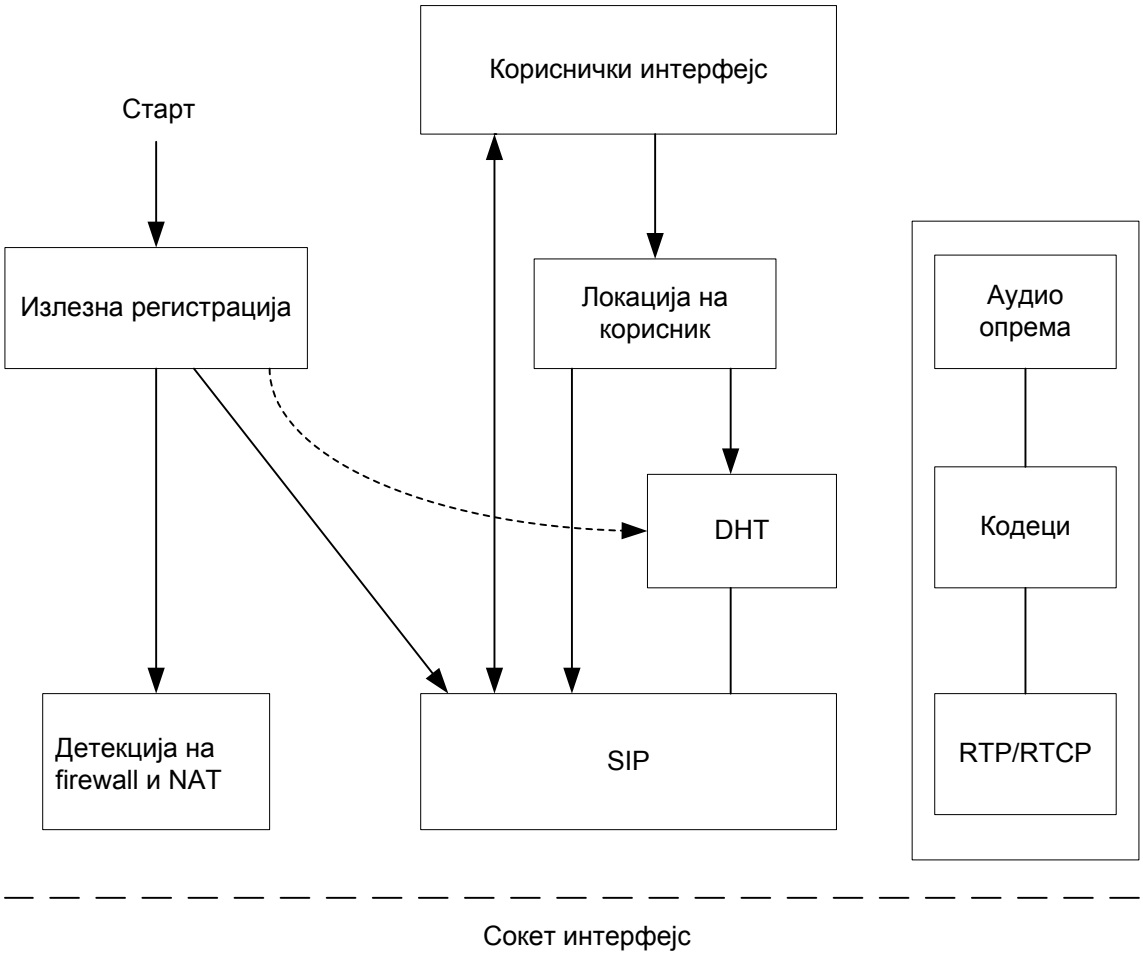
Во продолжението на оваа глава е предложена P2P VoIP архитектура врз основа на различните алтернативи споменати во претходното поглавје. Сликата 7.14 претставува блок дијаграм на различните компоненти кај P2P.

Кога јазолот стартува и корисникот се пријавува со неговото ID се активира модулот за регистрација за да иницира NAT и firewall детекција, откривање на ентитет и SIP регистрација.

Корисничкиот интерфејс комуницира директно со корисникот и го повикува модулот за локација на корисник да ги лоцира корисниците.

Модулот за локација на корисник може да го повика SIP модулот или ако станува збор за супер јазол, DHT модулот, за да го лоцираат корисникот.

DHT модулот се користи кога станува збор за супер јазол за чување на информацијата за јазлите кои го формираат DHT и за извршување на логиката на DHT, а таа е преминувања од обични во супер јазли и обратно. За оваа цел тој ги користи SIP пораките за да комуницира со другите јазли.



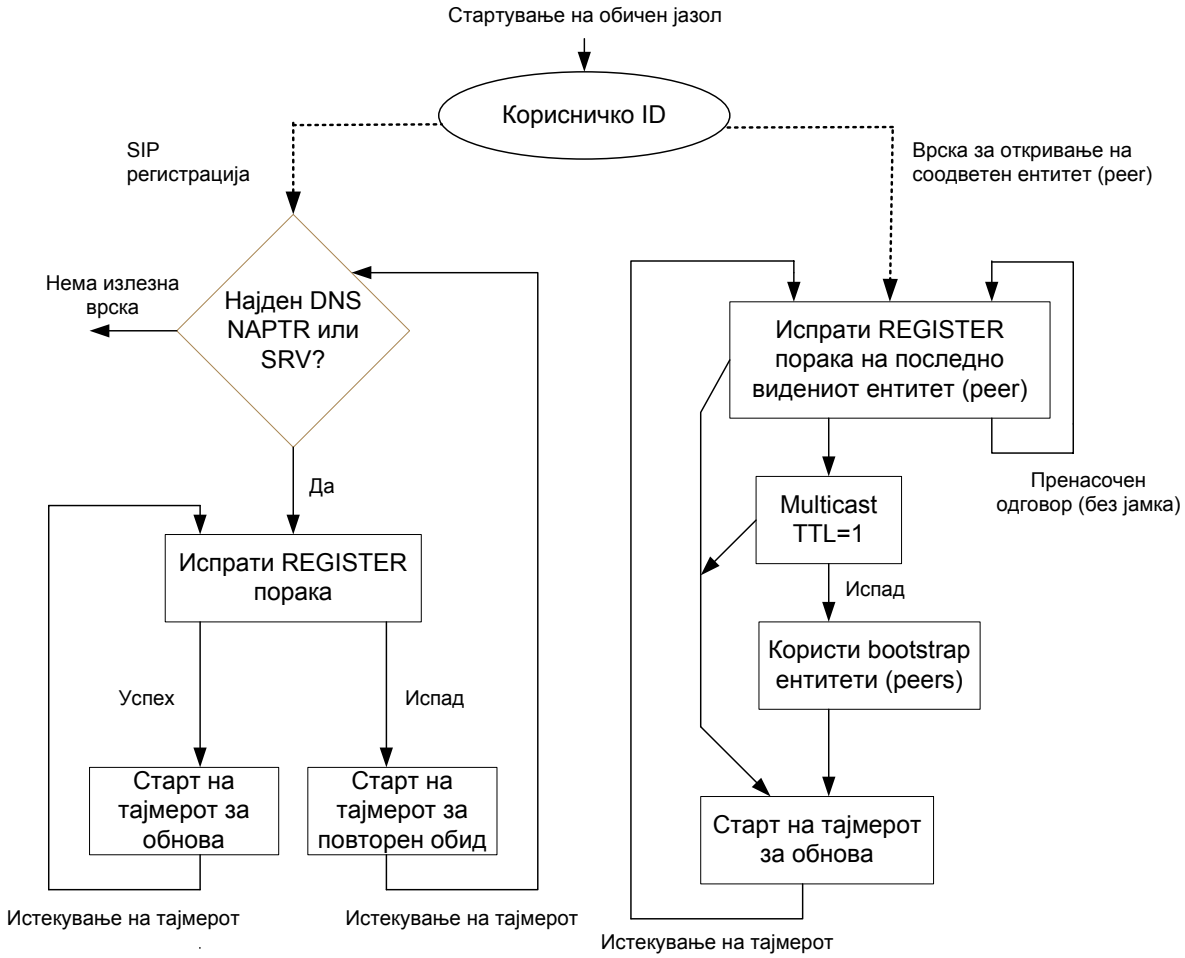
Слика 7.14 Блок дијаграм на P2P-SIP јазол

Во практиката, јазолот може да се обиде да ги користи и P2P и SIP базираниот начин на барање на корисник. Кога јазолот стартува и корисникот го внесува своето ID, јазолот ја пронаоѓа адресата на SIP серверот користејќи DNS и испраќа SIP REGISTER порака како што е покажано на Сликата 7.15.

Ако SIP регистрацијата е успешна јазолот ќе биде достапен со употреба на стандардниот SIP механизам. Исто така, јазолот се обидува да открие можни супер јазли за да се придружи на P2P системот:

- Мултикаст со многу мало "време на живот" (Time To Live – TTL) може да се користи за откривање на peer-ови и за добивање на повеќе информации од супер јазолот за овие peer-ови. Ова мултикаст- базирано откривање на јазол може да резултира во многу дисконектирани DHT компоненти. За да се спречи оваа појава, само постоечките DHT јазли (супер јазли) одговораат на мултикаст барањата.

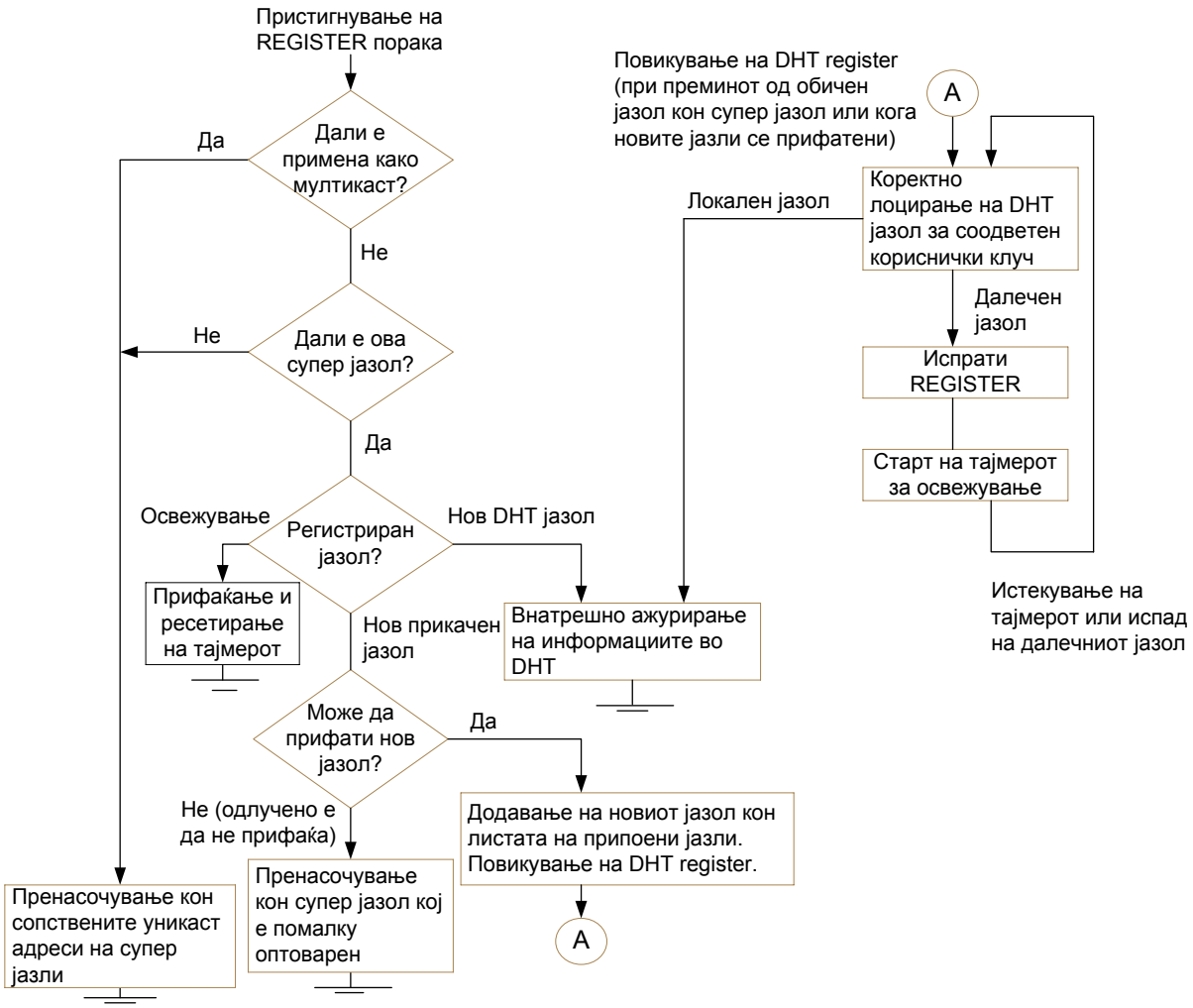
- Ако peer-адресите се кешираат, тогаш повеќе информации за супер јазолот може да се добијат од оние peer-ови за кои се претпоставува дека се уште се активни и не ја смениле локацијата при последната проверка.



Слика 7.15 Стартување на јазол и излезна регистрација

Информацијата за супер јазолот се кешира за следните регистрации кога корисникот повторно се логира или одјавува. Ова значи дека откривањето на јазол е само прашање на време, освен ако сите кеширани супер јазли не ја смениле положбата или исчезнале.

Кога корисникот детектира множество на супер јазли, тој избира два и им испраќа SIP REGISTER пораки за да се регистрира со нив. Двата супер јазли се користат за редундантност. Request-URI кореспондира со адресата на супер јазолот.



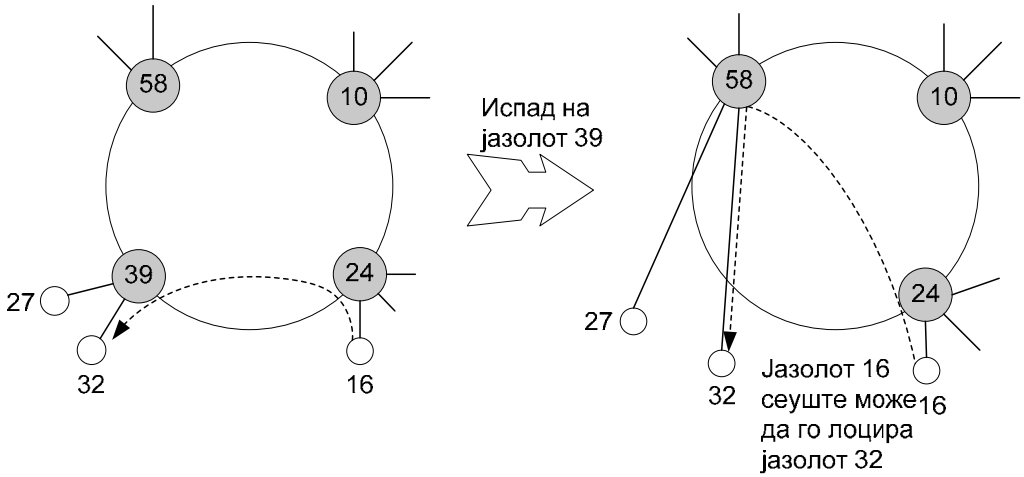
Слика 7.16 Влезна регистрација

Обичниот јазол е само SIP кориснички агент, додека супер јазолот служи и како SIP кориснички агент и како регистер за другите јазли. Супер јазолот испраќа REGISTER пораки во име на прикачените јазли кон дестинационите супер јазли во DHT. Исто така тој зема активно учество при барањето на корисник заедно со останатите јазли во DHT.

Обичните јазли периодично испраќаат освежувачки REGISTER пораки кои помагаат да се детектира секој испад на супер јазолот. Супер јазлите можат периодично да испраќаат меѓу себе SIP OPTIONS пораки. Интервалот на освежување, т.е. интервалот меѓу две освежувачки REGISTER пораки се прилагодува врз основа на оптовареноста на системот. Заради тоа OPTIONS пораката не се испраќа ако јазолот комуницирал со дестинацискиот јазол во претходниот интервал.

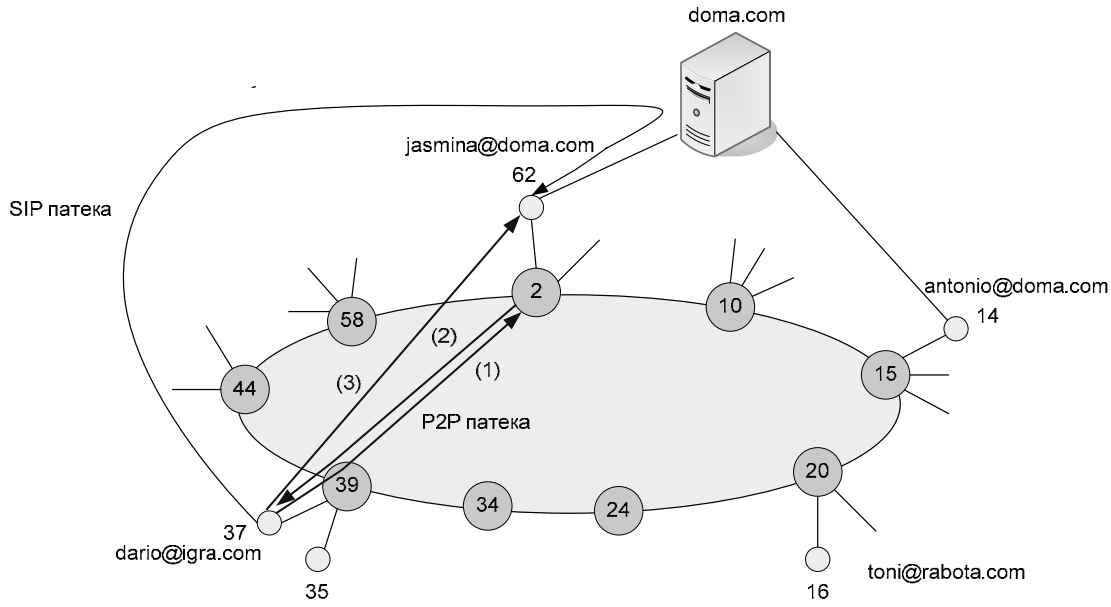
Кога еден обичен јазол прима REGISTER порака, тој испраќа SIP пренасочувачки одговор за да го пренасочи праќачот кон сопствените супер јазли како што е покажано

на Сликата 7.16. Кога супер јазолот ја прима REGISTER пораката и праќачот е дел од неговите прикачени јазли, тој ја препраќа пораката кон соодветните јазли во DHT како моментален кориснички клуч на праќачот. Ако праќачот не е дел од јазлите кои се прикачени на супер јазолот, самиот супер јазол самостојно одлучува дали да го прими или отфрли новиот јазол. Ако евентуално одлучи да го отфрли, тој ја пренасочува пораката кон некои други јазли кои се помалку оптоварени од конкретниот јазол.



Слика 7.17 Испад на DHT супер јазол

Кога еден супер јазол го напушта системот, тогаш треба да се изврши информирање на обичните прикачени јазли како и на соседните супер јазли. Во случај на исклучување на супер јазол, корисничките записи што ги поседува ги пренесува на други јазли во P2P мрежата. Ова гарантира дека другите корисници можат да лоцираат запис кога DHT јазолот е исклучен. Тој испраќа SIP REGISTER порака кон DHT јазлите кои ќе ги поседуваат корисничките записи откако овој јазол ќе се исклучи. При тоа нема потреба да се информираат прикачените обични јазли. Прикачените јазли ќе го детектираат испадот при следното освежување на регистрацијата и ќе се обидат да го пронајдат и да се конектираат на другиот супер јазол кој го поседува соодветниот запис.



