

P R E V O D

T R E Ć E G

Umrežavanje računara

Od vrha ka dnu sa
Internetom u fokusu

James F. Kurose

Univerzitet Masačusets, Amherst ♦

Keith W. Ross

Politehnički univerzitet, Bruklin

Umrežavanje računara
Od vrha ka dnu sa Internetom u fokusu

ISBN 86-7991-267-0

Autorizovan prevod sa engleskog jezika prvog izdanja knjige Computer Networking: A Top-Down Approach Featuring the Internet

Original Copyright© 2005. by Pearson Education, Inc. Copyright°
prevoda, 2005. RAF Računarski fakultet, Beograd i CET Computer
Equipract and Trade, Beograd

Sva prava zadržana. Nijedan deo ove knjige ne može biti reproducovan, snimljen, ili emitovan na bilo
koji način: elektronski, mehanički, fotokopiranjem, ili drugim vidom, bez pisane dozvole izdavača.
Informacije korišćene u ovoj knjizi nisu pod patentnom zaštitom. U pripremi ove knjige učinjeni su
svi napor da se ne pojave greške. Izdavač i autori ne preuzimaju bilo kakvu odgovornost za
eventualne greške i omaške, kao ni za njihove posledice.

Prevodioci

Jasna Gonda, Stanislav Koščal, dr Radomir Janković

Recenzent

prof, dr Stevan Milinković

Lektor

Milanka Stojanović-Vorkapić

Urednik

Radmila Ivanov

Izvršni urednik

Aleksandra Dimic

Tehnički urednik

Dušan Čašić

Organizator projekta

Vladan Mirković

Prelamač

Predrag Stanisavljević

Izdavač

RAF Računarski fakultet, Beograd, Knez Mihailova 6A/[tel.

011 627-613, 633-321; <http://www.raf.edu.yu>

CET Computer Equipment and Trade, Beograd, Skadarska 45

tel/fax: 011 3243-043, 3235-139, 3237-246; <http://www.cet.co.yu>

Za izdavača

Dragan Stojanović, direktor

Obrada korica Predrag
Stanisavljević

Tiraž

1000

Štampa

„Svetlost”, Čačak

O autorima

Jim Kurose

Jim Kurose je profesor računarskih nauka na Masačuselskom univerzitetu u Amherstu.

Dr Kurose je dobio brojna priznanja za svoj prosvetni rad; između ostalog, osam puta je nagradivan za dostignuća u nastavi samo od strane Nacionalnog tehničkog univerziteta, a slična priznanja je dobio i od Masačuselskog univerziteta i Severoistočnog udruženja visokoškolskih ustanova. Dobitnik je medalje Taylor Booth za obrazovanje koju dodeljuje IEEE, a poznat je i po uspešnom vođenju Masačuselske savezne inicijative za informatičku tehnologiju. Dobitnik je stipendije GE [General Engineering] koju dodeljuje CIC {Committee on Insfilational Corpotion}, IBM-ove nagrade za dostignuća u nastavi, kao i stipendije Lilly koja se dodeljuje za unapređenje predavackih sposobnosti.



Dr Kurose je bio glavni urednik časopisa *IEEE Transactions on Communications*, kao i *IEEE Transaction on Networking*. Dugi niz godina bio je aktivan u programskim, odborima IEEE Infocom, ACM SIGCOMM i ACM SIGMETRICS i obavlja funkciju : ' kopredsedavajućeg u tehničkim programima na tim konferencijama. Član je organizacija IEEE i ACM. U istraživanjima se bavi mrežnim protokolima i arhitekturom, mrežnim merenjima, senzorskim mrežama, multimedijiskom komunikacijom, modeliranjem i procenama performansi. Doktorat iz oblasti računarskih nauka odbranio je na Kolumbijskom univerzitetu. .

Keith Ross

Keith Ross je profesor računarskih nauka na Pojtehničkom univerzitetu u Brucklinu, koji je stao na čelo fondacije Leonarda J. Shusleka za računarske nauke. Od 1985. do 1998. godine bio je profesor na Odeljenju za sistemski inženjeringu no Pensilvanijskom univerzitetu. Od 1998. do 2003. godine bio je profesor na Odeljevju za multimedijiske komunikacije no insititu Eurecom u Francuskoj. Keith Ross je lokode glavni osnivač i prvi direktor organizacije VVimba koja se bavi razvojem tehnologija za prenos govora preko IP-a za tržište elektronskog učenja.

■ Dr Ross je objavio brojne istraživačke izveštaje i dve knjige.

Bio je član izdavačkih saveta mnogih časopisa, uključujući i *IEEE/ACM Transaction on Multimedijiskim programskim odbora, koo Što su ACM SIGCOMM i IEEE Infocom.* Bio je mentor za 1-5 doktorskih teza. Njegova predavačka i istraživačka interesovanja ; obuhvalju P2P sisteme, multimedijiski umrežavanje, mrežne protokole i slahastičke mreže. Doktorat je odbranio na Mičigenskom univerzitetu.



Uvod

Dobrodošli u treće izdanje knjige *Umrežavanje računara: Od vrha ka dnu sa Internetom u fokusu*. Od kada se pre otprilike dve godine pojavilo prvo izdanje, ova knjiga je prihvaćena na stotinama koledža i univerziteta, prevedena je na više od 10 jezika i koristile su je hiljade studenata, ali i stručnjaka širom sveta. Od veoma velikog broja čitalaca dobili smo povratne informacije i to su najčešće bile pohvale.

Jedan od osnovnih aduta ove knjige, po našem ubedenju, jeste njen svež pristup problematici umrežavanja računara. Zbog čega insistiramo na svežem pristupu? Zbog toga što smo poslednjih godina svedoci dve revolucionarne promene u oblasti umrežavanja kojima nije posvećena odgovarajuća pažnja u radovima o umrežavanju objavljenim tokom 1980-ih i 1990-ih. Najpre, Internet je potpuno osvojio oblast računarskih mreža. Svaka iole ozbiljnija rasprava o umrežavanju mora da ima Internet u fokusu. Zatim, u poslednjih nekoliko godina izuzetan pomak dogodio se na polju mrežnih usluga i aplikacija. Što se može zaključiti na osnovu pojave Weba, sveopšte prisutnosti elektronske pošte, protoka audio i video signala u realnom vremenu, Internet telefona, trenutne razmene poruka, aplikacija za mreže ravноправnih računara, kao i onlajn kupovine i prodaje.

Staje novo u trećem izdanju?

Iako su u ovom, trećem, izdanju knjige načinjene mnoge izmene, u knjizi je ipak zadržano ono što smatramo njenim najznačajnijim aspektima, a predavači i studenti koji su koristili prvo izdanje potvrdili su daje zaista tako. Ukratko, to su: pristup od vrha ka dnu, fokus na Internetu, ravnopravnost teorijskih principa i vežbanja, pristupačan stil i karakteristično objašnjenje umrežavanja.

Međutim, u trećem izdanju napravili smo i neke značajnije izmene - primera radi, ubacili smo potpuno novo poglavje o bežičnim i mobilnim mrežama. Danas smo svedoci velikih promena u načinu na koji korisnici pristupaju Internetu i njegovim uslugama. Korisnici sada ostvaruju bežični pristup Internetu iz svojih kancelarija, domova i javnih ustanova. Čitav niz uređaja, kao što su laptop i PDA računari i telefoni, omogućavaju vam da pristupite Internetu dok ste na putu ili u pokretu. U poglavlu koje smo posvetili bežičnim i mobilnim tehnologijama pronaći ćete detaljan prikaz standarda 802.11, zatim prikaz pristupa Internetu putem mobilnih telefona, kao i iscrpljeno razmatranje mobilnosti u kontekstu Interneta i mreža mobilne telefonije. Sa dodavanjem ovog novog poglavlja, u udžbeniku sada postoje četiri napredna i specijalizovana poglavља. Čija su tema bežične i mobilne mreže, zatim multimedijalne mreže, mrežna bezbednost i upravljanje mrežama.

Drugi značajan dodatak predstavlja skup praktičnih Ethereal laboratorijskih vežbi. Ethereal je besplatna alatka za pregledanje i analizu paketa koja se nalazi u javnom domenu i može da se koristi pod svim popularnim operativnim sistemima,

uključujući tu i najrasprostranjenije operativne sisteme tipa Wind6ws. Iz izuzetno bogate funkcionalnosti ove alatke izdvajamo intuitivni korisnički interfejs i mogućnost analize blizu 400 protokola. Osim starih i novih programerskih zadataka u knjizi sada postoji i šest Ethereal laboratorijskih vežbanja koja su uskladena sa gradivom i koja svaki student može da provežba na svom računaru. (U budućnosti ćemo dodati još ovakvih laboratorijskih vežbanja.) Tokom ovih vežbanja studenti mogu da posmatraju mrežne protokole na delu i da prate interakciju između entiteta protokola koji se izvršavaju na njihovim računarima i drugih takvih entiteta koji se izvršavaju negde na Internetu. Dakle, ove vežbe omogućavaju učenje kroz praktičan rad. Osim toga, dodali smo i dva nova programerska zadatka: zadatak u vezi sa protokolom UDP i zadatak u vezi sa proksi veb serverom.

Ovo nije sve. Ažuriranjem trećeg izdanja pokušali smo da obuhvatimo sve značajne i brze promene koje su se dogodile u oblasti umrežavanja u proteklih nekoliko godina. U pitanju su novi ili dopunjeni sadržaji o mrežama ravnopravnih računara, protokolima BGP i MPLS, mrežnoj bezbednosti, o rutiranju sa difuznim emitova-njem i adresiranju i prosleđivanju na Internetu. Poglavlje 4 sada ima malo izmenjenu strukturu zato što smo želeli da što jasnije izložimo uloge prosleđivanja i rutiranja, kao i njihovu interakciju sa mrežnim slojem.

Ciljna grupa

Ovaj udžbenik zamišljen je kao prva nastavna jedinica o umrežavanju računara i može se koristiti i u računarstvu i u elektrotehnici. U pogledu programskih jezika u ovoj knjizi se pretpostavlja da student ima iskustva u radu sa programskim jezicima C, C++ ili Java. Polaznik koji je programirao samo u programskim jezicima C ili C++, a nije u Javi, neće imati nikakvih problema u praćenju ovog materijala i pored toga stoje on predstavljen upravo u kontekstu programskega jezika Java. Iako se ova knjiga odlikuje većom preciznošću i analitičnošću u odnosu na mnoge druge uvodne udžbenike o umrežavanju, u njoj nećete naći mnogo matematičkih koncepata sa kojima se već niste sreli u gimnaziji. Namerno smo se potrudili da izbegnemo sve naprednije proračune, verovatnoću i koncepte stohastičkih procesa. Prema tome, ova knjiga odgovaraće studentima na redovnim studijama kao i onima koji su na početku postdiplomskih studija. Isto tako, ona bi mogla da bude izuzetno korisna stručnjacima koji već rade u telekomunikacionoj industriji.

Sta je jedinstveno u ovoj knjizi?

Umrežavanje računara je izuzetno složena tema koja obuhvata mnoštvo koncepata, protokola i tehnologija, kao i njihove ništa manje složene interakcije. Da bi mogli da se izbore sa ovom izuzetnom složenošću, razni autori su svoje tekstove o umrežavanju računara organizovali oko „slojeva“ mrežne arhitekture. Slojovita organizacija omogućava studentima da uče svu složenost umrežavanja računara. Drugim recima, učeći o konkretnim konceptima i protokolima u jednom delu arhitekture oni i dalje mogu da vide čitavu sliku i način na koji se sve komponente uklapaju u nju.

Primera radi, veliki broj tekstova organizovan je oko sedmoslojne arhitekture referentnog modela OSI. Gledano iz pedagoške perspektive, ovakav slojevit pristup je veoma poželjan u nastavi. Međutim, uočili smo da tradicionalni pristup od dna ka vrhu, odnosno od fizičkog sloja ka aplikativnom, u savremenoj nastavi iz oblasti umrežavanja računara nije najpodesniji.

Pristup od vrha ka dnu

Pre otprilike 4 godine ovaj udžbenik je doneo jednu veoma bitnu novinu - pristup od vrha ka dnu. To znači da ćemo ovde početi od aplikativnog sloja i zatim ići naniže ka fizičkom. Ovakav pristup ima nekoliko važnih prednosti. Pre svega, u njemu je naglasak na aplikativnom sloju koji je u umrežavanju predstavlja segment sa najvećom ekspanzijom. Mnoge od nedavnih revolucionarnih pojava u oblasti umrežavanja - uključujući Web, protokol audio i video signala u realnom vremenu (*streaming*), i distribuciju sadržaja - odigrale su se upravo na aplikativnom sloju. Naglasak na aspektima aplikativnog sloja od samog početka izlaganja, razlikuje se od pristupa u drugim tekstovima slične tematike, u kojima je mrežnim aplikacijama, njihovim zahtevima, paradigmama aplikativnog sloja (odnosno klijent/server) i programskim interfejsima aplikacije posvećeno veoma malo (ili nimalo) vremena.

Zatim, kao predavači, uverili smo se da izlaganje o mrežnim aplikacijama na početku obrade ove nastavne oblasti značajno motiviše studente za druge nastavne jedinice. Studente veoma zanima na koji način funkcionišu mrežne aplikacije (na primer, aplikacije za elektronsku poštu ili Web) sa kojima se svakodnevno susreću. Onog trenutka kada studenti razumeju način rada ovih aplikacija, sa njima se može pričati i o mrežnim uslugama koje podržavaju ove aplikacije. Nakon toga oni mogu da razmišljaju i o različitim načinima na koje se ove usluge obezbeđuju i implementiraju na nižim slojevima. Dakle, rani prikaz aplikacija deluje kao jak motivacioni faktor za Čitanje ostatka teksta.

Konačno, pristup od vrha ka dnu omogućava predavačima da uvedu razvoj mrežnih aplikacija na samom početku nastave iz ove oblasti. Pored toga što mogu da nauče na koji način funkcionišu popularne aplikacije i protokoli, studenti mogu da vide i koliko je jednostavno da se naprave vlastite mrežne aplikacije i protokoli aplikativnog sloja. Dakle, pristupom od vrha ka dnu postignuto je da se studenti veoma rano u nastavi upoznaju sa pojmovima kao što su programski interfejs aplikacije (API), modeli usluga i protokoli - veoma važnim konceptima na koje ćemo se vraćati u svim narednim poglavljima. Primerima o programiranju *soketa* u Javi pokušali smo da istaknemo centralne koncepte, a da pri tom ne zbumimo studente, previše složenim kodom. Studenti elektrotehnike i računarstva ne bi trebalo da imaju problema u praćenju ovog koda.

Fokus na Internetu

Kao što možete da zaključite iz naslova ove knjige, u njoj ćemo se u značajnoj meri baviti Internetom, čiji su arhitektura i protokoli iskorišćeni kao sredstvo za proučavanje nekih fundamentalnijih koncepata umrežavanja. Naravno, ovde ćete pronaći

koncepte i protokole koji pripadaju drugim mrežnim arhitekturama, ali je naglasak očigledno na Internetu, što se vidi i iz organizacije same knjige koja prati petoslojnju arhitekturu Interneta (aplikativni, transportni, mrežni sloj, zatim sloj veze i, na kraju, fizički sloj).

Druga prednost isticanja Interneta jeste to što je većina studenata elektrotehnike i računarstva prilično nestrpljiva da sazna što više o Internetu i njegovim protokolima. Oni svakodnevno koriste Internet (makar za slanje elektronske pošte i krstarenje po Webu) i sasvim sigurno su mnogo puta čuli daje Internet revolucionarna tehnologija koja menja svet. Imajući u vidu ogroman značaj Interneta, prirodno je da su studenti veoma nestrpljivi da proniknu u njegove tajne. Zahvaljujući tome, predavaču je sasvim jednostavno da pažnju studenata veže za neke osnovne principe kada kao primer koristi Internet.

Insistiranje na principima

Dva jedinstvena aspekta ove knjige - pristup od vrha ka dnu i fokus na Internetu -pojavljuju se u njenom podnaslovu. Da smo u podnaslov mogli da uklonimo i treću ključnu reč, to bi svakako bila reč *principi*. Polje umrežavanja je već dovoljno zrelo daje u njemu moguće identifikovati nekoliko fundamentalnih aspekata. Primera radi, u fundamentalne aspekte transportnog sloja ubrajaju se pouzdana komunikacija kroz nepouzdani mrežni sloj, zatim uspostavljanje/raskidanje veze i sinhronizacija, zagruženje mreže i kontrola toka, kao i multipleksiranje. U mrežnom sloju, dva fundamentalno važna aspekta su pronalaženje „dobrih“ putanja između dva rutera i rukovanje vezama između većeg broja heterogenih sistema. U sloju povezivanja podataka, fundamentalni problem predstavlja zajedničko korišćenje kanala za višestruki pristup. Na polju mrežne bezbednosti, tehnike za obezbeđivanje tajnosti, proveru autentičnosti i integriteta poruka zasnivaju se na osnovnim principima šifrovanja. U ovoj knjizi identifikovani su fundamentalni aspekti umrežavanja, kojima je zatim posvećena odgovarajuća pažnja. Kada ih savladaju, studenti će steći znanje koje će biti relevantno i u budućnosti, onda kada aktuelni protokoli i standardi postanu deo prošlosti. Verujemo da će kombinacija korišćenja Interneta za motivi-sanje studenata, zatim isticanja određenih aspekata i traženje rešenja, omogućiti studentima da veoma brzo steknu znanja pomoću kojih će moći da razumeju praktično svaku mrežnu tehnologiju.

Web lokacija

Svim čitaocima ove knjige skrećemo pažnju i na sveobuhvatnu prateću web lokaciju <http://www.aw.com/kurose-ross> na kojoj će moći da pronađu:

- ◆ *Materijal za interaktivno učenje.* Na ovoj lokaciji ćete pronaći interaktivne Java aplete koji ilustruju ključne mrežne koncepte. Osim toga, ova web lokacija (preko pretraživača) obezbeđuje direktni pristup programima kao sto je, na primer,

Traceroute koji pokazuje putanje kojima se paketi kreću kroz Internet. Profesori ovaj interaktivni materijal mogu da iskoriste kao svojevrsne mini-laboratorije. Ova web lokacija takođe obezbeđuje direktni pristup mehanizmima za pretraživanje dokumenata o Internetu (*Internet Drafts*), kao i elektronskoj konferenciji na kojoj se razmatraju teme koje su obrađene u ovoj knjizi. Konačno, na ovoj web lokaciji postoje i interaktivni testovi koji omogućavaju studentima da provere svoje osnovno razumevanje izložene problematike.

- ◆ *Više od pet stotina veza sa relevantnim materijalom.* Kao što svi mi Internet entuzijasti znamo, veliki deo materijala koji opisuje Internet nalazi se upravo na Internetu. Potrudili smo se da na ovu web lokaciju postavimo URL-ove za najveći mogući broj referenci za ovu knjigu. Bibliografija je onlajn i biće ažurirana onako kako se adrese budu menjale i kako se bude pojavljivao novi materijal. U pitanju su adrese RFC dokumenata, časopisa i članaka sa konferencija, ali i lokacija više pedagoškog karaktera, uključujući i neke stranice o pojedinim aspektima Internet tehnologije i članke koji izlaze u Časopisima za onlajn trgovinu. Materijal koji se krije iza ovih adresa profesori mogu da preporuče kao dodatnu ili čak obaveznu literaturu.
- ◆ *Laboratorijski zadaci.* Na pomenutoj web lokaciji pronaći ćete puno detaljnih programerskih zadataka, kao i Ethereal laboratorijska vežbanja. U programer-ske zadatke spadaju pravljenje višenitnog web servera, pravljenje klijenta za elektronsku poštu sa GUI interfejsom, programiranje predajne i prijemne strane protokola za pouzdani prenos podataka, programiranje algoritma distribuiranog rotiranja, kao i mnogi drugi. Na web lokaciji ćete pronaći i uputstva za Ethereal laboratorijske vežbe o kojima smo govorili ranije.

Pedagoški aspekti

Svaki od nas dvojice predaje umrežavanje računara već skoro 20 godina. U ovu knjigu ugrađeno je, ukupno, preko 30 godina predavačkog iskustva sa više od 3 000 studenata. Osim toga, u navedenom periodu bavili smo se i aktivnim istraživačkim radom na polju umrežavanja računara. (U stvari, Džim i Kit su se upoznali na magisterskim studijama o umrežavanju, na predmetu koji je na Univerzitetu Kolumbija 1979. godine držao Miša Švarc.) Upravo zato mislimo da nam sve to zajedno daje precizan pogled na to gde se umrežavanje trenutno nalazi i kako će se razvijati u budućnosti. Trudili smo se da izbegnemo približavanje ove knjige vlastitim istraživačkim projektima, a ukoliko vas oni zanimaju, uvek možete da posetite naše web lokacije. Dakle, pred vama je knjiga o savremenom umrežavanju računara - o savremenim protokolima i tehnologijama, kao i o principima na kojima se ti protokoli i tehnologije zasnivaju. Ubeđeni smo da učenje (i predavanje!) o umrežavanju može biti i te kako zabavno. Nadamo se da su šale, korišćenje analogija i primjeri iz stvarnosti učinili ovu knjigu zabavnijom i lakšom za čitanje.

Istorijski osvrti i principi u praksi

Oblast umrežavanja računana, počev od kraja 1960-ih godina, ima bogatu i fascinantnu istoriju, i mi smo se potrudili da vam je kroz ovaj tekst ukratko ispričamo. U poglavlju 1 nalazi se poseban odeljak o istoriji, dok će u ostalim poglavljima pronaći kratke istorijske osvrte. U ovim odeljcima spominjemo pronalazak komuti-ranja paketa, evoluciju Interneta, pojavljivanje svojevrsnih mrežnih giganata kao što su kompanije Cisco i 3Com, kao i mnoge druge značajne događaje. Za većinu studenata ove informacije biće podsticajne. U svakom od poglavlja koja slede postoje izdvojeni segmenti u kojima su istaknuti važni principi iz oblasti umrežavanja raču-nara. Pomoću ovih informacija studenti će lakše savladati neke od fundamentalnih koncepta savremenog umrežavanja.

Intervjui

U ovoj knjizi postoji još jedna originalna stvar koja bi trebalo da inspiriše i motiviše studente, a to su intervjuji sa priznatim inovatorima na polju umrežavanja kao što su Leonard Klajnrok, Tim Berners-Li, Seli Flojd, Vint Serf, Sajmon Lam, Čarli Per-kins, Hening Šulcrin, Stiven Belovin i Džef Kejs.

Dodatak za predavače

Kako bismo pomogli profesorima da šio lakše prebrode prelaz, obezbedili smo dodatni komplet za kvalifikovane predavače. Sav ovaj materijal nalazi se u de!u veb lokacije koji je namenjen profesorima - [hUp://www.aw.com/kurose-ross](http://www.aw.com/kurose-ross). Da biste pristupili ovom segmentu naše veb lokacije, obratite se odeljenju za prodaju kompanije Addison-Wesley ili to učinite putem elektronske pošte na adresu aw.cse@aw. com.

- ◆ *PowerPoint slajdovi.* Na veb lokaciji koja prati ovu knjigu nalaze se PowerPo-int slajdovi koji veoma detaljno pokrivaju svako od osam poglavlja ove knjige. U njima su (umesto monotonih tekstualnih nabranja) upotrebljeni grafički elementi i animacije što ih čini zanimljivim i vizuelno prijemčivim. Ove Power-Point slajdove profesori mogu da menjaju i tako prilagode svojim konkretnim potrebama. Veliki broj ovih slajdova upravo i potiče od drugih predavača koji su držali nastavu koristeći našu knjigu.
- ◆ *Rešenja domaćih zadataka.* Na pomenutoj veb lokaciji postoje i rešenja domaćih zadataka zadatih u ovoj knjizi. Ova rešenja su namenjena isključivo predavačima.

- ◆ *Diskusione grupe i doprinos drugih predavača.* Na veb lokaciji postoji i deo u kome predavači mogu da postavljaju svoje komentare, pitanja i odgovore. Na ovom mestu pronaći ćete i materijal prikupljen od drugih predavača koji takođe koriste našu knjigu.

Međuzavisnosti poglavlja

Prvo poglavlje ove knjige predstavlja zaokruženi prikaz umrežavanja računara. Predstavljanjem mnogih ključnih koncepta i termina, ovo poglavlje će vas pripremiti za sve ono što sledi u ostaku knjige. Drugim recima, sva ostala poglavlja knjige direktno se nadovezuju na materiju prvog poglavlja. Profesorima preporučujemo da, nakon poglavlja 1, redom ispredaju poglavlja od 2 do 5 i na taj način podvuku filozofiju od vrha ka dnu. Svako od ovih prvih pet poglavlja oslanja se na ona poglavlja koja su ispred njega.

Nakon što ispredaje prvih pet poglavlja, profesor će imati daleko vize slobode. Između poslednja četiri poglavlja ne postoje nikakve međuzavisnosti, što znači da mogu da se predaju bilo kojim redosledom. S druge strane, svako od njih oslanja se na materiju iz prvih pet poglavlja. Ustanovili smo da veliki broj predavača najpre predaje prvih pet poglavlja, ostavljajući poslednja četiri kao svojevrsnu poslasticu.

Poslednja napomena: Želeli bismo da čujemo i vaš glas

Predavače i studente želimo da podstaknemo u pravljenju novih Java apleta koji bi ilustrovali koncepte i protokole predstavljene u ovoj knjizi. Ukoliko imate aplet za koji mislite da bi bio zgodan za ovaj tekst, molimo vas da ga pošaljete autorima. One aplete (zajedno sa notacijom i terminologijom) koji su zaista podesni veoma rado ćemo postaviti na prateću veb lokaciju za ovu knjigu, zajedno sa odgovarajućim preporukama njihovih autora. Takođe želimo da podstaknemo predavače da nam pošalju nove probleme (i rešenja) za domaće zadatke koje ćemo objaviti u segmentu veb lokacije koji je namenjen predavačima.

Konačno, i studenti i predavači mogu da nam se obrate putem elektronske pošte i pošalju svoje komentare o ovoj knjizi. Izuzetno su nas obradovale reakcije iz svih delova sveta koje su pratile prvo izdanje ove knjige. Takođe možete da nam pošaljete i zanimljive URL-ove, ukažete na tipografske greške, možete da se suprotstavite bilo kojoj našoj tvrdnji i kažete nam šta, prema vašem mišljenju, funkcioniše, a šta ne. Vaši komentari i sugestije će nam svakako pomoći u izboru tema za sledeće izdanje ove naše knjige. Svu elektronsku poštu možete da nam pošaljete na adrese kurose@cs.umass.edu i ross@poly.edu.

Sadržaj

Poglavlje 1 Računarske mreže i Internet

	1
1.1 Staje Internet?	
1.1.1 Osnovne komponente Interneta	2
1.1.2 Opis iz aspekta usluga	5
1.1.3 Staje protokol?	
1.2.1 Krajnji sistemi - klijenti i serveri	9
1.2.2 Usluge sa konekcijom i bez nje	11
1.3 Jezgro mrežel4	
1.3.1 Komutiranje vodova i komutiranje paketa	14
1.3.2 Mreže sa komutiranjem paketa: mreže sa datagramima i mreže sa virtuelnim kolima	21
1.4 Pristup mreži i fizički medijumi	
1.4.1 Pristupne mreže	25
1.4.2 Fizički medijumi	31
1.5 Posrednici za Internet usluge i okosnice Interneta	34
1.6 Kašnjenje i gubitak paketa u mrežama sa komutiranjem paketa	37
1.6.1 Tipovi kašnjenja	37
1.6.2 Kašnjenje usled stajanja u redu i gubljenje paketa	40
1.6.3 Kašnjenja i rute na Internetu	43
1.7 Slojevi protokola i njihovi modeli usluga	
1.7.1 Slojevita arhitektura	45
1.7.2 Slojevi, poruke, segmenti, datagrami i okviri	50
1.8 Istoriski pregled umrežavanja i Interneta	
1.8.1 Razvoj komutiranja paketa: 1961 - 1972	52
1.8.2 Specijalne mreže i međusobno povezivanje mreža: 1972 - 1980	53
1.8.3 Veliki porast broja mreža: 1980- 1990	56
1.8.4 Eksplozija Interneta: poslednja decenija prošlog veka	56
1.8.5 Aktuelni trendovi	58
1.9 Rezime	59
Mapa knjige	60
Domaći zadatak: problemi i pitanja	61
Problemi	62

Teze za diskusiju	68	Poglavlje 3 Transportni sloj	183
Ethereal laboratorijska vežbanja	69	3.1 Usluge transportnog sloja	184
Intervju: Leonard Klajnrok	71	3.1.1 Odnos između transportnog i mrežnog sloja	187
		3.1.2 Kratak pregled transportnog sloja u Internetu	187
Poglavlje 2 Aplikativni sloj	73	3.2 Mulripleksiranje i demultiplexiranje	189
2.1 Principi rada mrežnih aplikacija	74	3.3 Prenos bez uspostavljanja konekcije: UDP	196
2.1.1 Arhitektura mrežnih aplikacija	75	3.3.1 Struktura UDP segmenta	199:
2.1.2 Komunikacija procesa	78	3.3.2 UDP kontrolni zbir	200
2.1.3 Protokoli aplikativnog sloja	81		
2.1.4 Koje su usluge potrebne aplikaciji?	82	3.4 Principi pouzdanog transfera podataka	203:
2.1.5 Usluge koje obezbeđuju Internet transportni protokoli	84	3.4.1 Pravljenje protokola za pouzdan transfer podataka	203,
2.1.6 Mrežne aplikacije o kojima ćemo govoriti u ovoj knjizi	87	3.4.2 Cevovodni pouzdani protokoli za transfer podataka	214
2.2 Web i HTTP	87	3.4.3 GBN	217.
2.2.2 Nepostojane i postojane veze	90	3.4.4 Selektivno ponavljanje	221
2.2.3 Format HTTP poruke	93		
2.2.4 Interakcija između korisnika i servera: kolačići	98	3.5 Transport sa konekcijom: TCP	228
2.2.5 HTTP sadržaj	100	3.5.1 TCP konekcija	231
2.2.6 Web keširanje	101	3.5.2 Struktura TCP segmenta	236
2.2.7 Uslovno preuzimanje	105	3.5.3 Procena vremena povratnog puta i tajm-aut	236
2.3 Transfer datoteke: protokol FTP	106	3.5.4 Pouzdani transfer podataka	239
2.4 Elektronska pošta na Internetu	109	3.5.6 Upravljanje TCP konekcijom	249
2.4.2 Poređenje sa protokolom HTTP	115		
2.4.3 Formati elektronske pošte i MIME	115	3.6 Principi kontrole zagušenja	254
2.4.4 Protokoli za prijem elektronske pošte	118	3.6.1 Uzroci i posledice zagušenja	260
2.5 DNS - Internetova usluga direktorijuma	123	3.6.2 Pristupi kontroli zagušenja	260
2.5.1 Usluge koje obezbeđuje DNS	123	3.6.3 Primer kontrole zagušenja pomoću mreže:	
2.5.2 Prikaz načina rada usluge DNS	126	Kontrola zagušenja ATM ABR	261
2.5.3 DNS zapisi i poruke	132	3.7 TCP kontrola zagušenja	264
2.6 P2P razmena datoteke	136	3.7.2 Modeliranje TCP kašnjenja	275
2.7 Programiranje soketa za protokol TCP	146	3.8 Rezime	284
2.7.1 Programiranje soketa za protokol TCP	147	Domaći zadatak: problemi i pitanja	
2.7.2 Primer klijentsko-serverske aplikacije u Javi	149	285	Problemi
2.8 Programiranje soketa za protokol UDP	156	287	diskusiju
2.9 Pravljenje jednostavnog web servera	164	294	zadatak
2.9.1 Funkcije web servera	164	295	Programerski
2.10 Rezime	169	Laboratorija	Ethereal:
Domaći zadatak: problemi i pitanja	170	Istražite	TCP
Problemi	171		295
Teze za diskusiju	177		
Zadaci sa programiranjem soketa	178	Intervju: Sali Flojd	297
Ethereal laboratorijske vežbe	180		
Intervju: Tim Bemers-Li	181	Poglavlje 4 Mrežni sloj	299
		4.1 Uvod	300
		4.1.1 Prosleđivanje i rutiranje	301
		4.1.2 Modeli mrežne usluge	304
		4.2 Mreže sa virtualnim kolima i sa datagramima	
		4.2.1 Mreže sa virtuelnim kolima	307
		4.2.2 Mreže sa datagramima	310

4.1.3 Poreklo usluga datagrama i virtualnog kola	313					
4.3 Šta ima u ruteru?						
4.3.1 Ulagani portovi	315					
4.3.2 Komutatorska mreža	318					
4.3.3 Izlagani portovi	320					
4.3.4 Gde dolazi do Čekanja u redu?	320					
4.4 Internet protokol (IP): Prosleđivanje i adresiranje na Internetu						
4.4.1 Format datagrama	325					
4.4.2 IPv4 adresiranje	331					
4.4.3 ICMP: Internet Control Message Protocol	342					
4.5 Algoritmi rutiranja						
4.5.1 Algoritam rutiranja prema stanju linkova	354					
4.5.2 Algoritam rutiranja sa vektorom rastojanja	358					
4.6 Rutiranje na Internetu						
4.6.1 Unutrašnji protokoli rutiranja autonomnih sistema na Internetu: R1P	371					
4.6.2 Unutrašnji protokoli rutiranja autonomnih sistema na Internetu: OSPF	374					
4.6.3 Rutiranje među autonomnim sistemima: BGP	378					
4.7 Difuzno i višezačno rutiranje						
4.7.1 Algoritmi difuznog rutiranja	385					
4.7.2 Višezačno upućivanje	391					
4.8 Rezime	400					
Domaći zadatak: problemi i pitanja	401					
Problemi	404					
Teze za diskusiju	412					
Programerski zadatak	412					
Laboratorijski zadatak	413					
Intervju: Vinion G. Cerf	414					
Poglavlje 5 Sloj veze podataka i lokalne mreže računara	417					
5.1 Sloj veze podataka: uvod i usluge	419					
5.1.1 Usluge koje obezbeđuje sloj veze	419					
5.1.2 Komuniciranje adaptera	422					
5.2 Tehnike za otkrivanje i ispravljanje grešaka						
5.2.1 Provere parnosli	425					
5.2.2 Metode kontrolnog zbiranja	427					
5.2.3 Ciklična provera redundantnosti (CRC)	428					
5.3 Protokoli višestrukog pristupa						
5.3.1 Protokoli sa deljenjem kanala	433					
5.3.2 Protokoli sa slučajnim pristupom	435					
5.3.4 Protokoli tipa „na koga je red“	442					
5.3.5 Lokalne računarske mreže (LAN-ovi)	443					
5.4 Adresiranje sloja linka						
5.4.1 MAC adrese	445					
5.4.2 Protokol razrešavanja adresa (ARP)	447					
5.4.3 Protokol za dinamičko konfigurisanje glavnog računara	451					
5.5 Ethernet						
5.5.1 Struktura Ethernetskog okvira	455					
5.5.2 CSMA/CD: Ethernetski protokol sa višestrukim pristupom	460					
5.5.3 Ethernet tehnologije	453					
5.6 Medusobne veze: labovi i komutatori						
5.6.1 Labovi	455					
5.6.2 Komutatori sloja linka	457					
5.7 PPP: protokol od tačke do tačke						
5.7.1 PPP uokvirivanje podataka	479					
5.7.2 PPP protokol kontrole linka (LCP) i protokoli kontrole mreže	480					
5.8 Vizuelizacija Jinka: mreža kao sloj veze	482					
5.8.1 Asinhroni režim prenosa (ATM)						
5.8.2 Višeprotokolna komutacija na osnovu oznaka (MPLS)	488					
5.9 Rezime	493	Domaći zadatak:	problemni i pitanja			
	494	Teze	za			
	498	Ethereal				
Intervju: Simon S. Lam						
Poglavlje 6 Bežične i mobilne mreže						503
6.1 Uvod						504
6.2 Bežični linkovi i mrežne karakteristike	508					CDMA
6.3 Wi-Fi: Bežični LAN-ovi tipa 802.11						509
6.3.1 Arhitektura 802.11						513
6.3.2 MAC protokol 802.11						514
6.3.4 Mobilnost unutar iste IP podmreže						517
6.3.5 802.11 i Bluetooth						526
6.4 Celularni pristup Internetu						528
6.4.1 Opšti pregled celularne arhitekture						529
6.4.2 Celularni standardi i tehnologije: kratak pregled						531
6.5 Upravljanje mobilnošću: principi	536					532
6.5.2 Rutiranje prema mobilnom Čvoru						540
6.6 Mobilni IP						546
Adresiranje						

6.7 Upravljanje mobilnošću u celularnim mrežama		
6.7.1 Rutiranje poziva prema mobilnom korisniku	552	
6.7.2 Predavanje u GSM-u	553	
6.8 Bežične mreže i mobilnost: posledice po protokole viših nivoa	556	
6.9 Rezime559	Domaći zadatak: problemi i pitanja	Problemi
559		diskusiju
560	Teze za	Ethereal
562	Laboratorija	
		562
		563
Intervju: Čarli Perkins		
Poglavlje 7 Multimedijsko umrežavanje	565	
7.1 Multimedijiske aplikacije umrežavanja	566	
7.1.1 Primeri multimedijskih aplikacija	566	
7.1.2 Prepreke za multimedije na današnjem Internetu	569	
7.1.3 Kako bi Internet trebalo da se razvija da bi bolje podržao multimedije?		
7.1.4 Kompresija zvuka i videa	572	
7.2 Protok memorisanog zvučnog signala i video signala u realnom vremenu	574	
7.2.1 Pristupanje zvuku i videu kroz veb server	576	
7.2.2 Slanje multimedija iz servera za protok signala u realnom vremenu ka pomoćnoj aplikaciji	578	
7.2.3 Protokol za protok signala u realnom vremenu (Real-Time Streaming Protocol, RTSP)	580	
7.3 Najbolja od najbolje moguće usluge: primer Internet telefona		
7.3.1 Ograničenja najbolje moguće usluge	585	
7.3.2 Uklanjanje treperenja kod primaoca zvuka	587	
7.3.3 Oporavljanje od gubitaka paketa	590	
7.3.4 Protok memorisanog zvučnog signala i video signala u realnom vremenu		
7.4 Protokoli za interaktivne aplikacije u realnom vremenu		
7.4.1 RTP	594	
7.4.2 RTP kontrolni protokol (RTCP)	599	
7.4.3 SIP	602	
7.4.4 H.323	608	
7.5 Distribuiranje multimediju ma: mreže za distribuciju sadržaja	610	
7.6 Dalje od najbolje mogućeg614	7.6.1 Scenario 1: Zvučna aplikacija od 1 Mb/s i FTP prenos	
		615
7.6.2 Scenario 2: Zvučna aplikacija od 1 Mb/s i FTP prenos visokog prioriteta		616
7.6.3 Scenario 3: Zvučna aplikacija koja se loše ponaša i FTP prenos		617
7.6.4 Scenario 4: Dve zvučne aplikacije od 1 Mb/s preko preopterećenog linka od 1,5 Mb/s		619
7.7 Mehanizmi za rasporedivanje i upravljanje		
7.7.1 Mehanizmi rasporedivanja '		621
7.7.2 Upravljanje: Šuplja kofa		625
7.8 Integrisane usluge		
7.8.1 Intserv		626
7.8.2 Diferencirane usluge		631
7.9.1 Ponašanja po skoku		634
7.9 RSVP		
7.9.1 Suština RSVP-a		637
7.5.1 Nekoliko jednostavnih primera		639
7.10 Rezime	643	
Domaći zadatak: problemi i pitanja		644
Problemi		645
Teze za diskusiju		649
Programerski zadatak		649
Intervju: Hening Šulcrine		651
Poglavlje 8 Bezbednost u računarskim mrežama	653	
8.1 Šta je bezbednost mreže	654	
8.2 Principi kriptografije	657	
8.2.1 Kriptografija simetričnog ključa		658
8.2.2 Šifrovanje javnim ključem		664
8.3 Autentifikacija		
8.3.1 Autentifikacioni protokol ap1.0		670
8.3.2 Autentifikacioni protokol ap2.0		671
8.3.3 Autentifikacioni protokol ap3.0		672
8.3.4 Autentifikacioni protokol ap3.1		672
8.3.5 Autentifikacioni protokol ap4.0		673
8.3.6 Autentifikacioni protokol ap5.0		674
8.4 Integritet 678		
8.4.1 Stvaranje digitalnih potpisa		678
8.4.2 Izvodi poruka		679
8.4.3 Algoritmi heš funkcija		681

8.5	Distribucija i overavanje ključeva	686			
8.5.1	Centar za distribuciju ključeva	686			
8.5.2	Overa javnog ključa	687			
8.6	Kontrola pristupa: mrežne barijere				
8.6.1	Filtriranje paketa	692			
8.6.2	Aplikacioni mrežni prolaz	695			
8.7	Napadi i kontramere				
8.7.1	Preslikavanje	697			
8.7.2	Uhodenje paketa	698			
8.7.3	Varanje	699			
8.7.4	Napadi odbijanja usluge i distribuiranog odbijanja usluge	700			
8.7.5	Pljačkanje	701			
8.8	Bezbednost u mnogo slojeva: studije primera				
8.8.1	Bezbedna elektronska pošta	703			
8.8.2	Sloj bezbednih soketa (SSL) i bezbednost transportnog				
	sloja (TLS)				
8.8.3	Bezbednost mrežnog sloja: IPsec	712			
8.8.4	Bezbednost u IEEE 802.11	716			
8.9	Rezime	721			
	Domaći zadatak: problemi i pitanja	722			
	Problemi	723			
	Teze za diskusiju	725			
	Intervju: Stiven M. Belovin	651			
avljie 9	Upravljanje mrežom	729			
9.1	Sta je upravljanje mrežom?	730			
9.2	Infrastruktura za upravljanje mrežom	734			
9.3	Standardni upravljački radni okvir za Internet	738			
9.3.1	Komponenta SMI	740			
9.3.2	Upravljačka informaciona baza: MIB	743			
9.3.3	Funkcionisanje protokola SNMP i transportna preslikavanja	745			
9.3.4	Bezbednost i administracija	749			
9.4	ASN.I	753			
9.5	Zaključak757	Domaći zadatak:	problemi	i	pitanja
	758				Problemi
	759	Teze		za	diskusiju
					760
					Intervju: Jeff Case
Reference		763			
Indeks		797			

Umrežavanje računara

Od vrha ka dnu sa Internetom u fokusu

Prevod trećeg izdanja

Intervju: Jeff Case

1

Računarske mreže i Internet

Sveopšta prisutnost računarskih mreža neprekidno se povećava pronalaženjem novih atraktivnih aplikacija - priineri su mnogobrojni - čitači Weba u mobilnim telefonima, kafei sa javnim bežičnim pristupom Internetu, kućne mreže širokopojasnog pristupa, zatim tradicionalna IT radna infrastruktura sa umreženim računarom na svakom stolu svake kancelarije. Umrežavaju se automobili, ali i senzori za praćenje stanja životne sredine. Konačno, Internet sada već ima interplanetarni značaj. Izgleda da su računarske mreže svuda oko nas! Ova knjiga pružiće vam savremeni uvod u veoma dinamično polje umrežavanja računara i ponuditi principe i praktična znanja koji će vam pomoći da razumete ne samo aktuelne mreže, već i one koje će postojati u budućnosti.

Ovo prvo poglavlje zamišljeno je kao svojevrstan prikaz umrežavanja računara i Interneta. U njemu smo želeli da iscrtamo konture umrežavanja i da ovu pojavu posmatramo uopšteno. Videćete da smo u ovom uvodnom poglavljtu obradili mnogo tema, neke čak i veoma detaljno, a da pri tome nismo gubili iz vida opštu sliku. Zahvaljujući tome, ovo poglavlje predstavlja osnovu na koju se nadovezuju sva naredna poglavља.

Evo kako će izgledati naš prikaz računarskih mreža u ovom prvom poglavljju. Nakon što vam predstavimo neke osnovne termine i koncepte, ispitaćemo osnovne hardverske i softverske komponente koje Čine računarsku mrežu. Počećemo od „periferije“ računarske mreže i prikazati vam krajnje sisteme i mrežne aplikacije

koje se izvršavaju u okviru mreže. Govorićemo takođe o transportnim uslugama koje ove aplikacije koriste. Nakon toga preći ćemo na „jezgro“ računarskih mreža i ispitati linkove i komutatore kojima se podaci prenose, zatim pristup mrežama i same fizičke medijume koji krajnje sisteme povezuju sa jezgrom mreže. Videćete da Internet predstavlja mrežu koja objedinjuje mnogo drugih mreža i naučiti na koji način su sve te mreže povezane.

Nakon prikaza „periferije“ i „jezgra“ računarskih mreža, zauzećemo poziciju koja će nam omogućiti sagledavanje nešto uopštenije slike. Ispitaćemo uzroke kašnjenja i gubitka podataka u računarskim mrežama i ponuditi vam jednostavne kvantitativne modele za određivanje kašnjenja od jednog do drugog kraja mreže. Ovi modeli prilikom izračunavanja u obzir uzimaju kašnjenje u prenosu, putovanju i formiranju redova. Potom ćemo vam predstaviti i neke ključne principe u arhitekturi umrežavanja računara kao što su, na primer, slojevitost protokola ili modeli usluga. Konačno, ovo poglavlje ćemo završiti kratkim istorijskim prikazom računarskih mreža.

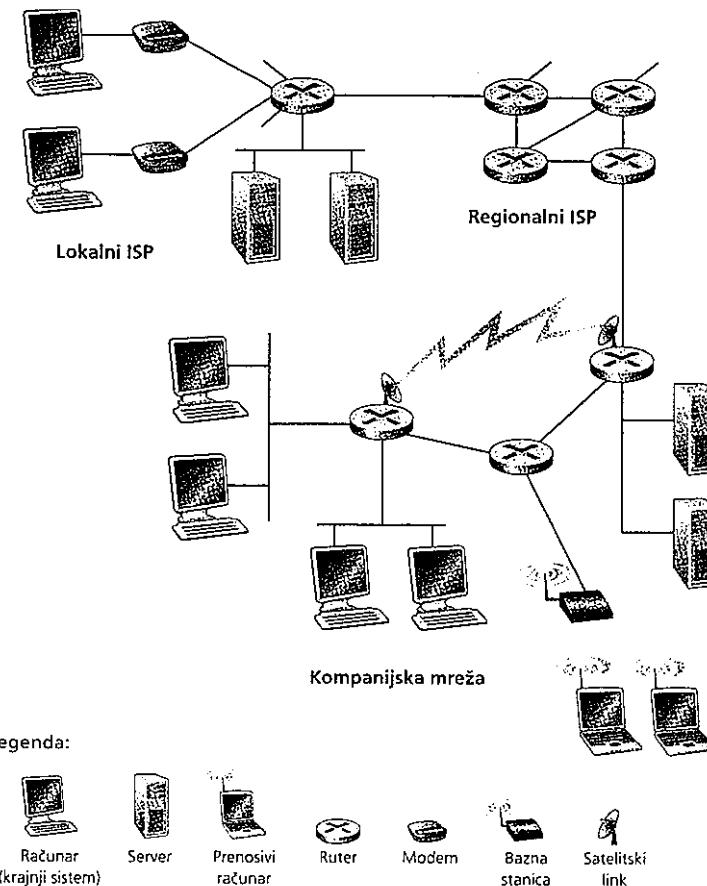
1.1 Staje Internet?

U ovom knjizi kao osnovno sredstvo za razmatranje načina funkcionisanja mrežnih protokola koristimo javni Internet - konkretnu računarsku mrežu. Ali šta je Internet? Zaista bismo želeli da vam pružimo odgovor u jednoj rečenici - definiciju koju biste mogli da koristite u razgovoru sa članovima porodice i prijateljima. Nažalost, Internet je, u pogledu hardverskih i softverskih komponenti, kao i usluga koje pruža, veoma složen, tako da jednostavno nije moguće dati toliko kratku definiciju.

1.1.1 Osnovne komponente Interneta

Umesto da pokušamo da definišemo Internet u jednoj rečenici, opredelili smo se za nešto deskriptivniji pristup, u kome postoje dva različita puta. Prvi bi podrazumevao opisivanje onoga što se nalazi ispod „haube“ Interneta, a to su osnovne hardverske i softverske komponente od kojih se on sastoji. Iz druge perspektive, Internet može da se opiše u smislu mrežne infrastrukture koja obezbeđuje usluge distribuiranim aplikacijama. Počećemo od prvog opisa i kao ilustraciju upotrebiti sliku 1.1.

Javni Internet predstavlja svetsku računarsku mrežu koja se sastoji od miliona računara raspoređenih širom sveta. Donedavno su većinu ovih uređaja činili tradicionalni stoni PC računari, UNIX radne stanice i tzv. serveri koji čuvaju i prenose informacije kao što su veb stranice i elektronska pošta. Međutim, osim njih, danas se sa Internetom povezuje i sve veći broj netradicionalnih krajnjih sistema (uređaja), kao što su PDA (*Personal Digital Assistant*) računari, televizori, prenosivi računari, mobilni telefoni, automobili, senzori za praćenje stanja životne sredine, okviri za slike, kućni elektronski i bezbednosni sistemi, veb kamere, pa čak i tosteri [BBC 2001]. Staviše, imajući u vidu broj netradicionalnih uređaja koji se danas povezuju sa Internetom, termin *računarska mreža* već počinje da zvuči pomalo arhaično. U internetskom žargonu, svi ovi uređaji nazivaju se računari ili krajnji sistemi.



Slika 1.1 ♦ Neki elementi Interneta

U januaru 2003. godine Internet je koristilo oko 233 miliona krajnjih sistema i taj broj se veoma brzo povećava [ISC 2004].

Krajnji sistemi su između sebe povezani **komunikacionim linkovima**. U odeljku 1.4 videćete da postoji veliki broj tipova komunikacionih linkova koji se prave od različitih fizičkih medija, kao što su koaksijalni kablovi, bakarni provodnici, optički kablovi i spektar radio talasa.

Različiti linkovi imaju i različitu **brzinu prenosa** podataka koja se izražava brojem bitova u sekundi.

Krajnji sistemi obično nisu direktno međusobno povezani jednim komunikacionim linkom. Ta veza najčešće je indirektna i vodi preko posredničkih uređaja za komutiranje koji se nazivaju **komutatori paketa**. Komutator paketa preuzima određenu količinu informacija koje stižu nekim od ulaznih komunikacionih linkova i prosledjuje ih nekim od svojih izlaznih komunikacionih linkova. U žargonu računarskog umrežavanja za ovu „određenu količinu informacija“ koristi se termin **paket**. Danas ima mnogo različitih oblika i modela komutatora paketa, a najrasprostranjeniji su **ruteri i komutatori sloja veze**. Zajednička karakteristika obe ove vrste komutatora jeste to da pakete prosleđuju do njihovih konačnih odredišta. O ruterima ćemo detaljnije govoriti u poglavljiju 4, a o komutatorima sloja veze u poglavljiju 5.

Put kojim paket putuje od krajnjeg sistema koji ga je poslao, kroz niz komunikacionih linkova i ruta, do krajnjeg sistema kome je namenjen, naziva se **ruta ili putanja**. Umesto da za komunikaciju krajnjih sistema obezbedi *namensku* putanju, Internet koristi tehniku **komutiranja paketa**, koja omogućava da veći broj krajnjih sistema u isto vreme i zajednički koristi celu putanju ili neki njen deo. Prve mreže sa komutiranjem paketa koje su se pojatile 1970-ih godina predstavljaju najranije pretke današnjeg Interneta. Tačan iznos obima saobraćaja današnjeg Interneta predmet je sporenja [Odvolsko 2003], ali po nekim konzervativnim procenama mesečno kroz američke međugradske mreže prode oko 100 000 terabajta informacija. Ovome treba dodati i podatak da se obim saobraćaja svake godine udvostručava.

Krajnji sistemi pristupaju Internetu preko **posrednika za Internet usluge** (*Internet Service Provider, ISP*) i to mogu biti kućni posrednici, kao što su AOL ili telefonske kompanije i kablovski operateri, zatim korporativni posrednici, univerzitetski posrednici, kao i posrednici kakav je, na primer, kompanija T-Mobile koja obezbeđuje bežični pristup na aerodromima, u hotelima, kafeima i na drugim javnim mestima. Svaki posrednik za Internet usluge, u stvari, predstavlja mrežu ruta i komunikacionih linkova. Različiti posrednici obezbeđuju krajnjim korisnicima i različite načine za pristup mreži, kao što su, primera radi, modemski pristup brzine **56 Kb/s**, rezidencijalni širokopojasni pristup kablovskom ili DSL vezom, zatim LAN pristup velike brzine ili bežični pristup. Ovi posrednici takođe obezbeđuju pristup Internetu onim kompanijama koje obezbeđuju sadržaje, povezujući veb lokacije direktno sa Internetsom. Da bi omogućili komunikaciju korisnika Interneta i pristup sadržajima širom sveta, ovi posrednici nižeg reda povezani su sa nacionalnim i internacionalnim posrednicima višeg reda, kao što su AT&T ili Sprint. ISP-ovi višeg reda, u stvari, predstavljaju određeni broj ruta velike brzine koji su između sebe povezani takođe veoma brzim optičkim kablovima. Svakom ISP mrežom, bez obzira na to da li je višeg ili nižeg reda, upravlja se nezavisno i u svakoj se koriste protokol IP (videti u produžetku), kao i odredene konvencije imenovanja i adresiranja. U odeljku 1.5 detaljnije ćemo ispitati posrednike za Internet usluge i njihove međusobne veze.

Krajnji sistemi, ruteri, kao i ostali delovi Interneta, koriste **protokole** koji kontrolišu slanje i prijem informacija u okviru Interneta. **TCP (Transmission Control Protocol)** i **IP (Internet Protocol)** predstavljaju dva najvažnija Internet protokola.

Protokolom IP precizira se format paketa koje razmenjuju ruteri i krajnji sistemi. Glavni Internet protokoli kolektivno se nazivaju **TCP/IP**. Već u ovom uvodnom poglavljju pozabavilićemo se mrežnim protokolima, ali to je samo početak. U nastavku ove knjige ćemo se veoma često vraćati ovoj temi.

Imajući u vidu značaj protokola na Internetu, veoma je važno da postoji opšti konsenzus o tome za Staje svaki od tih protokola zadužen. Upravo zato i postoje standardi. **Internet standarde** je razvila Tehnička radna grupa za Internet (*Internet Engineering Task Force, IETF*) [[IETF 2004](#)]. Dokumenti sa standardima grupe IETF nazivaju se **RFC** (*request for comments*, zahtevi za komentarima) dokumenti. RFC dokumenti pojavili su se kao uopšteni zahtevi za komentarima (otuda i njihov naziv) koji bi omogućili rešavanje problema u vezi sa arhitekturom mreže koja je prethodila Internetu. Ovi dokumenti su puni tehničkih pojedinosti, veoma su detaljni, i definišu protokole kao što su TCP, IP, HTTP (za Web) i SMTP (za elektronsku poštu). Tehnička radna grupa za Internet takođe je standardizovala protokole koji se izvršavaju na matičnim računarima [[RFC 1122](#); [RFC 1123](#)] i ruterima [[RFC 1812](#)]. Trenutno ima više od **3500** različitih RFC dokumenata. Postoje još neka tela koja se bave standardizacijom mrežnih komponenti, posebno kada su u pitanju mrežni linkovi. Primera radi, grupa **IEEE 802 LAN/MAN Standards Committee** [[IEEE 802 2004](#)] bavi se standardizacijom Ethernet i bežičnih Wi-Fi standarda.

Javni Internet (dakle, globalna mreža koja povezuje mnogo drugih mreža, kako smo ga nedavno definisali) jeste mreža za koju ljudi obično vezuju termin Internet. Međutim, ima još mnogo drugih privatnih mreža kao što su razne korporativne ili vladine mreže, čiji računan ne mogu da razmenjuju poruke sa računarima izvan njih (osim ukoliko te poruke ne prođu kroz tzv. mrežne barijere koje kontrolišu tok poruka ka mreži i iz nje). Budući da se u ovim privatnim mrežama koriste iste vrste računara, ruta, linkova i protokola kao i u javnom Internetu, za njih se često vezuje termin **intranet**.

1.1.2 Opis iz aspekta usluga

U prethodnom odeljku identifikovali smo mnoge delove od kojih se sastoji Internet. Za trenutak ćemo napustiti ovaj pristup i pokušati da ga sagledamo iz perspektive usluga.

- ◆ Internet omogućava razmenu podataka između **distribuiranih aplikacija** koje se izvršavaju na krajnjim sistemima. Među ovim aplikacijama su daljinska prijava, elektronska pošta, krstarenje Webom, trenutna razmena poruka, protok audio i video signala u realnom vremenu, internetska telefonija, distribuirane igre, deoba datoteka između ravnopravnih korisnika (**P2P**) i mnoge druge. Želimo da naglasimo da „Web“ nije posebna mreža, već pre jedna od mnogih distribuiranih aplikacija koje koriste komunikacione usluge koje Internet obezbeđuje.
- ◆ Internet obezbeđuje dve usluge svojim distribuiranim aplikacijama: **pouzdanu uslugu sa konekcijom i nepouzdanu uslugu bez konekcije**. Uopšteno govoreći, pouzdana usluga sa konekcijom garantuje da će podaci koji se prenose od pošiljaoca do primaoca u krajnjoj liniji biti isporučeni primaocu i to u potpunosti. Kod nepouzdanih usluga bez konekcije nema nikakvih garancija u vezi sa kona-

čnom isporukom. Distribuirane aplikacije obično koriste jednu (nikada obe) od ove dve usluge.

- ◆ U ovom trenutku Internet ne obezbeđuje uslugu kojom bi mogao da garantuje *trajanje* putovanja paketa od pošiljaoca do primaoca. Izuzev povećanja propusne moći kod vašeg posrednika za Internet usluge, trenutno ne postoji drugi način za obezbeđivanje boljih usluga (recimo, u smislu ograničenja kašnjenja), čak ni dodatnim plaćanjem, što je mnogim ljudima (naročito Amerikancima) prilično čudno. Ipak, u poglavljiju 7 ćemo vam prikazati vrhunsko istraživanje o Internetu koje bi moglo da promeni ovu situaciju.

Ovaj drugi način definisanja Interneta - u smislu usluga koje obezbeđuje distribuiranim aplikacijama - nije uobičajen, ali je veoma važan. Napredak sastavnih desova Interneta u sve većoj meri proistiće iz potreba novih aplikacija. Stoga je veoma važno imati u vidu daje Internet svojevrsna *infrastruktura* u kojoj se nove aplikacije neprekidno pronalaze i puštaju u rad.

Upravo smo vam ponudili dve moguće definicije Interneta - jednu iz aspekta hardverskih i softverskih komponenti i drugu, iz aspekta usluga koje Internet pruža distribuiranim aplikacijama. Ipak, sasvim je moguće da se još uvek pitate šta Internet, u stvari, predstavlja. Šta su komutiranje paketa, TCP/IP i usluge sa konekcijom? Šta su ruteri? Koje vrste komunikacionih linkova postoje u ovoj mreži? Šta je distribuirana aplikacija? Ukoliko još nemate odgovore na ova pitanja, ne brinite. Cilj ove knjige jeste da vam predstavi i sastavne delove Interneta i principе koji njime upravljaju. U poglavljima koja slede, objašnijemo vam sve ove važne termine i dati odgovore na pitanja koja vas možda još uvek muče.

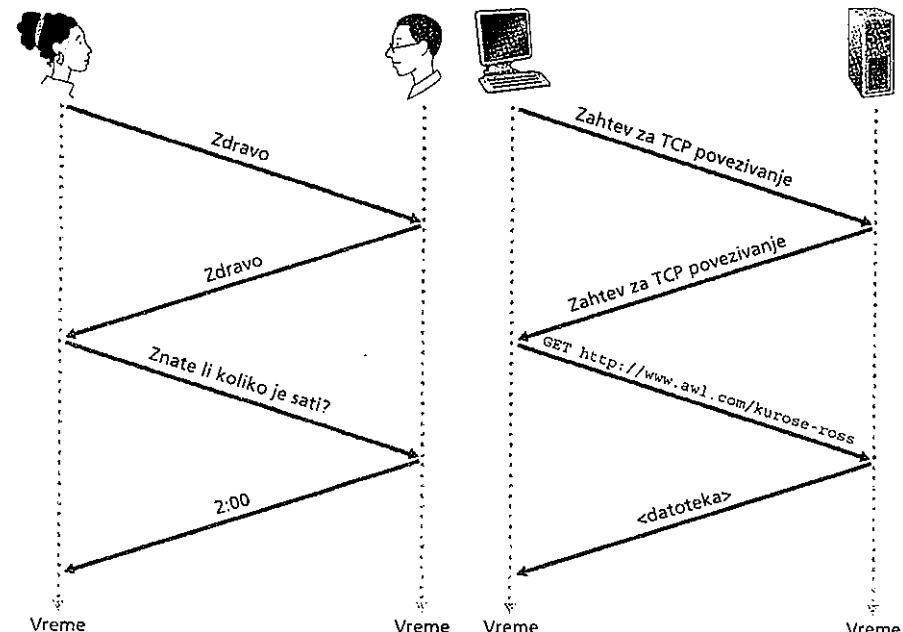
1.1.3 Staje protokol?

Sada, kada ste stekli makar osnovnu predstavu o tome šta je Internet, razmotrićemo još jednu veoma važnu reč u oblasti umrežavanja računara - *protokol*. Staje protokol? Šta on radi? Kako biste ga prepoznali kada biste se suočili sa njim?

Analogija sa ljudima

Verovatno najjednostavniji način za objašnjenje pojma mrežnog protokola jeste razmatranje nekih analogija sa ljudima, budući da kod nas uvek postoje nekakvi protokoli. Zamislite situaciju u kojoj nekoga treba da pitate koliko je sati. Uobičajena razmena reči prikazana je na slici 1.2.

Ljudski protokol (ili makar, dobiti maniri) nalaže da se komunikacija sa drugom osobom započne pozdravom (prvo „Zdravo“). Tipičan odgovor bi takođe glasio „Zdravo“, s tim što bi ova poruka imala srdačan prizvuk, stoje znak da prva osoba može da nastavi i postavi pitanje. Drugačiji odgovor na inicijalno „Zdravo“ (recimo, „Ostavi me na miru!“ ili „Ne govorim srpski!“, ili možda nešto nepristojno) ukazivao bi na to da druga osoba ne želi ili ne može da odgovori na komunikaciju. U tom



Slika 1.2 ◆ Poređenje ljudskog i računarskog protokola

slučaju, u skladu sa ljudskim protokolom, prva osoba ne bi upitala koliko je sati. Dešava se da na svoj pozdrav prva osoba ne dobije nikakav odgovor i u tom slučaju ona takođe ne nastavlja sa svojim drugim pitanjem. Skrećemo vam pažnju na to da u ljudskom protokolu postoje *konkretnе poruke koje šaljemo i konkretnе akcije koje su odgovor na primljene poruke* *Hi druge dogadaje (kao što je ignoštanje)*. Jasno je da poslate i primljene poruke, kao i akcije koje se preduzimaju nakon slanja i prijema tih poruka imaju ključnu ulogu u ljudskom protokolu. Ukoliko ljudi imaju različite protokole (na primer, jedna osoba ima dobre manire, a druga ne, ili jedna razume pojmom vremena, a druga ne), njihova interakcija nije moguća, što znači da se ne može postići ništa korisno. Isti principi važe i u umrežavanju - za postizanje bilo čega i izvršavanje bilo kakvog zadatka potrebno je da dva (ili više) uređaja u međusobnoj komunikaciji koriste isti protokol.

Razmotrimo sada drugu analogiju sa ljudima. Pretpostavimo, primera radi, da se nalazite u jednoj studijskoj grupi (to može da bude i kurs o umrežavanju!) i da predavač na veoma nezanimljiv način govori o umrežavanju. Nakon završetka svog predavanja, on postavlja uobičajeno pitanje: „Ima li nekih pitanja?“ (poslatu poruku

primaju svi studenti koji nisu zaspali). Vi dižete ruku (šaljući implicitnu poruku predavaču), on se nasmeši, izgovarajući: „ Da...“ (poruka koja vas ohrabruje da postavite pitanje), vi postavljate pitanje (šaljete svoju novu poruku) i, konačno, on sluša vaše pitanje (prima poruku) i odgovara na njega (šalje vam svoj odgovor). I iz ovog primera možete da zaključite da ključni segment protokola pitanja i odgovora predstavljaju prenos i prijem poruka, uključujući i sve one akcije koje se preduzimaju nakon prijema ili slanja poruka.

Mrežni protokoli

Mrežni protokol je veoma sličan primeru sa ljudima, osim što su entiteti koji razmenjuju poruke i preduzimaju akcije u ovom slučaju hardverske i softverske komponente nekog uređaja (recimo, računara, rutera ili nekog drugog uređaja koji može da se umreži). Svakom aktivnošću na Internetu koja podrazumeva komunikaciju dva ili više udaljenih entiteta upravlja protokol. Na primer, protokoli u ruterima određuju putanje paketa od njegovog izvora pa do njegovog odredišta; hardverski implementirani protokoli u mrežnim interfejsima dva fizički povezana računara kontrolišu tok bitova kroz „žicu“ između ova dva interfejsa; protokoli za kontrolu zagruženja saobraćaja u krajnjim sistemima kontrolišu brzinu prenosa paketa između pošiljaoca i primaoca. Internet je prepun protokola i upravo zato je dobar deo ove knjige i posvećen protokolima računarskih mreža.

Kao primer protokola koji vam je verovatno poznat, zamislite Šta se dešava kada pošaljete zahtev veb serveru, odnosno, kada u svoj čitač upišete URL neke veb strane. Ova situacija opisana je u desnoj polovini slike 1.2. Dakle, vaš računar naj-pre šalje „zahtev za povezivanje“ veb serveru i zatim čeka njegov odgovor. Server u krajnjoj liniji prima ovaj zahtev i odgovara porukom daje povezivanje moguće. Znajući da može da zatraži dokument, vaš računar zatim u poruci tipa „GET“ šalje ime veb stranice koju želi da preuzme sa veb servera. Na kraju, veb server šalje traženu stranicu vašem računaru.

Imajući u vidu i primer sa ljudima i primer iz prave mreže, ključni elementi koje definiše svaki protokol jesu razmena poruka i akcije koje se preduzimaju nakon slanja ili prijema ovih poruka:

Protokol definije format i redosled poruka koje se razmenjuju između dva ili više komunicirajući/! entiteta, kao i akcije koje se preduzimaju nakon slanja ili prijema poruke ili nekog drugog dogadaja.

U okviru Interneta, ali i svih drugih računarskih mreža, protokoli se koriste u značajnoj meri i to za postizanje različitih zadataka u sferi komunikacije. Čitajući ovu knjigu saznaćete da postoje sasvim jednostavnii, ali i veoma složeni protokoli. Ovladavanje oblaštu umrežavanja računara praktično bi moglo da se poistoveti sa razumevanjem svih aspekata mrežnih protokola.

1.2 Periferija mreže

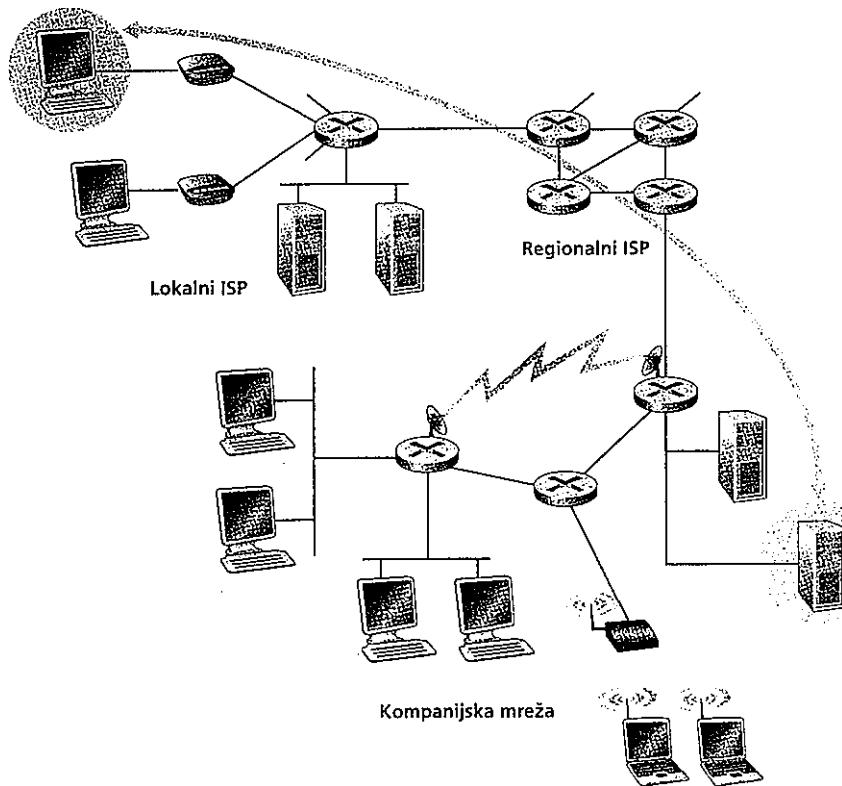
U prethodnim odeljcima predstavili smo vam Internet i mrežne protokole na uopšteniji način, a sada sledi detaljniji prikaz komponenti računarske mreže (i posebno Interneta). Počećemo od periferije mreže i od onih komponenti koje su nam svima najpoznatije, a to su računari koje svakodnevno koristimo. Nakon toga ćemo od periferije mreže da se uputimo ka njenom centru i objasnimo pojmove komutiranja i rutiranja u računarskim mrežama. Konačno, u odeljku 1.4 pozabavljemo se i samim fizičkim linkovima koji prenose signale između računara i komutatora.

1.2.1 Krajnji sistemi - klijenti i serveri

U terminologiji umrežavanja, za računare koji su povezani sa Internetom često se koristi i termin **krajnji sistemi**. Oni se smatraju krajnjim sistemima zato što se nalaze na samom obodu Interneta (slika 1.3). Kategorija krajnjih sistema Interneta obuhvata nekoliko različitih tipova računara. Krajnji korisnici nalaze se u neposrednom kontaktu sa ovim računarama u koje spadaju stoni (PC, Mac ili UNIX radne stanice) i prenosivi računari (laptop i PDA računari i telefoni sa bežičnim pristupom Internetu). Pored toga, sve je veći broj alternativnih uređaja poput jednostavnih klijenata i uređaja za domaćinstvo [Thinplanet 2002], veb televizora, uređaja koji se povezuju sa televizorima [Nesbitt 2002], digitalnih kamera, uređaja za domaćinstvo, fabričkih postrojenja ili senzora za praćenje stanja životne sredine, koji se kao krajnji sistemi povezuju sa Internetom (obavezno pročitajte i istorijski osrvt koji sledi).

Za krajnje sisteme koristi se termin matični računari, ili domaćini (engl. *hosts*), zato što se na njima izvršavaju aplikacije kao što su programi za Čitanje vebova, Čitači elektronske pošte, programi za prosledivanje elektronske pošte i veb serveri. U nastavku ove knjige ćemo ravnopravno koristiti tennine (matični) računar i krajnji sistem, što znači da im je značenje identično. Matični računari se često dalje dele na kategorije **klijenata i servera**. Uopšteno govoreći, klijenti su stoni, laptop i PDA računari, dok su serveri nešto moćnije mašine koje se koriste za skladištenje i distribuiranje veb strana, protok video materijala u realnom vremenu, prosledivanje elektronske pošte i slično.

U kontekstu mrežnog softvera postoji još jedna definicija klijenta i servera na koju ćemo se pozivati u ovoj knjizi. **Klijentski program** se izvršava najednom krajnjem sistemu koji zahteva i dobija usluge od **serverskog programa** koji se izvršava na drugom krajnjem sistemu. Ovakav klijentsko/serverski model (koji ćemo detaljno ispitati u poglavljju 2) nesumnjivo predstavlja najrasprostranjeniju strukturu kada su u pitanju aplikacije za Internet. Web, elektronska pošta, transfer datoteka, daljinsko prijavljivanje (na primer, Telnet), elektronske konferencije..., samo su neki od primera aplikacija u kojima je primenjen model klijent/server. S obzirom na to da se klijentski program po pravilu izvršava najednom računaru, a serverski na drugom, klijentsko/serverske Internet aplikacije su, po definiciji, **distribuirane**. Klijentski i serverski program ostvaruju interakciju uzajammim slanjem poruka putem Interneta. Na ovom nivou apstrakcije, ruteri, linkovi i ostali „delići“ Interneta imaju ulogu „crne kutije“ za prenos poruka između distribuiranih komponenti internetskih



Slika 1.3 ♦ Interakcija krajnjih sistema

aplikacija, koje su u međusobnoj komunikaciji. Upravo ovaj nivo apstrakcije možete uočiti na slici 1.3. Međutim, nemaju sve savremene Internet aplikacije. Čiste klijentske programe koji ostvaruju interakciju sa čistim serverskim programom. Primera radi, kod aplikacija za razmenu datoteka u mrežama ravnopravnih računara (kao Stoje KaZaA) aplikacija na oba krajnja sistema ponaša se i kao klijentski i kao serverski program. Kada zahtevaju podatke sa drugog računara ovi programi se ponašaju kao klijenti, a kada ih šalju drugom računom onda imaju ulogu servera. U Internet telefoniji dve komunicirajuće strane su takođe ravnopravne; drugim recima, ovde se ne događa da jedna strana zahteva neku uslugu od druge. U poglavlju 2 ove knjige nalazi se detaljno poređenje klijentsko-serverske i P2P arhitekture.