Слика 7.18 Локација на корисник и воспоставување на повик

P2P пребарувањето е покажано на Слика 7.18. При P2P пребарување, еден обичен јазол испраќа INVITE или MESSAGE кон супер јазолот на кој е прикачен. Супер јазолот го лоцира дестинацискиот јазол со помош на клучот во DHT. Откако мапирањето е извршено, тој може или да ја замени или да ја пренасочи пораката. Пренасочувањето е попогоден начин бидејќи со него се зема супер јазол надвор од јамката на повикот. Сепак, во некои случаи како тие што вклучуваат firewall и NAT, замената е единствена опција.

7.7 Дискусија

Телефонијата е најзначајниот телекомуникациски сервиси и по својата природа е уникатен бидејќи ја овозможува конверзацијата, односно пренос на говор во реално време со посредство на технологија, каде што говорот е природниот начин на комуникација меѓу луѓето. Целта на телефонијата е да обезбеди чувство кај соговорниците (два или повеќе) при разговор на оддалеченост како истите да се наоѓаат еден до друг. Клучен параметар за тоа да се обезбеди е квалитетот на сервисот, каде што доцнењето од крај до крај и џитерот се најважните параметри, но исто така е потребен гарантиран битски проток и ограничени загуби на говорните информации. Телефонијата обезбедена преку Интернет протоколниот стек се нарекува Интернет

телефонија или VoIP. Притоа, за телефонијата е потребно да се обезбеди пренос во двете насоки меѓу два корисника (од едниот кон другиот корисник, и обратно), а за таа цел да се реализира потребно е прво да се обезбеди воспоставувањето на повикот преку соодветна сигнализација во Интернет (со SIP или H.323). Сигнализацијата не е потребна за клиент-сервер комуникацијата кај електронската пошта или кај веб сервисот, но е неопходна кај VoIP. Тоа е така затоа што кај клиент-сервер комуникацијата во Интернет клиентот секогаш побарува нешто од серверот и го добива како одговор на барањето. Кај VoIP е потребно да се пронајде бараниот корисник и да се испорача повикот до него, да се одржува повикот и на крајот да се терминира. За таа цел сигнализацијата која се наметнува како стандард во Интернет на подолги патеки е SIP. Истата може да се користи и за операторски VoIP, како и за peer-to-peer VoIP.

Операторскиот VoIP (иако ова може да не е општо прифатен термин) е телефонија понудена од одреден телеком оператор кој е всушност во тој случај телефонски сервис провајдер. Притоа операторот ги гарантира перформансите за говорот преку IP, кои не треба да бидат полоши од оние што се понудени во традиционалните телефонски мрежи базирани на комутација на канали (PSTN) кои пак обезбедуваат одличен квалитет за телефонијата, кон која всушност се стреми VoIP како првична цел (посебно во однос на доцнењето од крај до крај). Секако, VoIP не може да има мало доцнење како кај PSTN, бидејќи доцнењето заради кодирање/декодирање, сместување на говорот во пакети и негова репродукција на крајот, баферирањето во попатните мрежни јазли и во терминалните уреди итн., но доцнењето во IP средина сепак може да се намали на прифатливо ниво (под 400 msec од крај до крај во секоја од двете насоки) со соодветна архитектура за VoIP мрежата и соодветни решенија за поддршка на потребниот квалитет на сервисот во IP средина, како и стандардизирана сигнализација (SIP).

Од друга страна, peer-to-peer VoIP, барем во време на пишување на оваа книга, е целосно базиран на изворниот best-effort принцип на кој иницијално е и изграден Интернет, а тоа е прифаќање на секој повик без притоа да се води сметка дали потребниот квалитет на сервисот може да биде обезбеден или не (нешто што зависи од тековниот интензитет на Интернет сообраќајот на линковите по кои минува и говорниот сообраќај). Во такви услови, P2P VoIP може да биде корисен тогаш кога имаме доволно "слободен" битски проток за да се има прифатлив квалитет од аспект на корисникот. Најпознатиот P2P VoIP е Skype, заради што акцентот на анализата на P2P VoIP во оваа глава беше ставен токму на Skype. Притоа, ваквите сервиси (како што е

Skype) се целосно базирани на best-effort принципот (немаат никакви гаранции за квалитетот на сервисот). Но, во телекомуникациите е клучен принципот на трговија цена - перформанси, па така ваквите сервиси (P2P VoIP, како што е Skype) наоѓаат своја употреба, посебно во меѓународниот телефонски сообраќај заради значително ниските цени на услугите кои пак потекнуваат од значително помалите трошоци на P2P VoIP сервис провајдерите заради неимплементирањето на механизми за обезбедување на квалитет на сервисот.

Во секој случај, едно е сигурно, а тоа е дека иднината на телефонијата и глобално и локално е VoIP т.е. телефонија преку Интернет. Тој процес е веќе започнат и зема се поголем замав, така што традиционалната телефонија ќе биде постапно заменета со операторски VoIP сервис (ќе продолжи да егзистира и P2P VoIP) за на крајот да остане само VoIP како телефонски сервис, односно пренос на говорот преку Интернет како единствена мрежа за сите телекомуникациски сервиси во иднина.

Глава 8

Управување во Интернет

8.1. Вовед

Управување во Интернет подразбира одржување и надгледување на мрежните јазли, контрола на рутирање, администрирање и сл. Но, управувањето во Интернет се разликува од управувањето во хомогени мрежи (на пример: во телефонски мрежи), бидејќи Интернет опфаќа хетерогени физички мрежи и уреди (рутери, модеми, работни станици, сервери, принтери итн.).

Различните машини кои се контролираат може да бидат во различни мрежи, од што следува дека управувањето во Интернет треба да биде со протокол кој ќе овозможи комуникација од крај до крај преку Интернет што може да се постигне само со протокол на апликативно ниво (со користење на TCP/IP транспортните протоколи), [36]. Пристапот за управување на апликативно ниво за протокол за управување има и недостатоци, на пример:

- Ако е оштетена рутирачката табела нема да може да се пристапи до некој рутер
- Ако е оштетен оперативниот систем нема да може да работи апликацијата за управување

TCP/IP мрежното управување се дели на два дела стандарди, [37]:

- **за мрежно управување** (**SNMP**-Simple Network Management Protocol)
- **за управуваната информација** (**MIB**-Management Information)

Основен протокол за управување во Интернет е SNMP (Simple Network Management Protocol). Во основа SNMP претставува збир на прости операции, како и начин на претставување на информациите кои овие операции ги обработуваат. Тој му овозможува на администраторот навремено да интервенира кога некој уред, кој го поддржува SNMP протоколот, ќе има било каков проблем. На пример со помош на SNMP може да се исклучи интерфејс на некој рутер, или да се провери брзината на сообраќајот на некој Ethernet интерфејс. Со овој протокол дури може и да се надгледува температурата на некој мрежен уред и да се прати известување доколку таа надмине одредени граници.

Иако SNMP претставува наследник на SGMP (Simple Gateway Management Protocol) кој е наменет исклучиво за мониторинг на рутери, тој може да се користи и за мониторинг на секаков тип на уреди: UNIX и Windows системи, принтери, модеми, системи за напојување и др. Односно секој уред кој го подржува овој протокол, може да се менаџира, при што тоа не мора да бидат физички уреди туку може да бидат и софтверски алатки и компоненти, како што се сите видови на сервери (Web, DNS, Mail) како и сите видови бази на податоци.

Друг поглед на мрежниот менаџмент е мрежниот мониторинг, односно мониторинг на целата мрежа составена од рутери, сервери, хостови и друга опрема. RMON - далечински мониторинг на мрежи (Remote Network Monitoring) е развиено за да може да се види како функционира целокупната мрежа и како поодделни уреди во мрежата делуваат на целокупната мрежа.

8.1.1 Историјата и развојот на SNMP

Организација која го создала и се грижи за развојот на SNMP протоколот се нарекува IETF (Internet Engineering Task Force). Историски гледано IETF ги има предложено и развивано следниве RFC-а поврзани со SNMP протоколот:

- SNMPv1 првата стандардизирана верзија на SNMP протоколот. Оваа верзија е дефинирана во RFC 1157, [38], кој е целосно IETF стандард (STD 15, т.е. IETF стандард број 15). Сигурноста што е воведена во првата верзија е со помош на клучеви (community strings) кои не се ништо друго туку password-и кои овозможуваат апликациите кои ги знаат овие зборови да пристапат до SNMP

информациите на уредот. Постојат три типови на пристап (password-и) дефинирани со SNMPv1: read-only, read-write, trap.

- SNMPv2 е дефинирана во RFC 1901 (се води како RFC во експериментална фаза), 1905-1909, како и RFC 2578-2580 и никогаш не станала официјален IETF стандард, [23]. И покрај оваа овој протокол е широко распространет и се користи од многу производители.
- SNMPv3 е наредната верзија на овој протокол, која постои тековно како IETF стандард дефиниран со RFC 3411-3418 (STD 62, т.е. IETF стандард број 62), [1]. Во основа овој протокол не нуди ништо ново од аспект на менаџирањето на мрежата туку само ја подобрува сигурноста, затоа што со претходните верзии до било кој уред се пристапува со некодиран password, кој може да биде пресретнат од страна на несакани субјекти во мрежата.

Значи може да се заклучи дека SNMP протоколот е стандардизиран и го имплементираат скоро сите поголеми производители со цел да се овозможи подобро и посигурно менаџирана мрежа.

8.1.2 SNMP агенти и менаџери

Во светот на SNMP постојат два типа на ентитети кои можат да функционираат на мрежните јазли: менаџери (managers) и агенти (agents). Менаџери всушност се сервери на кои оперира некој менаџмент софтвер кој може да управува и ја набљудува мрежата. Менаџерите уште и се нарекуваат „Станици за менаџирање на мрежа“ – Network Management Stations (NMSs). NMS всушност е одговорна за примање на информации од агентите во мрежата, односно за извлекување (polling) информации од агентите и примање на аларми (traps) од агентите во мрежата. Извлечените информации подоцна се дообработуваат и се користат за да се набљудува состојбата на мрежата. Алармот всушност е начин како агентот да ја извести менаџмент станицата дека нешто се случило, променило во состојбата на мрежниот јазел. Алармите се асинхрони настани кои се праќаат кога ќе се случи нешто. Така на пример ако некој интерфејс на рутерот падне, рутерот праќа аларм до менаџмент станицата (NMS) дека тој интерфејс е паднат.

За разлика од NMS, агент е софтвер кој функционира на мрежните уреди што се менаџирани. Тоа може да биде посебен процес кој функционира на некој сложен сервер кој работи со некој оперативен систем (Linux, Windows), или може да биде интегриран во оперативниот систем како на пример оперативниот систем на Cisco рутерите (IOS) или на некој UPS апарат. Всушност во поново време сите производители ги интегрираат SNMP агентите во оперативните системи на нивните уреди со што многу повеќе се олеснува одржувањето на мрежата. Агентот всушност го набљудува самиот мрежен уред и ја снабдува менаџмент станицата со информации за неговата состојба. На пример агентот може да ги набљудува сите интерфејси на еден рутер и да праќа аларми до менаџмент станицата доколку некој од нив падне, исто така може да прати известување и кога состојбата ќе се врати во нормала. Исто така самата станица може да праќа и барања (queries) до SNMP агентот за состојбата на некои параметри на мрежниот уред. Треба да се забележи дека барањата и алармите може да се случат истовремено и дека тие не се корелирани.

8.1.3 Што се SMI и MIB?

Структурата на менаџмент информацијата (Structure of Management Information – SMI) дефинира начин на претставување на набљудуваните елементи во уредот од страна на SNMP агентите. Значи секој агент има листа на објекти што ги набљудува во самиот уред, како на пример температура, состојба на интерфејс и др. Оваа листа на објекти овозможува менаџмент станицата да ја процени генералната состојба на уредот, дали тој функционира добро и што е расипано.

Оваа листа на набљудувани објекти може да се третира како база на податоци на менаџирани објекти (Management Information Base –MIB). Секоја моментална или статистичка информација која може да биде набљудувана од менаџмент станицата е дефинирана во оваа база на објекти односно во MIB базата. Значи SMI го дефинира начинот на кој се дефинираат менаџираните објекти, додека MIB е самата дефиниција на објектот. Во секој агент може да се имплементираат многу MIB објекти меѓутоа во секој агент е имплементирана барем MIB-II (RFC 1213) базата. Оваа стандардна база ги интегрира основните TCP/IP објекти, за да овозможи менаџирање на основните TCP/IP функционалност (како на пример брзина на интерфејсот, примени октети, загубени

пакети итн.). Меѓутоа оваа база не ги содржи сите можни објекти што би сакал одреден производител на опрема да ги направи достапни за менаџирање, па затоа постојат и многу други останати MIB бази специфични за производителот и за делови од неговата опрема. Постојат и MIB бази дефинирани за одредени протоколи како на пример:

- ATM MIB (RFC 2515)
- BGP Version 4 MIB (RFC 2619)
- RDBMS MIB (RFC 1697)
- Mail Monitoring MIB (RFC 2249)
- FNS Server MIB (RFC 1611) и многу други.

Значи сите производители дури и поединци може да дефинираат MIB променливи за нивните потреби. Зошто е ова вака може да се илустрира со едноставен пример. Така, на пример, ако некој производител исфрла нов производ на пазарот, на пример нов рутер со некоја нова технологија која не е дефинирана во стандардните MIB бази тој ќе сака да може да обезбедува до менаџмент станицата информации карактеристични и за новата технологија која ја имплементирал во неговиот производ. Доколку ова не е можно, не е можно и да се менаџираат новитетите што ги нуди оваа технологија.

8.1.4 Менаџмент на сервери

Менаџирањето на ресурсите на северите (простор на дискот, зафатеност на меморија и др.) претставува важен дел од мрежниот менаџмент. Во денешно време се губи границата помеѓу менаџментот на мрежата и систем администрацијата. Затоа и е дефинирана MIB база (RFC 2790), која содржи објекти за основните ресурси на Windows и Unix оперативните системи. Во неа се содржани објекти кои прикажуваат основните ресурси значајни за еден сервер: простор на тврдиот диск, број на корисници, број на активни процеси, инсталиран софтвер, искористеност на работната меморија и др.

8.1.5 Далечинско управување - Remote Monitoring (RMON)

Постојат две верзии на протоколот за далечинско набљудување: RMONv1, RMONv2 кои се дефинирани со RFC препораките 2819 и 2021 соодветно. RMONv1, всушност го имплементира набљудувањето на мрежата на второ ниво од OSI моделот, LAN и WAN, додека RMONv2 е надградба која вклучува и набљудување и на повисоките нивоа како што се мрежното и апликациското. RMON всушност преставува еден вид на апликација која прибира статистички информации со тек на времето за да не мора една NMS станица константно да праќа барања до уредот и да го загушува мрежниот сообраќај. Врз база на овие податоци во самата апликација може да се дефинираат threshold (гранични) вредности кои ако се надминат самата апликација праќа аларм до NMS станицата.

8.2 Опис на SNMP протоколот

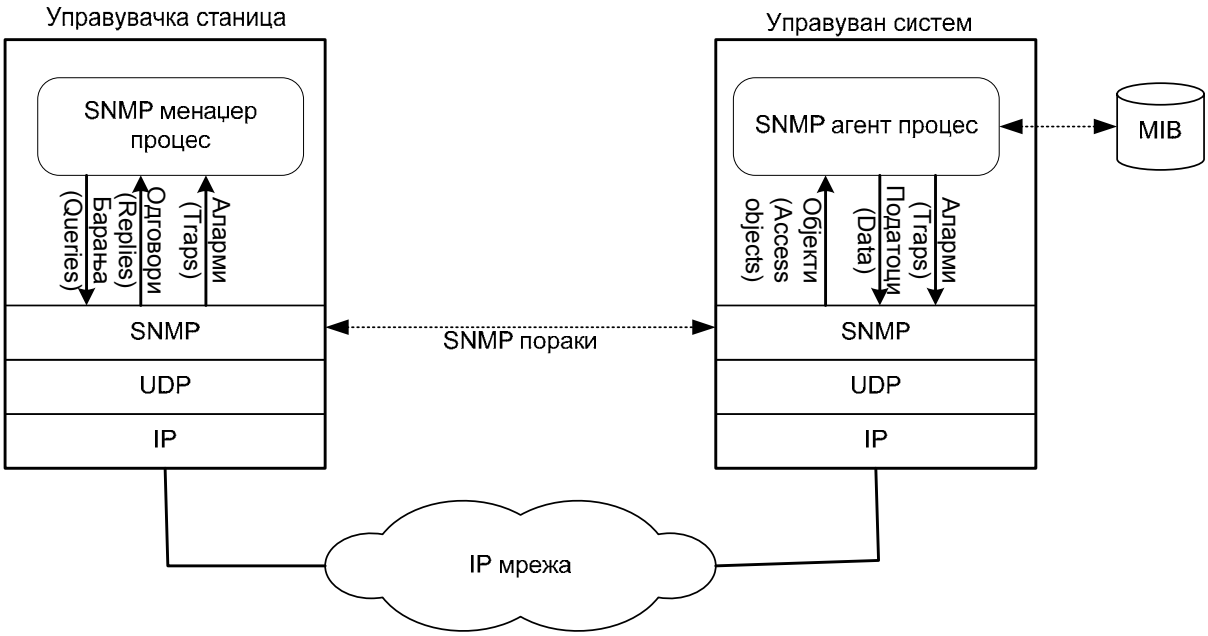
SNMP го користи UDP (User Datagram Protocol) протоколот како транспортен протокол за пренос на податоци меѓу менаџерите и агентите. Избран е UDP протоколот пред TCP (Transmission Control Protocol) иако тој е неконекциски ориентиран и недоверлив протокол каде информацијата се пакува во датаграми. Значи самата SNMP апликација треба да одлучи дали одреден датаграм пристигнал до потребната дестинација и да го препрати по потреба. Ова е остварено со поставување на одреден бројач на време и доколку помине тоа време а NMS станицата не добила повратен одговор од агентот таа го повторува барањето. Исто така може да се дефинира и бројот на ретрансмисии, за да не трае овој процес бесконечно долго.

Користењето на UDP протоколот при стандардното барање на информации од агентите не е проблем затоа што менаџмент станицата доколку не добие одговор ќе прати ново барање. Меѓутоа кога агентот праќа аларм, доколку тој не стигне до менаџмент станицата таа нема начин да знае за тоа, а исто така и агентот не знае дека менаџмент станицата не го примила алармот.

И покрај овие недостатоци избран е UDP протоколот поради малиот overhead што го користи и користи помали мрежни ресурси, односно не ја загушува мрежата. Иако постојат некои SNMP апликации кои го користат TCP протоколот со текот на

времето се покажале како непрактични. SNMP е создаден за да се набљудуваат мрежи во кои се јавуваат проблеми. Значи ако мрежата е крајно оптоварена подобро е да се користи протокол кој само ќе прати информација кога има проблем отколку да се труди да стигне информацијата по секоја цена вршејќи ретрансмисии со што уште повеќе ќе ја загуши мрежата.

SNMP ја користи портата 161 за праќање барања и примање одговори, додека портата 162 за примање на аларми од агентите на набљудуваните уреди.



Слика 8.1 Размена на информации во SNMP

Како всушност се одвива размената на информациите меѓу SNMP менаџерите и агентите низ TCP/IP протоколниот стек е прикажано на слика 8.1. Реализирањето на SNMP комуникацијата низ протоколните нивоа е следна:

- Апликациско ниво

  Најпрво самата SNMP апликација (во NMS) одлучува дали ќе прати SNMP барање до некој агент, или агентот ќе прати одговор на некое барање или аларм до NMS станицата и го предава на транспортното ниво.