KRATAK OSVRT

NEVEROVATAN SPEKTAR KRAJNJIH SISTEMA

Ne tako davno, jedini krajnji sistemi koji su se povezivali sa Internetom bili su tradicionalni sloni računari i moćni serveri. Međutim, krajem 1990-ih počeo je trend povezivanja najrazličitijih uređaja sa ovom globalnom mrežom. Za sve ove uređaje zajednička je potreba razmene digitalnih informacija sa nekim drugim uređajima. Kada se uzmu u obzir sveopšto prisutnost Interneta, zatim njegovi dobro definisani (i standardizovani) protokoli, kao i pristupačnost uređaja koji se sa njim povezuju, sasvim je prirodno to što se ova mreža koristi za povezivanje svih pomenutih uređaja. Za neke od ovih uređaja mogli bismo da kažemo da su napravljeni isključivo radi zabave. Tako, na primer, stoni ram za sliku koji podržava protokol IP [Ceivo 2004] preuzima digitalne fotografije sa udaljenog servera i prikazuje ih kao da su uramljene na klasičan način; Internet tester preuzima meteorološke podatke sa servera i odgovarajuću sliku dnevne vremenske prognoze (recimo, sliku promenljivog oblaknog i sunčanog vremena) vam prikazuje na ispečenom parčetu hleba [BBC 2001]. Ipak, postoje i uređaji koji mogu do nam obezbediti korisne informacije - veb kamere na monitoru ili nečem drugom prikazuju aktuelne saobraćajne i vremenske uslove; razvijene su čok i moštine za pranje vesa koje su povezane sa Internetom. Rad ovakvih mašina može da se prati preko Čitača Weba, a one same generišu elektronsku poštu kojom obaveštavaju korisnike o završetku svog rada. Mobilni telefoni sa ugradenim protokolom IP doneli su novu dimenziju pretraživanja VWeba, elektronske pošte i trenutne razmene poruka. Nova klasa umreženih senzorskih sistema donosi revoluciju u načinu praćenja životne sredine. Ovi umreženi senzori koji su ugrađeni u fizičke objekte omogućavaju praćenje zgrada, mostova i ostalih objekata koje su ljudi napravili [Elgamal 2001], žalim praćenje seizmičke aktivnosti [CISN 2004], živog sveta na Zemlji [Moimvaring 2002], esluora reka [Baptista 2003], biomedicinskih funkcija [Schwiebert 2001], vetrova i meteoroloških opasnosti u donjim slojevima atmosfere [ČASA 2004]. Sve prikupljene informacije dostupne su odmah svim zainteresovanim udaljenim korisnicima. Cilj centra CENS [Center for Embedded Networked Sensing] pri Univerzitetu UCLA upravo je primena tehnologije umreženih, ugradenih senzora, u važnim naučnim i društvenim oblastima.

1.2.2 Usluge sa konekcijom i bez nje

Krajnji sistemi svoju međusobnu komunikaciju ostvaruju preko Interneta. Konkretno, programi na krajnjim sistemima koriste usluge Interneta za međusobno slanje poruka. Linkovi, ruteri i ostali sastavni delovi Interneta obezbeđuju sredstva za transport ovih poruka između programa na krajnjim sistemima. Međutim, koje su karakteristike komunikacionih usluga koje Internet pruža svojim krajnjim sistemima?

TCP/IP mreže, a to znači i Internet, obezbeđuju dve vrste usluga aplikacijama krajnjih sistema: **usluge sa konekcijom i usluge bez konekcije**. Programer koji

pravi aplikaciju za Internet (recimo, aplikaciju za elektronsku poštu, aplikaciju za transfer datoteka, aplikaciju za Web ili za Internet telefoniju) mora da je napravi tako da koristi jednu od ove dve usluge. U nastavku ćemo ukratko da vam opišemo svaku od njih. (Više informacija o ovim uslugama pronaći ćete u poglavljju 3 u kome ćemo se baviti protokolima transportnog sloja.)

Usluga sa konekcijom

Kada neka aplikacija koristi uslugu sa konekcijom, klijentski i serverski programi (koji se nalaze na različitim krajnjim sistemima), pre slanja samih podataka (recimo, elektronske pošte), najpre jedan drugom šalju kontrolne pakete. Ovaj postupak sinhronizacije upozorava klijenta i servera da treba da se pripreme za slanje paketa. Nakon završetka procedure sinhronizacije kaže se daje između dva krajnja sistema uspostavljena konekcija.

Zanimljivo je daje ova procedura inicijalne sinhronizacije veoma slična protokolu koji ljudi koriste u svojim odnosima. Razmena poruka „Zdravo“ sa slilje 1.2 predstavlja primer ljudskog „protokola sinhronizacije - rukovanja“ (mada između dva čoveka ne mora da dode i do samog čira rukovanja). U vebovskoj interakciji, koja je takođe prikazana na slici 1.2, prve dve razmenjene poruke takođe su poruke sinhronizacije. Dve naknadne poruke - poruka GET i poruka sa odgovorom u kojoj se nalazi i datoteka - smatraju se samim podacima i šalju se tek nakon uspostavljanja konekcije.

Možda se pitate zbog čega se kaže „usluga sa konekcijom“, a ne „usluga konekcije“. Ova terminološka razlika proističe iz činjenice da su dva krajnja sistema u stvari povezana veoma labavo. Konkretno, jedino sami krajnji sistemi znaju za postojanje ove konekcije - komutatori paketa (ruteri) na Internetu nemaju pojma o njenom postojanju. Štaviše, konekcija preko Interneta nije ništa više od dodeljene privremene memorije i promenljivih stanja na krajnjim sistemima. Posrednički komutatori paketa ne održavaju nikakve informacije u vezi sa stanjem konekcije.

Usluga sa konekcijom obično dolazi u paketu sa nekoliko drugih usluga, kao što su pouzdani transfer podataka, kontrola toka podataka i kontrola zagruženja. Pod **pouzdanim transferom podataka** podrazumeva se to da aplikacija, u smislu isporuke svih podataka bez grešaka i njihovog pravilnog redosleda, može da se osloni na konekciju. Pouzdanost se na Internetu postiže korišćenjem potvrda i ponovnih pre-nosa. **Da** biste stekli najosnovniju predstavu o načinu na koji Internet implementira pouzdanu transportnu uslugu, zamislite aplikaciju koja je uspostavila vezu između krajnjih sistema A i B.

Kada krajnji sistem B primi paket od sistema A, on mu šalje poruku kojom to i potvrđuje. Zahvaljujući toj poruci, krajnji sistem A zna daje odgovarajući paket definitivno primljen. Ukoliko sistem A ne primi poruku o potvrdi, on pretpostavlja da krajnji sistem B nije primio poslati paket, pa ga zato šalje ponovo. **Kontrolom toka** izbegava se da jedna strana preoptereti onu drugu prebrzim slanjem prevelikog broja paketa. U poglavljju 3 ćete videti da se Internet usluga za kontrolu toka imple-

mentira korišćenjem privremene memorije za slanje i prijem na krajnjim sistemima. Usluga za kontrolu zagruženja Interneta pomaže u sprečavanju da ova mreža dospe u stanje potpunog zastoja. Kada na nekom ruteru dove do zagruženja, moguća su prekoračenja njegove privremene memorije i gubitak paketa. Kada bi u ovakvim okolnostima svaki par krajnjih sistema nastavio da „upumpava“ podatke u mrežu najbrže što može, došlo bi do zastoja i samo mali broj paketa bi zaista dospeo do svog odredišta. Na Internetu se ovaj problem izbegava tako što se krajnji sistemi prinuduju na smanjivanje brzine emitovanja paketa u periodima zagruženja. Onog trenutka kada prestanu da primaju potvrde o prispeću paketa koje su poslali, krajnji sistemi postaju svesni postojanja ozbiljnijeg zagruženja.

Odve želimo da naglasimo da, iako usluge sa konekcijom dolaze u paketu sa pouzdanim transferom podataka, kontrolom toka i kontrolom zagruženja, ove tri komponente ni u kom slučaju ne bi mogle da se nazovu osnovnim komponentama ove vrste usluga. Postoje računarske mreže koje svojim aplikacijama obezbeđuju usluge sa konekcijom bez ijedne od ovih komponenti. S druge strane, svaki protokol koji pre prenosa podataka sinhronizuje entitete koji komuniciraju, pripada uslugama sa konekcijom.

Internetska usluga sa konekcijom ima svoje ime: **TCP (Transmission Control Protocol)**. Osnovna verzija protokola TCP definisana je dokumentom RFC 793 [RFC 793]. U usluge koje protokol TCP obezbeđuje aplikacijama spadaju apstrakcija toka podataka, pouzdani transport, kontrola toka i kontrola zagruženja. Veoma je važno naglasiti da aplikacija treba da brine samo o obezbedenim uslugama; za nju uopste nije bitno na koji način protokol TCP implementira pouzdanost, kontrolu toka ili kontrolu zagruženja. Naravno, za nas su ova pitanja važna, tako da ćemo vam u poglavljju 3 pružiti detaljne odgovore na njih.

Usluga bez konekcije

Kod usluge bez konekcije ne postoji procedura sinhronizacije (rukovanja). U ovom slučaju, kada jedna strana aplikacije želi da pošalje pakete drugoj strani, program za slanje paketa to jednostavno i učini. S obzirom na to da ovde prenosu podataka ne prethodi procedura sinhronizacije, isporuka podataka je nešto brža. S druge strane, ovde ne postoji ni pouzdan transfer podataka, tako da pošiljalac nikada ne može da bude siguran koji su paketi zaista stigli do svog odredišta. Štaviše, u internetskoj usluzi bez konekcije nema mesta ni za kontrolu toka ni za kontrolu zagruženja: Internetska usluga bez konekcije naziva se **UDP (User Datagram Protocol)**, a definisana je dokumentom RFC 768.

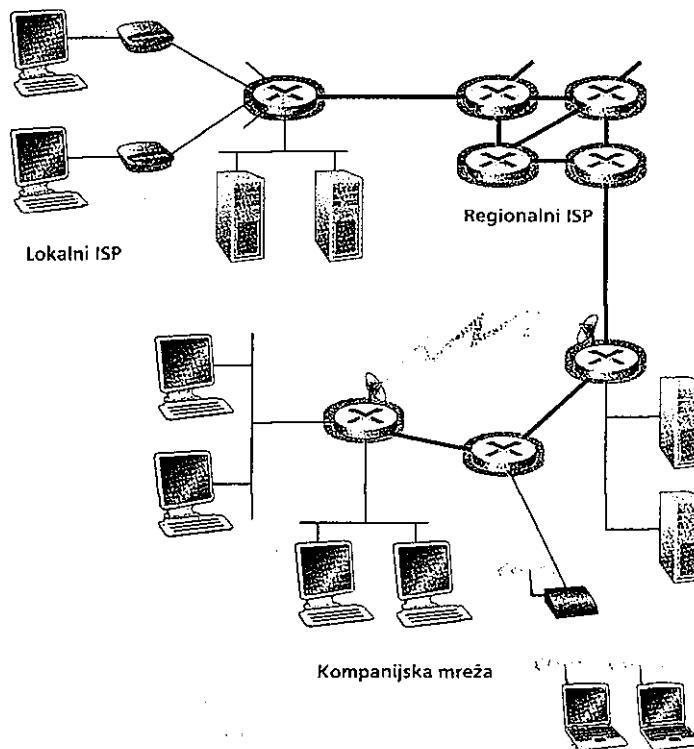
Najveći deo popularnih aplikacija za Internet koristi uslugu sa konekcijom (TCP). Među ovim aplikacijama su Tebe (za daljinsko prijavljivanje), SMTP (za elektronsku poštu), FTP (za transfer datoteka) i HTTP (za Web). Međutim, i uslugu bez konekcije (UDP) koristi sasvim solidan broj aplikacija, naročito neke novije multimedijalne aplikacije kao što su, na primer, internetska telefonija i video konferencije.

1.3 Jezgro mreže

Pošto smo ispitali krajnje sisteme Interneta i njihov transportni model usluga, u nastavku teksta preći ćemo na „unutrašnjost“ ove mreže. U ovom odeljku naša tema biće samo jezgro ove mreže, odnosno mreža rutera koji povezuju krajnje sisteme. Na slici i .4 jezgro mreže je prikazano korišćenjem debljih zelenih linija.

1.3.1 Komutiranje vodova i komutiranje paketa

U formiranju mrežnog jezgra postoje dva osnovna pristupa i to su komutiranje vodova i komutiranje paketa.



Slika 1.4 ♦ Jezgro mreže

U mrežama sa komutiranjem vodova, neophodni resursi koji obezbeđuju komunikaciju između dva krajnja sistema (privremena memorija, propusna moć linka) na celoj dužini putanje paketa ostaju *rezervisani* sve vreme trajanja komunikacione sesije. U mrežama sa komutiranjem paketa, pomenuti resursi *nisu* rezervisani već ih poruke tokom sesije koriste na zahtev, i zato poneka moraju da Čekaju (stanu u red) na pristup komunikacionom linku. Kao jednostavnu analogiju zamislite sada dva restorana - jedan u kome su rezervacije neophodne i drugi u kome se one ne prihvataju. U restoranu u kome su rezervacije neophodne imamo jedino taj zahtev: da se najavimo pre nego što krenemo. Ali kada dodemo u restoran, praktično istog trenutka možemo da komuniciramo sa konobarom i poručimo obrok. U restoranu koji ne prihvata rezervacije ne moramo prethodno da se najavljujemo i rezervišerno sto. Ali kada stignemo u restoran, najpre ćemo morali da sačekamo na red za sto i tek potom možemo da komuniciramo sa konobarom.

Opšteprisutne telefonske mreže predstavljaju primere mreža sa komutiranjem vodova. Zamislite situaciju u kojoj jedna osoba telefonskom mrežom želi da pošalje informaciju (glasovnu ili faksimil) drugoj osobi. Da bi pošiljalac mogao da pošalje svoje informacije, telefonska mreža najpre mora da ustpostavi vezu između pošiljaoca i primaoca. Nasuprot TCP vezi o kojoj smo pričali u prethodnom odeljku, ovo je *bona jide* veza u kojoj komutatori na putanji između pošiljaoca i primaoca održavaju njen status povezanosti. U žargonu telefonije ovakva veza naziva se vod (kolo). Kada ustpostavi vod, mreža za daru vezu rezerviše i konstantnu brzinu pre-nosa tokom celog njenog trajanja. S obzirom na to da ova veza između pošiljaoca i primaoca ima rezervisani propusnu moć, pošiljalac može da prenosi podatke *garan-tovanom* konstantnom brzinom.

Današnji Internet u osnovi predstavlja mrežu sa komutiranjem paketa. Zamislite sada situaciju u kojoj jedan računar želi da pošalje paket nekom drugom računaru preko Interneta. Kao i kod mreža sa komutiranjem vodova, dati paket treba da prođe kroz seriju komunikacionih linkova. Ali u mrežama sa komutiranjem paketa, dati paket šalje se kroz mrežu bez ikakve rezervacije propusne moći. Ukoliko je neki od linkova zagušen zato što njime u tom trenutku već prolaze drugi paketi, naš paket će morati da sačeka u privremenoj memoriji na prednjom kraju prenosnog linka, a to znači daje kašnjenje neizbežno. Internet „daje sve od sebe“ kako bi se obezbedio pravovremena isporuka paketa, ali ne postoje nikakve garancije u tom pogledu.

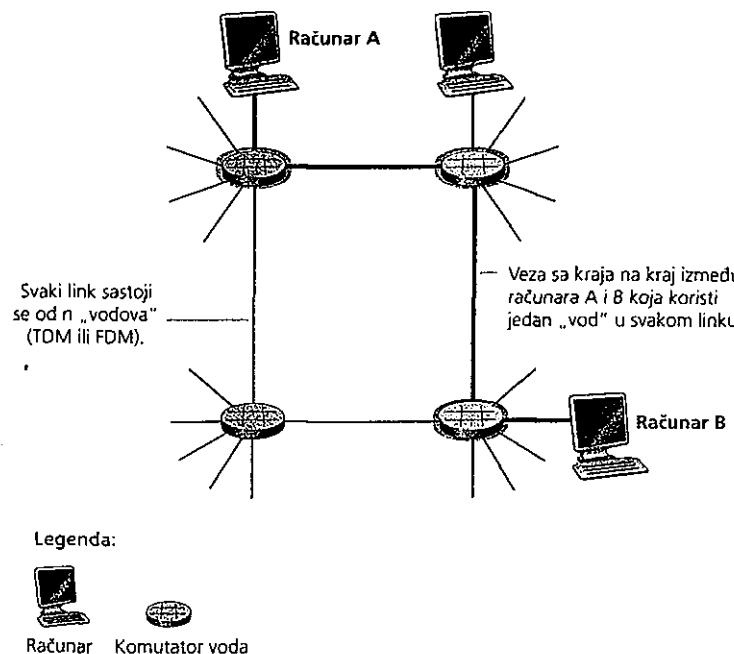
Konačno, ne mogu sve telekomunikacione mreže da se uklope u ovu podelu na mreže sa komutiranjem vodova i mreže sa komutiranjem paketa. Međutim, ovakva osnovna klasifikacija predstavlja odličnu polaznu tačku za razumevanje tehnologije telekomunikacionih mreža.

Komutiranje vodova

Ova knjiga je posvećena računarskim mrežama, Internetu i komutiranju paketa, a ne telefonskim mrežama i komutiraju vodova. Ipak, veoma je važno da razumete zašto Internet i ostale računarske mreže koriste komutiranje paketa, a ne postojeću

tehnologiju komutiranja vodova koja je prisutna u telefonskim mrežama. Upravo zato sledi kratak prikaz tehnologije komutiranja vodova.

Na slici 1.5 prikazana je mreža sa komutiranjem vodova. U ovoj mreži Četiri komutatora vodova međusobno su povezana putem četiri linka. Svaki od ovih linkova ima n vodova, što znači da svaki link može da podrži n simultanih veza. Svi računari (PC-ji i radne stanice) direktno su povezani sa jednim od komutatora. Kada dva računara žele da komuniciraju, mreža između njih uspostavlja namensku vezu sa **kraja na kraj**. (Konferencijski pozivi između više od dva uređaja takođe su mogući, ali ćemo, da bi sve bilo što jednostavnije, pretpostaviti da u svakoj vezi postoje samo po dva računara.) Prema tome, da bi računar A mogao da pošalje poruke računaru B, mreža najpre mora da rezerviše jedan vod na svakom od dva linka. S obzirom na to da svaki link ima n vodova, za svaki upotrebljeni link ova veza sa kraja na kraj dobija i/n ukupne propusne moći linka i to tokom čitavog trajanja veze.



Slika 1.5 ♦ Jednostavna mreža sa komutiranjem vodova koju čine četiri komutatora i četiri linka

Multipleksiranje u mrežama sa komutiranjem vodova

U okviru linka, vod se realizuje ili frekventnim multipleksiranjem (*Frequency-Division Multiplexing*, FDM) ili vremenskim multipleksiranjem (*Time-Division Multiplexing*, TDM). U FDM tehnologiji, sve veze koje su uspostavljene duž linka dele njegov frekventni spektar. Konkretno, link dodeljuje određeni frekventni opseg svakoj vezi tokom čitavog njenog trajanja. U telefonskim mrežama ovaj frekventni opseg obično ima širinu od 4 kHz (4 000 herca ili 4 000 ciklusa u sekundi). Širina ovog opsega se, nimalo iznenadujuće, naziva propusni opseg (propusna moć). FM radio stanice takođe koriste tehnologiju FDM kako bi zajednički mogle da koriste raspoloživi frekventni spektar između 88 i 108 MHz.¹

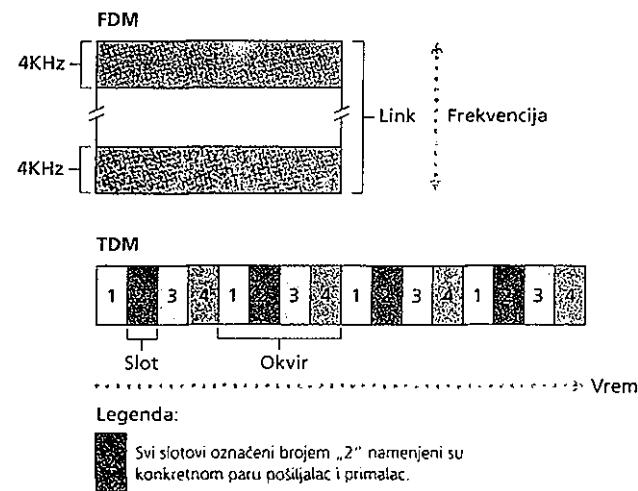
U TDM linkovima, vreme je podeljeno na okvire fiksne dužine, a svaki od njih podeljen je na fiksni broj vremenskih odsečaka. Kada mreža uspostavi vezu duž određenog linka, toj vezi se namenjuje po jedan vremenski odsečak u svakom okviru. Ovi vremenski odsečci su namenski - može da ih koristi samo data veza, a svaki od njih (u svakom okviru) može da prenosi podatke.

Na slici 1.6 ilustrovane su tehnologije FDM i TDM na konkretnom mrežnom linku koji podržava do četiri voda. U FDM pristupu, frekventni spektar podeljen je na četiri opsega od kojih svaki ima propusni opseg od po 4 kHz. U verziji TDM, vremenski spektar podeljen je na okvire sa po četiri slota u svakom od njih; svakom vodu dodeljuje se isti namenski vremenski odsečak u rotirajućim TDM okvirima. U ovom slučaju, brzina prenosa voda jednaka je brzini prenosa okvira podeljenog sa brojem bitova u odsečku. Na primer, ukoliko link prenosi 8 000 okvira u sekundi, a svaki vremenski odsečak čini osam bitova, brzina prenosa voda tada iznosi 64 kb/s.

Pristalice komutiranja paketa uvek su tvrdile da komutiranje vodova predstavlja rasipništvo zato što su namenski vodovi u stanju mirovanja tokom tihih perioda. Na primer, kada jedna osoba u telefonskom razgovoru prestane da priča, druga aktivna veza ne može da upotribe mrežne resurse koji tada prelaze u stanje mirovanja (frekventne opsege ili vremenske odsečke u linkovima duž putanje veze). Drugi primer neiskorišćenosti resursa bio bi, recimo, radiolog koji putem mreže sa komutiranjem vodova dolazi do serije rendgenskih snimaka. Dakle, on uspostavlja vezu, zahteva snimak koji zatim analizira, i na kraju zahteva novi snimak. Tokom perioda analiziranja snimaka mrežni resursi su potpuno neiskorišćeni. Zagovornici komutiranja paketa takođe sa neskrivenim zadovoljstvom ističu da su uspostavljanje vodova od kraja do kraja i rezervisanje propusnog opsega komplikovani i za njih je neophodan složen softver za signalizaciju koji treba da koordinira radom komutatora duž ovakve putanje.

Pre nego što završimo razmatranje komutiranja vodova, navešćemo i jedan numerički primer koji će vam dodamo ilustrovati razlike između navedenih tehnologija. Dakle, pogledajmo koliko je vremena potrebno za slanje datoteke od 6 400 000 bitova od računara A do računara B kroz mrežu sa komutiranjem vodova. Pretpostavimo da svi linkovi u mreži koriste tehnologiju TDM i da imaju brzinu od 1,536 Mb/s. Takođe ćemo pretpostaviti daje za uspostavljanje voda od kraja do kraja potrebno 500 ms, nakon čega računar A počinje emitovanje podataka. Koliko

Ovo koriste i AM stанице i lo mnogo očiglednije, npr. TV. Generalno, FDM u ovom kontekstu je AM! (prim. rec.)



Slika 1.6 ♦ U tehnologiji FDM svaki vod stalno ima na raspolaganju deo ukupne propusne moći. U tehnologiji TDM svaki vod periodično, tokom kratkih vremenskih intervala (odsečaka), dobija celokupnu propusnu moć.

je vremena potrebno da datoteka stigne do svog odredišta? Svaki vod ima brzinu prenosa od $(1,536 \text{ Mb/s})/24 = 64 \text{ kb/s}$, što znači da je za prenos datoteke potrebno $(640\,000 \text{ bitova})/(64 \text{ kbps}) = 10 \text{ sekundi}$. Kada ovome dodamo i vreme potrebno za uspostavljanje veze, dobijamo 1 0.5 sekundi. Skrećemo vam pažnju na to da vreme prenosa ne zavisi od broja linkova - prenos traje 10 sekundi bez obzira na to da li vod prolazi kroz jedan ili sto linkova. (Stvarno kašnjenje u vezama od kraja do kraja zavisi od kašnjenja usled propagacije; pročitajte odeljak 1.6.)

Komutiranje paketa

U odeljku 1.1 videli ste da aplikacije svoje zadatke obavljaju razmenom **poruka**. U porukama može da se nalazi sve što je dizajner protokola zamislio. One mogu imati kontrolnu funkciju (poruke „Zdravo“ iz primera o sinhronizaciji) ili mogu da sadrže podatke - na primer, elektronsku poštu, JPEG sliku ili MP3 muzičku datoteku. U savremenim računarskim mrežama pošiljalac dugačke poruke razbija na manje skupine podataka koje se nazivaju **paketi**. Svaki od ovih paketa, od svog izvora pa do svog odredišta, treba da prođe kroz komunikacione linkove i **komutatore paketa** (ili kroz **rutere**, ili kroz komutatore sloja veze). Paketi se kroz svaki komunikacioni link prenose brzinom koja je jednaka njegovoj *punoj* brzini prenosa. Najveći broj komutatora paketa koristi prenos tipa **memorisanje i prosleđivanje**. Pod ovim se podrazumejava da komutator počinje da prenosi prvi bit nekog paketa kroz svoj

izlazni link tek kada primi čitav taj paket. Stoga kod komutatora sa memorisanjem i prosleđivanjem postoji izvesno kašnjenje na svakom ulaznom linku i to na Čitavoj putanji datog paketa. Ovo kašnjenje je proporcionalno dužini paketa u bitovima. Konkretno, kada bi se paket sastojao od L bitova i kada bi trebalo da se pošalje izlaznim linkom brzine R bitova u sekundi, kašnjenje memorisanja i prosleđivanja komutatora iznosilo bi L/R sekundi.

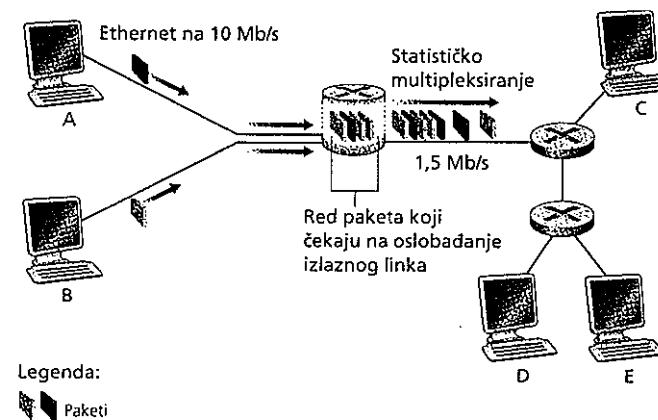
Svaki ruter je povezan sa većim brojem linkova. Za svaki od svojih linkova ruter ima izlaznu privremenu memoriju (naziva se i izlazni red čekanja) u koju se smještaju paketi koje ruter tek treba da pošalje kroz dati link. Upravo ova izlazna privremena memorija ima ključnu ulogu u komutiranju paketa. Ukoliko je link kroz koji neki paket treba da se pošalje trenutno zauzet, pristigli paket mora da sačeka u izlaznoj privremenoj memoriji. Zbog toga, osim kašnjenja usled memorisanja i prosleđivanja, pakete pogoda i kašnjenje usled Čekanja u redu. Ova kašnjenja variraju i zavise od nivoa zagrušenosti mreže. Budući daje količina prostora u privremenoj memoriji konačna, pristigli paket može da se suoči i sa situacijom daje ova memorija već popunjena drugim paketima koji čekaju u redu za prenos. U tom slučaju dolazi do gubljenja paketa - ispušta se ili upravo pristigli paket ili jedan od onih koji su već u redu. Ako se za trenutak vratimo analogiji sa restoranom, Čekanje u redu moglo bi da se poistoveti sa vremenom koje provedete Čekajući da se neki sto osloboodi, dok bi gubljenje paketa bilo jednakno obaveštenju da morate da odete zato što previše ljudi već čeka na oslobođanje nekog od stolova.

Na slici 1.7 prikazana je jednostavna mreža sa komutiranjem paketa. Na ovoj, kao i na sledećim slikama, paketi su predstavljeni trodimenzionalnim pločicama. Širina pločica predstavlja dužinu paketa. Na ovoj slici sve pločice imaju jednaku širinu, što znači da su svi paketi jednake dužine. Pretpostavimo da računari A i B šalju pakete računaru E. Ovi paketi najpre putuju Ethernet linkovima brzine 10 Mb/s, a zatim ih komutator preusmerava na link čija je brzina 1,5 Mb/s. Ukoliko se dogodi daje brzina kojom paketi pristižu veća od one kojom komutator može da ih prosledi kroz izlazni link brzine 1,5 Mb/s, paketi će, pre slanja kroz ovaj link, morati da stojte u redu u njegovoj izlaznoj privremenoj memoriji.

Pogledajmo sada koliko je vremena potrebno za slanje paketa od L bitova od jednog do drugog računara, kroz mrežu sa komutiranjem paketa. Pretpostavimo da između njih postoji Q mreža od kojih svaka ima brzinu od R bitova u sekundi. Pre-ispovadićemo i da su kašnjenja usled stajanja u redu, propagacije s kraja na kraj i uspostavljanja veze neznačna. Dakle, paket se, nakon izlaska iz računara A, najpre prenosi kroz prvi link i za to je potrebno L/R sekundi. Nakon toga, paket mora da prođe i kroz svaki od preostalih $Q-1$ linkova; odnosno, on mora da se memoriše i prosledi $Q-1$ puta. Prema tome, ukupno kašnjenje iznosi QL/R .

Poređenje komutiranja paketa i komutiranja vodova: statističko multipleksiranje

Sada kada smo vam opisali i komutiranje vodova i komutiranje paketa, uporedićeemo ove dve tehnologije. Protivnici komutiranja paketa uvek su tvrdili da ono, zbog svog



Slika 1.7 ♦ Komutiranje paketa

nepredvidivog i promenljivog kašnjenja između krajnjih tačaka (usled nepredvidivog i promenljivog trajanja Čekanja u redu), nije podesno za usluge u realnom vremenu (na primer, za telefonske pozive i video konferencije). S druge strane, zagovornici komutiranja paketa tvrde da ova tehnika (1) obezbeđuje bolju deobu propusne moći i daje (2) jednostavnija, efikasnija i jeftinija za implementiranje od komutiranja vodova. Veoma zanimljivo poređenje komutiranja paketa i komutiranja vodova možete pronaći kod autora [Molinero-Femandez]. Uopšteno govoreći, ljudi koji ne vole da razmišljaju o rezervacijama u restoranu prednost daju komutiranju paketa.

Zastoje komutiranje paketa efikasnije? Evo jednostavnog primera. Prepostavimo da korisnici zajednički koriste link od 1 Mb/s i da svaki korisnik ima period veće aktivnosti u kome generiše 100 kb/s i period mirovanja u kome ne generiše nikakve podatke. Prepostavimo, dalje, daje svaki korisnik aktivan oko 10 procenata vremena (ostalih 90 procenata piće kafu i ne generiše nikakav saobraćaj). Kod komutiranja vodova 100 kb/s mora da bude rezervisano za svakog korisnika i to sve vreme. Ukoliko bi, primera radi, u tehnologiji TDM komutiranja vodova okvir od 1 sekunda bio podeljen na 10 vremenskih odsečaka od po 100 ms, svakom korisniku bi tada bio dodeljen jedan vremenski odsečak po okviru.

To znači da link u jednom trenutku može da podrži samo 10 ($= 1 \text{ Mb/s}/100 \text{ kb/s}$) korisnika. Kod komutiranja paketa verovatnoća da je određeni korisnik aktivan iznosi 0,1 (ili 10 procenata). Kada bi u mreži bilo 35 korisnika, verovatnoća da njih 11 ili više u istom trenutku budu aktivni iznosi 0,0004. (U domaćem zadatku 8 videćete na koji način smo dobili ovaj rezultat.) Kada je istovremeno aktivno 10 ili manje korisnika (verovatnoća je 0,9996) prosečna brzina prenosa jednaka je ukupnoj propusnoj moći od 1 Mb/s, ili je nešto manja od nje. Prema tome, kada je u

mreži aktivno 10 ili manje korisnika, njihovi paketi se kroz link kreću praktično bez ikakvog kašnjenja, kao sto je slučaj u komutiranju vodova. U situacijama kada je istovremeno aktivno 10 ili više korisnika, prosečan broj prispevki paketa prevaziđa izlazni kapacitet linka i red na izlazu počinje da raste. (On raste sve dok prosečna ulazna brzina ne padne ispod 1 Mb/s, kada izlazni red počinje da se smanjuje.) Imajući u vidu to daje u ovom primeru verovatnoća da više od 10 korisnika bude istovremeno aktivno izuzetno mala, komutiranje paketa obezbeđuje praktično jednakе performanse kao i komutiranje vodova, ali za *tri puta veći broj korisnika*.

Predimo sada na drugi jednostavan primer. Prepostavimo da jedan od nekih deset korisnika iznenada generiše hiljadu paketa od po 1 000 bitova, a da ostalih devet miruje i ne generiše nikakav saobraćaj. Kod TDM komutiranja vodova sa 10 slotova od po 1 000 bitova po svakom okviru, aktivan korisnik može da koristi za prenos podataka samo jedan (svog) vremenski odsečak u okviru; preostalih devet vremenskih odsečaka biće neiskorišćeno. Za prenos jednog miliona korisnikovih bitova biće potrebno 10 sekundi. Kod komutiranja paketa aktivan korisnik može kontinuirano da šalje svoje podatke kroz link koristeći njegovu punu brzinu od 1 MB/s, s obzirom na to da nijedan drugi korisnik ne šalje pakete koje bi trebalo multipleksirati sa paketima aktivnog korisnika. U ovom slučaju svi korisnikovи podaci bili bi preneti za jedan sekund.

Prethodnim primerima pokušali smo da vam ilustrujemo dve situacije u kojima su performanse komutiranja paketa superiorne u odnosu na performanse komutiranja vodova. Istovremeno, ovim primerima istakli smo i najznačajniju razliku između ova dva načina deljenja propusne moći linka. Kod komutiranja vodova unapred se rezerviše deo propusne moći linka, što znači da onaj deo koji nije rezervisan ostaje neiskorišćen. Nasuprot tome, kod komutiranja paketa link se koristi *na zahtev*. Pre-nosni kapacitet linka, u smislu „*paket po paket*“ dele samo oni korisnici koji imaju pakete za prenos kroz link. Ovakva deoba resursa na zahtev (nasuprot dodeljivanju unapred) nekada se naziva i **statističko multipleksiranje** resursa.

Iako u današnjim telekomunikacionim mrežama postoji i komutiranje vodova i komutiranje paketa, trend se ipak sve više okreće ka komutiranju paketa. Mnoge telefonske mreže sa komutiranjem vodova lagano prelaze na komutiranje paketa. Konkretno, mnoge telefonske kompanije često koriste komutiranje paketa za skupe prekomorske telefonske veze.

1.3.2 Mreže sa komutiranjem paketa: mreže sa datagramima i mreže sa virtuelnim kolima

Postoje dve vrste mreža sa komutiranjem paketa - mreže sa datagramima i mreže sa virtuelnim kolima. Ove dve vrste mreža razlikuju se po tome da li njihovi komutatori za prosledivanje paketa do njihovog odredišta koriste odredišne adrese ili tzv. brojve virtuelnih kola. Sve mreže u kojima se za prosledivanje paketa koriste adrese odredišnih računara svrstajuemo u kategoriju mreža sa datagramima. Ruteri prosledjuju pakete upravo na ovaj način - dakle, Internet je mreža sa datagramima.

S druge strane, sve mreže u kojima se za prosleđivanje paketa koriste brojevi vir-tuelnih kola nazivamo **mrežama sa virtuelnim kolima**. U tehnologije komutiranja paketa u kojima se koriste virtuelna kola spadaju X-25, Frame Relay i ATM (*Asynchronous Transfer Mode*). Iako je razlika između mreža u kojima se koriste određene adrese i onih u kojima se koriste brojevi virtuelnih kola na prvi pogled gotovo beznačajna, od izbora jednog od ova dva standarda zavisi način podešavanja i administrirani a rutera.

Mreže sa virtuelnim kolima

Kao što iz samog imena možete da zaključite, **virtuelno kolo** (VC) predstavlja vir-tuelnu konekciju između izvornog i određenog računara. Međutim, uspostavljanje i održavanje virtuelnog kola podrazumeva i odgovarajuće podešavanje absolutno svakog rutera na putanji između ova dva računara. **Identifikator virtuelnog kola (VC ID)** dodeljuje se virtuelnom kolu prilikom inicijalnog uspostavljanja veze između izvora i odredišta. Svaki paket koji je deo virtuelnog kola u svom zaglavju mora imati VC ID broj. Svaki komutator paketa poseduje tabelu u kojoj su VC 1D brojevi povezani sa odgovarajućim izlaznim linkovima. Kada neki paket stigne do komutatora paketa, komutator ispituje njegov VC ID broj, indeksira svoju tabelu i prosle-duje dati paket ka odgovarajućem izlaznom linku. Skrećemo vam pažnju na to da se izvor i odredište virtuelnog kola samo indirektno identifikuju putem VC ID broja; za izvođenje komutiranja nisu neophodne stvarne adrese izvornog i određenog krajnjeg sistema. To znači da komutiranje paketa može da se izvede brzo (kontrolom VC ID broja pristiglog paketa u maloj tabeli za VC prevođenje, umesto da se određena adresa traži u potencijalno velikoj tabeli sa adresama).

Ukoliko se u nekoj mreži koriste virtuelna kola, njeni komutatori moraju da održavaju **informacije o stanju** aktivnih veza. Konkretno, kad god se u nekom komutatoru uspostavi nova veza, dodaje se novi zapis njegovoj tabeli prevođenja VC brojeva; svako prekidanje veze povlaci brisanje odgovarajućeg zapisa iz ove labelc. Skrećemo vam pažnju i na to daje neophodno evidentirati informacije o stanju koje povezuju VC brojeve sa brojevima izlaznih interfejsa. Čak i kada ne postoji nikakvo prevođenje VC ID brojeva. Evidentiranje informacija o stanju u komutatoru ma paketa predstavlja ključni aspekt ove tehnologije i na njega ćemo se ubrzo vratili.

Mreže sa datagramima

Mreže sa datagramima po mnogo čemu podsećaju na poštansku službu. Kada pošiljalac želi da pošalje pismo na određenu adresu, on ga najpre stavlja u kovertu na kojoj piše adresu primaoca. Ova određena adresa ima hijerarhijsku strukturu. Pri-mera radi, na pismima koja se šalju u Sjedinjene Američke Države potrebno je napisati naziv zemlje (SAD), države (recimo, Pensilvanija), grada (recimo, Filadelfija), ulice (Volnat Strit) i broja kuće u daljoj ulici (na primer, 421). Poštanska služba koristi adresu na koverti za usmeravanje pisma ka njegovom odredištu. Na primer, ukoliko se pismo šalje iz Francuske, onda će poštanska služba ove zemlje najpre da ga prosledi do poštanskog centra u SAD, koji će, dalje, da ga prosledi poštanskom centru u

Filadelfiji, da bi ga poštarski radi u ovom gradu na kraju doneo do njegovog konačnog odredišta.

U mrežama sa datagramima, svaki paket koji putuje kroz mrežu sadrži adresu svog odredišta u svom zaglavljku. Kao i kod poštanske službe, ova adresa ima hijerarhijsku strukturu. Kada do njega stigne određeni paket, komutator paketa ispituje deo njegove određene adrese i prosleđuje ga susednom komutatoru. Konkretno, svaki komutator paketa ima tabelu prosleđivanja u kojoj se određena adresa (ili njeni delovi) preslikava u neki izlazni link. Kada paket dospe do komutatora, ovaj odmah ispituje njegovu adresu i u svojoj tabeli pokušava da pronade odgovarajući izlazni link, kojim ga i usmerava dalje.

Proces rutiranja s kraja na kraj mogao bi da se uporedi i sa vozačem koji ne koristi auto-mape, već više voli da lično pita za put. Na primer, pretpostavimo da Džo iz Filadelfije treba da stigne do adrese 156Lejksajd Drajv u Orlandu na Floridi. Dakle, Džo se najpre dovezao do benzinske pumpe i pitao kako da stigne do navedene adrese. Radnik na pumpi je iz cele adrese izvukao deo Florida i uputio našeg Džoa na autoput 1-95 Jug, u koji može da se uključi odmah pored pumpe. Takođe mu je rekao da čim stigne do države Floride odmah od nekoga zatraži pomoć. Džo se vozio putem 1-95 Jug sve dok nije stigao do Džeksonvila u Floridi gde je ponovo svratio do benzinske pumpe i zainižio pomoć. Radnik na pumpi je iz cele adrese izvukao segment Orlando i rekao Džou da nastavi putem 1-95 Jug sve do Dejtona Biča i da tamo ponovo potraži pomoć. U Dejtona Biču se čovek na sledećoj pumpi ponovo vezao za segment Orlando i uputio Džoa na put 1-4 koji vodi direktno u Orlando. Džo se vozio ovim putem sve dok nije stigao do ulaska u Orlando. Na ulasku u Orlando Džo je ponovo stao na pumpi i zamolio za pomoć. Radnik na ovoj pumpi je iz cele adrese izvukao ulicu Lejksajd Drajv i objasnio Džou kako da stigne do nje. Konačno, kada je pronašao ovu ulicu, Džo je upitao jednog dečka na biciklu gde je broj 156. Ovaj dečko je iz cele adrese izvukao broj 156 i pokazao Džou gde se nalazi njegov konačno odredište.

U nastavku ove knjige, veoma detaljno ćemo razmatrati prosleđivanje paketa u mrežama sa datagramima. Na ovom mestu ćemo reći samo to da se, nasuprot VC mrežama, u rulerima mreža sa datagramima ne održavaju informacije o stanju veze. Štaviše, u Čistoj mreži sa datagramima, komutator paketa je potpuno nesvestan saobraćaja koji prolazi kroz njega. Komutator paketa donosi sve odluke o prosleđivanju na osnovu određene adrese paketa, a ne na osnovu veze kojoj dati paket pripada. S obzirom na to da je u VC mrežama neophodno održavati informacije o stanju veze - informacije koje moraju da se instaliraju i uklanjaju zajedno sa svakim novim virtuelnim kolom i sreduju (brišu) u slučaju njegovog prisilnog kraja, ove mreže zahtevaju postojanje relativno složenih protokola za održavanje stanja, koji nisu neophodni u mrežama sa datagramima.

Da li biste želeli da vidite putanju s kraja na kraj paketa na Internetu? Sada ćemo da vas pozovemo da malo „zasučete rukave“ i poiigrate se sa programom Tra-ceroute koji ćete naći na web lokaciji <http://www.traceroute.org>. (Obavezno pročitajte i odeljak 1.6 koji je posvećen ovom alatu.)

Taksonomija mreže

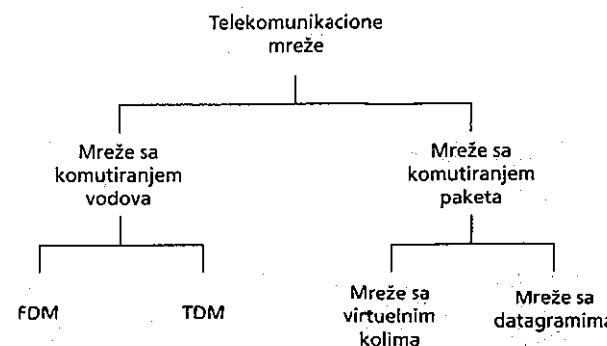
Do sada smo vam predstavili nekoliko veoma važnih koncepata umrežavanja kao što su komutiranje vodova, komutiranje paketa, virtualna kola, usluge bez konekcije i usluge sa konekcijom. Pogledajmo sada kako se svi ti koncepti uklapaju u jednu celinu.

Najpre, u našem pojednostavljenom primeru, u telekomunikacionoj mreži pri-menjuje se ili komutiranje vodova ili komutiranje paketa (slika 1.8). Zatim, na linku mreže sa komutiranjem vodova koristi se ili tehnologija FDM ili tehnologija TDM. Mreže sa komutiranjem paketa su ili mreže sa virtuelnim kolima ili mreže sa datagramima. Komutatori u mrežama sa virtuelnim kolima prosleđuju pakete u skladu sa VC brojevima paketa i održavaju informacije o stanju veze. Komutatori u mrežama sa datagramima prosleđuju pakete u skladu sa njihovim odredišnim adresama i ne održavaju informacije o stanju veze.

1.4 Pristup mreži i fizički medij umi

U odeljcima 1.2 i 1.3 ispitali smo uloge krajnjih sistema i rutera u računarskim mrežama, a u odeljku koji sledi bavićemo se mrežnim pristupom - odnosno, fizičkim vezama koje povezuju jedan krajnji sistem sa njegovim perifernim ruterom, što je, u stvari, prvi ruter na putanji između dva udaljena krajnja sistema. Pristupna mreža, dakle, obezbeđuje infrastrukturu za povezivanje potrošača sa mrežnom infrastrukturom.

Na slici 1.9 prikazano je nekoliko tipova pristupnih linkova koji povezuju krajnji sistem sa perifernim ruterom; ove veze iscrtane su debljim linijama. S obzirom na to da je tehnologija mrežnog pristupa tesno povezana sa tehnologijama fizičkih medijuma (optički i koaksijalni kablovi, telefonski kablovi sa upredenim paricama ili spektar radio talasa), u ovom odeljku ćemo se baviti i jednom i drugom temom.



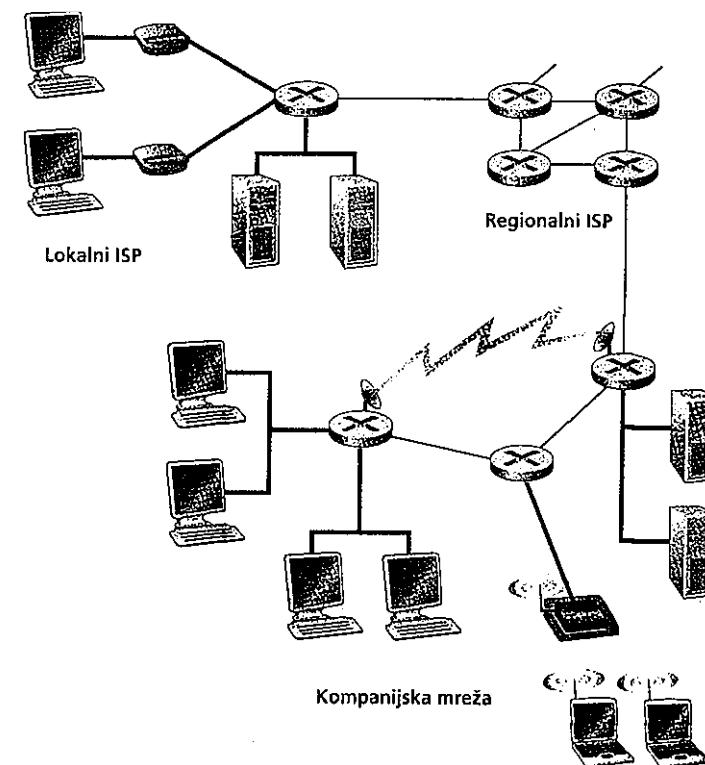
Slika 1.8 ♦ Taksonomija telekomunikacionih mreža

1.4.1 Pristupne mreže

Načini pristupa računarskim mrežama uopšteno govoreći mogu da se podele na tri sledeće kategorije:

- ♦ pristup od kuće - koji povezuje kućne krajne sisteme u zajedničku mrežu;
- ♦ poslovni pristup - koji povezuje krajne sisteme u poslovnim i obrazovnim institucijama u zajedničku mrežu;
- ♦ bežični pristup - koji povezuje mobilne krajne sisteme u zajedničku mrežu.

Ove kategorije, međutim, ne treba shvatiti previše striktno. Krajnji sistemi nekih kompanija, primera radi, mogu da koriste tehnologiju pristupa koju smo pripisati kućnom pristupu, i obratno. U tom smislu, opisi koji slede odnose se na najuobičajenije situacije.



Slika 1.9 ♦ Pristupne mreže

Pristup od kuće

Pod pristupom od kuće podrazumeva se povezivanje kućnog krajnjeg sistema (najčešće PC računara, ali i Web TV-a, pa čak i nekog drugog uređaja) sa perifernim ruterom. Za pristup od kuće najčešće se koriste standardni modem i obična analogna telefonska linija preko kojih se povezuju sa posrednikom za kućne Internet usluge (kao što je, na primer, America Online). Modem služe za pretvaranje digitalnih signala iz računara u analogni format, koji je podesan za prenos telefonskim linijama. U pomenutim analognim telefonskim linijama koriste se bakarni provodnici sa upredenim paricama i to su iste one linije koje koristimo i za obične telefonske razgovore. (O kablovima sa upredenim paricama govorimo u nastavku ovog odeljka.) Na drugoj strani analogne telefonske linije, modem posrednika za Internet usluge ove analogne signale враћa u digitalni format podesan za prosleđivanje ruteru. Prema tome, pristup mreži u ovom slučaju čini par modema na modemskoj telefonskoj liniji tipa od tačke do tačke. Današnji modemi omogućavaju brzine prenosa do 56 kb/s, ali je, zbog lošeg stanja linija između kućnog sistema i posrednika za Internet usluge, stvarna brzina prenosa često daleko manja.

Za mnoge korisnike koji pristupaju od kuće, modemski pristup brzine 56 kb/s je previše spor. Primera radi, za preuzimanje jedne trominutne pesme u MP3 formatu potrebno je oko osam minuta. Pored toga, modemski pristup blokira telefonsku liniju korisnika - dok ovu liniju koristi za krstarenje VVebom, korisnik ne može da koristi telefon. Srećom, nove tehnologije za širokopojasni pristup donele su kućnim korisnicima veće brzine prenosa, uz istovremeno oslobođanje telefonske linije. Za širokopojasni pristup od kuće obično se koriste dve tehnologije - **digitalna preplatnička linija (DSL)** [DSL 2004] i **hibridni optičko-koaksijalni kabl (HFC)** [Cable Labs 2004].

Do 2003. godine, širokopojasni pristup od kuće bio je mnogo manje zastupljen od standardnog modemskeg (56 kb/s) pristupa. Procentualni ideo domaćinstava sa širokopojasnim pristupom Internetu tada je iznosio 23% u Južnoj Koreji, 13% u Kanadi i 7% u SAD, dok je u Evropi bilo oko 10 procenata takvih domaćinstava [Point Topic 2003]. Međutim tehnologije DSL i HFC veoma brzo osvajaju svet - u SAD dominira tehnologija HFC, a u Evropi i Aziji DSL.

DSL pristup obično obezbeđuju telefonske kompanije (na primer, Verizon ili France Telecom), nekada i u saradnji sa nekim nezavisnim posrednikom za Internet usluge, lako je konceptualno sličan standardnim modemima, DSL predstavlja novu modemsku tehnologiju koja i dalje koristi standardne telefonske linije sa upredenim bakarnim paricama. Međutim, ovde se postiže daleko veće brzine prenosa, ograničavanjem udaljenosti između korisnikovog i ISP-ovog modema. Brzina prenosa obično je asimetrična - nešto brža od ISP-ovog rutera do korisnika nego u obrnutom smeru. Ova asimetrija brzine prenosa utemeljena je na konstrukciji daje kućni korisnik pre konzumenta nego proizvođač informacija (zato je veća brzina na putu informacija ka njemu). U teoriji, tehnologija DSL može da obezbedi brzine do 10 Mb/s od ISP-a do korisnika, i nešto više od 1 Mb/s u obrnutom smeru. Međutim, u praksi su te brzine ipak značajno manje. Godine 2004, uobičajene brzine prenosa kretale su se između 1 i 2 Mb/s ka korisniku, i nekoliko stotina kb/s ka posredniku za Internet usluge.

Tehnologija DSL koristi frekventno multipleksiranje koje smo opisali u prethodnom odeljku. Konkretno, ovde se komunikacioni link između ISP-a i domova deli na tri nepreklapajuća frekventna opsega:

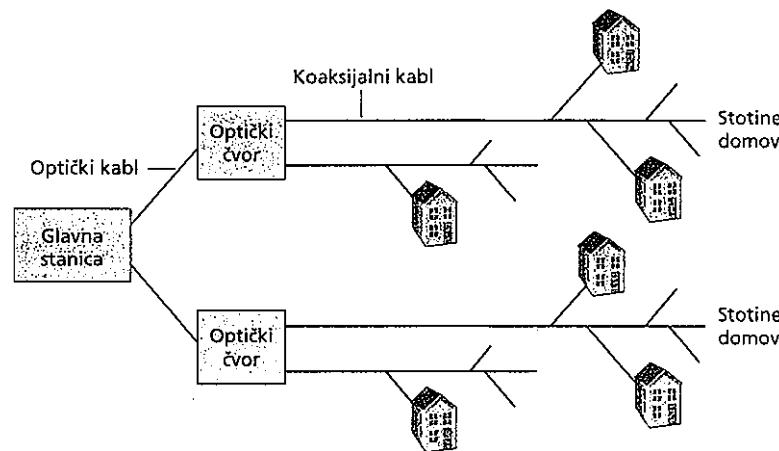
- ◆ veoma brzi kanal ka korisniku u opsegu od 50 kHz do 1 MHz;
- ◆ srednje brzi kanal ka ISP-u u opsegu od 4 kHz do 50 kHz i
- ◆ standardni dvosmerni telefonski kanal u opsegu od 0 do 4 kHz.

Stvarni nizvodni i uzvodni propusni opseg koji potencijalno стоји na raspolažanju datom korisniku zavisi od udaljenosti između njegovog i ISP-ovog modema, prečnika kabla sa upredenim paricama i stepena električnih smetnji. U stvari, za razliku od standardne modemske veze, tehnologija DSL je upravo i napravljena za male udaljenosti između kućnih modema i modema posrednika za Internet usluge, zahvaljujući čemu se postiže značajno veće brzine prenosa.

Dok DSL i standardni modemi koriste obične telefonske linije, HFC pristupne mreže predstavljaju proširenje mreža koje se koriste za emitovanje kablovske televizije. U današnjim kablovskim sistemima, kablovska glavna stanica emituje ka domovima kroz distribucionu mrežu koju čine koaksijalni kablovi i pojačivači. Kao što možete da vidite na slici 1.10, glavna stanica je optičkim kablovima povezana sa raskrsnicama u pojedinim delovima grada, od kojih vode klasični koaksijalni kablovi ka pojedinačnim domovima. Svaka ovakva raskrsnica obično podržava između 500 i 5 000 domova.

Poput tehnologije DSL, HFC takođe zahteva korišćenje posebnih modema koji se nazivaju kablovski modemi. Kompanije koje obezbeđuju kablovski pristup Internetu obično zahtevaju da njihovi korisnici ili kupe ili iznajme jedan ovakav uređaj. Kablovski modem je najčešće eksterni uređaj koji se sa računaram povezuje preko 10-BaseT Ethernet porta. (U poglavljju 5 ćemo se detaljnije pozabaviti Ethernetom.) Kablovski modemi dele HFC mrežu na dva kanala - nizvodni i uzvodni. Kao i kod tehnologije DSL, nizvodnom kanalu se obično dodeljuje veći propusni opseg, tako daje za njega karakteristična i veća brzina prenosa.

Veoma važna karakteristika tehnologije HFC jeste to što predstavlja deljeni emisioni medijum. Konkretno, svaki paket koji glavna stanica pošalje putuje istim nizvodnim linkovima do svakog doma; isto tako, svaki paket koji pošalje krajnji korisnik putuje istim uzvodnim kanalom do glavne stanice. Zbog toga, ukoliko više korisnika istovremeno preuzima MP3 datoteke putem nizvodnog kanala, stvarna brzina preuzimanja svakog od njih biće manja od maksimalne brzine nizvodnog kanala. S druge strane, ukoliko je samo nekoliko korisnika aktivno i svi oni krstare Webom, najverovatnije će svi dobijati tražene stranice punom brzinom zato što se retko dogada da više korisnika zatraži neku stranicu u apsolutno istom trenutku. S obzirom na to da se i uzvodni kanal deli, koristi se protokol za distribuirani istovremeni pristup koji koordinira prenos i izbegava kolizije. (O kolizijama ćemo takođe govoriti u poglavljju 5, u vezi sa Ethernet mrežama.) Zagovornici DSL mreža često ističu da DSL veže između domova i ISP-ova pripadaju tipu tačka-tačka, što znači daje celokupnu propusnu moć na raspolažanju i da nije deljena. Pristalice



Slika 1.10 ♦ Hibridni optičko-koaksijalni pristup mreži

kablovskih mreža, s druge strane, tvrde da HFC mreže razumnih dimenzija obezbeđuju veći propusni opseg od DSL mreža. Borba između tehnologija DSL i HFC za veoma brzi pristup od kuće tek se zahuktava.

Jedno izuzetno privlačno svojstvo DSL i HFC mreža jeste to sto su njihove usluge uvek dostupne. To znači da korisnik može da ostavi svoj računar uključen i povezan sa posrednikom za Internet usluge, a da istovremeno prima telefonske pozive ili sam nekoga poziva.

Poslovni pristup

U preduzećima i na univerzitetima, za povezivanje krajnjeg sistema i perifernog ruter-a obično se koristi lokalna računarska mreža (LAN). Kao što ćete videti u poglavljiju 5, postoji mnogo različitih tipova LAN tehnologija. Ipak, tehnologija Ethernet je trenutno ubedljivo najrasprostranjenija tehnologija pristupa u ovakvim mrežama. Brzina Etherneta je 10 Mb/s ili 100 Mb/s (sada čak i 1 Gb/s i 10 Gb/s). Za medusobno povezivanje većeg broja korisnika, kao i za njihovo povezivanje sa perifernim ruterom koriste se bakarni kablovi sa upredenim paricama ili koaksijalni kablovi. Periferni ruter je zadužen za rutiranje paketa čije je odredište izvan LAN-a. Poput tehnologije HFC, Ethernet podržava deljivi pristup medijumu, tako da krajnji korisnici između sebe deli propusnu moć LAN-a. Odnedavno Ethernet polako postaje komutirana tehnologija. U komutiranom Ethernetu koristi se više Ethernet segmenta sa upredenim paricama koji su povezani preko „komutatora“ čime se omogućava istovremena isporuka punog propusnog opsega različitim korisnicima LAN-a. O deljenom i komutiranom Ethernetu detaljnije ćemo govoriti u poglavljju 5.

Bežični pristup

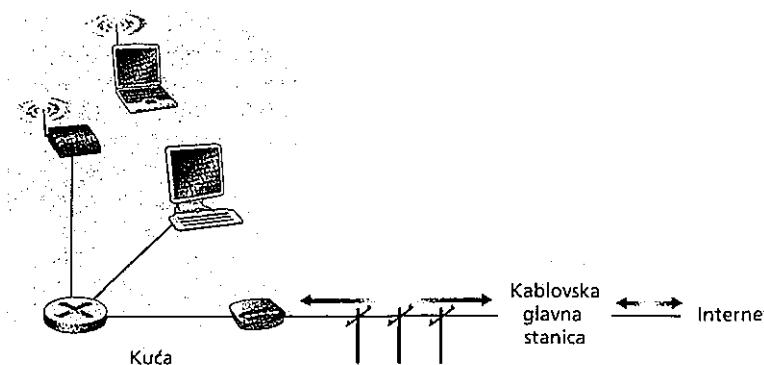
Osim aktuelne internetske revolucije, revolucija u oblasti bežičnih tehnologija takođe je imala izuzetan uticaj na život i rad savremenog čoveka. Godine 2000, u Evropi je više ljudi imalo mobilni telefon nego računar ili automobil. Mobilni trend se i dalje nastavlja, tako da se predviđa da će bežični (i često mobilni) ručni uređaji kao što su, na primer, mobilni telefoni ili PDA računari, preuzeći dominaciju od umreženih računara kada je u pitanju pristup Internetu. Danas postoje dve široke kategorije bežičnog pristupa Internetu. U bežičnim LAN-ovima mobilni korisnici emituju podatke ka baznoj stanicu (poznatoj i kao bežična tačka pristupa) ili ih primaju od nje i to u prečniku od nekoliko desetina metara. Bazna stanica je obično povezana sa standardnom žičanom vezom sa Internetom i služi za povezivanje bežičnih korisnika sa delom mreže u kome postoje kablovi. U regionalnim bežičnim pristupnim mrežama baznom stanicom upravlja posrednik za telekomunikacione usluge i ona obično može da usluži korisnike u prečniku od nekoliko desetina kilometara.

Bežični LAN-ovi koji su zasnovani na tehnologiji IEEE 802.11 (poznatoj po imenima bežični Ethernet i Wi-Fi) trenutno doživljavaju pravu ekspanziju na univerzitetima, zatim u preduzećima, kafeima i domovima. Primera radi, na univerzitetima na kojima rade autori ove knjige instalirane su bazne stanice IEEE 802.11. Ova bežična LAN infrastruktura omogućava studentima slanje i prijem elektronske pošte i krstarenje Webom iz bilo kog dela studentskog grada (iz biblioteke, spavaonice, učionice ili dvorišta). Tehnologija 802.11, o kojoj ćemo takođe govoriti u poglavljju 6, obezbeđuje deljeni propusni opseg od 11 Mb/s.²

Danas su u mnogim domovima, kombinacijom širokopojasnog kućnog pristupa (kablovskih ili DSL modema) sa jeftinom bežičnom LAN tehnologijom, dobijene moćne kućne mreže. Slika 1.11 predstavlja šematski prikaz prosečne kućne mreže (u stvari, to je upravo postavka mreže oba autora). Ovu kućnu mrežu Čine jedan prenosivi laptop i tri stacionarana računara (dva su povezana kablovima, jedan nije), zatim bazna stanica (bežična pristupna tačka) koja komunicira sa pokretnim računa-rom, kablovski modem koji obezbeđuje pristup Internetu i ruter koji povezuje baznu stanicu i stacionarni računar sa kablovskim modemom. Ovakva mreža omogućava Članovima domaćinstva da ostvare širokopojasni pristup Internetu, a da se jedan od njih, pri tom, slobodno kreće kroz kuću. Ukupna fiksna cena ovakve mreže iznosi manje od 250 dolara (uključujući i kablovski /DSL modem).

Kod pristupa Internetu putem bežične LAN tehnologije obično treba da budete na nekoliko metara od bazne stanice. To je ostvarivo kada se nalazite u stanu, ili u Internet kafeu ili, uopšteno govoreći, kada ste u nekoj zgradi. Ali šta ako se nalazite na plaži ili u kolima, a potreban vam je pristup Internetu? Za ovakav regionalni pristup pokretni korisnici Interneta koriste infrastrukturu mobilnih telefona i preko nje pristupaju baznim stanicama koje mogu da se nalaze i nekoliko desetina kilometara od njih.

² Ovo važi za 802.11b dok 802.11g ima 54 MB/s! (prim. rec.)



Slika 1.11 ♦ Šematski prikaz prosečne kućne mreže

WAP (*Wireless Access Protocol*, version 2) [WAP 2004] koji je rasprostranjen u Evropi, kao i i-mode koji je rasprostranjen u Japanu, predstavljaju dve tehnologije koje omogućavaju pristup Internetu putem infrastrukture mobilnih telefona. WAP telefoni, koji podsećaju na standardne bežične telefone sa nešto krupnjim ekranom, omogućavaju relativno dobar pristup Internetu, male i srednje brzine, kao i bežičnu telefonsku uslugu. Umesto jezika HTML, WAP telefoni koriste poseban marker-ski jezik - WML (*WAP Markup Language*) - koji je optimiziran za manje ekrane i manju brzinu pristupa. U Evropi protokol WAP funkcioniše preko izuzetno uspešne GSM infrastrukture za bežičnu telefoni ju, pri Čemu se WAP 2.0 oslanja na TCP/IP protokole. S druge strane, zaštićena tehnologija i-mode, koja je konceptualno i funkcionalno veoma slična tehnologiji WAP, u Japanu je doživela veliki uspeh.

Telekomunikacione kompanije trenutno preduzimaju velika ulaganja u bežičnu tehnologiju 3G (*Third Generation*) koja bi trebalo da obezbedi regionalni bežični pristup Internetu uz brzine koje prelaze 384 kb/s [Kaaranen 2001][Korhonen 2003], 3G sistemi bi trebalo da obezbede pristup velike brzine Webu i interaktivnom videu, kao i kvalitet glasa koji je superioran u odnosu na klasičnu telefoniju. Prvi 3G sistemi su primenjeni u Japanu. Imajući u vidu ogromna ulaganja u 3G tehnologiju, njenu infrastrukturu i licence, mnogi analitičari (i investitori) pitaju se da li će ova tehnologija zaista biti toliko uspešna koliko se pretpostavlja, ili će možda izgubiti bitku od neke druge tehnologije kao Stope, na primer, IEEE 802.11? ili će se, možda, kombinacijom ove dve tehnologije dobiti sveopštii ali heterogeni pristup. (Pročitajte [VVeinstein 2002] i istorijski osrt u odeljku 6.2.) O tehnologijama 802.11 i 3G detaljnije ćemo govoriti u poglavljju 6.

1.4.2 Fizički medij umi

U prethodnom odeljku prikazali smo vam neke od najvažnijih tehnologija za ostvarivanje mrežnog pristupa koje se koriste na Internetu. Opisujući ove tehnologije, naveli smo i fizičke medijume koji se u njima koriste. Na primer, rekli smo da se u tehnologiji HFC koristi kombinacija optičkih i koaksijalnih kablova, dok standardni modemi brzine 56 kb/s i tehnologija ADSL koriste bakarne kable sa upredenim paricama. Rekli smo i to da se u mrežama sa mobilnim pristupom kao medijum koristi spektar radio talasa. U ovom odeljku ćemo ukratko opisati ove i druge preno-sne medijume koji se često koriste u okviru Interneta.

Da bismo definisali Šta se podrazumeva pod fizičkim medijumom, zamisli-ćemo kako izgleda kratak život jednog bita. Dakle, zamislite jedan bit koji putuje od jednog krajnjeg sistema, kroz seriju linkova i rutera, do drugog krajnjeg sistema. Taj bit mora da se prenese nebrojeno mnogo puta. Izvorni krajnji sistem emituje ovaj bit i ubrzo zatim njega prima prvi ruter u nizu. Zatim ga taj prvi ruter dalje emituje da bi ga nakon toga primio drugi ruter i tako redom. Prema tome, naš bit, putujući od svog izvora da svog odredišta, mora da prode kroz seriju primopredajnih parova. Između svih ovih primopredajnih parova bit putuje u vidu elektromagnetskog talasa, električnog ili optičkog impulsa, kroz fizičke medijume. Fizički medijumi imaju mnogo različitih oblika i ne moraju biti istog tipa između svakog primopredajnog para na putanji. U primere fizičkih medijuma ubrajaju se bakarni kablevi sa upredenim paricama, zatim koaksijalni kablevi, multimodna optička vlakana, zemaljski spektar radio talasa i satelitski spektar radio talasa. Postoje dve kategorije fizičkih medijuma - voden i nevoden medijumi. Kod vodenih fizičkih medijuma signali se vode kroz čvrst medijum kao što su, na primer, kablevi od optičkih vlakana, bakarni kablevi sa upredenim paricama ili koaksijalni kablevi. Kod nevodenih medijuma talasi se šire kroz atmosferu ili kroz vavionski prostor, kao sto je to slučaj u bežičnim LAN-ovima ili kod digitalnih satelitskih kanala.

Pre nego što predemo na karakteristike različitih tipova medijuma, recimo nekoliko reči i o njihovim cenama. Stvarna cena fizičkog linka (bakarnog ili optičkog kabla) često je praktično zanemarljiva u poređenju sa ostalim troškovima za mrežu. Na primer, troškovi vezani za radnu snagu kod naknadne instalacije fizičkih linkova. Često premašuju cenu samog materijala. Iz tog razloga, u mnogim novijim zgradama su u svakoj prostoriji instalirani i kablevi sa upredenim paricama i optički i koaksijalni kablevi. Čak i ako se u startu koristi samo jedan od ovih medijuma, velike su šanse da će u bliskoj budućnosti zatrebati i neki drugi, tako da se na ovaj način smanjuju troškovi naknadnog postavljanja kablova.

Bakarni kablovi sa upredenim paricama

Najjeftiniji i najrasprostranjeniji vodeni prenosni medijum jesu bakami kablevi sa upredenim paricama. Ove kableve već više od 100 godina koriste telefonske mreže. U stvari, na više od 99 procenata ožičenih veza između telefonskog aparata i lokalnog telefonskog komutatora koriste se upravo bakami kablevi sa upredenim paricama. Ove kableve ste sasvim sigurno mnogo puta videli u svom domu. Sastoje se

od dva izolovana bakama provodnika debljine oko 1 mm koji su spiralno upredeni. Upredanjem provodnika smanjuju se elektromagnetne smetnje drugog para provodnika koji bi mogao da se nalazi u blizini. Obično se ovaj par provodnika grupiše u kabl tako što se uvija u zaštitni oklop. Jedan par provodnika čini jedan komunikacioni link. **Neoklopljene upredene parice** (*Unshielded TwistedPairs*, UTPs) veoma često se koriste u lokalnim računarskim mrežama u okviru iste zgrade. Brzina prenosa podataka kroz ovakve kablove kreće se između 10 Mb/s i 1 Gb/s i zavisi od debljine kablova i udaljenosti između predajnika i prijemnika.

Kada se 1980-ih pojavila tehnologija optičkih vlakana, mnogi ljudi su počeli da omalovažavaju kablove sa upredenim paricama zbog njihove relativno male brzine prenosa. Neki su čak išli toliko daleko da su predviđali da će tehnologija optičkih vlakana potpuno potisnuti kablove sa upredenim paricama. Međutim, upredene parice se nisu tako lako predale. Savremena tehnologija upredenih parica, kao što su kablovi kategorije 5, omogućava brzine prenosa od 100 Mb/s na udaljenostima od nekoliko stotina metara. Na kraćim udaljenostima moguće su čak i veće brzine prenosa. Danas kablovi sa upredenim paricama i dalje predstavljaju dominantno rešenje za lokalne računarske mreže velike brzine.

Kao što smo već napomenuli u odeljku o pristupnim mrežama, kablovi sa upredenim paricama veoma često se koriste za pristup Internetu od kuće. Videli ste da tehnologija telefonskih modema omogućava brzine prenosa do 56 kb/s kroz kablove sa upredenim paricama. Ali, takođe ste videli i to je da tehnologija DSL (*Digital Subscriber Line*) omogućila kućnim korisnicima pristup Internetu brzinom od oko 6 Mb/s preko istih tih kablova sa upredenim paricama (ukoliko žive blizu ISP-ovog modema).

Koaksijalni kablovi

Poput kablova sa upredenim paricama, koaksijalni kablovi se takođe sastoje od dva bakarna provodnika, s tom razlikom što oni ovde nisu postavljeni paralelno, već koncentrično. Zahvaljujući ovakvoj konstrukciji, posebnoj izolaciji i oklopu, ovi kablovi omogućavaju veće brzine prenosa. Ova vrsta kablova veoma često se koristi u kablovskim televizijskim sistemima. Kao što smo već rekli, odnedavno proširenje ovih sistema predstavljaju kablovski modemi koji kućnim korisnicima omogućuju pristup Internetu koji može da bude brži od 1 Mb/s. Kod širokopojasnih koaksijalnih kablova predajnik podiže signal u neki konkretan frekventni opseg, a zatim se rezultujući analogni signal šalje ka jednom prijemniku ili ka više njih. Obe varijante koaksijalnih kablova mogu da se koriste kao vođeni **deljeni medij u m.** To, konkretno, znači da sa istim kablom može da bude povezan veći broj krajnjih sistema i svi oni tada primaju sve ono što šalju svi drugi krajnji sistemi.

Kablovi od optičkih vlakana

Optički kablovi su tanki i fleksibilni medijumi koji provode svetlosne impulse, a svaki svetlosni impuls predstavlja jedan bit. Jedan optički kabl može da podrži

izuzetno velike brzine prenosa, čak do nekoliko desetina pa i stotina gigabita po sekundi. Ovi kablovi su imuni na elektromagnetne smetnje, imaju neznatno slabljenje signala do razdaljine od 100 kilometara i veoma teško se prisluškuju. Zahvaljujući ovim karakteristikama, optički kablovi su najbolji vođeni prenosni medijumi za duže relacije, posebno kada su u pitanju prekomorske veze. U mnogim međugradskim telefonskim mrežama u SAD, ali i u drugim krajevima sveta, sada se koriste isključivo kablovi od optičkih vlakana. U okosnici Interneta takođe dominira ova vrsta kablova. Međutim, visoka cena optičkih uređaja, kao što su predajnici, prijemnici i komutatori ograničila je njihovu primenu u transportu na kraćim relacijama - recimo, u LAN-ovima ili domovima. U radovima [IEC Optical 2003], [Goralski 2001], [Ramasvami 1998] i [Mukherjee 1997] pronaći će veoma detaljan prikaz raznih aspekata optičkih mreža. U linkovima od optičkih vlakana brzine prenosa obično se mere desetinama gigabajta u sekundi.

Zemaljski radio talasi

Radio stanice prenose signale koristeći elektromagneti spektar. Oni predstavljaju veoma zanimljiv medijum zato što ne zahtevaju instaliranje kablova, mogu da prođu i kroz zidove, obezbeđuju povezivost mobilnim korisnicima i prenose signale na velike daljine. Karakteristike radio kanala u velikoj meri zavise od karakteristika sredine kroz koju se prenose, kao i udaljenosti koje treba da premoste. Sredina kroz koju se signali prenose može na njih da utiče u smislu gubitka putanje i slabljenja zbog senke (jačina signala se smanjuje sa povećanjem udaljenosti i na putu oko prepreka ili kroz njih), slabljenja usled višestrukih putanja (signali se odbijaju o objekte koji im se nadu na putu) i smetnji (koje stvaraju drugi radio kanali ili elektromagnetni signali).

Zemaljski radio kanali mogu grubo da se podele u dve grupe - one koji pokrivaju lokalne oblasti (od nekoliko desetina pa do stotinak metara) i one koji pokrivaju veće oblasti (nekoliko desetina kilometara). Bežični uređaji za LAN koje smo pomenuli u odeljku 1.4.1 koriste radio talase lokalnog dometa; s druge strane, tehnologije WAP, i-mode i 3G koje smo takođe pominjali u odeljku 3.4.3, koriste radio kanale regionalnog dometa. Detaljno istraživanje ove tehnologije i pratećih proizvoda pronaći će u radu [Doman 2001]. O radio talasima detaljnije ćemo govoriti u poglavljju 6.