- Транспортно ниво (UDP)

  Ова ниво овозможува два система да комуницираат меѓусебно. UDP заглавието меѓудругото го содржи и бројот на портата која ја користи апликацијата за

предавање на информациите на транспортното ниво, во случајов тоа се портите 161 и 162 во зависност од типот на информацијата.

- Мрежно ниво (IP)

  Ова ниво е должно да ги достави SNMP пакетите до пратената дестинација според нивната IP адреса.

- Податочното ниво (MAC)

  Последното што треба да се направи е SNMP пакетот да ја достигне саканата дестинација. Тоа е всушност задачата на ова ниво кое е должно пакетите да ги адаптира и прати на самиот физички медиум, а потоа на другата страни да ги прими и правилно проследи на повисоките нивоа.

8.2.1 SNMP клучеви

SNMPv1 и SNMPv2 употребуваат клучеви (communities) за комуникација помеѓу менаџерите и агентите. Секој агент има дефинирано три типа на клучеви: за читање (read-only), за читање и запишување (read-write), и за аларми (trap). Всушност секој клуч е еден вид на лозинка (password) за да се пристапи до агентот на уредот. Секој клуч овозможува различни активности кои може да се превземаат на уредот. Така на пример со клучот за читање може само да се читаат вредности меѓутоа не може да се променат. Додека со клучот за читање и запишување може да се читаат вредности, да се ресетираат бројачите, дури и да се менува конфигурацијата на уредот. Клучот пак за аларми овозможува примање на аларми. Овие клучеви при купување на нова опрема обично се дефинирани како на пример public за клучот за читање и private за клучот за читање и запишување. Меѓутоа е пожелно тие да се променат за да се осигури приватноста на мрежата. Кога се конфигурираат SNMP агентите важно е да се дефинира и дестинацијата (адресата) на кое ќе се праќаат алармите. Понекогаш е добро да се конфигурира агентот да праќа аларми кога некој сака да му пристапи на уредот со погрешен клуч. Со ова може да се утврди кога некој натрапник сака да му пристапи на уредот.

Бидејќи самите клучеви се еден вид на лозинки тие треба да се поставуваат според истите правила како и лозинки: да не бидат зборови од речник, имиња на личности туку е препорачливо да бидат комбинации од големи и мали букви и бројки.

Проблемот со SNMP клучевите е што тие се праќаат како обичен текст и некој може да ги пресретне на мрежата, затоа е и развиен SNMPv3 протоколот кој употребува сигурносна автентикација и комуникација помеѓу SNMP уредите. Иако во денешно време може да се обезбеди некоја сигурност со помош на firewall-и, access листи на рутерите и VPN мрежи. Така на пример самиот firewall може да се конфигурира да пропушта UDP сообраќај на порта 162 од агентите кон NMS станицата и на порта 161 од агентите до станицата.

8.3 Структура на менаџмент информација (SMI)

При менаџирање на некој уред најважно е да се знае какви информации може тој уред да прати, и како таа информација се презентира во SNMP. Ова всушност е дефинирано со RFC 1155, односно со SMIv1 (Structure of Management Information Version 1). SMIv1 прецизно дефинира како се нарекуваат SNMP објектите кои може да се набљудуваат, и го специфицира типот на информацијата. Подоцна се појавува и SMIv2 (RFC 2578, [39]) која всушност е надградба на SMIv1 и одговара на SNMPv2 протоколот. За секој објект во SMIv1 се дефинирани три атрибути:

- **Име**

  Името или идентификаторот на објектот (object identifier – OID), еднозначно го дефинира објектот. Имињата се појавуваат во два формата: бројчан и разбирлив за човекот. И во двата случаја имињата се премногу долги и незгодни за користење.

- **Тип на објектот и синтакса**

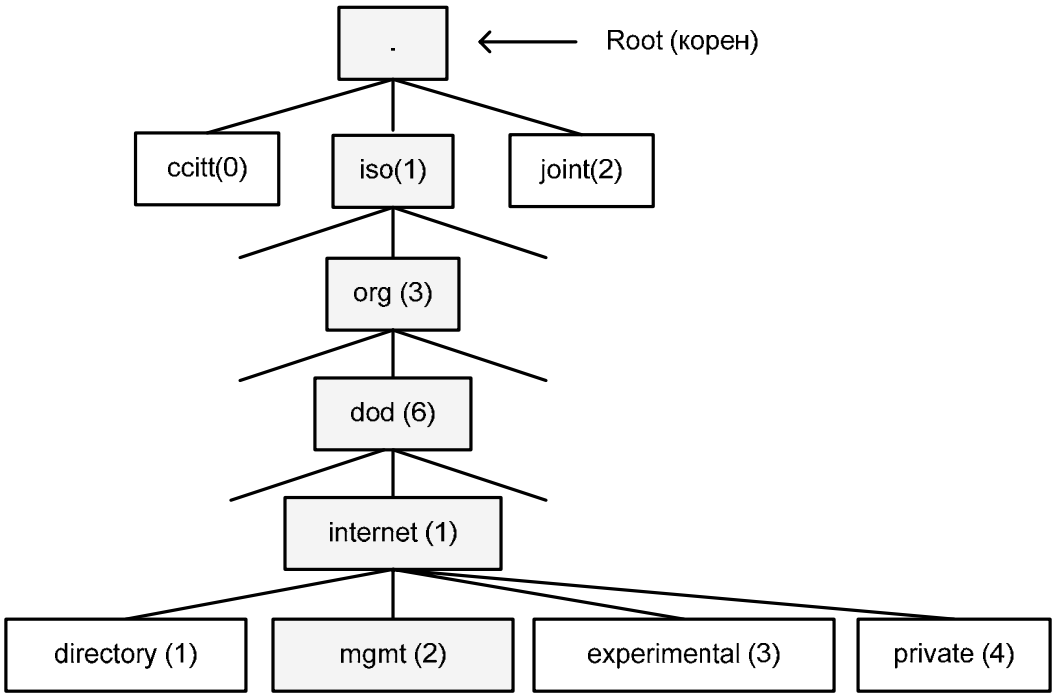
  Типот на податок што го содржи еден објект е дефиниран според ASN.1 стандардот. Тоа е стандард кој дефинира како се пренесуваат и интерпретираат податоците помеѓу менаџерите и агентите. Најважно е што овој стандард не зависи од типот на хардверот и оперативниот систем, така што сите уреди може лесно да комуницираат меѓу себе.

- **Кодирање**

  Вредностите на објектите кога се кодираат во октети со помош на BER (Basic Encoding Rules). Всушност со BER е дефинирано како објектите се кодираат и

декодираат за да може да се пратат преку транспортниот медиум како на пример некоја Ethernet мрежа.

При именувањето SNMP објектите се организирани во нивовска структура во вид на стебло како што е прикажано на Сликата 8.2. Всушност името на објектот (ID) се состои од серија на броеви, дефинирани како јазли на стеблото, одделени меѓу себе со точки. Исто така постои дуалност во именувањето за да биде името поразбирливо за човекот. Така за секоја бројка постои збор кој поточно го опишува стеблото. Во практиката се користат и двата начини на именување.



Слика 8.2 Основно SMIv1 стебло

Како што се гледа од Слика 8.2 јазелот на врвот од стеблото се нарекува корен (root) јазел, сите останати јазли што имаат подјазли се нарекуваат подстебло (subtree) или гранка (branch) јазли, а останатите кои се краеви на стеблото се нарекуваат јазли листови (leaf nodes). Најголема важност за мрежниот менаџмент има подстеблото iso(1).org(3).dod(6).internet(1), оваа патека репрезентирана во форма на OID е 1.3.6.1 или iso.org.dod.internet. Секој објект всушност има нумерички OID, на кој му одговара соодветен текстуален формат како во претходниот пример. Од гранките на internet јазелот најмногу се користат mgmt(2), во кој се дефинирани основните SNMP менаџмент објекти, и private(4), во кој се дефинирани специфичните објекти на секој

производител. Од останатите два directory(1) не се користи додека experimental(4) се користи за истражување.

Табела 8.1 SNMPv1 типови податоци

Тип на податок	Опис
INTEGER	32 битен број кој може да претставува било каков податок
OCTET STRING	Низа од 0 или повеќе октети (бајти) кој се користи претежно за прикажување на текст и некои физички адреси
Counter	32 битен бројач со минимална вредност 0 и максимална вредност од 2^{32}-1 (4 294 967 295). Неговата вредност само може да се зголемува и да се ресетира на 0, што се случува автоматски кога ќе се достигне максималната вредност.
OBJECT IDENTIFIER	Низа од броеви разделени со точки која претставува специфичен OID објект
NULL	Не се користи
SEQUENCE	Листа што содржи ASN.1 типови на податоци
SEQUENCE OF	Објект составен од секвенца од ASN.1 типови
IpAddress	32 битна IPv4 адреса
NetworkAddress	Исто како IpAddress но може да претстави различни типови на мрежни адреси
Gauge	32 битен бројач со минимална вредност 0 и максимална вредност од 2^{32}-1 (4 294 967 295). За разлика од Counter вредноста може и да му се зголемува и намалува, но никогаш не може да ја достигне максималната вредност
TimeTicks	32 битен број со минимална вредност 0 и максимална вредност од 2^{32}-1 (4 294 967 295). Со него се мери време изразено во секунди
Opaque	Овозможува еден бајт да биде кодиран со било кој ASN.1 тип на кодирање

Постои специјална институција IANA (Internet Assigned Numbers Authority) која се грижи за доделување на OID броеви во private(4), гранката на претпријатија, личности, институции, организации и др. Како пример може да се наведе компанијата

Cisco Systems на која и е доделен во enterprises бројот 9. Значи целиот OID на Cisco е iso.org.dod.internet.private.enterprises.cisco, односно 1.3.6.1.4.1.9. На Cisco му е дозволено да прави што сака со оваа гранка, односно си дефинира свои сопствени подгранки и јазли. Со ова се овозможува поголем број на информации да може да се набљудуваат во еден Cisco уред. Меѓутоа не само компаниите може да имаат свои OID броеви, секој поединец може бесплатно да добие свој сопствен број под private.enterprises со праќање на барање до IANA.

Една од најважните работи при дефинирање на OID објектите е од кој тип е информацијата содржана во нив, односно дали е број, опис, бројач, адреса и др. Типот на променливи кои се подржани во SMIv1 односноно со протоколот SNMPv1 се дадени во Табелата 8.1.

Како што е веќе кажано MIB претставува збир на дефинирани објекти. Значи во секој MIB се дефинирани самите објекти, нивниот OID, типот на податокот на самиот објект и неговите карактеристики. Изгледот на дефиницијата на еден објект е:

```
<name> OBJECT-TYPE
    SYNTAX <datatype>
    ACCESS <either read-only, read-write, write-only, or not-accessible>
    STATUS <either mandatory, optional, or obsolete>
    DESCRIPTION
        "Textual description describing this particular managed object."
    ::= { <Unique OID that defines this object> }
```

Од тука се гледа како се дефинираат параметрите што се од значење за еден SMIv1објект дефиниран во некој MIB. Попросто речено може да се каже дека MIB претставува збир на дефиниции за одредени објекти, кои ги дефинира некоја фирма, поединец или институција за да може да се пристапи до тие објекти преку агентот на некој уред.

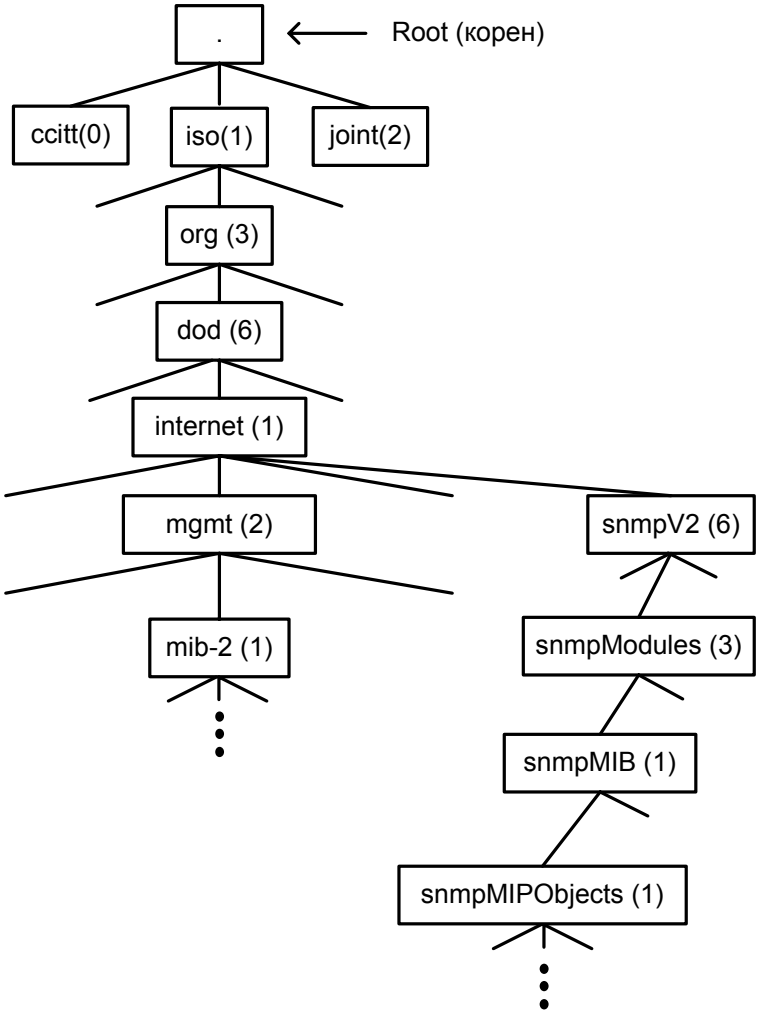
8.3.1 Проширување на SMIv1 во SMIv2

Со појавувањето на SNMPv2 дошло до потреба од проширување на OID дефинициите и можностите при нивното дефинирање во MIB базите. Така на Слика 8.3 може да се видат дополнувањата на SMI стеблото.

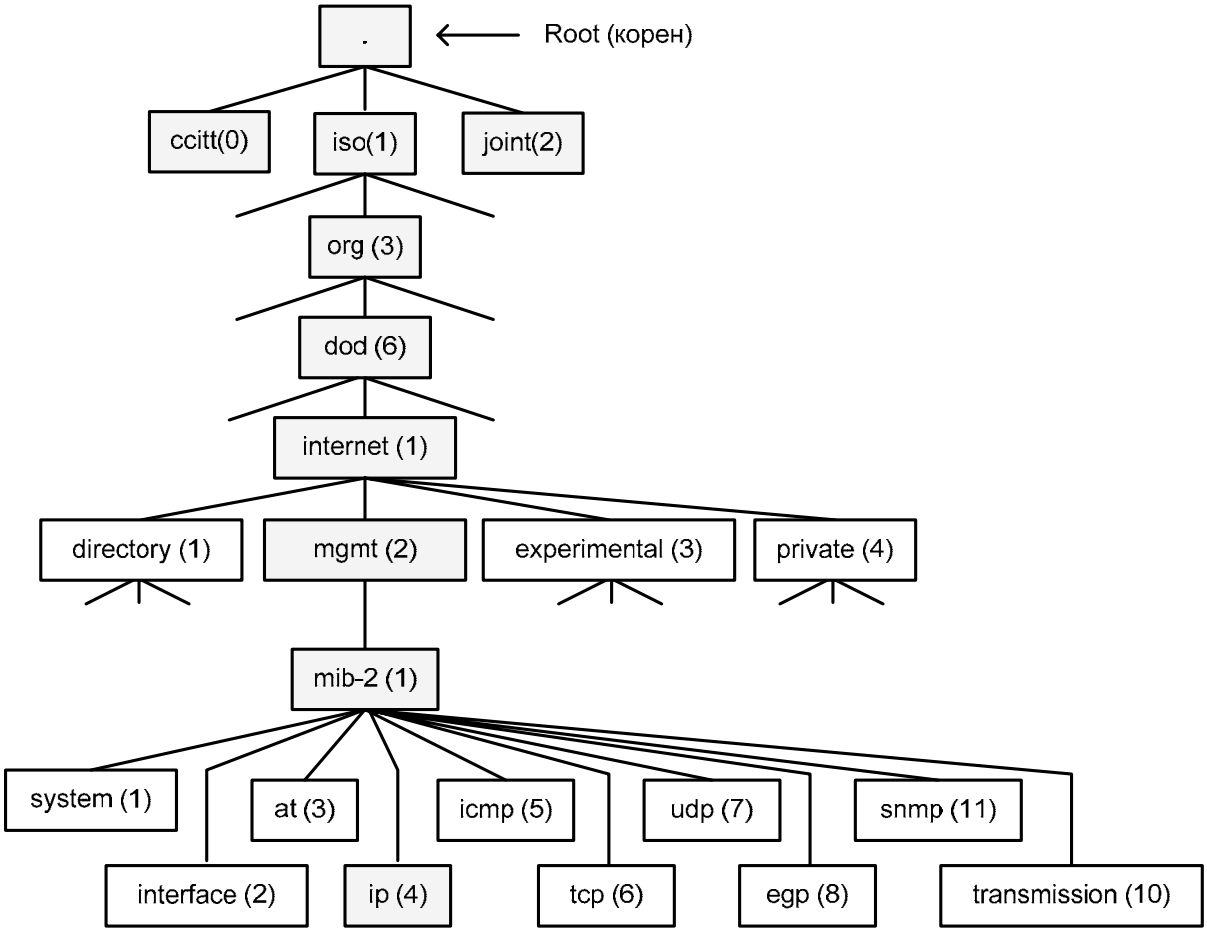
Од сликата 8.2 се гледа дека се додадени јазлите mib-2(1) како подјазел на mgmt(2) и snmpV2(6) подјазел на internet(1).

Јазелот Mib-2(1) што е подјазел на mgmt(2), претставува многу важен дел од SMI стеблото прикажано подетално на Слика 8.4.

Како што се гледа од Сликата 8.4 mib-2 јазелот има повеќе групи на објекти опишани во Табела 8.2.



Слика 8.3 Проширено SMIv2 стебло



Слика 8.4 SMIv2 стебло со посебен осврт на mib-2 гранката

Табела 8.2 Опис на гранките на mib-2 јазелот

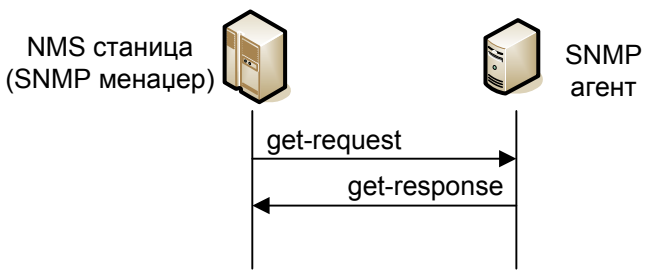
Име на подстебло	OID	Опис
system	1.3.6.1.2.1.1	За карактеристиките и состојбата на системот
interfaces	1.3.6.1.2.1.2	За карактеристиките и состојбата на интерфејсите
at	1.3.6.1.2.1.3	За преведување на адреси
ip	1.3.6.1.2.1.4	За IP карактеристиките и рутирачките табели
icmp	1.3.6.1.2.1.5	За ICMP карактеристиките и состојбата на уредот
tcp	1.3.6.1.2.1.6	За TCP конекциите и нивната состојба
udp	1.3.6.1.2.1.7	За UDP статистика
egp	1.3.6.1.2.1.8	За различни EGP статистики и табели
transmission	1.3.6.1.2.1.10	Се користи при дефинирање на други MIB бази
snmp	1.3.6.1.2.1.11	За перформансите и карактеристиките на SNMP протоколот на уредот

8.4 SNMP операции

До сега беше опишано како за SNMP се организирани информациите, но сеуште не е кажано ништо како тој функционира и како тие информации се добиваат од агентите и се праќаат до посакуваната дестинација. Постои посебен формат кој се користи за праќање на пораки меѓу SNMP агентите и менаџерите кој се нарекува PDU (Protocol Data Unit). Постои посебен PDU формат за секоја SNMP операција:

- Get
- Get-next
- Get-bulk (SNMPv2 и SNMPv3)
- Set
- Get-response
- Trap
- Notification (SNMPv2 и SNMPv3)
- Inform (SNMPv2 и SNMPv3)
- Report (SNMPv2 и SNMPv3)

Операцијата get е всушност барање што го праќа NMS станицата до агентите. Агентите кога ќе го примат ова барање се трудат да и одговорат на менаџмент станицата во најбрзо можно време, меѓутоа доколку уредот е преоптоварен тој едноставно може да го отфрли барањето. Ако агентот успее да ги прибере бараните информации од уредот тој праќа одговор (get-response) назад до NMS станицата каде потоа се процесираат. Оваа процедура е прикажана на Сликата 8.5.



Слика 8.5 Комуникација меѓу SNMP агент и NMS станица при извршување на get операција

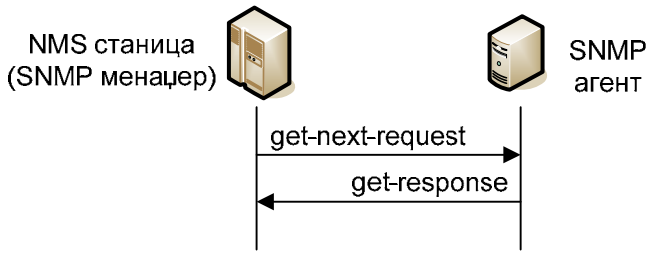
Како некој агент знае што побарува NMS станицата? Тоа се остварува така што во самото барање постои листа на побарани MIB објекти (varbind) што му овозможува на агентот да знае кои податоци треба да ги прати до менаџмент станицата. Секој објект е дефиниран со сопствен OID број кој всушност се праќа до агентот.

Основната функција што ја остварува командата get-request (прикажана на слика 8.6) е праќање на get барање до некој SNMP агент за да собере информации од некој објект. При тоа мора се дефинира уредот од кој се бараат информациите (на пример, cisco.ora.com) или неговата адреса, потоа клучот (public) како и OID идентификацијата на објектот и бројот на интерфејсот (.1.3.6.1.2.1.1.6.0). Значи од последните два броја .6.0, 6 го означува објектот чија информација се бара, додека 0 ја означува инстанцата за која се бара оваа информација. Ова е затоа што информацијата за секој објект дефиниран со сопствен OID број (.1.3.6.1.2.1.1.6) може да постои за различни инстанци. Така ако има повеќе интерфејси тогаш може сите да имаат своја IP адреса, па во барањето треба да се дефинира чија IP адреса е потребна. За објекти кои немаат повеќе инстанци, односно за скаларни објекти, вредноста на инстанцата е 0. Од тука може да се заклучи дека get барањето се користи за добивање на вредноста на единствен MIB објект.

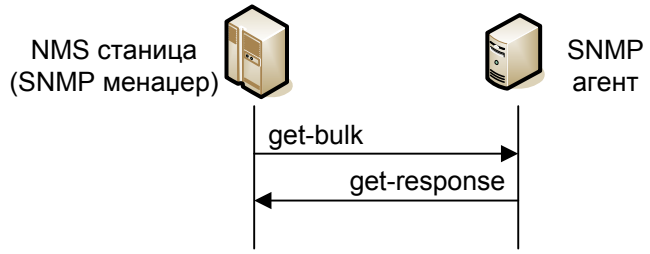
Понекогаш е потребно одеднаш да се побараат вредности на повеќе MIB објекти за ова се користи get-next операцијата. Со оваа операција се овозможува до агентот да се однесе барање за група на вредности. Со оваа команда се пребарува SMI стеблото јазел по јазел од коренот (root) кон другите гранки. Ова е лесно затоа што стеблото е хиерархиски подредено, односно OID вредноста на објектот е серија на бројни вредности хиерархиски подредени. Всушност get-next операцијата се извршува повеќе пати последователно се додека агентот нема повеќе објекти што може да ги прати. Оваа команда се заснова на принципот на лексикографско подредување на MIB објектите во SMI стеблото.

Како што се гледа од сликата за секој јазел од стеблото најпрво се проверува јазелот со најнизок доделен број. Така, се почнува од root, потоа се проверува ccitt, при што овој јазел нема гранки под него, па се продолжува понатаму. Потоа се проверува iso јазелот, па затоа што има подјазел потоа се проверува тој подјазол т.е. org. Процесот ќе продолжи се додека не се најде system јазелот. Ова е можно затоа што секој јазел во стеблото си има свое уникатно име. Кога ќе се најде system јазелот се продолжува понатаму и се проверуваат неговите подјазли, а на крајот и сите инстанци. Кога ќе се

дојде до краен јазел се праќа вредноста на објектот содржана во таа инстанца. Сите вредности се праќаат една по една.



Слика 8.6 SNMP get-next операција



Слика 8.7 SNMP get-bulk операција

Во SNMPv2 е дефинирана уште една операција get-bulk, слична со get-next, со тоа што во оваа команда може да се побараат вредности од повеќе тотално различни јазли, илустрирано со следниот пример со UNIX команда:

```
$ snmpbulkget -v2c -B 1 3 linux.ora.com public sysDescr ifInOctets
ifOutOctets
```

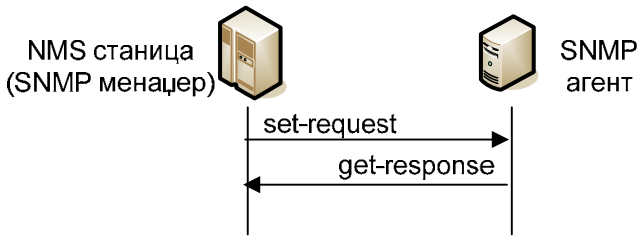
На горенаведената команда агентот го враќа следниот одговор:

```
system.sysDescr.0 = "Linux linux abc #1 Fri May 11 19:33:18 GMT+1 2012"
interfaces.ifTable.ifEntry.ifInOctets.1 = 70840
interfaces.ifTable.ifEntry.ifOutOctets.1 = 70840
interfaces.ifTable.ifEntry.ifInOctets.2 = 143548020
interfaces.ifTable.ifEntry.ifOutOctets.2 = 111725152
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
```

При што како што се гледа мора да се дефинира дека е SNMPv2 операција, бројот на објекти што се бараат со единствена get-next операција (1) како и

максималниот број на повторувања на get-next операцијата за останатите објекти (3). Комуникацијата меѓу агентите и менаџмент станицата е прикажана на Слика 8.7.

Понекогаш се појавува и потреба да може да се променат вредностите на MIB објектите или да се направи нова редица во некоја табела, за тоа се користи set операцијата. Со оваа команда може да се променат вредностите на MIB објектите што се дефинирани како read-write и write-only. Комуникацијата меѓу агентот и менаџерот е прикажана на Слика 8.8.



Слика 8.8 SNMP set операција

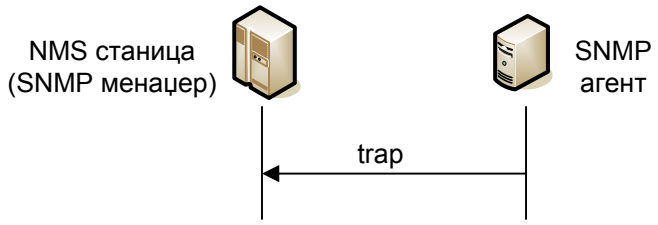
Иако комуникацијата на прв поглед изгледа исто како и кај другите операции, ова е правата операција со која се менува нешто во конфигурацијата на уредот. Со следните UNIX команди е илустрирано како работи set операцијата:

```
$ snmpget cisco.ora.com public system.sysLocation.0
system.sysLocation.0 = ""
$ snmpset cisco.ora.com private system.sysLocation.0 s "Skopje, MK"
system.sysLocation.0 = "Skopje, MK"
$ snmpget cisco.ora.com public system.sysLocation.0
system.sysLocation.0 = "Skopje, MK"
```

Како што се гледа дека по извршена set операција е сменета вредноста на објектот system.sysLocation на инстанца 0 во “Skopje, MK”.

8.4.1 SNMP аларми (Traps)

Алармот претставува начин со кој агентот праќа информација до NMS станицата дека нешто не во ред со уредот, односно дека нешто се случило, променило. Начинот на комуникација меѓу агентот и менаџмент станицата е прикажан на Слика 8.9.



Слика 8.9 SNMP trap операција

Од сликата се гледа дека алармот се генерира од страна на агентот и се праќа до одредиштето дефинирано во самиот агент, кое е најчесто IP адресата на самата NMS станица. Меѓутоа никогаш не се праќа потврда до агентот дека алармот стигнал до NMS станицата. Поради ова и поради тоа што за транспортен протокол се користи UDP, алармите се склони кон губење во мрежата. Меѓутоа ова не е толку голем проблем затоа што и доколку алармот не стигне до дестинацијата сепак агентот се потрудил да ја информира станицата дека има проблем, а проблемот ќе се увиди покасно.

Кога NMS станицата ќе прими аларм, таа треба да го преведе за да знае што всушност се случува. Алармот со ознака 6 е специјален аларм, кој е дефиниран од страна на самите производители или корисници, тој има специјална OID идентификација во MIB подстеблото: iso.org.dod.internet.private.enterprises. Алармот најчесто содржи големо количество на информации. Типот и податокот на секоја информација содржана во алармот е всушност вредност на некој MIB објект од SMI стеблото. Конечно може да се заклучи дека алармите се пораки кои се праќаат до NMS станицата и во себе содржат повеќе вредности на MIB објекти со цел да може NMS станицата полесно да го идентификува, а потоа и разреши проблемот.

Поради тоа што информациите во алармите и оние добиени со операциите set и get во SNMPv1 се со различен PDU формат, во SNMPv2 се дефинирани останати типови на аларми, како што се за известување (notification) исто како и trap во SNMPv1, информирање (inform) и извештај (report) алармите. Како што е кажано порано алармот за известување се праќа од агент до агент и ја има истата улога како и поимот аларм дефиниран во SNMPv1.

Вториот тип на аларми се алармите за информирање. Тие се всушност пораки што се праќаат помеѓу SNMP менаџмент станици. Овие аларми се користат кога во мрежата постојат повеќе NMS станици. За овие аларми се праќа известување назад до

праќачот доколку NMS станицата го прими. Значи испраќач може да биде и SNMP агент, кој се наоѓа на критичен уред во мрежата и има потреба да се потврди дека NMS станицата го примила алармот, ако не тој може да се препрати.

Третиот тип на аларми се алармите за праќање на извештаи помеѓу SNMP надгледуваните уреди. Иако овој аларм никогаш не заживеал, тој наоѓа своја примена во SNMPv3.

8.5 Дискусија

Управувањето во IP мрежа е толку поважен сегмент колку што е поголема мрежата, односно колку што има повеќе мрежни елементи, како што се рутери, комутатори, сервери, бази на податоци, итн. Кога има мал број на мрежни елементи (на пример, два или три елементи), тогаш управувањето на мрежата со стандардизирани решенија не е неопходно. Такви примери на мрежи се домашните IP мрежи кои се состојат вообичаено од еден рутер или комутатор со неколку уреди поврзани жично или безжично (компјутери, телефони, мобилни уреди итн.). Но, во секоја IP мрежа во која има повеќе десетици мрежни елементи или повеќе (со хардвер и различни оперативни системи и софтверски модули на нив), автоматизираното и централизирано управување станува неопходно за непрекинато функционирање на мрежата и сервисите кои се овозможуваат преку неа (на пример, сервисите базирани best-effort Интернет пристап, VoIP, мултимедија, сигнализација, пристап до веб сајтови, итн.).

Управувањето во Интернет се реализира со протоколот SNMP (Simple Network Management Protocol) кој претставува рамка за управување со уредите во Интернет мрежа со користење на TCP/IP протоколниот модел. Во SNMP централизиран менаџер (хост) ги контролира мрежните елементи, кои можат да бидат крајни хостови (сервери, бази на податоци) или мрежни уреди (рутери, комутатори), кои пак имаат имплементирани SNMP клиенти. За своето функционирање SNMP ги користи SMI (Structure of Management Information) и MIB (Management Information Base) за хиерархиско именување и групирање на објектите кои се предмет на управување од страна на SNMP. Во практика, клучниот елемент при управувањето со SNMP е имплементацијата на SNMP менаџерот чии функционалности во однос на управувањето (што ќе се управува и како) зависат и од намената на IP мрежата чии

елементи се управуваат (дали е фиксна или мобилна, јавна или корпорациска, дали е наменета за сервиси во реално време како VoIP и стриминг, или пак за сервиси вон реално време како што се веб и електронска пошта, или пак за сигнализација).

Глава 9

Сигурност во Интернет

9.1 Вовед

Сигурноста во Интернет е фундаментална при поставувањето на Интернет мрежа, било да се работи за локална мрежа, метрополитен или скелетна мрежа. Користењето на Интернет за транспорт на пакети кои носат најразлични типови на информации меѓу најразлични комбинации од крајни корисници кои можат да бидат на било кое место во светот каде што има конекција кон глобалниот Интернет, создава сосема друга димензија кога се зборува за опасностите што демнат и за сигурносните механизми кои треба да се применат за побезбедна комуникација преку IP.

Што е тоа Интернет сигурност? Одговорот не е еднозначен, туку е покомплексен. Ќе се обидеме да го одговориме прашањето преку неколку аспекти, но тоа не значи дека се е опфатено или секој детал. Па, Интернет сигурност е:

- **Доверливост:** само праќачот и примачот треба да ја разберат содржината на пораката:
    - Праќачот ја шифрира (шифрира) пораката
    - Примачот ја дешифрира (дешифрира) пораката
- **Автентикација:** праќачот и примачот треба да го потврдат својот идентитет меѓу себе, или кон мрежата (на пример, кон Интернет сервис провајдерот)
- **Интегритет на пораката:** праќачот, примачот сакаат да бидат сигурно дека пораката не е променета при преносот
- **Достапност:** сервисите мора да бидат пристапни за корисниците

Што може да направи злонамерен Интернет корисник?

Секако има повеќе можности за тоа. Еве неколку од нив:

- **Прислушување (eavesdrop):** да се копираат пораките кои поминуваат низ некоја мрежа или јазел
- Активно да се вметнуваат (**insert**) пораки во конекцијата
- Лажно претставување со промена на изворната адреса во IP пакетот (**spoof**) или било кое поле во пакетот
- **hijacking:** преземање на тековна конекција преку отстранување на праќачот или примачот и нивно заменување со "напаѓачот"
- **denial of service:** спречување на даден сервис да биде користен од корисниците на тој сервис (на пример, преку преоптеретување на серверите), итн.

Според тоа, бидејќи постојат различни опасности по сигурноста на корисникот кој користи сервиси преку Интернет (дел од нив се наведени погоре), секоја комуникациска мрежа треба да поседува безбедносни механизми за заштита од несакани напади. Со оглед на ова имплементирани се безбедносни механизми на сите нивоа според OSI. Во табела 9.1 може да се видат безбедносните протоколи имплементирани на различни нивоа според OSI. Во понатамошниот текст секој од нив ќе биде разгледан подетално.

Табела 9.1 Безбедносни протоколи на различни нивоа според OSI

Ниво според OSI	**Безбедносни протоколи**
Ниво 7 – Апликациско ниво	Firewall, антивируси
Ниво 6 – Презентациско ниво	
Ниво 5 – Сесиско ниво	
Ниво 4 –Транспортно ниво	SSL/TLS
Ниво 3 – Мрежно ниво	IPSec (VPN)
Ниво 2 – Дата линк ниво	802.1X, WPA, WEP
Ниво 1 – Физичко ниво	Полиси за корисничка контрола на пристап базирани на физичката локација

Имплементацијата на безбедносните механизми во пристапните мрежи треба да биде транспарентна за апликациите. Оттаму безбедносните механизми може да се имплементираат во сите нивоа до петтото ниво (Сесиско ниво), бидејќи Презентациското ниво (ниво 6) ретко се користи. Безбедноста во линк нивото е базирана на принципот делница-по-делница (hop-by-hop) имајќи предвид дека тоа работи со MAC – адресите на хардверот, додека безбедноста на мрежното ниво се базира на принципот од крај-до-крај. Безбедносните механизми кои ќе се применат зависат од намената на мрежата (јавна мрежа или корпоративна мрежа). Кај корпоративните мрежи се нагласува безбедноста на податоците за разлика од едноставноста за користење на истата, додека кај јавните IP мрежи едноставноста на користење на мрежата игра поголема улога за разлика од нивото на безбедност на податоците имплементирано во нив. Секогаш постои баланс кој треба да се постигне меѓу безбедноста на системот и негова едноставност за користење, особено кај јавните мрежи.

9.2 Безбедносни предуслови при дизајн на Интернет мрежи

Прв чекор при дизајн на една Интернет мрежа е определување на неговите потреби за безбедност. Дизајнот на една јавна IP мрежа ги диктира следните безбедносни параметри:

- Физичка безбедност
- Пристап до Интернет им е дозволен само на авторизирани корисници
- Пристап до локалните Web сервери (Web сајт со локални информации) им е дозволен на сите корисници
- Сите корисници (авторизирани и неавторизирани) треба да имаат физички пристап до мрежата
- Авторизациските параметри како корисничко име и лозунг треба да се сместени во централна база на податоци за Интернет корисниците на дадениот сервис провајдер

- Системот треба да има минимални барања за имплементација на клиентската страна (за да не биде пречка за технички помалку обучените корисници, кои всушност го сочинуваат најголемиот број од Интернет корисниците).

9.3 Автентикација, авторизација и тарифирање (AAA) во Интернет

AAA (Authentication, Authorization, Accounting) ознаката е кратенка од првите букви на основните сервиси кои ги обезбедува еден AAA сервер, а тоа се: Автентикација, Авторизација и Акаунтинг (Тарифирање) . Накратко, секоја од овие функции може да се опише на следниот начин: Автентикациската функција овозможува идентификација на корисникот, Авторизациската функција определува што му е дозволено на корисникот, односно кои сервиси може тој да ги користи, додека функцијата за тарифирање врши мерење на искористеноста на ресурсите кои ги користел корисникот со цел оваа услуга за искористените ресурси потоа да ја наплати.

Постојат повеќе вакви системи кои овозможуваа имплементација на овие три функции и тоа: (Remote Dial-In Remote Access Service) – RADIUS), потоа системот наречен TACACS+ (Terminal Access Controller Access Control System plus), и системот DIAMETER.