Satelitski radio kanali

Komunikacioni sateliti povezuju dva ili više zemaljskih predajnika i prijemnika koji se nazivaju zemaljskim stanicama. Satelit prima emisije u jednom frekventnom opsegu, zatim regeneriše primljeni signal pomoću repetitora (objašnjeno u nastavku teksta) i onda emituje signal u drugoj frekvenciji. Propusni opseg koji omogućavaju sateliti meri se gigabitima po sekundi. Za potrebe komunikacija koriste se dve vrste satelita - **geostacionarni i niski sateliti.**

Geostacionarni sateliti trajno ostaju iznad iste tačke na Zemlji. Nepromenljivost pozicije postiže se postavljanjem satelita u orbitu na visini od 36 000 kilometara iznad površine Zemlje. Zbog ogromne udaljenosti koju signal treba da prede od

³ Ovo u praksi nikada nije slučaj, (prim. rec.)

zemaljske stanice do satelita i natrag do druge zemaljske stanice, javlja se kašnjenje od 250 milisekundi. Međutim, i pored ovog kašnjenja, satelitski linkovi koji obezbeđuju brzine od nekoliko stotina Mb/s, često se koriste u telefonskim mrežama i internetskim okosnicama.

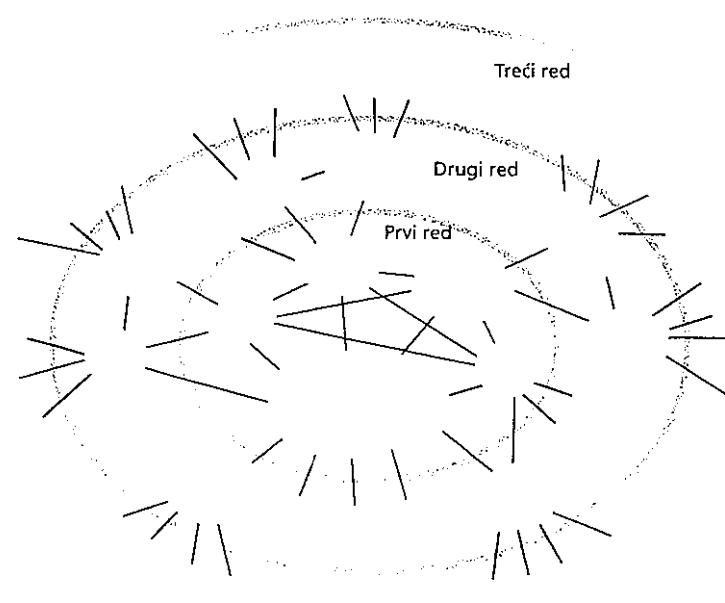
Niski sateliti postavljaju se na mnogo manjoj udaljenosti od Zemlje. Oni ne ostaju sve vreme iznad iste tačke na Zemljinoj površini već, poput Meseca, rotiraju oko naše planete. Za kontinuiranu pokrivenost odredene oblasti potrebno je mnogo satelita koji su postavljeni u orbitu. Trenutno ima mnogo niskih komunikacionih sistema koji su u fazi razvoja. Lojdova veb stranica o konstelaciji satelita [Wood 2004] predstavlja dobar izvor informacija o položaju satelitskih komunikacionih sistema. Moguće je da će se u budućnosti za pristup Internetu koristiti upravo tehnologija niskih satelita.

1.5 Posrednici za Internet usluge i internetske okosnice

Već smo rekli da se krajnji korisnici (PC i PDA računari, veb serveri, serveri za elektronsku poštu, itd.) povezuju sa Internetsom putem pristupne mreže. Podsećamo vas da ta pristupna mreža može biti ožičena ili bežična lokalna računarska mreža (na primer, u nekoj kompaniji, školi ili biblioteci) ili mreža ISP-a (na primer, AOL ili MSN) do koje se dolazi putem standardnog, kablovskog ili DSL modema. Međutim, povezivanje krajnjih korisnika i obezbeđivača raznih sadržaja u pristupne mreže predstavlja samo mali deo slagalice koju čine stotine miliona korisnika i stotine hiljada mreža od kojih se sastoji Internet. Internet može da se definije kao *mreža svih mreža* i razumevanje ove fraze je ključ za rešavanju ove velike slagalice.

U javnom Internetu, pristupne mreže koje se nalaze na njegovoj periferiji povezane su sa ostatkom Interneta kroz slojevitu hijerarhiju posrednika za Internet usluge (**ISP**), kao što je to prikazano na slici 1.12. Pristupni ISP-ovi (na primer, rezidencijalni posrednici za Internet usluge) kao što je AOL i kompanijski posrednici koji koriste LAN-ove) nalaze se na dnu ove hijerarhije. Na njenom vrhu nalazi se relativno mali broj tzv. ISP-ova prvog reda. ISP prvog reda bi u mnogo čemu mogao da se poistoveti sa bilo kojom drugom mrežom - on ima linkove i rutere i povezan je sa drugim mrežama. S druge strane, ISP-ovi prvog reda imaju i neke jedinstvene karakteristike. Brzine njihovih linkova često prelaze 622 Mb/s, a veći posrednici prvog reda imaju čak i linkove brzine 2,5 do 10 Gb/s; njihovi ruteri su, u skladu sa tim, sposobni za prosleđivanje paketa veoma velikom brzinom. Posrednici za Internet usluge prvog reda se, pored onoga što smo već rekli, prepoznaju po sledećim karakteristikama:

- ◆ direktno se povezuju sa *svim* drugim posrednicima prvog reda;
- ◆ povezani su sa velikim brojem posrednika drugog reda, kao i sa ostalim komercijalnim mrežama;
- ◆ pokrivaju veći broj zemalja.



Slika 1.12 ♦ Međusobna povezanost posrednika za Internet usluge

Posrednici za Internet usluge prvog reda obično se nazivaju **internetskim okosnicama**. Među najvećim kompanijama okosnicama nalaze se Sprint, MCI (nekadašnji UUNetAVworldCom), AT&T, Level3 (koji je kupio Genuity), Qwest i Cable and Wireless. Sredinom 2002. godine kompanija WorJdCom bila je ubedljivo najveći internetski posrednik prvog reda - po mnogim parametrima čak dva puta veći od najbližeg rivala [Telegraph 2002]. Veoma je zanimljivo i to da nijedna grupa nije zvanično zadužena za posrednike prvog reda; ukoliko treba da pitate da li ste Član grupe, onda to najverovatnije niste.

Posrednici za Internet usluge drugog reda obično imaju regionalni ili nacionalni značaj i (stoje veoma značajno) povezani su samo sa nekoliko posrednika prvog reda (slika 1.32). Prema tome, da bi mogao da stigne do svih delova globalnog Interneta, ISP drugog reda mora da rutira svoj saobraćaj kroz mrežu posrednika prvog reda sa kojim je povezan. U međusobnoj hijerarhiji ISP drugog reda je **korisnik**, dok je ISP prvog reda **dobavljač**. Mnoge velike kompanije povezuju svoje mreže direktno sa posrednicima prvog ili drugog reda i na taj način postaju njihove mušte-

rije. ISP u poziciji dobavljača naplaćuje svojim mušterijama određenu naknadu čija je visina obično povezana sa propusnim opsegom linka koji ih povezuje. Posrednik drugog reda može i direktno da se poveže sa mrežom drugog posrednika drugog reda i u tom slučaju saobraćaj između njih ne mora da ide kroz mrežu posrednika prvog reda. Ispod posrednika drugog reda nalaze se posrednici nižeg reda koji se sa ostatom Internetom povezuju preko jednog posrednika drugog reda. Na dnu ove hijerarhije nalaze se pristupni posrednici za Internet usluge. Da situacija bude još složenija pobrinuli su se neki posrednici prvog reda koji su istovremeno i poreznici za Internet usluge drugog reda (vertikalno su integrirani) i prodaju pristup Internetu direktno krajnjim korisnicima i obezbeđivačima sadržaja, ali i posrednicima nižeg reda. Za dva ISP-a koji su direktno povezani kaže se da su u **ravnopravnom odnosu**. Autori veoma zanimljive studije [Subramanian 2002] pokušali su preciznije da definišu slojevitost Interneta analizom njegove topologije u smislu termina kori-snik-dobavljač i ravnopravnih odnosa.

U okviru mreže posrednika za Internet usluge, tačke u kojima se dati posrednik povezuje sa drugim posrednicima (svejedno da lije to u hijerarhiji na istom nivou, iznad ili ispod) nazivaju se **prikљučnim tačkama** (*Point of Presence, POP*). Priklučnu tačku čini grupa rutera mreže jednog posrednika za Internet usluge sa kojima su povezani ruteri nekog drugog posrednika. Posrednik prvog reda obično ima mnogo priključnih tačaka koje se nalaze na različitim geografskim lokacijama i sa kojima je obično povezano više posrednika u rangu korisnika. Posrednici u rangu korisnikao-bično za povezivanje sa posrednikom ranga dobavljača koriste linkove velike brzine koje iznajmljuju od dobavljača telekomunikacionih usluga. Dva ISP-a prvog reda mogu da ostvare ravnopravan odnos međusobnim povezivanjem svojih priključnih tačaka.

Pored međusobnog povezivanja u privatnim ravnopravnim tačkama, posrednici za Internet usluge se često povezuju u mrežnim **pristupnim tačkama** (*Network Access Point, NAP*) koje su u vlasništvu neke nezavisne telekomunikacione kompanije koja njima upravlja ili pripadaju intemetskim okosnicama. Kroz ove tačke cir-kuliše velika količina saobraćaja između većeg broja posrednika za Internet usluge. Ipak, posrednici prvog reda u sve većoj meri zaobilaze NAP tačke i direktno se povezuju u privatnim ravnopravnim tačkama [Kende 2000]. Trenutno je trend takav da se ISP-ovi prvog reda međusobno direktno povezuju u privatnim tačkama, a da se posrednici drugog reda između sebe i sa posrednicima prvog reda povezuju preko NAP tačaka. S obzirom na to da NAP tačke prenose i komutiraju ogromne količine saobraćaja, one same po sebi predstavljaju složene i veoma brze mreže sa komutiranjem, koje se često koncentrišu u jednoj zgradi.

Dakle, mogli bismo da zaključimo daje topologija Interneta izuzetno složena i da je čine desetine ISP-ova prvog i drugog reda, kao i hiljade posrednika nižeg reda. Ovi posrednici se između sebe razlikuju prema oblasti koju pokrivaju - neki se prostiru na više kontinenata, dok su drugi prisutni samo u veoma malim delovima sveta. Posrednici nižeg reda povezuju se sa posrednicima višeg reda, koji za međusobno

povezivanje (obično) koriste privatne ravnopravne tačke i NAP tačke. Korisnici i kompanije koje obezbeđuju sadržaj na Internetu predstavljaju korisnike posrednika nižeg reda, koji su, opet, korisnici ISP-ova višeg reda.

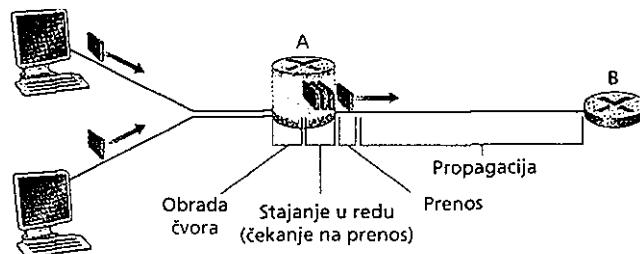
Ovaj odeljak ćemo završiti napomenom da svako ko ostvari vezu sa Internetom može da postane ISP. Potrebno je samo kupiti neophodnu opremu (recimo, ruter i odgovarajući broj modema) i omogućiti ostalim korisnicima da se povezu sa njom. To znači daje uključivanje novih posrednika nižih redova onoliko jednostavno koliko i dodavanje Lego kockica u neku postojeću Lego konstrukciju.

1.6 Kašjenje i gubitak paketa u mrežama sa komutiranjem paketa

Pošto smo ukratko prikazali osnovne delove strukture Interneta - aplikacije, krajnje sisteme, transportne protokole sa kraja na kraj **mreže**, rutere i linkove, pogledajmo šta se dešava sa paketom na njegovom putu od izvora do odredišta. Podsećamo vas da putovanje paketa započinje u (izvornom) računam, da on zatim prolazi **kroz** seriju rutera i završava u drugom (odredišnom) računam. Na putu od jednog (računar ili ruter) do drugog čvora (opet, računar ili ruter), paket **moe** da zadesi nekoliko različitih tipova kašnjenja i to na *svakom* od Čvorova na njegovoj putanji. Među ovim kašnjenjima najznačajnija su: **kašjenje usled obrade na čvoru**, **kašjenje usled stajanja u redu**, **kašjenje usled prenosa** i **kašjenje usled propagacije**. Sabiranjem svih ovih kašnjenja dobija se **ukupno kašjenje Čvora**. Da biste mogli da razumete proces komutiranja paketa, kao i računarske **mreže** u celini, veoma je važno da shvatite prirodu i značaj pomenutih kašnjenja.

1.6.1 Tipovi kašnjenja

Razmotrimo **sva** pomenuta kašnjenja u kontekstu slike 1.13. Kao deo sopstvene putanje između izvornog i odredišnog računara, paket se šalje iz Čvora koji emituje uzvodno **kroz** ruter A do ruteru B. U ovom slučaju cilj nam je da ustanovimo kašjenje čvora na ruteru A. Skrećemo vam pažnju na to da ruter A ima izlazni link koji vodi ka ruteru B. Ispred ovog linka nalazi se red (ili privremena memorija ruteru). Kada paket iz uzvodnog toka stigne do ruteru A, ruter ispitivanjem zaglavila paketa ustanavljava odgovarajući izlazni link za dati paket i odmah ga usmerava ka tom linku. U **ovom** primeru to je link koji vodi ka ruteru B. Prenos paketa linkom moguć je samo **ako** se tim linkom trenutno ne prenosi nijedan drugi paket i ako u redu za prenos **nema** drugih paketa. Ukoliko je link trenutno zauzet ili ukoliko postoje paketi koji trenutno čekaju u redu za prenos, novopristigli paket mora da stane u red.



Slika 1.13 ♦ Kašnjenje čvora na ruteru A

Kašnjenje usled obrade

Vreme koje je potrebno za ispitivanje zaglavja paketa i donošenje odluke o njegovom usmeravanju naziva se **kašnjenje usled obrade**. Na kašnjenje usled obrade mogu da utiču i drugi faktori, na primer, trajanje provere grešaka na nivou bitova u paketima do kojih je došlo prilikom uzvodnog prenosa od računara do ruter A. Kašnjenje usled obrade se u veoma brzim ruterima meri mikrosekundama, pa Čak i manjim jedinicama.

Nakon ove obrade, ruter usmerava paket u red koji se nalazi ispred linka ka rutemu B. (U poglavlju 4, detaljno ćemo vam prikazati način rada ruter.)

Kašnjenje usled stajanja u redu

U redu u kome paket mora da sačeka na upućivanje na link nastaje **kašnjenje usled stajanja u redu**. Ovo kašnjenje uvek zavisi od broja paketa koji su pre datog paketa već stali u red i Čekaju na prenos. Ukoliko je red prazan i na linku nema nijednog drugog paketa, onda je kašnjenje usled stajanja u redu datog paketa jednako nuli. S druge strane, ukoliko je saobraćaj obiman i mnogo paketa stoji u redu, kašnjenje usled stajanja u redu biće značajno. Uskoro ćete videti da broj paketa koje novopristigli paket može da očekuje da će zateći predstavlja funkciju intenziteta i prirode saobraćaja koji stiže u red. U praksi, ovaj vid kašnjenja meri se mikrosekundama ili milisekundama.

Kašnjenje usled prenosa

Ukoliko pretpostavimo da se paketi opslužuju i prenose onim redom kojim su pristigli, Stote i praksa u mrežama sa komutiranjem paketa, prenos našeg paketa može da se nastavi tek nakon prenosa svih **paket**a koji su stigli pre njega. Dužinu paketa izra-zičemo brojem L bitova, a brzinu prenosa linka koji povezuje ruter A i B brojem R bitova u sekundi. Brzinu R određuje brzina prenosa linka do ruteru B. Primera radi, u linku tipa 3 0 Mb/s Ethernet, R iznosi 10 Mb/s; u linku 100 Mb/s Ethernet, brzina R iznosi 100 Mb/s. **Kašnjenje usled** prenosa (naziva se još i kašnjenje memorisanja i

prosleđivanja, o čemu je bilo reči u odeljku 1.3) dobija se kada se L podeli sa R . To je vreme koje je potrebno da se svi bitovi paketa izbace (prenesu) na link. U praksi je ovo kašnjenje obično reda veličine mikrosekundi ili milisekundi.

Kašnjenje usled propagacije

Kada bit dospe na link, njega je potrebno transportovati do rutera B. Vreme potrebno za transport paketa od početka linka do rutera B naziva se kašnjenje usled propagacije. Bitovi paketa transportuju se brzinom propagacije linka koja zavisi od fizičkog medijuma linka (to mogu biti optički ili bakami kablovi) i kreće se u opsegu:

$$2 \cdot 10^8 \text{ metara u sekundi} \text{ do } 3 \cdot 10^8 \text{ metara u sekundi.}$$

Ova brzina je praktično jednaka brzini svetlosti. Kašnjenje usled propagacije izračunava se tako što se udaljenost između dva ruteru podeli sa brzinom propagacije (d/s , gde d predstavlja udaljenost ruteru A i B, a s brzinu propagacije lirika). Čim se i poslednji bit paketa transporiše do rutera B, on i svi ostali bitovi datog paketa se memoru u ovom ruteru. Zatim se ceo proces ponavlja, s tom razlikom što sada ruter B prosleđuje paket. U regionalnim računarskim mrežama kašnjenje usled propagacije meri se milisekundama.

Poređenje kašnjenja usled prenosa i kašnjenja usled propagacije Ljudi koji nemaju mnogo iskustva u oblasti umrežavanja često ne mogu da naprave razliku između kašnjenja usled prenosa i kašnjenja usled propagacije. Ova razlika je, doduše, mala, ali veoma značajna. Kašnjenje usled prenosa predstavlja vreme koje je ruteru potrebno da izgura paket; ono zavisi od dužine paketa i prenosne brzine linka, ali nema nikakve veze sa udaljenošću dva ruteru. Kašnjenje usled propagacije, s druge strane, predstavlja vreme za koje se bit transportuje od jednog do drugog ruteru; ono zavisi od udaljenosti između ruteru ali nema nikakve veze sa dužinom paketa niti sa brzinom prenosa datog linka⁴.

Da bismo vam pomogli u razlikovanju ovih pojmove, napravićemo jednu analogiju. Zamislite autoput koji na svakih 100 kilometara ima naplatnu rampu. Segmente autoputa između ovih naplatnih rampi možete da poistovetite sa linkovima, a same naplatne rampe sa ruterima. Prepostavimo da automobili putuju (odnosno, propagiraju) ovim autoputem brzinom od 100 km/h (i da, Čim napuste naplatnu rampu, trenutno dostignu ovu brzinu koju održavaju sve do sledeće naplatne rampe). Prepostavimo, dalje, da putuje karavan od 10 vozila koja su postrojena jedno za drugim u fiksnom redosledu. Svaki pojedinačni automobil bi, u ovom slučaju, mogao da se poistoveti sa bitom, a ceo karavan sa paketom. Takođe ćemo prepostaviti da svaka naplatna rampa svaki automobil opslužuje (prenosi) tačno 12 sekundi, a pošto se to dešava kasno noću, ovi automobili su jedini automobili na putu. Konačno, prepostavljamo i to da automobil koji prvi stigne do naplatne rampe, mora na ulazu da sačeka dolazak svih 9 preostalih automobila. Dakle, ceo karavan najpre mora

⁴ Propagacijen zavisi isključivo od električnih ili elektromagnetskih osobina medijuma (prim. rec.)

da se „memoriše“ da bi se tek nakon toga „prosledio“. Vreme koje je potrebno da naplatna rampa propusti ceo karavan na autoput iznosi: $(IO \text{ automobila})/(5 \text{ automobila/min}) = 2 \text{ minuta}$. Ovo vreme bi moglo da se poistoveti sa kašnjenjem usled prenosa na ruterima. Za prelazak puta između dve naplatne rampe automobilima je potrebno: $100 \text{ km}/(100 \text{ km/čas}) = 1 \text{ čas}$. Ovo vreme bi, s druge strane, moglo da se poistoveti sa kašnjenjem usled propagacije. Prema tome, vreme od trenutka kada se karavan „memoriše“ ispred naplatne rampe do trenutka kada se to isto dogodi ispred sledeće naplatne rampe predstavlja sumu „kašnjenja usled prenosa“ i „kašnjenja usled propagacije“ - u ovom primeru to iznosi 62 minuta.

Zadržimo se još malo na ovoj analogiji. Sta bi se dogodilo kada bi opsluživanje karavana na naplatnoj rampi trajalo duže od putovanja automobila između dve rampe? Primera radi, pretpostavimo da se automobili kreću brzinom od 1 000 km/h, a da naplatna rampa opslužuje jedan automobil u minuti. U tom slučaju putovanje automobila trajalo bi 6 minuta, a opsluživanje čitavog karavana 10 minuta. To znači da bi prvi nekoliko automobila stiglo do druge naplatne rampe u trenutku kada poslednji automobil napuštaju prvu rampu. Ova situacija se događa u mrežama sa komutiranjem paketa - prvi bitovi nekog paketa mogu da stignu do drugog rutera, a da poslednji bitovi istog paketa još uvek Čekaju na prenos u prethodnom.

Ukoliko sa t/obrada , d^t , $\langle \wedge \text{prenos}, d \rangle$ označimo kašnjenja usled obrade, stajanja u redu, prenosa i propagacije, onda se ukupno kašnjenje Čvora izračunava na sledeći način:

$$\wedge \text{Cvor} \wedge \text{obrada} \wedge \text{red} \wedge \text{prenos} \wedge \text{prop}$$

Pojedinačni doprinos svakog sabirka u ukupnom kašnjenu na čvoru može prilično da varira. Primera radi, sabirak $d_{p_{top}}$ može da bude beznačajan (svega nekoliko mikrosekundi) ukoliko je reč o linku koji povezuje dva rutera u istom studentskom gradu. Isto tako, ovo kašnjenje može da iznosi i stotine milisekundi, ukoliko je reč o dva rutera koji su povezani geostacionarnim satelitskim linkom i onda je ono najkrupniji sabirak u ukupnom zbiru d_{top} . Slično tome, i sabirak $d_{p_{nm}}$ takođe može da varira od sasvim beznačajnih, pa do prilično značajnih vrednosti. Njegov doprinos je obično beznačajan pri brzinama prenosa od 10 Mb/s i većim (na primer, u LAN-ovima). Međutim, kod velikih internetskih paketa koji se šalju sporim telefonskim modemskim linkovima, u pitanju su već stotine milisekundi. Kašnjenje usled obrade $\wedge \text{obrada}$ često je beznačajno ali, s druge strane, značajno utiče na maksimalnu ukupnu propusnu moć rutera, odnosno maksimalnu brzinu kojom ruter može da prosledi pakete.

1.6.2 Kašnjenje usled stajanja u redu i gubljenje paketa

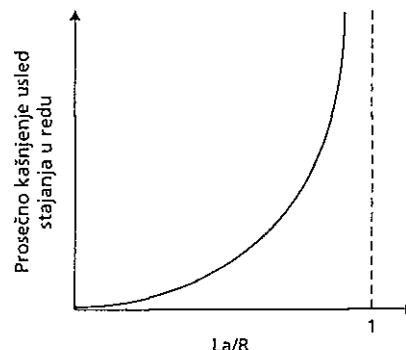
Najsloženija i najzanimljivija komponenta ukupnog kašnjenja na čvoru jeste kašnjenje usled stajanja u redu - d_{red} . Ovaj vid kašnjenja je toliko važan i zanimljiv aspekt umrežavanja, da su mu posvećeni brojni radovi i veliki broj knjiga [Bertsekas 1991; Daigle 1991; Kleinrock 1975, 1976; Ross 1995]. Mi smo vam ponudili samo intuitivno razmatranje visokog nivoa kašnjenja usled stajanja u redu; za bolji uvid u pro-

blematiku, možete da prelistate neke od knjiga koje se bave ovom temom. Za razliku od ostala tri kašnjenja ($\langle \wedge \text{obrada}, d_{p_{KTOS}}, d_{p_{top}}$), trajanje kašnjenja usled stajanja u redu razlikuje se od paketa do paketa. Na primer, ako u istom trenutku u prazan red stigne deset paketa, kod prvog prenetog paketa neće uopšte biti kašnjenja, dok će kod poslednjeg ono biti najveće (on mora da sačeka prenos prethodnih devet paketa). Otud, kada se govori o kašnjenu usled stajanja u redu, obično se koriste statistički parametri kao što su prosečna vrednost kašnjenja, varijansa kašnjenja ili verovatnoća da će kašnjenje usled stajanja u redu premašiti neku naznačenu vrednost.

Kada je kašnjenje usled stajanja u redu osetno, a kada beznačajno? Odgovor na ovo pitanje u velikoj meri zavisi od obima saobraćaja koji stiže u red, brzine prenosa linka i prirode pristiglog saobraćaja (da li on stiže periodično ili u naletima). Da bismo vam malo približili ovaj problem, uvešćemo novu promenljivu i slovom a označiti prosečnu brzinu kojom paketi stižu u red (meri se brojem paketa u sekundi). Kao i ranije, R je brzina prenosa, odnosno brzina (u bitovima u sekundi) kojom se bitovi guraju iz reda. Da bi sve bilo što jednostavnije, pretpostavćemo da se svi paketi sastoje od L bitova. U tom slučaju, prosečna brzina kojom bitovi stižu u red iznosi La bitova u sekundi. Konačno, pretpostavćemo i to da je red veoma dugačak, što znači da u njemu može da se nalazi beskonačan broj paketa. Odnos La/R koji se naziva i intenzitet saobraćaja često igra veoma važnu ulogu u proceni kašnjenja usled stajanja u redu. Ukoliko je $La/R > 1$, onda prosečna brzina kojom bitovi dospevaju u red premašuje brzinu kojom oni mogu da se prenesu iz tog reda. U ovakvoj nesrećnoj situaciji red bi se bezgranično produžavao, a vrednost kašnjenja usled stajanja u redu približila bi se beskonačnosti. Dakle, jedno od zlatnih pravila u vezi sa dimenzioniranjem saobraćaja glasi: *Dizajnirajte svoj sistem tako da vrednost intenziteta saobraćaja nikada ne bude 1*.

Sada zamislite situaciju u kojoj je $La/R < 1$. U ovom slučaju priroda pristižu-ćeg saobraćaja utiče na vrednost kašnjenja usled stajanja u redu. Na primer, ukoliko paketi stižu periodično (jedan na svakih L/R sekundi), svaki od njih će zateći prazan red i nikakvog čekanja neće biti. Ali, ukoliko paketi stižu u periodičnim naletima, prosečno kašnjenje usled čekanja u redu može biti značajno. Pretpostavimo, na primer, da broj od N paketa stiže istovremeno svakih $(L/R)N$ sekundi. Tada prvi preneti paket nema kašnjenje, drugi ima kašnjenje od L/R sekundi, a »ti paket kašnjenje od $(n - 1)L/R$ sekundi. Ostavićemo vam da, kao svojevrsno vežbanje, za ovaj primer izračunate prosečno kašnjenje usled stajanja u redu.

Za dva opisana primera o periodičnom pristizanju paketa moglo bi da se kaže da su pomalo akademski. Paketi obično u red stižu *nasumično*, bez ikakvog Šablona i u različitim vremenskim intervalima. U ovom, nešto realnijem primeru, vrednost La/R obično nije dovoljna za puno defmisanje statistike kašnjenja. Ipak, ovo intuitivno razumevanje dužine kašnjenja usled stajanja u redu može samo da vam koristi. Konkretno, ako je vrednost intenziteta saobraćaja bliska nuli, to znači da paketi stižu retko i sa dugim pauzama između njih, tako da su šanse da pristigli paket u redu zatekne neki drugi paket veoma male. Zbog toga će prosečna vrednost kašnjenja u redu biti bliska nuli. Nasuprot tome, kada je vrednost intenziteta saobraćaja bliska broju 1, to znači da postoje periodi u kojima brzina kojom paketi pristižu prevazilazi



Slika 1.14 ♦ Zavisnost prosečne vrednosti kašnjenja usled stajanja u redu od intenziteta saobraćaja

prenosni kapacitet (zbog intervala sa naletima); sto povlaci formiranje reda. Kako se vrednost intenziteta saobraćaja približava broju 1, prosečna dužina kašnjenja usled stajanja u redu postaje sve veća. Na slici 1.14 prikazana je kvalitativna zavisnost prosečnog kašnjenja usled stajanja u redu od intenziteta saobraćaja.

Veoma važan aspekt slike 1.14 predstavlja činjenica da sa približavanjem vrednosti intenziteta saobraćaja vrednosti 1, prosečna vrednost kašnjenja usled stajanja u redu veoma brzo raste. Mali procenat povećanja prvog parametra povlači procentualno mnogo veći porast drugog. Ovaj fenomen ste najverovatnije upoznali na putevima. Ukoliko se redovno vozite putem koji je često zagušen, to znači da je vrednost intenziteta saobraćaja blizu 1. Ukoliko neki dogadjaj prouzrokuje makar i neznatno povećanje saobraćaja, kašnjenja koja iz toga proističu mogu biti veoma duga.

Gubitak paketa

U prethodnom razmatranju pretpostavili smo da se u redu može nalaziti beskonačno mnogo paketa. U stvarnosti red koji prethodi izlaznom linku ima ograničen kapacitet, u zavisnosti od dizajna i cene komutatora. S obzirom na to daje kapacitet reda konačan, u stvarnosti, kašnjenje paketa se ne približava beskonačnosti sa približavanjem intenziteta saobraćaja vrednosti 1. Upravo suprotno, može se dogoditi da pristigli paket nađe na popunjenoj red. Ukoliko nema mesta za njegovo memorisanje, ruter jednostavno ispušta (odbacuje) dati paket; drugim recima, paket je izgubljen. Iz perspektive krajnjeg sistema, to izgleda kao da je paket emitovan u jezgro mreže, ali se nije pojavio na svom odredištu. Procenat izgubljenih paketa povećava se sa povećanjem intenziteta saobraćaja. Prema tome, performanse određenog čvora mere se, ne samo dužinom kašnjenja, već i verovatnoćom gubljenja paketa. Kao što ćete videti u poglavljima koja slede, izgubljeni paketi mogu da se emituju ponovo, za šta su zaduženi ili aplikacija ili protokol transportnog sloja.

Kašnjenje na putu sa kraja na kraj mreže

Sve do sada razmišljali smo samo o kašnjenju na nivou čvora, odnosno najednom ruteru. Upravo zato ćemo našu priču o različitim vidovima kašnjenja završiti kratkim razmatranjem ukupnog kašnjenja od izvora do odredišta. Da biste lakše stekli predstavu o ovom konceptu, pretpostavimo da između izvornog i odredišnog računara postoji A'-l ruter. Takođe ćemo pretpostaviti da u mreži nema zagušenja (kašnjenja usled stajanja u redu su zanemarljiva), zatim da kašnjenje usled obrade na svakom ruteru iznosi d_{obrada} , da brzina kojom svi ruteri i izvomi računar prenose informacije iznosi R bitova u sekundi i da je kašnjenje usled propagacije na svakom linku d . Dakle, akumuliranjem kašnjenja na nivou čvorova dobijemo ukupno kašnjenje od jedne do druge krajnje tačke prenosa.

\wedge kraj-kraj ~ \wedge obrada \wedge prenos \wedge pfp

I u ovom slučaju $d_{\text{Q}} = L/R$, gde L predstavlja veličinu paketa. Uopštavanje ove formule za slučaj heterogenih kašnjenja na čvorovima i za postojanje prosečnog kašnjenja usled stajanja u redu ostavićemo vama.

1.6.3 Kašnjenja i rute na Internetu

Za sticanje konkretnije slike o kašnjenju u računarskim mrežama može da posluži dijagnostički program Traceroute. Reč je o jednostavnom programu koji se može izvršavati na svakom računaru koji je deo Interneta. Kada korisnik navede ime odredišnog računara, program u izvornom računaru odmah ka njemu šalje niz posebnih paketa. Putujući ka svom odredištu ovi paketi prolaze kroz seriju ruteru. Čim primi neki od ovih posebnih paketa, ruter odmah šalje kratku poniku izvornom računaru u kojoj su najčešće navedeni njegovo ime i adresa.

Konkretno, ponovo ćemo pretpostaviti da između izvornog i odredišnog računara postoji AM ruter. Izvorni računar u mrežu šalje W posebnih paketa od kojih je svaki adresiran na konačno odredište. Ovi paketi su obeleženi brojevima od 1 do N , tako da prvi paket nosi broj 1, a poslednji N . Kada »ti ruter primi n -ti paket, on taj paket uništava! Šalje poruku izvornom računaru. Kada odredišni računar primi A^n -ti paket on ga takođe uništava i takođe šalje poruku izvornom računaru. Od trenutka kada pošalje pakete, izvorni računar meri vreme do prijema odgovarajućih poruka, beležeći i ime i adresu rutera (ili odredišnog računara) od koga je poruka stigla. Na ovaj način, izvorni računar može da rekonstruiše rutu kojom paketi putuju od izvora do odredišta, i da odredi kašnjenja na povratnim putevima do svih rutera koji učestvuju u prenosu. Program Traceroute ponavlja opisani eksperiment tri puta, tako da izvomi računar ka odredišnom, u stvari, šalje $3-N$ paketa. Program Traceroute detaljno je opisan u dokumentu RFC 1393.

Evo jednog primera rezultata programa Traceroute u kome je trasirana ruta između izvornog računara gai.cs.umass.edu (na Univerzitetu Masačusets) i odredišnog računara cis.poly.edu (na Politehničkom univerzitetu u Brukljinu). Ovaj listing

ima šest kolona. Prvu kolonu čini vrednost *rt* koju smo upravo opisali i koja predstavlja broj ruta na dатoj ruti. U drugoj koloni navedeno je ime ruter, a u trećoj njegova adresa (u formi xxx.xxx.xxx.xxx). Poslednje tri kolone predstavljaju kašnjenja povratnih puteva za tri eksperimenta. Ukoliko (usled gubitka paketa) primi manje od tri poruke od svakog ruter na putanji, program Traceroute postavlja zvezdicu odmah iza broja ruta i za dati ruter prijavljuje manje od tri vremena odziva.

```
1 cs-gw (128.119.240.254) 1.009 ms 0.899 ms 0.993 ms
2 128.119.3.151 (128.119.3.154) 0.931 ms 0.441 ms 0.651 ms
3 border4-rt-gi-l-3.gw.umass.edu (128.119.2.194) i.032 ms 0.484 ms 0.481 ms
4 acrl-ge-2-1-0.Boston.cw.net (208.172.51.129) 10.006 ms 8.150 ms 8.460 ms
5 agr4-loopback.NewYork.cw.net (206.24.194.104) 12.272 ms 14.344 ms 13.267 ms
6 acr2-loopback.NewYork.cw.net (206.24.194.62) 13.225 ms 12.292 ms 12.148 ms
7 pos10-2.core2.NewYork1.Level3.net (209.244.160.133) 12.218 ms 11.823 ms 11.793 ms
8 gige9-1-52.hsipaccess1.NewYork1.Level3.net (64.159.17.39) 13.081 ms 11.556 ms 13.297 ms
9 p0-0.polyu.bbnplanet.net (4.25.109.122) 12.116 ms 13.052 ms 12.786 ms
10 cis.poly.edu (128.238.32.126) 14.080 ms 13.035 ms 12.802 ms
```

U trasi koja je prikazana ovim listingom, postoji 9 ruter izmedu izvora i odredišta. Većina ovih ruta ima ime i svi imaju adrese. Na primer, ime ruter 3 je border4-rt-gi-l-3 . gw . umass . edu, a njegova adresa je 128.119.2.194. Gledajući podatke koje je ovaj ruter poslao vidimo da je kašnjenje na povratnom putu u prvom od tri pokušaja bilo 1,03 msec. Kašnjenje na povratnom putu za sledeća dva pokušaja iznosilo je 0,48 i 0,45 msec. Ova kašnjenja obuhvataju sva već pomenuta kašnjenja - kašnjenje usled prenosa, kašnjenje usled propagacije, kašnjenje usled obrade u ruteru i kašnjenje usled stajanja u redu. S obzirom na to da kašnjenje usled stajanja u redu varira, kašnjenje na povratnom putu za paket *n* koji je poslat ruteru *n* može biti duže od istog tog kašnjenja za paket *n+1* na ruteru *n+1*. U vezi sa tim, skrećemo vam pažnju na to da su kašnjenja u ruteru 6 veća od kašnjenja u ruteru 7. Ovo je posledica samog postupka merenja, bez obzira na činjenicu da svaki paket upućen ruteru 7 mora da prođe i kroz ruter 6.

Ukoliko želite da i sami isprobate program Traceroute, preporučujemo vam da posetite veb lokaciju <http://www.traceroute.org> na kojoj ćete pronaći iscrpnu listu izvora za trasiranje ruta. Dovoljno je da izaberete izvor i upišete ime bilo kog odredišnog računara, a program Traceroute će učiniti ostalo.

1.7 Slojevi protokola i modeli njihovih usluga

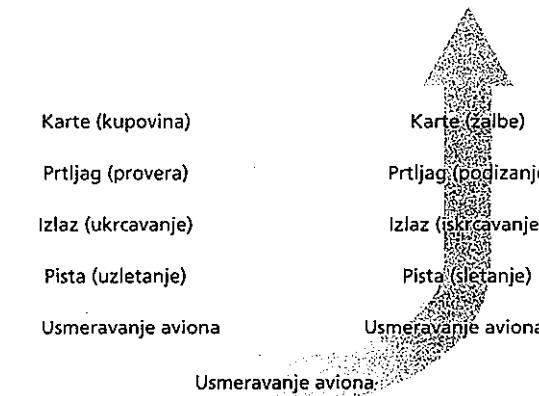
Iz dosadašnjeg izlaganja očigledno je daje Internet *Izuzetno* složen sistem. Kao što ste videli, on se sastoji iz mnoštva sastavnih delova - brojnih aplikacija i protokola, raznih tipova krajnjih sistema i veza izmedu njih, zatim ruta, kao i različitih tipova

medijuma koji se koriste kao linkovi. Imajući u vidu ovu izuzetnu složenost, postavlja se pitanje da lije uopšte moguće zamisliti njegovu mrežnu arhitekturu i da lije to moguće učiniti u ovoj knjizi. Srećom, odgovor na ova ova pitanja je potvrđan.

1.7.1 Slojevita arhitektura

Pre nego što pokušamo da zamislimo arhitekturu Interneta, napravićemo još jednu analogiju sa svetom koji nas okružuje. U stvari, mi u svakodnevnom životu praktično sve vreme imamo posla sa veoma složenim sistemima. Zamislite, primera radi, da neko od vas zatraži da opišete sistem avionskog prevoza. Kako biste osmisili strukturu koja bi mogla da opiše ovaj izuzetno složeni sistem, koji obuhvata prodaju karata, proveru prtljaga, osoblje na terminalima, pilote, same avione, kontrolu leta i svetski sistem za usmeravanje aviona? Jedan od načina za opisivanje ovog sistema bio bi da pokušate da opišete sve što treba da učinite (ili što drugi treba da učine za vas) prilikom putovanja avionom. Dakle, najpre kupujete kartu, zatim proveravate prtljag, odlazite na određeni izlaz i konačno se ukrcavate u avion. Avion uzleće i zatim se usmerava ka svom odredištu. Kada sleti, vi se iskravate, opet prolazite kroz terminal i podižete svoj prtljag. Ukoliko je let bio neprijatan, žalićete se agentu od koga ste kupili kartu. Ovaj scenario opisan je na slici 1.15.

Prepostavljamo da već uočavate neke sličnosti sa računarskim mrežama - avion vas prevozi od mesta iz koga polazite do željene destinacije; paket se prenosi od izvornog do odredišnog računara na Internetu. Ipak ovo nije baš ona analogija koju tražimo. Nama je potrebna određena *struktura* i nju ćemo potražiti na slici 1.15. Skrećemo vam pažnju na to da na oba kraja linije postoji funkcija u vezi sa kartama; osim toga postoji i funkcija za prtljag putnika koji su kupili karte, kao i funkcija



Slika 1.15 ♦ Putovanje avionom: akcije

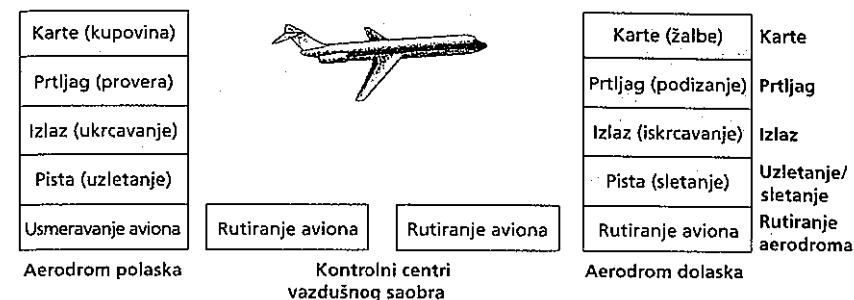
izlaza za putnike sa kupljennom kartom i predatim prtljagom. Za putnike koji su prošli kroz izlaz (kupili su kartu, predali prtljag i prošli kroz izlaz) ostale su još funkcije uzletanja i sletanja, a tokom trajanja leta i funkcija usmeravanja aviona. Iz ovoga sledi da funkcije sa slike 1.15 mogu da se posmatraju u smislu *horizontalne povezanosti*, kao što je to i prikazano na slici 1.16.

Na slici 1.16, funkcije avionskog prevoza podelili smo na slojeve i na taj način dobili globalni okvir unutar kojeg možemo razgovarati o avionskom prevozu. Skrećemo vam pažnju na to da svaki sloj, u kombinaciji sa slojevima koji se nalaze ispod njega, obezbeđuje neku funkcionalnost, odnosno neku *uslugu*. Na sloju prodaje karata i nižim slojevima, obavlja se transfer putnika od aerodroma do aerodroma. Na sloju koji se bavi prtljagom i nižim slojevima, odvija se transfer putnika i njihovog prtljaga, od provere do preuzimanja prtljaga. Skrećemo vam pažnju na činjenicu da sloj zadužen za prtljag svoje usluge pruža samo putnicima koji su već kupili kartu. Na sloju izlaza, obavlja se transfer putnika i njihovog prtljaga, od odlaznog terminala do dolaznog terminala, dok se na sloju uzletanja i sletanja obavlja transfer putnika i njihovog prtljaga od jedne piste do druge. Svaki sloj obezbeđuje svoje usluge (1) izvršavanjem određenih akcija u okviru tog sloja (primera radi, na sloju terminala vrši se ukrcavanje ljudi u avion i iskrcavanje iz njega) i (2) korišćenjem usluga koje se nalaze u sloju ispod njega (sloj terminala koristi uslugu transfera putnika od piste do piste koji pripada sloju uzletanja i sletanja).

Kao što smo već rekli, slojevita arhitektura omogućava izdvajanje pojedinih dobro defmisišanih segmenta iz velikog i složenog sistema. Pojednostavljenje koje se na taj način postiže sasvim je konkretno i značajno. Osim toga, kada sistem ima slojevitu strukturu mnogo je lakše izmeniti implementaciju usluga koje određeni sloj obezbeđuje. Sve dok neki sloj obezbeđuje usluge koje pruža i sloj iznad njega i koristi usluge iz sloja koji je ispod njega, izmena njegove implementacije ne odražava se na ostatak sistema. (Skrećemo vam pažnju na to da se izmena implementacije usluge značajno razlikuje od izmene same usluge.) Ukoliko, primera radi, izmenimo način rada terminala (recimo, tako da se ljudi ukrcavaju i iskrcavaju po visini), to se neće odraziti na ostatak sistema avionskog prevoza zato što izlaz i dalje pruža neizmenjenu funkciju (ukrcavanje i iskrcavanje putnika); ona se samo implementira malo drugačije. U velikim i složenim sistemima koji se neprekidno ažuriraju, slojevitost donosi još jednu prednost, a to je mogućnost izmene implementacije usluge bezposledica po ostale komponente sistema.

Slojevitost protokola

Pošto smo se dovoljno pozabavili avionskim prevozom, okrenimo se sada mrežnim protokolima. U cilju smanjenja složenosti dizajna, mrežni dizajneri organizuju protokole - ali i mrežni hardver i softver koji te protokole implementira - u slojeve. U slojevitoj arhitekturi protokola svaki protokol pripada jednom sloju, isto kao što svaka funkcija avionskog prevoza prikazanog na slici 1.16 takođe pripada nekom



Slika 1.16 ♦ Horizontalna slojevitost funkcija avionskog transporta

sloju. I u ovom slučaju zanimaju nas usluge koje određeni sloj pruža slojevima koji se nalaze iznad njega - takozvani uslužni model sloja. Baš kao i u primeru sa avionskim prevozom, svaki sloj pruža svoje usluge (1) izvršavanjem određenih radnji i (2) korišćenjem usluga sloja koji se nalazi direktno ispod njega. Na primer, pod uslugom koju obezbeđuje sloj *n* mogla bi da se podrazumeva pouzdana dostava poruka sa jednog kraja mreže na drugi. Ova usluga mogla bi da se implementira korišćenjem usluge za nepouzdanu dostavu poruka sa kraja na kraj sloja »*1* i dodavanjem funkcionalnosti za prepoznavanje i ponovno emitovanje izgubljenih poruka sloju *n*.

Sloj protokola može da se implementira u softveru, hardveru ili kombinaciji ova dva okruženja. Protokoli aplikacijskog sloja kao što su, na primer, HTTP i SMTP gotovo uvek su implementirani u softveru krajnjih sistema; isti je slučaj i sa protokolima transportnog sloja. S obzirom na to da su fizički sloj i sloj veze podataka odgovorni za rukovanje komunikacijom preko konkretnog linka, oni se obično implementiraju u kartici mrežnog interfejsa (na primer, kartice Ethernet ili Wi-Fi) koja je povezana sa datim linkom. Mrežni sloj obično se implementira kombinovano - i u hardveru i u softveru. Skrećemo vam pažnju i na to da se, poput funkcija slojevitog avionskog prevoza koje su distribuirane između raznih aerodroma i centara za kontrolu leta od kojih se sastoji ceo ovaj sistem, protokol sloja *n* takođe *distribuira* između krajnjih sistema, komutatora paketa i ostalih komponenti koje čine mrežu. To znači da se u svakoj od ovih mrežnih komponenti Često nalazi i deo protokola sloja *n*.

Kada se posmatraju u celini, protokoli svih slojeva zajedno nazivaju se familija protokola. Familija Internet protokola sastoji se od pet slojeva i to su: fizički sloj, sloj veze podataka, mrežni sloj, transportni sloj i aplikacijski sloj (slika 1.17).

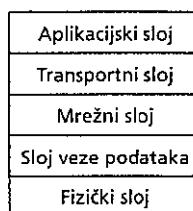
Aplikacijski sloj

U aplikacijskom sloju nalaze se mrežne aplikacije i njihovi protokoli aplikacijskog sloja. U Internetovom aplikacijskom sloju ima mnogo različitih protokola kao što su protokoli HTTP (podrška za zahtevanje i transfer veb strana), SMTP (podrška za transfer elektronske pošle) i FTP (podrška transfera datoteka između dva krajnja sistema). Pored toga, videćete i da se određene mrežne funkcije kao stope, na primer, prevođenje ljudima razumljivih Internet imena krajnjih sistema (recimo, gaia.cs.umass.edu) u 32-bitne mrežne adrese, takođe izvršavaju pomoću protokola aplikacijskog sloja, DNS-a (sistemska imena domena). U poglavlju 2 videćete daje veoma lako napraviti i sопstvene nove protokole aplikacijskog sloja.

Podsećamo vas na definiciju protokola iz odeljka 1.1, kada smo rekli da distribuirani entiteti koji implementiraju neki protokol između sebe razmenjuju poruke. U ovoj knjizi ćemo te vrste poruka nazivati porukama aplikacijskog sloja.

Transportni sloj

Transportni sloj obezbeđuje usluge transporta poruka aplikacijskog sloja između klijentske i serverske strane aplikacije. Na Internetu postoje dva transportna protokola - TCP i UDP i oba mogu da transportuju poruke aplikacijskog sloja. Protokol TCP svojim aplikacijama obezbeđuje uslugu sa konekcijom. Ova usluga podrazumeva garantovanu isporuku poruka aplikacijskog sloja do odredišta i kontrolu toka (usa-glašavanje brzina pošiljaoca i primaoca). Pored toga, protokol TCP segmentira duge poruke na kraće segmente i obezbeđuje mehanizme za kontrolu zagušenja tako što izvor smanjuje brzinu prenosa u periodima zagušenja mreže. Nasuprot tome, protokol UDP svojim aplikacijama obezbeđuje uslugu bez konekcije koja je (kao što ste videli u odeljku 1.2), u velikoj meri, usluga bez suvišnih detalja. Pakete transportnog sloja ćemo u ovoj knjizi nazivati segmentima.



Slika 1.17 ♦ Familija Internet protokola

Mrežni sloj

Mrežni sloj odgovoran je za rutiranje paketa mrežnog sloja - *datagrama* od jednog računara do drugog. Internetski protokol transportnog sloja (TCP ili UDP) izvornog računara prosledjuje segment transportnog sloja i odredišnu adresu mrežnom sloju, na isti način kao što mi poštanskoj službi predajemo pismo sa njegovom odredišnom adresom. Nakon toga, o isporuci segmenta transportnom sloju odredišnog računara brine mrežni sloj.

Internetski mrežni sloj ima dve osnovne komponente. Prva komponenta je protokol koji definiše polja u datagramu, kao i način reagovanja krajnjih sistema i ruta na sadržaj ovih polja. Reč je o čuvenom protokolu IP. Postoji samo jedan protokol IP i sve internetske komponente koje imaju mrežni sloj moraju da ga koriste. Mrežni sloj takođe sadrži protokole za rutiranje koji određuju rute kojima se datagrami kreću od izvora do odredišta. Na Internetu ima mnogo protokola za rutiranje. Kao što smo rekli u odeljku 1.5, Internet predstavlja mrežu svih mreža, a u okviru svoje mreže administrator ima slobodu korišćenja bilo kog protokola za rutiranje. Iako se u mrežnom sloju nalaze i protokol IP i brojni protokoli za rutiranje, on se često naziva samo IP sloj, što govori o činjenici daje protokol IP komponenta koja ceo Internet drži na okupu.

Sloj veze podataka

Mrežni sloj rutira datagram kroz seriju komutatora paketa (na Internetu se nazivaju *ruterima*) koji se nalaze između izvora i odredišta. Da bi mogao da prenese paket od jednog čvora (računara ili komutatora paketa) do prvog sledećeg, mrežni sloj mora da se osloni na uslugu sloja veze podataka. Konkretno, na svakom čvoru mrežni sloj predaje datagram naniže sloju veze podataka, koji treba da ga isporuči sledećem Čvoru na putanji (ruti). Na tom sledećem Čvoru sloj veze podataka predaje datagram naviše, mrežnom sloju.

Usluge koje obezbeđuje sloj veze podataka zavise od konkretnog protokola veze podataka koji je primjenjen na datom linku. Primera radi, neki protokoli obezbeđuju pouzdanu isporuku na nivou linka, odnosno od emitujućeg Čvora, kroz link, do prijemnog čvora. Skrećemo vam pažnju na to da se ova usluga pouzdane isporuke razlikuje od jednog krajnjeg sistema do drugog. U primere sloja veze podataka spadaju Ethernet i PPP. S obzirom na to da datagrami na svom putu od izvornog do odredišnog računara moraju da prođu kroz nekoliko linkova, njima u različitim linkovima mogu da rukuju različiti protokoli sloja veze podataka. Recimo, u jednom linku datagramom može da rukuje Ethernet, a već u sledećem protokol PPP. Mrežni sloj od svakog protokola sloja veze podataka može da dobije različite usluge. U ovoj knjizi pakete sloja veze podataka nazivaćemo **okvirima**.

Fizički sloj

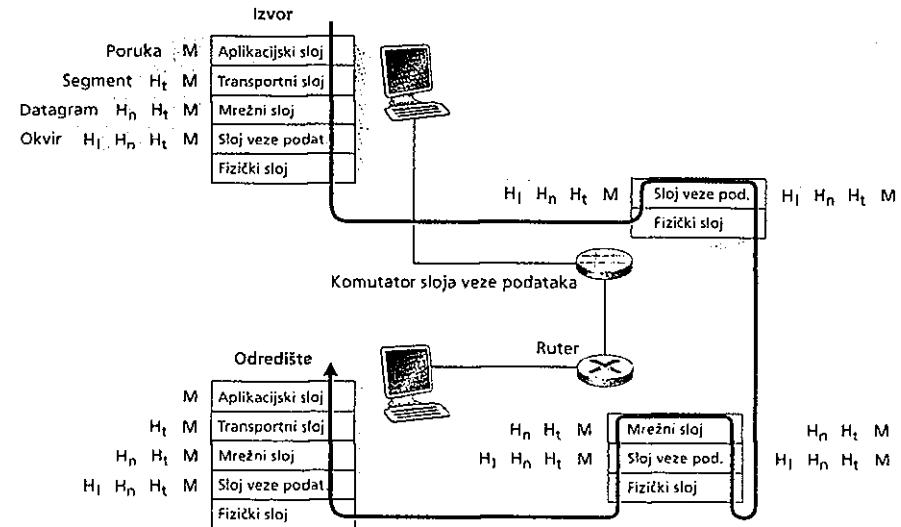
Dok je sloj veze podataka zadužen za prenošenje celih okvira od jednog mrežnog elementa do drugog, „posao“ fizičkog sloja jeste prenošenje *pojedinačnih bitova* istih tih okvira između susednih čvorova. Protokoli ovog sloja zavise od sloja veze podataka, ali i od samog prenosnog medijuma linka (bakarni kablovi sa upredenim paricama, monomodna optička vlakna). Primera radi, standard Ethernet ima mnogo protokola fizičkog sloja - jedan za bakarne kablove sa upredenim paricama, drugi za koaksijalne kablove, treći za optičke kablove, itd. U svakom konkretnom slučaju pojedinačni bit se prenosi kroz link na drugačiji način.

Ukoliko prelistate sadržaj, uočićete da je ova knjiga organizovana upravo korišćenjem slojeva familije Internet protokola. Pri tom smo se opredelili za **pristup od vrha ka dnu**, odnosno od aplikacijskog sloja naniže.

1.7.2 Slojevi, poruke, segmenti, datagrami i okviri

Na slici 1.18 prikazana je stvarna fizička putanja kojom se podaci kreću naniže kroz familiju protokola prednjog sistema, naviše i naniže kroz familije protokola posredujućeg komutatora sloja veze podataka ili rutera i zatim naviše kroz familiju protokola prijemnog sistema. Kao što ćete videti u nastavku knjige, i ruteri i komutatori sloja veze podataka pripadaju komutatorima paketa. Poput krajnjih sistema, i kod njih je hardver i softver za umrežavanje slojevitovo organizovan. Međutim, kod rutera i komutatora sloja veze ne implementiraju se *svi* slojevi familije protokola; obično se na njima implementiraju samo donji slojevi. Kao što možete da vidite na slici 1.18, komutatori sloja veze implementiraju slojeve 1 i 2, a ruteri slojeve od 1 do 3. To, primera radi, znači da internetski ruteri mogu da implementiraju protokol IP (protokol trećeg sloja), a da komutatori sloja veze podataka to ne mogu. Nešto kasnije videćete da, iako ne prepoznaju IP adresu, komutatori sloja veze mogu da prepoznaju adresu drugog sloja kao što su, recimo, Ethernemet adresе. Podvlačimo činjenicu da krajnji sistemi implementiraju svih pet slojeva, što potvrđuje konstataciju daje periferija Interneta previše složena.

Na slici 1.18 ilustrovan je i veoma važan koncept **enkapsulacije**. U prednjom računaru **poruka aplikacijskog sloja** (M na slici 1.18) predaje se transportnom sloju. U najjednostavnijem scenariju transportni sloj preuzima poruku i dodaje joj dopunske informacije (informacije zaglavljaju transportnog sloja; H_t na slici 1.18) koje su namenjene prijemnoj strani transportnog sloja. Poruka aplikacijskog sloja i zaglavljje transportnog sloja zajedno Čine **segment transportnog sloja**. Segment transportnog sloja, prema tome, enkapsulira poruku aplikacijskog sloja. Među dodatnim informacijama mogu se nalaziti informacije na osnovu kojih prijemna strana transportnog sloja isporučuje ovu poruku odgovarajućoj aplikaciji, i mogu se nalaziti bitovi za prepoznavanje grešaka koji prijemnoj strani omogućavaju da ustanovi da li su bitovi poruke usput promenjeni. Transportni sloj zatim prosleduje segment mrežnom sloju, koji ovom segmentu dodaje svoje zaglavlje (H_n na slici 1.18) sa podacima kao što su, na primer, adrese izvornog i odredišnog sistema, formirajući



Slika 1.18 ♦ Računari, ruteri i komutatori sloja veze podataka; svaki sadrži različit skup slojeva što je posledica njihove različite funkcionalnosti.

na taj način **datagram mrežnog sloja**. Ovaj datagram se zatim prosleduje sloju veze podataka koji mu sada dodaje svoje zaglavlje i pravi **okvir sloja veze podataka**.

Ovaj koncept ćemo vam dodatno ilustrovati korišćenjem analogije sa internim dopisom u firmi koji se šalje javnom poštanskom službom. Sam dopis predstavlja poruku aplikacijskog sloja. On se stavlja u odgovarajući koverat na čijoj su prednjoj strani ispisani ime i odeljenje primaoca. Ova informacija služi arhivi u zgradu prijemne kancelarije da primljeni dopis isporuči onome kome je poslat. Koverat za interni dopis koji sadrži zaglavlje (ime i odeljenje primaoca) i enkapsulira poruku aplikacijskog sloja (dopis), mogao bi da se poistoveti sa segmentom transportnog sloja. Arhiva predajne sirane uzima ovaj dopis i stavlja ga u koverat koji je podesan za prenos kroz javnu poštansku službu, upisuje poštansku adresu predajne i prijemne strane i dodaje marku. Ovaj poštanski koverat odgovara datagramu - on enkapsulira segment transportnog sloja (koverat za slanje dopisa i sam sadržaj) koji enkapsulira originalnu poruku (dopis). Arhiva predajne strane isporučuje koverat arhivi prijemne strane koja otvara koverat i iz njega vadi koverat za interni dopis, koji se, zatim, prosleduje primaocu koji otvara ovaj drugi koverat i iz njega vadi sam dopis.

Proces enkapsuliranja može biti i složeniji od ovog koji smo upravo opisali. Primera radi, neka velika poruka može da se podeli na više segmenata transportnog

sloja (koji dalje mogu da se podele na više datagrama mrežnog sloja). Na prijemnoj strani, takav segment bi onda morao da se rekonstruiše od datagrama na koje je razbijen.

1.8 Istorijski pregled umrežavanja i Interneta

U odeljcima od 1.1 do 1.7 napravili smo prikaz tehnologija računarskih mreža i Interneta. Iz njega ste mogli da naučite dovoljno da biste impresionirali porodicu i prijatelje. Međutim, ukoliko želite da zaista zadivite nekoga svojim znanjem, pročitajte pažljivo deo koji sledi, u kome smo pokušali da prikažemo fascinantu istoriju Interneta.

1.8.1 Razvoj komutiranja paketa: 1961 - 1972

Koreni računarskih mreža i današnjeg Interneta mogu se pratiti sve do početka 1960-ih kada je telefonska mreža bila dominantna svetska komunikaciona mreža. Podsećamo vas da smo u odeljku 1.3 rekli da se u telefonskim mrežama za prenos informacija od pošiljaoca do primaoca koristi komutiranje vodova - što predstavlja adekvatan izbor s obzirom na to da se glas od jedne do druge krajnje tačke prenosi konstantnom brzinom. Imajući u vidu sve veći značaj (i cenu) računara u to vreme, kao i pronalazak računara sa deobom vremena, bilo je sasvim prirodno (ili bar to tako izgleda iz današnje perspektive) razmišljati o načinima njihovog povezivanja kako bi mogli da ih koriste korisnici na različitim geografskim lokacijama. Saobraćaj koji bi generisali takvi korisnici imao bi karakteristike „naleta“ - intervali aktivnosti kao stoje, na primer, slanje komande udaljenom računaru, bili bi praćeni intervalima neaktivnosti u Čekanju na odgovor ili razmišljanju o primljenom odgovoru.

Tri istraživačke grupe u različitim delovima sveta, potpuno nezavisno jedna od druge [Leiner 1998], došle su do tehnologije komutiranja paketa kao efikasne alternative za komutiranje vodova. Prvi objavljen rad o tehnikama komutiranja paketa bio je rad Leonarda Klajnroka [Kleinrock 1961; Kleinrock 1964], u to vreme studenta postdiplomca na MIT-u. Koristeći teoriju stvaranja redova, Klajnrok je u svom radu elegantno demonstrirao efikasnost pristupa sa komutiranjem paketa i njegovu primerenost situacijama sa naletima saobraćaja. Godine 1964. Pol Baran[Baran 1964] iz Instituta Rand započeo je istraživanje o korišćenju komutiranja paketa za bezbedan prenos glasa u vojnim mrežama. Istovremeno, Donald Dejvis i Rodžer Skantberi iz Nacionalne fizičke laboratorije u Velikoj Britaniji razvijali su vlastite ideje o komutiranju paketa.

Istraživačkim radom ove tri grupe postavljeni su temelji današnjeg Interneta. Ali, Internet ima i dugi istoriju praktičnog pristupa Čiji počeci takođe sežu u rane 1960. godine. J. C. R. Liklider [DEC 1990] i Lorens Roberts, inače obojica Klaj-

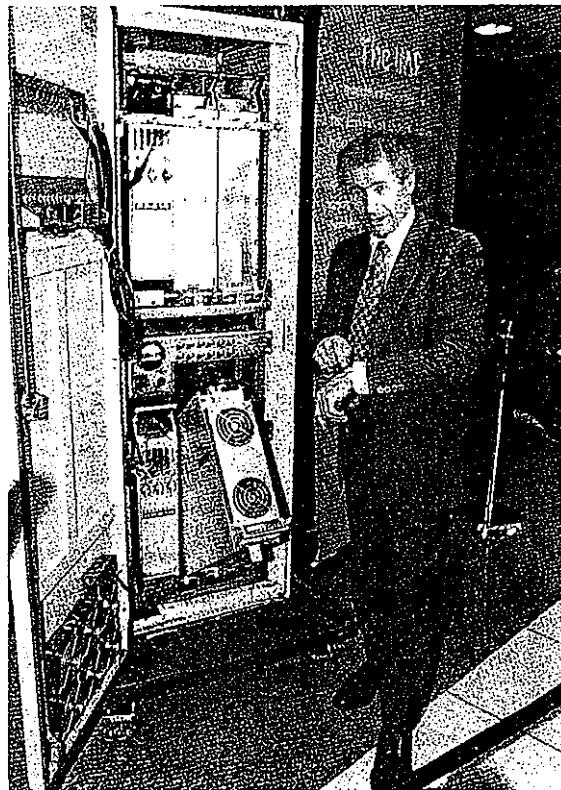
nrokovke kolege na MIT-u, pokrenuli su istraživački program u okviru Agencije za napredne istraživačke projekte (*Advanced Research Project Agency*, ARPA). Roberts je objavio sveukupni plan za tzv. ARPAnet [Roberts 1967], prvu računarsku mrežu sa komutiranjem paketa, koja je direktni predak savremenog Interneta. Prvi komutatori paketa bili su poznati pod nazivom **procesori interfejsnih poruka** (*Interface Message Processor*, IMP), a ugovor za pravljenje ovih komutatora dobila je kompanija BBN. Za Praznik rada 1969. godine, pod supervizijom Leonarda Klajnroka, na Univerzitetu UCLA, instaliranje prvi IMP komutator. Nedugo zatim, instalirana su još tri IMP komutatora - u Istraživačkom institutu Stanford (SRI), na Univerzitetu Santa Barbara i na Univerzitetu Juta (slika 1.19). Ovaj predak Interneta je tako krajem 1969. godine imao ukupno četiri čvora. Prema Klajnrokovim recima, prva upotreba mreže u cilju daljinskog prijavljivanja iz Univerziteta UCLA na Institutu SRJ doveo je do pada sistema [Kleinrock 2004].

Do 1972. godine mreža ARPAnet se proširila na otprilike 15 čvorova i imala prvu javnu demonstraciju koju je izveo Robert Kan na Međunarodnoj konferenciji o računarskim komunikacijama (*International Conference on Computer Communication*). Tada je završen i prvi protokol od računara do računara za komunikaciju između krajnjih sistema ARPAneta [RFC 001], po imenu NCP (*Network Control Protocol*). Sa uspostavljanjem protokola koji je mogao da spoji krajnje korisnike, moglo se otpočeti sa pisanjem aplikacija, tako da je Rej Tomlinson iz kompanije BBN prvi program za elektronsku poštu napisao već 1972. godine.

1.8.2 Specijalne mreže i međusobno povezivanje mreža: . 1972- 1980

Inicijalni ARPAnet predstavlja je zaokruženu i zatvorenu mrežu. Da bi mogao da komunicira sa ARPAnet računaram, korisnik je morao da se poveže sa IMP komutatorom ove mreže. U ranim i srednjim 1970-im pojavile su se i dodatne mreže sa komutiranjem paketa:

- ♦ ALOHAnet - mikrotalasna mreža koja je povezivala univerzitete na Havajskim ostrvima [Abramson 1970], kao i paketne satelitske [RFC 829] i paketne radio mreže [Kahn 1978] agencije DARPA.
- ♦ Telenet - komercijalna mreža sa komutiranjem paketa kompanije BBN koja se zasnivala na ARPAnet tehnologiji.
- ♦ Cyclades - francuska mreža sa komutiranjem paketa koju je osmislio Luj Puzan [Think 2002].
- ♦ Mreže sa deobom vremena kao što su, recimo, Tvmnet ili mreža kompanije GE Information Services kasnih 1960-ih i ranih 1970-ih [Schwartz 1977].
- ♦ IBM-ova mreža SNA (1969-1974), takođe uradena po uzoru na ARPAnet [Schvartz 1977].



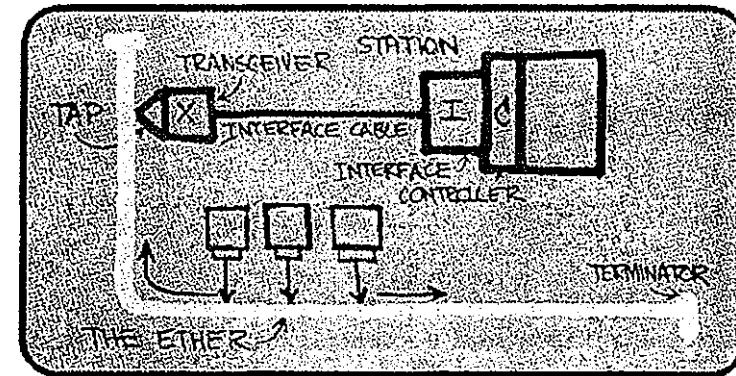
Slika 1.19 4 Prvi procesor interfejsnih poruka (IMP) i L. Klajrtrok (Mork j. Teril, AP/Wide World Photos)

Broj mreža nezadrživo se povećavao. Opet iz perspektive sadašnjeg iskustva, moglo bi se reći daje upravo tada sazreo trenutak za razvijanje arhitekture koja bi obuhvatila sve ove mreže. Pionirski rad na temu medusobnog povezivanja mreža, opet pod sponzorstvom agencije DARPA (*Defense Advanced Research Projects Agency*), obavili su Vinton Cerf i Robert Kahn [Cerf 1974]; za opisivanje povezivanja mreža upotrebljen je termin *internetting*.

Ovi arhitektonski principi ugradeni su u protokol TCP. Međutim, prve verzije protokola TCP bitno su se razlikovale od današnje verzije ovog protokola. U tim prvim verzijama, pouzdana, sekvensijalna isporuka podataka putem ponovnog pre-

nosa krajnjeg sistema (i dalje je deo protokola TCP) kombinovana je sa funkcijama za prosledivanje (što danas čini protokol IP). Prvi eksperimenti sa protokolom TCP, kao i sagledavanje značaja koji za aplikacije ima transportnu uslugu nepouzdanog prenosa s kraja na kraj bez kontrole toka (kao što je slučaj sa paketiranjem glasa), doveli su do izdvajanja protokola IP iz protokola TCP, kao i do pojave protokola UDP. Tri ključna protokola savremenog Interneta - TCP, UDP i IP - konceptualno su definisana krajem 1970-ih.

Osim istraživanja koje je sprovodila DARPA, odvijale su se i mnoge druge važne aktivnosti u vezi sa mrežama. Na Havajima, Norman Abramson razvijao je ALOHAnet-radio-mrežu zasnovanu na paketima, koja je omogućavala medusobnu komunikaciju udaljenih lokacija na Havajskim ostrvima. Protokol ALOHA [Abramson 1970] bio je prvi protokol za tzv. višestruki pristup koji je omogućio da korisnici sa različitih lokacija zajednički koriste isti medijum za komunikaciju difuznom emisijom (radio-frekvencija). Abramsonov rad na protokolu za višestruki pristup nastavili su Metcalf i Boggs, i razvili protokol Ethernet [Metcalfe 1976] namenjen kablovskim mrežama sa zajedničkom difuznom emisijom (slika 1.20). Veoma zanimljivo jeste to da je pravljenje ovog protokola bilo motivisano potrebom povezivanja većeg broja računara, štampača i zajedničkih diskova [Perkins 1994]. Dakle, pre više od 25 godina, znatno pre revolucije personalnih računara i eksplozije mreža, Metcalf i Biggs postavili su temelje današnjih LAN-ova. Tehnologijom Ethernet napravljen je veoma značajan korak i u smislu povezivanja mreža. Svaki Ethernet LAN predstavljao je zaokruženu mrežu, a kako je broj ovakvih mreža rastao, rasla je i potreba za njihovim medusobnim povezivanjem. U poglavlju 5 detaljnije ćemo se baviti tehnologijama Ethernet, ALOHA i mnogim drugim LAN tehnologijama.



Slika 1.26 ♦ Originalna koncepcija Etherjeta koju je zamislio Robert Metcalf

1.8.3 Veliki porast broja mreža: 1980 - 1990

Do kraja 1970-ih sa ARPAnetom je bilo povezano oko 200 matičnih računara. Svega desetak godina kasnije, krajem 1980-ih, broj računara povezanih sa javnim Internetom - konfederacijom mreža u kojoj se nazirao današnji Internet - dostigao je 100 000, Dakle, 1980. označavaju period intenzivnog razvoja pojave umrežavanja.

Dobar deo te ekspanzije umrežavanja početkom ove decenije proistekao je iz odvojenih nastojanja da se oforme računarske mreže koje bi povezivale univerzitete. Tako je mreža BITNET omogućila razmenu elektronske pošte i transfer datoteka između univerziteta u severoistočnom delu SAD. Mreža CSNET (*Computer Science Network*) napravljena je sa ciljem povezivanja univerzitetskih istraživača koji nisu imali pristup ARPAnetu. Godine 1986. napravljena je i mreža NSFNET koja je omogućavala pristup superračunarskim centrima pod sponzorstvom kompanije NSF. Okosnica ove mreže, čija je brzina na početku iznosila 56 kb/s, a krajem decenije već 1,5 Mb/s, poslužila je kao okosnica za povezivanje većeg broja regionalnih mreža.

Istovremeno, u ARPAnet zajednici mnogi elementi arhitekture današnjeg Interneta su lagano zauzimali svoje mesto. Prvog januara 1983. godine zvanično je pušten u rad protokol TCP/IP koji je time postao standardni protokol ove mreže (zamenivši protokol NCP). Prelazak [RFC 801] sa protokola NCP na protokol TCP/IP predstavlja je svojevrstan „štafetni“ događaj - tog dana svi računari ove mreže morali su da se prebace na protokol TCP/IP. Krajem 1980-ih načinjena su proširenja ovog protokola kojima su u njega implementirani mehanizmi kontrole zagruženja na strani računara [Jacobson 1988]. Veoma brzo razvijen je i sistem imena domena (*Domain Name System*, DNS) koji je omogućio preslikavanje između ljudima razumljivih Internet imena i odgovarajućih 32-bitnih IP adresa [RFC 1034].

Uporedo sa razvojem ARPAneta (koji je predstavljao glavninu napora ove vrste u SAD), početkom 1980-ih u Francuskoj je predstavljen projekat Minitel - veoma ambiciozan plan uvođenja mreža u sve domove. Sistem Minitel, koji je sponzorisala francuska vlada, sastojao se od javne mreže sa komutiranjem paketa (zasnovane na familiji protokola X.25 koja koristi virtuelna kola), Minitel servera i jeftinih terminala sa ugrađenim modemima male brzine. Mreža Minitel je zabeležila veliki uspeh 1984. godine, kada je francuska vlada besplatno delila Minitel terminale svim porodicama koje su izrazile želju da ih poseduju. U ovoj mreži postojale su besplatne (recimo, lokacija sa telefonskim imenikom) i privatne lokacije čije su korišćenje korisnici morali da plate. Sredinom 1990-ih, na vrhu svoje popularnosti, mreža Minitel je nudila više od 20 000 različitih usluga - od kućnog bankarstva, pa do specija-lizovanih istraživačkih baza podataka. Ovu mrežu koristilo je više od 20 procenata populacije Francuske; ona je generisala više od milijarde dolara prihoda godišnje i 10 000 radnih mesta. Moglo bi se reći da se Minitel našao u mnogim francuskim domovima deset godina pre nego što je većina Amerikanaca čula za Internet.

1.8.4 Eksplozija Interneta: poslednja decenija prošlog veka

Ulazak u 1990. godine obeležen je nizom događaja koji su simbolizovali kontinuiranu evoluciju sve prisutnije komercijalizacije Interneta. Mreža ARPAnet, praktično direktni predak Interneta, prestala je da postoji. Mreže MILNET i Defense Data

Network toliko su se razvile tokom 1980-ih da su mogle da prihvate najveći deo saobraćaja američkog Ministarstva odbrane, dok je mreža NSFNET dobila ulogu okosnice koja je povezivala regionalne mreže u SAD i nacionalne prekočne-anske mreže. Godine 1991. prestala su da važe ograničenja u smislu komercijalnog korišćenja NSFNET-a. Godine 1995. mreža je prešla u ruke komercijalnih posrednika za Internet usluge, koji su tada počeli da upravljaju saobraćajem kroz ovu internetsku okosnicu.