9.3.1 TACACS+

TACACS+ (Terminal Access Controller Access Control System plus) е наследник на TACACS, системот кој повеќе не се користи. Додека TACACS е UDP базиран автентикациски и пристапен протокол, TACACS+ е TCP базиран протокол кој претставува подобрена верзија на TACACS протоколот (исто како и XTACACS кој преставува проширена верзија на TACACS протоколот, преработена од страна на Cisco) на начин што е извршена сепарација на функциите на автентикација, авторизација и тарифирање, со воведување на шифрирање на сообраќајот помеѓу серверот за контрола на мрежниот пристап и автентикацискиот сервер. Сепак, доколку корисникот сака

безбедна комуникација од точка до точка, тогаш тој ќе користи IPSec шифрирање на комуникацијата од клиентот (пример, Персонален Компјутер) до серверот. Ова значи дека дополнителната шифрирање која е инкорпорирана во TACACS+ протоколот не е неопходна при нормален начин на работа. TACACS+ протоколот овозможува исклучиво временски базирано тарифирање, билинг информациите во себе содржат податоци за времето на почеток на комуникацијата, идентификатор на корисникот, адресата од каде е иницирана конекцијата, почетното време и крајното време, но не содржат информации за поддршка на волуменски базирано тарифирање како на пример број на испратени или примени октети.

9.3.2 RADIUS

RADIUS (Remote Dial-In Remote Access Service) е иницијално развиен за dial-up сервиси. Сепак тој преставува клиент/сервер протокол, каде серверот за контрола на мрежен пристап оперира како RADIUS клиент кој комуницира со RADIUS серверот. На пример RADIUS клиент може да биде AP (Access Point) со вградена функција на RADIUS клиент.

RADIUS серверите се одговорни за разрешување на барањата за корисничка конекција. Иницијално, серверот за контрола на мрежниот пристап (Network Access Server – NAS) прима корисничко барање за автентикација (пример, Барање за Најавување на мрежа), тогаш RADIUS клиентот имплементиран во мрежниот елемент ги испраќа потребните информации за остварување на процесот на автентикација до назначениот RADIUS сервер. Потоа RADIUS серверот ја проверува својата база и враќа повратен одговор, кој може да биде потврден, негативен или challenge response (зависно од резултатите добиени со проверка во RADIUS базата).

Покрај автентикацијата и авторизацијата кои се нераздвоиви во имплементацијата на AAA во RADIUS протоколот, RADIUS протоколот исто така поддржува тарифирање, кој покрај временски може да биде и волуменски базирано (број на бајти испратени и примени од одреден корисник). RADIUS серверот може да ги спроведува автентикациско/авторизирачките и функциите на тарифирање независни една од друга и може да се конфигурира да работи со само една од нив или и двете.

Пакетите од RADIUS протоколот се испраќаат како чист текст со шифрирање применета само врз делот наменет за пренесување на кориснички лозунг.

RADIUS го користи UDP како трансмисиски протокол. Објаснувањето кое што стои зад изборот на UDP како трансмисиски протокол се базира на природата на функционирање на системите базирани на RADIUS протокол. Имено RADIUS клиентите најчесто, поради безбедносни причини, користат примарен и секундарен RADIUS сервер за испраќање на своите барања (вториот се користи во случај на недостапност на првиот). Ретрансмисиите кон вториот сервер бараат поставување на тајмери кои се разликуваат од оние кои се дефинирани во TCP протоколот. Покрај тоа, UDP овозможува да се одржува отворена конекцијата меѓу RADIUS клиентот и серверот и тогаш кога има пречки или прекин во линијата. Значи, доверливоста при преносот на пораките е целосно базирана на RADIUS механизмите на апликациско ниво (не на транспортното ниво како што е кај TCP).

Во денешните IP базирани мрежи (како во жичните, така и во безжичните) RADIUS протоколот е доминантен AAA протокол, и како таков е добар избор за имплементација во систем кој користи AAA.

9.3.3 DIAMETER

DIAMETER протоколот се гледа како наследник на RADIUS протоколот. Значителна промена во концептот е тоа што тој работи како peer-to-peer (ниво до ниво) комуникација, наместо постоечкиот усвоен начин на клиент/сервер комуникација, на кој функционира и постоечкиот RADIUS протокол.

Базичниот протокол ги дефинира сите сервиси кои треба да функционираат во сите DIAMETER сервери.

DIAMETER протоколот нуди можност за шифрирање на целиот пакет или само на негови делови, додека RADIUS протоколот го шифрира само корисничкиот лозунг. Имајки предвид дека DIAMETER е креиран како наследник на RADIUS протоколот тој во себе ги вклучува сите негови функционалности како и дополнителни апликациски екстензии (како што се: IPv6 поддршка, Mobile IP, и сл.).

Иако DIAMETER е понов протокол од RADIUS протоколот со поддршка за RADIUS клиентите и дополнителни можности, тој не е поддржан од ниеден од

поголемите производители на мрежна опрема и во оваа фаза е сеуште само за лабораториска употреба. Иднината на DIAMETER протоколот е исто така несигурна, бидејќи го менува механизмот на комуникација од досега усвоениот со RADIUS протоколот а тоа е клиент/сервер во peer-to-peer комуникација. Постои можност DIAMETER да не заживее, барем не скоро, имајќи ја во предвид доминантната улога на RADIUS за AAA.

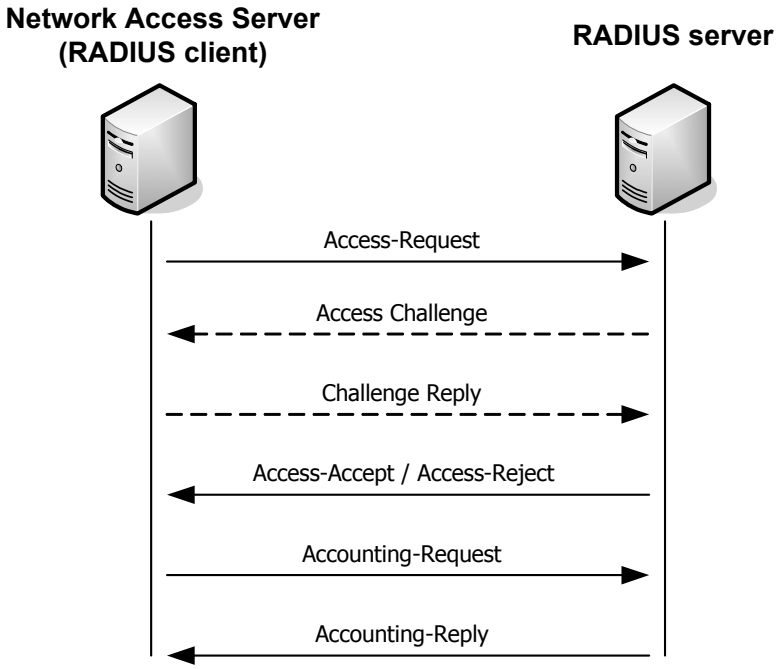
9.4 Опис на RADIUS протоколот

RADIUS протоколот преставува изведба на AAA систем и како таков тој е дефиниран преку RFC 2865, [40], и RFC 2866, [41], а одредени негови апликациски екстензии се дефинирани и во RFC 2869, [42]. Основни негови карактеристики се следните:

- Клиент/Сервер модел – Серверот за контрола на пристап (NAS) функционира како клиент на RADIUS сервер. Како клиент тој е одговорен за предавање на корисничката информација до одредишниот RADIUS сервер, и спроведување на добиениот одговор. RADIUS серверите се одговорни за прием на корисничката информација, автентикација на корисникот и потоа праќање на одговор со целата потребна информација за да може клиентот да му го пружи бараниот сервис на корисникот. RADIUS серверот може да игра улога на редистрибуциски RADIUS сервер (RADIUS proxy) сервер кон други RADIUS или други автентикациски сервери (на пример, при роаминг на Интернет корисникот во мрежа на друг ISP).
- Мрежна безбедност – Трансакциите помеѓу клиентот и RADIUS серверот се автентицирани преку користење на заеднички лозунг, кој никогаш не се праќа преку мрежата. Како една од мерките на безбедност која во себе ги вклучува RADIUS протоколот е шифрирање на корисничкиот лозунг со цел да се спречи неговото откривање од непожелни лица кои би ја фатиле RADIUS комуникацијата.
- Поддршка за мноштво од автентикациски механизми – RADIUS протоколот поддржува различни автентикациски механизми и тоа:
- PAP (Cleartext Password Authentication Protocol)

- CHAP (Challenge Handshake Authentication Protocol)
- Challenge – Response процедури
- EAP (Extensible Authentication Protocol)
- Екстензибилен протокол – RADIUS протоколот нуди можност од имплементација на нови параметри без нарушување на постоечките имплементации на протоколот.

Како што беше наведено погоре RADIUS е UDP базиран протокол. Користењето на UDP наместо TCP за транспортен протокол како во TACACS+ протоколот не е претставува никаква пречка или несовршеност на RADIUS протоколот, туку како што е опишано во RFC 2865 ова е направено од технички причини. Прва причина е потребата од алтернативна трансмисија на RADIUS пакетите кон секундарниот RADIUS сервер доколку примарниот не е присутен.(поради оваа причина потребно е да се чува копија од пораката над транспортното ниво). Втора причина преставуваат различните барања за време во однос на TCP, пр. TCP ретрансмисиите не се потребни. Трета причина е поедноставувањето на програмската изведба, имено секој од двете страни (клиентот или серверот) може да го отвори својот UDP транспортен порт само еднаш и да го држи отворен за цело време на комуникацијата без разлика дали има мрежни или комуникациски прекини. Четврта причина е што UDP ја симплифицира имплементацијата на multithreading во RADIUS серверот (креирање на одделни процеси за паралелна автентикација на повеќе различни корисници). Точно еден RADIUS пакет се енкапсулира во еден UDP пакет. RADIUS UDP пакетите при една радиус комуникација треба да го имаат како дестинациски порт дефиниран портот 1812 за автентикација и 1813 за тарифирање. Постарите верзии на RADIUS ги користат портите 1645 и 1646 соодветно (сеуште постојат имплементации на сервери кои ги користат овие порти). Базична RADIUS комуникација помеѓу клиентот и серверот е прикажана на слика 9.1. На пример, ако RADIUS клиентот сака да автентицира одреден корисник тогаш тој праќа пакет наречен Access Request (Барање за пристап) и RADIUS серверот ќе одговори со Access-Accept или Access-Reject пакет.



Слика 9.1 RADIUS комуникација

Слика 9.2 Формат на RADIUS пакет

Обликот на пакетот на RADIUS протоколот е прикажан на слика 9.2 Како што може да се види на сликата **код** полето е со должина од 1 бајт, и се користи за идентификација на типот на пакетот. Сите можни типови на **код** идентификаторот се дадени во табела 9.2.

Полето означено како **Идентификатор** е долго 1 бајт и се користи за спарување на барањата и одговорите. RADIUS серверот може да детектира дуплицирано барање ако има иста клиентска изворишна адреса, ист изворен UDP порт и ист идентификатор добиен во краток временски интервал.

Полето означено како должина, ја дефинира должината на пакетот во бајти вклучувајки ги сите полиња (минимална вредност е 20 бајти а максимална 4096).

Полето означено како **Автентификатор** е со должина од 16 бајти и се користи во алгоритамот за шифрирање и заштита на корисничкиот лозунг. Вредноста во полето

означено како Автентификатор се вика Request Authenticator (Автентификатор за процесирање на барањето) во пакетите означени со Access-Request, додека во пакетите означени како Access-Accept, Access-Reject и Access-Challenge оваа вредност се вика Response Authenticator (Автентификатор кој одговорил на барањето). Ова поле е базирано на алгоритамот означен како Message Digest Algorithm (MD5), а е резултат на примена на овој алгоритам врз одреден случаен број и заедничкиот лозунг за меѓусебна автентикација на RADIUS серверот и RADIUS клиентот.

Код	Тип на пакет
1	Access-Request
2	Access-Accept
3	Access-Reject
4	Accounting-Request
5	Accounting-Response
11	Access-Challenge
12	Status-Server(експериментално)
13	Status-Client(експериментално)
255	Reserved

Табела 9.2 Кодни типови – Типови на RADIUS пакети

Последното поле означено како **Атрибут** е променливо со својата должина и во себе содржи листа од атрибути кои се потребни за типот на сервис, исто како и саканите опционални атрибути. Според RFC 2865 дефинирани се повеќе 60 атрибути а оставен е и простор за дефинирање на дополнителни атрибути според потребите на производителот. Меѓутоа доколку се применуваат овие дополнителни атрибути не постои гаранција дека меѓусебната комуникација помеѓу сервер и клиент од различни производители ќе функционира.

9.4.1 Процесирање на барањата од страна на RADIUS серверот

Откако RADIUS серверот ќе го прими Access-Request пакетот наменет за него ќе пристапи кон верификација на пакетот, односно ќе провери дали серверот го знае заедничкиот лозунг за комуникација помеѓу серверот и клиентот од каде е испратен

пакетот. Ако серверот не го знае лозунгот пакетот не се процесира понатаму. Доколку тој го знае лозунгот тогаш тој со негова помош ќе го примени врз User-Password атрибутот алгоритамот за дешифрирање и ќе го дешифрира корисничкиот лозунг во облик на чист текст. По добивање на лозунгот во облик на чист текст серверот ќе провери во својата база дали поседува пар од корисничко име и лозунг кој одговараат на оние испратени во Access-Request пакетот. Доколку лозунгот е валиден тогаш серверот креира Access-Accept пакет кој му го праќа назад на клиентот. Ако лозунгот не е валиден тогаш серверот креира Access-Reject пакет и истиот го праќа на клиентот. И Access-Accept и Access-Reject пакетот го користат истиот идентификатор добиен од Access-Request пакетот и го пополнува полето означено како Одговор на Автентификатор со својата адреса.

9.4.2 Процесирање на барањата од страна на RADIUS клиентот

Откако RADIUS клиентот ќе го прими пакетот добиен како одговор на неговото барање од RADIUS серверот, тој пробува да го поврзе со точното барање, според полето за идентификација. Ако барање со таков идентификатор не е пратен од овој клиент тој го отфрла примениот пакет. Доколку таков идентификатор постои тогаш клиентот го проверува полето означено како одговор на Автентификаторот. Процедурата на проверка е иста со онаа на серверот, односно со помош на заедничкиот лозунг го применува алгоритмот за проверка и според добиениот резултат определува дали одговорот е добиен од серверот кон кого е пратено барањето. Доколку со споредбата не се добие бараниот резултат тогаш пакетот се отфрла. Доколку претходните проверки се во ред и како резултат на барањето се добие Access-Accept пакет, тогаш се смета дека испратениот пар од корисничко име и лозунг е исправен и корисникот е автентициран. Доколку како резултат на барањето по проверките се добие Access-Reject пакет тогаш се смета дека испратениот пар од корисничко име и лозунг не е исправен и корисникот не е автентициран.

9.4.3 RADIUS Тарифирање

RADIUS протоколот според поддржува тарифирање на остварените кориснички сесии контролирани од неговите RADIUS клиенти. Делот за тарифирање од RADIUS протоколот може да се користи одделно од неговиот Автентификациски и Авторизациски дел со цел да се наплати искористениот сервис од страна на корисниците. Делот за RADIUS тарифирање од RADIUS протоколот е исто така базиран на клиент-сервер моделот. Серверот за контрола на мрежниот пристап (NAS), функционира како клиент на серверот за RADIUS тарифирање. Тој собира информации за корисничката активност како што се: времетраењето на корисничката сесија, испратени и примени бајти и пакети. Од друга страна (RADIUS) серверот за тарифирање ги прима барањата за тарифирање добиени од страна на клиентот а повратно враќа одговор дека барањето е успешно примено.

Кога клиентот е конфигуриран да го користи делот за RADIUS тарифирање на почеток на сервисот тој испраќа Accounting Start пакет со информации за сервисот и клиентот (пр. NAS праќа почетна порака кон серверот за тарифирање по пристап на корисникот до NAS). Серверот го потврдува примањето на пакетот. RADIUS клиентот може да испраќа периодични пораки на конфигурабилни периодични интервали (пр. интервалите можат да бидат конфигурирани во опсег од 1-4096 минути). На крајот на корисничката сесија клиентот генерира Accounting Stop пакет со опис на сервисот кој се испорачува и опционо информација како што е изминато време, испратени и примени октети или испратени и примени пакети. RADIUS серверот испраќа Access-Response пакет кон RADIUS клиентот само доколку успешно го прими пакетот за тарифирање. Делот за RADIUS тарифирање од RADIUS протоколот го користи UDP протоколот. Првите негови имплеметации го користат UDP портот 1646, додека официјалниот доделен порт за користење е 1813 и новите имплементации го користат токму овој порт. Битно да се напомене е и фактот дека одредени имплеметации на RADIUS може да обезбедуваат само сервис за Автентикација и Авторизација, а не и за Тарифирање.

9.4.4 Имплементации на RADIUS сервер

Денес се познати повеќе различни имплементации на RADIUS сервер направени од различни производители. Како најпопуларни и најпознати имплементации издвоени се следните:

- **FreeRadius –** open source RADIUS имплементација на Linux платформа
- **IAS (Internet Authentication Services) –** Microsoft RADIUS сервер
- **Cisco Secure –** RADIUS сервер имплементација направена од Cisco

Постојат уште многу различни имплементации на RADIUS сервер како open source така и комерцијални, но овие три наведени доминираат меѓу академскиот свет, како и на светскиот пазар. Сите се базирани на RFC 2865 и 2866 за RADIUS. Додека FreeRadius е open source имплеметација на RADIUS додека другите две не се. Овој факт укажува на тоа дека доколку е потребно да се извршат некакви модификации на кодот на RADIUS протоколот, тогаш FreeRadius е правилен избор, додека во спротивно било кој од останатите два сервери е можен избор.

9.5 Примена на 802.1X и EAP

Подобрување на безбедноста во IP мрежите е можно со користење на 802.1X и EAP. 802.1X дава автентификациска рамка за IP мрежа (на пример, Интернет сервис провајдер), овозможувајки еден корисник да биде автентициран од еден централен автентикациски сервер. Вистинскиот алгоритам за автентикација кој го користи 802.1X е отворен, што значи дека можна е имплеметација на повеќе различни алгоритми, како на пример:

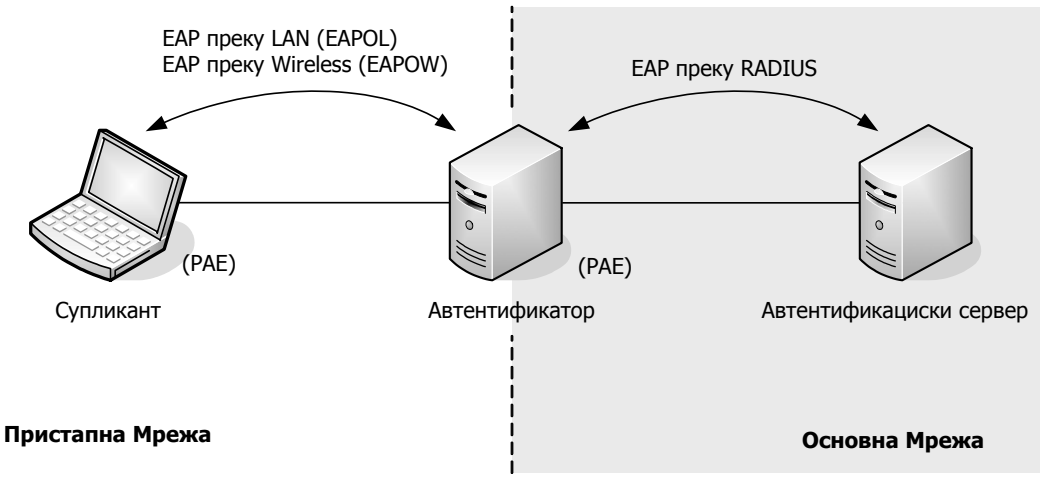
- Алгоритми базирани на Сертификати, како што се EAP- Transport Layer Security (EAP-TLS)
- Алгоритми базирани на кориснички лозунг, како што се EAP- One Time Password (EAP-OTP), и EAP – Message Digest 5 (EAP- MD5)
- Решенија базирани на смарт-картички, како што се EAP – Subscriber Identification Module (EAP-SIM).

- Хибридни решенија, како што се EAP – Tunnelled TLS Authentication Protocol (EAP-TTLS) кое користи и сертификати и кориснички лозунг.
- Специјални EAP решенија, како што е Cisco LEAP (Lightweight EAP)

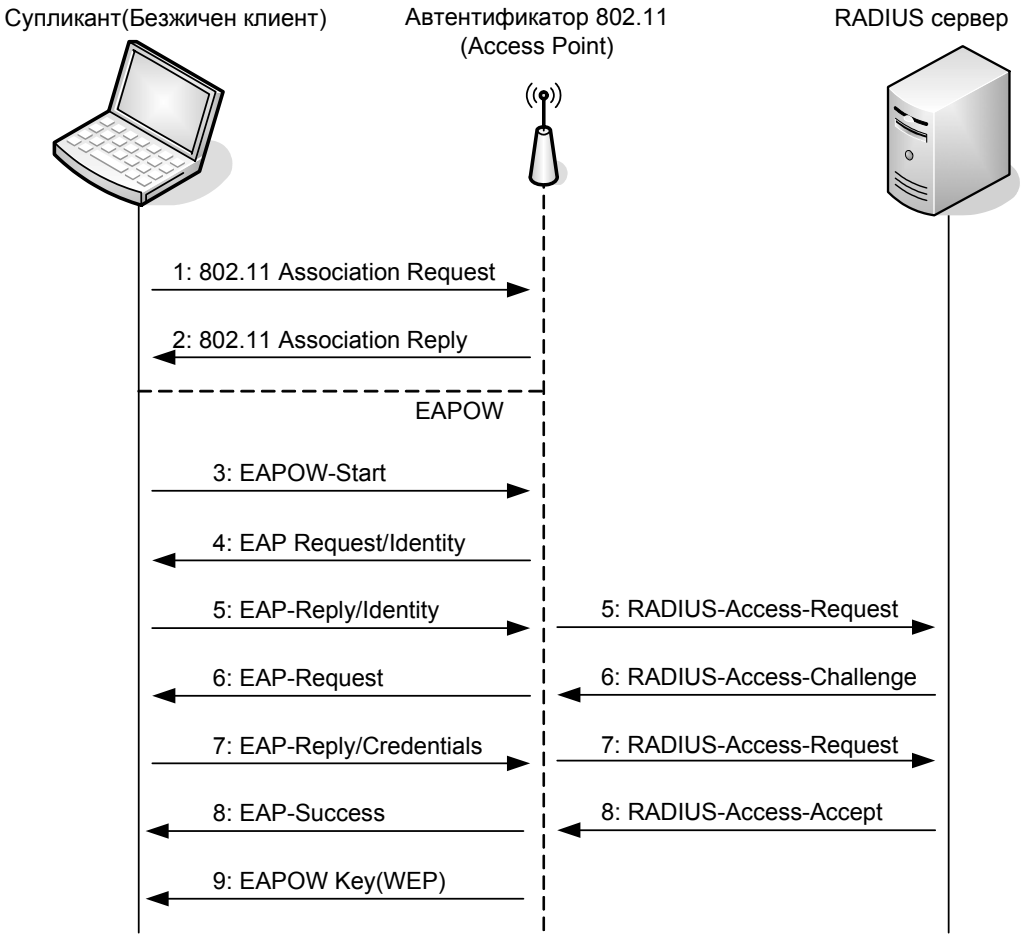
Во сите овие решенија за подобрување на безбедноста 802.1X користи EAP, како протокол (специфициран како RFC 2284, [43]) кој може да се користи и во жични (пр. Ethernet) и во безжични мрежи за размена на пораки во процесот на автентикација. IEEE 802.1X може да се користи за екстракција на клуч со цел да се обезбеди автентикација, авторизација, интегритет и доверливост на ниво на пакет, но IEEE 802.1X не обезбедува шифрирање самиот по себе. Поради тоа потребна е имплеметација на друг шифрирачки алгоритам за таа намена, како што се (само ќе ги споменеме тука некои од нив: WEP, 3DES или AES). Шифрирачките алгоритми се користат заедно со алгоритмите за екстракција на клуч како што е TLS.

9.5.1 Употреба на 802.1X

IEEE 802.1X дефинира три компоненти во процесот на автентикација, кои се прикажани на слика 9.3. Супликант преставува крајниот корисник кој бара пристап до мрежните ресурси. Мрежниот пристап е контролиран од автентикатор, кој ја игра улогата на серверот за контрола на мрежен пристап во dial-up мрежите. Супликантот и автентикаторот се означени уште како (PAE – Port Authentication Entities) ентитети со автентификациски порти. Автентикаторот ги терминира само автентикациските барања на линк нивото, и не чува никакви информации за корисникот кај себе. Сите автентикациски барања тој ги пренасочува кон автентикацискиот сервер, како што е некој RADIUS сервер. Портите на станицата која работи со 802.1X се во т.н. авторизирана состојба кога тие се овозможени за работа, а се во неавторизирана состојба кога тие се затворени и преку нив не може да се работи. Дури и кога се во неавторизирана состојба според спецификациите кон нив е дозволен DHCP и друг тип на иницијализирачки сообраќај.



Слика 9.3 802.1X архитектура



Слика 9.4 802.1X - Автентикација

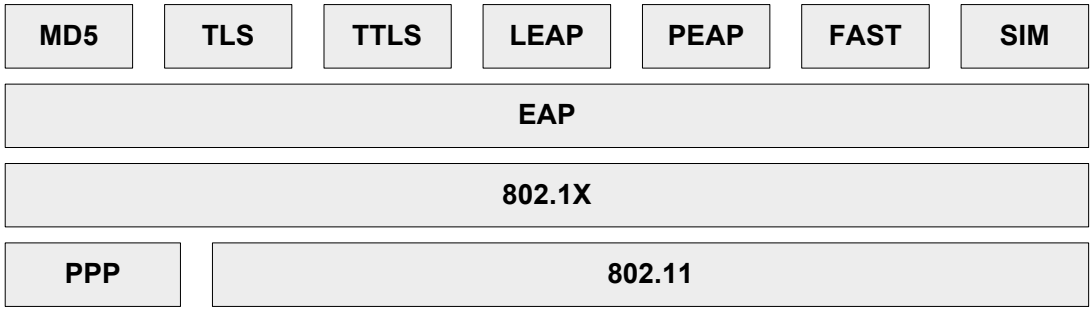
Процесот на автентикација на логичко ниво се одвива помеѓу супликантот и автентикацискиот сервер, каде автентификаторот игра улога на мост помеѓу нив. Издвојувањето на EAP се користи од страна на автентификаторот со цел да може да ги

препраќа барањата и одговорите меѓу супликантот и автентикацискиот сервер. Од страна на супликантот кон автентикаторот, протоколот кој се користи е EAP преку LAN (EAPOL) или EAP преку WLAN (EAPOW), додека од другата страна протоколот кој се користи е RADIUS и најчесто се означува како EAP преку RADIUS. На пример, кога еден неавтентициран корисник ќе се обиде да се конектира на Интернет преку некој пристапен сервер користејќи 802.1X, само на EAP пакетите им е дозволено да поминат преку пристапната IP мрежата (на пример: Етернет, безжична локална мрежа). EAP пакетите се пропуштаат кон автентикацискиот сервер (пример, RADIUS) додека целиот друг сообраќај се блокира се до моментот на верификација на корисничкиот идентитет. Кога автентикацискиот сервер ќе потврди дека корисникот е успешно автентициран тогаш AP го подигнува својот firewall и дозволува пристап на клиентските податоци до безжичната мрежа. Клучевите се генерираат меѓу RADIUS клиентот и RADIUS серверот, така што серверот му го праќа клучот на клиентот. Бидејќи за нормална комуникација според RADIUS протоколот клиентот и серверот делат заеднички лозунг, тој се користи за шифрирање на атрибутите кои го содржат клучот. Трансмисијата на клучот до клиентскиот уред се врши користејќи EAPOW-Key пакет. 802.1X не е единствен автентикациски метод, но е еден од оние кои користи EAP како своја автентикациска основа. Ова значи дека свичовите и пристапните точки во IP мрежата кои поддржуваат 802.1X можат да користат повеќе различни автентикациски методи. Еден од основните проблеми во некои IP мрежи е недостаток на взаемна автентикација (пример, мрежата може да го автентицира корисникот, но не и обратно). Взаемна автентикација не е експлицитно обезбедена и во 802.1X. Без можност за взаемна автентикација можен е напад врз клиентот со појава на лажна пристапна точка при користење на EAP со ограничена автентикација (како EAP-MD5). Решение за овој проблем е NAS да се конфигурира да користи EAP кој обезбедува взаемна автентикација. Битен момент што треба да се напомене е дека 802.1X ги поддржува сите AAA протоколи, иако RADIS е предложен како можно решение.

9.5.2 EAP – Extensible Authentication Protocol

Основен механизам врз кој е базиран 802.1X е EAP протоколот. EAP бил иницијално дизајниран за користење со (PPP – Point-to-Point-Protocol). EAP преставува

peer-to-peer автентификациски протокол кој се користи помеѓу супликантот и автентификацискиот сервер. Тој може да работи врз било кое линковско ниво, како што се PPP и 802.11 (како што е прикажано и на слика 4). EAP е дефиниран како генерален автентификациски протокол, кој на пример може да се извршува врз RADIUS протоколот. Во тој случај автентификаторот може да ги разбира само RADIUS пораките. Генерално EAP протоколот овозможува примена на различни автентикациски методи и тоа: EAP-MD5, EAP-TLS, EAP-TTLS, LEAP, PEAP, EAP-FAST и EAP-SIM. Во портфолиото на EAP протоколот се забележани и други методи, но овие се најчесто користени. Во делот што следи е даден краток опис на секој од нив.



Слика 9.5 802.1X/EAP протоколен стек

EAP – MD5

EAP-MD5 преставува автентификациски протокол базиран на барање и одговор - Challenge Handshake Authentication Protocol (CHAP), и како таков преставува протокол за базична EAP поддршка меѓу 802.1X уредите. Тој преставува најнесигурна верзија на EAP бидејќи користи кориснички имиња и лозунзи за автентикација кои можат лесно да се откријат. Значи тој е ранлив на напад извршен со менување на поими добиени со помош на речник од зборови.

EAP – TLS

EAP-TLS (TLS - Transport Layer Security) е отворен стандард и е поддржан од најголемиот број на производители. TLS не е специјално дизајниран за работа во одреден тип мрежи. Тој се базира на SSL – Secure Socket Layer (како негов претходник), и преставува безбедносен метод кој се користи на апликациско ниво (пример, кај Интернет Пребарувачи). EAP – TLS е природно поддржан во Windows XP, Windows 2000 и Windows 2003 серверите. Единствена забелешка е тоа што сите кориснички

уреди треба да поседуваат сертификат. Меѓутоа кога ќе се постави еднаш EAP-TLS тогаш тој е виртуелно транспарентен за корисникот.

EAP – TTLS

EAP-TTLS и EAP-TLS се слични во тоа што и двата го користат TLS како технологија и по тоа што и двата применуваат силна криптографија. Меѓутоа EAP-TTLS се разликува по тоа што се бара да само RADIUS серверите поседуваат сертификат. Корисникот се автентицира на мрежата со користење на обичен лозунг, чија употреба е направена да биде отпорна на пасивни и активни напади со тоа што неговата размена се одвива преку TLS . TTLS е направен да поддржува повеќе различни автентикациски методи и тоа: PAP, CHAP, MS-CHAPv1, MS-CHAPv2, PAP/Token Card или EAP. Со оглед на покажаните перформанси и безбедносни можности тој има големи изгледи да стане дефинитивен главен автентификациски метод, при што главен ривал на EAP-TTLS од аспект на автентикациски метод е Protected EAP (PEAP).

LEAP

Cisco беше еден од првите производители кој на Интернет пазарот појави своја верзија на EAP протокол, наречена LEAP – Light EAP. Сепак LEAP преставува решение на еден производител, а не стандард и како таков е поддржан од Cisco мрежни уреди. Интересен е фактот што тој претставува прво решение кое комерцијално го користи EAP врз 802.1X во безжични мрежи. LEAP го користи MS-CHAPv1 механизмот, овој механизам е познато дека има свои недостатоци, врз база на кои се развиени и алатки за пробивање на истиот. Имајки ги предвид овие недостатоци тој не е погоден за имплементација во јавна мрежа.

PEAP

Protected EAP (PEAP) преставува безбедносен механизам базиран на авторизациски параметри (корисничко име и лозунг) кој овозможува начин за остварување на безбедна EAP автентикација. Тој е еден од двата автентикациски метода (покрај EAP-TTLS) кој е дизајниран со цел да се надмине барањето на TLS алгоритамот за употреба на клиентски сертификати. Во PEAP како и кај TTLS, IP станицата се идентифицира со користење на корисничко име и лозунг. Иако EAP иницијално е креиран за користење со PPP, од тогаш наваму тој е прилагоден за негово користење со 802.1X. По неговата апликација во реални решенија, одреден дел на негови недостатоци испливаа на

површина. Во овие недостатоци спаѓаат следниве: Недостаток на заштита на корисничкиот идентитет, не постоење на стандарден начин на размена на клучеви, без вградена поддршка за фрагментација и реасемблирање, како и недостаток на алгоритам за брзо реконектирање. Сите овие недостатоци на EAP во безбедноста се покриени со PEAP. За да го постигне ова PEAP врши заштита на EAP протоколот со помош на TLS. Секој EAP метод кој работи со PEAP е способен за размена на клучеви и сесиско обновување. PEAP исто така обезбедува транспарентен роаминг(преод) меѓу пристапните точки. Иако PEAP е продукт кој е развиен од Cisco, тој е лиценциран од повеќе компании меѓу кои и Cisco и Microsoft.

PEAP v2

Овој автентификациски метод во моментот сеуште се развива и треба да биде наследник на PEAP. Според спецификациите PEAPv2, за автентикација треба да користи MS-CHAPv2, како механизам за автентикација.

EAP-FAST

EAP-FAST овозможува безбедна комуникација помеѓу клиентот и серверот со користење на EAP-TLS за креирање на взаемно автентициран тунел. Како и PEAP протоколот овој протокол е базиран на TLS. Притоа се направени одредени подобрувања за да му се овозможи на EAP-FAST да иницира тунел за размена на клучеви користејќи симетрична криптографија, каде овој тунел се користи за заштита на послаби автентификациски методи кои ќе се одвиваат преку него. EAP-FAST користи MS-CHAPv2 за автентикација слично како и PEAPv2. EAP-FAST е еден од поновите предлози дадени на IETF од страна на Cisco, и имајќи го ова во предвид доколку остане на ова ниво и е применет само во мрежните елементи произведени од Cisco тогаш тој нема да биде применлив во јавни IP мрежи.

EAP-SIM

EAP-SIM е EAP метод кој овозможува хардверска автентикација користејќи GSM Subscriber Identity Module (SIM) чип. SIM картичките се користат од страна на GSM мобилните телефони за автентикација на корисник на мобилната мрежа. Оваа функција за автентикација е инкорпорирана во GSM/GPRS мобилните телефони.

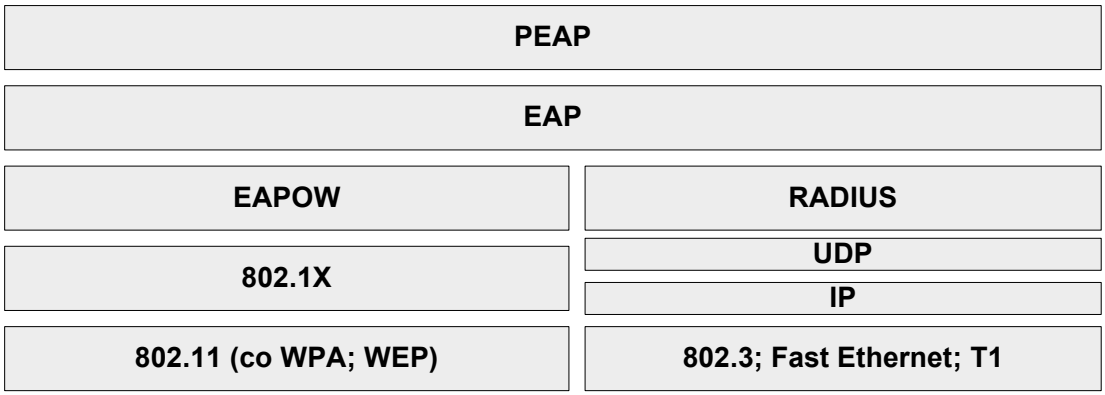
9.5.3 Избор на EAP автентификациски метод за 802.1X/EAP автентикација

Како што беше зборено погоре EAP преставува генерален концепт за автентикација и како таков тој не дефинира конкретен метод за автентикација. Од тие причини прва задача е да се донесе одлука за типот на автентикација за Интернет корисниците.

EAP-MD5 е широко раширен и поддржан од скоро сите производители, но е ранлив и не е отпорен на напади со помош на речник од зборови, и не поддржува динамичка размена на клучеви.

EAP-TLS е привлечен протокол од причина што овозможува взаемна автентикација, но бара поддршка од страна на RADIUS сервер, односно негова имплементација во RADIUS серверот. Користењето на EAP-TLS побарува и имплементација на CA – Certification Authority за издавање и менаџирање на клиентски сертификати. Ваквиот тип на автентикација е базирана на корисничко име и лозунг. Кога би требале да избереме помеѓу TTLS и PEAP, тогаш PEAP би бил прв наш избор од причина што тој е поддржан од најголемите светски производители на супликанти, а тоа се Cisco и Microsoft.

Имплементација на приватни решенија, како LEAP, е можеби добро решение на корпоративно ниво каде вработените би имале мрежна опрема од дадениот производител (Cisco), а и целата мрежа би била базирана на опрема од истиот производител. Но во јавна IP мрежа, мора да се понуди можност да корисниците имаат слободен избор од производители на мрежна опрема од каде ќе го набават својот клиент. Од тие причини ваквите приватни решенија не се добар избор за јавна мрежа. Од друга страна EAP-FAST ги решава проблемите на LEAP како техничките и неговите ранливи точки, така и од аспект на стандардизација (предлог IETF), но не е широко распространет и можно е да има иста позиција на пазарот како и сегашниот LEAP, особено по појавата на PEAPv2 кој е исто така предлог на IETF.



Слика 9.7 Протоколен стек за 802.1X со PEAP

Еден можен избор за безбеден пристап во јавна IP мрежа претставува користење на 802.1X/EAP со имплементација на PEAP (додека шифрирањета на податоците може да биде според некој шифрирачки алгоритам). Протоколниот стек на 802.1X со PEAP е прикажан на слика 9.7.

9.6 IPsec

На мрежно ниво стандард за сигурносна комуникација во Интернет е IPsec, [44], кој опфаќа цело множество од протоколи и сервиси базирани на криптографија што се користат за шифрирање на податоците за тие да не бидат прочитани или изменети на нивниот пат низ IP мрежите.