Ipak, glavni događaj 1990-ih predstavlja je pojava World Wide Weba koji je doneo Internet u domove i firme miliona ljudi širom sveta. Web je poslužio i kao platforma koja je omogućila puštanje u rad stotina novih aplikacija kao što su, na primer, onlajn trgovina akcijama i bankarstvo, protok multimedijalnih usluga u realnom vremenu ili informacione usluge. Kratku istoriju početaka Weba možete potražiti u knjizi [W3C 1995].

Sam Web je između 1989. i 1991. godine u istraživačkom centru CERN izmislio Tim Berners-Li [Berners-Lee 1989] i zasnovao ga na idejama koje su proistekle iz prvih radova o hipertekstu: Buša iz 1940-ih [Bush 1945] i Teda Nelsona iz 1960-ih [Ziff-Davis 1998]. Berners-Li i njegovi saradnici su razvili prve verzije HTML-a, HTTP-a, servera i čitača Weba - inače njegove četiri ključne komponente. Prvobitni CERN-ovi čitači imali su samo linijski interfejs. Krajem 1992. godine u funkciji je bilo već oko 200 servera i to je bio samo vrh ledenog brega onoga što je, na tom polju, tek trebalo da se dogodi. U tom trenutku je već veliki broj istraživača razvijao čitače sa GUI interfejsima. Među njima bio je i Mark Andrisen koji je predvodio programiranje popularnog GUI čitača po imenu Mosaic. Andrisen i njegovi saradnici su 1993. godine objavili alfa verziju svog čitača, a godine 1994. on i Džim Klark su osnovali kompaniju Mosaic Communications, iz koje je kasnije nastala kompanija Netscape Communications Corporation [Cusumano 1998; Cjuitner 1998]. Već 1995. godine studenti su koristili čitače Mosaic i Netscape za svakodnevno krstarenje Webom. Otprilike u to vreme, male i velike kompanije su počele da koriste veb servere i onlajn trgovinu. Godine 1996. kompanija Microsoft je počela da pravi svoje čitače, što je označilo početak rata između kompanija Netscape i Microsoft koji je, nekoliko godina kasnije, kompanija Microsoft okončala u svoju korist [Cusumano 1998].

Druga polovina 1990-ih predstavlja je period izuzetne ekspanzije Interneta, kao i mnogobrojnih inovacija za koje su zaslужne velike korporacije, ali i hiljade malih, koje su pravile proizvode i usluge za Internet. Nastavljena je evolucija elektronske pošte dodavanjem adresara, mogućnosti prilaganja dokumenata, brzih veza i transporta multimedijalnih sadržaja u programe za Čitanje elektronske pošte. Krajem prošlog veka Internet je podržavao stotine popularnih aplikacija, među kojima i *četiri enormno popularne*.

- ◆ elektronsku poštu, uključujući prilaganje dokumenata i pristup pošti na Webu;
- ◆ Web, uključujući pretraživanje Weba i elektronskutrgovinu;
- ◆ trenutnu razmenu poruka sa „kontakt listama“ koju je razvio ICQ i
- ◆ razmenu MP3 datoteka između ravnopravnih korisnika koju je razvio Napster.

Veoma zanimljivo je i to da su prve dve aplikacije razvili istraživački timovi, dok su druge dve delo nekolicine mladih entuzijasta.

Za period između 1995. i 2001. karakteristični su usponi i padovi Interneta u smislu finansijskih tržišta. I pre nego što su postale komercijalne kompanije, stotine malih internetskih preduzeća počelo je da se kotira na berzi. Vrednost mnogih kompanija je procenjena na milijarde dolara bez ikakvih značajnih izvora prihoda. Nagli pad internetskih akcija dogodio se 2000-2001. kada je mnogo preduzeća zatvoreno. S druge strane, kompanije kao što su Microsoft, Cisco, AOL, Yahoo, e-Bay i Ama-zon su ojačale u ovim dogadjajima i iz njih iizašle kao pobednici (i pored toga što su vrednosti njihovih akcija takođe pale).

Tokom 1990-ih u istraživanju i razvoju mreža postignut je značajan napredak na poljima ruta i rutiranja velike brzine (poglavlje 4) i lokalnih računarskih mreža (poglavlje 5). Tehnička zajednica je imala problema sa definisanjem i implementiranjem modela internetskih usluga za saobraćaj za koji je nužno odvijanje u realnom vremenu, kao što su *streaming* aplikacije (poglavlje 7). Potreba za bezbednošću i upravljanjem Internetovom infrastrukturom (poglavlja 8 i 9) takođe je izbila u prvi plan zato što su aplikacije za elektronsku trgovinu doživele veliku ekspanziju, dok je Internet postao centralna komponenta svetske telekomunikacione infrastrukture.

1.8.5 Aktuelni trendovi

Sfera umrežavanja računara razvija se veoma brzim tempom. Napredak se postiže na svim poljima - u razvoju novih aplikacija, bezbednosti, distribuciji sadržaja, Internet telefoniji, brzinama prenosa u LAN-ovima i brzini ruta. Ipak, tri pojave zaslužuju posebnu pažnju - širokopojasni kućni pristup Internetu, bežični pristup Internetu i P2P umrežavanje.

Kao što smo rekli u odeljku 1.4, sve rasprostranjeniji širokopojasni pristup Internetu iz domova putem kablovskih modema i DSL veza pripremio je teren za nove multimedijalne aplikacije kao što su, na primer, protok visokokvalitetnog video materijala u realnom vremenu na zahtev i interaktivne video konferencije visokog kvaliteta. Sve veća prisutnost veoma brzih (0,1 Mb/s i bržih) javnih Wi-Fi mreža i pristupa Internetu srednje brzine (brzine od nekoliko stotina kb/s) mreža za mobilnu telefoniju omogućavaju, ne samo neprekidnu povezanost sa Internetom, već i čitav niz usluga vezanih za trenutnu lokaciju korisnika. O bežičnim i mobilnim mrežama govorimo u poglavljju 6.

Nakon serije napada tipa uskraćivanja usluga koji su pogodili neke važne web servere krajem 3 990-ih i sve češćih napada parazitskim programima (na primer, Blasterom) koji inficiraju krajnje sisteme i opterećuju mreže saobraćajem, mrežna bezbednost postala je izuzetno značajna tema. U odgovor na ove napade pojavili su se sistemi za prepoznavanje uljeza koji na vreme obaveštavaju o njihovom prisustvu, zatim mrežne barijere koje filtriraju saobraćaj pre nego što ga propuste u mrežu, kao i sistemi koji mogu da pronadu IP adresu sa koje napadi potiču. O ovim i mnogim drugim aspektima bezbednosti govorimo u poglavljju 8.

Poslednja inovacija kojoj ćemo ovde posvetiti pažnju jeste pojava P2P umrežavanja. Aplikacije za P2P umrežavanje koriste resurse u računarima korisnika - prostor na disku, sadržaj, CPU i prisustvo samog korisnika - i ima značajnu autonomiju u odnosu na centralne servere. Računari (ravnopravnih) korisnika najčešće su samo povremeno povezani. U vreme pisanja ovog udžbenika, program KaZaA predstavljao je najpopularniji sistem za P2P razmenu datoteka.

1.9 Rezime

U ovom poglavlju obuhvatili smo zaista širok spektar tema. Najpre smo naveli različite hardverske i softverske komponente od kojih se sastoje Internet i računarske mreže u celini. Pošli smo od „periferije“ mreže i najpre vam prikazali krajnje sisteme i aplikacije, kao i transportne usluge koje su ovim aplikacijama potrebne. Koristeći mrežne distribuirane aplikacije kao primer, predstavili smo vam pojam protokola, inače ključnog koncepta u umrežavanju. Nakon toga, ušli smo dublje u unutrašnjost mreže i stigli do njenog jezgra. Ovde smo identifikovali komutiranje paketa i komutiranje vodova kao dva osnovna pristupa prilikom transporta podataka kroz telekomunikacione mreže i prikazali vam prednosti i mane ova pristupa. Potom smo se bavili i najnižim (u smislu arhitekture) delovima mreže - tehnologijama povezivanja podataka i fizičkim medijumima koji se često sreću u pristupnim mrežama. Ispitali smo strukturu globalnog Interneta i tu videli daje Internet mreža koja objedinjuje veliki broj drugih mreža. Videli ste da Internet ima hijerarhijsku strukturu koja omogućava postojanje hiljada mreža i sastoji se od posrednika za Internet usluge višeg i nižeg reda.

U drugom delu ovog uvodnog poglavlja ispitali smo nekoliko tema koje se mogu smatrati centralnim temama umrežavanja računara. Najpre smo ispitali uzroke kašnjenja i gubitka paketa u mrežama sa komutiranjem paketa. Razvili smo jednostavne kvantitativne modele za kašnjenja u vezi sa prenosom, propagacijom i čekanjem u redu; u domaćim zadacima koje ćemo vam postavljati u ovoj knjizi koristićemo ove modele u značajnoj meri. Zatim smo ispitali slojevitost protokola i modele usluga - ključne arhitektoniske principe umrežavanja na koje ćemo se takođe vraćati u nastavku knjige. Ovaj uvod u umrežavanje završili smo kratkim istorijskim prikazom računarskih mreža. Dakle, ovo prvo poglavlje samo za sebe može da se posmatra kao svojevrstan mini-kurs o umrežavanju računara.

Kao što vidite, u prvom poglavlju smo vam zaista ponudili veoma mnogo informacija. Ukoliko mislite daje to previše, ništa ne brinite. U poglavljima koja slede vratitićemo se na sve ove teme i istražiti ih mnogo detaljnije (ovo shvatite kao obećanje, ne kao pretjeru). U ovom trenutku važno je da imate samo najosnovnija intuitivna znanja o elementima računarske mreže, da u solidnoj meri poznajete terminologiju umrežavanja i da imate želju za daljim učenjem (na ovo poglavlje uvek možete da se vratite). Sve drugo će doći na svoje mesto kako budete napredovali. Čitajući ovu knjigu.

Mapa knjige

Uvek je dobro da, pre nego što krenete na neki put, bacite pogled na mapu, kako biste se upoznali sa glavnim putevima i raskrsnicama koje vas čekaju. Konačni cilj našeg puta jeste duboko razumevanje svih aspekata računarskih mreža. Našu mapu čine poglavlja ove knjige:

1. Računarske mreže i Internet
2. Aplikacijski sloj
3. Transportni sloj
4. Mrežni sloj
5. Sлој veze podataka i lokalne računarske mreže
6. Bežično i mobilno umrežavanje
7. Multimedijalno umrežavanje
8. Bezbednost u računarskim mrežama
9. Upravljanje mrežama

Gledajući ovu mapu, rekli bismo da poglavlja od 2 do 5 predstavljaju četiri ključne celine knjige. Verovatno primećujete da je svakom od Četiri gornja sloja familije protokola za Internet posvećeno po jedno poglavlje. Skrećemo vam pažnju i na to da naše putovanje počinje od vrha ove familije protokola - dakle, od aplikacijskog sloja, i da se zatim nastavlja naniže. Ovakav pristup od vrha ka dnu usvojen je zato što ćete, kada upoznate aplikacije, mnogo lakše razumeti mrežne usluge koje treba da podrže rad tih aplikacija. Dakle, upoznavanje aplikacija motiviše vas za ostatak teksta.

U drugoj polovini knjige (poglavlja od 6 do 9) usredsredili smo se na Četiri veoma važne (i na neki način nezavisne) teme savremenih računarskih mreža. U poglavlju 6, bavimo se bežičnim i mobilnim tehnologijama, uključujući Wi-Fi LAN-ove, GSM i mobilni IP. U poglavlju 7, „Multimedijalno umrežavanje“ ispitaćemo aplikacije za audio i video kao što su Internet telefonija, video konferencije i protok uskladištenih medija u realnom vremenu. Osim toga, videćete i na koji način se dizajniraju mreže sa komutiranjem paketa ukoliko se želi dosledan kvalitet usluga audio i video aplikacija. U poglavlju 8, „Bezbednost u računarskim mrežama“, najpre ćemo se baviti osnovama šifrovanja i mrežne bezbednosti, a zatim ćemo videti na koji se način te osnovne teorijske postavke konkretizuju u velikom broju različitih situacija na Internetu. U poslednjem poglavlju, „Upravljanje mrežama“, ispitaćemo ključne aspekte upravljanja mrežama, kao i primarne Internet protokole koji se za to koriste.

Domaći zadatak: problemi i pitanja Poglavlje 1

Kontrolna pitanja ODEUCI 1 . 1 - 1 . 5

1. U čemu je razlika između „računara“ i „krajnjeg sistema“? Navedite različite tipove krajnjih sistema. Da li je veb server krajnji sistem?
2. Reč *protokol* često se u medijima povezuje sa diplomatom. Navedite jedan primer diplomatskog protokola.
3. Staje klijentski program? Staje serverski program? Da li serverski program zahteva i dobija usluge od klijentskog programa?
4. Navedite dva tipa usluga koje Internet pruža svojim aplikacijama. Navedite neke njihove karakteristike.
5. Rečeno je da su kontrola toka i kontrola zagruženja ekvivalentni pojmovi. Da li to važi i za Internet usluge sa konekcijom? Da li su ciljevi kontrole toka i kontrole zagruženja identični?
6. Ukratko ispričajte na koji način Internet usluga sa konekcijom obezbeđuje pouzdan transport.
7. U čemu je prednost mreže sa komutiranjem vodova u odnosu na mreže sa komutiranjem paketa? U čemu je prednost tehnologije TDM u odnosu na FDM tehnologiju u mrežama sa komutiranjem vodova?
8. Zbog čega se kaže da komutiranje paketa primenjuje statističko multipleksiranje? Uporedite statističko multipleksiranje sa multipleksiranjem koje se odvija u okviru TDM tehnologije.
9. Prepostavimo da se između izvornog i odredišnog računara nalazi tačno jedan komutator paketa. Brzine prenosa između izvornog računara i komutatora, i komutatora i odredišnog računara, su R_1 i R_2 . Ukoliko prepostavimo daje reč o komutatora sa memorisanjem i prosleđivanjem, koliko iznosi ukupno kašnjenje paketa dužine L od jednog do dragog kraja njegovog puta? (Zanemariti kašnjenja usled stajanja u redu, propagacije i obrade.)
10. Šta se u mreži sa virtuelnim kolima podrazumeva pod informacijom o statusu veze? Ukoliko se u komutatora jedne virtuelne mreže veze uspostavljaju i raskidaju brzinom od jedne veze po milisekundi (prosečno), kojom brzinom treba da se ažurira tabela za prosleđivanje ovog komutatora?
11. Prepostavimo da razvijate standard za novu vrstu mreže i da treba da se opre-delite ili za virtuelna kola ili za rutiranje datagrama. Šta su prednosti, a Šta mane virtuelnih kola?
12. Navedite šest različitih tehnologija mrežnog pristupa. Klasifikujte svaku od njih u kategorije kućnog, poslovnog i mobilnog pristupa.
13. Šta predstavlja ključnu razliku između ISP-a prvog reda i ISP-a dragog reda?
14. U čemu je razlika između POP i NAP tačke?

16. Da li je propusna moć HFC mreža namenska ili deljena između korisnika? Da li su na nizvodnom HFC kanalu moguće kolizije? Zašto jesu ili zašto nisu?
17. Koja je brzina prenosa Ethernet LAN-ova? Da li data brzina znači da svaki korisnik LAN-a može daje ostvaruje u kontinuitetu?
18. Koji fizički medijumi mogu da se koriste u Ethernet mrežama?
19. Za kućni pristup Internetu koriste se telefonski modemi, HFC i ADSL veze. Za svaku od ovih tehnologija navedite moguće brzine prenosa, kao i to da li je propusna moć namenska ili deljena.

ODEUCI 1.6-1.8

19. Zamislite slanje serije paketa od izvornog do odredišnog računara fiksnom rutom, a zatim navedite sve pojedinačne komponente celokupnog kašnjenja paketa od jednog do drugog kraja. Koja su kašnjenja konstantna, a koja pro-menljiva?
20. Navedite pet zadataka koje sloj može da izvrši. Da li je moguće da neki od tih zadataka izvrše neka dva druga sloja (ili više njih)?
21. Navedite pet slojeva familije Internet protokola i osnovne nadležnosti svakog od njih.
22. Šta je poruka aplikacijskog sloja? Šta je segment transportnog sloja? Šta je datagram mrežnog sloja? Šta je okvir veze podataka?
23. Koji su slojevi u familiji Internet protokola zaduženi za proces rutiranja, koji za proces komutiranja sloja veze, a koji za serverske usluge?

Problemi

1. Dizajnirajte i opišite protokol aplikacijskog nivoa koji će se koristiti između automatskog bankomata i centralnog računara banke. Ovaj protokol treba da omogući proveru korisnikove kartice i lozinke, zatim proveru njegovog računa (koji se čuva u centralnom računaru), kao i povlačenje određenog novčanog iznosa sa datog računa (odnosno, isplatu novca korisniku). Elementi vašeg protokola treba da budu u stanju da izadu na kraj sa veoma čestom pojavom da je traženi iznos veći od onog koji je na računu. Svoj protokol uobičiće navođenjem poruka koje će biti razmenjene kao i akcija koje bankomati i centralni računar treba da preduzmu kada dobiju ove poruke. Skicirajte rad svog protokola za slučaj jednostavnog povlačenja bez ikakvih grešaka i nacrtajte dijagram koji je sličan slici 1.2. Eksplicitno navedite pretpostavke koje postavlja vaš protokol u vezi sa transportnom uslugom od jednog do drugog kraja, koja se nalazi ispod njega.

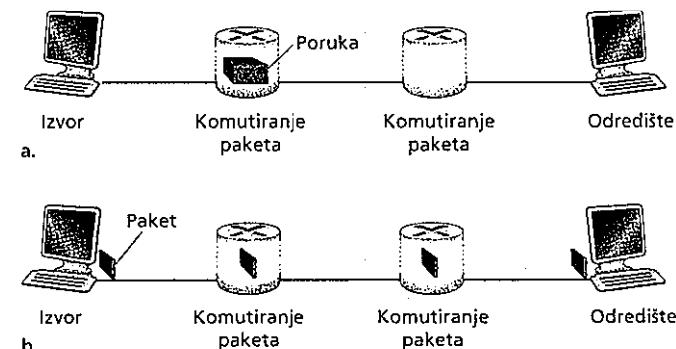
2. Zamislite aplikaciju koja svoje podatke prenosi ustaljenom brzinom (primera radi, pošiljalac generiše N bitova svakih k vremenskih jedinica, gde je k kratak i fiksni vremenski period). Nakon pokretanja, ova aplikacija treba da bude aktivna relativno dug vremenski period. Odgovorite na sledeća pitanja i ukratko obrazložite svoje odgovore:
 - a) Da li bi za ovu aplikaciju bila podesnija mreža sa komutiranjem paketa ili mreža sa komutiranjem vodova? Zašto?
 - b) Prepostavimo da se koristi mreža sa komutiranjem paketa i da sav njen saobraćaj stvara opisana aplikacija. Pored toga, prepostaviti ćemo da je zbir brzine prenosa aplikacije manji od kapaciteta svakog linka. Da lije u mreži potreban neki mehanizam za kontrolu zagušenja? Zašto?
3. Vratimo se za trenutak na mrežu sa komutiranjem paketa sa slike 1.5. Podsećamo vas da u njoj svaki link ima n vodova.
 - a) Koji je maksimalan broj istovremenih aktivnih veza u bilo kom trenutku u ovoj mreži?
 - b) Prepostavimo da se sve veze nalaze između komutatora u levom gornjem uglu i komutatora u desnom donjem uglu slike. Koji je maksimalan broj istovremenih aktivnih veza?
4. Vratimo se sada na analogiju sa povorkom automobila iz odeljka 1.6. I u ovom slučaju prepostavitićemo da brzina propagacije iznosi 100 km/h.
 - a) Prepostavimo sada da karavan treba da putuje 200 kilometara - da krene od jedne naplatne rampe, samo prode kroz drugu i svoj put završi na trećoj naplatnoj rampi. Koliko iznosi ukupno kašnjenje od jedne do druge krajnje tačke?
 - b) Ponovo odgovorite na pitanja pod a), ali sada sa sedam automobila umesto sa deset.
5. Zamislite slanje paketa od F bitova putanjom od Q linkova brzine prenosa R b/s. Mreža je samo blago opterećena tako da nema kašnjenja usled stajanja u redu. Kašnjenje usled propagacije je beznačajno.
 - a) Prepostavimo daje u pitanju mreža sa komutiranjem paketa i virtuelnim kolima. Neka trajanje pripreme virtualnih kola bude t , sekundi. Prepostavitićemo i to da izvorni slojevi dodaju ukupno h bitova zaglavja svakom paketu. Koliko je vremena potrebno za slanje ove datoteke od izvora do odredišta?
 - b) Prepostavimo daje u pitanju mreža sa datagramima i komutiranjem paketa, da se koristi usluga bez konekcije, kao i to da svaki paket ima zaglavje dužine $2h$. Koliko je vremena sada potrebno za slanje paketa?
 - c) Konačno, prepostavimo daje reč o mreži sa komutiranjem vodova i da brzina prenosa između izvora i odredišta iznosi R b/s. Ukoliko je i , trajanje pripreme veze, a h broj bitova zaglavja koje se dodaje čitavom paketu, koliko je vremena potrebno za slanje ovog paketa?

6. Ovim elementarnim problemom započinjemo istraživanje kašnjenja usled propagacije i kašnjenja usled prenosa, inače dva centralna aspekta umrežavanja. Zamislite dva računara A i B koji su povezani jednim linkom čija je brzina prenosa R b/s. Prepostavimo da su ova dva računara međusobno udaljena m metara i da je brzina propagacije linka s metara u sekundi. Računar A treba da pošalje paket dužine L bitova računani B.
- Izrazite kašnjenje usled propagacije d_{prop} u funkciji m i s .
 - Odredite trajanje prenosa paketa d_{prenos} u odnosu na L i R .
 - Zanemarujući kašnjenja usled obrade i stajanja u redu, napišite izraz za kašnjenje od jednog do drugog kraja.
 - Prepostavimo da računar A počne da emituje paket u trenutku $t = 0$. Gde se u trenutku $t = d_{prenos}$ nalazi poslednji bit paketa?
 - Prepostavimo daje d_{prop} veće od d_{prenos} . Gde se u trenutku $t = d_{prenos}$ nalazi prvi bit paketa?
 - Prepostavimo daje d_{prop} manje od d_{prenos} . Gde se u trenutku $t = d_{prenos}$ nalazi prvi bit paketa?
 - Prepostavimo da je $s = 2,5 \cdot 10^8$, $L = 100$ bitova, a $R = 28$ kb/s. Odredite udaljenost m tako da d_{prop} bude jednak sa d_{prenos} .
7. U ovom problemu bavimo se slanjem glasa od računara A do računara B kroz mrežu sa komutiranjem paketa (na primer, Internet telefonija). Računar A u hodu pretvara analogni signal glasa u digitalni niz bitova od 64 kb/s, a zatim ove bitove grupiše u 48-bitne pakete. Između računara A i računara B je jedan link čija je brzina prenosa 1 Mb/s, a kašnjenje usled prenosa 2 ms. Čim napravi paket, računar A ga odmah šalje računaru B. Čim računar B primi čitav paket, on njegove bitove pretvara u analogni signal. Koliko vremena prolazi od trenutka nastajanja bita (originalni analogni signal na računaru A) pa do trenutka njegovog dekodovanja (kao dela analognog signala na računaru B)?
8. Prepostavimo da neki korisnici zajednički koriste link brzine 1 Mb/s i da je svakom od njih za emitovanje potrebna propusna moć od 100 kb/s, ali da to čine samo 10 procenata ukupnog vremena. (Pročitajte odeljak 1.3 u kome smo govorili o komutiranju paketa i komutiranju vodova.)
- Koliko korisnika može da bude podržano komutiranjem vodova?
 - U nastavku ovog zadatka prepostavimo da se koristi komutiranje paketa. Odredite verovatnoću emitovanja za određenog korisnika.
 - Prepostavimo da u mreži ima 40 korisnika. Odredite verovatnoću istovremenog emitovanja tačno n korisnika. (Savet: Koristite binomnu raspodelu.)
 - Odredite verovatnoću istovremenog emitovanja 11 ili više korisnika.
9. Vratimo se za trenutak na odeljak 1.3 u kome smo poredili komutiranje paketa i komutiranje vodova i naveli primer sa linkom kapaciteta 1 Mb/s. Kada su korisnici aktivni, oni stvaraju saobraćaj od 100 kb/s, ali verovatnoća njihove aktivnosti je samo $p = 0,1$. Prepostavimo sada da se link od 1 Mb/s zameni linkom od 1 Gb/s.
- Koliko je N , tj. maksimalan broj korisnika koje komutiranje vodova može da podrži istovremeno?
 - Prepostavimo sada da je u pitanju mreža sa komutiranjem paketa od M korisnika. Napišite formulu (u funkciji p , M , N) za izračunavanje verovatnoće da više od N korisnika istovremeno šalje svoje podatke.
 - Zamislite sada stajanje u redu u privremenoj memoriji ruteru (koja prethodi izlaznom linku). Prepostavimo da je dužina svih paketa L bitova, brzina prenosa R b/s i da //paketista ovremeno stiže u privremenu memoriju svakih LM/R sekundi. Odredite prosečno kašnjenje paketa usled stajanja u redu. (Savet: Kašnjenje usled stajanja u redu prvog paketa je nula, drugog L/R , trećeg $2L/R$. $M-X$ paket je već prenet kada u ruter dospe druga serija paketa.)
 - Ponovo zamislite stajanje u redu u privremenoj memoriji nekog ruteru. Slovom / označićemo intenzitet saobraćaja; $/ = La/R$. Prepostavimo da je kašnjenje usled stajanja u redu $ILIR(1-I)$ za < 1 .
 - Napišite formulu za ukupno kašnjenje koje se dobija sabiranjem kašnjenja usled stajanja u redu i kašnjenja usled prenosa.
 - Nacrtajte ukupno kašnjenje u funkciji L/R .
 - a) Generalizujte formulu za kašnjenje od jednog do drugog kraja mreže iz odeljka 1.6 za različite brzine obrade i prenosa, kao i za različito kašnjenje usled propagacije.
 - b) Ponovite postupak pod a) ali sada prepostavite da na svakom čvoru prosečno kašnjenje usled stajanja u redu iznosi d_{red} .
 - Programom *Traceroute* u tri različita doba dana testirajte vezu između izvornog i odredišnog računara na istom kontinentu.
 - Odredite prosečnu vrednost i standardnu devijaciju kašnjenja povratnog puta u svakom od tri doba dana.
 - Ustanovite broj ruta na putanji u svakom od tri doba dana. Da li se putanje menjaju tokom dana?
 - Pokušajte da identifikujete broj ISP mreža kroz koje paketi programa *Traceroute* prolaze na svom putu od izvora do odredišta. Rutere sličnih imena i (ili) sličnih IP adresa treba smatrati delom mreže istog ISP-a. Da li se u vašim eksperimentima najduža kašnjenja javljaju na tačkama vezivanja susednih posrednika za Internet usluge?
 - Ponovite prethodne zadatke, ali sada za izvorne i odredišne računare na različitim kontinentima, a zatim uporedite kontinentalne i interkontinentalne rezultate.
 - Prepostavimo da su dva računara međusobno udaljena 10 000 kilometara i da su povezani direktnim linkom brzine $R = 1$ Mb/s. Takođe ćemo prepostaviti i da je brzina propagacije linka iznosi $2,5 \cdot 10^8$ metara u sekundi.
 - Izračunajte „proizvod propusni opseg-kašnjenje”, $R \cdot t_{prop}$.
 - Od računara A do računara B treba poslati datoteku od 400 000 bitova. Prepostavimo da se ova datoteka šalje kao jedna velika kontinuirana poruka. Koji je maksimalan broj bitova koji u bilo kom trenutku može da se nade na linku?

- d) Objasnite rezultat proizvoda propusne moći i kašnjenja.
- e) Koja je širina bitova (u metrima) u linku? Da li su duži od fudbalskog igrališta?
- f) Izvedite uopšteni izraz za izračunavanje širine bitova na osnovu brzine propagacije s , propusnog opsega R i dužine linka m .
15. Vraćajući se na prethodni problem, pretpostavimo da možemo da izmenimo parametar R . Za koju vrednost ovog parametra je dužina bita jednaka dužini linka?
16. Ponovo se vraćamo na problem pod rednim brojem 14, ali će sada propusna moć linka biti $R = 1 \text{ Gb/s}$.
- Izračunajte proizvod propusne moći i kašnjenja, $R \bullet t_{prop}$.
 - Uzmimo da od računara A do računara B treba da se pošalje datoteka dužine 400 000 bitova i to ponovo kao jedinstvena poruka. Koji je maksimalan broj bitova koji u bilo kom trenutku može da se nade na linku?
 - Koja je širina (u metrima) bitova u linku?
17. Ponovo se vratite na problem pod rednim brojem 14.
- Koliko je vremena potrebno za slanje datoteke pod pretpostavkom da se ona šalje kontinuirano?
 - Pretpostavimo sada daje datoteka razbijena na deset paketa od po 40 000 bitova, da svaki paket primalac treba da potvrđi, kao i to daje trajanje prenosa paketa sa potvrdom zanemarljivo. Konačno, pretpostavimo i to da pošiljalac ne može da pošalje sledeći paket sve dok ne dobije potvrdu o prijemu prethodnog. Koliko je vremena potrebno za slanje ove datoteke?
 - Uporedite rezultate pod a) i b).
18. Pretpostavimo da se između geostacionarnog satelita i njegove bazne stanice na Zemlji nalazi mikrotalasni link propusnog opsega 10 Mb/s . Svakog minuta satelit snima po jednu digitalnu fotografiju koju zatim šalje baznoj stanci. Pretpostavljamo da brzina propagacije iznosi $2,4 \bullet 10^8 \text{ metara u sekundi}$.
- Koliko iznosi kašnjenje usled propagacije ovog linka?
 - Izračunajte proizvod propusne moći i kašnjenja, $R \bullet t_{prop}$.
 - Uzmimo da je x veličina fotografije. Koja je minimalna veličina x sa kojom mikrotalasni link emituje u kontinuitetu?
19. Vratimo se sada na analogiju sa avionskim transportom iz odeljka 1.7 i dodavanje zaglavja jedinicama podataka protokola na njihovom putu naniže kroz familiju protokola. Postoji li ekvivalentan pojam ovom dodavanju zaglavja kod putnika i prtljaga kod njihovog kretanja naniže kroz protokol avionskog transporta?
20. U savremenim mrežama sa komutiranjem paketa izvorni računar segmentira duge poruke aplikacijskog sloja (recimo, sliku ili muzičku datoteku) na manje

pakete i šalje ih kroz mrežu. Primalac zatim ponovo sastavlja ove pakete i od njih dobija originalnu poruku. Ovaj proces nazvali smo *segmentiranjem*. Na slici 1.21 ilustrovan je transport sa kraja na kraj mreže sa segmentiranjem i bez njega. Zamislite da od izvornog do odredišnog računara sa slike 1.21 treba poslati poruku dužine $7,5 \bullet 10^6$ bitova. Pretpostavljamo da svaki link ima brzinu prenosa od $1,5 \text{ Mb/s}$, a zanemariti kašnjenja usled propagacije, stajanja u redu i obrade.

- Zamislite slanje poruke od izvora do odredišta *bez* njenog segmentiranja. Koliko je vremena potrebno da ona stigne do prvog komutatora? Imajući u vidu činjenicu da svaki komutator memoriše i prosleđuje pakete, koliko je ukupno vremena potrebno za prebacivanje poruke od izvornog do odredišnog računara?
 - Pretpostavimo sada daje ista poruka segmentirana na 5 000 paketa od po 1 500 bitova. Koliko je vremena potrebno za prenos prvog paketa od izvornog računara do prvog komutatora? Čim se prvi paket pošalje iz prvog komutatora ka drugom, drugi paket se šalje od izvornog računara ka prvom komutatoru. U kom trenutku će drugi paket dospeti do prvog komutatora?
 - Koliko je vremena potrebno za prenos cele datoteke od izvornog do odredišnog računara ukoliko se koristi segmentiranje? Uporedite ovaj rezultat sa odgovorom pod a) i obrazložite.
 - Raspravljajte o nedostacima segmentiranja poruka.
21. Eksperimentišite malo sa Java apletom za segmentaciju poruka koju ćete pronaći na veb lokaciji ove knjige. Da li kašnjenja u ovom apletu odgovaraju kašnjenjima iz prethodnog pitanja? Na koji način kašnjenja usled propagacije utiču na ukupno kašnjenje od jednog do drugog kraja u komutiranju paketa (sa segmentacijom poruka), a na koji u komutiranju poruka?



Slika 1.21 ♦ Transport poruke sa kraja na kraj mreže: (a) sa segmentiranjem; (b) bez segmentiranja.

22. Pretpostavimo da od računara A do računara B treba poslati veliku datoteku od F bitova.

Između računara A i B postoje dva linka (ijedan komutator) koji nisu zagušeni (dakle, nema stajanja u redu). Računar A segmentira datoteku na segmente od po S bitova i svakom segmentu dodaje zaglavje od 40 bitova, čime se dobijaju paketi dužine $L = 40 + S$ bitova. Brzina prenosa svakog linka iznosi R b/s. Pronadite vrednost parametra S koja minimizira kašnjenje prilikom premeštanja datoteke sa računara A na računar B. Zanemarite kašnjenje usled propagacije.

Teze za diskusiju

1. Koje vrste bežičnih mobilnih usluga postoje u vašem kraju?
2. Korišćenjem bežične LAN tehnologije 802.1 lb dizajnirajte mrežu koja bi povezala vaš dom i dom vaših roditelja. Navedite uređaje koje biste upotrebili i njihove cene.
3. Staje „od PC-ja do telefona“? Pronadite veb lokacije koje se bave ovim poslom.
4. Šta se podrazumeva pod uslugom SMS (*Short Message Service*)? Da li je ova usluga popularna u svim delovima sveta? Gde je najpopularnija? Da li je moguće sa veb lokacije poslati SMS poruku mobilnom telefonu?
5. Šta se podrazumeva pod protokom usklađenog audio materijala u realnom vremenu? Opišite neke od postojećih proizvoda za internetski protokol audio signala u realnom vremenu. Pronadite neke veb lokacije kompanija koje se time bave.
6. Šta su Internet video konferencije? Opišite neke od postojećih proizvoda za Internet video konferencije. Pronadite veb lokacije kompanija koje se time bave.
7. Navedite pet kompanija koje obezbeđuju uslugu P2P razmene datoteka i za svaku od njih navedite vrstu datoteka čiju razmenu podržavaju.
8. Staje trenutna razmena poruka? Postoje li PDA ili slični uređaji koji bi mogli da vam omoguće pristup uslugama za trenutnu razmenu poruka?
9. Ko je izmislio ICQ, inače prvu uslugu za trenutnu razmenu poruka? Kada je ova usluga izmišljena i koliko su bili stari njeni pronalazači? Koje izmislio uslugu Napster? Kada je ova usluga izmišljena i koliko su bili stari njeni pronalazači?
10. Uporedite bežični pristup Internetu tehnologijama Wi-Fi i 3G. Koje brzine prenosa omogućavaju ove dve usluge? Koja je njihova cena? Razgovarajte o romingu i sveopštoj prisutnosti pristupa Internetu.
11. Zašto je Napster prestao da postoji. Šta je RIAA i koje mere ovo telo predu-zima u cilju sprečavanja P2P razmene sadržaja zaštićenog autorskim pravima? Objasnite razliku između direktnog i indirektnog kršenja autorskih prava.

12. Da li mislite da će za deset godina i dalje biti razmene zaštićenog materijala kroz računarske mreže? Obrazložite svoj odgovor.



Ethereal laboratorijska vežbanja

„Kaži mi i zaboraviću. Pokaži mi i zapamtiću. Uključi i mene u rad i razumeću.“

Kineska poslovica

Razumevanje mrežnih protokola često zavisi od toga da li ste bili u prilici da ih vidite na delu i poigrate se sa njima - prateći razmenu poruka između dva mrežna entiteta, ulazeći dublje u načine njihovog rada, iniciranje određenih njihovih akcija i praćenje tih akcija i njihovih posledica. Ovo se može učiniti u simuliranim scenarijima ili u stvarnom mrežnom okruženju kao što je, na primer, Internet. U Java aplletima koje ćete pronaći na pratećoj veb lokaciji ove knjige primenjen je prvi princip, dok smo se u Ethereal laboratorijskim vežbanjima pridržavali drugog. Koristeći mrežne aplikacije na svom računaru kod kuće, na poslu ili u laboratoriji, nači ćete se u najrazličitijim situacijama. Pratićete rad mrežnih protokola, ali i razmenjivati poruke sa entitetima protokola koji se izvršavaju u nekom drugom kraju Interneta. Prema tome, vi i vaš računar bićete integralni delovi ovih pet laboratorijskih vežbi. Posmatraćete i, u krajnjoj liniji, naučiti radeći praktično.

Osnovna alatka za praćenje razmene poruka između aktivnih entiteta protokola naziva se **sakupljač paketa** (engl. *packet sniffer*). Kao što iz naziva ove alatke možete da zaključite ona pasivno kopira (uvlači) poruke koje se šalju iz vašeg računara ili ka njemu. Osim toga, ova alatka prikazuje sadržaj različitih polja protokola ovako uhvaćenih poruka. Na slici 1.22 možete da vidite snimak ekrana sakupljača paketa Ethereal. U pitanju je besplatan program koji može da se izvršava na računa-rima pod operativnim sistemima Windows, Linux/Unix i Mac. U poglavljima od I do 5 pronaći ćete Ethereal laboratorijska vežbanja pomoću kojih ćete istražiti veliki broj protokola pomenutih u ovom poglavlju. U prvom vežbanju najpre ćete da nabavite i instalirate kopiju programa Ethereal. Nakon toga pristupiće nekoj veb lokaciji kako biste uhvatili i ispitati poruke protokola koje razmenjuju vaš čitač Weba i odgovarajući veb server.

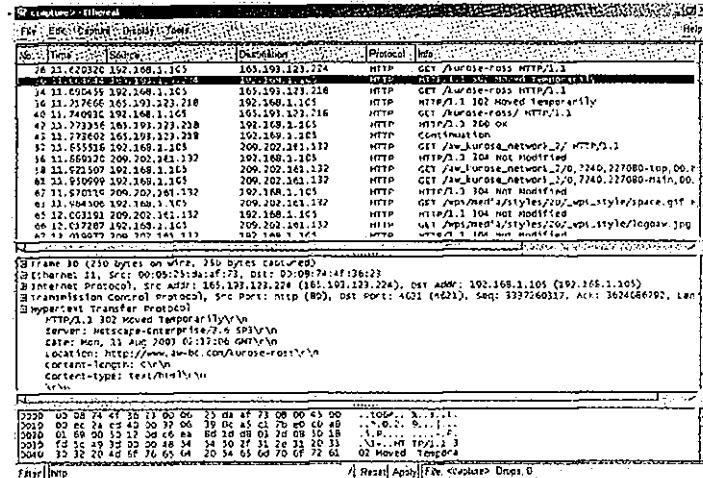
Sve detalje u vezi sa ovim prvim Ethereal vežbanjem (uključujući i instrukcije o nabavljanju i instaliranju ovog programa) nalaze se na veb lokaciji <http://www.awl.com/kurose-ross>.

Komandni meni

Listing uhvaćenih poruka

Detalji o zaglavljivanju izabranog paketa

Sadržaj paketa u heksadecimal- nom i ASCII obliku



Slika 1.22 ♦ Snimak Ethereal ekrana

INTERVJU

Leonard Klajnrolc

Leonard Klajnrok je profesor računarskih nauka na Univerzitetu UCLA. Godine 1969. njegov računar na ovom univerzitetu postao je prvi čvot Interneta.

Tehnologijo komutiranja paketa koju je izumeo 196]. godine posaoia je okosnicu Interneta. Leonard je i predsednik i osnivač kompanije Nomadix, Inc. čija tehnologija obezbeđuje boju pristupačnost širokopojasnih Interne! usluge.

Diplomirao je na Gradskom koledžu u Njujorku [CCNY], dok je magisterske doktorske studije u oblasti inženjerstva elektrotehnike završio na MIT-u.

Zbog čega ste se opredelili za specijalizaciju u oblasti tehnologije umrežavanja i Interneta?

Na doktorskim studijama na MIT-u 1959. godine primetio sam da se većina mojih kolega opredelila za informatiku i teoriju kodiranja. Na MIT-u je radio Klod Šanon, izuzetan istraživač, koji je pokrenuo ove oblasti i sam već rešio mnoge značajne probleme. Preostali istraživački problemi bili su teški i nisu bili toliko značajni. Zato sam odlučio da sam pokrenem novu oblast - nešto čime se do tada нико nije bavio. Podsećam vas da sam na MIT-u bio okružen velikim brojem računara i ubrzo mi je postalo jasno da svi ti računari treba da komuniciraju. U to vreme jednostavno nije postojao način da se ta komunikacija ostvari, tako da sam odlučio da razvijem tehnologiju koja bi omogućila stvaranje efikasnih mreža za protokol podataka.

Sta je bio vaš prvi posao u računarskoj Industriji? Kako je on izgledao

Između 1951. i 1957. sam kao student posećivao večernja predavanja na Siti koledžu u Njujorku. Tada sam radio, najpre kao tehničar, a zatim i kao inženjer, u maloj firmi za industrijsku elektroniku po imenu Photobell. Radeci za tu firmu u njihovo proizvodnu liniju uveo sam digitalnu tehnologiju. U osnovi, koristili smo fotoelektrične uređaje za detektovanje prisustva određenih objekata (kutija, ljudi i slično). Električna kola poznata pod nazivom *bisiabihi multivibratori* bila su upravo one vrsta tehnologije koja nam je bila potrebna za uvođenje digitalne obrade u ovo polje. Ispostavilo se da su ova kola postala osnovni delovi računara, a u današnjem žargonu nazivaju „flip-flopovi“ ili prekidači (preklonici).

O Čemu ste razmišljali prilikom slanja prve poruke od računara do računara (iz Univerziteta UCLA ka Istraživačkom institutu Stanford)?

Prva ponika između dva računara predstavljala je svojevrstan antiktimaks. Mnogo impre-sivniji trenutak za mene bio je 2. septembar 1969. godine kada je prvi delić mrežne opreme (IMP) povezan sa prvim spoljnim operativnim sistemom (bio je to moj računar na Univerzitetu UCLA). Tada je rođen Internet. Nešto ranije iste godine, u jednom saopštenju za javnost Univerziteta UCLA rekao sam da će odmah nakon podizanja i puštanja u rad računarskih mreža, biti moguć prisprom računarima iz naših domova ili kancelarija i da će to biti isto onoliko jednostavno koliko i ulazak u električnu ili telefonsku mrežu. Dakle, još

tada sam imao viziju o tome da će Internet biti sveobuhvatan, uvek prisutan i uvek dostupan svakome ko ima neki uređaj za povezivanje. Ipak, nikada ne bih mogao da prepostavim da će danas Čak i moja 94-godišnja majka koristiti Internet, što ona zaista Čini.

Kako Izgleda vaša vizija budućnosti umrežavanja?

Najjasniji deo moje vizije predstavljaju nomadsko računarstvo i inteligentan prostor. Dostupnost lakih, jeftinih i moćnih prenosivih računara, sa jedne strane, kao i sveobuhvata-nost Interneta, s druge, omogućili su nam da postanemo nomadi. Pojam nomadsko računarstvo odnosi se na tehnologiju koja omogućava krajnjim korisnicima da putujući od jednog do drugog mesta na jednostavan način ostvaruju pristup Internet uslugama bez obzira na to gde putuju. Međutim, nomadsko računarstvo je samo prvi korak. Sledеći korak će nam omogućiti iskorak iz paklenog virtuelnog prostora u fizički svet intelligentnog prostora. Naše okruženje (stolovi, zidovi, vozila, satovi, kaiševi) oživeće kroz tehnologiju aktuatora, senzora, logike, obrade, skladištenja, kamera, mikrofona, zvučnika, ekrana i komunikacije. Ova ugrađena tehnologija će omogućiti našem okruženju da nam obezbedi IP usluge koje želimo. Kada uđem u sobu, soba će znati da sam to učinio. Moći ću da komuniciram sa svojim okruženjem prirodno, služeći se pri tom engleskim jezikom; moji zahtevi će generisati odgovore koji će mi biti predstavljeni u vidu veb stranica na zidnim ekranima, u mojim nao-čarima, kao govor, ili u obliku holograma.

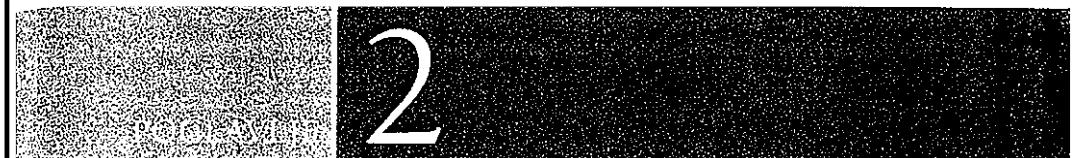
Gledajući još dalje u budućnost, umrežavanje vidim sa još nekim dodatnim komponentama. Vidim inteligentne softverske mrežne agente čiji su zadaci skupljanje podataka, reagovanje na prikupljene podatke, praćenje trendova i adaptivno i dinamičko izvršavanje zadataka. Vidim mnogo već obim mrežnog saobraćaja koji ne generišu ljudi, već ti ugrađeni uređaji i softverski agenti. Ovu ogromnu mrežu konirolisaće veliki skupovi samoorganizujućih sistema. Kroz ovu mrežu trenutno će se kretati ogromna količina informacija koje će prolaziti kroz izuzetno obimnu obradu i filtriranje. Internet će u osnovi predstavljati prožimajući nervni sistem. Sve ove i mnoge druge stvari vidim u našem putu kroz 21. vek.

Ko vas je u profesionalnom smislu najviše impresionirao?

Bez konkurenциje Klod Šanon sa MIT-a - brilljantan istraživač koji je posedovao dar da svoje matematičke ideje poveže sa fizičkim svetom na krajnje intuitivne načine. On je bio član komisije pred kojom sam branio doktorsku tezu.

Da li imate savet za studente koji tek zakoračuju u polje umrežavanja i Interneta?

Internet i sve ono što on omogućava predstavlja ogroman front pun neverovatnih izazova. U njemu i te kako ima prostora za inovacije. Ne dozvolite da vas ograniči današnja tehnologija. Zakoračite u njega, zamislite šta bi moglo da bude i zatim učinite da se to i ostvari.



Aplikacijski sloj

Mrežne aplikacije predstavljaju *raison d'être* računarskih mreža. Bez korisnih aplikacija ne bi bilo razloga ni za dizajniranje mrežnih protokola koji ih podržavaju. Srećom, u proteklih 35 godina napravljene su mnoge izvrsne mrežne aplikacije. Pokušajmo da se prisjetimo makar onih najvažnijih. Najpre, to su klasične aplikacije zasnovane na tekstu koje su postale popularne 1980-ih, kao što su bili elektronska pošta, daljinski pristup računarima, transfer datoteka, elektronske konferencije i interaktivni razgovor, svi zasnovani na tekstu. Zatim, tu je i najpopularnija aplikacija 1990-ih - Web. Nedugo zatim, pojavile su se i mnoge multimedijalne aplikacije, kao što je protokol video signala u realnom vremenu (*streaming*), Internet radio, Internet telefonija i video konferencije. Konačno, pomenućemo i dve izuzetne aplikacije s kraja milenijuma - instantna razmena poruka sa kontakt listama i P2P razmena datoteka.

U ovom poglavlju bavićemo se konceptualnim aspektima mrežnih aplikacija, kao i aspektima njihove implementacije. Počećemo definisanjem ključnih koncepcata aplikacijskog sloja, kao što su njegovi protokoli, zatim klijenti i serveri, procesi, soketi i interfejsi transportnog sloja. Zatim ćemo detaljno ispitati i nekoliko mrežnih aplikacija, kao što su Web, elektronska pošta, DNS i P2P razmena datoteka.

Nakon toga prikazaćemo vam i proces programiranja mrežnih aplikacija, bez obzira na to da li koriste protokol TCP ili protokol UDP. Zadržaćemo se na soketima i proći kroz neke jednostavne klijentsko-serverske aplikacije koje su urađene u programskom jeziku Java. Konkretno, videćete na koji način u Javi može da se napravi jednostavan veb server. Osim toga, na kraju poglavlja nalazi se i nekoliko zanimljivih zadataka u vezi sa programiranjem soketa.

Aplikacijski sloj predstavlja odlično mesto za početak našeg poučavanja protokola, pre svega zato što je u pitanju poznati teren. Sigurno su vam već dobro poznate mnoge aplikacije koje se oslanjaju na protokole o kojima ćemo govoriti. Preko aplikacijskog sloja stići ćete tačnu predstavu o protokolima i upoznati se sa mnogim aspektima na koje ćemo se vratiti kada budemo govorili o protokolima transportnog i mrežnog sloja, kao i sloja veze podataka.

2.1 Principi rada mrežnih aplikacija

U najpopularnije mrežne aplikacije spadaju:

- ◆ elektronska pošta;
- ◆ Web;
- ◆ instantna razmena poruka;
- ◆ prijavljivanje na udaljene računare (na primer, Telnet ili SSH);
- ◆ P2P razmena datoteka;
- ◆ transfer datoteka između dva naloga na dva računara (na primer, FTP);
- ◆ mrežne igre sa velikim brojem korisnika;
- ◆ protok uskladištenog video materijala u realnom vremenu; ◆ Internet telefonija;
- ◆ video konferencije u realnom vremenu.

Na većinu ovih aplikacija vratićemo se, manje ili više detaljno, u nastavku knjige. Primera radi, u ovom poglavlju govorićemo o elektronskoj pošti, Webu i P2P razmeni datoteka. U poglavlju 7 u kome će biti reči o multimedijalnom umrežavanju vratićemo se na protok video signala u realnom vremenu, kao i Internet telefoniju.

Prepostavimo sada da imate izuzetnu ideju za novu mrežnu aplikaciju. Sasvim je nevažno da li će ona predstavljati dobrobit za čovečanstvo, izvor materijalne koristi za vas ili samo zanimljiv izazov u smislu programiranja. Dakle, ne ulazeći u vaše razloge, pogledajmo na koji način biste mogli da ostvarite ovu svoju ideju.

Bez sumnje najvažniji deo procesa programiranja mrežne aplikacije predstavlja pisanje programa koji će se izvršavati na različitim mrežnim sistemima i međusobno komunicirati kroz mrežu. Primera radi, u aplikaciji za Web postoje dva različita programa koji komuniciraju jedan sa drugim: program za Čitanje koji se izvršava na korisnikovom računaru (stonom, laptop, PDA, mobilnom telefonu) i serverska veb aplikacija koja se izvršava na računaru koji ima ulogu veb servera. Kao drugi ovakav primer mogla bi da nam posluži P2P razmena datoteka u kojoj, na svakom računaru koji učestvuje u ovom sistemu, postoji odgovarajući program. U ovom slučaju sasvim je moguće da su ti programi slični ili čak isti.

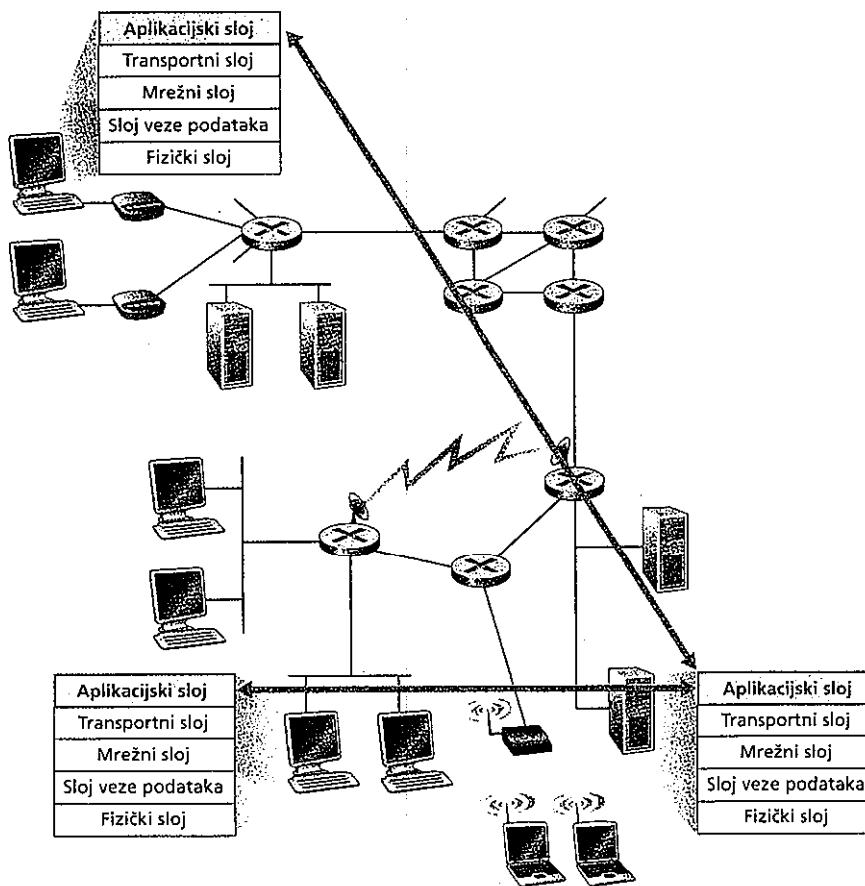
Dakle, programirajući svoju novu aplikaciju moraćete da napišete softver koji će se izvršavati na različitim računarima. U ovoj situaciji veoma je važno to što prilikom pisanja programa ne morate da razmišljate o njegovom radu na ključnim mrežnim uređajima, kao što su ruteri ili Ethernet komutatori. Čak i kada biste želeli da napišete softver za ove uređaje, to jednostavno ne bi bilo moguće. Kao što ste mogli da vidite u poglavlju 1 i posebno na slici 1.18, ključni mrežni uređaji ne funkcionišu na aplikacijskom sloju, već samo na nižim slojevima - na mrežnom i onima koji se nalaze ispod njega. Upravo ovo zadržavanje aplikativnog softvera samo na nivou krajnjih sistema (slika 2.1) zaslužno je za izuzetno brz razvoj široke lepeze aplikacija za Internet,

2.1.1 Arhitektura mrežnih aplikacija

Prilikom pravljenja nove mrežne aplikacije najpre morate da odlučite kakva će biti njena arhitektura. Staviš, samom programiranju uvek treba da prethodi globalni arhitektonski plan aplikacije. Uvek imajte u vidu to da se arhitektura aplikacije bitno razlikuje od mrežne arhitekture (podsećamo vas na petoslojnu arhitekturu Interneta o kojoj smo govorili u poglavlju 1). Iz perspektive programera aplikacije, mrežna arhitektura je fiksna i ona aplikacijama obezbeđuje konkretni skup usluga. **Arhitekturu aplikacije**, s druge strane, dizajnirao je njen programer i njome je defini-san način na koji se aplikacija organizuje na različitim krajnjim sistemima. Prilikom izbora arhitektonskog rešenja, programer se opredeljuje za jednu od tri dominantne arhitekture savremenih mrežnih aplikacija. To može da bude klijentsko-serverska arhitektura, zatim P2P arhitektura, ili hibridna arhitektura koja ima osobine obe koje su prethodno pomenute.

U klijentsko-serverskoj arhitekturi mora da postoji uvek dostupan *server*, računar koji opslužuje zahteve mnogo drugih računara koji se nazivaju *klijentima*. Klijentski računari mogu biti povremeno ili stalno uključeni. Klasičan primer ovakve arhitekture predstavlja veb aplikacija u kojoj neprekidno dostupni serveri opslužuju zahteve Čitača koji se izvršavaju na klijentskim računarama. Kada od nekog klijentskog računara primi zahtev za određeni objekat, veb server odmah odgovara slanjem traženog objekta. Skrećemo vam pažnju na to da u klijentsko-serverskoj arhitekturi ne postoji direktna komunikacija klijenata; recimo, u veb aplikaciji ne postoji direktna komunikacija dva Čitača. Drugu važnu karakteristiku ove arhitekture predstavlja činjenica da server ima fiksnu, dobro poznatu IP adresu (uskoro ćemo se pozabaviti ovim adresama). S obzirom na to da server ima nepromenljivu i dobro poznatu adresu i daje uvek dostupan, klijent u svakom trenutku može da kontaktira sa njim šaljući odgovarajući paket na ovu adresu. U poznatije klijentsko-serverske aplikacije spadaju Web, transfer datoteka, prijavljivanje na daljinu i elektronska pošta. Na slici 2.2(a) možete da vidite kako izgleda ova arhitektura.

U klijentsko-serverskim aplikacijama dešava se da server jednostavno ne može da odgovori na sve zahteve svojih klijenata. Na primer, na popularnim veb lokacijama za vesti mogući su zastoji u situacijama kada samo jedan server treba da odgo-



Slika 2.1 ♦ U mrežnim aplikacijama, komunikacija između krajnjih sistema odvija se kroz aplikacijski sloj.

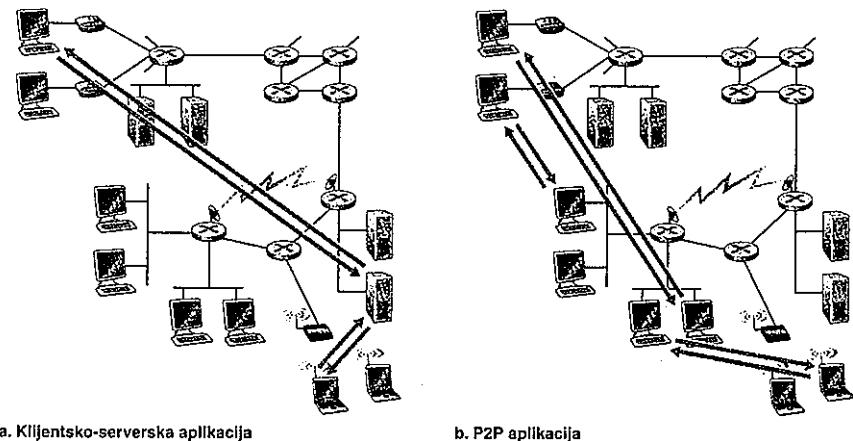
vori na sve zahteve klijentata. Upravo zato se, u klijentsko-serverskoj arhitekturi, koriste klasteri servera (za koje se nekada koristi i termin serverska **farma**) da bi se dobio moćan virtualni server.

U čistoj **P2P** arhitekturi u samom centru aplikacije ne postoji uvek dostupan server. Umesto toga, parovi ravnopravnih računara (*sng\peers*) direktno komuniciraju između sebe. Budući da se komunikacija ravnopravnih računara odvija bez posredovanja nekog servera, ova arhitektura se naziva arhitekturom ravnopravnih

korisnika (engl. *peer-to-peer*). U ovoj arhitekturi nijedan ravnopravni korisnik ne mora sve vreme da bude uključen; osim toga, moguće je da računar koji učestvuje u P2P razmeni datoteke svaki put ima drugu IP adresu. Dobar primer aplikacije sa Čistom P2P arhitekturom predstavlja Gnutella [Gnutella 2004], otvorena aplikacija za P2P razmenu datoteka. U ovoj aplikaciji svaki računar može da zahteva i šalje datoteke, upitom traži datoteke, odgovara na upite drugih računara i prosledjuje tude upite. O programu Gnutella nešto detaljnije ćemo govoriti u odeljku 2.6, a P2P arhitekturu smo ilustrovali na slici 2.2 (b).

Jedna od najznačajnijih dobrih strana P2P arhitekture jeste njena skalabilnost. Primera radi, kod ovakvih aplikacija moguće je da u razmeni datoteka učestvuju milioni ravnopravnih korisnika i da svaki od njih funkcioniše i kao server doprinoseći na taj način resursima zajednice. Prema tome, što generiše opterećenje zahtevajući određene datoteke, svaki ravnopravni računar takođe doprinosi sveukupnom obimu usluge odgovarajući na zahteve drugih korisnika. To znači daje P2P razmena datoteka suštinski podesive veličine - svaki korisnik povećava, ne samo potražuju, već i uslužni kapacitet. U današnjem Internetu P2P razmena datoteka čini veoma značajan deo sveukupnog saobraćaja [Saroiu 2002].

S druge strane, visok nivo distribuiranosti i decentralizovana priroda P2P aplikacija čini ih veoma teškim zaupravljanje. Ovde je sasvim moguće da se jedina kopija neke važne datoteke nalazi na računaru koji u svakom trenutku može da se isključi. Zbog toga je još uvek veliko pitanje da lije moguće napraviti industrijska P2P rešenja za enterprajz aplikacije [Adya 2004].



Slika 2.2 ♦ (a) Klijentsko-serverska arhitektura; (b) P2P arhitektura

Klijentsko-serverska i P2P su dve uobičajene arhitekture mrežnih aplikacija. Ipak, postoje i aplikacije koje su koncipirane kao **hibridi** ove dve arhitekture. Dobar primer ovakvih aplikacija predstavlja, sada već zaboravljeni, Napster koji je predstavljao prvu aplikaciju za razmenu MP3 muzike. Napster pripada P2P aplikacijama u tom smislu što korisnici direktno razmenjuju MP3 datoteke, bez posredovanja nekog servera koji bi sve vreme morao da bude dostupan. Međutim, Napster pripada i klijentsko-serverskim aplikacijama zato što ravnopravni korisnici ispitivanjem centralnog servera dolaze do informacija o MP3 pesmama koje traže. (Arhitekturu aplikacije Napster detaljnije smo prikazali u odeljku 2.6.) Druga aplikacija koja takođe ima hibridnu arhitekturu jeste instantna razmena poruka. Kod instantne razmene poruka interaktivni razgovor između dva korisnika je obično P2P; poruke koje oni razmenjuju ne prolaze kroz posrednički server koji je stalno dostupan. Međutim, kada Alisa pokrene svoju aplikaciju za instantnu razmenu poruka, ona se automatski registruje na centralnom serveru. Isto tako, kada Bob poželi da razgovara sa nekim sa svoje liste prijatelja njegov klijent za instantnu razmenu poruka na centralnom serveru odmah proverava koji su njegovi prijatelji trenutno dostupni.

2.1.2 Komunikacija između procesa

Pre nego što počnete da programirate svoju mrežnu aplikaciju potrebno je da, bar u najosnovnijim crtama, znate na koji način komuniciraju programi koji se nalaze na različitim krajnjim sistemima. U žargonu operativnih sistema ova komunikacija se, u stvari, ne odvija između programa, već između procesa. S druge strane, proces možete da zamislite i kao program koji se izvršava na krajnjem sistemu. Ukoliko se komunicirajući procesi izvršavaju na istom krajnjem sistemu, reč je o međuprocesnoj komunikaciji. Pravilima međuprocesne komunikacije upravlja operativni sistem krajnjeg sistema. Međutim, u ovoj knjizi nećemo se baviti komuniciranjem između procesa na istom računari, već načinima na koje komuniciraju procesi koji se izvršavaju na *različitim krajnjim sistemima* (to može da bude i u okviru različitih operativnih sistema).

Komunikacija između dva procesa koji se odvijaju na dva različita krajnja sistema odvija se razmenom poruka koje se šalju kroz računarsku mrežu. Predajni proces formira poruke i upućuje ih u mrežu; prijemni proces prima poruke i eventualno odgovara sopstvenim porukama. Na slici 2.1 ilustrovana je komunikacija procesa na aplikacijskom sloju petoslojne familije protokola.

Klijentski i serverski procesi

Mrežne aplikacije sastoje se od parova procesa koji jedni drugima šalju poruke kroz mrežu. Na primer, u veb aplikaciji, proces klijentskog Čitača razmenjuje poruke sa procesom veb servera. U sistemu P2P razmene, datoteke se prenose od procesa jednog ravnopravnog korisnika do procesa drugog ravnopravnog korisnika. U svakom paru komunicirajućih procesa jedan od njih naziva se **klijent**, a drugi **server**.

Kod Weba, čitač je klijentski proces, a veb server serverski proces. U P2P razmeni datoteka, ravnopravni korisnik koji preuzima datoteku naziva se klijent, dok predajni ravnopravni korisnik ima ulogu servera.

Verovatno ste primetili da u nekim aplikacijama, kao stoje, recimo, P2P razmena datoteka, isti proces može biti i klijent i server, Zaista, u sistemu za P2P razmenu datoteka isti proces može i da preuzima i da predaje datoteke. Ipak, u kontekstu komunikacione sesije između dva procesa uvek jedan od njih možemo da nazovemo klijentskim a drugi serverskim. Evo na koji način definišemo ova dva procesa.

U kontekstu komunikacione sesije između dva procesa, proces koji inicira komunikaciju (započinje kontakt sa drugim procesom na početku sesije) označava se kao klijent. Proses koji čeka na kontakt i tek nakon toga preduzima određenu akciju zove se server.

Na Webu, čitač inicira kontakt sa serverskim procesom; čitačev proces je klijentski, a proces sa veb servera je serverski. Kada u P2P razmeni datoteka ravnopravni korisnik A zatraži od korisnika B da mu pošalje neku konkretnu datoteku, on je tada klijent, dok korisnik B, u kontekstu date komunikacione sesije, ima ulogu servera. U situacijama u kojima nema sumnje u pogledu ove podele uloga nekada ćemo koristiti i terminologiju „klijentska i serverska strana aplikacije“. Na kraju poglavlja proći ćemo kroz jednostavan kod klijentske i serverske strane mrežne aplikacije.

Postoji još jedan terminološki aspekt koji je potrebno dublje razjasniti. U odeljku 2.1.1 arhitekturu mrežnih aplikacija smo podelili na klijentsko-serversku, P2P i hibridnu. Ova klasifikacija je korisna budući da nam pruža uopšteni okvir za planiranje arhitekture mrežnih aplikacija. Ipak, moramo imati u vidu i činjenicu da većina mrežnih aplikacija, uključujući tu i aplikacije sa P2P arhitekturom, u stvari obuhvata veći broj parova procesa koji su u međusobnoj komunikaciji. Međutim, u okviru jedne sesije u kojoj učestvuju dva procesa, jedan je uvek klijent, a drugi server.

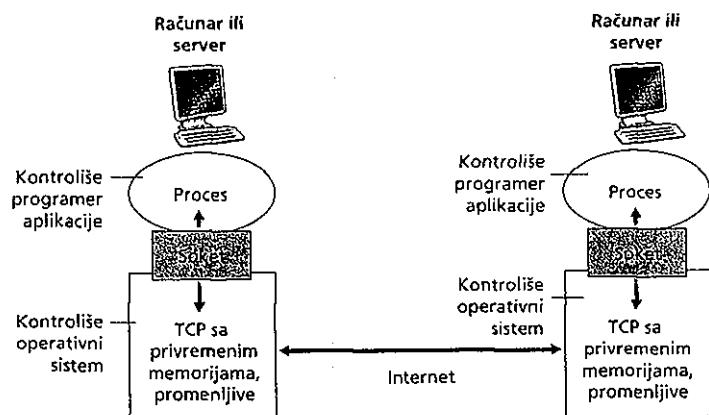
Soketi

Kao što smo već rekli, aplikacije u većini slučajeva imaju dva procesa na dva različita računara i ti procesi između sebe komuniciraju korišćenjem mreže. Procesi svoje poruke šalju u mrežu i primaju poruke iz nje preko soketa. Najjednostavnije je da soket procesa zamislite kao svojevrsna vrata dialog procesa, koji bi, opet, mogao da se poistoveti sa kućom. Kada neki proces želi da pošalje poruku drugom procesu, on tu poruku izbacuje u mrežu kroz svoja vrata (soket). Pri tom, on prepostavlja da s druge strane njegovih vrata postoji transportna infrastruktura koja će poruku preneti kroz Internet sve do vrata odredišnog procesa. Kada dospe do odredišnog računara, poruka prolazi kroz vrata primajućeg procesa (soket), koji zatim može da preduzme traženu akciju.

Na slici 2.3 ilustrovano je komuniciranje soketa dva procesa preko Interneta. (Na ovoj slici pretpostavlja se da se u osnovi ove komunikacije nalazi protokol TCP, iako to na Internetu može da bude i protokol UDP.) Kao što možete da vidite, soket je **interfejs** koji se nalazi između aplikacijskog i transportnog sloja računara. Naziva se još i **aplikacijski programski interfejs** (*Application Programming Interface, API*) i služi za posredovanje između aplikacije i mreže, budući da su upravo korišćenjem ovog programskog interfejsa napravljene mrežne aplikacije na Internetu. Programer aplikacije može da kontroliše sve ono što se događa na strani aplikacijskog sloja soketa, ali ne i ono što se događa sa njegove druge strane - u transportnom sloju. Jedino na šta programer može da utiče u transportnom sloju jesu 1) izbor transportnog protokola i 2) eventualno postavljanje nekoliko parametara transportnog sloja, kao što su maksimalan iznos privremene memorije ili maksimalna veličina segmenta. Onog trenutka kada se programer opredeli za određeni transportni protokol (ukoliko uopšte ima mogućnost izbora), on pravi aplikaciju imajući u vidu usluge transportnog sloja datog protokola. O soketima ćemo detaljnije govoriti u odeljcima 2.7 i 2.8.

Adresiranje procesa

Da bi jedan proces mogao da pošalje poruku procesu na drugom računaru, on najpre mora da prepozna taj prijemni proces. Za prepoznavanje prijemnog procesa obično



Slika 2.3 ♦ Prosesi i soketi aplikacije i transportni protokol

su potrebne dve informacije: 1) ime ili adresa računara i 2) identifikator kojim se u odredišnom računaru naznačava prijemni proces.

Zadržimo se najpre na adresi računara. U internetskim aplikacijama odredišni računar se identificuje svojom **IP adresom**. O IP adresama ćemo detaljnije govoriti u poglavljju 4. Zasad, dovoljno je da znate samo to da IP adresa predstavlja 32-bitni broj koji *jedinstveno* identificuje računar. (Najpreciznije je reći **da** ona jedinstveno identificuje mrežni interfejs kojim je računar povezan sa Internetom.) S obzirom na to da IP adresa svakog računara koji je povezan sa Internetom mora biti *globalno* jedinstvena, dodeljivanjem IP adresa upravlja se izuzetno pažljivo, o čemu ćemo takođe govoriti u poglavljju 4.

Osim poznavanja adrese računara kome se poruka upućuje, otpremna aplikacija mora da obezbedi i informacije koje će odredišnom računaru omogućiti da usmeri poruku ka pravom procesu. Na Internetu se u tu svrhu koriste **brojevi portova**. Popularnim protokolima aplikacijskog sloja dodeljeni su određeni brojevi portova. Primera radi, veb serverski procesi identificuju se brojem porta 80, dok se serverski procesi za elektronsku poštu (pod protokolom SMTP) identificuju brojem 25. Spisak poznatih brojeva portova za sve standardne Internet protokole nalazi se na adresi <http://www.iana.org>. Kada neki programer napravi neku novu mrežnu aplikaciju, ona mora da dobije* i novi broj porta. Više detalja o brojevima portova pronaći ćete u poglavljju 3.

2.1.3 Protokoli aplikacijskog sloja

Upravo ste videli da se međusobna komunikacija mrežnih procesa odvija slanjem poruka kroz sokete. Ali kakva je struktura ovih poruka? Koje je značenje njihovih pojedinih polja? Kada procesi šalju poruke? Sva ova pitanja dovode nas u oblast protokola aplikacijskog sloja. **Protokoli aplikacijskog sloja** definisu način razmene poruka između aplikacijskih procesa koji se izvršavaju na različitim računarima. Konkretnije, protokoli aplikacijskog sloja određuju:

- ◆ tipove poruka koje se razmenjuju (poruke zahteva ili poruke odgovora);
- ◆ sintaksu različitih tipova poruka, kao što su polja poruke i način njihovog interpretiranja;
- ◆ semantiku polja, odnosno, značenje informacija u njima;
- ◆ pravila za određivanje vremena i načina na koje proces šalje poruke i odgovara na njih.

Specifikacije nekih protokola aplikacijskog sloja navedenesu u RFC dokumentima. Što znači da se nalaze u javnom domenu. To je, primera radi, slučaj sa protokolom HTTP (*Hypertext Transfer Protocol* [RFC 2616]). Ukoliko se programer čitača pridržava pravila o protokolu HTTP iz RFC dokumenta, njegov čitač će moći da prima veb strane sa bilo kog veb servera koji je takođe napravljen u skladu sa tim pravilima.

Mnogi drugi protokoli aplikacijskog sloja vlasništvo su pojedinaca ili organizacija i namerno se ne nalaze u javnom domenu. Na primer, mnogi postojeći proizvodi za Internet telefoniju predstavljaju vlasničke protokole aplikacijskog sloja.

Veoma je važno napraviti razliku između mrežnih aplikacija i protokola aplikacijskog sloja. Protokol aplikacijskog sloja predstavlja samo jedan (ali izuzetno značajan) deo mrežne aplikacije. Pogledajmo nekoliko primera. Web je mrežna aplikacija koja omogućava korisnicima da, na svoj zahtev, preuzmu dokumente sa web servera. Web aplikacija se sastoji od velikog broja komponenti među kojima su standard za formatiranje dokumenata (HTML), Čitači vebova (Netscape Navigator ili Microsoft Internet Exp!orer), web serveri (za Apache, Microsoftovi i Netscape-ovi serveri) i protokol aplikacijskog sloja. Protokol Webovog aplikacijskog sloja, HTTP (*Hypertext Transfer Protocol*) [RFC 2616], definiše format i redosled poruka koje razmenjuju veb čitač i web server. Prema tome, protokol Webovog aplikacijskog sloja predstavlja samo jedan delić veb aplikacije. Kao drugi primer mogla bi da nam posluži aplikacija za elektronsku poštu. Internetska elektronska pošta takođe ima mnogo komponenti. To su odgovarajući serveri na kojima se nalaze poštanski sandučići, čitači elektronske pošte koji omogućavaju korisnicima da čitaju i prave poruke, standard za defmisanje strukture poruke i protokoli aplikacijskog sloja koji definišu način razmene poruka među serverima, i između servera i čitača, kao i način interpretacije određenih segmenata (na primer, zaglavlja) date poruke. SMTP (*Simple Mail Transfer Protocol*) je osnovni protokol aplikacijskog sloja za elektronsku poštu [RFC 2821]. Dakle, i kod elektronske pošte, protokol aplikacijskog sloja je samo jedna (ali veoma značajna) komponenta čitave aplikacije.

2.1.4 Koje su usluge potrebne aplikaciji?

Podsećamo vas da soket predstavlja interfejs između aplikacijskih procesa i transportnog protokola. Predajna aplikacija šalje poruke kroz ova vrata. Sa druge strane vrata, transportni protokol preuzima odgovornost za prenos poruke kroz mrežu, sve do istih takvih vrata prijemnog procesa.

U mnogim mrežama, a takav je slučaj i sa Internetom, postoji više transportnih protokola. Upravo zato je prilikom programiranja aplikacija neophodno izabrati jedan od raspoloživih mrežnih protokola. Na koji način se donosi ova odluka? Najčešće tako što se pažljivo prouče usluge koje obezbeđuju raspoloživi transportni protokoli i onda izabere ona koja najviše odgovara potrebama vaše aplikacije. Ova situacija podseća na izbor između voza i aviona za putovanje između dva grada. Svaki od ova dva načina transporta nudi različite usluge, a vi birate onaj koji vam više odgovara. (Na primer, pogodnost voza je što polazite iz centra i stižete u centar grada, dok je pogodnost aviona mnogo kraće trajanje transporta.)

Koje su usluge transportnog protokola potrebne jednoj aplikaciji? Uopšteno govoreći, zahtevi jedne aplikacije mogli bi da se svrstaju u tri kategorije: gubitak podataka, propusni opseg i vreme.

Pouzdani transfer podataka

Aplikacijama, kao što su elektronska pošta, instantna razmena poruka, transfer datoteka, daljinski pristup računaru, transfer vebovskih dokumenata i finansijskih aplikacija, potreban je absolutno pouzdan prenos podataka, dakle transfer bez ikakvog gubljenja podataka. Konkretno, gubitak podataka neke datoteke ili neke finansijske transakcije mogao bi da ima katastrofalne posledice (u drugom slučaju, po štedišu ili banku). Ostale aplikacije za koje se kaže da su **tolerantne na gubitak** - recimo, multimedijalne aplikacije za reprodukciju živog audio ili video materijala, ili snimljenih audio i video materijala, mogu da podnesu izvestan gubitak podataka. U ovim multimedijalnim aplikacijama posledica gubitka podataka može da bude samo mali poremećaj u reprodukciji, što ne mora da bude strašno. Uticaj ovakvog gubitka na kvalitetu aplikacije i tolerancija količine izgubljenih paketa zavisiće od aplikacije i primjenjenog načina kodiranja.

Propusni opseg

Efikasnost nekih aplikacija uslovljena je prenosom podataka određenom brzinom. Ukoliko, na primer, aplikacija za Internet telefoniju kodira glas u kvalitetu od 32 kb/s, onda je neophodno da se slanje podataka kroz mrežu i njihova isporuka odredišnoj aplikaciji odvijaju upravo ovom brzinom. Ukoliko nema na raspolaganju ovaj propusni opseg, aplikacija može ili da kodira drugom brzinom (i dobije dovoljno propusnog opsega za novu brzinu kodiranja) ili da odustane, budući da propusna moć upola manja od neophodne **aplikacijama zavisnim** od **propusne moći** jednostavno ništa ne znači. Danas mnoge multimedijalne aplikacije zavise od propusnog opsega, dok se za one buduće prepostavlja da će koristiti adaptivne tehnike kodiranja, koje će im omogućiti kodiranje sa onom brzinom bitova koja odgovara raspoloživom propusnom opsegu. Dok aplikacije zavisne od propusne moći zahtevaju tačno određeni propusni opseg, **elastične aplikacije** mogu da funkcionišu koristeći onoliko koliko im je dostupno. Elektronska pošta, transfer datoteka i veb transfer predstavljaju takve elastične aplikacije. Naravno, što više propusnog opsega, tim bolje! Ne kaže se uzalud da čovek ne može biti previše bogat, previše vitak i imati previše propusnog opsega!

Vreme

Poslednja kategorija usluga koje neophodne aplikacijama odnosi se na vreme. Efikasnost interaktivnih aplikacija u realnom vremenu poput Internet telefonije, virtuelnog okruženja, telekonferencija i igara za više igrača, direktno zavisi od veoma tačne isporuke podataka. Primera radi, mnoge od ovih aplikacija zahtevaju da ukupno kašnjenje od jednog do drugog kraja bude u okvirima od nekoliko stotina mili-sekundi ili čak kraće. (Pročitajte poglavlje 7 [Gauthier 1999; Ramjee 1994].) Duga kašnjenja u Internet telefoniji, recimo, stvaraju neprirodne pauze u komunikaciji; kod igara za više igrača ili virtualnih interaktivnih okruženja, duge pauze između akcije i reakcije okruženja (recimo, od igrača na drugom kraju veze) umanjuju realističnost date aplikacije. I kod aplikacija koje ne funkcionišu u realnom vremenu uvek je bolje daje kašnjenje što kraće, ali tu ne postoje nikakvi striktni zahtevi.

U tabeli sa slike 2.4 sumirane su potrebe u pogledu pouzdanosti, propusnog opsega i vremena za neke poznate i novije Internet aplikacije. Međutim, ovde su navedeni samo neki ključni zahtevi nekoliko popularnijih Internet aplikacija. Nije nam bio cilj ovde da ponudimo iscrpu klasifikaciju, već samo da identifikujemo neke od najznačajnijih kategorija mrežnih aplikacija.

2.1.5 Usluge koje obezbeđuju Internet transportni protokoli

Internet (i još uočenje, TCP/IP mreže) obezbeđuje aplikacijama dve transportne usluge - UDP (*User Datagram Protocol*) i TCP (*Transmission Control Protocol*). Prilikom pravljenja nove aplikacije za Internet, programer mora da se opredeli za jedan od ova dva protokola. Svaki od njih obezbeđuje nešto drugačije usluge aplikacijama koje ga koriste.

TCP usluge

U model TCP usluga spadaju usluga konekcije i usluga pouzdanog transfera podataka. Kada neka aplikacija pozove protokol TCP kao svoj transportni protokol, ona od njega dobija obe ove usluge.

◆ *Usluga konekcije:* Kod protokola TCP tok poruka aplikacijskog sloja počinje pre razmene poruka na aplikacijskom nivou između klijenta i servera. Ovaj postupak početne sinhronizacije upozorava klijenta i server i omogućava im da se pripreme za razmenu paketa. Po okončanju sinhronizacije kaže se daje uspostavljena TCP konekcija između soketa dva procesa. Reč je o punoj dupleksnoj vezi u kojoj oba procesa mogu istovremeno da šalju svoje poruke. Kada završi sa slanjem

Aplikacija	Gubitak podataka	Propusni opseg	Vremenska osjetljivost
Transfer datoteke	Nemo	Elastično	Ne
Elektronska pošta	Nemo	Elastično	Ne
Web dokumenti	Nemo	Elastično (nekoliko kb/s)	Ne
Audio i video zapisi u realnom vremenu	Tolerantne na gubitak	Audio: nekoliko kb/s do 1 Mb/s Video: 10 kb/s do 5 Mb/s	Da: nekoliko stotina milisekundi
Snimljeni audio i video zapisi	Tolerantne na gubitak	Isto kao prethodne	Da: nekolika sekundi
Interaktivne igre	Tolerantne na gubitak	Nekoliko kb/s do 10 kb/s	Da: nekoliko stotina milisekundi
Trenutna razmena poruka	Nemo	Elastična	Da i ne

Slika 2.4 ◆ Zahtevi odabralih mrežnih aplikacija

poruka, aplikacija mora da raskine konekciju. Ova usluga opisuje se kao usluga sa konekcijom a ne usluga „konekcije”, zato što su dva procesa povezana na veoma neobavezujući način. U poglavljiju 3 ćemo detaljno istražiti usluge konekcije i videti na koji način se one implementiraju.

◆ *Usluga pouzdanog transporta:* Procesi koji komuniciraju mogu slobodno da se oslonе na protokol TCP i da budu sigurni da će svi podaci biti isporučeni bez greške i pravilnim redosledom. Kada jedna strana aplikacije prosledi niz bajtova u soket, ona može da računa da će protokol TCP isporučiti isti taj niz prijemnom soketu, bez gubitaka ili duplikiranja bajtova.

U okviru protokola TCP postoji i mehanizam za kontrolu zagrušenja - usluga koja pre donosi direktnu korist samom Internetu nego komunicirajućim procesima. Ukoliko dođe do zagrušenja u mreži između klijenta i servera, ovaj mehanizam za kontrolu zagrušenja dozira intenzitet saobraćaja koji emituje (klijenta ili servera). Kao što ćete videti u poglavljju 3, mehanizam za kontrolu zagrušenja kod protokola TCP nastoji da ograniči svaku TCP konekciju na onaj deo ukupne propusne moći mreže za koji smatra daje fer.

Doziranje brzine prenosa može da ima veoma loše posledice po audio i video aplikacije u realnom vremenu zato što ove aplikacije imaju ograničenje po pitanju minimalne brzine prenosa. Primjer, aplikacije ove vrste su tolerantne na gubitak podataka i nije im potrebna potpuno pouzdana transportna usluga. Iz tih razloga, programeri aplikacija u realnom vremenu obično daju prednost protokolu UDP u odnosu na TCP.

Budući da smo skicirali usluge koje pruža protokol TCP, recimo nekoliko reci i o uslugama koje on ne pruža. Najpre, protokol TCP ne pruža garancije u pogledu minimalne brzine transporta. Konkretno, emitujućem procesu nije dozvoljeno da to čini onom brzinom kojom želi, već kontrola zagrušenja reguliše brzinu slanja kod protokola TCP. Ona može da prudi otpremnu stranu da to čini veoma malom prosečnom brzinom. Zatim, protokol TCP ne pruža nikakve garancije ni u pogledu kašnjenja paketa. Kada odašiljući proces prosledi svoje podatke TCP soketu, on može da bude siguran da će ti podaci stići do svog odredišta, ali protokol TCP pri tom ne precizira apsolutno ništa u pogledu trajanja tog puta. Kao što su mnogi od nas iskusili na „World Wide Waitu”, nekada je potrebno čekati desetine sekundi, pa čak i čitave minute da TCP isporuči poruku (sa, na primer, HTML datotekom) od veb servera do veb klijenta. Dakle, protokol TCP garantuje isporuku svih podataka, ali ne pruža nikakve garancije u pogledu brzine isporuke kao ni u pogledu eventualnih kašnjenja.

UDP usluge

Protokol UDP je pojednostavljeni transportni protokol sa minimalističkim modelom usluge. To je protokol bez konekcije, kod koga sinhronizacija ne prethodi početku komunikacije između dva procesa. UDP obezbeđuje nepouzdan transfer podataka, što znači da nakon predaje paketa UDP soketu, ne postoje garancije da će poruka zaista stići do prijemnog procesa. Osim toga, moguće je da u porukama koje stignu do odredišta bude poremećen redosled.

Kod protokola UDP ne postoji mehanizam za kontrolu zagušenja, tako da emi-tujući proces može da ubacuje podatke u UDP soket onom brzinom kojom želi (iako uvek postoji mogućnost da deo tih podataka ne stigne do odredišta). S obzirom na to da aplikacije u realnom vremenu obično mogu da tolerišu određeni gubitak podataka, ali istovremeno zahtevaju određenu minimalnu brzinu prenosa, njihovi programeri najčešće daju prednost protokolu UDP. Poput protokola TCP, ovaj protokol ne pruža nikakve garancije u pogledu kašnjenja.

Na slici 2.5 navedeni su transportni protokoli koje koriste neke popularne Internet aplikacije. Vidimo da aplikacije za elektronsku poštu, pristup udaljenim terminalima, Web i transfer datoteka koriste TCP. Programeri ovih aplikacija izabrali su ovaj protokol prvenstveno zato što on obezbeđuje pouzdanu transportnu uslugu, garantujući da će svi podaci stići do svog odredišta. Isto tako, vidimo da aplikacije za Internet telefoniju obično koriste UDP. Obe strane ove aplikacije moraju da šalju podatke kroz mrežu brzinom koja je veća od neke minimalne brzine (slika 2.4), a to je pre moguće uz upotrebu protokola UDP nego uz protokol TCP.

Kao što smo ranije napomenuli, ni jedan od ova dva protokola ne pruža nikakve garancije u pogledu trajanja transporta. Da li to znači da vremenski osetljive aplikacije ne mogu da se izvršavaju preko Interneta? Naravno da mogu - već godinama se veliki broj ovakvih aplikacija izvršava pod okriljem Interneta. Ove aplikacije funkcionišu relativno dobro zato što su napravljene tako da se nose (koliko god je to moguće) sa ovim nepostojanjem garancija. U poglavljiju 7 ćemo vam prikazati nekoliko takvih programerskih trikova. Ipak, ni veoma pametan dizajn ne može ništa da učini kod izuzetno dugotrajnih kašnjenja koja pogađaju javni Internet.

Dakle, današnji Internet može da obezbedi zadovoljavajuće usluge vremenski osetljivim aplikacijama, ali ne i garancije u pogledu vremena i propusne moći. U poglavljju 7 govorimo i o nekim novim uslugama, uključujući upravo uslugu sa garantovanim kašnjenjem koja je neophodna vremenski osetljivim aplikacijama.

Aplikacija	Protokol aplikativnog sloja	Transportni protokol koji se nalazi u osnovi
Elektronska pošta	SMTP [RFC 2821]	TCP
Daljinski pristup terminalu	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transfer datoteke	FTP [RFC 959]	TCP
Daljinski server datoteke	NFS [McKusik 1996]	UDP ili TCP
Protokol multimedijalnog sadržaja u realnom vremenu	Često vlasnički (npr. Real Networks)	UDP ili TCP
Internet telefonija	Često vlasnički (npr. Net2phone)	Obično UDP

Slika 2.5 ♦ Popularne Internet aplikacije, njihovi protokoli aplikativnog sloja i protokoli transportnog sloja koji se nalaze ispod njih.

2.1.6 Mrežne aplikacije o kojima ćemo govoriti u ovoj knjizi Nove Internet aplikacije pojavljuju se praktično svakodnevno - neke od njih su u javnom domenu, dok su druge vlasničke. Umesto da vam, poput neke enciklopedije, prikažemo veliki broj Internet aplikacija, mi ćemo se usredosrediti na manji broj važnih i popularnih aplikacija. U ovom poglavljiju detaljno ćemo vam prikazati pet veoma popularnih aplikacija - Web, transfer datoteka, elektronsku poštu, usluge direktorijuma i P2P razmenu datoteka. Počećemo od Weba - ne samo zato što je u pitanju enormno popularna aplikacija, već i zato stoje njen protokol aplikacijskog sloja, HTTP, relativno jednostavan i može da posluži kao dobra ilustracija mnogih ključnih principa kod mrežnih protokola. Zatim ćemo preći na transfer datoteka koji predstavlja dobar kontrast protokolu HTTP i takođe omogućava rasvjetljavanje još nekoliko principa. Nakon toga, govorimo o elektronskoj pošti, prvoj izuzetno popularnoj Internet aplikaciji. Videćete da savremene aplikacije za elektronsku poštu ne koriste jedan protokol aplikacijskog sloja, već nekoliko njih. Govorimo

o sistemu imena domena (*Domain Name System, DNS*) koji obezbeđuje uslugu direktorijuma za Internet. Mnogi korisnici nemaju nikakav direktni kontakt sa sistemom imena domena jer se ova usluga poziva indirektno, preko aplikacija (uključujući Web, transfer datoteka i elektronsku poštu). Usluga DNS veoma lepo ilustruje način implementiranja distribuirane baze podataka na Internetu. Konačno, govorimo i o P2P razmeni datoteka koja, po nekim merenjima (recimo, po obimu mrežnog saobraćaja) spada u grupu popularnih programa na Internetu.

2.2 Web i HTTP

Do 1990-ih Internet su za privajljivanje na udaljene računare, transfer datoteka sa lokalnih na udaljene računare i obratno, prijem i slanje vesti i elektronske pošte, koristili prvenstveno istraživači, akademski krugovi i studenti, lako su ove aplikacije bile (i još uvek su) veoma korisne, izvan ovih akademskih i istraživačkih zajednica Internet je bio praktično nepoznat. Međutim, ranih 1990-ih godina pojavila se nova aplikacija - World Wide Web [Bemers-Lee 1994]. Web predstavlja prvu Internet aplikaciju koja je privukla široku javnost. Ova aplikacija je dramatično izmenila način interakcije ljudi unutar njihovih radnih okruženja i izvan njih. Web je podigao Internet od samo jedne od mnogih mreža za razmenu podataka (poput onlajn mreža, kao što su Prodigy, America Online i CompuServe, nacionalne francuske mreže Minitel/ Transpac, privatne mreže X.25 i Frame Relay mreža) do pozicije nepriskosnovene i najveće mreže za razmenu podataka.

Kroz istoriju ima mnogo primera elektronskih i komunikacionih tehnologija koje su imale krunpe socijalne posledice. Prvu ovaku tehnologiju predstavlja je telefon koji je izmišljen 1870-ih. Telefon je prvi put omogućio dvema osobama koje se nalaze na različitim lokacijama da komuniciraju u realnom vremenu. Sledеću značajnu tehnologiju za elektronsku komunikaciju predstavljalo je emitovanje radio

televizijskih signala 1920-ih i 1930-ih. Radio i televizijske emisije su omogućile ljudima da primaju ogromne količine zvučnih i video informacija. Sledеća komuni-

kaciona tehnologija koja je bitno izmenila život i rad velikog broja ljudi bio je Internet. Dve njegove najpopularnije aplikacije - elektronska pošta i Web - iz korena su izmenile naše živote. O elektronskoj pošti govorimo u odeljku 2.4, a o Webu i njegovim protokolima aplikacijskog sloja već u ovom.

Većini ljudi najviše se sviđa to što Web funkcioniše *na zahtev*. To znači da korisnici primaju one informacije koje žele i onda kada to žele. Ovo se razlikuje od radio i televizijskih emisija u kojima se od korisnika traži da se „uključe“ onda kada stanicu određene sadržaje stavi na raspolaganje. Pored toga što funkcioniše na zahtev, Web ima i mnoge druge izuzetne funkcije koje su ga učinile veoma popularnim. Svaki pojedinac na krajnje jednostavan način može da ponudi neku informaciju putem Weba; dakle, svi mogu da postanu izdavači uz minimalne izdatke. Zatim, hiperveze i pretraživači olakšavaju navigaciju kroz pravi okean veb lokacija. Grafički elementi stimulišu naša ēula. Obrasci, jezik JavaScript, Java appleti i Active X, kao i mnoge druge komponente, omogućavaju interakciju sa veb stranicama i veb lokacijama. Dalje, Web obezbeđuje interfejs za pristup ogromnim količinama zvučnog i video materijala koji se čuva na Internetu. Ovim i drugim multimedijalnim sadržajima pristupa se takođe na zahtev.

2.2.1 Prikaz protokola HTTP

U samom srcu Weba nalazi se protokol **HTTP (HyperText Transfer Protocol)** - protokol njegovog aplikacijskog sloja. HTTP je defmisan dokumentima [RFC 1945] i [RFC 2616], a implementira se u dva programa - klijentskom i serverskom. Klijentski i serverski programi, koji se izvršavaju na dva različita računara, komuniciraju razmenom HTTP poruka. Protokol HTTP definije strukturu tih poruka i načina na koji ih klijent i server razmenjuju. Pre nego što detaljno objasnimo protokol HTTP, ne bi bilo loše da najpre usvojimo vebovsku terminologiju.

Veb strana (ili dokument) sastoji se od objekata. Objekat je jednostavno datoteka - HTML datoteka, JPEG slika, GIF slika, Java aplet, zvučna datoteka - koju je moguće adresirati jednim uniformnim lokatorom resursa (*Uniform Resource Locator, URL*). Veb stranice se u većini slučajeva sastoje od osnovne HTML datoteke i nekoliko referenciranih objekata. Ukoliko, primera radi, veb strana sadrži HTML tekst i pet JPEG slike, onda ona ima šest objekata: osnovnu HTML datoteku i pet slika. Osnovna HTML datoteka referencira ostale objekte na strani putem njihovih URL-ova. Svaki URL ima dve komponente: ime servera na kome se dati objekat nalazi i putanja na kojoj se on nalazi na tom serveru. Na primer, u URL-u

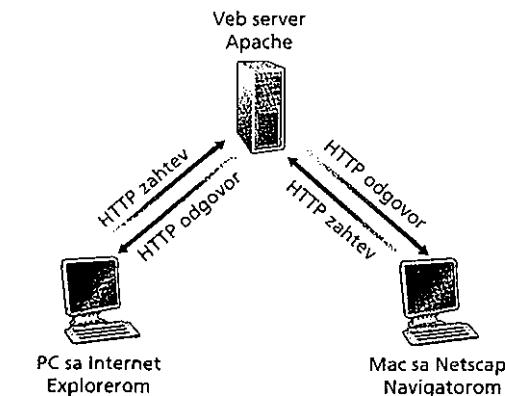
www.someSchool.edu/someDepartment/picture.gif

deo www . someSchool. edu predstavlja ime računara, dok je /someDepartment/picture . gif putanja do objekta. Čitač je korisnički agent za Web; on prikazuje tražene veb strane i obezbeđuje razne navigacione i konfiguracione funkcije. Pored toga, veb Čitači implementiraju i klijentsku stranu protokola HTTP. Prema tome, u kontekstu Weba ravnopravno ćemo koristiti termine *Čitač* i *klijent*. U popularne Čitače Weba spadaju Netscape Communicator i Microsoftov Internet Explorer. Veb server je računar u kome se nalaze veb objekti koji se adresiraju

navođenjem URL-a. Osim toga, veb serveri implementiraju serversku stranu protokola HTTP. U popularne servere spadaju Apache i Microsoftov Internet Information Server. Netcraft je dobar izvor gde ćete pronaći lepo istraživanje posvećeno veb ser-verima [Netcraft 2004].

Protokol **HTTP** definije način na koji veb klijenti (recimo, Čitači) traže veb strane od veb servera, kao i način na koji veb serveri šalju tražene strane klijentima. Kasnije ćemo posvetiti više pažnje interakciji između klijenta i servera; sada ćemo o tome dati samo nekoliko najosnovnijih informacija (slika 2.6). Kada korisnik zatraži neku veb stranicu (mišem izabere hipervezu), čitač šalje serveru HTTP poruke zahtevajući objekte sa date stranice. Server prima ove zahteve i odgovara HTTP porukama u kojima se nalaze traženi objekti. Do 1997. godine praktično svi čitači i serveri implementirali su protokol HTTP/1.0 - verziju koja je definisana dokumentom RFC 1945. Od 1998. godine, neki serveri i čitači počeli su da implementiraju HTTP/1.1 - verziju koja je definisana u dokumentu RFC 2616. Protokol HTTP/1.1 je kompatibilan unazad sa verzijom HTTP/1.0, što znači da veb server pod verzijom 1.1 može da komunicira sa čitačem pod verzijom 1.0, a Čitač pod verzijom 1.1 može da komunicira sa serverom pod verzijom 1.0. S obzirom na to da je protokol HTTP/1.1 sada dominantan, u ovoj knjizi ćemo pod protokolom HTTP uvek misliti upravo na ovu, noviju verziju.

Protokol HTTP kao osnovni transportni protokol koristi TCP (koji se ne izvršava povrh protokola UDP). Evo kako ova komunikacija izgleda. HTTP klijent najpre treba da inicira TCP konekciju sa serverom. Nakon uspostavljanja konekcije, procesi čitača i klijenata pristupaju TCP-u kroz svoje sokete. Kao što smo rekli u odeljku 2.1, na klijentskoj strani soket predstavlja vrata između klijentskog procesa i TCP konekcije; na serverskoj strani to su vrata između serverskog procesa i TCP konekcije,



Slika 2.6 ♦ Razmena HTTP poruka zahteva i odgovora

Klijent šalje svoje HTTP poruke sa zabtevima preko svog soketa i preko njega prima sve HTTP odgovore. Slično tome, HTTP server iz svog soketa prima poruke i svoje odgovore ponovo šalje kroz soket. Onog trenutka kada klijent pošalje svoju poruku ka svom soketu, ona više nije u njegovim rukama, već u rukama protokola TCP. Podsećamo vas na odeljak 2.1 u kome smo rekli da protokol TCP obezbeđuje pouzdano uslugu transfera podataka protokolu HTTP. To znači da svaka poruka sa HTTP zahtevom koju emituje neki klijent mora neizmenjena da stigne do servera. Isto tako, svaka poruka sa HTTP odgovorom koju server pošalje mora neizmenjena da stigne do klijenta. I upravo ovde naziremo jednu od velikih prednosti slojevitih arhitektura - protokol HTTP uopšte ne mora da razmišlja o gubitku podataka i načinima na koje ih protokol TCP obnavlja ili ispravlja greške u vezi sa njihovim redosleđem. To je posao protokola TCP i protokola u nižim slojevima familije protokola.

Veoma je važno naglasiti da server šalje tražene datoteke klijentu bez ikakvog evidentiranja informacija o statusu klijenta. Ukoliko neki klijent u roku od nekoliko sekundi dvaput zatraži isti objekat, server neće odgovoriti obaveštenjem da gaje upravo uslužio. On će traženi objekat poslati ponovo, nesvestan toga daje to već jednom učinio. S obzirom na to da HTTP server ne održava nikakve informacije o klijentima, za ovaj protokol se kaže daje protokol **bez nadgledanja stanja**.

Na ovom mestu želimo da vas podsetimo na odeljak 2.1 u kojem smo rekli da Web koristi klijentsko-serversku arhitekturu. Veb server je uvek dostupan, ima nepromenljivu IP adresu i svojim uslugama odgovara na zahteve nekada i miliona različitih čitača.

2.2.2 Nepostojane i postojane konekcije

Protokol HTTP može da koristi i nepostojane i postojane konekcije. Iako ovaj protokol u svom podrazumevanom režimu koristi postojane veze, HTTP klijenti i serveri mogu da se konfigurišu i tako da koriste nepostojane konekcije. U ovom odeljku ispitaćemo i jednu i drugu vrstu konekcija.

Nepostojane konekcije

Prodimo sada kroz postupak prenošenja veb stranice od servera do klijenta putem nepostojanih konekcija. Pretpostavimo da se data stranica sastoji od osnovne HTML datoteke i deset JPEG slika i da se svih 11 objekata nalazi na istom serveru. Pretpostavljemo i daje URL osnovne HTML datoteke

www.someSchool.edu/someDepartment/home.index

Evo šta se dešava:

1. HTTP klijent inicira TCP konekciju sa serverom www. someSchool. edu na portu 80 stoje podrazumevani broj porta za protokol HTTP.
2. HTTP klijent šalje serveru poruku sa HTTP zahtevom preko soketa koji je povezan sa TCP konekcijom koja je uspostavljena u prvom koraku. U okviru ove poruke nalazi se i ime putanje /someDepartment/home . index. (Uskoro ćemo nešto detaljnije govoriti o HTTP porukama.)

4. HTTP server prima poruku sa zahtevom preko soketa i njemu pridružene konekcije, učitava objekat /someDepartment/home. index iz svog prostora za skladištenje (RAM memorija ili disk) i zatim ga enkapsulira u poruku sa HTTP odgovorom koju, preko svog soketa, šalje klijentu.
5. HTTP server potom nalaže protokol TCP da zatvori TCP konekciju. (Protokol TCP ovu konekciju, međutim, ne prekida sve dok se sasvim ne uveri u to daje poruka neizmenjena stigla do klijenta.)
6. HTTP klijent prima poruku sa odgovorom i TCP konekcija se prekida. U poruci je naznačeno da je enkapsulirani objekat HTML datoteka. Klijent izvlači traženu HTML datoteku iz poruke, zatim je ispituje i pronađe reference do deset JPEG objekata.
7. Potom se za svaki od referenciranih JPEG objekata ponavljaju prva četiri koraka ovog postupka.

Čitač prikazuje korisniku primljenu veb stranicu. Uvek postoji mogućnost da dva različita čitača interpretiraju (odnosno, prikazuju) veb stranicu na dva različita načina. To, međutim, nema nikakve veze sa protokolom HTTP koji ne definiše način na koji čitač prikazuje veb strane. HTTP specifikacijama ([RFC 1945] i [RFC 2616]) definisan je samo komunikacioni protokol između klijentskog HTTP programa i ser-verskog HTTP programa.

U prethodnom postupku korišćene su nepostojane veze budući da je svaka TCP konekcija zatvarana nakon što je server poslao objekat - ona ne traje (ne postoji) za ostale objekte. Skrećemo vam pažnju na to da svaka TCP konekcija prenosi tačno jednu poruku sa zahtevom i jednu poruku sa odgovorom. Prema tome, u ovom primeru, od trenutka kada je korisnik zatražio veb stranu uspostavljeno je 11 TCP konekcija.

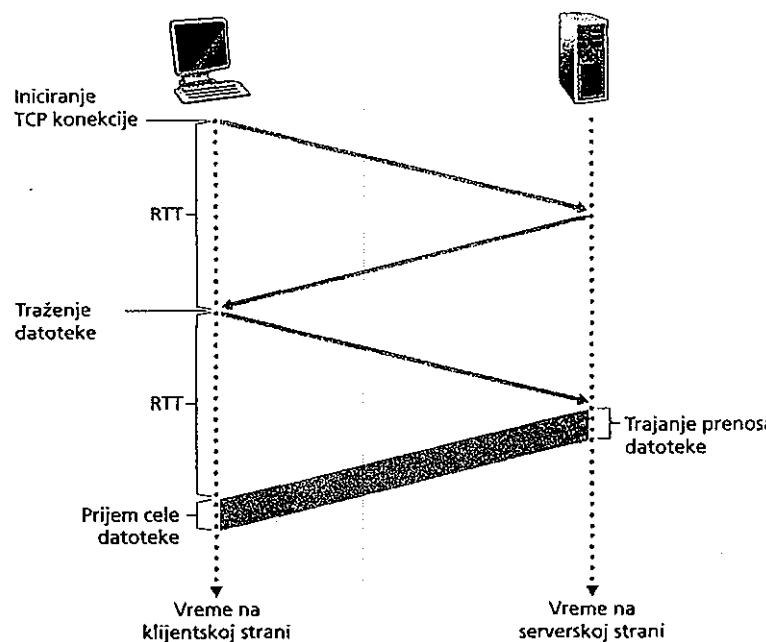
U koracima koje smo upravo naveli namerno smo bili nedovoljno konkretni u pogledu toga da li je deset JPEG slika klijent dobio putem deset serijskih TCP konekcija, ili su neke od njih dobijene putem paralelnih konekcija. U stvari, korisnici savremenih čitača mogu različitim načinima konfigurisanja da kontrolišu stepen paralelizma. U svojim podrazumevanim režimima većina čitača otvara pet do deset paralelnih TCP konekcija i svakom od njih odvija se jedna transakcija tipa zahtev-odgovor. Ukoliko korisnik to želi, on može da ograniči broj paralelnih konekcija na samo jednu i u tom slučaju u ovom primeru bi bilo uspostavljeno deset serijskih konekcija. Kao što ćete videti u sledećem poglavljju, korišćenje paralelnih konekcija ubrzava odziv.

Pre nego što nastavimo, pokušaćemo da procenimo koliko vremena prode od trenutka kada klijent zatraži osnovnu HTML datoteku pa do trenutka kada se prenese cela tražena datoteka na stranu klijenta. Za potrebe ovog računa definisatićemo vreme **povratnog puta** (*round-trip time, RTT*) kao vreme koje je potrebno malom paketu za put od klijenta do servera i natrag. RTT vreme obuhvata kašnjenja usled prostiranja paketa, zatim kašnjenja usled stajanja u redu u posredujućim ruterima i komulatorima, kao i kašnjenja usled obrade paketa (o ovim kašnjenjima govorili smo u odeljku 1.6). Zamislite sada šta se dešava kada korisnik izabere neku hiper-

vezu. Kao što možete da vidite na slici 2.7, čitač odmah inicira TCP konekciju sa web serverom. Ovo uspostavljanje konekcije obuhvata „trodelnu sinhronizaciju“ - klijent šalje mali TCP segment serveru, server potvrđuje i odgovara malim TCP segmentom i, konačno, klijent serveru šalje potvrdu o prijemu. Jedan RTT ciklus ističe nakon završetka prva dva dela sinhronizacije. Nakon završetka prva dva dela sinhronizacije, klijent kombinuje HTTP poruku (koja sadrži zahtev) sa trećim delom trodelne sinhronizacije (sa potvrdom) i šalje ih istom TCP konekcijom. Kada poruka sa zahtevom stigne do servera, on šalje HTML datoteku TCP konekcijom. Ovom HTTP komunikacijom tipa zahtev/odgovor zatvara se još jedan RTT ciklus. Dakle, uopšteno govoreći, ukupno vreme odziva dobija se sabiranjem dva RTT ciklusa i trajanja prenosa HTML datoteke od servera do klijenta.

Postojane veze

Nepostojane veze imaju svoje nedostatke. Najpre, *za svaki traženi objekat* mora da se uspostavi i održava nova konekcija. Svakoj od ovih konekcija dodeljuje se privremena TCP memorija, a uz to, na klijentu i serveru moraju da se održavaju parametri date konekcije. Sve ovo može u značajnoj meri da optereti web server koji nekada



Slika 2.7 ♦ Elementi za proračun vremena od zahteva do prijema datoteke

mora istovremeno da usluži zahteve nekoliko stotina različitih klijenata. Zatim, kao što smo upravo opisali, svaki objekat ima kašnjenje isporuke od dva RTT ciklusa - jedan se potroši za uspostavljanje TCP konekcije, a drugi prilikom traženja i prijema objekta.

Kod postojanih konekcija, server nakon slanja svog odgovora klijentu ostavlja TCP konekciju otvorenom i njome mogu da se pošalju i svi naknadni zahtevi i odgovori koje razmene klijent i server. Konkretno, to znači da istom, postojanom TCP konekcijom može da se pošalje čitava web stranica - u našem prethodnom primeru to su osnovna HTML datoteka i svih deset slika. Štaviše, istom postojanom TCP konekcijom klijentu može da se pošalje i više web stranica ukoliko se one nalaze na istom serveru. Obično HTTP server prekida ovaku vezu nakon isteka određenog vremena njenog nekoriscenja (ovaj vremenski interval može da se podeši).

Postoje dve verzije postojanih konekcija - **konekcije bez cevododne obrade** i veze sa **cevododnom obradom**. U verziji bez cevododne obrade klijent može da izda novi zahtev tek nakon prijema odgovora na prethodni zahtev. U ovom slučaju klijent se suočava sa jednim RTT ciklусom za traženje i prijem svakog referencira-nog objekta (deset slika u prethodnom primeru). Iako ovo predstavlja poboljšanje u odnosu na nepostojane veze u kojima su za svaki pojedinačni objekat potrebna dva RTT ciklusa, RTT kašnjenje može još više da se skratiti cevododnom obradom. Druga manja konekcija bez cevododne obrade jeste to što nakon slanja objekta sa servera, postojana konekcija miruje - ne radi baš ništa - čekajući dolazak novog zahteva. U ovom stanju mirovanja ne koriste se na optimalan način serverski resursi.

U svom podrazumevanom režimu protokol HTTP/1.1 koristi postojane cevododne veze. Kod cevododnih konekcija HTTP klijent izdaje zahtev čim nađe na neku referencu. To znači da on može da izda uzastopne zahteve za referencirane objekte - dnjim recima, može da pošalje novi zahtev pre nego što dobije potvrdu o prijemu prethodnog. Kada primi uzastopne zahteve, server na isti način (uzastopno) šalje tražene objekte. Kod cevododne obrade moguće je da samo jedan RTT ciklus bude potreban za prijem svih referenciranih objekata (kod konekcija bez cevododne obrade potreban je po jedan RTT ciklus za svaki referencirani objekat). Konačno, cevododna TCP konekcija se i kraće nalazi u stanju mirovanja. U domaćim zadacima ovog i narednog poglavlja izvršićemo i kvantitativno poređenje performansi nepostojanih i postojanih konekcija. Čitaoce koje ova tema posebno zanima upućujemo na dela [Heidemann 1997; Nielsen 1997].

2.2.3 Format HTTP poruke

Specifikacije protokola HTTP [RFC 2616] obuhvataju i definicije formata HTTP poruka. Postoje dve vrste HTTP poruka - poruke sa zahtevima i poruke sa odgovorima i o obema ćemo govoriti u odeljcima koji slede.

HTTP poruka sa zahtevom

Evo kako izgleda uobičajena HTTP poruka sa zahtevom:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language:fr
```

Posmatrajući ovu jednostavnu poruku sa zahtevom mogli bismo mnogo toga da naučimo. Pre svega, ona je napisana u standardnom ASCII tekstu, tako da i korisnik koji je prosečno računarski pismen može da je pročita. Zatim, ona se sastoji od pet redova iza kojih dolazi znak za povratak na početak reda i znak za novi red. Iako ova poruka sa zahtevom ima pet redova, poruke ove vrste mogu imati i mnogo više redova, ali i samo jedan red. Prvi red HTTP poruke sa zahtevom naziva se red zahteva; svi ostali redovi su redovi zaglavlja. Red zahteva ima tri polja - polje za metod, URL polje i polje sa HTTP verzijom. U polju za metod može da se nade nekoliko različitih metoda, kao što su, na primer, **GET**, **POST** i **HEAD**. Velika većina HTTP poruka sa zahtevom koristi metod **GET**. Ovaj metod se koristi kada čitač traži objekat koji je identifikovan u URL polju. U ovom primeru čitač od servera zahteva objekat `/somedir/page.html`. Što se polja sa verzijom tiče, ono samo sebe dovoljno dobro objašnjava - čitač u ovom primeru implementira protokol HTTP/1.1.

Pogledajmo sada redove zaglavlja našeg primera. Linijom zaglavlja `Host: www.someschool.edu` navodi se računar na kome se traženi objekat nalazi. Možda vam se čini da ovaj red zaglavlja nije neophodan, budući daje već uspostavljena TCP konekcija sa računaram. Međutim, kao što ćete videti u odeljku 2.2.6 informacija o zaglavljima neophodna je veb keš serverima. Redom `Connection: close` čitač dalje govori serveru da ne želi da koristi postojane veze, već da se veza prekida čim server pošalje traženi objekat. Dakle, i pored toga što implementira protokol HTTP/1.1, čitač ne želi postojanu vezu. Redom zaglavlja `User-agent: Mozilla/4.0` navodi se korisnički agent, odnosno, vrsta čitača koji je poslao zahtev. U ovom slučaju to je program Mozilla/4.0, Netscapeov čitač. Ovaj red zaglavlja je veoma koristan zato što omogućava serveru da različitim tipovima korisničkih agenata pošalje različite verzije istog objekta. (Svaka verzija adresirana je istim URL-om.) Konačno, zaglavljem `Accept-language: fr` navodi se da bi korisnik voleo da primi francusku verziju objekta, ukoliko takva postoji na serveru; u suprotnom, server će da pošalje podrazumevanu verziju traženog objekta. Zaglavljem `Accept-language: fr` je samo jedno od mnogih kojima se u okviru protokola HTTP pregovara o sadržaju.

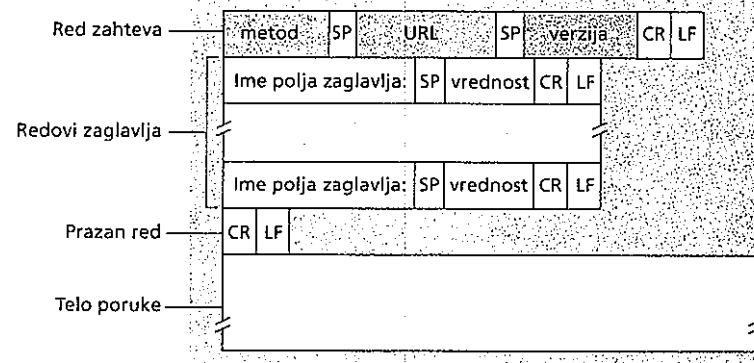
Pošto smo završili sa primerom, pogledajmo kako izgleda opšti format zahteva (slika 2.8). Već na prvi pogled vidi se da opšti format striktno prati naš prethodni primer. Ipak, možda ste primetili da nakon redova zaglavlja (i dodatnih znakova za prelazak u novi red i povratak na početak reda) sledi „telo poruke“. Ono je, doduše, prazno kod metoda **GET**, ali se koristi kod metoda **POST**. HTTP klijent često koristi metod **POST** kada korisnik ispunjava neki obrazac - primera radi, kada u odgovara-

juće polje pretraživača upisuje ključne reči. Kod poruke tipa **POST** korisnik još uvek od veb servera traži određenu veb stranicu, ali će njen konkretni sadržaj zavisiti od onoga stoje korisnik upisao u poljima obrasca. Ukoliko je kao metod izabran **POST**, onda se u telu poruke nalazi ono što je korisnik upisao u poljima obrasca.

Bili bismo zbilja nemarni kada ne bismo rekli i to da zahtev koji je generisan obrascem ne mora uvek da koristi metod **POST**. HTML obrasci često koriste i metod **GET** i tako prenose unete podatke (u polja obrasca) do datog URL-a. Na primer, ako obrazac koristi metod **GET**, ima dva polja u koja je upisano `monkeys` i `bananas`, onda će URL imati strukturu `www.somesite.com/animalsearch?monkey=ys&bananas`. U vašem svakodnevno krstarenju Webom verovatno ste mnogo puta videli ovakav URL.

Metod **HEAD** veoma je sličan metodu **GET**. Kada primi zahtev sa metodom **HEAD**, server odgovara HTTP porukom, ali izostavlja traženi objekat. Programeri aplikacija često koriste ovaj metod prilikom identifikovanja i uklanjanja problema.

Verzija HTTP/1.0 dozvoljava korišćenje samo tri metoda - **GET**, **POST** i **HEAD**. Nasuprot tome, specifikacija HTTP/1.1 dozvoljava korišćenje još nekoliko dodatnih metoda, kao što su, recimo, **PUT** i **DELETE**. Metod **PUT** često se koristi u kombinaciji sa alatkama za objavljuvanje na Webu. On omogućava korisniku ili aplikaciji da prebací konkretan objekat na konkretnu putanju (direktorijum) veb servera. Metod **DELETE** omogućava korisniku ili aplikaciji da briše neki objekat sa veb servera.



Slika 2.8 ♦ Opšti format poruke sa zahtevom

HTTP poruka sa odgovorom

U nastavku možete da vidite kako izgleda uobičajena HTTP poruka sa odgovorom. Ova poruka koju smo naveli mogla bi da predstavlja odgovor na poruku sa zahtevom o kojoj smo prethodno govorili.

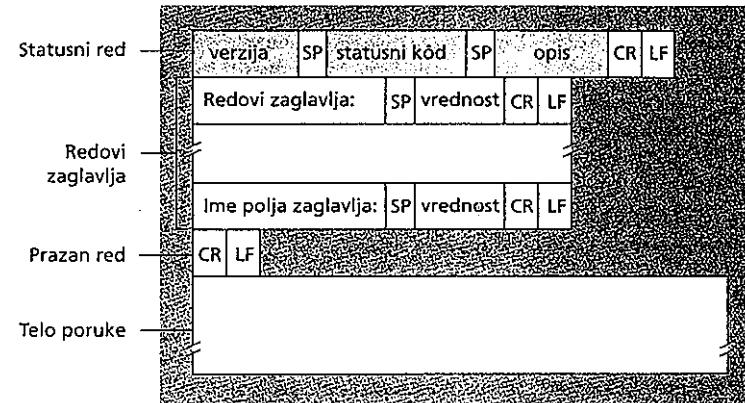
```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 03 Jul 2003 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun, 5 May 2003 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html
```

(podaci podaci podaci podaci podaci ...)

Pogledajmo malo pažnju ovoj poruci. Ona ima tri dela: inicijalni statusni red, šest redova zaglavja i telo poruke. Telo poruke predstavlja najveći deo poruke - ovde se nalazi traženi objekat (podaci podaci podaci podaci ...). Statusni red ima tri polja - polje sa verzijom protokola, statusni kod i odgovarajući poruka sa statusom. U ovom primeru statusnim redom navedeno je da server koristi protokol HTTP/1.1 i daje sve u redu (OK), odnosno, daje server pronašao i poslao traženi objekat.

Predimo sada na redove zaglavja. Server pomoću reda Connection : close , govori klijentu da će nakon slanja traženog objekta prekiniti TCP konekciju. Redom Date : navode se datum i vreme kada je server generisao i poslao HTTP odgovor. Srećemo vam pažnju na to da to nije vreme kada je traženi objekat napravljen ili poslednji put promenjen, već vreme kada gađe server učitao iz svog sistema datoteka, stavlja u poruku sa odgovorom i poslao klijentu. Redom Server : naznačeno je da je poruka generisao web server Apache; ovaj red odgovara redu User-agent: HTTP poruke sa zahtevom. U redu Last-Modif id: navodi se kada je traženi objekat napravljen ili poslednji put izmenjen. Zaglavje Last-Modif id : , komično uskoro posvetili nešto više pažnje, izuzetno je važno za keširanje objekata - kako na lokalnom klijentu, tako i na mrežnim serverima za keširanje (oni se nazivaju i proksi serveri). U zaglavju Content-Length: navodi se broj bajtova objekta koji se šalje. Konačno, redom zaglavja Content-Type: naznačava se tip datoteke koja se nalazi u telu poruke (u ovom slučaju to je HTML tekst). (Tip objekta se zvanično naznačava zaglavjem Content-Type: , a ne označom tipa datoteke.)

Pošto smo zajedno analizirali ovaj primer, pogledajmo kako izgleda opšti format poruke sa odgovorom (slika 2.9). Kao što vidite, opšti format ove poruke se u potpunosti podlapa sa našim primerom. Recimo ponešto i o statusnim kodovima i njihovim opisima. Statusnim kodom i odgovarajućim opisom navodi se rezultat zahteva. Evo nekih uobičajenih statusnih kodova i odgovarajućih opisa.



Slika 2.9 ♦ Opšti format poruke sa odgovorom

- ◆ 200 OK : Komanda je uspešno izvršena i zahtevana informacija se vraća u odgovoru.
- ◆ 301 Moved Permanently: Traženi objekat je trajno uklonjen; novi URL se navodi u zaglavju Location: poruke sa odgovorom. Klijentski softver tako automatski dolazi do novog URL-a.
- ◆ 400 Bad Request: Upoštevan kod sa greškom kojom se naznačava da server nije razumeo primljeni zahtev.
- ◆ 404 Not Found: Traženi dokument ne postoji na serveru.
- ◆ 505 HTTP Version Not Supported: Server ne podržava traženu verziju protokola HTTP.

Da li biste želeli da sada vidite pravu HTTP poruku sa odgovorom? Osim stoje veoma prepričljivo, to je i krajnje jednostavno. Dakle, uspostavite Telnet sesiju sa svojim omiljenim web serverom. Zatim upišite poruku sa zahtevom dužine jednog reda kojom ćete tražiti neki objekat koji se nalazi na serveru. Primera radi, u komandnoj liniji možete da upišete:

telnet cis.poly.edu 80

GET /ross/HTTP/1.1 Host: cis.poly.edu

(Nakon upisivanja poslednjeg reda dvaput pritisnite taster Enter.) Ovim se uspostavlja TCP konekcija sa portom 80 na računaru www.eurecom.fr i šalje HTTP poruku sa zahtevom. U poruci sa odgovorom trebalo bi da ugledate osnovnu HTML.

datoteku početne strane profesora Rosa. Ukoliko biste, umesto samog objekta, više voleli da vidite redove HTTP poruke, metod GET zamenite metodom HEAD. Konačno, zamenite i /ross/rečju /banana/i pogledajte kako će izgledati odgovor.

U ovom odeljku govorili smo o raznim redovima zaglavlja koji mogu da se koriste u HTTP porukama sa zahtevima i odgovorima. Specifikacijom protokola HTTP definisani su i mnogi drugi redovi zaglavlja koje mogu da ubace čitači, veb serveri i serveri za keširanje, a mi smo ovde prikazali samo mali deo njihovog ukupnog broja. Vrlo brzo ćemo pomenuti još neke od njih, dok ćemo o nekim drugim govoriti na kraju poglavљa u odeljku 2.2.6 koji je posvećen mrežnom keširanju Weba. Iscrpnu i čitku diskusiju o protokolu HTTP možete da pronađete i u delu [Krishnamurty 2001], a pogled iz perspektive programera u delu [Luotonen 1998]. Konačno, preporučujemo vam i jedno odlično uvodno razmatranje Weba [Yeager 1996].

Na koji način čitač odlučuje koji će se redovi naći u poruci sa zahtevom? Na koji način server odlučuje koji će redovi zaglavlja ući u poruku sa odgovorom? Redovi zaglavlja koje čitač generiše predstavljaju funkciju tipa i verzije čitača (na primer, čitač tipa HTTP/1.0 neće generisati zaglavljia tipa 1.1), zatim načina na koji gaje korisnik podesio (recimo, željeni jezik), kao i toga da li čitač ima keširanu, ali zastarelu verziju objekta. Veb serveri se ponašaju na sličan način - postoje različiti proizvodi, verzije i konfiguracije, što sve utiče na redove zaglavlja koji će se naći u poruci sa odgovorom.

2.2.4 Interakcija između korisnika i servera: kolačići

Nešto ranije napomenuli smo da su HTTP serveri aplikacije bez nadgledanja stanja. Ova činjenica je pojednostavila njihov dizajn i omogućila programerima da razviju veoma brze veb servere koji mogu da rukuju sa hiljadama istovremenih TCP konekcija. Ipak, postoje i situacije u kojima je poželjno identifikovati korisnike, bilo zato što server želi da ograniči pristup određenim sadržajima, bilo zato što se sadržaj nudi zavisno od toga koji je korisnik u pitanju. Protokol HTTP u ovim situacijama koristi kolačiće.

Kolačići, koji su definisani dokumentom RFC 2109, predstavljaju alternativan mehanizam pomoću koga veb lokacije mogu da prate korisnike. Kolačiće ne koriste sve veb lokacije već, pre svega, veliki portali (na primer, Yahoo), kao i lokacije za elektronsku trgovinu (Amazon) ili oglašavanje (DoubleClick) koji ih, s druge strane, koriste u veoma velikoj meri.

Tehnologija kolačića ima Četiri komponente - (1) red zaglavlja kolačića u HTTP poruci sa odgovorom, (2) red zaglavlja kolačića u HTTP poruci sa zahtevom, (3) datoteku kolačića koja se čuva u korisnikovom krajnjem sistemu, a kojom upravlja njegov čitač i (4) pozadinsku bazu podataka koja se nalazi na veb lokaciji. Prođimo sada zajedno kroz primer korišćenja kolačića na koji se često može naći. Prepostavimo daje Suzan, koja iz svog doma uvek pristupa Webu putem Internet Explo-rera, prvi put kontaktirala sa nekom lokacijom za elektronsku trgovinu koja koristi

kolačiće. Odmah nakon prijema zahteva, veb server pravi jedinstveni identifikacioni broj, kao i zapis u pozadinskoj bazi podataka koji se indeksira upravo napravljenim identifikacionim brojem. Server će zatim da pošalje odgovor Suzaninom Čitaču u kome će se naći i zaglavje Set-cookie : zajedno sa identifikacionim brojem. Primera radi, ovaj red zaglavlja mogao bi da izgleda ovako:

Set-cookie: 1678453

Kada Suzanin čitač primi HTTP poruku sa odgovorom, on će ugledati i zaglavje Set-cookie :. Čitač zatim dodaje jedan red specijalnoj datoteci kolačića kojom on sam upravlja. U ovaj red upisuju se ime servera i identifikacioni broj iz zaglavlja Set-cookie:. Dok Suzan bude pretraživala ovu lokaciju, prilikom svakog traženja neke nove veb stranice, njen čitač će konsultovati datoteku sa kolačićem, izvući njen identifikacioni broj za ovu lokaciju i staviti ga u zaglavje kolačića koje se dodaje HTTP poruci sa zahtevom. Konkretno, svaki njen HTTP zahtev upućen serveru za elektronsku trgovinu imaće i red zaglavlja:

Cookie: 1678453

Na ovaj način veb lokacija može da prati Suzaninu aktivnost. Iako veb lokacija ne zna Suzanino ime, ona tačno zna koje je stranice posetio korisnik broj 1678453, kojim redom i u koje vreme! Pomoću kolačića ova lokacija za elektronsku trgovinu zatim može da obezbedi uslugu kolica za kupljenu robu - tokom odredene sesije na datoj lokaciji, server održava listu svih Suzaninih kupovina u cilju objedinjenog plaćanja na kraju sesije.

Ukoliko se Suzan, na primer, nedelju dana kasnije vrati na ovu lokaciju, njen čitač će nastaviti da u poruke sa zahtevom stavlja zaglavje Cookie: 1 678453. Zavisno od toga koje je stranice Suzan posetila, lokacija za elektronsku trgovinu može da joj preporuči određene proizvode. Ukoliko odluči da se na ovoj lokaciji i registruje - odnosno, da upiše puno ime i prezime, e-adresu, običnu poštansku adresu i informacije u vezi sa svojom kreditnom karticom, lokacija za elektronsku trgovinu će ove informacije takođe da ubaci u svoju bazu podataka i tako Suzanino ime poveže sa njenim identifikacionim brojem (i svim stranama koje je u prošlosti posetila na daljoj lokaciji!). Upravo na ovaj način lokacije za elektronsku trgovinu obezbeđuju „kupovinu jednim potezom mišem“ - kada Suzan u nekoj narednoj poseti odluči da nešto kupi, neće morati ponovo da upiše svoje ime, broj kreditne kartice i adresu.

Iz ovoga možemo da zaključimo da kolačići mogu da se koriste za identifikaciju korisnika. Kada prvi put poseti neku lokaciju, korisnik može da se identifikuje (recimo, navođenjem svog imena). Nakon toga Čitač će proslediti zaglavje kolačića serveru prilikom svake buduće posete i tako identifikuje korisnika na serveru. Iz prethodne priče takođe možemo da zaključimo da kolačići mogu da se koriste za pravljenje sloja korisnikove sesije povrh protokola HTTP koji nema nadgledanje stanja. Kada se, primera radi, korisnik prijavi na veb aplikaciju za e-poštu, Čitač šalje

serveru informaciju o kolačiću, omogućavajući serveru da identifikuje korisnika tokom njegove sesije sa aplikacijom.

Iako značajno olakšavaju i pojednostavljaju kupovinu preko Interneta, kolačići su istovremeno i veoma kontroverzni zato što na neki način zadiru u privatnost korisnika. Kao što smo upravo rekli, kombinacijom kolačića i informacija o korisniku iz njegovog naloga, veb lokacija može da sazna dosta toga o njemu i svoja saznanja eventualno proda nekoj zainteresovanoj kompaniji. Štaviše, kolačići mogu da se upotrebe i za prikupljanje informacija o korisnikovom ponašanju na *veoma velikom broju* veb lokacija. Veb lokacije koje objavljaju reklame na svojim stranicama koriste upravo HTTP zahteve da bi uzele reklame (u formatima GIF ili JPEG) sa HTTP servera reklamne agencije. Svaki od zalueva HTTP serveru reklamene agencije može da sadrži i kolačić kojim upravlja ta agencija. S obzirom na to da reklamne agencije na Internetu isporučuju reklame mnoigm veb lokacijama, one mogu da sačine profil nečijeg ponašanja prilikom pretraživanja veb lokacija.

Na kraju ovog odeljka uputićemo vas na Persistent Client State HTTP Cookies [Netscape Cookie 1999] u kome ćete naći iscrpne i lako razumljive uvodne informacije o kolačićima. Takođe vam preporučujemo i Cookie Central [Cookie Central 2004] u kome ćete pronaći dosta informacija o kontroverznim stranama kolačića.

2.2.5 HTTP sadržaj

Kroz celo ovo poglavlje uzeli smo da su podaci koji se prenose u HTTP odgovorima u stvari objekti sa veb strana - HTML, GIF i JPEG datoteke, Java appleti itd. Protokol HTTP smo namerno predstavili u kontekstu Weba zato što smo želeli da vam damo primer koji vam je poznat. Međutim, bio bi veliki propust ne reći da se protokol HTTP često koristi i za transfer mnogih drugih tipova datoteka.

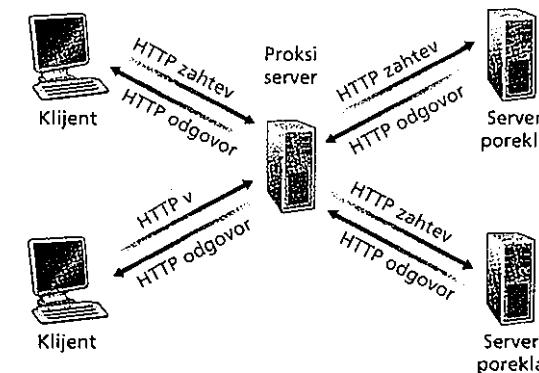
Na primer, protokol HTTP se u savremenim aplikacijama za elektronsku trgovinu često koristi za prenos XML datoteka sa jednog računara na drugi, a da ni na jednom od tih računara ne postoji čitač ili korisnik. Banke često koriste XML dokumente za strukturi sanje bankovnih informacija (recimo, informacija o računu korisnika), a njihovi računari za razmenu ovih strukturisanih informacija često koriste upravo protokol HTTP. (Diskusija o XML dokumentima ne spada u domen ove knjige. Ovde ćemo reći samo to da uobičajen XML dokument sadrži strukturisane podatke i indikaciju značenja tih podataka; ovde obično nema indikacija o formatiranju kao u HTML dokumentima). Protokol HTTP se takođe koristi za transfer VoIP-a, WML (markerski jezik WAP-a) i ostalih tipova XML dokumenata. Štaviše, protokol HTTP često se koristi i u P2P razmeni datoteka, što ćete videti na kraju ovog poglavlja, dok ćete u poglavlju 6 videti da se veoma često koristi i za protokol audio i video sadržaja u realnom vremenu.

2.2.6 Web keširanje

Veb keš ili proksi server jeste mrežni entitet koji izlazi u susret HTTP zahtevima u ime **servera porekla**. Veb keš ima sopstvene diskove za skladištenje na kojima čuva kopije nedavno traženih objekata. Kao što možete da vidite na slici 2.10, čitač može da se konfiguriše tako da se svi korisnikovi HTTP zahtevi najpre usmeravaju ka veb kesu. Primera radi, pretpostavimo da određeni čitač traži objekat <http://www.someschool.edu/campus.gif>. Evo šta se dešava:

1. Čitač uspostavlja vezu sa veb kesom i šalje mu HTTP poruku zahtevajući određeni objekat.
2. Veb keš zatim proverava da li, možda, posede lokalnu kopiju traženog objekta. Ukoliko traženi objekat postoji u veb kesu, on se, zajedno sa HTTP odgovorom, šalje klijentovom čitaču.
3. Ukoliko ne posede traženi objekat, veb keš uspostavlja TCP konekciju sa serverom porekla (u ovom slučaju, to je server www.someschool.edu) i od njega traži dati objekat. Nakon prijema ovog zahteva, server porekla šalje traženi objekat u okviru svog HTTP odgovora veb kesu.
4. Kada primi ovaj objekat, veb keš ga skladišti na svom disku i potom njegovu kopiju prosledjuje klijentovom čitaču u okviru HTTP poruke sa odgovorom (postojećom TCP konekcijom između klijentovog čitača i veb kesu).

Skrećemo vam pažnju na to da se veb keš istovremeno ponaša i kao server i kao klijent. U razmeni zahteva i odgovora sa klijentovim čitačem, veb keš ima ulogu servera, a u razmeni zahteva i odgovora sa serverom porekla, on je klijent.



Slika 2.10 ♦ Zahtevanje objekata kroz veb keš

Veb keš obično kupuje i instalira posrednik za Internet usluge. Primera radi, jedan univerzitet bi mogao da instalira veb keš u svojoj mreži, nakon čega bi svi čitači u studentskom gradu svojom konfiguracijom bili usmereni ka njemu. Isto lako, veliki rezidencija ili posrednik za Internet usluge (recimo AOL) mogao bi u svojoj mreži da instalira jedan ili više ovakvih servera i da zatim svoje čitače unapred konfiguriše da ga koriste.

Postoje dva razloga zbog kojih je veb keširanje uobičajena pojava na Internetu. Najpre, ono značajno skraćuje vreme odziva na zahtev klijenta, naročito ukoliko je usko grlo izraženije između klijentovog računara i servera porekla nego između klijentovog računara i veb kesa. Ako između klijentovog računara i veb kesa postoji veza velike propusne moći, a najčešće je tako, i ukoliko se traženi objekat već nalazi u vebovskoj keš memoriji, isporuka objekta klijentovom računaru biće mnogo brža. Zatim, kao što ćete, uostalom, i videti u našem primeru, veb keširanje značajno smanjuje intenzitet saobraćaja na pristupnom linku neke institucije (univerziteta ili kompanije) ka Internetu. Ovo smanjivanje intenziteta saobraćaja znači da nadgradnje propusnog opsega ne moraju biti toliko česte, čime se smanjuju i troškovi. Štaviše, veb keširanje značajno smanjuje intenzitet saobraćaja i kada se Internet posmatra u celini, što donosi poboljšanje performansi svih njegovih aplikacija.

Da biste na najbolji mogući način shvatili sve prednosti veb keširanja, evo jednog primera ilustrovanog na slici 2.11. Na ovoj slici prikazane su dve mreže - institucionalna mreža i javni Internet. Institucionalna mreža je LAN velike brzine. Ruter ove mreže i ruter u okviru Interneta povezani su linkom brzine 1,5 Mb/s. Serveri porekla su povezani sa Internetom, a nalaze se širom sveta. Pretpostavimo da je veličina prosečnog objekta 100 000 bitova i da institucijski čitači svake sekunde upute prosečno 15 zahteva serverima porekla. Pretpostavljamo i da su HTTP poruke zanemarljivo male tako da ne prave nikakav saobraćaj u mrežama ili pristupnom linku (od institucijskog rutera do rutera u okviru Interneta). Konačno, pretpostavljamo i da od trenutka kada ruter na strani pristupnog linka ka Internetu (slika 2.11) prosledi HTTP zahtev (u okviru IP datagrama) do trenutka kada dobije odgovor (obično u nekoliko IP datagrama) prođu prosečno dve sekunde. Potpuno neformalno, ovo kašnjenje ćemo nazivati „Internet kašnjenje“.

Ukupno vreme odziva - vreme koje protekne od trenutka kada čitač pošalje zahtev za objekat pa do njegovog prijema - predstavlja zbir kašnjenja LAN-a, kašnjenja pristupnog linka (kašnjenje između dva rutera) i kašnjenja Interneta. Sada ćemo napraviti grubu procenu ovog kašnjenja. Intenzitet saobraćaja LAN-a (odeljak 1.6) iznosi:

$$(15 \text{ zahteva/sekundi}) \cdot (100 \text{ kilobita/zahtev}) / (10 \text{ Mb/s}) = 0,15$$

gde je intenzitet saobraćaja u pristupnom linku (od rutera na Internetu do rutera institucije):

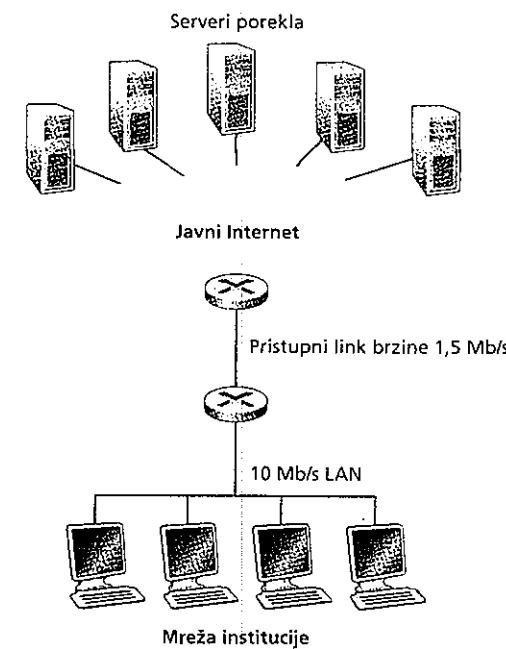
$$(15 \text{ zahteva/sekundi}) \cdot (100 \text{ kilobita/zahtev}) / (1,5 \text{ Mb/s}) = 1$$

Intenzitet saobraćaja od 0,15 u LAN-u obično povlači čekanje koje se, u najgorem slučaju, meri desetinama milisekundi; prema tome kašnjenje LAN-a možemo

da zanemarimo. Međutim, kao što smo rekli u odeljku 1.6, kako se intenzitet saobraćaja približava 1 (kao što je to slučaj na pristupnom linku na slici 2.11), kašnjenje postaje duže i povećava se neograničeno. To znači da će se prosečno vreme odziva na zahteve korisnika meriti minutima, što je iz perspektive korisnika ove institucije neprihvatljivo. Sasvim je očigledno da nešto treba učiniti.

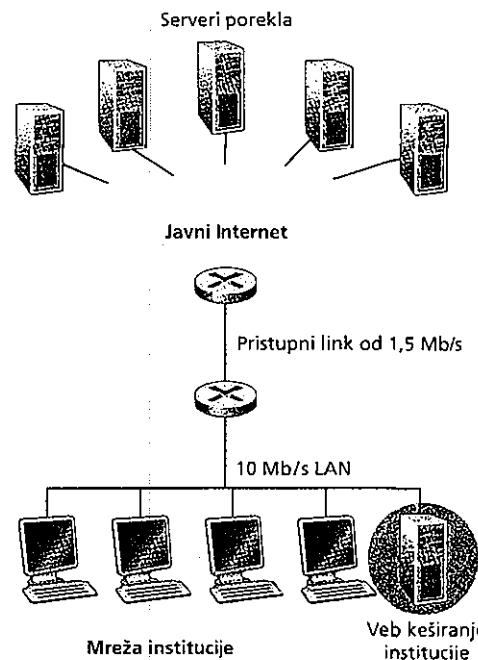
Jedno od mogućih rešenja bilo bi da se brzina pristupa sa 1,5 Mb/s poveća na, recimo, 10 Mb/s. Ovim će se intenzitet saobraćaja u pristupnom linku smanjiti na 0,15. Što znači da će kašnjenja između dva rutera postati zanemarljiva. U ovom slučaju ukupno vreme odziva iznosiće oko dve sekunde, koliko je i kašnjenje u okviru Interneta. Međutim, ovo rešenje podrazumeva nadgradnju pristupnog linka sa 1,5 Mb/s na 10 Mb/s, što može biti i veoma skupo.

Šta bi se dogodilo kada bismo se, umesto za nadgradnju pristupnog linka, oprimili za instaliranje servera za veb keširanje u mreži ove institucije? Ovo rešenje ilustrovano je na slici 2.12.



Slika 2.11 • Usko grlo između mreže institucije i Interneta

Udeo pribavljenih dokumenata - broj zahteva koje zadovolji server za veb keširanje - u praksi se obično kreće između 0,2 i 0,7. Za potrebe primera pretpostavimo da veb keširanje za ovu instituciju pribavlja dokumente u udelu od 0,4. S obzirom na to da su klijenti i server za veb keširanje povezani istim LAN-om velike brzine, to znači da će veb keširanje 40 procenata zahteva zadovoljiti praktično trenutno, sa vremenom odziva od, recimo, 10 milisekundi. Preostalih 60 procenata zahteva, ipak, moraju da zadovolje serveri porekla. Međutim, kako kroz pristupni link sada prolazi samo 60 procenata zahtevanih objekata, intenzitet saobraćaja se sa 1,0 smanjio na 0,6. U praksi, intenzitet saobraćaja koji je manji od 0,8 u linku brzine 1,5 Mb/s obično rezultira kašnjenjima od desetak milisekundi. Što nije mnogo. Ovo kašnjenje je zanemarljivo ukoliko se uporedi sa kašnjenjem Interneta od dve sekunde. Imajući u vidu sve navedene pretpostavke, prosečna vrednost kašnjenja iznosi



Slika 2.12 ♦ Dodavanje servera za veb keširanje u mrežu institucije

0,4 ■ (0,01 sekundi) + 0,6 ■ (2,01 sekundi)

stoje tek nešto malo više od 1,2 sekunde. Prema tome, ovo drugo rešenje donosi kraće vreme odziva od prvog, a ne zahteva nadgradnju linka ka Internetu. Naravno, institucija mora da kupi i instalira server za veb keširanje, ali to je jeftinije - mnogi serveri za veb keširanje koriste softver koji se nalazi u javnom domenu i izvršava na jeftinim personalnim računarima.

2.2.7 Uslovno preuzimanje

Iako keširanjem može da se skrati vreme koje korisnik primećuje, ono uvodi i jedan novi problem - kopija objekta u keš memoriji može da bude zastarela. Drugim recima, moguće je daje objekat koji se nalazi na veb serveru iznenjen nakon što je njegova kopija keširana u klijentskom računaru. Srećom, HTTP ima mehanizam koji omogućava korišćenje keširanja uz istovremenu proveru akmelnosti svih objekata koji su predati čitaču. Ovaj mehanizam naziva se uslovno preuzimanje. HTTP zahtev bi mogao da se nazove uslovnim preuzimanjem ukoliko (1) on sadrži metod GET i (2) ako se u odgovoru nalazi red zaglavlja If-Modified-Since::

Da bismo vam ilustrovali način rada uslovnog preuzimanja, poslužićemo se jednim primerom. Najpre čitač sa nekog servera traži objekat koji nema u svojoj keš memoriji.

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exoticquecuisine.com
```

Zatim, veb server ovom klijentu šalje odgovor i traženi objekat:

```
HTTP/1.1 200 OK
Date: Mon, 7 Jul 2003 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 2 Jul 2003 09:23:24
Content-Type: image/gif
```

(podaci podaci podaci podaci podaci ...)