Може да се каже дека IPsec е дополнување на IP и овозможуваат испраќање и примање на криптографски заштитени интернет пакети. Најчестата употреба на IPsec е во виртуелните приватни мрежи (VPN - Virtual Private Networks), и тоа во сите три облици на VPN архитектура: меѓу портни јазли (gateway-to-gateway), меѓу хост и портен јазол (host-to-gateway) и меѓу два хоста (host-to-host).

IPsec е рамка за отворен стандард што овозможува приватна комуникација преку IP мрежи и претставува најпопуларен начин за сигурносна контрола на мрежното ниво, [45]. Сигурносната заштита е овозможена со користењето на две дополнителни заглавија: Authentication Header (AH) и Encapsulating Security Payload header (ESP). Користењето на овие заглавија во IPv4 не е задолжително за разлика од нивната примена во IPv6 каде AH и ESP се задолжително вклучени во IP пакетот.

Во состав на IPsec се користи и апликацискиот протокол IKE (Internet Key Exchange) за размена на параметрите на врската, размена на клучеви и остварување на сигурносни поврзувања, што всушност ја дефинира сигурноста на IPsec заштитената конекција.

Пред да се оствари сигурната комуникација помеѓу испраќачот и примачот, тие мора да се согласат за сите параметри што се однесуваат на сигурноста: кои сигурносни заглавија да се применат (AH, ESP или двете), кои криптографски алгоритми ќе се употребат, тајните клучеви и слично. Сите тие договори формираат сигурносно поврзување на јазлите т.е. Security Association (SA). SA е збир на алгоритми и параметри (како на пример клучеви) што се користат за шифрирање и автентикација на сообраќајот во една насока. За да се осигури двонасочниот сообраќај, потребни се две сигурносни асоцијации (SA). Во пракса SA претставува табела со информации што им се познати само на двата крајни учесници во комуникацијата. SA се чува на сигурно место во комуникацискиот систем и им е достапна само за системските процеси одговорни за комуникацијата.

9.6.1 Автентикациско заглавие (uthentication Header - AH)

Автентикациското заглавие (Authentication Header - AH) овозможува заштита на интегритетот на податоците и автентикација на корисниците. Дополнително AH може да овозможи и заштита од повторно испраќање и заштита на пристап. AH не може да ги шифрира пакетите. Во првичната верзија на IPsec ESP протоколот овозможувал само шифрирање, а не и автентикација, па затоа AH и ESP биле заедно користени за да овозможат истовремено и доверливост и заштита на интегритетот на пораките. Бидејќи во втората верзија на IPsec на ESP му е додадена можноста за автентикација, AH станал помалку важен и помалку употребуван, така што денес постојат IPsec софтверски решенија во кои AH воопшто не се користи. Сепак, AH сеуште има своја вредност бидејќи тој може да прави автентикација на некои делови од пакетите коишто ESP не може да ги автентицира.

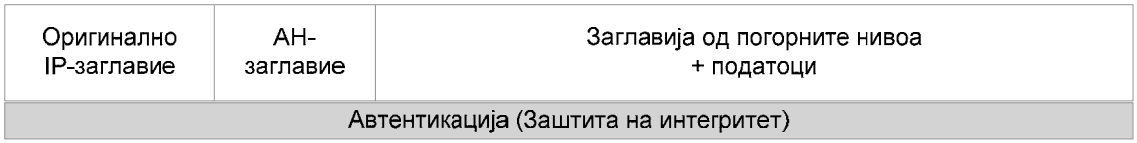
AH има два режима на работа: транспортен и тунелски режим. Во тунелски режим AH креира ново IP-заглавие за секој пакет. Во транспортен режим AH не креира

ново IP-заглавие за секој пакет. Во IPsec архитектурата во која се користи портен јазол (gateway) вистинската изворна или дестинациска IP адреса на пакетите мора да се промени во IP адресата на портниот јазол (gateway). Бидејќи во транспортниот режим не може да се менува оригиналното IP заглавие или да се креира ново заглавие, транспортниот режим обично се користи во хост-до-хост (host-to-host) архитектура. Како што може да се види на слика 9.8 и слика 9.9, AH овозможува заштита на интегритетот на целиот пакет, без разлика на тоа кој режим на работа се користи.

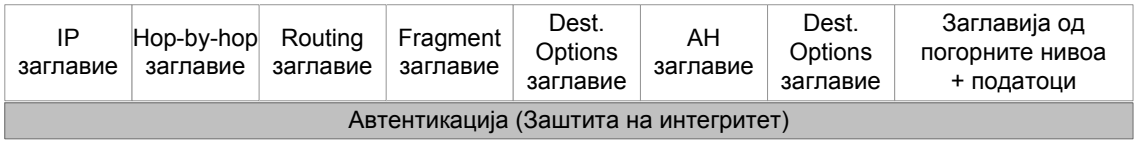
а) IPv4

б) IPv6

Слика 9.8 IP пакет со AH во тунелски мод



а) IPv4



б) IPv6

Слика 9.9 IP-пакет со AH во транспортен мод

9.6.2 Encapsulating Security Payload header (ESP)

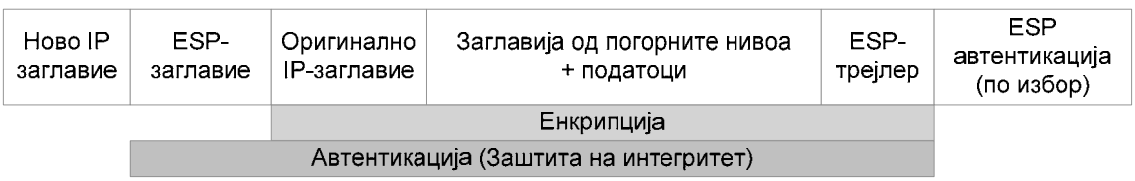
ESP е втората важна компонента на IPsec протоколот. Во првичната верзија на IPsec, ESP овозможувал само шифрирање на податочниот дел од пакетот (payload). Доколку има потреба, заштитата на интегритетот ја прави AH протоколот. Во втората верзија на IPsec, ESP станува пофлексибилен. Тој прави и автентикација со што се овозможува заштита на интегритетот, но не го опфаќа IP заглавието. ESP шифрирањата може да биде исклучена преку поставување Null ESP Encryption Algorithm. Значи во сите верзии, освен во најстарата, ESP може да се користи за да обезбеди само шифрирање, шифрирање и заштита на интегритетот или само заштита на интегритетот.

ESP модови

ESP има два мода: транспортен и тунелски мод. Во тунелски мод, ESP формира ново IP заглавие за секој пакет во кое крајните точки на ESP тунелот (на пример, меѓу два IPsec портни јазли) претставуваат изворна и дестинациска адреса за пакетот. Како резултат на тоа, тунелскиот мод може да се користи во сите три модели на VPN архитектура. Како што се гледа на слика 9.10, тунелскиот мод може да шифрира и/или да го заштити интегритетот на податоците и оригиналното IP заглавие на секој пакет.

Шифрирањата ги штити податоците од пристап или промени од неавторизирани страни. Со шифрирањата на IP-заглавието се прикриваат изворната и дестинациската адреса на пакетот, односно самата патека на комуникација. Ако се користи ESP автентикација за заштита на интегритетот, тогаш секој пакет ќе има ESP Authentication поле после ESP опашката.

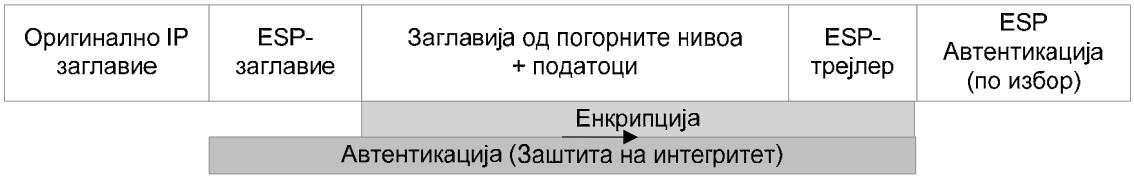
ESP тунелскиот мод е многу попопуларен од ESP транспортниот мод. Во транспортен мод ESP го користи оригиналното IP заглавие наместо да создава ново. На слика 9.10 е прикажано дека во транспортниот мод ESP може да го шифрира или заштити интегритетот само на податочниот дел (payload) од пакетот, а не и на IP заглавието.



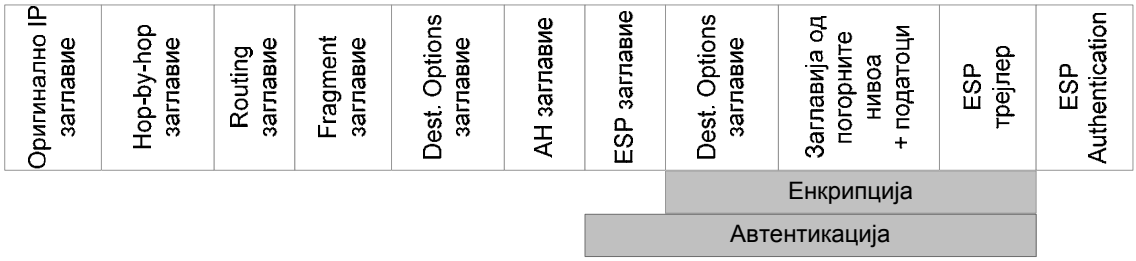
а) IPv4

б) IPv6

Слика 9.9 IP пакет со ESP заглавие во тунелски мод



а) IPv4



б) IPv6

Слика 9.10 IP пакет со ESP заглавие во транспортен мод

Како и кај АН, транспортниот мод на ESP се користи само во хост-до-хост (host-to-host) архитектура. Исто така, и кај ESP транспортниот мод е некомпатибилен со NAT (Network Address Translation), [28]. Во пресметката на checksum (делот за заштитно кодирање) се вклучени и изворната и дестинациската адреса во IP заглавието. Кога се користи NAT едната или двете адреси се менуваат, а со тоа се менува и пресметката на checksum. Кога ESP ги шифрира пакетите во транспортен мод, тогаш шифрирано ќе биде и оригиналното заглавие. Употребата на NAT не дозволува да се пресмета истиот checksum и поради тоа NAT е некомпатибилен со транспортниот мод на ESP. Ова не е случај во тунелскиот мод каде што оригиналното заглавие е скриено, па NAT нема да направи промена на checksum.

Процес на шифрирање

ESP користи симетрична криптографија за да овозможи шифрирање на IPsec пакетите. Тоа значи дека на двата краја на една IPsec конекција мора да се користи ист клуч за шифрирање и дешифрирање. Кога едната крајна точка го шифрира пакетот, таа всушност ги дели податоците и клучот на мали делови и потоа применува збир на повеќе криптографски операции, т.е. циклуси, во кои ги користи блоковите од податоците и клучот. Алгоритмите за шифрирање што функционираат на овој начин се познати како алгоритми за шифрирање на блокови. Кога другата крајна точка ќе ги прими шифрираните податоци, таа ги дешифрира со користење на истиот клуч и слична постапка, но по обратен редослед на криптографските операции. Целта на овој дел не е да се опфатат алгоритмите за шифрирање, туку да се направи преглед на нивната примена кај IPsec. Во таа насока, најчесто користени алгоритми за ESP шифрирање се: AES (Adanced Encryption Standard), AES-Cipher Block Chaining (AES-CBC), AES Counter Mode (AES-CTR), DES (Data Encryption Standard) и 3DES (Triple DES).

9.6.3 Резиме за IPsec

IPsec се применува на мрежното ниво што овозможува да се заштитат сите типови на интернет сообраќај, независно од специфичната апликација што ја иницира врската. Апликациите не се „свесни“ за заштитата при користење на IPsec и врз нив не треба да се прават дополнителни промени за да ја поддржат заштитата.

Грануларноста на IPsec заштитата е многу голема: една сигурносна асоцијација (SA) може да ја заштити целата комуникација меѓу два јазли, или пак може да заштити само одреден тип на сообраќај, или само некоја апликациска сесија (на одредена порта) или разни други варијанти на избор на сообраќај за заштита.

Нивото или типот на IPsec заштитата, како и клучевите со кои се овозможува таа заштита се флексибилни и за нив јазлите се договараат.

Бидејќи параметрите за конфигурација на IPsec се добиваат како заедничко решение на крајните јазли, тој не е секогаш направен на наједноставниот можен начин.

Постојат многу различни варијации на примена на IPsec во согласност со различните потреби и можности на корисниците, меѓутоа од особена важност е при конфигурацијата и примената на IPsec да не се премине во некоја од двете крајности:

- Со цел да се овозможи што поголема заштита, IPsec протоколот да биде толку многу комплексен што ќе биде практично неприменлив.
- Со цел да се овозможи што е можно поширока примена на IPsec, да се направи негова нестандардна конфигурација или да се применат нецелосни IPsec решенија и со тоа да се загрози сигурноста на комуникацијата.

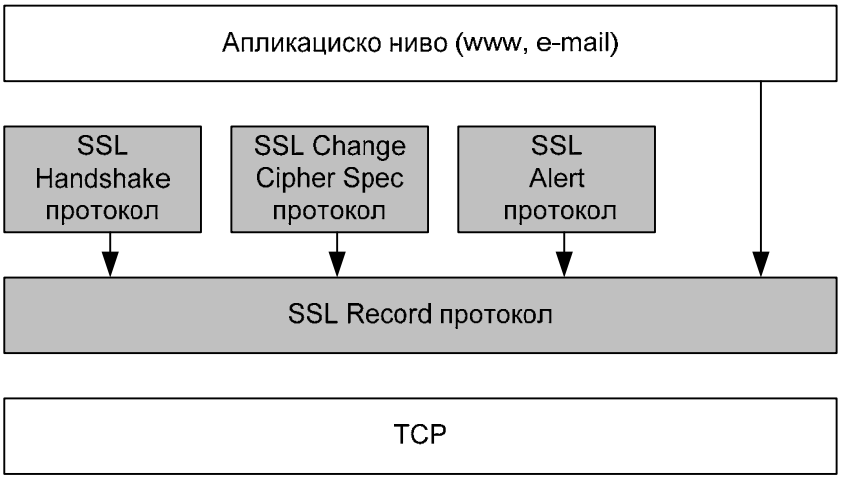
9.7 Интернет сигурност на транспортно ниво

За обезбедување на Интернет сигурност на транспортно ниво (според протоколниот стек) постојат два главни протоколи: Secure Socket Layer (SSL) и Transport Layer Security (TLS). Постариот од двата протоколи е SSL кој иницијално бил равиен од компанијата Netscape во првата половина на 1990-те години. Од друга страна TLS е стандардизира верзија од IETF, кој е базиран во голем дел на SSL како негов претходник. Во 2011 година, SSL верзија 3 (драфт RFC од 1996 год.) е публикуван од IETF како историска верзија (RFC 6101), бидејќи SSL никогаш не станал официјален Интернет стандард.

И двата протоколи се сместени во протоколниот стек меѓу транспортното и апликациското ниво. Целта на овие протоколи е да обезбедат автентикација меѓу клиентот и серверот, како и интегритет на податоците кои се пренесуваат, [23]. На пример, апликациските програми како што е веб (HTTP) при користење на SSL всушност ги сместуваат податоците во SSL (или TLS) пакети. Во продолжение ќе говориме за SSL, при што треба да се има во предвид дека TLS е продолжение на SSL, дефинирано со RFC 2246 (за TLS верзија 1.0 од 1999 год., која што всушност е SSL верзија 3.1), потоа проширено со RFC 4346 (за TLS верзија 1.1 од 2006 год.), а подоцна и со RFC 5246 (за TLS верзија 1.2 од 2008 год.). Од тие причини често пати се користи терминологијата SSL за да се референцира на IP сигурноста на транспортно ниво која е дефинирана од IETF (која го вклучува и TLS), па во продолжение тука ќе го користиме ваквиот термин.

9.7.1 SSL архитектура

Архитектурата на SSL (протоколниот стек) е прикажана на слика 9.11. SSL го користи TCP протоколот на транспортно ниво за да обезбеди од крај до крај сигурносен сервис. Во овој случај не се работи за еден протокол, туку за неколку протоколи кои ја дефинираат архитектурата на SSL и кои извршуваат различни функционалности. Основниот SSL Record протокол обезбедува основна сигурност на протоколите од апликациското ниво. Најчесто користен апликациски протокол на апликациско ниво над SSL е HTTP протоколот, т.е. WWW (World Wide Web).



Слика 9.11 SSL архитектура

Дефинирани се три протоколи во SSL архитектурата, над основниот SSL Record протокол, а тоа се: SSL Handshake протоколот, SSL Change Cipher Spec протоколот и Alert протоколот. Носечкиот протокол е Record протоколот, кој ги пренесува пораките од останатите три SSL протоколи, како и податоците од апликацијата (на пример, веб).

Генерално, има два концепти на SSL, а тоа се SSL конекција и SSL сесија:

- **SSL сесија** (SSL session) е воспоставена асоцијација меѓу клиентот и серверот, што значи дека двете страни ги имаат разменето општите информации како што идентификаторот на сесијата, сертификатите за автентикација на секоја од страните (ако се користат, а тогаш тоа се X.509 сертификати), сигурносните параметри, методот на компресија (ако се користи), како и мастер клучот (master key, кој е 48 бајтен таен клуч кој е

познат и на клиентот и серверот). Еден пар клиент и сервер може да има и повеќе воспоставени сесии во исто време, но тоа ретко се користи (обично се воспоставува една сесија меѓу пар клиент - сервер).

- **SSL конекција** (SSL connection) се креира откако претходно е воспоставена SSL сесија. Според тоа, една конекција може да припаѓа само на една SSL сесија. Конекцијата е транспорт (според протоколниот OSI модел) која обезбедува даден тип на сервис (type of service). Конекциите се peer-to-peer врски. Така, во рамките на една сесија може да има повеќе воспоставени SSL конекции.

Слика 9.12 Handshake за SSL

Handshake протокол

Овој протокол ја започнува SSL комуникацијата меѓу клиентот и серверот. Составен е од 4 фази како што е прикажано на слика 9.12. Handshake протоколот овозможува да се автентицираат серверот кон клиентот и обратно, клиентот на

серверот (доколку е потребно), како и да се договорат начинот на шифрирање (encryption), MAC (Message Authentication Codes) алгоритмот, како и криптографските клучеви кои ќе се користат. Handshake протоколот се состои од серија на пораки кои се разменуваат меѓу клиентот и серверот (слика 9.12) со што воспоставуваат логичка конекција меѓусебе. Прво клиентот праќа client_hello порака која меѓу другото ја содржи највисоката верзија на SSL поддржана од клиентот, 32 бајтен случаен број, ID за сесијата (session ID), листа на поддржани методи за шифрирање (encryption methods), како и листа на поддржани методи за компресија (compression methods). Серверот одговара со server_hello порака која ја содржи SSL верзијата која ќе се користи (највисоката што ја поддржуваат и клиентот и серверот), 32 бајтен случаен број, session ID, како и методите за шифрирање и за компресија кои серверот одлучил дека ќе се користат (секако, тие мора да припаѓаат на соодветните листи за овие методи испратени од клиентот во client_hello пораката претходно). Понатаму, за автентикација на серверот, самиот сервер праќа кон клиентот сертификат (Certificate), кој содржи т.н. јавен клуч на сертификатот на серверот и root сертификатот на тој што го издал сертификатот (Certificate Authority). По праќањето на сертификатот од страна на серверот, клиентот одлучува дали да му верува или не на серверот, при што во оваа фаза може да биде прашан и самиот корисник (доколку претходно не специфицирал како да реагира клиентот во таа ситуација за сертификат од даден сервер). Ако серверот сака да го автентицира клиентот (што не е задолжително) тогаш тоа го бара со пораката certificate_request на што клиентот одговара со сертификат порака шифрирана со јавниот клуч на серверот. Серверот го означува крајот на hello делот, по што клиентот ги праќа информациите неопходни за шифрирање на пораките (т.н. криптографски тајни клучеви).