Veb keš prosleduje traženi objekat čitaču koji ga je tražio, ali ga i lokalno memo-riše. Što je još važnije, veb keš zajedno sa ovim objektom kešira i datum njegove poslednje izmene. Pretpostavimo, dalje, da je korisnik nakon nedelju dana ponovo zatražio ovaj keširani objekat. S obzirom na to daje u toku tih nedelju dana traženi objekat možda izmenjen na veb serveru, čitač proverava njegovu aktuelnost putem uslovnog preuzimanja. Konkretno, veb keš šalje sledeću poruku:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exoticquecuisine.com
If-modified-since: Wed, 2 Jul 2003 09:23:24
```

Skrećemo vam pažnju na to da je vrednost zaglavljia If-modified-since: identična vrednosti zaglavljia Last-Modif ied : iz poruke koju je server poslao pre nedelju dana. Ovim uslovnim metodom GET serveru se govori da traženi objekat pošalje samo ukoliko je on u međuvremenu izmenjen. Prepostavimo da objekat nije menjan od trenutka 2 Jul 2003 0 9 : 23 : 24. U tom slučaju, veb server odgovara sledećom porukom:

```
HTTP/1.1 304 Not Modified
Date: Mon, 14 Jul 2003 15:39:29
Server: Apache/1.3.0 (Unix)
(prazno telo poruke)
```

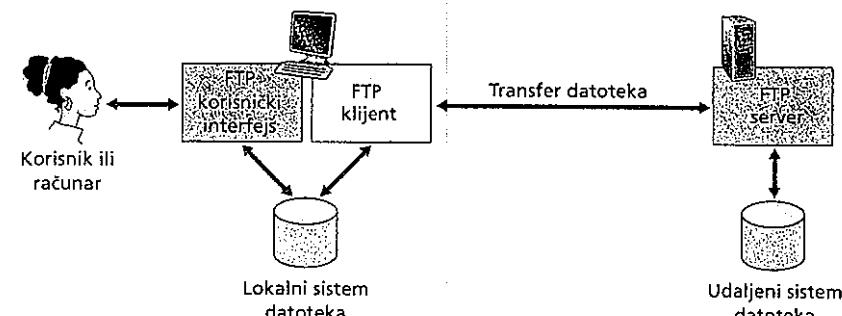
Vidimo daje, kao odgovor na poruku sa uslovnim metodom GET. veb server odgovorio, ali da u toj poruci nije bilo traženog objekta. Njegovim stavljanjem u poruku bespotrebno bi se trošila propusna moć, a odziv bi bio sporiji, naročito ukoliko je reč o većem objektu. Skrećemo vam pažnju i na to da se u poslednjoj poruci nalazi i statusni red 304 Not Modified, kojim se veb kesu saopštava da može da prosledi svoju keširanu kopiju objekta.

2.3 Transfer datoteka: protokol FTP

U uobičajenoj FTP sesiji korisnik sedi ispred jednog (lokalnog) računara i pokušava da prebaci neke datoteke na udaljeni računar ili sa njega. Da bi mogao da pristupi nalogu na udaljenom računaru, korisnik mora da se identificuje i upiše svoju lozinku. Nakon upisivanja ovih informacija za ovlašćivanje, korisnik može da počne sa transferom datoteka iz lokalnog sistema datoteka na udaljeni ili obratno. Kao što možete da vidite na slici 2.13, korisnikova interakcija sa protokolom FTP odvija se preko FTP korisničkog agenta. Korisnik u ovom slučaju najpre upisuje ime udaljenog računara, nakon čega FTP klijentski proces uspostavlja TCP konekcijom sa FTP serverskim procesom udaljenog računara. Nakon toga, korisnik treba da upiše svoje korisničko ime i svoju lozinku koji se zatim šalju TCP konekcijom kao deo FTP komandi. Čim ga server ovlasti, korisnik može da počne da prebacuje jednu ili više datoteka sa lokalnog sistema na udaljeni (ili obratno).

Protokoli HTTP i FTP pripadaju grupi protokola za transfer datoteka tako da imaju mnogo zajedničkih osobina; primera radi, oba protokola funkcionišu povrh protokola TCP. Ipak, između ova dva protokola aplikacijskog sloja postoje i neke bitnije razlike. Najočiglednija razlika jeste to što protokol FTP za transfer datoteka koristi dve paralelne TCP konekcije - kontrolnu konekciju i konekciju za podatke.

Kontrolna konekcija koristi se za razmenu kontrolnih informacija između dva računara. U ovu vrstu informacija spadaju identifikacija i lozinka korisnika, komande

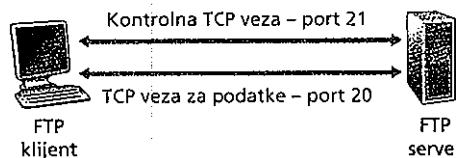


Slika 2.13 ♦ Protokol FTP prebacuje datoteke sa lokalnog sistema na udaljeni i obratno.

za izmenu udaljenog direktorijuma i komande za „stavljanje“ i „uzimanje“ datoteka. Konekcija za podatke obično se koristi za prenos datoteka. S obzirom na to da kod protokola FTP postoji posebna kontrolna konekcija, kaže se da ovaj protokol svoje kontrolne informacije šalje izvan opsega. U pogлављу 7 videćete da protokol RTSP, koji se koristi za kontrolu transfera kontinuiranih medija, kao sto su audio ili video materijal, takođe svoje kontrolne informacije šalje izvan opsega. Podsećamo vas da protokol HTTP redove zaglavljia zahteva i odgovora šalje istom TCP konekcijom kojom se prenosi i sama datoteka. Zato se za protokol HTTP kaže da svoje kontrolne informacije šalje u opsegu. U sledećem odeljku videćete da protokol SMTP, koji se koristi za prenos elektronske pošte, takođe svoje kontrolne informacije šalje u opsegu. Na slici 2.14 ilustrovali smo kontrolnu vezu i vezu za podatke protokola FTP.

Kada korisnik pokrene FTP sesiju sa udaljenim računaram, klijentska strana protokola FTP (korisnik) najpre inicira kontrolnu TCP konekciju sa serverskom stranom (udaljeni računar) i to na serverovom portu broj 21. Klijentska strana protokola FTP ovom kontrolnom vezom potom šalje identifikaciju i lozinku korisnika, kao i komande za izmenu udaljenog direktorijuma. Kada serverska strana ovom kontrolnom vezom primi komande za transfer datoteka (sa tog računara ili na njega), ona inicira uspostavljanje TCP konekcije za podatke do klijentske strane. Protokol FTP ovom konekcijom za podatke šalje samo jednu datoteku i odmah zatim je prekida. Ukoliko, tokom iste sesije, korisnik želi da prebaci još neku datoteku, protokol FTP uspostavlja novu vezu za podatke. Dakle, kod protokola FTP kontrolna konekcija ostaje otvorena tokom cele korisnikove sesije, dok se za svaku novu datoteku uspostavlja i nova konekcija za podatke (konekcije za podatke nisu postojane).

Tokom sesije FTP server mora da održava statusne informacije o korisniku. Konkretno, server mora da poveže kontrolnu vezu sa konkretnim korisničkim nalogom, ali i da prati njegovo kretanje kroz stablo direktorijuma. Neophodnost praćenja ovih informacija za svaku korisničku sesiju značajno ograničava ukupan broj sesija.



Slika 2.14 ♦ Kontrolna vezai veza za podatke

koje protokol FTP može da održava istovremeno. Podsećamo vas da, nasuprot tome, protokol HTTP nema nadgledanje stanja - odnosno, on ne evidentira nikakve informacije o statusu korisnika.

2.3.1 FTP komande i odgovori

Ovaj odeljak završićemo kratkom pričom o FTP komandama koje se često koriste. Komande (od klijenta do servera) i odgovori (od servera ka klijentu) šalju se kontrolnom vezom u sedmobitnom ASCII formatu. To znači da, poput HTTP komandi, ove komande ljudi mogu da pročitaju. Da bi se susedne komande razdvojile, iza svake komande nalazi se znak za povratak na početak reda i znak za prelazak u novi red. Svaka komanda ispisana je pomoću četiri ASCII znaka (velika slova) uz dodatak nekih opcionih argumenata. Evo nekih komandi koje se najčešće upotrebljavaju:

- ♦ **USER username:** Koristi se za slanje korisnikove identifikacije serveru.
- ♦ **PASS password:** Koristi se za slanje korisnikove lozinke serveru.
- ♦ **LIST:** Koristi se za traženje liste svih datoteka koje se nalaze u aktuelnom direktorijumu udaljenog servera. Lista datoteka šalje se (novom i nepostojanom) konekcijom za podatke, a ne kontrolnom TCP konekcijom.
- ♦ **RETR f ilename:** Koristi se za prijem (odnosno, dobijanje) datoteke iz aktuelnog direktorijuma udaljenog računara. Podstiče udaljeni računar da inicira konekciju za podatke i njom pošalje traženu datoteku.
- ♦ **STOR f ilename:** Koristi se za skladištenje (odnosno, stavljanje) datoteke u aktuelni direktorijum udaljenog računara.

Komande koje korisnik izda i FTP komande koje se pošalju kontrolnom vezom, obično stoje u odnosu jedan na jedan. Iza svake komande dolazi i odgovor koji server pošalje klijentu. Ovi odgovori su obično trocifreni brojevi iza kojih se nalazi opcionalna poruka. Po svojoj strukturi oni podsećaju na statusne kodove i opise u statusnom redu HTTP odgovora; kreatori protokola su namerno insistirali na ovoj sličnosti. Evo nekih uobičajenih odgovora zajedno sa opcionim porukama:

- ♦ 331 Username OK, password required
- ♦ 125 Data connection already open; transfer starting

- ♦ 425 Can't open data connection
- ♦ 452 Error writing file

Čitaocu koji žele da upoznaju i druge FTP komande i odgovore upućujemo na dokument RFC 959.

2.4 Elektronska pošta na Internetu

Elektronska pošta postoji od samog početka Interneta i u njegovim najranijim danima predstavljala je njegovu najpopularniju aplikaciju [Segaller 1998]. Kako su godine prolazile, e-pošta se sve više razvijala, postajala sve moćnija i taj trend se zadržao sve do danas, kada je ona i dalje jedna od nekoliko najpopularnijih aplikacija Interneta.

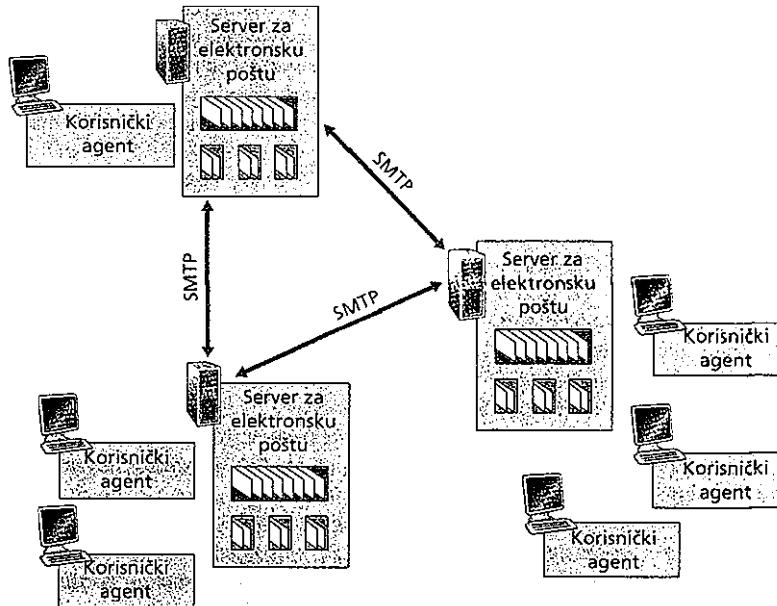
Poput standardne poštanske službe, e-pošta je asinhroni komunikacioni medijum - ljudi šalju i čitaju poruke onda kada im to odgovara a da ne moraju da usklađuju svoje aktivnosti. Ali, za razliku od standardne poštanske službe, e-pošta je brza, jeftina i jednostavno se distribuirala. Savremena e-pošta ima mnogo korisnih osobina. Korišćenjem diskusionih lista poruke e-pošte, ali isto tako i neželjena pošta mogu istovremeno da se upute hiljadama različitih primalaca. Štaviše, u porukama savremene e-pošte često se nalaze priloženi dokumenti, hiperveze, tekst u HTML formatu ili slike. Iako može da se iskoristi i kao platforma za asinhroni prenos glasovnih i video poruka, e-pošta se još uvek najviše koristi za razmenu klasičnih tekstualnih poruka [Ross 2003].

U ovom odeljku ispitaćemo protokole aplikacijskog sloja koji se nalaze u srcu internetske e-pošte. Ipak, pre nego što se pozabavimo detaljima ovih protokola, pogledajmo kako globalno izgleda ovaj sistem.

Slika 2.15 predstavlja opšti prikaz Intemetovog sistema za e-poštu. Dakle, sa ovog dijagrama vidimo da ovaj sistem ima tri osnovne komponente - korisničke agente, servere elektronske pošte i protokol SMTP (Simple Mail Transfer Protocol). Sada ćemo svaku od ovih komponenti da opišemo na primeru pošiljaoca, devojke Alise koja šalje e-poštu primaocu Bobu. Korisnički agenti omogućavaju čitanje poruka, odgovaranje na njih, kao i njihovo prosledivanje, snimanje i sastavljanje. (Korisnički agenti za elektronsku poštu nekada se nazivaju i *čitači elektronske pošte*, ali ćemo mi u ovoj knjizi izbegavati ovaj termin.) Kada Alisa sastavi svoju poruku, njen korisnički agent će poslati ovu poruku njenom serveru za e-poštu, gde će ona biti smeštena u izlazni red sa porukama. Kada Bob bude želeo da pročita ovu poruku, njegov korisnički agent će je preuzeti iz njegovog poštanskog sandučeta na serveru za elektronsku poštu. Krajem 1990-ih postali su veoma popularni korisnički agenti sa grafičkim korisničkim interfejsom (GUI), koji su korisnicima omogućili da pregledaju i sastavljaju multimedijalne poruke. U ovom trenutku Eudora, Microsoft Outlook i Netscape Messenger predstavljaju najpopularnije korisničke agente.

ove vrste. U javnom domenu ima i mnogo tekstualnih korisničkih interfejsa za elektronsku poštu, kao što su, recimo, *mail.pine* i *elm*.

Server za elektronsku poštu Čine jezgro infrastrukture elektronske pošte. Svaki primalac, poput Boba, ima svoje poštansko sanduće koje se nalazi na jednom serveru. Uobičajena poruka svoj život započinje u korisničkom agentu pošiljaoca, zatim putuje do primaočevo servera za elektronsku poštu, na kome se smešta u pri-maočevo poštansko sanduče. Kada Bob poželi da pristupi porukama koje se nalaze u njegovom poštanskom sandućetu, njegov server za elektronsku poštu najpre treba da proveri Bobovu autentičnost (korisničko ime i lozinka). Alisin server za elektronsku poštu takođe mora da prevaziđe i eventualne probleme na Bobovom serveru. Ukoliko njen server ne uspe da isporuči poštu Bobovom serveru, on će poruku zadržati u redu za poruke i pokušati da je isporuči kasnije. Ovi pokušaji se ponavljaju otprilike na svakih tridesetak minuta. Ako ni nakon nekoliko dana ne uspe da isporuči poruku, server je uklanja i o tome obaveštava pošiljaoca (Alisu) jednom porukom.



Slika 2.15 ♦ Uopšteni prikaz Internetovog sistema za elektronsku poštu

Protokol SMTP predstavlja osnovni protokol aplikacijskog sloja za Internet elektronsku poštu. Prilikom prenosa pošte od pošiljačevog do primaočevo poštanskog servera, ovaj protokol koristi uslugu pouzdanog transfera podataka protokola TCP. Poput većine protokola aplikacijskog sloja, protokol SMTP ima dve strane - klijentsku, koja se izvršava na pošiljačevom serveru za e-poštu, i serversku, koja se izvršava na primaočevoj serveru za e-poštu. U klijentski i serverski strani ovog protokola izvršavaju se na svakom serveru. Kada neki server za elektronsku poštu šalje poruke drugim serverima, on se ponaša kao SMTP klijent. Nasuprot tome, kada isti taj server prima poruke od drugih servera, on se ponaša kao SMTP server.

KRATAK OSVRT

HOTMAIL

U decembru 1995. godine Sobir Bhotio i Džek Smit posetili su Draper Fišer Jurvetsonu, kapitalistu čiji su poslovi bili vezani sa Internetom i predložili mu razvoj besplatnog sistema za elektronsku poslu zasnovanog na Vebu. Ideja je bila da se svakom ko loži omogući pravljenje besplatnog naloga za elektronsku poštu kome je moguće pristupiti so bilo kog mesta. Kod elektronske pošte zasnovane na Vebu svako ko ima pristup Vebu - recimo, iz biblioteke ili škole - može da prima i šalje elektronsku poštu. Sto je još važnije, elektronska pošta zasnovana na Vebu omogućava veliku pokretljivost svojim korisnicima. Za vlasništvo nad 15 procenata kompanije, Draper Fišer Jurvetson je finansirao kompaniju koju su Sabir Bhatio i Džek Smit nazvali Hotmail. Sa ukupno tri stalno zaposlene osobe i 12 do 14 povremeno zaposlenih koji su rodili samo za mogućnost da kupu akcije u neko kasnije vreme, oni su uspeli da razviju i lansiraju ovu uslugu jula 1996. godine. Samo mesec dana nakon pokretanja imali su već 500000 članova. Broj članova se veoma brzo uvećavao, i svi oni su, čitajući svoju elektronsku poštu, morali da i čitaju reklamne natpise. U decembru 1997. godine, samo 18 meseci nakon pokretanja usluge, kompaniju Hotmail koja je tada imala 12 miliona članova, otkupio je Microsoft, po nekim procenama za 400 miliona dolara. Uspeh usluge Hotmail najčešće se vezuje za „prednost prvog poteza”, kao i za „virusni marketing“ elektronske pošte. Kompanija Hotmail je imala prednost prvog poteza zato što je bila prva kompanija koja je nudila usluge elektronske pošte zasnovane na Vebu. Ostale kompanije su, naravno, sledile ovaj put, ali sa bor šest meseci zakašnjenja. Ova prednost prvog poteza postiže se originalnom idejom i njenom brzom i tajnom realizacijom. Za uslugu ili proizvod se kaže da ima potencijal virusnog marketinga ukoliko reklamira samog sebe. Elektronska pošta predstavlja klasičan primer ovakve usluge - pošiljalac šalje poruku jednom primaocu ili većem broju primalaca i onda svi oni postaju svesni dote usluge. Primer Hotmaila je pokazao da se kombinacijom prednosti prvog poteza i virusnog marketinga može stići do izuzetno popularne aplikacije. Moguće je da će baš među čitaocima ove knjige biti novih pronalazača koji će uspeti da učine nešto slično.

2.4.1 Protokol SMTP

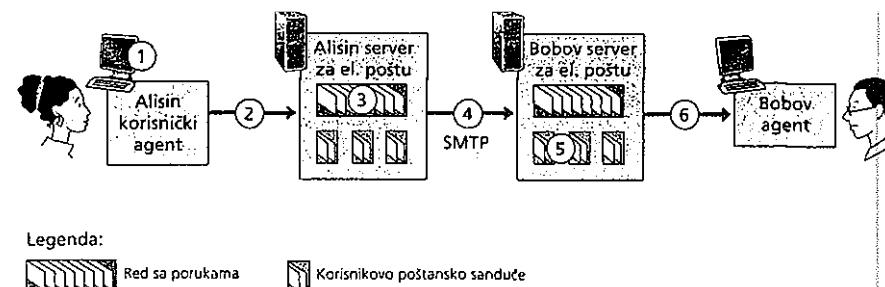
Protokol SMTP, koji je definisan dokumentom RPC 2821, je sama srž internetske elektronske pošte. Kao što smo već rekli, SMTP prenosi poruke između pošiljačevog i primaocevog servera za e-poštu. Protokol SMTP je znatno stariji od protokola HTTP. (Originalni SMTP RFC dokument datira iz 1982. godine, ali se sam SMTP pojavio znatno ranije.) Iako se protokol SMTP odlikuje mnogim izuzetnim osobinama, o čemu svedoči i njegova opšta prisutnost na Internetu, ipak je tu reč o nasleđenoj tehnologiji koja ima i neke „arhaične“ karakteristike. Na primer, ovaj protokol ograničava telo (ne samo zaglavljje) svih poruka na sedmobitni ASCII kod. Ovo ograničenje je imalo smisla: početkom 1980-ih kada je prenosni kapacitet bio skroman i niko nije slao velike slike ili audio i video datoteke. Ali u današnjoj multimedijalnoj eri, ograničenje na sedmobitni ASCII kod predstavlja problem. Binarni multimedijalni podaci moraju da se kodiraju u ASCII kako bi mogli da se prenose protokolom SMTP; nakon SMTP transporta ASCII poruka mora da se dekodira i vrati u prvobitni binarni oblik. Podsećamo vas na odeljak 2.2 u kome smo rekli da protokol HTTP ne zahteva ovu vrstu kodiranja pre transfera.

Da bismo vam ilustrovali način rada protokola SMTP, poslužićemo se jednim uobičajenim scenarijom. Prepostavimo da Alisa želi da pošalje Bobu jednostavnu ASCII poruku.

1. Alisa aktivira svog korisničkog agenta za e-poštu, upisuje Bobovu adresu (na primer, bob@omeschool.edu), sastavlja poruku i zatim nalaže korisničkom agentu daje pošalje.
2. AHsin korisnički agent šalje poruku njenom serveru za e-poštu, na kome se ona smešta u red za poruke.
3. Kada klijentska strana protokola SMTP koja se izvršava na Alisinom serveru za e-poštu uoči ovu poruku, ona uspostavlja TCP konekciju sa serverskom stranom protokola SMTP koja se izvršava na Bobovom serveru za e-poštu.
4. Nakon inicijalnog SMTP rukovanja (synchronizacija), klijentska strana protokola SMTP šalje Alisinu poruku TCP konekcijom.
5. Na Bobovom serveru za e-poštu, serverska strana protokola SMTP prima poruku koju Bobov server zatim smešta u Bobovo poštansko sanduće.
6. Bob pokreće svog korisničkog agenta onda kada on želi da pročita svoju poštu.

Ovaj scenario ilustrovali smo na slici 2.16.

Veoma je važno da primetite da se protokol SMTP prilikom isporuke e-pošte obično ne oslanja na posredničke servere, čak i kada se dva servera nalaze na drugim krajevima sveta. Kada bi se Alisin poštanski server nalazio u Hong Kongu, a Bobov u Sent Luisu, između njih bi se uspostavila direktna TCP konekcija. Ukoliko bi Bobov server bio privremeno u kvaru, poruka ne bi bila smeštena u nekom posredujućem serveru, već bi ostala u Alisinom serveru i u njemu sačekala na neki novi pokušaj.



Slika 2.16 ♦ Alisa šalje poruku Bobu.

Pogledajmo sada izbliza na koji način protokol SMTP prebacuje poruku sa pošiljačevog servera na primaocev server za elektronsku poštu. Uskoro ćete videti da protokol SMTP ima mnogo sličnosti sa protokolima koji se koriste u direktnoj komunikaciji ljudi.

Najpre, klijentska strana protokola SMTP (koja se izvršava na pošiljaču a očevom poštanskom serveru) treba da na svom portu 25 uspostavi TCP konekciju sa serverskom stranom protokola SMTP (koja se izvršava na primaocevom serveru za e-poštu). Ukoliko je server u kvaru, klijent pokušava ponovo. Nakon uspostavljanja veze, klijentska i serverska strana protokola prelaze na proces sifronizacije aplikacijskog sloja, slično kao što se ljudi jedni drugima predstavljaju pre nego što počnu da razmenjuju informacije. Tokom faze sifronizacije, SMTP klijent navodi adresu e-pošte pošiljaoca (osobe koja je napravila i poslala poruku) i odgovarajuću adresu primaoca. Nakon ove faze međusobnog upoznavanja, klijentska strana protokola SMTP započinje slanje poruke. Protokol SMTP se, kada je u pitanju isporuka poruke do drugog servera bez grešaka, oslanja na uslugu pouzdanog transfera podataka protokola TCP. Ukoliko ima još poruka koje treba da pošalje, klijentska strana ponavlja ovaj postupak istom TCP konekcijom; ako nema, ona nalaže protokolu TCP da prekine konekciju.

Pogledajmo sada kako izgleda primer transkripta razmene poruka između SMTP klijenta (C) i SMTP servera (S). Ime računara klijenta glasi crepes . f r, a ime računara servera hamburger. edu. Redovi ASCII teksta ispred kojih stoji C: su upravo oni redovi koje je klijent poslao u svoj TCP soket, dok je ASCII redove ispred kojih stoji S: upravo takve u svoj soket poslao SMTP server. Transkript koji sledi započinje odmah nakon uspostavljanja TCP konekcije.

```

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection

```

U ovom primeru, klijent sa servera crepes . f r šalje poruku („ Do you like ketchup? How about pickles?“) serveru hamburger . edu. Kao deo dijaloga klijent je izdao pet komandi: HELO (skraćenica od HELLO), MAIL FROM, RCPT TO, DATA i QUIT. Ove komande u dovoljnoj meri same sebe objašnjavaju. Osim toga, klijent je postao i red u kome se nalazi samo jedna tačka, kojom se naznačava kraj poruke upućene serveru. (U ASCII žargonu svaka poruka se završava sa CRLF. CRLF, gde se slova CR odnose na povratak na početak reda, a LF na prelazak u novi red.) Server no svaku od ovih komandi daje odgovore koji imaju odgovarajući kod i neko (opciono) objašnjenje na engleskom jeziku. Na ovom mestu reći ćemo i to da protokol SMTP koristi postojane veze - ukoliko pošiljačev server za elektronsku poštu treba da pošalje primaočevom serveru nekoliko poruka, on će ih sve poslati istom TCP konekcijom. Za svaku pojedinačnu poruku klijent započinje proces sa MAIL FROM: crepes . f r. dok njen kraj označava izolovanom tačkom; komandu QUIT izdaje tek nakon što pošalje sve poruke.

Veoma je preporučljivo da za direktni dijalog sa SMTP serverom koristite program Telnet. Da biste to učinili, potrebno je da izdate sledeću komandu:

```
telnet serverName :25
```

gde parametar serverName predstavlja ime lokalnog servera za elektronsku poštu. Čineći ovo vi, jednostavno, uspostavljate TCP konekciju između vašeg lokalnog računara i servera za elektronsku poštu. Odmah nakon upisivanja ovog reda, trebalo bi da od servera primite odgovor 2 2 0. Zatim, pravovremeno izdajte SMTP komande HELQ, MAIL FROM, RCPT TO, DATA, CRLF . CRLF i QUIT. Na kraju ovog poglavљa obavezno provežbajte programerski zadatak 2 u kome treba da napravite jednostavnog korisničkog agenta koji implementira klijentsku stranu protokola SMTP. Pomoću ovog korisničkog agenta moći ćete da šaljete poruke bilo kom primaocu putem lokalnog servera za elektronsku poštu.

2.4.2 Poredenje sa protokolom HTTP

U nastavku teksta ukratko ćemo uporediti protokole SMTP i HTTP. Dakle, ova dva protokola koriste se za transfer datoteka sa jednog računara na drugi; HTTP prenosi datoteke (nazivaju se i objektima) od veb servera do veb klijenta (najčešće čitača); SMTP prenosi datoteke (elektronsku poštu) od jednog servera za elektronsku poštu do drugog. Prilikom transfera datoteka postojani protokol HTTP i SMTP koriste postojeće veze. Prema tome, moglo bi se reći da ova dva protokola imaju neke zajedničke karakteristike. Međutim, između njih postoje i veoma značajne razlike. Najpre, protokol HTTP je u najvećoj meri prijemni **protokol** - neko postavlja određene informacije na veb serveru, □ korisnici pomoću protokola HTTP primaju te informacije sa servera onda kada oni to žele. Konkretno, TCP konekciju inicira onaj računar koji želi da primi podatke. S druge strane, protokol SMTP je prvenstveno predajni **protokol** - pošiljačev server za elektronsku poštu predaje datoteku primaočevom serveru za elektronsku poštu. U ovom slučaju TCP konekciju inicira onaj računar koji želi da pošalje (preda) svoju datoteku.

Druga razlika, koju smo već pomenuli, jeste to što protokol SMTP zahteva da svaka poruka, uključujući tu i telo svake poruke, bude u sedmobitnom ASCII formatu. Ako poruka sadrži znake koji nisu sedmobitni ASCII kod (na primer, francuska slova sa akcentima), ili sadrže binarne podatke (recimo, slike), onda ona mora da se kodira u sedmobitni ASCII kod. Protokol HTTP ne nameće ovo ograničenje.

Treća bitna razlika tiče se načina rukovanja dokumentima koji sadrže tekst i slike (i neke druge tipove medija). Kao što ste videli u odeljku 2.2, protokol HTTP enkapsulira svaki objekat u sopstvenu poruku sa odgovorom. Internetska elektronska pošta, kao što ćete uskoro videti, sve objekte poruke objedinjuje u jednu poruku.

2.4.3 Formati elektronske pošte i M1ME

Kada Alisa napiše Bobu standardno pismo, ona na njegovom vrhu može da upiše sve vrste perifernih informacija, kao što su Bobova adresa, njena povratna adresa i datum. Slično tome, prilikom slanja elektronske pošte od jedne drugoj, zaglavje sa perifernim informacijama prethodi telu same poruke. Ove periferne informacije nalaze se u seriji redova zaglavja koja su definisana dokumentom RFC 822. Redovi zaglavja odvojeni su od tela poruke jednim praznim redom (odnosno, kontrolnim znakovima CRLF). Dokument 822 precizira tačan format redova zaglavja, kao i njihove semantičke interpretacije. Poput protokola HTTP, svaki red zaglavja sadrži neki čitljiv tekst koji Čini ključna reč praćena znakom dve tačke i vrednošću. Neke ključne reči su obavezne, dok su druge samo opciione. Svako zaglavje mora imati redove From: i To :, a u njemu se može naći i red Subject:, kao i neki drugi opcioni redovi. Ovde želimo da naglasimo da se ovi redovi zaglavja razlikuju od SMTP komandi o kojima smo govorili u odeljku 2.4. 1 (i pored toga što se i kod jednih i kod drugih javljaju reči „from“ i „to“). Komande o kojima smo govorili u ovom odeljku bile su deo sinhronizacije protokola SMTP; redovi zaglavja o kojima sada pričamo Čine deo same poruke elektronske pošte.

Uobičajeno zaglavje poruke izgleda ovako:

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

Nakon zaglavlja poruke sledi prazan red, pa zatim telo poruke (u ASCII kodu). Preporučujemo vam da pomoću programa Telnet pošaljete serveru za elektronsku poštu poruku koja sadrži neke redove zaglavlja, uključujući i red Subject:. U tu svrhu izdajte komandu telnet serverName 25.

MIME proširenja za podatke koji nisu u ASCII kodu

Zaglavla koja su opisana u dokumentu RFC 822 pogodna su za slanje običnog ASCII teksta, ali nisu dovoljno bogata za multimedijalne poruke (na primer, poruke sa slikama, audio i video materijalom) ili za prenos teksta koji nije u ASCII kodu (na primer, znaci koji se koriste u nekim jezicima). Za slanje sadržaja koji se razlikuje od ASCII teksta, korisnički agent pošiljalac mora u poruku da ubaci dodatna zaglavla. Ova dodatna zaglavla su definisana dokumentima RFC 2045 i RFC 2046 - višenamenskim proširenjima Internet poste (*Multipurpose Internet Mail Extensions*, MIME) koji su proširenja dokumenta RFC 822.

Dva ključna MIME zaglavla za podržavanje multimedijalnog sadržaja jesu zaglavla Content-Type : i Content-Transfer-Encoding :. Zaglavje Content-Type : omogućava primajućem korisničkom agentu da preduzme odgovarajuću akciju u vezi sa porukom. Primera radi, naznačavanjem da se u telu poruke nalazi JPEG slika, korisnički agent primaoca može da uputi telo poruke ka funkciji za dekomprimovanje JPEG formata. Da biste razumeli neophodnost zaglavja Content-Transfer-Encoding: posrećamo vas da sve poruke koje nisu u formatu ASCII moraju da se prebace u ovaj format kako ne bi zbunile protokol SMTP. Zaglavje Content-Transfer-Encoding: upozoravajućeg korisničkog agenta daje telo poruke kodirano u ASCII format i navodi vrstu primjenjenog kodiranja. Kada korisnički agent primi poruku sa ova dva zaglavla, on najpre, korišćenjem vrednosti zaglavja Content-Transfer-Encoding : pretvara telo poruke u njegov prvočitni (ne-ASCII) oblik, a zatim, pomoću zaglavja Content-Type :, određuje koje akcije treba da preduzme u vezi sa datim telom poruke.

Predimo sada na jedan konkretni primer. Pretpostavimo da Alisa želi da pošalje JPEG sliku Bobu. Dakle, ona otvara svog korisničkog agenta za elektronsku poštu, navodi Bobovu adresu elektronske pošte, naznačava temu poruke i umeće JPEG sliku u njenu telo. (Zavisno od toga koji je korisnički agent u pitanju, moguće je da će slika biti umetnuta kao priloženi dokument.) Kada završi sastavljanje svoje poruke, Alisa će pritisnuti dugme „Send“ svog korisničkog agenta. Odmah zatim, njen korisnički agent će generisati MIME poruku koja bi mogla da izgleda otprilike ovako:

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
```

```
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

```
(base64 kodirani podaci .....
.....
.....base64 kodirani podaci)
```

Iz ove MIME poruke možemo da zaključimo daje Alisin korisnički agent kodirao JPEG sliku korišćenjem kodiranja tipa base64. To je jedna od nekoliko standardizovanih MIME [RFC 2045] tehnika za kodiranje u prihvatljivi sedmobitni ASCII format. Drugu popularnu tehniku za kodiranje predstavlja tehniku za kodiranje sadržaja tipa *quoted-printable* koja se obično koristi za pretvaranje osmobilnih ASCII poruka (na primer, sa karakterističnim slovima nekih jezika) u sedmobitni ASCII format.

Kada Bob bude želeo da pročita svoju elektronsku poštu, njegov korisnički agent će se suočiti sa istom ovom MIME porukom. Čim uoči zaglavje Content-Transfer-Encoding : base64, Bobov korisnički agent prelazi na dekodiranje tela poruke. Sledеće zaglavje, Content-Type : image/jpeg, govori Bobovom korisničkom agentu da telo poruke treba da dekomprimuje funkcijom za dekomprimovanje JPEG formata. Konačno, u poruci стоји и zaglavje MIME-Version: koje, naravno, naznačava verziju upotrebljenog MIME proširenja. Skrećemo vam pažnju na to da u svim drugim aspektima poruka sledi standardni RFC 822/SMTP format. Na primer, nakon zaglavlja poruke sledi prazan red, pa zatim telo poruke.

Primljena poruka

Napravili bismo veliki propust ukoliko ne bismo pomenuли još jednu klasu redova zaglavlja - onu koja u poruku umeće *prijemni* SMTP server. Nakon što primi poruku sa redovima zaglavlja tipa RFC 822 i MIME, ovaj server na njen sam vrh dodaje red zaglavla Received: ■ u ovom redu navodi se ime SMTP servera koji je poslao poruku („from“), ime SMTP servera koji je primio („by“), kao i vreme kada je poruka primljena. Prema tome, kada stigne do svog konačnog odredišta, poruka izgleda ovako:

```
Received: from crepes.fr by hamburger.edu; 12 Oct 98
15:27:39 GMT
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

```
base64 kodirani podaci .....
.....
.....base64 kodirani podaci
```

Gotovo svako koje koristio elektronsku poštu bio je u prilici da vidi red zaglavlja Received : (i ostale redove) koji prethodi poruci. (Ovaj red često može da se vidi na ekranu ili prilikom štampanja poruka.) Isto tako, možda ste primetili da jedna poruka može imati više zaglavlja Received: i komplikovanje zaglavljve Peturn-Path :. To je znak daje, na putu između pošiljaoca i primaoca, poruka prosledena većem broju SMTP servera. Kada bi, primera radi, Bob od svog servera hamburger . edu tražio da se sve njegove poruke proslede serveru sushi . jp, sve njegove poruke bi počinjale otprilike ovako:

Received: from hamburger.edu by sushi.jp; 3 Jul 01 15:30:01 GMT
 Received: from crepes.fr by hamburger.edu; 3 Jul 01 15:17:39 GMT

Ovi redovi zaglavlja omogućavaju prijemnom korisničkom agentu da prati koji su SMTP serveri posećeni i kada.

2.4.4 Protokoli za prijem elektronske pošte

Kada protokol SMTP isporuči poruku sa Alisinog poštanskog servera Bobovom serveru, poruka se smešta u Bobovo poštansko sanduče. U celom ovom izlaganju podrazumevali smo da Bob do svoje elektronske pošte dolazi tako što se prijavljuje određenom serveru, a zatim se njegov čitač elektronske pošte izvršava na tom serveru. Sve do 1990-ih ovo je bio i standardni način za dolaženje do elektronske pošte. Međutim, današnji tipični korisnik do svoje pošte dolazi tako što se njegov korisnički agent izvršava na njegovom krajnjem sistemu, što može da bude njegov PC u kancelariji, ili kućni Mac, ili možda PDA računar. Izvršavanje korisničkih agenata na lokalnim računarama donelo je krajnjim korisnicima čitav niz novih funkcija, uključujući i pregledanje multimedijalnih poruka i priloženih dokumenata.

Ukoliko se Bobov korisnički agent izvršava na njegovom računaru, onda bi, valjda, bilo očekivano da se i server za elektronsku poštu takođe nalazi na Bobovom lokalnom računaru. Sledeći ovakav pristup, Alisin poštanski server morao bi da ostvaruje direktni dijalog sa Bobovim računaram. Međutim, ovde postoji i jedan problem. Ako se sećate, rekli smo da server za elektronsku poštu upravlja poštanskim sandučićima i da se na njemu izvršavaju klijentska i serverska strana protokola SMTP. Kada bi se Bobov server za e-poštu nalazio na njegovom lokalnom računaru, taj računar bi morao da bude neprekidno uključen i povezan sa Internetsom, jer bi jedino tako mogao da primi poštu koja može da pristigne u svakom trenutku. Ovakva koncepcija je zato nepraktična i neprihvatljiva za veliku većinu savremenih korisnika Interneta. Umesto toga, prosečan korisnik pokreće korisničkog agenta na lokalnom računaru, ali pristupa poštanskom sandučetu preko zajedničkog servera za e-poštu -onog koji je uvek uključen i uvek povezan sa Internetsom. Ovakve servere zajednički koristi veći broj korisnika, a obično ih održava njihov posrednik za Internet usluge (recimo, univerzitet ili neka kompanija).

Pogledajmo sada kako izgleda putanja kojom elektronska pošta putuje od Alise do Boba. Upravo ste naučili da u nekom delu ove putanje poruka mora da se sačuva u Bobovom serveru za elektronsku poštu. Ovaj cilj se postiže veoma jednostavno - tako što Alisin korisnički agent direktno pošalje poruku Bobovom serveru za elek-

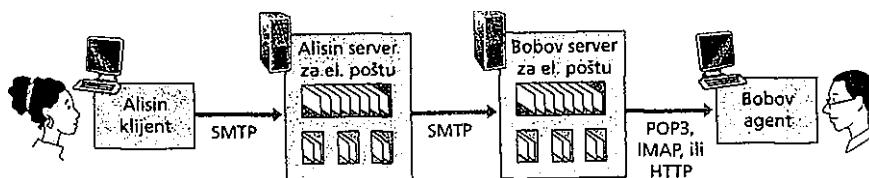
tronsku poštu. Kao što već verovatno prepostavljate, za ovo je zadužen protokol SMTP, koji je upravo i napravljen za predaju (guranje) elektronske pošte od jednog računara do drugog. Međutim, Alisin korisnički agent ne ostvaruje direktnu komunikaciju sa Bobovim serverom za elektronsku poštu, već pomoću protokola SMTP predaje elektronsku poštu Alisinom serveru za elektronsku poštu koji zatim, opet pomoću protokola SMTP (njegove klijentske strane), predaje poruku Bobovom serveru za elektronsku poštu (slika 2.17). Verovatno se sada pitate čemu ovakva dvofazna procedura. Pre svega zato što, bez posredovanja njenog servera za e-poštu, Alisin korisnički agent ne bi mogao ništa da zna o eventualnom kvaru određenog poštanskog servera. Kada Alisin server primi njenu e-poštu, on može svakih 30 minuta da pokušava da je pošalje sve dok se Bobov server ne vrati u funkciju. (U slučaju kvara njenog poštanskog servera, ona uvek može da se žali administratoru sistema.) RFC dokumentima o protokolu SMTP uređenje način korišćenja SMTP komandi za prosleđivanje poruke između većeg broja servera.

Ipak, u celoj ovoj zagonetki nedostaje nam još jedan detalj. Na koji način primalac (poput Boba) sa korisničkim agentom koji se izvršava na njegovom lokalnom računaru stiže do poruka koje se nalaze na serveru za e-poštu njegovog posrednika za Internet usluge? Skrećemo vam pažnju na to da Bobov korisnički agent ne može do ovih poruka da stigne pomoću protokola SMTP zato što je preuzimanje poruka operacija prijema (povlačenja), a protokol SMTP je protokol predaje (guranja). Upravo tu na scenu stupa i poslednji detalj ove naše slagalice - poseban protokol za pristup elektronskoj pošti koji će da prebací poruke od Bobovog servera za elektronsku poštu do njegovog lokalnog računara. Trenutno je popularno više ovakvih protokola, uključujući i POP3 (Post Office Protocol - Version 3), IMAP (Internet Mail Access Protocol) i HTTP.

Na slici 2.17 možete da vidite koji protokoli učestvuju u razmeni internetske elektronske pošte: SMTP se koristi za prebacivanje pošte od pošiljačevog korisničkog agenta do njegovog poštanskog servera, kao i od njegovog servera do servera primaoca. Za prebacivanje pošte od primačevog servera za e-poštu do njegovog korisničkog agenta koriste se protokoli za pristup elektronskoj pošti, kao sto je, na primer, protokol POP3.

POP3

Protokol POP3 je izuzetno jednostavan protokol za pristup elektronskoj pošti. Definišan je u dokumentu RFC 1939, koji je veoma kratak i razumljiv. Zbog ove jednostavnosti, funkcionalnosti ovog protokola prilično je ograničena. Protokol POP3 stupa na scenu onog trenutka kada korisnički agent (klijentska strana) otvoriti TCP konekciju sa serverom za elektronsku poštu (serverska strana) na portu 110. Nakon uspostavljanja TCP konekcije, POP3 prolazi kroz tri faze - ovlašćivanje, transakciju i ažuriranje. Tokom prve faze - ovlašćivanja - korisnički agent slanjem korisničkog imena i lozinke identificuje korisnika koji želi da preuzme svoju poštu. U fazi transakcije koja je sledeća, korisnički agent preuzima poruke, ali i markira one koje su za brisanje, uklanjanje oznake za brisanje i preuzima statističke podatke. Treća faza - ažuriranje - nastupa nakon što klijent izda komandu quit, kojom se završava



Slika 2.17 ♦ Protokoli za elektronsku poštu i njihovi komunicirajući entiteti

POP3 sesija; u ovom trenutku server za elektronku postu briše poruke koje su markirane za brisanje.

U POP3 transakciji korisnički agent izdaje komande na koje server odgovara. U ovom smislu postoje dva moguća odgovora: **+OK** (nekada odmah iza slede podaci koje server šalje klijentu), kojom server odgovara da je sa prethodnom komandom sve u redu i **-ERR**, kojom ukazuje na postojanje nekog problema u vezi sa prethodnom komandom.

Faza ovlašćivanja ima dve osnovne komande: **user <korisničko ime>** i **pass <password>**. Da biste sc upoznali sa ovim komandama, preporučujemo vam da, koristeći port 110, uspostavite direktnu Telnet sesiju sa POP3 serverom i da ih zatim zadate. Kada bi ime vašeg poštanskog servera glasilo mail Server, videli biste otprilike ovakvu razmenu komandi:

```

telnet mailServer 110
+OK POP3 server
ready user bob
+OK
pass hungry
+OK user successfully logged on
  
```

Ukoliko neku komandu napišete pogrešno, POP3 server će odgovoriti porukom -

ERR.

Usredsredirno se sada na fazu transakcije. Korisnički agent koji koristi protokol POP3 može da se podesi (to čini korisnik) tako da „ preuzme i briše“ ili „ preuzme i zadrži“ poruke. Konkretni sled komandi koje izdaje POP3 korisnički agent zavisi od toga za koji se od ova dva režima korisnik opredelio. U režimu „ preuzmi i briši“ korisnički agent izdaje komande **list**, **retr** i **dele**. Primera radi, pretpostavimo da neki korisnik u svom poštanskom sandučetu ima dve poruke. U dijalogu koji sledi C: (od klijent) je korisnički agent, a S: (od server) server za elektronku poštu. Cela transakcija izgleda otprilike ovako:

```

C:list S: 1
498 S: 2 912
  
```

```

S: .
C: retr 1
S: (bla bla ...
S: .....
S: .....bla)
S: .
C: dele 1C: retr
2 S: (bla bla
...
S: .....
S: .....bla)
S: .
C: dele 2C:
quit
S:+OK POP3 server se odjavljuje
  
```

Korisnički agent najpre od servera za elektronku poštu traži da navede veličinu svake sačuvane poruke. Nakon toga, korisnički agent najpre preuzima, a zatim briše sve poruke sa servera. Skrećemo vam pažnju na to daje, nakon faze ovlašćivanja, korisnički agent izdao samo četiri komande: **list**, **retr**, **dele** i **quit**. Sintaksa ovih komandi definisana je u dokumentu RFC 1939. Nakon izdavanja komande **quit**, POP3 server prelazi u fazu ažuriranja i iz poštanskog sandučeta uklanja poruke ! i 2.

Problem u vezi sa ovim pristupom preuzimanja i brisanja jeste to što bi primalac Bob mogao da bude stalno u pokretu, a da pri tom želi da sa svih računara koje koristi (u kući, na poslu i sa prenosivog računara) može da pristupi svojoj elektronskoj pošti. Kod preuzimanja i brisanja Bobove poruke bi se u tom slučaju nalazile razdeljene na tri računara: konkretno, ako je neku poruku prvo pročitao na računaru u kancelariji, on ne bi mogao uveče ponovo daje pročita na svom prenosivom računaru. U režimu preuzimanja i zadržavanja, korisnički agent, nakon preuzimanja, ostavlja poruke na serveru za elektronku poštu. Ovaj režim, dakle, omogućava Bobu da ponovo pročita svoju elektronku poštu na nekom drugom računaru; može prvo daje pročita na poslu, a onda i kod kuće, nedelju dana kasnije.

Tokom POP3 sesije između korisničkog agenta i servera za elektronku poštu, POP3 server održava neke statusne informacije. Konkretno, on evidentira koje je poruke korisnik markirao za brisanje. Međutim, POP3 server ove statusne informacije ne prenosi iz jedne sesije u drugu, što značajno pojednostavljuje njegovu implementaciju.

IMAP

Kod POP3 pristupa Bob, nakon što preuzme poruke na svoj lokalni računar, može da napravi direktorijume za elektronku poštu i u njih premesti preuzete poruke. On potom može da ih briše, premešta iz jednog direktorijuma u drugi i pretražuje (prema imenu pošiljaoca ili temi). Ali, ova paradigma direktorijuma i poruka na lokalnom računaru predstavlja problem za korisnike koji su u pokretu i kojima bi

više odgovaralo da se ovakva hijerarhija direktorijuma napravi i održava na udaljenom serveru, kako bi onda mogli da joj pristupe sa bilo kog računara koji koriste. Ovo, međutim, nije moguće postići sa protokolom POP3 u okviru koga ne postoje nikakva sredstva koja bi omogućila pravljenje direktorijuma na udaljenim serverima i raspoređivanje poruka u njih.

Da bi se otklonio ovaj nedostatak, izmišljen je protokol IMAP (*Internet Mail Access Protocol*) koji je definisan u dokumentu RFC 2060. Poput protokola POP3, IMAP je takođe protokol za pristup elektronskoj pošti. Međutim, on ima daleko više funkcija i znatno je složeniji, tako daje i implementacija njegove klijentske i server-ske strane složenija.

IMAP server svaku poruku povezuje sa određenim direktorijumom; čim poruka dospe u server, ona se povezuje sa primaočevim direktorijumom INBOX. Nakon toga primalac može daje prenesti i u neki drugi svoj direktorijum, daje pročita, izbriše itd. U okviru protokola IMAP postoje komande koje korisnicima omogućavaju da prave direktorijume i prebacuju poruke iz jednog svog direktorijuma u drugi. Protokol IMAP takođe obezbeđuje komande koje omogućavaju korisnicima da pretražuju udaljene direktorijume (poruke) prema nekim konkretnim kriteriju-mima. Skrećemo vam pažnju i na to da, za razliku od protokola POP3, IMAP server prenosi statusne informacije iz jedne sesije u drugu - primera radi, imena direktorijuma i veze poruka sa određenim direktorijumima. Druga značajna osobina protokola IMAP jesle to što u njemu postoje komande koje omogućavaju korisnicima da preuzmu samo delove poruke. Primera radi, korisnički ageni može da preuzme samo njeno zaglavje ili jedan deo višedelne MIME poruke. Ova osobina dolazi do punog izražaja kod konekcije niske propusne moći (na primer, kod bežične ili spore modemske veze) između korisničkog agenta i njegovog servera za elektronsku poštu. Kod ovakvih konekcija sasvim je moguće da korisnik ne želi da preuzme sve poruke u svoje poštansko sanduče, naročito ne dugačke poruke sa zvučnim ili video materijalom. Dodatne informacije o protokolu IMAP možete da potražite na zvaničnoj web lokaciji ovog protokola [IMAP 2004].

Elektronska pošta zasnovana na Webu

Svakim danom sve je veći broj korisnika koji svoju elektronsku poštu šalju i primaju kroz čitače Weba. Ovu mogućnost pokrenula je kompanija Hotmail sredinom 1990-ih, a danas ona postoji na većini portala, univerziteta i većih kompanija. Kod ove usluge korisnički agent je običan čitač Weba, a korisnik sa svojim udaljenim poštanskim sandučetom komunicira preko protokola HTTP. Kada primalac, kao Što je Bob, želi da pristupi svojoj elektronskoj pošti, poruka se iz njegovog poštanskog sandučeta, umesto protokolima POP3 ili IMAP, šalje protokolom HTTP. Isto tako, kada pošiljalac kao stoje, na primer, Alis, želi da pošalje elektronsku poštu, ponika se iz njenog Čitača ka serveru za elektronsku poštu takođe šalje protokolom HTTP (umesto protokolom SMTP).

Ovakav način pristupa elektronskoj pošti posebno je podesan za korisnike koji su u stalnom pokretu. Dovoljno je da pristupe nekom čitaču Weba i moći će da pročitaju svoje poruke. Ovaj čitač može da se nalazi u Internet kafeu, kod prijatelja, na

PDA računaru, u hotelskoj sobi koja ima Web TV itd. Kao i kod protokola IMAP, korisnici elektronske pošte zasnovane na Webu mogu da organizuju svoje poruke u hijerarhiji direktorijuma na udaljenom serveru. U stvari, mnoge implementacije ovakve elektronske pošte koriste upravo IMAP servere kako bi omogućile funkcionalnost direktorijuma. U ovom slučaju, pristup direktorijumima i porukama ostvaruje se pomoću skriptova koji se izvršavaju na HTTP serveru; ovi skriptovi za komunikaciju sa IMAP serverima koriste upravo protokol IMAP.

2.5 DNS - Internetova usluga direktorijuma

Ljudi se međusobno identifikuju na mnogo načina. Primera radi, možemo se identifikovati na osnovu imena na našim krštenicama, zatim na osnovu jedinstvenih matičnih brojeva ili, recimo, na osnovu brojeva registarskih tablica naših automobila, lako svaki od ovih identifikatora može da se koristi za identifikovanje ljudi, u određenim kontekstima neki od njih su podesniji od drugih. Računari u agenciji IRS (poreska agencija u SAD) daju prednost brojevima osiguranja građana koji su fiksne dužine, u odnosu na njihova imena na krštenicama. Nasuprot tome, ljudima su, od raznih identifikacionih brojeva, mnogo zgodnja imena. (Možete li da zamislite da neko kaže „Zdravo, ja sam 132-67-9875. Ovo je moj suprug - 178-87-1146.“)

Poput ljudi i računari na Internetu mogu da se identifikuju na više načina. Jedan od načina za njihovo identifikovanje predstavljaju imena matičnih računara. Imena računara - cnn . com, www.ydlio.com, gaia . cs . umass . edu i surf . eurecom . fr - su mnemonička, tako da ih ljudi razumeju i daju im prednost. Međutim, imena računara pružaju veoma šture ili gotovo nikakve informacije o lokaciji datog računara u okviru Interneta. (Ime surf . eurecom . fr koje se završava sa .fr govori nam satno to da se računar verovatno nalazi u Francuskoj i ništa više.) Osim toga, budući da imena računara mogu da se sastoje od alfanumeričkih znakova promenljive dužine, ruteri bi imali velike teškoće prilikom njihove obrade. Upravo zato, računari se identifikuju i IP adresama.

O IP adresama ćemo nešto detaljnije govoriti u poglavljiju 4, a sada samo nekoliko korisnih informacija. IP adresa ima četiri bajta i striktnu hijerarhijsku strukturu. U uobičajenoj IP adresi (na primer, 121.7.106.83) između bajtova koji su izraženi decimalnim brojevima od 0 do 255 nalaze se tačke. Za IP adresu se kaže da su hijerarhijske zato što čitajući je sleva udesno dolazimo do sve konkretnijih informacija o lokaciji nekog računara u okviru Interneta (u kojoj je mreži). Na isti način, Čitajući poštansku adresu odozdo naviše, stižemo do sve konkretnijih informacija o primaocu.

2.5.1 Usluge koje obezbeduje DNS

Upravo ste videli da matični računar može da se identifikuje na dva načina - imenom i IP adresom. Ljudima su bliskija mnemonička imena računara, dok ruterima više odgovaraju hijerarhijski uredene IP adrese fiksne dužine. Ova dva suprotstavljena zahteva pomirena su uslugom direktorijuma koji imena računara prevodi u IP adrese. Upravo to je i osnovni zadatak Internetove usluge DNS (Domain Name System). Usluga DNS može da se definiše kao (1) distribuirana baza podataka koja

se implementira u hijerarhiju **DNS servera** i (2) protokol aplikacijskog sloja koji omogućava računarama pretraživanje distribuirane baze podataka. DNS serveri su obično UNIX računari pod Berkeiev Internet Name Domain (BIND) softverom [BDND 2004]. Protokol DNS funkcioniše preko protokola UDP i koristi port 53.

DNS uslugu obično upotrebljavaju drugi protokoli aplikacijskog sloja - HTTP, SMTP i FTP - i to za prevodenje imena računara koje upisuju korisnici u odgovarajuće IP adrese. Kao primer zamišljamo situaciju u kojoj čitač (odnosno, HTTP klijent) koji se izvršava na računaru nekog korisnika zatraži URL `www. someschool . edu/index. html`. Da bi mogao da pošalje HTTP zahtev veb serveru `www. someschool. edu`, korisnički računar najpre mora da dođe do njegove IP adrese. To se događa na sledeći način.

1. Na računaru ovog korisnika istovremeno se izvršava i klijentska strana DNS aplikacije.
2. Čitač iz URL-a izvlači ime računara `www. someschool. edu` i predaje ga klijentskoj strani DNS aplikacije.
3. DNS klijent šalje ime računara čija je adresa potrebna DNS serveru.
4. Nakon toga DNS klijent u odgovoru dobija traženu IP adresu.
5. Korisnikov čitač zatim uspostavlja TCP konekciju sa procesom HTTP servera koji se nalazi na daljoj IP adresi.

Iz ovog primera vidimo da DNS usluga u sve Internet aplikacije koje je koriste uvodi dodatno (nekada i prilično značajno) kašnjenje. Srećom, kao što ćete ubrzati i sami videti, tražena IP adresa najčešće se nalazi u keš memoriji „obližnjeg“ DNS servera, što i te kako pomaže u smanjivanju intenziteta DNS saobraćaja i skraćenju kašnjenja koje prouzrokuje ova usluga.

Pored prevodenja imena matičnih računara u adrese, DNS obezbeđuje i sledeće usluge:

- ♦ **Dodeljivanje pseudonima.** Računar komplikovanog imena može imati jedan pseudonim ili više njih. Primera radi, ime računara `relayl .west-coast. enterprise . com` moglo bi da ima, recimo, dva pseudonima kao što su `enterprise. com` i `www. enterprise. com`. U ovom slučaju za ime računara `relayl.west-coast.enterprise.com` kaže se da predstavlja **kanoničko ime**. Pseudonimi imena računara, ukoliko postoje, obično su u većoj meri mnemonička od kanoničkih imena. DNS uslugu može da aktivira i aplikacija koja iz pseudonima i IP adrese računara želi da izvuče kanoničko ime.
- ♦ **Pseudonimi servera za elektronsku poštu.** Iz sasvim očiglednih razloga bitno je da adrese elektronske pošte budu mnemoničke. **Kada** bi, primera radi, Bob imao korisnički nalog na Hotmailu, njegova adresa elektronske pošte mogla bi da glasi samo `bob@hotmail. com`. Međutim ime Hotmailovog poštanskog servera je mnogo složenije i mnogo manje mnemoničko od imena `hotmail. com` (njegovo kanoničko ime moglo bi, recimo, da glasi `relayl. west-coast .hotmail. com`). DNS usluga može, dakle, da pozove i aplikaciju za elektronsku poštu kojoj su potrebni kanoničko ime i IP adresa matičnog računara

PRINCIPI U PRAKSI

DNS: KRITIČNE MREŽNE FUNKCIJE PUTEM KLUENTSKO-SERVERSKE PARADIGME

Poput protokola HTTP, FTP i SMTP, DNS je takođe protokol aplikacijskog sloja s obzirom no to da se (1) izvršava između komunicirajućih krajnjih točaka oslanjajući se na klijent-serversku paradigmu i da se [2] u transportu DNS poruka između komunicirajućih krajnjih sistema oslanja na transportni protokol koji se nalazi u njegovoj osnovi. Međutim, u drugom smislu uloga DNS usluge je potpuno različita od uloge VVeba, transfera datoteka ili aplikacija za elektronsku poštu. Za razliku od navedenih aplikacija, DNS usluga nije aplikacija sa kojom korisnik ostvaruje direktnu interakciju. Umesio toga, DNS usluga obezbeđuje jednu od ključnih funkcija u vezi sa Internetom - za potrebe korisničkih aplikacija i ostalog softvera na Internetu ona prevodi imena matičnih računara u odgovarajuće IP adrese. Podsećamo vas da smo u odeljku 1.2 rekli da se najveći deo složenosti arhitekture Interneta nalazi na „periferiji“ ove mreže. Usluga DNS, koja implementira izuzetno važan proces prevodenja imena u adrese koristeći klijente i servere koji se nalaze na periferiji mreže, predstavlja još jednu potvrdu izrečene tvrdnje.

Čiji pseudonim ima. Štaviše, zapis MX (objašnjen u nastavku) omogućava da server za elektronsku poštu i veb server neke kompanije imaju ista imena (pseudonime); primera radi, i jedan i drugi server neke kompanije mogu imati pseudonim `enterprise. com`.

♦ **Distribucija opterećenja.** DNS usluga se u sve većoj meri koristi za distribuiranje opterećenja između nekoliko rezervnih servera. Stoji Čest slučaj kod veb servera. Veoma posećene lokacije, kao što je, recimo `cnn . com`, imaju kopije na nekoliko servera od kojih se svaki izvršava na drugom krajnjem sistemu sa drugaćijom IP adresom. Kod rezervnih veb servera, *skup* IP adresa se, prema tome, vezuje zajedno kanoničko ime. Ovaj skup IP adresa nalazi se u bazi podataka DNS-a. Kada klijent učini DNS upit za ime koje je preslikano na ovaj skup adresa, server odgovara sa čitavim skupom IP adresa, s tim što u svakom svom odgovoru rotira njihov redosled. S obzirom na to da klijent obično svoju HTTP poruku sa zahtevom šalje na prvu IP adresu na spisku, DNS usluga na ovaj način obezbeđuje raspodelu saobraćaja između servera sa kopijama. DNS rotacija se koristi i kod elektronske pošte kako bi veći broj servera za elektronsku poštu mogao da ima isto ime. Odnedavno neke kompanije, među kojima i Akamai [Akamai 2004], sofisticiranim korišćenjem DNS usluge ostvaruju distribuciju veb sadržaja (procitajte poglavlje 7).

Usluga DNS definisana je u dokumentima RFC 1034 i RFC 1035, a njena ažuriranja nalaze se u još nekoliko dodatnih RFC dokumenata. Reč je o veoma složenom sistemu. Čije smo ključne aspekte samo dotakli na ovom mestu. Dodatne informacije o njemu možete potražiti u navedenim RFC dokumentima, zatim u knjizi koju su napisali Abic i Liu [Abitz 1993], kao i u radu [Mockapetris 1988].

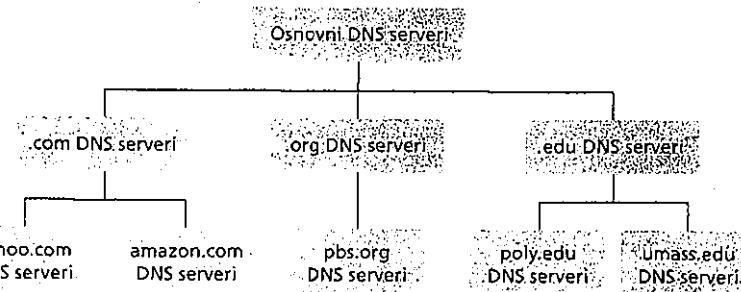
2.5.2 Prikaz načina rada usluge DNS

Sada ćemo vam iz nešto uopštenije perspektive prikazati način rada usluge DNS. U izlaganju koje sledi usredstvilićemo se na uslugu prevođenja imena računara u IP adresu.

Prepostavimo da neka aplikacija (recimo, čitač Weba ili elektronske pošte) koja se izvršava na računaru nekog korisnika treba da prevede ime računara u IP adresu. Ova aplikacija će najpre da pozove klijentsku stranu usluge DNS, navodeći joj ime matičnog računara koje treba da se prevede. (Na većini računara pod UNIX-om proces prevodenja se pokreće tako što aplikacija upućuje funkcionalni poziv gethost-byname(). U odeljku 2.7 pokazaćemo vam na koji način Java aplikacija može da pozove uslugu DNS.) Klijentska strana usluge DNS u korisnikovom računaru zatim preuzima inicijativu i šalje poruku sa upitom u mrežu. Sve DNS poruke - i upiti i odgovori - šalju se u okviru UDP datagrama kroz port 53. Nakon izvesnog vremena, što mogu da budu milisekunde, ali i desetine sekundi, DNS u korisnikovom računaru prima poruku sa odgovorom u kojoj je navedeno traženo preslikavanje. Ovo preslikavanje se potom prosleđuje aplikaciji koja ga je i zatražila. Dakle, iz perspektive aplikacije koja ga poziva, DNS je crna kutija koja obezbeđuje jednostavnu uslugu prevodenja. Međutim, ova crna kutija je, u stvari, veoma složena i obuhvata veliki broj DNS servera koji su raspoređeni širom sveta, ali i protokol aplikacijskog sloja kojim se precizira komunikacija između DNS servera i računara koji šalju upite.

U pojednostavljenoj varijanti, DNS usluga činio bi jedan DNS server u kome bi se nalazila sva preslikavanja. U ovakvom centralizovanom dizajnu klijenti bi sve upite upućivali jednom serveru koji bi direktno odgovarao svim svojim klijentima. Iako je jednostavnost ovakve konцепциje privlačna, ona je jednostavno nepodesna za potrebe današnjeg Interneta i sve većeg broja računara. U nastavku navodimo razloge koji objašnjavaju zašto ovakav centar 1 i zo van i dizajn nije poželjan.

- ◆ Mreža se oslanja na jednu tačku. U slučaju kvara DNS servera, cela mreža (Internet) postaje neupotrebljiva.
 - ◆ Intenzitet saobraćaja. Jedan DNS server morao bi da se nosi sa svim DNS upitim (svi HTTP zahtevi i elektronska pošta stotina miliona računara).
 - ◆ Udaljena centralizovana baza podataka. Jedan DNS server ne bi mogao da bude „blizu“ svih klijenata. Ukoliko bismo ga postavili u Njujorku, svi upiti iz Australije bi morali da putuju na drugi kraj sveta - možda sporim i zagušenim vezama. Kašnjenja bi u tom slučaju bila zaista značajna.
- Možemo zaključiti da centralizovana baza podataka najednom serveru imena *nema mogućnost jednostavnog i pouzdanog proširivanja*. Zato je usluga DNS samim svojim dizajnom distribuirana i predstavlja veoma lep primer implementiranja distribuirane baze podataka na Internetu.
- ◆ Održavanje. Jedan DNS server morao bi da čuva zapise za sve računare na Internetu. Ovakva centralizovana baza podataka bi bila ogromna i moralna bi veoma često da se ažurira (za svaki novi računar u mreži).



Slika 2.18 ♦ Deo hijerarhiјe DNS servera

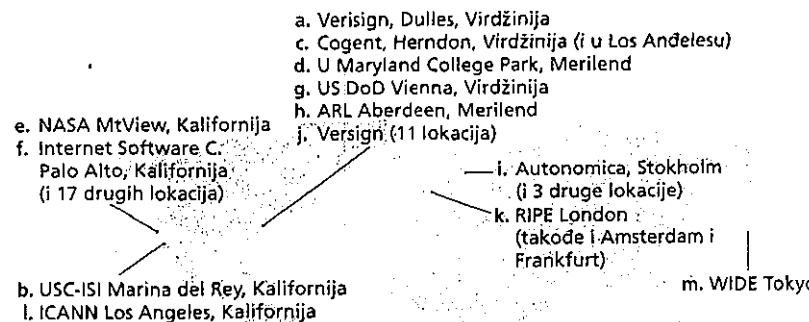
Distribuirana, hijerarhijska baza podataka

Da bi mogla da se izbori sa svim zabevima, DNS usluga koristi veliki broj DNS servera koji su hijerarhijski organizovani i raspoređeni širom sveta. Nijedan od ovih servera nema sva preslikavanja za sve računare na Internetu. Umesto toga, preslikavanja su distribuirana na većem broju DNS servera. Uopšteno govoreći, postoje tri tipa DNS servera - lokalni DNS serveri, DNS serveri domena najvišeg nivoa (*top-level domain, TLD*) i DNS serveri od autoriteta koji su organizovani u hijerarhiju koja je prikazana na slici 2.18. Da biste razumeli interakciju ove tri klase servera, prepostavimo da je DNS klijentu potrebna IP adresa za ime www. amazon. com. Evo Šta se u ovoj situaciji dešava. Klijent najpre kontaktira sa nekim od osnovnih DNS servera koji mu vraćaju IP adresu TLD servera na vrhu domena .com. Nakon toga, klijent se obraća jednom od ovih TLD servera koji mu vraća IP adresu servera od autoriteta za domen amazon. com, od koga će u trećem koraku dobiti IP adresu za ime www. amazon. com. Ovaj postupak DNS pretraživanja ćemo vam uskoro prikazati malo detaljnije, ali pre toga pogledajmo izbliza ove tri klase DNS servera.

- ◆ Osnovni DNS serveri. Na Internetu postoji trinaest osnovnih DNS servera (imaju oznake od A do M), od kojih se većina nalazi u Severnoj Americi. Na slici 2.19 možete da vidite mapu osnovnih DNS servera iz februara 2004. godine, a aktuelnu listu osnovnih DNS servera možete da potražite u dokumentu [Root-servers 2004]. Iako smo svaki od trinaest osnovnih DNS servera označili kao da je u pitanju jedan računar, „server“ u stvari predstavlja klaster servera formiran zbog bolje bezbednosti i pouzdanosti.
- ◆ Serveri domena najvišeg nivoa (TLD serveri). Ovi serveri odgovorni su za domene najvišeg nivoa, kao što su domeni .com, .org, .net, .edu i .gov, kao i za sve domene najvišeg nivoa raznih zemalja - .uk, .fr, .ca i .jp. U vreme pisanja ove knjige (leto 2004), kompanija Network Solutions vodila je računa o TLD serverima domena najvišeg nivoa .com, dok je kompanija Educause bila zadužena za TLD servere domena najvišeg nivoa .edu.

♦ DNS serveri od autoriteta. Svaka organizacija koja ima javno dostupne računare (recimo, veb servere ili servere za elektronsku poštu) na Internetu, mora da obezbedi javno dostupne DNS zapise u kojima se imena ovih računara preslikavaju u IP adrese. Ovi DNS zapisi čuvaju se upravo na DNS serverima od autoriteta tih organizacija. Jedna organizacija može da ima sopstveni DNS server od autoriteta, ili može da plati da se ovi zapisi čuvaju na takvom DNS serveru nekog posrednika za Internet usluge. Univerziteti i velike kompanije najčešće implementiraju i održavaju sopstvene primarne i sekundarne (rezervne) DNS servere od autoriteta.

Osnovni, TLD i DNS serveri od autoriteta pripadaju hijerarhiji DNS servera koja je prikazana na slici 2.18. Međutim, postoji još jedan veoma važan tip DNS servera - reč je o lokalnim DNS serverima. Lokalni DNS serveri se ne uklapaju sasvim u navedenu hijerarhiju servera, ali bez sumnje zauzimaju centralno mesto u DNS arhitekturi. Svaki posrednik za Internet usluge (ISP) - što može da bude univerzitet, neka akademска ustanova, kompanija ili rezidencijalni ISP - ima svoj lokalni DNS server (naziva se i podrazumevani DNS server). Kada se neki računar poveže sa posrednikom za Internet usluge datog korisnika, on od njega dobija IP adresu jednog ili više lokalnih DNS servera (obično kroz protokol DHCP o kome ćemo govoriti u odeljku 5.4.3). IP adresu svog lokalnog DNS servera možete da vidite na prozoru za praćenje mrežnog statusa operativnih sistema Windows ili UNIX. Lokalni DNS server obično se nalazi „blizu“ klijenta. Kod pristupa Internetu kroz

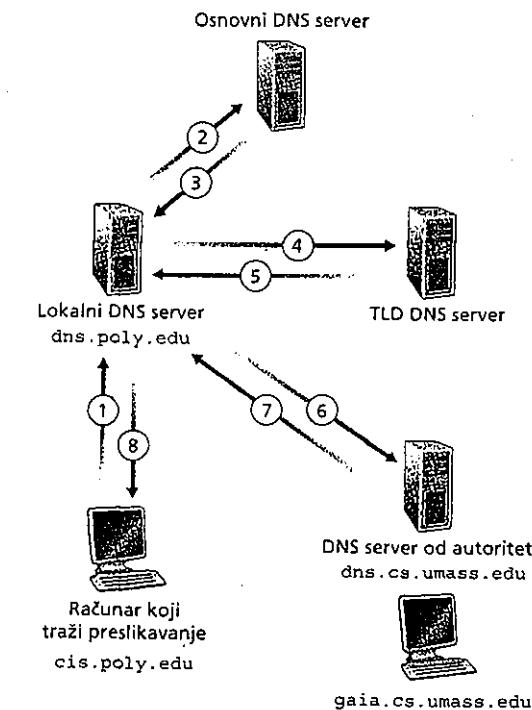


Slika 2.19 ♦ Osnovni DNS serveri početkom 2004. godine (ime, organizacija, lokacija)

mrežu neke kompanije, ovaj server može da se nalazi i u istom LAN-u kao i računar koji šalje upit. Kod rezidencijalnog pristupa Internetu DNS server je od klijentskog računara udaljen svega nekoliko ruta.

Kada klijent izvrši DNS upit, taj upit se šalje lokalnom DNS serveru koji se ponaša kao proxy server i prosledjuje ga u DNS hijerarhiju.

Predimo sada na jedan jednostavan primer. Prepostavimo daje računaru cis . poly.edu potrebna IP adresa računara gaia . cs . umass . edu. Такође ćemo prepostaviti i to daje lokalni DNS server Politehničkog univerziteta dns . poly . edu, a daje, za računar gaia . cs . umass . edu, DNS server od autoriteta dns . umas . edu. Kao što možete da vidite na slici 2.20, računar cis . poly . edu najpre šalje DNS poruku sa upitom svom lokalnom serveru imena - serveru dns . pol y . edu. U ovom upitu navedeno je ime računara (gaia.cs.umass.edu) koje treba da se prevede u IP adresu. Lokalni DNS server prosledjuje ovaj upit osnovnom serveru imena koji, videvši sufiks .edu, vraća lokalnom DNS serveru listu



Slika 2.20 ♦ Interakcija DNS servera

IP adresa TLD servera koji su odgovorni za domen .edu. Nakon loga, lokalni DNS server poruku sa upitom šalje jednom od ovih TLD servera.

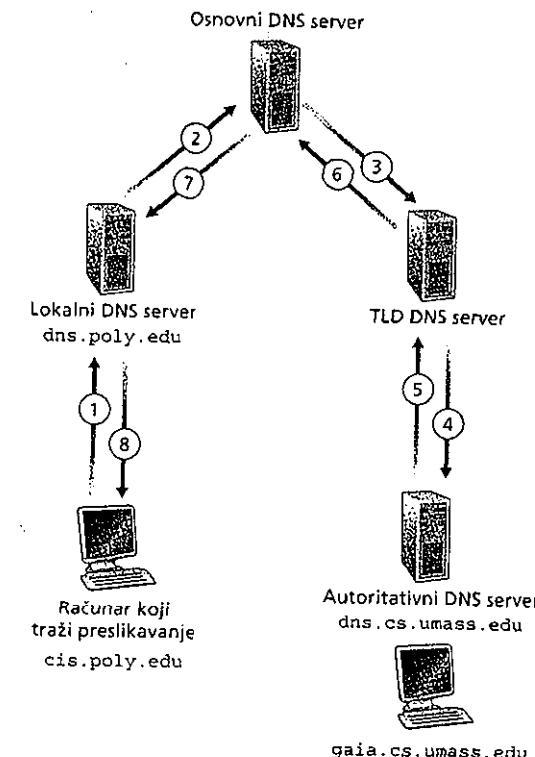
TLD server zatim obraća pažnju na sufiks umass . edu i odgovara IP adresom DNS servera od autoriteta na Univerzitetu u MasaČusetsu - dns . umass . edu. Konačno, lokalni DNS server istu poruku sa upitom šalje direktno serveru dns . umass . edu od koga dobija IP adresu računara gaia . cs . umass . edu. Skrećemo vam pažnju na to daje za dobijanje IP adrese jednog računara poslalo čak osam DNS poruka - četiri upita i četiri odgovora. Ipak, uskoro ćete videti na koji način to može da se smanji DNS keširanjem.

U ovom primeru pretpostavili smo da TLD DNS server zna IP adresu DNS servera od autoriteta za imena traženog matičnog računara. Ova pretpostavka, međutim, ne mora da bude tačna. Moguće je da TLD DNS server zna samo IP adresu posredujućeg DNS servera koji će znati IP adresu DNS servera od autoriteta za traženo ime. Pretpostavimo sada da Univerzitet u Masačusetsu ima DNS server univerziteta po imenu dns . umass . edu. Pretpostavimo i to da svaki departman ovog univerziteta ima vlastiti DNS server i da svaki od njih ima autoritet za one računare koji se nalaze u datom departmanu. U ovom slučaju, kada posredujući DNS server dns . umass . edu primi upit za računar čije se ime završava sa cs . umass . edu, on DNS serveru dns . poly . edu vraća IP adresu DNS servera dns . cs . umass . edu koji je zadužen za upite koji se odnose na računare čije se ime završava sa cs . umass . edu. Lokalni DNS server dns . poly . edu zatim šalje upit DNS serveru od autoriteta, koji mu odgovara traženim preslikavanjem. U ovom slučaju razmeni se ukupno deset DNS poruka.

U primeru sa slike 2.20 korišćeni su rekurzivni i iterativni upiti. Upit koji je klijent cis . poly . edu poslao serveru dns . poly . edu je rekurzivan s obzirom na to daje klijent od servera zatražio da u njegovo ime pronađe traženo preslikavanje. Međutim, sledeća tri upita su iterativna zato što se svi odgovori šalju direktno serveru dns . poly . edu. U teoriji, svaki DNS upit može da bude iterativan ili rekurzivan. Primera radi, na slici 2.21 prikazanje lanac DNS upita u kome su svi oni rekurzivni. U praksi, međutim, upiti obično prate šablon koji je ilustrovan na slici 2.20: prvi upit je rekurzivan, a svi ostali iterativni,

DNS keširanje

Dosadašnjim razmatranjem još uvek nismo dotakli jednu veoma važnu funkciju DNS usluge - DNS keširanje. U stvarnosti ova usluga se u izuzetnoj meri oslanja na keširanje koje skraćuje kašnjenja i smanjuje broj DNS poruka u mreži. Sam koncept keširanja je krajnje jednostavan. Kada u lancu upita DNS server primi DNS odgovor (na primer, preslikavanje imena nekog matičnog računara u IP adresu), server ovu informaciju skladišti u svojoj lokalnoj memoriji. Primera radi, lokalni DNS server dns . poly . edu sa slike 2.20 može da kešira sve informacije koje primi od nekog drugog DNS servera u lancu upita. Ukoliko ponovo dobije upit koji se odnosi na keširano preslikavanje imena računara u IP adresu, DNS server može odmah da



Slika 2.21 ♦ Rekurzivni DNS upiti

odgovori, čak i ako on nije autoritet za dati matični računar. Keširani zapis se nakon isteka određenog vremena (najčešće dva dana) obično briše, zato što računari i preslikavanja imena u IP adresu nisu trajni.

Primera radi, pretpostavimo da je računar [apricot.poly.edu](#) serveru dns . poly . edu poslao DNS upit tražeći IP adresu računara [cnn . com](#). Такође ћемо pretpostaviti и то даје неколико сати касније један други računar sa Poliehni-čkog univerziteta, [recimo kiwi . poly . fr](#), поново poslao isti DNS upit. Zahvaljujući keširanju, lokalni DNS server može odmah da pošalje IP adresu računara [cnn . com](#) bez prethodnog kontaktta sa drugim DNS serverima. Lokalni DNS server takođe može da kešira IP adresu TLD servera, što mu omogućava da u lancu upita preskoči osnovne DNS servere (ovo se često dešava),

2.5.3 DNS zapisi i poruke

DNS serveri koji zajedno implementiraju distribuiranu DNS bazu podataka Čuvaju zapise o resursu (*resource record, RR*) koji se odnose na preslikavanja imena računara u IP adresu. U svakoj DNS poruci sa odgovorom nalazi se jedan ili više ovakvih zapisa o resursu. U ovom i sledećem odeljku nalazi se sažet prikaz zapisa o resursu i poruka DNS usluge. Detaljnije informacije možete pronaći u knjizi [Abitz 1993] ili u RFC dokumentima koji se odnose na uslugu DNS [RFC 1034; RFC 1035]. Zapis o resursu ima sledeća četiri polja:

(Name, Value, Type, TTL)

Polje TTL predstavlja rok trajanja zapisa; ovim poljem se određuje kada se dati zapis o resursu uklanja iz keširane memorije. U primerima zapisa koje navodimo u produžetku zanemarili smo polje TTL. Značenja polja Name i Value zavise od polja Type:

- ◆ Ukoliko je Type=A, onda je Name ime matičnog računara, a Value njegova IP adresa. Dakle, zapis tipa A obezbeđuje standardno preslikavanje imena računara u IP adresu. Primera radi, zapis (relayl. bar. f oo . com, 145.37.93.126, A) pripada zapisima tipa A.
- ◆ Ako je Type=NS, onda je polje Name domen (na primer, f oo . com), a Value ime DNS servera od autoriteta koji može da obezbedi IP adresu računara u tom domenu. Ovaj zapis se koristi za rutiranje DNS upita kroz komunikacioni lanac. Na primer, zapis (f oo . com, dns . f oo . com, NS) pripada tipu NS.
- ◆ Ukoliko je Type=CNAME, polje Value je kanoničko ime računara Čiji je pseudonim Name. Dakle, pomoću ovog zapisa računari mogu da dodu do kanoničkog imena za ime određenog računara. Primera radi, zapis (f oo . com, relayl. bar. foo. com, CNAME) predstavlja zapis tipa CNAME.
- ◆ Ukoliko je Type=MX, polje Value predstavlja kanoničko ime servera elektronske pošte čiji je pseudonim Name. Zapis (foo. com. mail. bar. foo. com, MX) pripada zapisima tipa MX. Ovaj tip zapisa omogućava da serveri elektronske pošte imaju jednostavne pseudonime. Skrećemo vam pažnju na to da korištenje MX zapisa omogućava jednoj kompaniji da koristi isti pseudonim za svoj server elektronske pošte i za neki drugi server (recimo, veb server). Kada mu je potrebno kanoničko ime servera elektronske pošte, DNS klijent u upitu traži MX zapis, a kada mu je potrebno kanoničko ime nekog drugog servera, klijent traži zapis tipa CNAME.

DNS server od autoriteta u svojoj bazi podataka ima zapise tipa A za sve računare koji su u njegovoj nadležnosti. (Čak i ako on nije autoritet za neki računar, DNS server u svojoj keširanoj memoriji može da sadrži zapis tipa A koji se odnosi na datu' računar.) Ukoliko DNS server nije autoritet za određeni računar, on će u svojoj bazi

podataka imati zapis tipa NS koji se odnosi na domen u kome se dati računar nalazi; osim toga, on će imati i zapis tipa A sa IP adresom DNS servera koji je naveden u polju Va 1 ue zapisa NS. Primera radi, pretpostavimo da edu TLD server nije autoritet za računar gaia. cs .umass .edu. U tom slučaju ovaj server imaće zapis koji se odnosi na domen kome pripada ovaj računar - cs .umass .edu - na primer, (umass .edu, dns .umass .edu, NS). Osim toga, edu TLD DNS server imaće i zapis tipa A koji preslikava ime DNS servera dns .umass .edu u IP adresu - na primer, (dns .umass .edu, 128.119.40.111, A).

DNS poruke

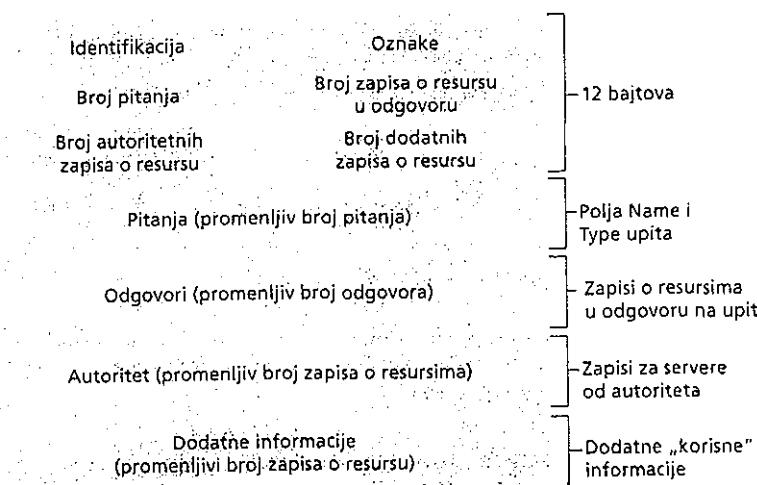
Na početku ovog dela pomenuli smo DNS upite i odgovore. Ovo su ujedno i jedine vrste poruka u okviru DNS-a. Kao što možete da vidite na slici 2.22, DNS upiti i odgovori imaju isti format. U nastavku objašnjavamo značenje polja DNS poruka.

- ◆ Prvih 12 bajtova predstavljaju *odeljak zaglavљa*, koji ima nekoliko polja. Prvo polje je 16-bitni broj koji identifikuje upit. Ovaj identifikator se kopira u odgovor. Što omogućava klijentu da upoređe poslate upite sa primljenim odgovorima. U polju za oznake stoji nekoliko oznaka. Jednobitnom oznakom lipa upit/odgovor naznačava se da lije u pitanju upit (0) ili odgovor (1). Jednobitna oznaka autoriteta stavlja se u odgovor ukoliko je DNS server server od autoriteta za traženo ime. Jednobitna oznaka poželjnosti rekurzivnog upita postavlja se kada klijent (računar ili DNS server) želi da DNS server, ukoliko nema traženi zapis, izvrši rekurzivni upit. Ako podržava rekurzivne upite, DNS server u odgovoru postavlja jednobitnu oznaku o ostvarivosti rekurzivnog upita. U zaglavljtu postoje i Četiri polja tipa „number of“ kojima se naznačava broj pojavljivanja Četiri tipa odeljaka za „podatke“ koji slede iza zaglavlja.
- ◆ *Odeljak za pitanje* sadrži informacije o aktuelnom upitu. U ovom odeljku se nalaze (I) polje sa imenom na koje se upit odnosi i (2) polje za tip u kome se navodi tip pitanja - na primer, adresa računara povezuje se sa tipom A, dok se ime servera za elektronsku poštu povezuje sa tipom MX.
- ◆ U odgovoru DNS servera *odeljak za odgovor* sadrži zapise o resursu koji se odnose na traženo preslikavanje. Podsećamo vas da svaki zapis o resursu ima polja Type (na primer, A, NS, CNAME ili MX), Value i TTL. Budući da isto ime matičnog računara može da ima više IP adresa, u odeljku za odgovor jedne poruke može da se nade više zapisa o resursu (podsećamo vas na veb servere sa kopijama o kojima smo govorili ranije).
- ◆ *Odeljak za autoritet* sadrži zapise ostalih servera od autoriteta.
- ◆ U *dodatačnom odeljku* nalaze se ostali korisni zapisi. Primera radi, u polju za odgovor u uzvratnoj poruci na MX upit nalazi se zapis o resursu u kome je navedeno kanoničko ime računara ili servera za elektronsku poštu.

Da li biste želeli da poruku sa TNS upitom pošaljete nekom DNS serveru direktno sa računara na kome radite? Ovo je prilično jednostavno ukoliko koristite program nslookup koji je sastavni deo većine Windows i UNIX platformi. Primera radi, na računaru pod operativnim sistemom Windows otvorite komandni prozor i upisivanjem „nslookup” pokrenite ovaj program. Nakon toga DNS upit možete da pošaljete bilo kom DNS serveru (osnovnom, TLD ili serveru od autoriteta). Kada od DNS servera primi poruku sa odgovorom, nslookup prikazuje zapise koji su dobijeni u odgovoru (u formatu koji može da se čita). Osim što ovaj program možete da pokrenete sa svog računara, isto to možete da učinite sa neke od mnogih veb lokacija koje omogućavaju njegovo daljinsko korišćenje. (U bilo kom pretraživaču upišite „nslookup” i naći ćete se na nekoj od ovih lokacija.)

Ubacivanje zapisa u DNS bazu podataka

U čelom ovom izlaganjugovorili smo samo o načinima pribavljanja informacija iz DNS baze podataka, tako daje moguće da se sada pitate na koji način te informacije uopšte dospevaju u nju. Pogledajmo kako to izgleda na konkretnom primeru. Prepostavimo da ste upravo oformili kompaniju po imenu Network Utopia. Prvo što ćete sasvim sigurno želiti da učinite jeste da u registru registrijete ime domena networkutopia . com. Registrar je komercijalni entitet koji proverava jedinstvenost imena domena, unosi ime u DNS bazu podataka (uskoro ćete videti kako) i za sve svoje usluge dobija malu nadoknadu. Do 1999. godine registrar Network Solutions imao je monopol na registraciju domena .com, .net i .org.



Slika 2.22 ♦ Format DNS poruke

Međutim, danas postoji mnogo registrara koji se bore da pridobiju korisnike, a sve njih akrediluje korporacija ICANN (*Internet Corporation for Assigned Names and Numbers*). Kompletnu listu akreditovanih registrara možete da potražite na lokaciji <http://www.internic.net>.

Prilikom registracije domena networkutopia . com kod nekog registrara potrebno je da im date imena i IP adrese primarnog i sekundarnog DNS servera. Prepostavimo da su imena dns1. networkutopia . com, dns2 . networkutopia . com, a adrese 212.212.212.1 i 212.212.212.2. Za svaki od ova dva DNS servera od autoriteta registrar ubacuje NS i A zapis u TLD servere domena .com. Konkretno, za primarni server od autoriteta za domen networkutopia . com registrar ubacuje sledeća dva zapisa u DNS sistem:

(networkutopia.com, dns1.netwprkutopia.com, NS
{dns1.netwprkutopia.com, 212.212.212.1, A})

Pored toga, neophodno je i da se zapis tipa A za veb server www. networkutopia . com i zapis tipa MX za server elektronske pošte mail. networkutopia . com unesu u DNS servere od autoriteta. (Sve donedavno sadržaj svakog DNS servera konfigurisan je statički - recimo, pomoću konfiguracione datoteke koja je dobitljena u Sistem Manageru. Međutim, protokol DNS sada ima opciju UPDATE koja omogućava dinamičko dodavanje i brisanje podataka iz baze podataka putem DNS poruka. Ovakvo dinamičko ažuriranje opisano je u dokumentu RFC 2136.)

Kada uradite sve što smo ovde naveli, ljudi će moći da posete vašu veb lokaciju i pošalju elektronsku poštu zaposlenima u vašoj kompaniji. Izlaganje o DNS-u završćemo proverom istinitosti upravo izrečene tvrdnje. Osim toga, ova provera će vam pomoći da utvrđite znanja o ovoj usluzi. Prepostavimo da Alisa iz Australije želi da poseti veb lokaciju www. networkutopia . com. Kao što smo već rekli, njen računar će najpre da pošalje DNS upit njenom lokalnom DNS serveru. Ovaj lokalni DNS server se odmah zatim obraća TLD .com serveru. (Ukoliko nema kaštranu adresu TLD .com servera, lokalni DNS server će najpre morati da se obrati nekom osnovnom DNS serveru.) U TLD serveru nalaziće se zapisi o resursima tipa A i NS budući da ih je registrar već uneo u sve TLD .com servere. TLD server zatim u svom odgovoru Alisinom lokalnom DNS serveru šalje dva zapisa o resursima. Nakon toga lokalni DNS server šalje DNS upit na adresu 212.212.212.1 tražeći zapis tipa A koji odgovara imenu www. networkutopia . com. U ovom zapisu nalazi se IP adresa željenog veb servera - recimo, 212.212.71.4 koju lokalni DNS server vraća Alisinom računaru. Nakon toga čitač sa Alisinog računara može da inicira TCP konekciju sa računaram 212.212.71.4 i pošalje mu HTTP zahtev. Eto, izgleda da krstarenje Webom i nije toliko na prvi pogled izgleda.

2.6 P2P razmena datoteka

U vreme pisanja ove knjige (početak 2004. godine) P2P razmena datoteka pravila je više saobraćaja na Internetu od svih drugih aplikacija, uključujući tu i sam Web. Prema tome, kada se ima u vidu isključivo obim saobraćaja, P2P razmena datoteka može se smatrati najvažnijom Internet aplikacijom. Savremeni sistemi za P2P razmenu datoteka koriste se, ne sama za razmenu MP3 muzike (datoteke obično imaju između 3 i 8 MB), već i za razmenu video materijala (od 10 do 1000 MB), softvera, dokumenata i slika. U nastavku teksta bavićemo se komunikacijom i aspektima umrežavanja P2P razmene datoteka. Međutim, ovde ima i mnogo drugih zanimljivih aspekata vezanih za bezbednost, privatnost, anonimnost, kršenje autorskih prava i intelektualnu svojinu. Radoznače čitaoce upućujemo na autore Lohraana i Klarka [Lohmann 2003; Clarke 2002] u čijim će knjigama pronaći informacije o ovim i nekim drugim aspektima ovakvog načina umrežavanja,

Pre nego što opišemo unutrašnju građu sistema za razmenu datoteka između ravnopravnih korisnika, pogledajmo kako jedan tipičan korisnik, recimo Alisa, može da ga upotrebi. Ona koristi P2P aplikacije za preuzimanje MP3 datoteka i to tako što na svom kućnom („ravnopravnom“) računaru aktivira softver za P2P razmenu datoteka a zatim se, korišćenjem ADSL veze, povezuje sa Internetom. Alisa obično noću isključuje svoj računar. Njen računar nema ime, tako da, prilikom svakog povezivanja, od ISP-a dobije novu IP adresu.

Pretpostavimo da je Alisa u ovom trenutku povezana sa Internetom i da je upravo pokrenula svoju P2P aplikaciju. Pomoću ove aplikacije ona namerava da potraži pesmu „Network Love“. Ubrzo nakon zadavanja komande za početak pretraživanja, aplikacija će joj prikazati listu svih ravnopravnih korisnika koji su trenutno povezani sa Internetom, imaju kopiju ove pesme i spremni su daje razmene. Najčešće su to obični korisnici, poput nje, koji imaju obične računare. Za svakog korisnika na ovoj listi, aplikacija može da prikaže i neke pomoćne informacije, kao što su, na primer, njegova propusna moć i procena trajanja preuzimanja datoteke. Nakon toga, Alisa će od nekog korisnika, recimo od Boba, zatražiti MP3 datoteku koja je zanima. Između njenog i Bobovog računara uspostaviće se direktna TCP konekcija kojom će datoteka biti prebačena sa jednog računara na drugi. Ukoliko Bob tokom preuzimanja iz nekog razloga prekine svoju vezu sa Internetom, Alisin P2P softver može da pokuša da ostatak datoteke preuzme od nekog drugog korisnika koji je ima. Isto tako, dok ona od Boba preuzima pesmu „Network Love“, moguće je da neki treći korisnik, na primer, Kler sa njenog računara preuzima pesmu „Network 'n' Roll“. Dakle, u mrežama ravnopravnih korisnika svaki učesnik je i konzument i dobavljač sadržaja.

P2P razmena datoteka zasniva se na paradigmi direktnе distribucije sadržaja zato što se sav saobraćaj prenosi direktno između običnih ravnopravnih korisnika, bez učešća servera trećih lica. To znači da se u P2P umrežavanju distribucija sadržaja ostvaruje iskorišćavanjem resursa (propusna moć, skladišni prostor i procesori)

ogromnog broja ravnopravnih korisnika - nekada i miliona. Drugim recima, mogućnosti proširivanja ovde su praktično neograničene.

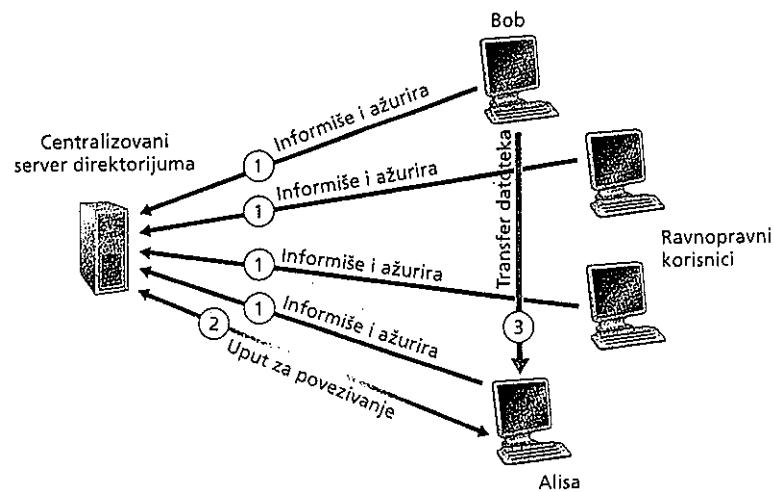
Iako transfer ovde nije centralizovan, u njemu učestvuju i serveri trećih lica. Veoma je važno reći da se u P2P razmeni datoteka ipak koristi paradigma klijent-server koja je toliko prisutna na Internetu (pročitajte odeljak 2.1.2). U ovom slučaju, korisnik koji traži datoteku ponaša se kao klijent, a korisnik sa čijeg računara se vrši transfer - kao server. Datoteka se, putem protokola za prenos datoteka, prebacuje sa računara koji se ponaša kao server na računar koji se ponaša kao klijent, S obzirom na to da ovde svaki računar može biti izabran, svi ravnopravni računari treba da budu sposobni za izvršavanje i klijentske i serverske strane protokola za transfer datoteka.

Zamislite sada situaciju u kojoj se kao protokol za P2P transfer datoteka koristi protokol HTTP (obično je zaista tako). Nadovezujući se na naš prethodni primer, kada Alisa izabere Boba za preuzimanje pesme „Network Love“, njen računar šalje Bobovom HTTP zahtev za ovu pesmu, na koji Bobov računar odgovara HTTP porukom u kojoj se ova pesma nalazi. Skrećemo vam pažnju na to da se sada Bobov računar ponaša i kao veb klijent i kao privremeni veb server. Za ovaj računar kažemo da je veb server zato što, na HTTP zahteve, obezbeđuje određene sadržaje. Atribut „privremeni“ upotrebljen je zato što ovaj računar nije neprekidno povezan sa Internetom i moguće je da pri svakom novom povezivanju ima drugačiju IP adresu.

Do sada smo vam objasnili samo lakši deo razmene datoteka između ravnopravnih računara, a to je način na koji se sadržaj direktno prebacuje sa jednog računara na drugi. Međutim, još ništa nismo rekli o načinu na koji jedan računar ustanovljava da neki drugi računari poseduju traženi sadržaj. U P2P sistemu razmene datoteka obično postoji veoma veliki broj povezanih jačunara od kojih svaki ima objekte za razmenu - MP3 muziku, video materijal, slike i programe. Ukoliko je ravnopravni računar X zainteresovan za preuzimanje nekog konkretnog objekta, on mora da zna kako da ustanovi IP adrese svih povezanih ravnopravnih računara na kojima postoji kopija datog objekta. Imajući u vidu to da se u ovakvoj mreži računari često uključuju i isključuju, to nije nimalo jednostavan problem. U nastavku teksta pomenu-ćemo tri arhitekture za lociranje sadržaja koje se koriste u različitim sistemima za P2P razmenu datoteka. Čitaoцима koje ova tema posebno zanima preporučujemo da pročitaju i istraživačke radove [Stoica 2001; Rowstron 2001; Ratnasamy 2001; Zhao 2004; Mavmounkov 2002; Garcés-Erce 2003] u kojima su ponuđena inovativna rešenja za problem lociranja sadržaja,

Centralizovani direktorijum

Jedan od najjednostavnijih pristupa u lociranju sadržaja jeste postojanje centralizovanog direktorijuma - onako kako je to učinila kompanija Napster, inače prva komercijalna kompanija koja je napravila P2P aplikaciju za razmenu MP3 muzike u većim razmerama. U ovoj koncepciji sistem za P2P razmenu datoteka koristi veliki server (ili grupu servera) koji je zadužen za uslugu direktorijuma. Kao što možete da vidite na slici 2.23, kada korisnik pokrene svoju P2P aplikaciju, ona odmah uspostavlja kontakt sa serverom direktorijuma. Konkretno, ona ga obaveštava o svojoj



Slika 2.23 ♦ P2P paradigm sa centralizovanim direktorijumom

IP adresi i nazivima objekata na lokalnom disku koji su spremni za razmenu (na primer, o nazivima svih MP3 datoteka). Na ovaj način server direktorijuma može da zna koje je objekte dati korisnik spreman da razmenjuje. Sakupljujući ove informacije od svih aktivnih korisnika, server direktorijuma pravi centralizovanu dinamičku bazu podataka koja svaki objekat preslikava na skup IP adresa. Ukoliko neki korisnik obezbedi novu datoteku ili ukloni neku staru, on o tome obaveštava server direktorijuma koji odmah ažurira svoju bazu podataka.

Da bi mu baza podataka bila uvek ažurna, server direktorijuma treba da zna kako da ustanovi kada se neki korisnik isključio. Korisnik se isključuje ili zatvaranjem P2P aplikacije ili jednostavnim prekidanjem veze sa Internetom.

Periodičnim slanjem poruka na koje korisnici treba da odgovore server može da ustanovi koji su korisnici trenutno povezani. Kada ustanovi da određeni klijent više nije u mreži, server direktorijuma uklanja njegovu IP adresu iz svoje baze podataka.

U P2P razmeni datoteka sa centralizovanim direktorijumom upotrebljena je hibridna klijentsko-serverska P2P arhitektura koju smo pominjali u odeljku 2.1.1. Korišćenje centralizovanog direktorijuma za lociranje sadržaja predstavlja jednostavan koncept koji ima i brojne nedostatke.

- ♦ *Oslanjanje na jednu tačku.* Kvar servera direktorijuma onesposobljuje čitavu aplikaciju. Čak i ako se koristi grupa redundantnih servera, uvek postoji mogućnost kvara na vezi tih servera sa Internetom, što takođe povlači za sobom pad čitave aplikacije,
- ♦ *Postojanje uskog grla u pogledu performansi.* U velikim P2P sistemima sa stotinama hiljada povezanih korisnika centralizovani server mora da održava ogro-

mnu bazu podataka i odgovara na hiljade upita u sekundi. Godine 2000. kada je bio bez premcia najpopularnija P2P aplikacija, Napster je imao mnogo problema sa saobraćajem na svom centralizovanom serveru.

- ♦ *Kršenje autorskih prava.* Iako ova tema prevazilazi tematiku ove knjige, kratko ćemo pomenuti i to daje diskografska industrija, blago rečeno, zabrinuta zbog toga što sistemi za P2P razmenu datoteka omogućavaju korisnicima da besplatno dođu do sadržaja koji su zaštićeni autorskim pravima. Ukoliko P2P kompanija ima centralizovani server direktorijuma, pravni postupak protiv nje može da rezultira zabranom rada i isključivanjem ovog servera, Decentralizovanu arhitekturu je, međutim, mnogo teže isključiti. ■

Prema tome, osnovnu manu korišćenja centralizovanog servera direktorijuma predstavlja to stoje P2P aplikacija u tom slučaju samo delimično decentralizovana. Sam transfer datoteka između ravnopravnih korisnika jeste decentralizovan, ali je proces lociranja sadržaja visoko centralizovan. Što predstavlja veliki problem kada su u pitanju pouzdanost i performanse.

Poplava upita

>

Gnutella je aplikacija za P2P razmenu datoteka iz javnog domena u kojoj se sadržaj locira kroz potpuno distribuirani pristup. Za razliku od Napstera, ovoj aplikaciji za traženje sadržaja za razmenu nije potreban centralizovani server. Štaviše, u Gnutelli je problem lociranja sadržaja rešen na dijametralno suprotan način.

Gnutella klijent (koji ima i korisnički interfejs) implementira protokol Gnutella i izvršava se kao običan ravnopravni korisnik. Specifikacija ovog protokola je minimalna, što ostavlja značajnu slobodu programerima Gnutella klijenta. Kao što postoji veliki broj Čitača Weba koji podržavaju protokol HTTP, isto tako postoji i dosta Gnutella klijenata koji implementiraju istoimeni protokol.

U Gnutelli ravnopravni korisnici formiraju apstraktну, logičku mrežu za koju se koristi termin preklopjena mreža, koja se u grafičko-teorijskom pogledu definiše na sledeći način: Ukoliko ravnopravni korisnik X održava TCP konekciju sa ravnopravnim korisnikom Y, onda kažemo da između njih postoji grana (engl. edge). Grafikon koji čine svi aktivni ravnopravni korisnici i njihove međusobne TCP konekcije definiše aktuelnu Gnutella mrežu. Skrećemo vam pažnju na to da grana ne predstavlja fizičke komunikacione linkove. Primera radi, moguće je da se ona odnosi na TCP konekciju između ravnopravnih korisnika u Litvaniji i Brazilu.

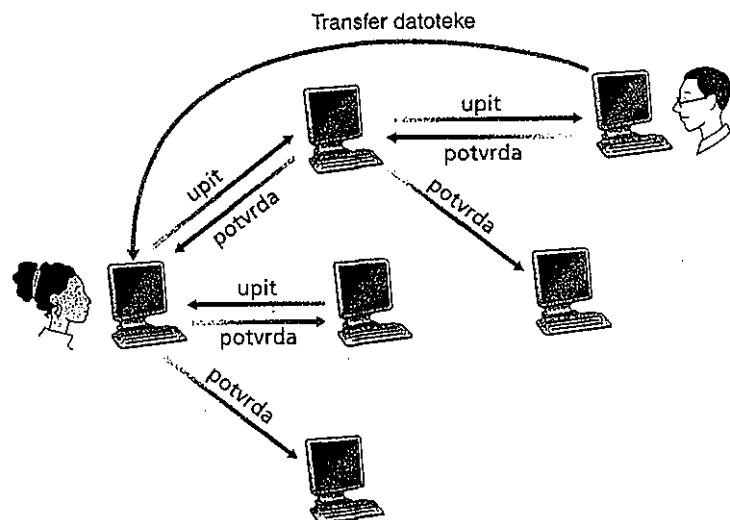
Iako mreža Gnutella može imati stotine hiljada ravnopravnih korisnika, svaki od njih je obično povezan sa manje od deset drugih čvorova preklopljene mreže (slika 2.24). Nešto kasnije videćete na koji način se mreža Gnutella formira i održava u kontekstu ulaska ravnopravnih članova u mrežu i izlaska iz nje. Za sada, pretpostavimo da je preklopljena mreža već formirana i pogledati na koji način ravnopravni korisnici lociraju i pribavljaju sadržaj.

U Gnutelli ravnopravni korisnici šalju poruke svojim susedima u preklopljenoj mreži putem unapred postojećih TCP konekcija. Kada Alisa želi da locira pesmu „Netvork Love”, njen Gnutella klijent svim svojim susedima šalje poruku sa upitom (Query) u kojoj su navedene ključne reči „Netvork Love”. Svi Alisini susedi zatim

prosleđuju ovaj upit svojim susedima i tako redom. Ovaj proces, koji smo ilustrovali na slici 2.24, naziva se **poplava upita**. Kada primi upit, svaki ravnopravni korisnik proverava da li se ključne reči poklapaju sa datotekama koje je ponudio za deobu. Ukoliko se ustanovi poklapanje, korisnik odgovara Alisi potvrdom porukom (QueryHit) zajedno sa imenom datoteke i njenom veličinom. Ova poruka kreće se istom onom putanjom kojom je pristigao i upit, pa se zato ove veze i nazivaju unapred postojećim TCP konekcijama.

Alisin Gnutella klijent može da primi potvrde poruke i od više korisnika. U tom slučaju Alisa bira nekog od njih - u ovom slučaju to će biti Bob. Njen klijent zatim uspostavlja direktnu TCP konekciju sa Bobovim, i šalje mu HTTP GET poruku sa nazivom konkretnе datoteke. Nakon toga, Bobov klijent, u svom HTTP odgovoru, šalje Alisi datoteku koju je zatražila. Iako upiti i potvrde poruke (Query i QueryHit) putuju kroz preklopljenu mrežu, poruka HTTP GET i odgovor ne idu kroz ovu mrežu, već direktno od korisnika do korisnika (slika 2.24). Čim Alisa prima čitavu datoteku, TCP konekcija između nje i Boba se prekida.

Decentralizovani dizajn aplikacije Gnutella je elegantan i atraktivан ali se kri-tikuje, pre svega, zbog toga što ova mreža nije proširiva. Konkretno, kod poplave upita, upit jednog ravnopravnog korisnika se šalje svakom drugom članu mreže. To znači da se u mrežu ispušta ogromna količina saobraćaja i da se svaki član prosti zasipa upitima. Dizajneri mreže Gnutella pokušali su da ovaj problem reše **ograničavanjem dometa poplave upita**. Drugim recima, kada Alisa pošalje svoj inicijalni upit, u polju za brojanje čvorova ove poruke upisuje se odredena vrednost



Slika 2.24 ♦ Traženje i transfer datoteka u Gnutelli

(recimo, 7). Svaki put kada poruka stigne do nekog novog ravnopravnog korisnika, on smanjuje vrednost ovog polja za jedan, a zatim prosleđuje poruku dalje, svojim susedima. Kada neki ravnopravni korisnik primi poruku u kojoj je vrednost polja za brojanje čvorova nula, on je više ne prosleđuje. Na ovaj način poplava upita loka-izlazi se na određeni region preklopljene mreže, čime se zaista smanjuje obim saobraćaja, ali po cenu nepronalaženja svih ravnopravnih korisnika koji imaju željeni sadržaj. U Gnutelli, dakle, uvek postoji mogućnost da neki sadržaj ne možete da locirate samo zato što se nalazi izvan dometa vaše poplave upita.

Najvažniji aspekt distribuiranih P2P aplikacija jeste način na koji one rukuju ulascima novih korisnika u mrežu, kao i izlascima iz nje. Sada ćemo vam opisati što se događa kada ravnopravni korisnik X želi da uđe u Gnutella mrežu. Akcije koje Gnutella preduzima prilikom izlaska ravnopravnog korisnika iz mreže opisane su u domaćim zadacima.

1. Ravnopravni korisnik X najpre mora da pronade nekog drugog ravnopravnog korisnika koji je veću mreži. Klijent X ovaj **problema podizanja** može da prevaziđe tako što će da održava listu ravnopravnih korisnika (njihove IP adrese) koji su često u Gnutella mreži; isto tako, korisnik X može da poseti i Gnutellinu web lokaciju na kojoj postoji takva lista.
2. Kada dođe do ove liste korisnik X redom pokušava da uspostavi TCP konekciju sa korisnicima sa liste sve dok sa nekim korisnikom Y ne uspe u tome.
3. Kada se između ravnopravnih korisnika X i Y uspostavi TCP konekcija, ravnopravni korisnik X šalje Gnutella Ping poruku korisniku Y. U okviru ove poruke nalazi se i polje za brojanje ravnopravnih korisnika. Kada primi ovu Ping poruku korisnik Y je dalje prosleđuje svim svojim susedima u preklopljenoj mreži. Ostali ravnopravni korisnici nastavljaju da prosleđuju ovu poruku na isti način sve dok vrednost polja za brojanje ravnopravnih članova ne postane nula.
4. Čim korisnik Z primi Gnutella Ping poruku, on na nju odgovara Gnutella Pong porukom koju kroz preklopljenu mrežu vraća korisniku X. U okviru Pong poruke nalaze se IP adresa ravnopravnog korisnika Z, zatim broj datoteka koje su spremne za razmenu, kao i njihova ukupna veličina.
5. Nakon prijema Pong poruka, uz korisnika Y sada i korisnik X zna IP adrese mnogih drugih ravnopravnih korisnika Gnutella mreže. Sa nekim od njih on može da uspostavi TCP konekcije i tako od sebe formira nekoliko grana. Osim toga, ravnopravni korisnik X može da pokuša da uspostavi TCP konekcije i sa mnogim drugim čvorovima za podizanje. U specifikaciji mreže Gnutella, ne precizira se sa koliko ravnopravnih korisnika korisnik X može da se poveže, tako daje tu ostavljena izvesna sloboda programerima Gnutella klijenata.

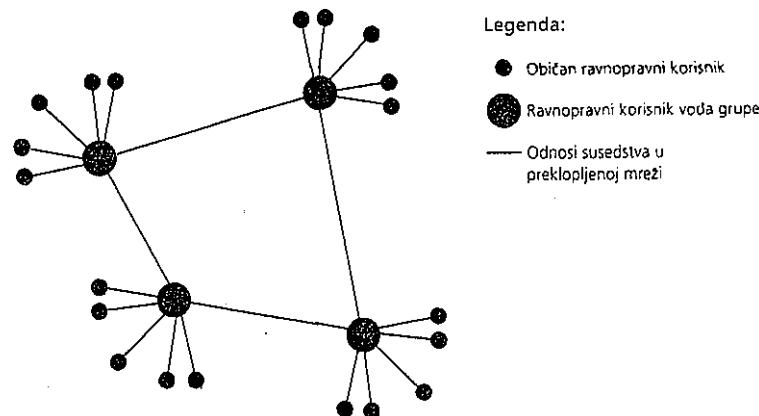
U ovom delu govorili smo o svim najvažnijim karakteristikama protokola Gnutella, a u domaćim zadacima ćemo im posvetiti još malo pažnje. Dakle, Gnutella predstavlja jednostavan distribuiran P2P sistem koji omogućava traženje datoteka koje se nalaze kod susednih korisnika (susedni korisnici su oni koji su u preklopljenoj mreži udaljeni nekoliko skokova).

Korišćenje heterogenosti

Do sada ste videli da aplikacije Napster i Gnutella prilikom lociranja sadržaja koriste dva dijametralno suprotna pristupa. Napster koristi centralizovani server direktorijuma i može da locira sav sadržaj bilo kog ravnopravnog korisnika. Gnutella koristi potpuno distribuiranu arhitekturu, ali locira samo onaj sadržaj koji se nalazi kod susednih ravnopravnih korisnika u preklopljenoj mreži. U aplikaciji po imenu KaZaA [KaZaA 2004] pozajmljene su ideje iz oba prethodno pomenuta pristupa, a kao krajnji rezultat dobijen je izuzetno moćan sistem za P2P razmenu datoteka koji je u vreme pisanja ove knjige stvarao više saobraćaja na Internetu od bilo koje druge aplikacije [Gummadi 2003] [Saroiu 2003].

Tehnologija aplikacije KaZaA je vlasnička. Uz to, ova aplikacija sav svoj saobraćaj (osim samih datoteka koje se razmenjuju) šalje u šifrovanim obliku. Međutim, iz osnovnih opisa koji mogu da se pronađu na web lokaciji KaZaA, kao i iz nekih njihovih merenja [Liang 2004] moguće je zaključiti na koji način ovaj protokol funkcioniše. Prvo što bi moglo da se kaže o ovoj aplikaciji jeste to da KaZaA koristi heterogenost na veoma intuitivan i prirodan način.

Poput Gnutelle, KaZaA prilikom traženja i lociranja sadržaja za razmenu ne koristi namenski server (ili grupu servera). Međutim, ovde, za razliku od Gnutelle, nisu svi ravnopravni korisnici isti. Moćniji korisnici - oni sa linkovima veće propusne moći i većom mogućnošću povezivanja - imaju uloge „voda grupe“ i uz to veće odgovornosti. Kada neki ravnopravni korisnik nije voda grupe, on postaje običan korisnik i dodeljuje se nekom vodi grupe (slika 2.25). Voda grupe najčešće može imati i nekoliko stotina podređenih ravnopravnih korisnika.



Slika 2.25 • Hijerarhijska preklopljena mreža za P2P razmenu datoteka

Nakon pokretanja aplikacije KaZaA, ravnopravni korisnik uspostavlja TCP konekciju sa nekim vodom grupe i informiše ga o tome koje datoteke nudi za deobu. Ove informacije služe vodi grupe za održavanje baze podataka u kojoj se nalaze identifikatori svih članova njegove grupe, zatim metapodaci o samim datotekama, kao i odgovarajuće IP adrese članova grupe kod kojih se te datoteke nalaze. Na ovaj način svaki voda grupe postaje mini (Napster) hab. Ali, za razliku od Napstera, voda grupe nije namenski server, već i dalje običan korisnik.

Kada bi svaki od ovih mini-habova, zajedno sa svojim članovima, predstavljao izolovan entitet, mreža KaZaA bi se, u osnovi, sastojala od hiljada izolovanih mini-Napster mreža u kojima ulogu haba ima ravnopravni korisnik a ne namenski server. Ovakav izolovani pristup bi, sa svoje strane, značajno ograničio obim sadržaja koji može da stoji na raspolažanju svakom ravnopravnom korisniku. Ovo ograničenje otklonjeno je međusobnim TCP konekcijama između voda grupe i stvaranjem pre-klopljene mreže između njih samih. To znači da vode grupe formiraju mrežu koja podseća na mrežu aplikacije Gnutella. U mreži KaZaA informacije o dostupnom sadržaju razmenjuju se između grupa. Ova međusobna razmena informacija može da se postigne na nekoliko načina. Primera radi, hab jedne grupe može da se obrati habovima drugih grupa i u njih dobije kopije njihovih baza podataka, koje zatim može da objedini sa sopstvenim informacijama, čime se postiže praćenje sadržaja daleko većeg broja korisnika. Isto tako, svaki hab bi mogao da prati samo sadržaj članova sopstvene grupe, ali da, kada primi neki upit, taj upit prosledi svim ostalim habovima sa kojima je povezan. (Ovaj pristup podseća na poplavu upita protokola Gnutella, sa tom razlikom što u preklopljenoj mreži voda grupe poplava ima sasvim ograničene razmere.)

U mreži KaZaA svaka datoteka identificuje se *heš* identifikatorima na koje ćemo se vratiti za koji trenutak. Osim toga, ovde svaki objekat ima i svoj deskriptor koji obuhvata ime datoteke, kao i njegov nestrukturisani tekstualni opis. Ravnopravni korisnici u upitima koriste ključne reči, što podseća na način korišćenja pretraživača. Prilikom traženja ključnih reči, dati ravnopravni korisnik šalje svom vodi grupe upit sa ključnim recima. Na ovaj upit voda grupe odgovara listom na kojoj su navedeni ravnopravni korisnici (unutar svoje grupe ili izvan nje) koji imaju datoteke čiji se deskriptori poklapaju sa ključnim recima, kao i identifikatori tih datoteka. Voda grupe zatim može da prosledi dobijeni upit u vodama drugih grupa sa kojima je povezan. Njihovi odgovori vraćaju se obrnutom putanjom kroz preklopljenu mrežu.

Dakle, u arhitekturi mreže KaZaA postoji *heterogenost ravnopravnih korisnika* u tom smislu što mali broj moćnijih korisnika dobija ulogu voda grupe koji, dalje, formiraju najviši sloj hijerarhije preklopljene mreže (slika 2.25). U poređenju sa dizajnom mreže Gnutella koji se odlikuje ravnim preklapanjem i ograničenim opsegom poplave upita, hijerarhijski dizajn omogućava prveru značajno većeg broja korisnika, bez preobimnog saobraćaja. (Ubedljivo najveći deo današnjeg saobraćaja mreže aplikacije KaZaA odlazi na transfer samih datoteka.)

U aplikaciji KaZaA koristi se čitav niz tehnika koje poboljšavaju njene performanse. U ove tehnike koje mogu da se pominje u skoro svim sistemima za P2P razmenu datoteka spadaju:

- 1. Stavljanje zahteva u red.** Svaki korisnik može u svom programu da ograniči broj istovremenih preuzimanja sa svog računara. (Podrazumevane vrednosti kreću se od **tri** do sedam.) Ukoliko Alisa u svom programu ograniči broj preuzimanja sa njenog računara na **tri** i ukoliko su sva tri preuzimanja već u toku, Bobov zahtev za preuzimanje neke datoteke, kao i svi sledeći zahtevi smeštaju se u lokalni red. Ovakvim lokalnim ograničavanjem obezbeđuje se da svaka datoteka koja se preuzima od čvora koji je nekome predaje dobije adekvatan deo propusne moći.
- 2. Podsticanje prioriteta.** Prilikom stavljanja drugih ravnopravnih korisnika u red, Alisa će dati prednost onima koji su do tada predali više datoteka nego što su preuzeли. Primera radi, ako su u njenom redu Bob **i** Kler, a Bob je preuzeo više nego stoje predao, dok je Kler predala više nego stoje preuzeala, Alisa će najpre izaći u susret zahtevu koji joj je stigao od Kler. Ovim mehanizmom korisnici se podstiču da predaju više datoteka čime se povećava sveukupna proširivost sistema KaZaA.
- 3. Paralelno preuzimanje.** Kada Kler potraži pesmu „Network Love“ KaZaA joj najčešće vraća listu sa nekoliko korisnika kod kojih je ustanovljeno poklanjanje ključne reči. Neka od ovih poklapanja odnosiće se na istu datoteku što ■ se proverava identifikatorima datoteka. Njen klijent može korišćenjem HTTP zaglavla da zahteva različite delove datoteke od različitih korisnika, odnosno da paralelno preuzima delove datoteke od različitih korisnika. Ukoliko, primera radi, Alisa **i** Bob imaju datoteku koju Kler traži, njen KaZaA klijent može od Alise da zatraži prvu polovicu datoteke, a od Boba drugu. Ova mogućnost značajno ubrzava transfer datoteka u najrazličitijim okolnostima,

Ukratko ćemo pomenuti i činjenicu da se za aplikaciju KaZaA vezuju i mnogi pokušaji reverznog inženjeringu; štaviše, u nekim projektima razvijene su modifiko-vane verzije KaZaA klijenata koje omogućavaju korisnicima pristup mreži KaZaA bez apsolutno doslednog praćenja ovog protokola. Trenutno je veoma popularan klijent po imenu *kazaa-lite* koji, pored toga što eliminiše ugrađeno oglašavanje i daje svim korisnicima najviši prioritet, izvršava i skakanje od jednog do drugog super-čvora, odnosno traži ključne reči redom od jednog do drugog vođe grupe.

Poput svoje prethodnice Napstera, KaZaA se razvio u izuzetno popularnu Internet aplikaciju koja je za samo nekoliko meseci privukla milione korisnika. Nevezatno brzo širenje aplikacija za P2P razmenu datoteka, kao i Weba i trenutne razmene poruka pre njih, svedoči o kvalitetu dizajna sveukupne arhitekture Interneta prilikom čijeg dizajniranja nije bilo moguće zamisliti kakve će se aplikacije razviti u narednih 25 godina. Mrežne usluge koje se stavljaju na raspolaganje internetskim aplikacijama - transport datagrama bez konekcije, pouzdani transport datagrama sa konekcijom, interfejs soketa, zatim usluge adresiranja i imenovanja (DNS) - pokazale su se dovoljno uspešnim za razvoj hiljada novih aplikacija. S obzirom na to da su sve ove aplikacije smeštene povrh postojeća četiri sloja Internet familije protokola, za njih je bilo potrebno samo razviti nov klijentsko-serverski softver za krajnje sisteme. Upravo zahvaljujući tome ove aplikacije se brzo prave i puštaju u rad,

SiiS- KRATAK OSVRT

KaZaA

U vreme pisanja ove knjige (aprila 2004. godine) u mreži KaZaA svakodnevno je učestvovalo vrše od 3 miliona korisnika koji su između sebe razmenjivali 5.000 rorabjato raznog sadržaja. S obzirom na to da najveći deo ovog sadržaja predstavljaju zaštićene pesme i filmovi, KaZaA je veliki neprijatelj i muzičke i filmske industrije. Osim toga, saobraćaj iz ove mreže izuzetno opterećuje različite posrednike za Internet usluge. Primera radi, u mreži Univerziteta u Washingtonu junca 2002. godine oko 37 procenata celokupnog TCP saobraćaja pravila je upravo ova aplikacija - što je bilo dva puta više od saobraćaja koji je stvarao VVeb [Sarou 2002]. Verovatno se sada pitate gde se ova fantastična aplikacija pojavila i koji su njeni krajnji dometni.

Privatna, holandska kompanija po imenu FastTrack je aprila 2000. godine objavila prvu verziju programa KaZaA. Već ova prva verzija imala je hijerarhijsku arhitekturu sa superčvorovima koja je i danas zaštitni znak ove aplikacije. U nekoliko narednih godina kompanije kao što su, MusicCity (Morpheus), Grokster i iMesh su licencirale softver kompanije FastTrack. Nakon tužbe Diskografske asocijacije SAD /Recording Industry Association of America, RIAA) i holandskih sudova, februara 2002. godine najveći deo kompanije FastTrack prešao je u ruke kompanije Sharman Networks. Struktura ove novonastale kompanije je veoma zamršena i složena. Kompanija Sharpman Networks zvanično je prijavljena na malom pacifičkom ostrvu Vonuatu, sedište joj je u Australiji, a veliki broj njenih programera živi i radi u Estoniju. Ovako složena i zamršeno struktura značajno otežava posao američkim kompanijama i udruženjima koji bi želeli da je tuže.

Veoma je zanimljivo i pitanje budućeg rasta mreže KaZaA, čok i u slučaju da se kompanije koje stope iza nje zatvore (iz pravnih ili finansijskih razloga). Kao što ste već videli, mreža KaZaA odlikuje se veoma decentralizovanom arhitekturom koja se ne oslanja na servere. Prema tome, vrlo je verovatno da bi i bez kompanije koje stope iza nje ova mreža nastavila da postoji u nekom obliku. Diskografska industrija na sve moguće načine pokušava da ograniči ovaku razmenu datoteka preuzimajući mere, kao što su tužbe protiv individualnih korisnika aplikacije KaZaA ili puštanje lažnih datoteka u mrežu [Overpeer 2004].

Konačno, mi [autori ove knjige] predviđamo pojavu bežičnih P2P sistema zo razmenu datoteka u kojima će pojedinci - recimo, studenti - imati PDA računare sa gigabajtima prostora i Wi-Fi (bežične) mogućnosti komunikacije. Ovi Wi-Fi često sakriveni u školskim torbama funkcionišu u takozvanom *ad hoc* režimu što će im omogućiti diskretnu razmenu datoteka bežičnim putem. Jgnsno je da će kompanijama i udruženjima kao što je, na primer, RIAA biti veoma teško da prate i eventualno ometu ovakve P2P sisteme.

Ipak, nisu sve nove Internet aplikacije imale toliko uspeha kao VVeb, trenutna razmena poruka i P2P razmena datoteka, Multimedijalne konferencije i protok multimedijalnih sadržaja u realnom vremenu su, recimo, dve aplikacije koje još uvek nisu doživele toliki uspeh. Vreme će pokazati da lije kod njih ograničavajući faktor predstavlja nedovoljan skup usluga aktuelne arhitekture Interneta (o predlozima za njeno proširivanje u smislu podrške ovih aplikacija govorićemo u poglavju 7) ili su to bili društveni i ekonomski faktori.

2.7 Programiranje soketa za protokol TCP

Sada kada smo vam pokazali nekoliko važnih mrežnih aplikacija, pogledajmo na koji način se ovakvi programi pišu. U ovom odeljku napisaćemo aplikacije koje koriste protokol TCP, dok će u sledećem odeljku biti aplikacije za protokol UDP.

Ako se sećate, u odeljku 2.1 rekli smo da jezgro mrežne aplikacije čine dva programa - klijentski i serverski. Izvršavanje ova dva programa dovodi do stvaranja klijentskog i serverskog procesa koji između sebe komuniciraju čitanjem iz soketa i pisanjem u njih. Prilikom programiranja mrežne aplikacije, osnovni zadatak programera je **da** napiše kod za klijentski i serverski program.

Postoje dve vrste klijentsko-serverskih aplikacija. Prvu vrstu Čine klijentsko-serverske aplikacije koje predstavljaju implementaciju standarda protokola koji je definisan RFC dokumentom. Kod ovakvih implementacija klijentski i serverski program moraju da budu u skladu sa pravilima koja diktiraju RFC dokumenti. Primera radi, klijentski program bi mogao da predstavlja implementaciju FTP klijenta (ode-Ijak 2.3) koji je definisan u dokumentu RFC 959, dok bi serverski program mogao **da** bude implementacija FTP servera koji je takođe definisan u dokumentu RFC 959. Ukoliko jedan programer napiše kod za klijentski program, a neki drugi, potpuno nezavisno, za serverski i pri tom se obojica dosledno pridržavaju pravila koja diktiraju RFC dokumenti, dva programa će biti interoperabilna. U stvari, u velikom broju savremene mrežne aplikacije podrazumevaju komunikaciju između klijentskih i serverskih programa koje su potpuno nezavisno napravili različiti programeri (situacije ove vrste su kada, na primer, Netscape Čitač komunicira sa Apache web serverom, ili kada FTP klijent na nekom personalnom računaru predaje neku datoteku UNIX FTP serveru). Kada klijentski ili serverski program implementira protokol koji je definisan u RFC dokumentu, on treba da koristi onaj broj porta koji je povezan sa datim protokolom. (O brojevima portova smo ukratko govorili u odeljku 2.1. U poglavlju 3 pružićemo vam više informacija o njima.)

Drugu vrstu klijentsko-serverskih aplikacija čine specijalne (vlasničke) aplikacije. U ovom slučaju ni klijentski ni serverski programi ne moraju biti u skladu sa bilo kojim RFC dokumentom. Njih pravi jedan programer (ili jedan tim) i on ima apsolutnu kontrolu nad svime. Što se dešava unutar koda. Ali, budući da ovakvi programi ne implementiraju protokol koji se nalazi u javnom domenu, drugi nezavisni programeri ne mogu da razviju kod koji bi bio interoperabilan sa njima. Prilikom programiranja vlasničkih aplikacija programeri treba da izbegnu korišćenje brojeva dobro poznatih portova koji su definisani u RFC dokumentima.

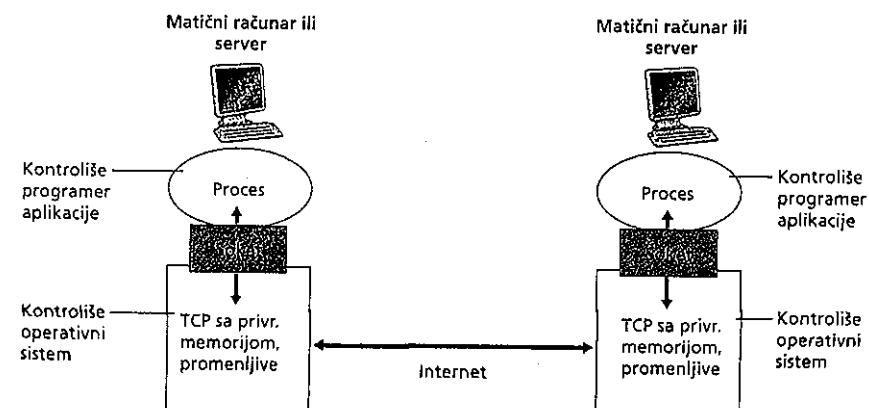
U ovom i u sledećem odeljku bavićemo se ključnim aspektima programiranja vlasničkih klijentsko-serverskih aplikacija. U fazi programiranja jedna od prvih programerovih odluka odnosi se na to da li će njegova aplikacija koristiti protokol TCP ili protokol UDP. Podsećamo vas **da** su TCP veze sa konekcijom i obezbeđuju pouzdane kanale za protok podataka između dva krajnja sistema. Kod UDP veza, nasuprot tome, nema konekcije i njima se nezavisni paketi podataka šalju od jednog do drugog krajnjeg sistema bez ikakvih garancija u pogledu isporuke.

U ovom odeljku razvićemo jednostavnu klijentsku aplikaciju koja koristi TCP, a u sledećem isto tako jednostavnu klijentsku aplikaciju koja koristi UDP. Ove jednostavne TCP i UDP aplikacije predstavljene su u programskom jeziku Java. Programe smo mogli da napišemo i u programskim jezicima C ili C++, ali smo se opredelili za Javu iz više razloga. Najpre, aplikacije napisane u Javi imaju jednostavniji i čistiji kod, sa manje redova od kojih svaki, bez ikakvih teškoća, može da se objasni i programeru početniku. Zatim, programiranje klijentsko-serverskih aplikacija u Javi je sve popularnije i može se očekivati da u narednim godinama postane standard. Ukoliko vam ovaj programski jezik nije poznat, ne očajavajte. Programersko iskustvo u bilo kom drugom programskom jeziku pomoći će vam da ovaj kod pratite bez ikakvih problema.

Za čitaoce koje posebno zanima programiranje klijentsko-serverskih aplikacija u programskom jeziku C postoji nekoliko dobrih referenci [Donahoo 2000; Stevens 1997; Frost 1994; Kurose 1996].

2.7.1 Programiranje soketa za protokol TCP

Ukoliko se sećate, u odeljku 2.1 rekli smo da procesi koji se izvršavaju na različitim računarima međusobno komuniciraju slanjem poruka u sokete. Rekli smo i to da su soketi za procese ono što su vrata za kuće. Kao što možete da vidite na slici 2.26, soket predstavlja vrata između procesa aplikacije i protokola TCP. Programer aplikacije ima kontrolu nad svime što se dešava sa strane soketa ka aplikaciji, dok na ono što se dešava sa njihove druge strane, ka transportnom sloju, nema skoro nimalo uticaja. (U najboljem slučaju programer aplikacije može da podesi nekoliko TCP parametara, kao što su, na primer, maksimalna veličina privremene memorije ili maksimalna veličina segmenata.)



Slika 2.26 ♦ Komunikacija procesa putem soketa

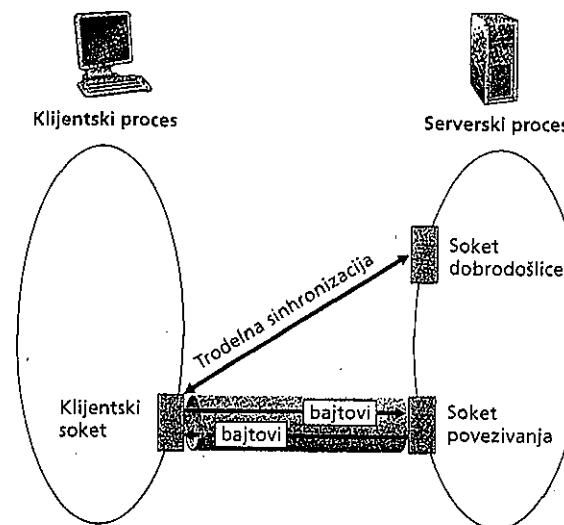
Pogledajmo sada izbliza kako izgleda interakcija klijentskog i serverskog programa. Klijent je taj koji treba da inicira komunikaciju sa serverom. Da bi mogao da odgovori na klijentov inicijalni kontakt, server mora biti spremjan. Sto podrazumeva dve stvari. Najpre, serverski program ne može da bude neaktivan; on mora da bude aktivan kao proces pre nego što klijent uopšte pokuša da uspostavi inicijalni kontakt. Zatim, serverski program mora imati neku vrstu vrata (odnosno, soket) koji će da reaguje na klijentov inicijalni kontakt. Vraćajući se za trenutak analogiji sa kućom i vratima, ovaj klijentov inicijalni kontakt često se naziva „kucanjem na vrata“.

Dakle, ako je serverski proces aktivan, klijentski proces može da inicira TCP konekciju sa serverom. Ovo se postiže tako što klijentski program pravi soket. Kada napravi svoj soket, klijent navodi adresu serverskog procesa, odnosno, IP adresu servera i broj porta procesa. Odmah nakon pravljenja soketa, protokol TCP klijentskog procesa inicira trodelnu sinhronizaciju i uspostavlja TCP konekciju sa serverom. Ova trodelna sinhronizacija je potpuno transparentna i za klijentski i za serverski program.

Tokom trodelne sinhronizacije, klijentski proces „kuca na vrata“ serverskog procesa. Kada „čuje“ ovo kucanje, serverski proces pravi nova vrata (odnosno, soket) koja su namenjena samo tom klijentu. U primerima koji slede, vrata na koja klijent kucaće objekat ServerSocket koji ćemo nazvati imenom welcomeSocket. Kada klijent zakuca na ova vrata, program priziva metod accept () objekta welcomeSocket koji za datog klijenta pravi nova vrata. Na kraju faze sinhronizacije između klijentovog soketa i novog soketa na serveru postoji TCP konekcija, tako da ćemo ovaj novi soket nazvati serverovim soketom povezivanja,

Iz perspektive aplikacije TCP konekcija je direktna virtuelna linija između klijentovog soketa i soketa povezivanja na serveru. Odmah nakon toga klijentski proces može da počne sa slanjem podataka u soket; protokol TCP garanjuje da će serverski proces primiti (kroz soket povezivanja) svaki poslati bajt. Štaviše, kao što ljudi kroz ista vrata mogu i da ulaze i da izlaze, klijentski proces putem svog soketa može da prima podatke, a serverski proces kroz soket povezivanja može da ih šalje. Ovaj koncept ilustrovan je na slici 2.27. S obzirom na to da soketi imaju ključnu ulogu u klijentsko-serverskim aplikacijama, za programiranje ovakvih aplikacija često se koristi termin programiranje soketa.

Pre nego što predemo na naš primer klijentsko-serverske aplikacije, ne bi bilo loše da razjasnimo i pojmom tok. Tok predstavlja seriju znakova koji ulaze u neki proces ili iz njega izlaze. Svaki tok je ili ulazni tok procesa ili njegov izlazni tok. Ukoliko je reč o ulaznom toku, on je uvek povezan sa nekim izvorom ulaza za dati proces, kao što je, na primer, standardni ulaz (putem tastature) ili soket u koji podaci ulaze sa Interneta. Ako je, s druge strane, reč o izlaznom toku, on je uvek povezan sa nekim izvorom izlaza za dati proces, što može biti standardni izlaz (monitor) ili soket iz koga podaci idu na Internet.



Slika 2.27 ♦ Klijentski soket, soket dobrodošlice i soket povezivanja

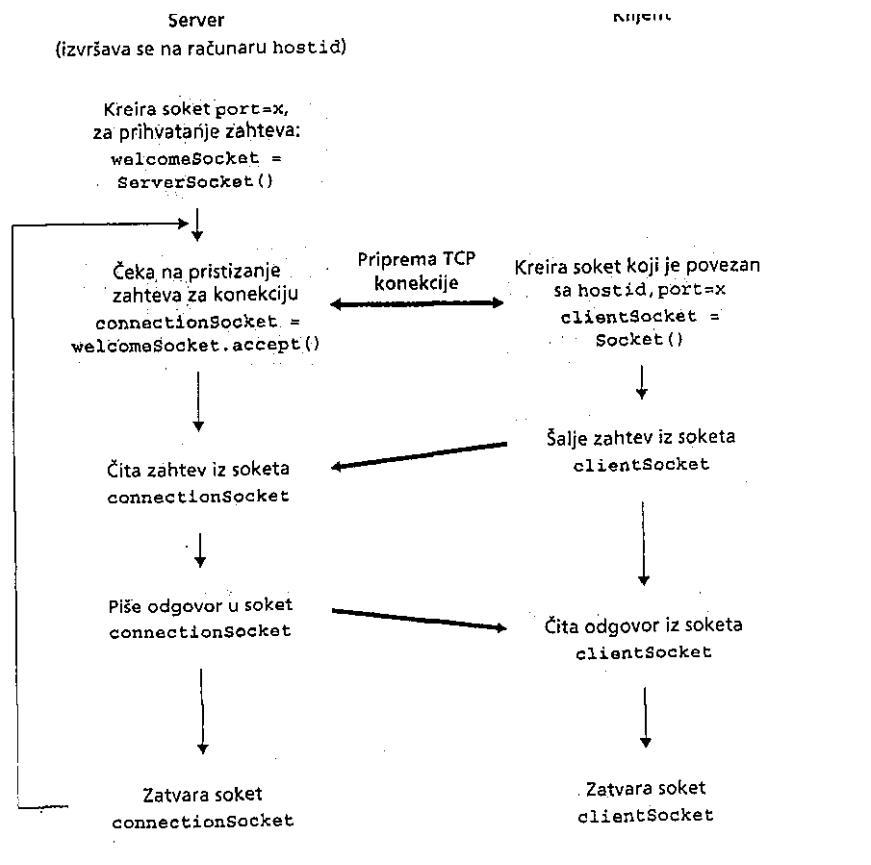
2.7.2 Primer klijentsko-serverske aplikacije u Javi

Za ilustraciju programiranja soketa za protokole TCP i UDP koristićemo sledeću jednostavnu klijentsko-serversku aplikaciju:

1. Klijent čita red teksta iz standardnog ulaza (tastature) i iz svog soketa šalje ovaj red serveru.
2. Server čita red iz svog konečnijskog soketa.
3. Server pretvara ovaj red teksta u velika slova.
4. Server iz svog konečnijskog soketa šalje izmenjeni red teksta klijentu.
5. Klijent čita primljeni red teksta iz svog soketa i izbacuje ga na svom standardnom izlazu (monitor).

Na slici 2.28 ilustrovane su glavne aktivnosti klijenta i servera u vezi sa soketima.

Zatim ćemo vam prikazati klijentsko-serverski par programa za TCP implementaciju aplikacije. Red po red, detaljno ćemo analizirati oba ova programa. Klijentski program nazvali smo TCPClient, a serverski TCPServ^r. Jave. Da bismo naglasili ključne aspekte, namerno smo ponudili kćed koji je tačan, ali ne i apsolutno stabilan. „Pravi kod“ bi svakako imao još nekoliko pomoćnih redova.



Slika 2.28 ♦ Klijentsko-serverska aplikacija koja koristi transportnu uslugu sa konekcijom.

Nakon kompajliranja ovih programa na njihovim matičnim računalima, serverski program se prvi izvršava čime se na serveru stvara proces. Kao što smo već rekli, serverski proces čeka na inicijalni kontakt klijentskog procesa. U ovom slučaju, nakon pokretanja klijentskog programa stvara se klijentski proces koji kontaktira sa serverom i sa njim uspostavlja TCP konekciju. Odmah zatim, korisnik klijentskog računara može da upotrebi aplikaciju i pošalje red teksta i potom primi isti red ispisani velikim slovima.

TCPClient.java

Evo kako izgleda kod klijentske strane aplikacije:

```

import java.io.*;
import java.net.*;
class TCPClient {
    public static void main(String argv[]) throws Exception {
        String sentence;
        String modifiedSentence;
        BufferedReader inFromUser = new BufferedReader(new
            InputStreamReader(System.in));
        Socket clientSocket = new Socket("hostname", 6789);
        DataOutputStream outToServer = new DataOutputStream(
            clientSocket.getOutputStream());
        BufferedReader inFromServer =
            new BufferedReader(new InputStreamReader(
                clientSocket.getInputStream()));
        sentence = inFromUser.readLine();
        outToServer.writeBytes(sentence + "\n");
        modifiedSentence = inFromServer.readLine();
        System.out.println("FROM SERVER: " + modifiedSentence);
        clientSocket.close();
    }
}
  
```

Kao što možete da vidite na slici 2.29, program TCPClient pravi tri toka i jedan soket. Ime soketa je clientSocket. Tok inFromUser predstavlja ulazni tok programa; on je povezan sa standardnim ulazom (tastaturom). Znaci koje korisnik upiše pomoću tastature ulaze u tok inFromUser. Tok inFromServer je drugi ulazni tok programa i on je povezan sa soketom. Znaci koji stižu iz mreže ulaze u tok inFromServer. Konačno, tok outToServer je izlazni tok programa i on je takođe povezan sa soketom. Znaci koje klijent pošalje u mrežu ulaze u tok outToServer.

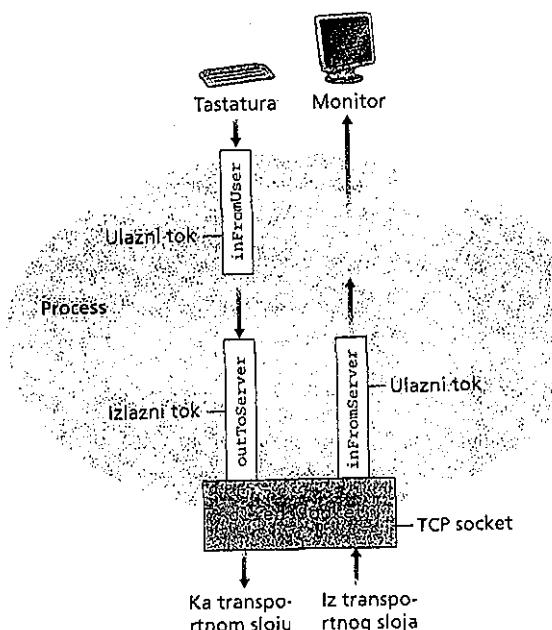
Obratimo sada pažnju na neke redove ovog koda.

```

import java.io.*;
import java.net.*;
  
```

Objekti `java.io.java.net` predstavljaju Java pakete. Paket `java.io` sadrži klase za ulazne i izlazne tokove. Konkretno, paket `java.io` sadrži klase `BufferedReader` i `DataOutputStream` koje program koristi za pravljenje tri prethodno ilustrovana toka. Paket `java.net` obezbeđuje klase za mrežnu podršku. Konkretno, on sadrži klase `Socket` i `ServerSocket`. Objekat `clientSocket` ovog programa izvodi se iz klase `Socket`.

```
class TCPClient {
    public static void main(String argv[])
        throws Exception
    {
        .....
    }
}
```



Slika 2.29 ♦ Program `TCPClient` ima tri toka kroz koje teku znaci

Sve što ste do sada videli predstavlja standardni način na koji započinje većine Java programa. Prvi red predstavlja početak bloka koji definiše klasu. Ključnom rečju `class` počinje definicija klase po imenu `TCPClient`. Klasa sadrži promenljive i metode koji su ograničeni velikim zagradama na početku i na kraju bloka za definiciju klase. Klasa `TCPClient` nema promenljive i ima samo jedan metod - `main()`. Metodi programskog jezika Java podsećaju na funkcije ili procedure programskog jezika C; metod `main` programskog jezika Java veoma je sličan istoimenoj funkciji programskih jezika C i C++. Kada Java interpreter aktivira aplikaciju (pozivanjem putem klase za kontrolu aplikacije), ona odmah poziva klasni metod `main`. Ovaj metod zatim poziva sve ostale metode koji su potrebni za izvršavanje aplikacije. Budući daje ovo uvod u programiranje soketa u programskom jeziku Java, na ovom mestu ćemo da zanemarimo ključne reči `public`, `static`, `void`, `main` i `throws Exceptions` (ove reči ipak moraju da budu u kodu).

```
String sentence;
String modifiedSentence;
```

Dva prethodna reda deklarišu objekte tipa `String`. Objekat `sentence` jeste string koji korisnik upisuje i koji se zatim šalje serveru. Objekat `modifiedSentence` jeste string koji se dobija od servera i šalje ka standardnom izlazu korisnika.

```
BufferedReader inFromUser = new BufferedReader( new
InputStreamReader(System.in));
```

Prethodni red koda pravi objekat toka `inFromUser` tipa `BufferedReader`. Ovaj ulazni tok inicijalizuje se objektom `System.in` koji ga povezuje sa standardnim ulazom. Ova komanda omogućava klijentu da pročita tekst koji je upisan njegovom tastaturom.

```
Socket clientSocket = new Socket("hostname", 6789);
```

Ovim redom koda pravi se objekat `clientSocket` tipa `Socket`. Pored toga, ovaj red inicira TCP konekciju između klijenta i servera. String "hostname" mora da se zameni imenom servera (recimo, "apple.poly.edu"). Pre same inicijalizacije TCP konekcije, klijent izvršava pretraživanje DNS baze podataka kako bi došao do potrebne IP adrese datog matičnog računara. Broj 6789 jeste broj porta. Uvek možete da upotrebite i neki drugi broj porta, ali isti taj broj morate da koristite i na serverskoj strani aplikacije. Kao što smo već rekli, serverski proces se identificuje IP adresom i brojem porta aplikacije.

```
DataOutputStream outToServer =
    new DataOutputStream(clientSocket.getOutputStream()); BufferedReader inFromServer =
    new BufferedReader(new InputStreamReader(
        clientSocket.getInputStream()));
```

Pomoću ova dva reda koda prave se objekti tokova koji se povezuju sa soketom. Tok outToServer obezbeđuje izlaz procesa ka soketu, dok tok inFromServer obezbeđuje ulaz procesa iz soketa (slika 2.29).

```
sentence = inFromUser.readLine();
```

Ovaj red koda zaduženje za smeštanje korisnikovog teksta u string sentence. String sentence nastavlja da prima znakove sve dok korisnik ne završi red znakom za povratak na početak reda. Korisnikov tekst iz standardnog ulaza (tastature), preko toka inFromUser prelazi u string sentence.

```
outToServer.writeBytes(sentence + "\n");
```

Ovaj red koda šalje string sentence koji je proširen znakom za povratak na početak reda u tok outToServer. Ovaj prošireni string protiče kroz klijentov soket i ulazi u TCP. Nakon toga, klijent čeka na prijem znakova od servera.

```
modifiedSentence = inFromServer.readLine();
```

Kada tekst stigne iz servera, on prolazi kroz tok inFromServer i smešta se u string modif iedSentence. Znaci se sakupljaju puneci string modif iedSentence sve do završetka teksta koji je označen znakom za povratak na početak reda.

```
System.out.println("FROM SERVER " + modif iedSentence);
```

Prethodni red koda zaduženje za ispisivanje na monitoru stringa modif iedSentence koji je dobijen od servera.

```
clientSocket.close();
```

Ovim poslednjim redom koda zatvara se soket, a to znači i TCP konekcija između klijenta i servera. To dalje znači da TCP klijentske strane šalje TCP poruku serverskoj strani TCP (pročitajte odeljak 3.5).