Change Cipher Spec протокол

Ова е протокол кој се состои само една порака која се разменува на крајот на handshake делот на SSL, по што се испраќа пораката за завршување на SSL handshake. Зошто постои оваа порака? Главна улога е да означи до кога оди размената на параметрите и тајните клучеви за шифрирање меѓу клиентот и серверот, бидејќи по испраќањето на оваа порака договорените параметри и клучеви (меѓу клиентот и серверот во handshake делот) се зачувуваат понатаму непроменети и се користат понатаму од Record протоколот за потпишување/потврдување (sign/verify) и

шифрирање/дешифрирање (encrypt/decrypt) на пораките.

Alert протокол

SSL го користи овој протокол за известувања за грешки или разни аномалии. Притоа се користи една порака која го опишува проблемот и нивото на алармот - предупредување (warning) или фатален проблем (fatal).

Record протокол

Record протоколот ги пренесува пораките од погорните SSL протоколи (слика 9.11) или од апликациските протоколи кои користат TCP на транспортно ниво (HTTP, POP3, FTP, итн.). Протоколот ги фрагментира податоците кои ги добива, т.е. ги дели на повеќе рамки со максимум 2^{14} = 16 384 бајти податочен дел во секоја рамка. Секој ваков дел опционално може да биде компресиран. На компресираната порака се пресметува Message Authentication Code (MAC), а потоа компресираниот дел и MAC делот се шифрираат. На крај, на рамката се додава заглавие кое е составено од 5 бајти, каде што:

- Првиот бајт е индикатор за протоколот од повисоко ниво кој ќе го процесира дадениот фрагмент (на пример, HTTP).
- Вториот бајт е главната верзија на SSL (major version). На пример, за SSLv3.1 овој бајт ќе ја содржи вредноста 3.
- Третиот бајт е споредната верзија на SSL (minor version). На пример, за SSLv3.1 вредноста на овој бајт ќе биде 1.
- Последните два бајти (т.е. 16 бити) од заглавието ја содржат големината на податочниот дел на фрагментот во број на бајти (за што се доволни 14 бити).

9.7.2 Употреба на SSL

Од аспект на употребата, протоколите за SSL/TLS може да ги користи секој протокол на апликациско ниво кој користи TCP на транспортно ниво. Во пракса,

најчесто SSL/TLS се користи за заштита на податоците кои се пренесуваат преку HTTP протоколот (т.е. вебот), како и за протоколите кои се користат за електронска пошта, SMTP и POP3. Имплементацијата на HTTP врз SSL/TLS протоколите се нарекува HTTP Secure (HTTPS). Соодветно, URL записот во DNS системите за серверите кои користат HTTPS започнува со "https://..". Притоа, нема никакви промени кај HTTP протоколот, единствено се додава SSL/TLS протоколната архитектура (меѓу HTTP и TCP протоколите) за да се добие HTTPS. За HTTPS се користи стандардно портата 443, додека за HTTP (без SSL/TLS) се користи портата 80 (погледни во глава 5).

Слично на HTTP, може да се користат и протоколите за електронска пошта преку SSL/TLS. Така, кога SMTP се користи преку SSL/TLS се нарекува SMTP Secure и ја користи портата 465 (SMTP ја користи портата 25), а доколку се користи POP3 преку SSL тогаш се добива POP3 Secure каде што се користи портата 995 (стандардната порта за POP3 протоколот е 110).

На овој начин, со користење на SSL/TLS архитектурата на двата најважни Интернет сервиси кои се "родени" како Интернет технологии, вебот (WWW) и електронската пошта (застапена преку SMTP и POP3 протоколите), се овозможува истите сервиси (т.е. апликации) по потреба да се користат и со стандардизиран начин на автентикација и заштита на интегритет на податоците кои се пренесуваат меѓу клиентот и серверот. Тоа овозможува денес да се користи сигурен начин на праќање и добивање на e-mail пораките (со STMP и POP3 преку SSL/TLS), како и користењето на веб технологијата (со HTTPS) за пристап до чувствителни податоци, како што се пристап до веб-базирана електронска пошта, заштита при најава на различни веб сајтови кои нудат одредени типови сервиси (на пример, социјални мрежи), проверка на банкарски податоци, плаќање со кредитни картички преку Интернет, трансакции преку Интернет, итн.

9.8 Дискусија

Во време кога сите телекомуникациски сервиси се префрлуваат во Интернет, сигурноста добива уште повеќе на значење. Под сигурност подразбираме заштита на доверливоста (да не може трета страна да ја дознае содржината на пораките што се пренесуваат), автентикација на корисниците (да се идентификува корисникот на

дефиниран начин), да се зачува интегритетот на податоците кои се пренесуваат, како и да се обезбеди достапност на сервисот во најголем дел од времето. За таа намена постојат стандардизирани решенија кои се дефинирани на различни протоколни нивоа, па според тоа обезбедуваат заштита на различни сегменти од дадена IP мрежа.

Стандард за автентикација на корисниците кој најмногу се користи денес е RADIUS протоколот, кој овозможува AAA (Authentication, Authorization, Accounting) за корисниците кои пристапуваат до Интернет преку некоја пристапна мрежа. Во иднина, RADIUS ќе биде заменет од неговиот наследник DIAMETER.

Сигурносни решенија кои се користат за заштита на ниво 2 (ниво на линк) се 802.1X за сите пристапни мрежи, како и WPA и WEP за безжичен пристап. Притоа заштитата е само на ниво на линк, но може да се комбинира протоколот на ниво на линк (на пример, 802.1X) со протокол за AAA, како што е RADIUS, преку користење на EAP (Extensible Authentication Protocol) рамката.

На мрежно ниво стандардизирано решение за заштита е IPsec кое најчесто се користи во комбинација со VPN (Virtual Private Networks). Денес најголемиот дел од транспортните мрежи користат IPsec/VPN решенија за заштита на интегритетот на Интернет сообраќајот што го пренесуваат.

Меѓу мрежното и апликациското ниво, на транспортното ниво, стандардизиран начин на заштита од крај до крај е употребата на SSL/TLS (Secure Socket Layer/ Transport Layer Security) протоколната архитектура, која може да се користи за Интернет апликациите што го користат TCP. SSL/TLS најчесто се употребува за Интернет сервисите веб (како HTTP Secure - HTTPS) и електронска пошта (за SMTP и POP3 протоколите). Примената на ова сигурносно решение зависи само од крајните елементи (на пример, HTTPS клиентот и серверот), односно не зависи од попатните мрежни јазли (рутери и комутатори). На овој начин се овозможува сервис провајдерите да обезбедат заштита на доверливоста и интегритетот на податоците кои се пренесуваат без да зависат притоа од IP мрежите преку кои истите се пренесуваат.

Глава 10

Заклучок – интеграција на IP и класичните телекомуникации

Оваа книга е посветена на Интернет технологиите, кои имаат се поголемо влијание во секојдневниот живот и во развојот на т.н. информациско општество (information society). Меѓутоа, до 90-те години доминантна улога во светот на телекомуникациите играа два сервиса: телефонијата и телевизијата (на која може да се додаде и радио дифузијата). Секој од овие сервиси се пренесуваше во одделни мрежи изградени специјално за секој од сервисите посебно и прилагодени на неговите карактеристики (на пример: соодветен фреквентен опсег кој е потребен за да се пренесе сигналот од или кон корисникот). На почетокот овие два сервиса беа аналогни и овозможени од аналогни системи. Меѓутоа, од 1970-те години започна дигитализацијата на телефонските мрежи, така што во првата декада на 21-от век е реткост да се наиде на аналогна телефонска мрежа (иако пристапот преку класичен телефон до телефонските централи и понатаму постои по аналогна линија, преку бакарна парица). Дигитализацијата на телевизијата е малку поместена во време во однос на телефонијата, така да според ITU треба целосно да бидат дигитализирани терестријалните телевизиски сигнали до 2015 година (во некои развиени земји овој рок е неколку години порано). Ако моментот на читање на оваа книга е по јуни 2015 година, тогаш веројатно тој процес е завршен.

Кога сите сервиси ќе се пренесуваат со дигитални сигнали, кои се поворки од дигити (во најчест случај тоа се бити), тогаш се губи потребата да имаме различни

мрежи за различните типови сервиси, бидејќи независно дали ќе пренесуваме фајлови (текст, слики, музички или видео клипови, итн.) или говор или телевизија, ќе се пренесуваат низ мрежата дигити т.е. бити. Но, останува потребата да имаме одредени гаранции за квалитетот на сервисите (т.е. услугите) како што се доцнењето, загубите или битските грешки, џитерот, битскиот проток (во бити/сек) по даден корисник.

Истовремено со овие процеси на дигитализација, кои се одвиваа одделно за телефонијата и за телевизијата и радиото во изминативе две-три децении, се развиваше Интернет како мрежа за пренос на податочни мултимедиски содржини, базирана на едноставност и ниска цена на мрежните елементи (комутатори, рутери) и без експлицитна поддршка за квалитет на сервисите за разлика од тоа што е обезбедено во класичните телефонски мрежи и мрежите за ТВ и радио дифузија. Токму принципот best-effort во Интернет, што значи дека секој IP пакет ќе биде примен во мрежата и секоја IP конекција што ќе се побара ќе биде воспоставена, се главната причина за успехот на IP, но истовремено и главниот предизвик за целосна интеграција на класичните телекомуникации (телефонијата и телевизијата) со Интернет. Но, со обезбедување на механизми за контрола на квалитет во пристапната IP мрежа во прва рака, а потоа и од крај до крај во Интернет мрежата, се создаваат услови за целосна миграција на класичните и најстарите телекомуникациски сервиси, телефонијата и телевизијата/радиото, кон Интернет. Секако, за секој од сервисите е потребен и соодветен битски проток, кој доста се разликува од сервис до сервис. Така, на пример, за говорот се потребни од неколку kbit/s до неколку десетици kbit/s зависно од квалитет со кој ќе се дигитализира говорот, додека за ТВ со квалитет на денешната аналогна ТВ се потребни околу 2-3 Mbit/s, па се до над 5-6 Mbit/s за HDTV (High Definition TeleVision).

Процесот на интеграција на Интернет и класичните телекомуникации е започнат, но многу веројатно е дека ќе се одвива низ повеќе децении кои следуваат. Првата фаза во таквиот интегративен процес е понуда на трите сервиси: Интернет, телефонија и телевизија, преку иста пристапна мрежа (на пример: преку кабелска мрежа, или преку безжична мрежа како што е WiMAX и сл.). Притоа, во прва инстанца се интегрира телефонијата со Интернет преку користење на VoIP (Voice over IP), како што беше дискутирано во глава 7 од оваа книга, а телевизијата ќе оди во други фреквентни опсези (по други канали), а по истиот медиум (на пример: преку истиот кабел). Во втора инстанца се интегрира и телевизијата со Интернет. Денес постојат форми на IPTV кои се целосно споредливи со она на што е навикнат корисникот во

класичната ТВ, бидејќи за да се придобијат корисниците треба да им се понуди најмалку истиот квалитет и дополнителни можности (на пример: интеракција со содржината на програмата, креирање на ТВ програми според кориснички профили итн., постојат многу можности и идеи).

Паралелно со овие процеси на интеграција на Интернет и класичните телекомуникации, настанува и пенетрација на Интернет технологиите во области во кои комуникациите воопшто не биле присутни порано, како што се апаратите во домаќинството, елементите во автомобилот, состојбите на дрвјата во овоштарниците или насадите на нивите или состојбата на болните во болниците (кои се следат преку сензори поврзани на Интернет) итн. итн., па во иднина бројот на предмети приклучени на Интернет (Internet of Things) многукратно ќе го надмине бројот на луѓе корисници на Интернет.

А понатаму, само имагинацијата е лимитот...

Литература

[1] B. Forouzan, “TCP/IP Protocol Suite”, Mcgraw-Hill, 2010.

[2] V. Cerf, Y. Dalal, C. Sunshine, "Specification of Internet Transmission Control Program", RFC 675, декември 1974.

[3] J. Postel, “Internet Protocol”, RFC 791, септември 1981.

[4] J. Postel, "Transmission Control Protocol", RFC 793, септември 1981.

[5] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0", RFC 1945, maj 1996.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, june 1999.

[7] S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460, декември 1998.

[8] K. Thompson, G.J. Miller, R. Wilder, “Wide-area Internet Traffic Patterns and Characteristics”, IEEE Network, November/December 1997.

[9] Douglas Comer, “Internetworking with TCP/IP- Principles, Protocols and Architectures, Fourth Edition”, Prentice Hall, 2000.

[10] R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, март 1997.

[11] Matthew G. Naugle, “Illustrated TCP/IP”, John Wiley and Sons, Inc., 1998.

[12] V. Jacobson, “Congestion Avoidance and Control,” ACM SIGCOMM’88, August 1988.

[13] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, january 1997.

[14] M. Mathis, S. Floyd, A. Romanow, "TCP Selective Acknowledgement Option", RFC 2018, октомври 1996.

[15] W. Richard Stevens, “Unix Network Programming”, Prentice Hall, New Jersey 1990.

[16] Gary R. Wright, W. Richard Stevens, "TCP/IP Illustrated, Volume 2: the Implementation", Addison Wesley, 1995.

[17] J. Postel, "Simple Mail Transfer Protocol", RFC 821, август 1982.

[18] D. Crocker, "Standard for The Format of Arpa Internet Text Messages", RFC 822, август 1982.

[19] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, ноември 1996.

[20] J. Myers, M. Rose, "Post Office Protocol - Version 3", RFC 1939, мај 1996.

[21] M. Crispin¸ "Internet Message Access Protocol - Version 4", RFC 1730, декември 1994.

[22] L. Parziale, D. Britt, C. Davis, J. Forrester, W. Liu, C. Matthews, N. Rosselot,, "TCP/IP Tutorial and Technical Overview, 8th Edition", IBM Redbooks, декември 2006.

[23] William Stallings, "Data and Computer Communications, Eighth Edition", Pearson Education, Inc., 2007.

[24] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, јули 2003.

[25] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, април 1998.

[26] Jon Crowcroft, Mark Handley, Ian Wakeman, "Internetworking Multimedia", Morgan Kaufmann, 1999.

[27] Andrew S. Tanenbaum, David J. Wetherall, "Computer Networks 5th Edition", Prentice Hall, 2010.

[28] Тони Јаневски, „Комутација и рутирање", Универзитет „Св. Кирил и Методиј", Скопје, 2011.

[29] E. Bryan Carne, "A Professional's Guide to Data Communication in a TCP/IP World", Artech House Inc., 2004.

[30] William C. Hardy, "VoIP Service Quality Measuring and Evaluating Packet-Switched Voice", McGraw-Hill, 2003.

[31] Toni Janevski, "Traffic Analysis and Design of Wireless IP Networks", Artech House Inc., 2003.

[32] Alan B. Johnston, "SIP - Understanding The Session Initiation Protocol, 2nd Edition", Artech House Inc., 2004.

[33] Henry Sinnreich, Alan B. Johnston, "Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol", Wiley, 2006.

[34] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", јуни 2002.

[35] Salman A. Baset, Henning G. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol", Infocom 2006, април 2006.

[36] W. Richard Stevens, “TCP/IP Illustrated, Vol. 1: the Protocols”, Addison Wesley, 1993.

[37] J. Richard Durke, "Network Management, Concepts and Practice: A Hands-on Approach", Prentice Hall, 2004.

[38] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)", RFC 1157, maj 1990.

[39] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", RFC 2578, април 1999.

[40] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, јуни 2000.

[41] C. Rigney, "RADIUS Accounting", RFC 2866, јуни 2000.

[42] C. Rigney W. Willats, P. Calhoun, "RADIUS Extensions", RFC 2869, јуни 2000.

[43] L. Blunk, J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, март 1998.

[44] S. Kent, K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, декември 2005.

[45] J. F. Kurose, K. W. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet, 3nd Edition", Prentice Hall, 2005.

Кратка биографија на Тони Јаневски

Д-р Тони Јаневски е редовен професор на Факултетот за електротехника и информациски технологии, Универзитет „Св. Кирил и Методиј“ во Скопје. Тој има дипломирано, магистрирано и докторирано на Факултетот за електротехника и информациски технологии во Скопје во 1996, 1999 и 2001 година, соодветно. Во периодот од 1996 до 1999 година како главен инженер во Мобимак (денес Т-Мобиле, Македонија), учествувал во дизајнирањето и имплементација на првата мобилна јавна мрежа во Република Македонија. Посебен придонес имал во планирањето и имплементацијата на радио мобилната мрежа на Мобимак. Во текот на 2001 година работел на истражувања во оптички комуникации во IBM T.J. Watson Research Center, Optical Link Design Department, во Yorktown Heights, Њујорк, САД. Во својата научно-истражувачка и апликативна дејност раководел со повеќе научно-истражувачки и апликативни проекти од областа на мобилните мрежи и интернет технологиите. Во периодот од 2005 до 2008 година бил член на Комисијата на Агенцијата за електронски комуникации на Република Македонија, каде што активно учествувал во воведувањето на низа на нови комуникациски технологии во Македонија, како што се WiMAX, 3G мобилните мрежи, како и на нови оператори и комуникациски услуги. Во мандатниот период 2008 - 2012 година, како и во периодот 2012 – 2016 година, е член на Сенатот на Универзитетот „Св. Кирил и Методиј“ во Скопје. Има публикувано повеќе голем број научни трудови во списанија и на меѓународни конференции. Проф. Тони Јаневски е автор на книгата “Traffic Analysis and Design of Wireless IP Networks”, објавена во 2003 година од реномираната издавачка куќа Artech House Inc, Boston, USA. Исто така, тој е автор на книгата „Комутација и рутирање“, издадена од Универзитетот „Св. Кирил и Методиј“ во Скопје во 2011 година. Автор е и на книгата „NGN Architectures, Protocols and Services“ издадена во април 2014 година од реномираната издавачка куќа John Wiley & Sons, UK. Одржал голем број меѓународни курсеви и предавања за ITU (International Telecommunication Union). Тој е Senior Member на меѓународна организација на инженери IEEE. Во 2012 година ја има добиено наградата „Гоце

Делчев“ како највисоко научно признание на Република Македонија. Исто така, ја има добиено наградата „Научник на годината“ на Универзитетот „Св. Кирил и Методиј“ за 2013 година. Области на негов истражувачки интерес се Интернет технологиите, безжичните и мобилните мрежи, мултимедиските мрежи, сервисите и квалитетот на сервисот, дизајнирање и моделирање на телекомуникациски мрежи, како и следната генерација на мрежи (NGN).

CIP - Каталогизација во публикација Национална и универзитетска библиотека "Св. Климент Охридски", Скопје

004.7:621.39(075.8)
004.738.5:621.39(075.8)

ЈАНЕВСКИ, Тони
Интернет технологии [Електронски извор] / Тони Јаневски. - Скопје : Универзитет "Св. Кирил и Методиј", 2015

Начин на пристап (URL): http://www.ukim.edu.mk/e-izdavastvo - Текст во ПДФ формат, содржи 274 стр.. - Наслов преземен од екранот. - Опис на изворот на ден 09.07.2015. - Кратка биографија на Тони Јаневски: стр. 273-274. - Библиографија: стр. 270-[272]

ISBN 978-9989-43-379-5

а) Информациско-комуникациски технологии - Интернет - Високошколски учебници б) Интернет - Примена - Високошколски учебници
COBISS.MK-ID 98987018