TCPServer.java

Pogledajmo sada kako izgleda serverski program.

```
import java.io.*; import java.net.*;
class TCPServer {
    public static void main(String argv[])
        throws Exception {
        String clientSentence; String capitalizedSentence;
        ServerSocket welcomeSocket = new ServerSocket(
            6789); while
        (true) {
            Socket connectionSocket = welcomeSocket._
                accept(); BufferedReader inFromClient =
                new BufferedReader(new InputStreamReader(
                    connectionSocket.getInputStream())); DataOutputStream outToClient =
                new DataOutputStream(
                    connectionSocket.getOutputStream()); clientSentence ~
                inFromClient.readLine(); capitalizedSentence =
                clientSentence . toUpperCase () + " " + "\n" ;
                outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```

Program TCPserver ima mnogo sličnosti sa programom TCPClient. Da vidimo kako ovaj program izgleda izbliza. U nastavku nećemo komentarisati one delove koda koji su identični ili slični sa kodom programa TCPClient.java.

Prvi red programa TCPserver veoma se razlikuje od onoga što ste videli u programu TCPClient i glasi:

```
ServerSocket welcomeSocket = new ServerSocket(6789);
```

Ovim redom koda pravi se objekat welcomeSocket tipa ServerSocket. Ovaj objekat predstavlja svojevrsna vrata koja osluškuju „kucanje“ klijenta i to cine na portu broj 6789. Sledeći red koda je:

```
Socket connectionSocket = welcomeSocket.accept();
```

Ovaj red pravi novi soket po imenu connectionSocket. Čim se klijent obrati objektu welcomeSocket. I ovaj soket ima broj porta 6789. (U poglavlju 3 obja-snićemo zašto oba soketa imaju isti broj porta.) Protokol TCP zatim uspostavlja direktni vod između objekta clientSocket na klijentskoj strani i objekta connectionSocket na serverskoj strani. Klijent i server nakon toga mogu da započnu međusobnu razmenu podataka, a svi bajtovi stižu predviđenim redosledom na drugu stranu. Nakon uspostavljanja objekta connectionSocket, server pomoću objekta welcomeSocket nastavlja da osluškuje eventualne zahteve drugih klijenata. (Ova verzija programa, u stvari, ne osluškuje zahteve drugih klijenata, ali dodavanjem programskih niti to može da se izmeni.) Program zatim pravi nekoliko objekata tokova koji su veoma slični odgovarajućim objektima napravljenim u clientSocket. Pređimo sada na sledeći značajan red;

```
capitalizedSentence = clientSentence.toUpperCase() + "\n";
```

Ova komanda predstavlja srce aplikacije. Ona uzima red teksta koji je poslao klijent, pretvara ga u velika slova i dodaje znak za povratak na početak reda. Komanda koristi metod toUpperCase(). Sve druge komande programa moguće bi da se nazovu perifernim; one služe samo za komuniciranje sa klijentom.

Da biste testirali ovaj par programa, program TCPClient.java ćete instalirati i kompajlirati najednom računam, a program TCPServer.java na drugom. Nemojte zaboraviti da u programu TCPClient.java navedete ime matičnog računara-servera. Nakon toga ćete na serveru da pokrenete kompajlirani program TCPServer.class. Ovaj program na serveru pravi proces koji miruje sve dok sa njim ne kontaktira neki klijent. Potom ćete na klijentskom računaru da pokrenete kompajlirani program TCPClient.class koji će na tom računaru da napravi proces i uspostavi TCP konekciju između klijentskog i serverskog procesa. Konačno, da biste upotrebili napravljenu aplikaciju, upisaćete jednu rečenicu praćenom znakom za povratak na početak reda.

Programiranje sopstvenih klijentsko-serverskih aplikacija možete da započnete malim izmenama ovog programa. Primera radi, umesto pretvaranja teksta u velika slova, server bi mogao da prebroji koliko se puta u tekstu pojavljuje slovo „s“ i da vam vrati samo taj broj.

2.8 Programiranje soketa za protokol UDP

U prethodnom odeljku rekli smo da komunikacija dva procesa preko protokola TCP iz perspektive tih procesa izgleda kao da između njih postoji vod koji je tu sve dok ga jedan od njih ne zatvori. Kada jedan od procesa želi da pošalje neke informacije drugom, on jednostavno pušta svoje bajtove u ovaj vod. Predajni proces ne mora uz bajtove da prilaže i odredišnu adresu, budući da vod logički povezan sa odredištem. Štaviše, ovaj vod obezbeđuje kanal za pouzdani tok podataka - redosled primljenih bajtova u prijemnom procesu identičan je sa redosledom bajtova koje je pošiljalac stavio u vod.

Protokol UDP takođe omogućava komunikaciju dva (ili više) procesa koji se izvršavaju na različitim računarima. Međutim, protokol UDP se po mnogo čemu suštinski razlikuje od protokola TCP. Najpre, protokol UDP je usluga bez konekcije - ovde ne postoji inicijalna faza sinhronizacije tokom koje se između dva procesa uspostavlja vod za podatke. S obzirom na to da kod protokola UDP ne postoji nikakav vod, predajni proces uz svaki paket podataka mora da prilaže odredišnu adresu. I to mora da se učini za svaku grupu bajtova koju predajni proces pošalje. Kao analogiju, zamislite grupu od 20 ljudi koja u pet taksija vozila treba da dođe do zajedničkog odredišta. Kada ovi ljudi uđu u taksije, svakog vozača pojedinačno treba da obaveste u kom pravcu da vozi. Dakle, moglo bi se reći da protokol UDP ima sličnosti sa taksijem. Odredišna adresa predstavlja zapis koga čine IP adresa odredišnog računara i broj porta odredišnog procesa. U nastavku teksta celinu koju Čine paket sa informacijama, odredišna IP adresa i broj porta nazivaćemo samo „paket“. Protokol UDP obezbeđuje nepouzdani model usluge orijentisani na poruke u tom smislu što preduzima sve što može u cilju isporuke paketa bajtova na odredište, ali u tom pogledu ne pruža nikakve garancije. Usluga UDP se bitno razlikuje od pouzdanog modela usluge sa tokovima bajtova protokola TCP.

Nakon što napravi paket, predajni proces ga, putem svog soketa, ubacuje u mrežu. Nadovezujući se na analogiju sa taksijem, mogli bismo da kažemo da se na drugoj strani soketa nalazi taksi koji čeka na paket. Međutim, ovaj taksi ne može da garantuje da će paket zaista stići do svog odredišta; taksi bi mogao da se pokvari ili bi mogao da mu se dogodi neki neočekivani problem. Drugim rečima, *protokol UDP svojim komuniciraj učim procesima obezbeđuje nepouzdanu transportnu uslugu* - on ne pruža nikakve garancije da će datagram stići do svoje odredišne adrese.

U ovom odeljku ilustrovaćemo vam programiranje soketa tako što ćemo da preradimo aplikaciju iz prethodnog odeljka i daje prilagodimo protokolu UDP. Videćete da je Java kod za protokol UDP bitno drugačiji od koda za protokol TCP. Konkretno, videćete da ovde (1) nema inicijalne sinhronizacije između procesa, pa samim tim ni potrebe za postojanjem soketa za dobrodošlicu, zatim da (2) nema tokova koji se povezuju sa soketima, da (3) predajni računar pravi pakete tako što svakoj skupini podataka prilaže odredišnu IP adresu i broj porta i da (4) prijemni proces mora da otvoriti svaki paket kako bi iz njega izvukao bajtove sa informacijama. Još jednom vas podsećamo na našu jednostavnu aplikaciju:

1. Klijent čita red teksta iz standardnog ulaza.
2. Server čita red teksta iz svog soketa.
3. Server prebacuje ovaj tekst u sva velika slova.
4. Server, kroz svoj soket, šalje izmenjeni tekst klijentu.
5. Klijent izmenjeni tekst kroz svoj soket i prikazuje ga na standardnom izlazu (monitoru).

Na slici 2.30 istaknute su osnovne aktivnosti vezane za sokete u situaciji kada klijent i server komuniciraju putem transportne usluge bez konekcije (UDP).

UDPCClient.java

Evo kako izgleda kod klijentske strane aplikacije:

```
import java.io.*; import
java.net.*; class UDPCClient {
    public static void main(String args[]) throws Exception
    {
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader (System.in));
        DatagramSocket clientSocket = new DatagramSocket(); InetAddress IPAddress <
        InetAddress.getByName("hostname"); byte[] sendData =
        new byte[1024]; byte[] receiveData = new byte[1024]; String sentence =
        inFromUser.readLine(); sendData = sentence.getBytes(); DatagramPacket
        sendPacket =
        new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
        clientSocket.send(sendPacket); DatagramPacket receivePacket =
        new DatagramPacket(receiveData, . receiveData.length);
        clientSocket.receive(receivePacket); String modifiedSentence =
        =
        new String(receivePacket.getData());
        System.out.println("FROM SERVER:" +
                           modifiedSentence);
        clientSocket.close();
    }
}
```

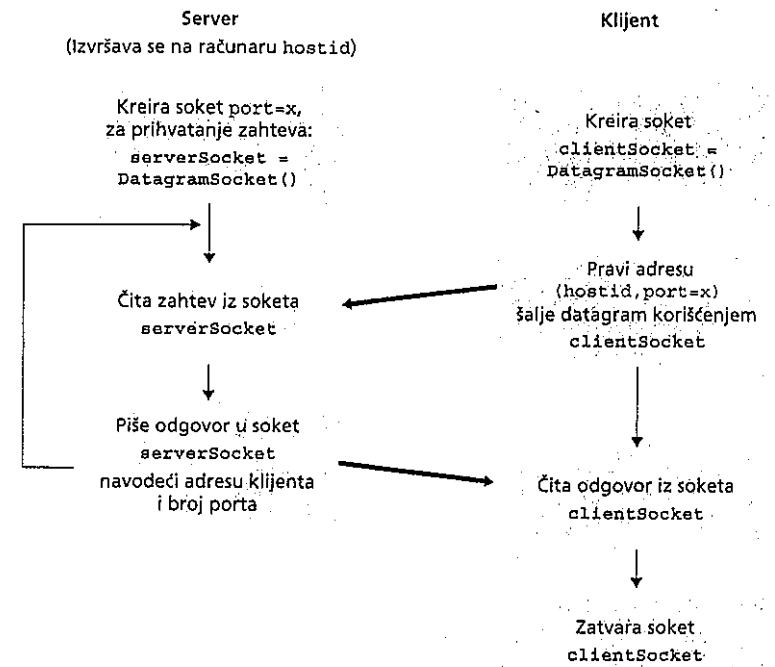
Kao što možete da vidite na slici 2.31, program UDPCClient.java pravi jedan tok i jedan soket. Naziv soketa je clientSocket i on pripada tipu DatagramSocket. Skrećemo vam pažnju na to da protokol UDP na klijentskoj strani koristi različitu vrstu soketa od protokola TCP. Konkretno, u UDP verziji klijent koristi soket DatagramSocket, dok je u TCP verziji to bio soket Socket. Tok inFromUser je ulazni tok programa; povezan je sa

standardnim ulazom, odnosno sa tastaturom. Podsećamo vas da smo u TCP verziji programa imali ekvivalentan tok. Sve što korisnik upiše pomoću tastature ulazi u tok inFromUser. Ali, za razliku od protokola TCP, ovde nema tokova (ni ulaznih ni izlaznih) koji su povezani sa soketom. Umesto da bajtovе sa informacijama šalje u tok koji bi bio povezan sa objektom Socket, protokol UDP sve pojedinačne pakete šalje kroz objekat DatagramSocket.

Obratimo sada pažnju na one redove koda koji se bitnije razlikuju od programa TCPClient.java.

```
DatagramSocket clientSocket = new DatagramSocket();
```

Ovim redom koda pravi se objekat clientSocket tipa DatagramSocket. Nasuprot programu TCPClient.java, ovaj red ne inicira TCP konekciju. Konkretno, nakon izvršenja ovog reda programa klijentski računar ne kontaktira sa serverom,

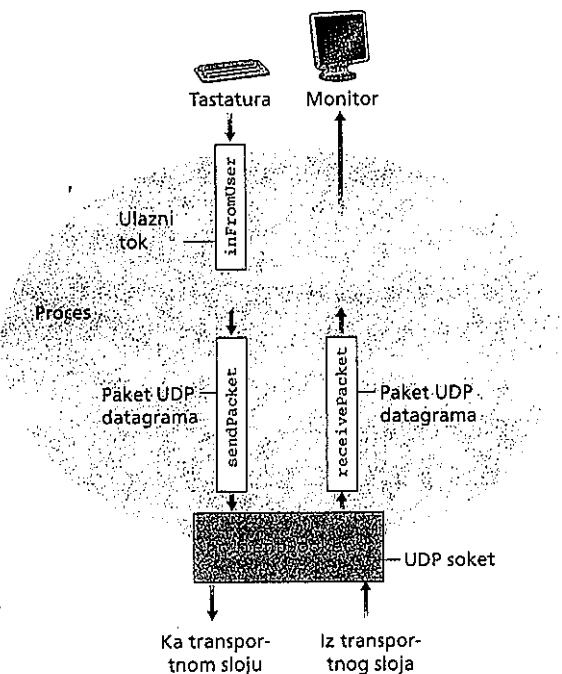


Slika 2.30 ♦ Klijentsko-serverska aplikacija koja koristi transportnu uslugu bez konekcije

Iz tog razloga konstruktor DatagramSocket() ne uzima ime servera i broj porta kao svoje argumente. Ukoliko se, za trenutak, vratimo analogiji sa vratima i vodom, mogli bismo da kažemo da se izvršavanjem ovog reda stvaraju vrata za klijentski proces, ali ne i vod između klijentskog i serverskog procesa.

```
InetAddress IPAddress = InetAddress. . . getByName("hostname");
```

Da bi slanje bajtova sa informacijama određenom procesu bilo moguće, potrebna nam je njegova adresa. Deo ove adrese predstavlja IP adresa određnog računara. Prethodnim redom programa aktivira se pretraživanje DNS baze podataka u kome se ime računara (u ovom slučaju naveo gaje sam programer) prevodi u odgovarajuću IP adresu. I kod TCP verzije klijentskog programa postojalo je pozivanje usluge DNS, s tom razlikom što je to tada učinjeno implicitno a ne eksplisitno. (Metod



Slika 2.31 ♦ Program UDPClient ima jedan tok; soket prihvata pakete od procesa i isporučuje ih procesu.

getByName() kao svoj argument uzima ime računara-servera, a vraća njegovu IP adresu i, zatim, tu adresu smešta u objekat IP Address tipa InetAddress.

```
byte[] sendData = new byte[1024]; byte[] receiveData =
new byte[1024];
```

Nizovi bajtova sendData i receiveData čuvaju podatke koje klijent pošalje i primi.

```
sendData = sentence . getBytes ( ) ,
```

Ovaj red koda, u osnovi, izvršava konverziju tipa. On uzima rečenicu u vidu stringa i menja joj ime u sendData, tj. u objekat koji predstavlja niz bajtova.

```
DatagramPacket sendPacket = new DatagramPacket( sendData, sendData.length,
        IPAddress, 9876);
```

Ovim redom koda pravi se paket sendPacket koji klijentski računar ubacuje u mrežu putem svog soketa. U paketu se nalaze sami podaci, zatim njihova dužina, IP adresa servera i broj porta aplikacije (u ovom slučaju, 9876). Skrećemo vam pažnju na to da objekat sendPacket pripada tipu DatagramPacket.

```
clientSocket.send(sendPacket) ;
```

U ovom redu metoda send() objekta clientSocket uzima upravo konstruisani paket i ubacuje ga u mrežu putem soketa clientSocket. I na ovom mestu vam skrećemo pažnju na to da protokol UDP šalje ovaj red teksta na drugačiji način od protokola TCP. Protokol TCP jednostavno ubacuje string znakova u tok koji ima direktnu logičku konekciju sa serverom. Nasuprot tome, protokol UDP pravi paket u kome je navedena adresa servera. Nakon što pošalje svoj paket, klijent čeka na prijem paketa od servera.

```
DatagramPacket receivePacket =
        new DatagramPacket(receiveData, receiveData.length);
```

Čekajući na prijem paketa od servera, klijent, ovim redom, rezerviše mesto za paket i to je objekat receivePacket tipa DatagramPacket.

```
clientSocket.receive(receivePacket);
```

Sve do prijema paketa, klijent se nalazi u stanju mirovanja; pristigli paket klijent smešta u objekat receivePacket.

```
String modifiedSentence =
    new String(receivePacket.getData());
```

Ovaj red programa izvlači podatke iz objekta receivePacket i izvršava konverziju tipa, pretvarajući niz bajtova u string modif iedSentence.

```
System.out.println("FROM SERVER: " + modifiedSentence);
```

Ovaj red, koji postoji i u programu TCPClient ispisuje string modif iedSentence na klijentovom monitoru.

```
clientSocket.close();
```

Poslednjim redom koda soket se zatvara. Budući da kod UDP veza nema konekcije, klijent serveru nakon ovog reda ne šalje poruku transportnog sloja (nasuprot programu TCPClient).

UDPServer.java

Pogledajmo sada kako izgleda serverska strana ove aplikacije.

```
import java.io.*;
import java.net.*;
class UDPServer
{
    public static void main(String args[])
        throws Exception {
        DatagramSocket serverSocket = new
            DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new
            byte[1024];
        while(true) {
            DatagramPacket receivePacket =
                new DatagramPacket(receiveData,
                    receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence =
                new String(
                    receivePacket.getData());
            InetAddress IPAddress =
                receivePacket.getAddress();
            int port =
                receivePacket.getPort();
            String capitalizedSentence =
                sentence.toUpperCase();
            sendData = capitalizedSentence.getBytes();
            DatagramPacket sendPacket =
                new DatagramPacket(sendData,
                    sendData.length,
                    IPAddress,
                    port);
            serverSocket.send(sendPacket);
        }
    }
}
```

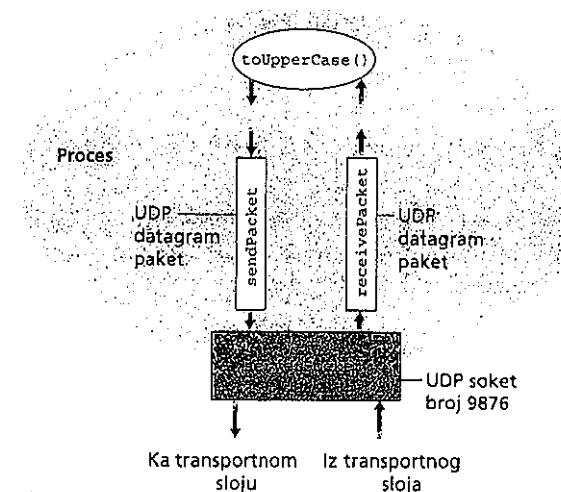
```
sendData = capitalizedSentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData,
    sendData.length, IPAddress, port);
serverSocket.send(sendPacket);
}
}
}
```

Kao što možete da vidite na slici 2.32, program UDPServer.java kreira jedan soket. Čije je ime serverSocket i pripada tipu DatagramSocket (takav soket postoji i na klijentskoj strani aplikacije). Ni u ovom slučaju sa soketom se ne povezuju nikakvi tokovi.

Obratimo sada pažnju na one redove koda koje se razlikuju od programa TCPServer.java.

```
DatagramSocket serverSocket = new DatagramSocket(9876);
```

Ovaj red koda kreira soket serverSocket tipa DatagramSocket na portu 9876. Svi poslati i primljeni podaci prolaze kroz ovaj soket. S obzirom na to da kod protokola UDP nema konekcije, ovde ne moramo da pravimo nov soket i oslušku-



Slika 2.32 ♦ U programu UDPServer nema tokova; soket prihvata pakete od procesa i isporučuje ih procesima.

jemo zahteve za konekcijom kao što je bio slučaj kod programa TCPServer.java. Ukoliko ovoj aplikaciji pristupi veći broj klijenata, svi oni svoje pakete šalju kroz ova vrata - ServerSocket.

```
String sentence = new String(receivePacket.getData()) ; InetAddress IPAddress = receivePacket.getAddress() ; int port = receivePacket.getPort();
```

Ova tri reda programa služe za raspakivanje klijentovog paketa. Prvi red izvlači podatke iz paketa i smešta ih u objekat Stringsentence; veoma sličan red postoji i u programu UDPCClient. Drugi red izvlači IP adresu, a treći *klijentov broj porta* koji je izabrao sam klijent i koji se razlikuje od serverovog porta broj 9876. (O klijentskim brojevima portova detaljnije ćemo govoriti u sledećem poglavlju.) Server mora da dode do klijentove adrese (IP adrese i broja porta) kako bi mogao da mu vrati rečenicu koja je prebačena u velika slova.

Ovim ujedno i završavamo našu analizu programskog para aplikacije za protokol UDP. Aplikaciju možete da proverite tako što ćete program UDPClient.java instalirati i kompajlirati na jednom računaru, a program UDPSServer.java na drugom, (Kod programa UDPClient.java obavezno treba da navedete ime servera.) Nakon toga pokrenite obe ova programa. Za razliku od TCP verzije programa, ovde možete prvo da pokrenete klijentsku stranu aplikacije zato što ona, kada je pokrenete, neće pokušavati da uspostavi konekciju sa serverom. Kada pokrenete i klijentski i serverski program, upisivanjem jednog reda teksta na klijentskom računaru isprobajte ovu aplikaciju.

2.9 Pravljenje jednostavnog veb servera

Sada kada već prilično detaljno poznajete protokol HTTP, a znate i kako se programiraju jednostavne klijentsko-serverske aplikacije u Javi, iskoristićemo ova vaša znanja i njihovim kombinovanjem napraviti jednostavan web server. Videćete da je to u stvari veoma jednostavno.

2.9.1 Funkcije veb servera

U ovom slučaju cilj nam je da napravimo server koji može da učini sledeće:

- ◆ rukuje samo jednim HTTP zahtevom;
 - ◆ prihvata i parsira HTTP zahtev;
 - ◆ uzima traženu datoteku iz svog sistema datoteka;
 - ◆ pravi HTTP poruku sa odgovorom koju čine zaglavlje i tražena datoteka;
 - ◆ Šalje svoj odgovor direktno klijentu.

Potrudili smo se da kod koji sledi bude što jednostavniji, kako vam ne bi smetao da se koncentrišete na aspekte umrežavanja. Naravno, on, zbog toga, nije savršeno stabilan. Primera radi, ovde se uopšte nismo pozabavili rukovanjem izuzecima, a pretpostavicemo i to da se datoteka koju klijent traži zaista nalazi u serverovom sistemu datoteka.

Vv'ebServer.java

Evo kako izgleda kod jednostavnog veb servera

```
import java.io.*; import
java.net.*;
import java.util.*;
class MebServer {
    public static void main(String argv[]) throws Exception {
        String requestMessageLine;
        String fileName;
        ServerSocket listenSocket = new ServerSocket(6789);
        Socket connectionSocket = listenSocket.accept();
        BufferedReader inFromClient =
            new BufferedReader(new InputStreamReader(
                connectionSocket.getInputStream()));
        DataOutputStream outToClient =
            new DataOutputStream(
                connectionSocket.getOutputStream());
        requestMessageLine = inFromClient.readLine();
        StringTokenizer tokenizedLine =
            new StringTokenizer(requestMessageLine); if
        (tokenizedLine.nextToken().equals("GET")){
            fileName = tokenizedLine.nextToken();
            if ((fileName.startsWith("<")) == true )
                fileName ~ fileName.substring(1);
            File file = new File(fileName);
            int numOfBytes = (int) file.length();
            FileInputStream inFile = new FileInputStream (
                fileName);
            byte[] fileInBytes = new byte[numOfBytes];
            inFile.read(fileInBytes) ;
            outToClient.writeBytes(
                "HTTP/1.0 200 Document Follows\r\n");
            if ((fileName.endsWith(".jpg")))
                outToClient.writeBytes("Content-Type:
                    image/jpeg\r\n");
            if ((fileName.endsWith(C.gif")))
```

```

        outToClient.writeBytes("Content-Type: image/gif\r\n");
        outToClient.writeBytes("Content-Length: " +
            numOfBytes + "\r\n"); outToClient.writeBytes("\r\n");
        outToClient.write(fileInBytes, 0, numOfBytes); connectionSocket.close(); }
    else System.out.println("Bad Request Message"); }

}

```

Pogledajmo sada izbliza ovaj kod. Kao što verovatno i sami primećujete, njegova prva polovina gotovo je identična sa programom TCPServer.j ava. Kao i u programu TCPServer.java, ovde smo takođe uveli pakete java.io i java.net. Pored ova dva paketa uveli smo i paket java.util koji sadrži klasu StringTokenizer, koja se koristi za parsiranje HTTP poruka sa zahtevima, U redovima u okviru klase WebServer definisali smo dva string objekta.

```
String requestMessageLine; String fileName;
```

Objekat requestMessageLine jeste string u kome će se nalaziti prvi red HTTP poruke sa zahtevom. Objekat fileName je takođe string u kome će se nalaziti ime tražene datoteke. Sledeći skup komandi identičan je sa odgovarajućim delom programa TCPServer.j ava,

```

ServerSocket listenSocket = new ServerSocket(6789); Socket connectionSocket =
listenSocket.accept(); BufferedReader inFromClient =
new BufferedReader(new InputStreamReader(
    (connectionSocket.getInputStream())); DataOutputStream
outToClient =
    new DataOutputStream(connectionSocket.getOutputStream());
```

Ovim delom programa napravljena su dva objekta iz kategorije soketa. Prvi ovakav objekat je listenSocket koji pripada tipu ServerSocket. Serverski program pravi objekat listenSocket pre nego što od klijenta dobije zahtev za TCP konekcijom. Ovaj objekat osluškuje port 6789 i na zahtev nekog klijenta uspostavlja TCP konekciju. Kada zahtev za uspostavljanjem konekcije stigne, metod accept() objekta listenSocket pravi novi objekat, connectionSocket,

tipa Socket. Zatim se prave dva toka - Buf f eredReader inFromClient i DataOutputStream outToClient. HTTP poruka sa zahtevom klijenta stiže iz mreže i, kroz soket connectionSocket, dolazi do toka inFromClient; HTTP poruka sa odgovorom servera ide u tok outToClient i potom kroz soket connectionSocket izlazi u mrežu. Ostatak ovog koda se, međutim, bitno razlikuje od programa TCPServer.java.

```
requestMessageLine = inFromClient.readLine();
```

Ova komanda Čita prvi red HTTP poruke sa zahtevom koja bi trebalo da bude u obliku:

```
GET file_name HTTP/1.0
```

Nakon toga, naš server treba da izvuče ime datoteke parsiranjem reda.

```
StringTokenizer tokenizedLine =
    new StringTokenizer(requestMessageLine); if
(tokenizedLine.nextToken().equals("GET")){ 'fileName =
tokenizedLine.nextToken(); if (fileName.startsWith("/") == true ) fileName =
fileName.substring(1);
```

Ove komande parsiraju prvi red poruke sa zahtevom i iz njega vade ime tražene datoteke. Objekat tokenizedLine možete da zamislite kao originalan red sa zahtevom u kome su „reči“ GET, file_name i HTTP/1.0 smeštene u odvojeni prostor, koji se zove token. Server zna, na osnovu RFC-a za protokol HTTP, da se ime tražene datoteke nalazi u tokenu koji se nalazi iza tokena "GET". Ovo ime datoteke se potom stavlja u string po imenu fileName. U prethodnom segmentu koda, poslednji iskaz if služi za uklanjanje kose crte koja bi mogla da prethodi imenu datoteke.

```
FileInputStream inFile = new FileInputStream(fileName);
```

Ova komanda povezuje tok inFile sa stringom fileName.

```
byte[] fileInBytes = new byte[numOfBytes]; inFile.read(fileInBytes);
```

Ove komande služe za određivanje veličine datoteke i konstruisanje niza bajtova te veličine, Ime ovog niza je fileInBytes. Poslednja komanda ovog segmenta čita iz niza inFile i zapisuje u niz fileInBytes. Program mora da izvrši ovu

konverziju u bajtove zato što izlazni tok outToClient može da se puni isključivo bajtovima. Sada smo potpuno spremni za sastavljanje HTTP poruke sa odgovorom. Ali, pre toga, moramo da pošaljemo zaglavje HTTP poruke sa odgovorom u tok DataOutputStream outToClient.

```
outToClient.writeBytes("HTTP/1.0 200 Document
Follows\r\n"); if
(fileName.endsWithC.jpg") )
    outToClient.writeBytes("Content-Type: image/jpeg\r\n"); if
{fileName.endsWith(".gif")}
    outToClient.writeBytes("Content-Type: _image/gif\r\n");
outToClient.writeBytes("Content-Length: " + numOfBytes + "\r\n");
outToClient.writeBytes ("\r\n");
```

Ovaj skup komandi je posebno zanimljiv. Inače, ove komande služe za pripremu zaglavja HTTP poruke sa odgovorom i njihovo slanje u izlaznu privremenu memoriju protokola TCP. Prva komanda šalje obavezni statusni red: HTT P /1. 0 2 00 Document FOIIOVJS, koju prate znak za povratak na početak reda i znak za prelazak u novi red. Sledeća dva reda pripremaju zaglavje. Ukoliko bi slao sliku u GIF formatu, server bi tada pripremio zaglavje tipa Content Type : image/ gif. Kada bi to bila slika u JPEG formatu, zaglavje bi glasilo Content Type : image / j peg. (Kod ovog našeg jednostavnog veb servera ne postoji red za tip sadržaja zato što objekat nije ni GIF ni JPEG slika.) Server zatim priprema i šalje red zaglavja koji se odnosi na dužinu sadržaja, kao i obavezan prazan red koji prethodi samom objektu koji se šalje. U ovom trenutku datoteka FileName šalje se u tok DataOutputStream outToClient.

```
outToClient.write(fileInBytes, 0, numOfBytes);
```

Ova komanda šalje traženu datoteku f ileInBytes u prednju memoriju protokola TCP koji će, zatim, daje pridruži upravo napravljenom zaglavju, segmentira (ukoliko je potrebno) i pošalje TCP segmente klijentu. Nakon usluživanja klijentovog zahteva slanjem datoteke, server zatvara soket connectionSocket,

```
connectionSocket.close();
```

Da biste testirali ovaj vebserverski program, instalirajte ga najednom računaru, a zatim na taj računar snimite i nekoliko datoteka. Nakon toga, sa nekog drugog računara čitačem zatražite neku od ovih datoteka. Prilikom traženja datoteke moraćete

da koristite onaj broj porta koji ste naveli u serverskom programu (na primer, 6789). Dakle, ako pretpostavimo da se vaš server nalazi na računaru somehost. some-where . edu, daje u pitanju datoteka somef ile. html i daje naveden port broj 6789, zahtev vašeg Čitača treba da izgleda ovako:

<http://somehost.somewhere.edu:6789/somefile.html>

2.10 Rezime

U ovom poglavlju govorili smo o konceptualnim i implementacionim aspektima mrežnih aplikacija. Upoznali ste se sa sveprisutnom klijentsko-serverskom paradigmom Internet aplikacija, a zatim ste videli i na koji način ovu paradigmu koriste protcoli HTTP, FTP, SMTP, POP3 i DNS. Govoreći o ovim veoma važnim protokolima aplikacijskog sloja, detaljno smo vam predstavili aplikacije sa kojima su oni povezani (Web, transfer datoteka, elektronska pošta i sistem imena domena). Nastavili smo pregledom savremenih tehnika za distribuciju sadržaja u mreži u koje spadaju proksi serveri za veb keširanje, mreže sa distribucijom sadržaja i P2P razmema datotek. Zatim ste videli na koji način soket API može da se iskoristi za pravljenje mrežnih aplikacija. Detaljno smo prošli kroz proces korišćenja soketa u transportnim uslugama sa konekcijom (TCP), kao i u onim transportnim uslugama bez konekcije (UDP), da bismo zatim, korišćenjem soketa, napravili jednostavan veb server. Naš prvi korak u putovanju od vrha slojevite mrežne arhitekture ka njenom dnu ovim je završen.

Na samom početku knjige, u odeljku 1.1 ponudili smo relativno nepreciznu i pojednostavljenu definiciju protokola rekavši da on određuje „format i redosled poruka koje razmenjuju dva ili više komunicirajućih entiteta, kao i akcije koje oni preduzimaju nakon prijema ovih poruka“. Ono što smo u ovom poglavlju rekli o protokolima HTTP, FTP, SMTP, POP3 i DNS, dodalo je značajnu težinu ovoj definiciji. Protokoli predstavljaju ključne aspekte umrežavanja; prikaz protokola aplikacijskog sloja će vam zasigurno pomoći da razvijete nešto intuitivniji osjećaj za najznačajnije aspekte samih protokola.

U odeljku 2.1 opisali smo modele usluga koje protcoli TCP i UDP nude aplikacijama koje ih pozovu. Razvijajući jednostavne aplikacije za protokole TCP i UDP (odeljci 2.7 - 2.9) videli smo kako ti modeli usluga uzgledaju izbliza. Ipak, rekli smo veoma malo o tome *kako* ovi protokoli obezbeđuju pomenute modele usluga. Primera radi, gotovo da ništa nismo rekli o načinu na koji protokol TCP obezbeđuje pouzdani transport svojim aplikacijama, U sledećem poglavlju ćemo zato obratiti pažnju ne samo na pitanje *Šta*, već i na aspekte *kako* i *zašto* kod transportnih protokola. Naoružani znanjima o strukturi Internet aplikacija i protokolima aplikacijskog sloja, spremni ste za nastavak našeg puta naniže kroz familiju protokola i ispitivanje transportnog sloja, o kome ćemo govoriti u poglavlju 3.

Domaći zadatak; problemi i pitanja

Poglavlje 2 Kontrolna pitanja

ODEUAK 2.1

1. Navedite pet nevlasničkih Internet aplikacija, kao i protokole aplikacijskog sloja koje one koriste.
2. U čemu je razlika između arhitekture mreže i arhitekture aplikacije?
3. U kom smislu instantna razmena poruka predstavlja hibridnu varijantu klijent-sko-serverske i P2P arhitekture?
4. U komunikacionoj sesiji između dva računara, ko je klijent, a ko server?
5. Da li se slažete sa siedećom izjavom: „U komunikacionim sesijama aplikacija za P2P razmenu datoteka ne postoji pojmovi klijenta i servera“. Obrazložite svoj odgovor,
6. Pomoću kojih informacija proces koji se izvršava najednom računaru prepoznaće proces koji se izvršava na drugom računaru?
7. Navedite nekoliko korisničkih agenata aplikacijskog sloja koje svakodnevno koristite.
8. Vraćajući se za trenutak na sliku 2.4 možemo da zaključimo da nijedna od navedenih aplikacija nije istovremeno osetljiva i na gubljenje podataka i na vreme. Možete li da zamislite aplikaciju koja bi zahtevala nepostojanje gubljenja podataka uz istovremenu vremensku osetljivost?

ODEUC12.2-2.6

9. Šta se podrazumeva pod protokolom za sinhronizaciju?
10. Zbog Čega se u podlozi protokola HTTP, FTP, SMTP, POP3 i IMAP nalazi protokol TCP a ne protokol UDP?
11. Zamislite veb lokaciju za elektronsku trgovinu koja želi da čuva podatke o svakom svom korisniku. Opišite kako je to moguće učiniti pomoću kolačića,
12. U Čemu je razlika između postojanih HTTP veza sa cevovodnom obradom i postojanih HTTP veza bez cevovodne obrade? Koju varijantu koristi verzija HTTP/I.I?
13. Objasnite na koji način veb keširanje može da skrati vreme potrebno za prijem traženog objekta. Da li ono skraćuje kašnjenje svih zatraženih objekata ili samo nekih od njih? Obrazložite svoj odgovor.
14. Uspostavite Telnet sesiju sa veb serverom i pošaljite višeredni zahtev. U zahtev ubacite zaglavje If-modif ied-since : kako biste u odgovoru dobili statusnu poruku 304 Not Modified.

16. Zbog čega se kaže da protokol FTP šalje kontrolne informacije „izvan opsega“?
17. Pretpostavimo da Alisa sa svog naloga za elektronsku poštu koji je otvoren na Webu (recimo, Hotmail) želi da pošalje poruku Bobu koji svom serveru za elektronsku poštu pristupa protokolom POP3. Opišite put ove poruke od Ali-sinog do Bobovog računara. Obavezno navedite seriju protokola aplikacijskog sloja koji se koriste za prenos poruke sa jednog na drugi računar.
18. Odštampajte zaglavje neke poruke koju ste nedavno primili. Koliko zaglavja tipa Received: ima u njoj? Analizirajte svako zaglavje poruke.
19. U čemu je, gledano iz perspektive korisnika, razlika između režima „preuzmi i briši“ i režima „preuzmi i zadrži“ protokola POP3?
20. Da li je moguće da veb server i server za elektronsku poštu neke organizacije imaju identični pseudonim kao svoje ime (recimo, foo. com)? Kog tipa bi bio RR zapis koji bi sadržao ime servera za elektronsku poštu?
21. Šta u sistemu za P2P razmenu datoteka predstavlja preklopljena mreža? Da li u njoj postoje ruteri? Šta su njene grane? Na koji način se formira i održava Gnu-tellina preklopljena mreža?
22. Pronadite tri kompanije koje se bave P2P razmenom datoteka. Koju vrstu sadržaja one distribuiraju? Na koji način svaki od njihovih dizajna omogućava korisnicima da lociraju određeni sadržaj?

ODEUCI 2.7-2.9

22. UDP serveru koji je opisan u odeljku 2.S bio je potreban samo jedan soket, dok su TCP serveru iz odeljka 2.7 bila potrebna dva. Zašto? Kada bi TCP server morao da podrži *n* istovremenih veza, koliko bi mu soketa bilo potrebito?
23. Zbog čega u klijentsko-serverskoj aplikaciji koja koristi protokol TCP iz odeljka 2.7 serverski program mora da se pokrene pre klijentskog? Zbog čega u klijentsko-serverskoj aplikaciji za protokol UDP (odeljak 2.8) klijentski program može da se pokrene pre serverskog?

Problemi

1. Tačni ili netačno?

- a) Pretpostavimo da je neki korisnik zatražio veb stranu koja se sastoji od tekstualnog dela i dve slike. Za ovu stranu klijent šalje jednu poruku a prima tri.
- b) Istom postojanom vezom mogu da se pošalju dve različite veb strane (na primer, www.mit.edu/research.html i www.mit.edu/stu-dents.html).
- c) Kada se između čitača i servera porekla uspostavi nepostojana konekcija, jedan TCP segment može da prenese dve različite HTTP poruke sa zahtevima.

- d) Zaglavje Date: u HTTP odgovoru pokazuje kada je objekat koji se nalazi u odgovoru poslednji put izmenjen.
2. Pročitajte RFC dokument 959 koji se odnosi na protokol FTP, a zatim navedite sve klijentske komande koje su podržane ovim dokumentom.
 3. Posetite veb stranu <http://www.iana.org>, a zatim navedite dobro poznate brojove portova za protokole SFTP (*Simple File Transfer Protocol*) i NNTP (*Network News Transfer Protocol*).
 4. Zamislite HTTP klijenta koji putem određenog URL-a treba da primi dokument sa Weba. IP adresa HTTP servera je inicijalno nepoznata. Veb dokument na datom URL-u ima jednu ugradenu sliku u GIF formatu koja se nalazi na istom serveru na kome je i originalni dokument. Koji su protokoli transportnog i aplikacijskog sloja (osim protokola HTTP) potrebeni u ovom scenariju?
 5. Pronadite specifikaciju HTTP/1.1 (RPC 2616) i zatim odgovorite na sledeća pitanja:
 - a) Objasnite mehanizme za signaliziranje kojima se klijent i server obaveštavaju da je trajna konekcija zatvorena. Da li klijent može da signalizira zatvaranje veze? Može li to da učini server i mogu li i jedan i drugi to da učine?
 - b) Koje usluge za Šifrovanje obezbeđuje protokol HTTP?
 6. Prepostavimo da ste u svom Čitaču izborom hiperveze otvorili neku veb stranu čija IP adresa nije keširana u vašem lokalnom računaru tako daje za njeno pribavljanje neophodno pretraživanje DNS baze podataka. Takođe ćemo prepostaviti i to da je dobijanjem IP adrese prethodila poseta » DNS servera; za sukcesivne posete potrebno je RTT vreme, od $RTT_p - RTT_n$. Zatim, ova veb strana je povezana sa linkom koji se odnosi na samo jedan objekat - malo HTML teksta, Sa RTT_0 označićemo RTT vreme između lokalnog računara i servera na kome se ovaj objekat nalazi. Ukoliko zanemarimo trajanje prenosa objekta, koliko vremena protekne od trenutka kada korisnik izabere hipervezu do trenutka kada klijent primi traženi objekat?
 7. Vraćajući se na problem 6, prepostavimo da HTML datoteka referencira tri veoma mala objekta na istom serveru. Ako zanemarimo trajanje prenosa, koliko je vremena potrebno kod:
 - a) nepostojanih HTTP veza bez para/efnih TCP konekcija,
 - b) nepostojanih HTTP veza sa paralelnim TCP konekcijama i
 - c) postojanih HTTP veza sa cevovodnom obradom?
 8. Dva HTTP metoda za traženje objekata su GET i POST. Da li u verziji HTTP/ 1.0 postoje još neki metodi? Ukoliko postoje, za šta se koriste? Da li postoje u verziji HTTP/1.1?
 9. Vratimo se sada na sliku 2.11 na kojoj je nacrtana mreža jedne institucije koja je povezana sa Internetom. Prepostavimo da je prosečna veličina objekta

900 000 bitova i da prosečna učestalost zahteva koje čitači ove institucije šalju serverima porekla iznosi 1,5 zahteva u sekundi. Takođe ćemo prepostaviti da od trenutka kada ruter na strani pristupnog linka ka Internetu prosledi HTTP zahtev pa do trenutka kada primi odgovor, prosečno proteknu dve sekunde (odeljak 2.2.6). Modelujte ukupno prosečno trajanje odziva kao sumu prosečnog kašnjenja kod pristupa (kašnjenje od ruta na rnetetu do rutera institucije) i prosečnog kašnjenja na Internetu. Prilikom izračunavanja prosečnog kašnjenja pristupa koristite formulu $A/(l - A/?)$, gde A predstavlja prosečno trajanje slanja objekta pristupnim linkom, a $?$ brzinu pristizanja objekata u pristupnom linku.

- a) Odredite ukupno prosečno vreme odziva.
 - b) Prepostavimo, za trenutak, daje keš instaliran u LAN-u institucije i daje indeks pronađenih objekata 0,4. Odredite sada ukupno trajanje odziva.
 10. Napišite jednostavan TCP program za server koji treba da prihvati nekoliko redova teksta koji je upisan na klijentskom računaru i izbacuje ih na svom standardnom izlazu. (Ovo možete da uradite i menjanjem programa TCPServer. j ava koji smo naveli u tekstu.) Komppajlirajte i pokrenite svoj program. Zatim na bilo kom drugom računaru koji ima čitač Weba podešite opciju za proksi server Čitač i usmerite ga na server na kome se izvršava vaš program; ne zaboravite i da na odgovarajući način konfigurirate broj porta, Vaš čitač bi nakon toga trebalo da pošalje svoje poruke tipa GET vašem serveru na čijem standardnom izlazu bi trebalo da vidite ispisane poruke. Pomoću ove platforme ustavljajte da li vaš Čitač generiše uslovne GET poruke za objekte koji se nalaze u lokalnoj keš memoriji.
 11. Pročitajte RFC dokument 1939 koji se odnosi na protokol POP3. Koja je uloga komande UIDL POP3?
 12. Zamislite da treba da proverite svoju elektronsku poštu putem protokola POP3. a)
- Prepostavimo da ste svog POP klijenta za elektronsku poštu konfigurisali tako da radi u režimu „preuzmi i obrisi“. Izvršite sledeću transakciju:
- ```

C: list
S: 1 498
. S: 2 912
S: .
C: retr L
S: bla bla . . .
S: bla
S: .
?
```

- b) Prepostavimo da ste svog POP klijenta za elektronsku poštu konfigurisali tako da radi u režimu „preuzmi i zadrži“. Izvršite sledeću transakciju:

```
C: list S: 1
498 S: 2 912
S: .
C: retr 1
S: bla bla ...
S: bla
S: . ?
```

- c) Prepostavimo da ste svog POP klijenta za elektronsku poštu konfigurisali za rad u režimu „preuzmi i zadrži“. Prepostavimo da ste korišćenjem transkripta u delu pod (b) primili poruke 1 i 2, da ste, zatim, **izašli iz** POP klijenta i da ste nakon pet minuta ponovo proverili svoju poštu. Prepostavimo da u **tih** pet minuta niste dobili nijednu novu poruku. Ispišite transkript ove druge POP sesije.

13. a. Staje *whois* baza podataka?
- b. Pomoću raznih *whois* baza podataka na Internetu pronađite **imena** dva DNS servera. Navedite koje ste *whois* baze podataka koristili.
- c. Pomoću programa *nslookup* sa svog lokalnog računara pošaljite DNS upite ka tri DNS servera - svom lokalnom DNS serveru i DNS serverima koje ste pronašli u delu (b). Pokušajte da ispitate izveštaje tipa A, NS i MX. Rezimirajte rezultate.
- d. Programom *nslookup* pronađite veb server koji **ima** više IP adresa. Da li veb server vaše institucije (škole ili kompanije) **ima** više IP adresa?
- e. Pomoću ARIN *whois* baze podataka odredite opseg IP adresa **koji** koristi vaš univerzitet.
- f. Opišite način na koji napadač pomoću *whois* baza podataka i alatke *nslookup* može da prikupi obaveštenja o instituciji pre samog napada.
- g. Obrazložite opravdanost javne dostupnosti *whois* baza podataka.
14. Prepostavimo da pomoću neke P2P aplikacije upravo preuzimate određene MP3 datoteke. Usko grlo vaše veze sa Internetom predstavlja vaš pristupni link koji ima brzinu od 128 kb/s u punom dupleksnom režimu. Tokom vašeg preuzimanja, 10 korisnika počelo je da preuzima MP3 muziku sa vašeg računara. Ukoliko prepostavimo daje **vaš** računar dovoljno **jak** da ga istovremeno preuzimanje i slanje podataka ne opterećuju (procesor, disk, itd), da li će **isto-vremeno** slanje podataka koji takođe moraju da prođu kroz usko grlovaše **veze** sa Internetom usporiti vaše preuzimanje? Obrazložite svoj odgovor, Sta bi se

dogodilo kada biste kao deo ADSL veze imali nizvodni kanal od 512 kb/s i uzvodni od 128 kb/s?

15. Prepostavimo da u Gnutella mreži ima aktivnih korisnika i da između svakog para korisnika postoji aktivna TCP konekcija. Prepostavimo i to da TCP konekcije prolaze kroz ukupno JW rutera. Koliko čvorova i grana ima u ovoj preklopljenoj mreži?
16. Govoreći o mreži Gnutella u odeljku 2.6 detaljno smo vam opisali na koji način novi ravnopravni korisnici ulaze u ovu mrežu. U ovom problemu zanima nas šta se događa kada neki ravnopravni korisnik izade iz ove mreže. Prepostavimo da svaki učestvujući čvor sve vreme održava TCP konekcije sa najmanje četiri druga korisnika, Prepostavimo i to da korisnik Peer X, koji održava TCP konekcije sa pet drugih korisnika, želi da istupi iz mreže.
  - a) Najpre razmislite o elegantnom izlasku pri kome Peer A eksplicitno zatvara svoju aplikaciju, odnosno prekida svih pet svojih TCP konekcija. Šta u tom slučaju preduzima pet ravnopravnih korisnika sa kojima je on prethodno bio povezan.
  - b) Sada prepostavimo da je ^iznenada prekinuo svoju vezu sa Internetom a da nije obavestio svojih pet suseda. Šta se u ovom slučaju događa?
17. U ovom problemu istražićemo rutiranje unazad QueryHit poruka u Gnutellinoj mreži. Prepostavimo daje Alisa izdala komandu Query i daje više takvih poruka stiglo do Boba (od nekoliko posredujućih ravnopravnih korisnika) koji ima datoteku koja odgovara upitu.
  - a) Podsećamo vas da ravnopravni korisnik koji ima traženu datoteku šalje svoju QueryHit poruku obrnutim smerom iste putanje kojom je i dobio Query poruku. Alternativni dizajn bio bi da Bob sa Alisom uspostavi direktnu TCP konekciju i preko nje pošalje poruku QueryHit. Šta su prednosti, a šta nedostaci ovakvog alternativnog dizajna?
  - b) Kada Alisa pošalje poruku Query, protokol Gnutella u njeno polje Messa-gelID ubacuje jedinstveni ID broj. Ukoliko ravnopravni korisnik Bob ima traženu datoteku, njegov program generiše QueryHit poruku sa identičnim poljem MessageID. Opišite način na koji ravnopravni korisnici pomoću ovih MessageID polja i lokalnih tabela rutiranja izvode rutiranje u obmu-tom smeru.
  - c) Postoji i alternativni pristup u kome se ne koriste identifikatori poruke. Kada poruka sa upitom stigne do datog korisnika on joj, pre nego stoje prosledi, dodaje i svoju IP adresu. Objasnite kako se korišćenjem ovog mehanizma izvodi rutiranje obrnutim smerom.
18. Ponovite problem 17, ali na pitanja odgovorite tako da se, umesto na poruke Query i QueryHit, odnose na Gnutelline poruke Ping i Pong.

19. U ovom problemu ispitaćemo dizajn sistema koji podseća na mrežu KaZaA u kome postoje obični čvorovi, vode grupe i vođe supergrupa.
  - a) Prepostavimo daje svaki voda supergrupe odgovoran za otprilike 200 voda grupa od kojih je svaki, dalje, odgovoran za otprilike 200 običnih Članova, Koliko je voda supergrupe potrebno za mrežu u kojoj bi bilo četiri miliona korisnika?
  - b) Koje informacije može da Čuva svaki voda grupe, a koje svaki voda supergrupe? Na koji način se izvodi pretraživanje u ovakvom troslojnem dizajnu.
20. Vratimo se sada na poplavu upita u P2P razmeni datoteka iz odeljka 2.6. Prepostavimo daje svaki ravnopravni korisnik preklopljene mreže povezan sa A<sup>s</sup>useda i da je u polju za brojanje čvorova inicijalno podešena vrednost K. Takođe ćemo prepostaviti da Alisa treba da izvrši upit. Odredite gomju granicu broja upita koji se šalju u preklopljenu mrežu.
21. Instalirajte i kompjajlirajte Java programe TCPClient i UDPClient na jednom računaru i TCPServer i UDPServer na drugom.
  - a) Prepostavimo da ste program TCPClient pokrenuli pre programa TCPServer. Šta se dogada i zašto?
  - b) Prepostavimo da ste program UDPClient pokrenuli pre programa UDPServer. Šta se dogada i zašto?
  - c) Šta se dogada ukoliko na klijentskoj i serverskoj strani upotrebite različite brojeve porta?
22. Promenite program TCPServer . java tako da može da prihvati više veza. (Savet: Moraćete da koristite niti.)
23. Prepostavimo da u programu UDPClient. java red

DatagramSocket clientSocket = new DatagramSocket( ); zamenimo redom

DatagramSocket clientSocket = new DatagramSocket(5432);

Da li ćemo morati da promenimo i program UDPServer .java? Koji se brojevi portova koriste za sokete u programima UDPClient i UDPServer? Kako je bilo pre ove promene?

## Teze za diskusiju

1. Šta mislite, zbog čega su aplikacije za P2P razmenu datoteka toliko popularne? Da li zato što se putem njih (pričinjeno ilegalno) distribuiraju muzika i video materijal ili, možda, zato što kod njih ogroman broj servera efikasno odgovara na masivnu potražnju?
2. Postoji li neki aktivan projekat u smislu pravljenja otvorenog sistema za P2P razmenu datoteka koji bi primenio heterogenost ravnopravnih korisnika (kao što je slučaj u vlasničkoj tehnologiji KaZaA)?
3. Pročitajte rad „The Darknet and the Future of Content Distribution“ autora Biddle, England, Peinado i VVillman [Biddle 2003] i nakon toga kažite da li se slažete sa svim idejama ovih autora. Obrazložite svoj odgovor.
4. Lokacije za elektronsku trgovinu, ali i druge veb lokacije često imaju pozadinske baze podataka. Na koji način HTTP serveri komuniciraju sa ovim bazama podataka?
5. **Staje** dinamički HTML? Navedite primere veb lokacija koje koriste dinamički HTML.
6. Navedite **neke** popularne serverske skript jezike. **Staje** svrha serverskih skript jezika?
7. Na koji način biste svoj čitač konfigurisali za lokalno keširanje? Koje opcije vam u tom smislu stope na raspolažanju?
8. Da li biste svoj čitač mogli da konfigurišete tako da istovremeno otvara više konekcija sa određenom **web** lokacijom? Navedite prednosti i nedostatke **postojanja** većeg **broja** istovremenih TCP konekcija.
9. U odeljku 2.4.4 rekli smo da se usluga elektronske pošte bazirane na Webu često implementira korišćenjem **web** i IMAP servera. Na koji način HTTP server komunicira sa IMAP serverom?
10. Videli ste da TCP soketi podatke koji se prenose tretiraju kao niz bajtova, dok UDP soketi prepoznaju granice između poruka. Koja je prednost a koji nedostatak ovakvog bajtovskog orientisanog API-ja u odnosu na API koji eksplicitno podržava granice poruka definisanih od strane aplikacije?
11. Prepostavimo da imate bazu sa astrološkim podacima koju želite da učinite dostupnom za čitače Weba na personalnim računarima, WAP telefone i obične telefone (korišćenjem konverzije teksta u govor). Na koji način biste mogli to da učinite?
12. Objasnite odnos između instantne razmene poruka i protokola SIP.
13. Na koji način funkcionišu liste kontakata servera za instantnu razmenu poruka?
14. Šta je Apache veb server, koliko košta i koje funkcije trenutno posede?

16. Prepostavimo da su organizacije za standardizaciju Weba odlučile da promene konvenciju imenovanja tako da se svaki objekat naziva i referencira jedinstvenim imenom koje je nezavisno od lokacije (tzv. „URN“). Izložite neke aspekte koji bi pratili ovaku promenu.
17. Analizirajte sistem kojim aplikacija KaZaA podstiče korisnike da u što većoj meri predaju svoje datoteke. Na koji način bi to još moglo da se učini?

### Zadaci sa programiranjem soketa Zadatak 1:

#### Višenitni veb server

Kada završite ovaj programerski zadatak, imaćete višenitni veb server u program-, skom jeziku Java koji istovremeno može da opsluži više zahteva. U ovom slučaju implementiraćete verziju 1.0 protokola HTTP, onako kako je on definisan u dokumentu RFC 1945.

Podsećamo vas da protokol HTTP/1.0 uspostavlja posebnu TCP konekciju za svaki par zahtev-odgovor. Svakom od ovih konekcija rukuje posebna nit. U programu postoji i glavna nit u kojoj server osluškuje klijente koji žele da uspostave konekciju. Da bismo vaš zadatak pojednostavili, program ćemo razviti u dve etape. U prvoj etapi napisaćete kod višenitnog servera koji samo prikazuje sadržaj HTTP zahteva koje dobija. Kada ustanovite da ovaj deo programa ispravno funkcioniše, dodaćete mu i deo koda koji treba da generiše željeni odgovor.

Tokom razvijanja koda svoj server uvek možete da proverite pomoću Čitača Weba. Skrećemo vam pažnju na to da ćete u okviru URL-a koji dajete svom čitaču morati da naznačite broj porta zato što, u ovom slučaju, nećete ići kroz standardni port 80. Primera radi, ukoliko prepostavimo da je ime računara host. someschool . edu, zatim da vaš server osluškuje port 6789 i da želite da preuzmete datoteku index. html, u čitaču biste morali da navedete sledeći URL:

<http://host.someschool.edu:6789/index.html>

Kada se suoči sa greškom, "vaš server bi trebalo da vam pošalje poruku sa odgovarajućom HTML datotekom kako bi informacija o grešci mogla da se prikaže u prozoru čitača. Sve detalje, u vezi sa ovim zadatkom, kao i važne delove Java koda možete da pronađete na veb lokaciji <http://www.awl.com/kurose-ross>.

#### Zadatak 2: Klijent za elektronsku poštu

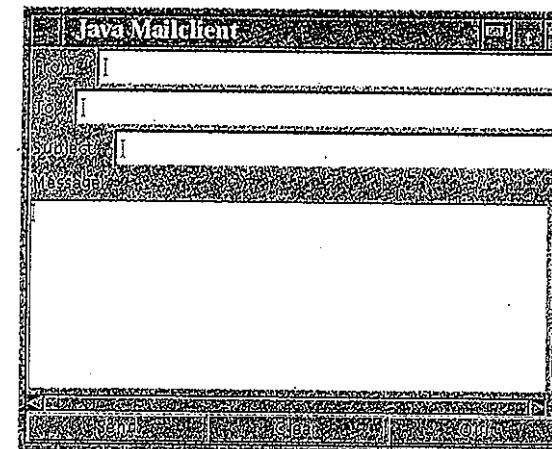
U ovom zadatku ćete, takođe u programskom jeziku Java, razviti korisničkog agenta za elektronsku poštu koji ima sledeće karakteristike:

\* Obezbeduje grafički interfejs za pošiljaoca sa poljima za adrese pošiljaoca i primaoca, temu poruke i samu poruku.

- ♦ Uspostavlja direktnu TCP konekciju sa serverom za elektronsku poštu. Šalje i prima SMTP komande od lokalnog servera za elektronsku poštu.

Evo kako vaš interfejs treba da izgleda:

Korisnički agent koji ćete razviti moći će da šalje poruku samo jednom primaocu u datom trenutku u vremenu. Osim toga, vaš korisnički agent treba da prepostavi



daje deo adrese primaoca koji se odnosi na domen identičan sa imenom SMTP servera koji rukuje pristiglim poštom datog primaoca. (Korisnički agent ne izvršava DNS pretraživanje MX zapisa, tako da pošiljalac mora da obezbedi i ime servera za elektronsku poštu.) Kompletne detalje u vezi sa ovim zadatkom, kao i važne delove Java koda možete da pronađete na veb lokaciji <http://www.awl.com/kurose-ross>.

#### Zadatak 3: Laboratorijsko vežbanje UDP Ping

U ovom laboratorijskom vežbanju implementiraćete jednostavnog Ping klijenta i servera zasnovanog na protokolu UDP. Po svojim funkcijama ovi programi su veoma slični standardnom programu Ping koji je deo savremenih operativnih sistema. Standardni Ping šalje ICMP (*Internet Control Message Protocol*) ECHO poruku koju udaljeni računar treba da vrati pošiljaocu. Pošiljalac na osnovu dobijenih informacija zatim može da odredi vreme povratnog puta između sebe i računara kome je poslao poruku.

U Javi ne postoji funkcionalnost za slanje ili prijem ICMP poruka. Upravo zato ćete Ping u ovom vežbanju da implementirate na aplikacijskom sloju sa standardnim UDP soketima i porukama. Kompletne informacije u vezi sa ovim zadatkom, kao i važne segmente Java koda možete da pronađete na veb lokaciji <http://www.awl.com/kurose-ross>.

### Zadatak 4: Veb proksi server

U ovom vežbanju ćete razviti jednostavan veb proksi server koji treba da kešira veb strane. Ovaj server od čitača prihvata GET poruke i prosleđuje ih odredišnom veb serveru, od koga, s druge strane, prihvata HTTP odgovore i prosleđuje ih čitaču. Reč je o veoma jednostavnom proksi serveru koji razume samo jednostavne GET zahteve. Ipak, ovaj server treba da rukuje svim objektima - ne samo HTML stranicama, već i slikama. Sve detalje o ovom zadatku, kao i relevantne segmente Java koda možete da pronađete na veb lokaciji <http://www.awl.com/kurose-ross>.

## Ethereal laboratorijske vežbe

### Ethereal vežba: HTTP

Budući da ste se u prvom laboratorijskom vežbanju upoznali sa programom Ethereal, ovu alatku sada ćemo da upotrebimo kako biste videli protokole na delu. U ovom vežbanju ispitaćemo nekoliko aspekata protokola HTTP, kao što su: osnovna interakcija između poruka GET i odgovora, formati HTTP poruka, prijem velikih HTML datoteka, prijem HTML datoteka sa ugrađenim URL-ovima, postojane i nepostojane veze i provera autentičnosti i bezbednost ovog protokola.

Kao i kod svih drugih Ethereal laboratorijskih vežbi, pun opis ove vežbe pronaći ćete na veb lokaciji koja prati ovu knjigu - <http://www.awl.com/kurose-ross>.

### Ethereal vežba: DNS

U ovom vežbanju analiziraćemo izbliza klijentsku stranu protokola DNS koji imena matičnih računara na Internetu prevodi u IP adresu. Podsećamo vas na odeljak 2.5 u kome smo rekli da je uloga klijenta u sistemu DNS relativno jednostavna - on šalje upit svom lokalnom DNS serveru i od njega prima odgovor. Ono što se događa u okviru hijerarhije DNS servera koji rekurzivno ili iterativno rešavaju dobijeni zahtev, DNS klijent ne može da vidi. Iz njegove perspektive ovaj protokol je sasvim jednostavan - lokalnom DNS serveru se šalje upit i zatim se od njega očekuje odgovor. Dakle, u ovom laboratorijskom vežbanju pratćemo rad protokola DNS.

Detaljan opis ove vežbe pronaći ćete na pratećoj veb lokaciji za ovu knjigu - <http://www.awl.com/kurose-ross>.

## INTERVJU

### Tim Berners-Li

Tim Berners-Li je direktor Konzorcijurria World Wide Weba (W3C) i Glavni istraživač MIT laboratorije za računarske nauke.<sup>1</sup> Godine 1989, radeći za European Particle Physics Laboratory CERN, Tim je pokrenuo hiperr'edijolnu inicijativu za globalnu razmenu informacija zasnovanu na Internetu ^ojo je mnogo poznatija kao World Wide Web. Godinu dona kasnije napisao je program za prvi veb klijent i veb server. Diplomirao, je fiziku na Oksfordu.



Studirali ste fiziku. Da li postoje sličnosti između fizike i umrežavanja? Studirajući fiziku polazite od pravila ponašanja lokalnih razmara i razmišljate o tome koja bi od njih mogla da se uzdignu na svetske razmere. Prilikom dizajniranja globalnog sistema, kao što je Web, pokušavate da definišete pravila ponašanja veb strana, linkova i ostalih elemenata koji bi, posmatrani u celini, mogli da formiraju svet širokih razmara, onakav kakav biste išeli. Jedno je analiza, drugo sinteza, ali su na neki način prilično slični.

Šta je uticalo da kao svoju specijalnost izaberete umrežavanje?

Nakon što sam diplomirao fiziku, kompanije koje su se bavile telekomunikacionim istraživanjem učinile su mi se najzanimljivije. Mikroprocesor se upravo pojavio i oblast telekomunikacija je veoma brzo počela da prelazi sa ožičenih sistema na sisteme zasnovane na procesorima. Bilo je to veoma zanimljivo.

Šta je najizazovniji deo vaše posla?

Kada se dve grupe izuzetno snažno razilaze u svojim mišljenjima, iako im je krajnji cilj isti pronaalaženje tačnog značenja njihovih pozicija i toga u čemu je nesporazum može biti izuzetno zahtevno. Svako ko je ikada vodio neku radnu grupu zna na šta mislim. Međutim to je veoma važna uloga za napredovanje ka konsenzusu na nekom višem nivou

Kako zamišljate budućnost umrežavanja i Interneta?

Kao što sam napisao u mojoj knjizi, sanjam Veb... i taj san ima dva dela.

U prvom delu Web će postati mnogo moćnije sredstvo za zajednički rad ljudi. Informacijski prostor sam uvek zamišljao kao nešto čemu svi mogu da pristupe odmah i intuitivno - ne samo da bi pretraživali, već i da bi stvarati... Štaviše, elektronska komunikacija ljudi kroz zajednička znanja mora biti moguća u svim razmerama i onoliko jednostavna koliko je danas lična komunikacija,

U drugom delu sna ova saradnja se proširuje i na računare. Računari će postati sposobni da analiziraju sve podatke na Webu - sadržaj, veze i transakcije između ljudi i računara. „Semiški V/eb”, koji to treba da omogući, tek treba da se pojavi, ali kada ga dobijemo, svakodnevnim mehanizmima trgovine, birokratije i naših života upravljače računari koji komuniciraju sa drugim računarima; na ljudima će bili samo da obezbede inspiraciju i intuiciju... Web koji mašine mogu da razumeju dobijemo kroz implementaciju serije tehničkih, pronalazaka i društvenih sporazuma koji su sada na samom početku svog razvoja.

Kada se ostvari ovaj dvodelni san, imaće VVeb koji predstavlja idealnu i moćnu simbiozu ljudskih bića i mašinskog načina razmišljanja.

Koji su vas ljudi najviše inspirirali u profesionalnom smislu?

Moji roditelji, koji su učestvovali u prvim danima računarstva, trajno su me fascinirali čitavom ovom temom, Majk Sendal i Pegi Rimer od kojih sam mnogo naučio na CERN-u, takođe su među onim ljudima koji su me učili i ohrabrili. Kasnije sam naučio da poštujem ljude, kao što su Vanevar Buš, Dag Englebart i Ted Nelson koji su svojevremeno imali *veoma* slične snove, ali nisu imali na raspolaganju personalne računare i Internet da ih i ostvare.

Misija Konzordjuma World Wide VVeb jeste „da vodi VVeb ka ostvarivanju njegovog punog potencijala”. Šta je, prema vašem mišljenju, pun potencijal Weba i na koji način on može da se ostvari?

Mi pokušavamo da ga vodimo u dva smera koja sam pomenuo - ka saradnji i ka semiškom Webu,,Sve vreme pokušavamo da proširimo univerzalnost VVeba u tom smislu da postoji samo jedan Web - bez obzira na vaš Čitač - Web koji je uvek dostupan i kome svako može da pristupi bez obzira na razlike u pogledu hardverskih uređaja, softvera, geografskog položaja, hindekiperanosti, jezika i kulture.



## TRANSPORTNI SLOJ

Kako se nalazi između aplikacijskog sloja i mrežnog sloja, transportni sloj predstavlja centralni deo slojevite mrežne arhitekture. On igra presudnu ulogu u direktnom povezivanju aplikacijskih procesa koji se izvršavaju na različitim računarima. U ovom poglavljju, naizmenično ćemo izlagati principe transportnog sloja i primenu tih principa u postojećim protokolima; kao i obično, poseban naglasak se stavlja na Internet protokole, naime, protokole transportnog sloja: TCP i UDP.

Počevemo sa opisivanjem odnosa između transportnog i mrežnog sloja. Pozabaviti ćemo se jednom od ključnih funkcija transportnog sloja, koja se ogleda u sledećem: usluga isporučivanja - koja se između dva krajnja sistema obavlja u mrežnom sloju - nastavlja se u transportnom sloju, pa se dobija isporučivanje između dva procesa iz aplikacijskih slojeva tih krajnjih sistema. Ovu funkciju ilustrujemo prikazom Internetovog transportnog protokola bez UDP konekcije.

Zatim se vraćamo izlaganju jednog od najosnovnijih problema umrežavanja računara - kako omogućiti pouzdalu komunikaciju preko medijuma na kojem može doći do gubitaka i oštećenja podataka, Nizom sve složenijih (a realnih!) scenarija izgradićemo tehnike koje transportni protokoli koriste u rešavanju ovih problema. Zatim ćemo prikazati kako su ovi principi ugradeni u TCP, Internetov transportni protokol sa konekcijom.

Nakon toga, prelazimo na drugi osnovni problem umrežavanja - kontrolu brzine prenosa entiteta u transportnom sloju kako bi se izbeglo zagušenje u mreži ili omogućio oporavak nakon zagušenja. Razmotrićemo uzroke i posledice zagušenja, kao i ubičajene tehnike za njegovu kontrolu. Kada dobro savladamo pitanja kontrole zagušenja, proučićemo TCP-ov pristup pojmu problematičnosti.

### 3.1 Usluge transportnog sloja

U prethodna dva poglavlja pomenuli smo ulogu transportnog sloja i usluge koje ga obezbeđuje. Ukratko ćemo ponoviti ono što smo o transportnom sloju već naučili.

Protokol transportnog sloja obezbeđuje logičku komunikaciju između aplikacionih procesa koji se izvršavaju na različitim računarima. Pod logičkom komunikacijom podrazumevamo da, iz perspektive aplikacije, izgleda kao da su računari na kojima se procesi izvršavaju direktno povezani; u praksi, računari se mogu nalaziti na suprotnim krajevima planete, povezani preko niza rutera i različitih vrsta linkova. Za međusobno slanje poruka, aplikacijski procesi koriste logičku komunikaciju koju obezbeđuje transportni sloj, oslobođeni brige o detaljima fizičke infrastrukture koja se koristi za prenos tih poruka. Na slici 3.1 prikazan je pojma logičke komunikacije.

Kao što se vidi na slici 3.1, protokoli transportnog sloja implementirani su u krajnjim sistemima, ali ne i u mrežnim ruterima. Mrežni ruteri koriste jedino polja mrežnog sloja u jedinici podataka PDU-3 (tj. jedinici podataka u protokolu mrežnog sloja); oni ne koriste polja transportnog sloja. Na otpremnoj strani, transportni sloj konverte poruke dobijene od aplikacionog procesa pošiljaoca u PDU-4 (tj. jedinicu podataka u protokolu transportnog sloja). Ovo se postiže (eventualnom) podelom aplikacione poruke na manje komade i dodavanjem zaglavja transportnog sloja na svaki takav deo da bi se napravio PDU-4. Transportni sloj zatim predaje PDU-4, mrežnom sloju gde se svaki PDU-4 enkapsulira u PDU-3. Na prijemnom kraju, transportni sloj prima PDU-4 od mrežnog sloja ispod njega, uklanja transportno zaglavje sa jedinice podataka PDU-4, ponovo sastavlja poruku i prosleđuje je aplikacijskom procesu koji je prima.

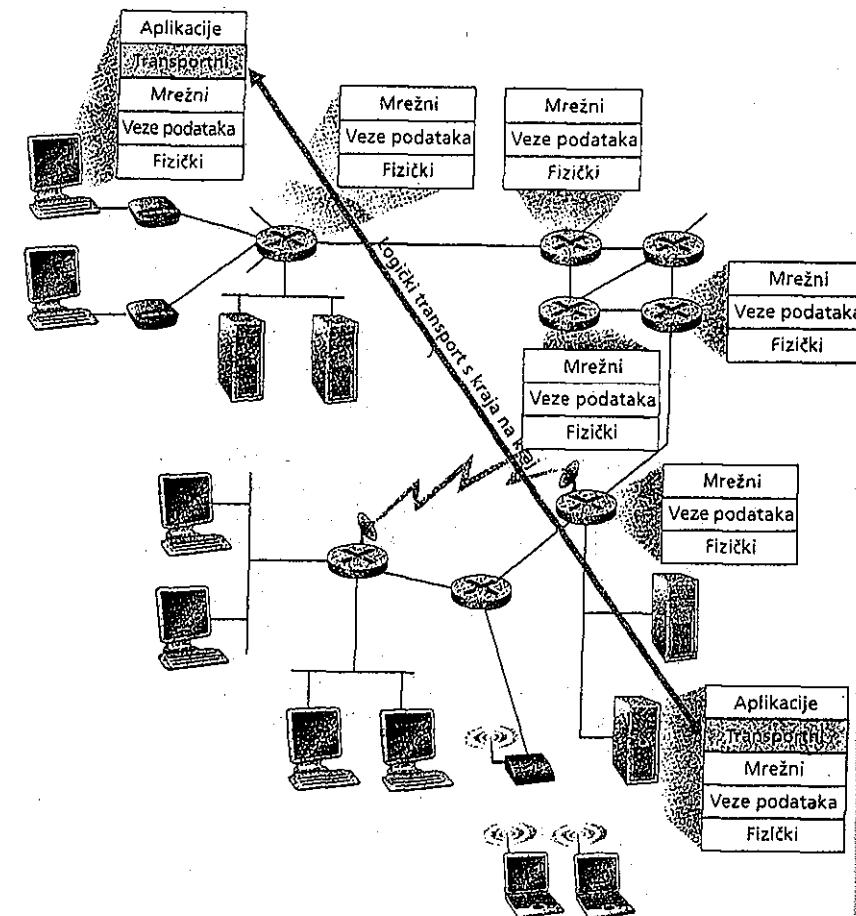
Mrežnim aplikacijama je često dostupno više protokola transportnog sloja. Na primer, Internet ih ima dva - TCP i UDP. Svaki od ovih protokola obezbeđuje drugačije usluge transportnog sloja pozivnoj aplikaciji.

#### 3.1.1 Odnos između transportnog i mrežnog sloja

Verovatno znate da se u familiji protokola transportni sloj nalazi tačno iznad mrežnog sloja. Dok protokol transportnog sloja obezbeđuje logičku komunikaciju između procesa koji se izvršavaju na različitim računarima, protokol mrežnog sloja

obezbeđuje logičku komunikaciju između računara. Ova razlika je suptilna, ali značajna. Objasnimo ovu razliku pomoću analogije iz svakodnevnog života.

Uzmimo dve kuće, jednu na Istočnoj, a drugu na Zapadnoj obali. U svakoj od tih kuća stanuje po dvanaestoro dece. Deca u kući na Istočnoj obali su rođaci dece u kući na Zapadnoj obali. Deca se dopisuju - svako dete svake nedelje piše po



**Slika 3.1** ♦ Transportni sloj obezbeđuje logičku, a ne fizičku komunikaciju između aplikacionih procesa.

jedno pismo svakom rođaku, a svako pismo se otprema u zasebnoj koverti, običnom poštrom. Tako svaka kuća svake nedelje šajje 144 pisma onoj drugoj kući. (Koliko bi samo ova deca uštedela kada bi imala e-poštu!) U svakoj od ovih kuća, po jedno dete -Ana na Zapadnoj obali i Bil na Istočnoj - zaduženo je za prikupljanje i distribuciju pošte. Svake nedelje Ana obilazi svu svoju braću i sestre, prikuplja poštu i predaje je poštanskom službeniku, koji svakoga dana obilazi kuću. Kada pisma stignu u kuću na Zapadnoj obali, Antun zadatuk je da podeći poštu svojoj braći i sestrama. Na Istočnoj obali, iste te poslove obavlja Bil.

U ovom primeru, poštanska služba obezbeđuje logičku komunikaciju između dve kuće - poštanska služba prenosi poštu od kuće do kuće, a ne od osobe do osobe. S druge strane, Ana i Bil obezbeđuju logičku komunikaciju između rođaka - Ana i Bil preuzimaju i isporučuju poštu svojoj braći i sestrama. Obratite pažnju na to da iz perspektive rođaka, Ana i Bil jesu poštanska služba, iako oni obavljaju samo jedan deo isporuke s kraja na kraj (deo unutar krajnjeg sistema). Ovaj primer predstavlja lepu analogiju kojom ćemo opisati odnos transportnog i mrežnog sloja:

poruke iz aplikacija = pisma u kovertama procesi

= rođaci

računari (ili krajnji sistemi) = kuće protokol

transportnog sloja = Ana i Bil

protokol mrežnog sloja = poštanska služba (uključujući i poštare)

Ako pratimo dalje ovu analogiju, uočavamo da Ana i Bil sav svoj posao obavljaju u vlastitim kućama; oni nisu uključeni, na primer, u sortiranje pošte u nekom tranzitnom poštanskom centru niti u prenošenje pošte iz jednog centra u drugi. Isto važi za protokole transportnog sloja koji se nalaze u krajnjim sistemima. Unutar krajnjeg sistema transportni protokol prenosi poruke od aplikacijskih procesa do mreže (tj. do mrežnog sloja) i obratno, ali ni na koji način ne utiče na način prenošenja poruke preko mreže. U suštini, karšto se vidi na slici 3.1, usputni ruteri ne koriste, niti prepoznaju, informacije koje je transportni sloj dodata poruci iz aplikacije.

Vratimo se našoj analogiji, i pretpostavimo da Ana i Bil idu na odmor, pa drugi par rođaka - na primer, Suzan i Harvi - preuzimaju njihove uloge prikupljanja i isporučivanja pošte unutar kuće. Na žalost te dve porodice, Suzan i Harvi ne prikupljaju i ne isporučuju tačno na isti način kao Ana i Bil. Pošto su mlađi, Suzan i Harvi prikupljaju i dele poštu rede, a povremeno i gube pisma (ponekad ih sažače njihovi psi). Prema tome, Suzan i Harvi ne pružaju iste usluge (tj. isti model usluge) kao Ana i Bil. Analogno tome, računarska mreža obezbeđuje različite transportne protokole i svaki od tih protokola nudi aplikacijama drugačiji model usluga.

Usluge koje su Ana i Bil u stanju da ponude jasno su ograničene mogućnostima poštanske službe. Na primer, ako poštanska služba ne garantuje maksimalno vreme isporuke između dve kuće (na primer 3 dana), Ana i Bil ni na koji način ne mogu garantovati koliko će rođaci najviše čekati na isporuku. Slično tome, usluge koje transportni protokol može da obezbedi često su ograničene modelom usluga svojstvenim protokolu mrežnog sloja. Ako protokol mrežnog sloja ne garantuje kašnjenje ili propusni opseg za slanje jedinice podataka PDU-4 od računara do računara, ni protokol transportnog sloja ne može da garantuje kašnjenje niti propusni opseg za poruke koje se šalju od procesa do procesa.

Međutim, transportni protokol može da ponudi neke usluge čak i kada mrežni protokol u osnovi ne nudi odgovarajuće uslugu u mrežnom sloju. Na primer, kao što ćemo videti u ovom poglavljiju, transportni protokol može da ponudi aplikaciji uslugu pouzdanog transfera podataka čak i kada mrežni protokol nije naročito pouzdan, tj. čak i kada mrežni protokol gubi, ošteće i duplira pakete. Drugi primer (koji ćemo detaljnije obraditi u poglavljiju 7 kada budemo izlagali o bezbednosti u mreži) bio bi da transportni protokol može koristiti Šifrovanje kojim se garantuje da aplikacijske poruke neće čitati neovlašćena lica. Čak i ako mrežni sloj ne može da garantuje poverljivost za PDU-4.

### 3.1.2 Kratak pregled transportnog sloja u Internetu

Verovatno znate da Internet, pa i svaka TCP/IP mreža, aplikacijskom sloju nudi dva različita transportna protokola. Jedan od tih protokola je UDP (*User Datagram Protocol*) koji pozivajući aplikaciju obezbeđuje nepouzdanu uslugu bez konekcije. Drugi protokol je TCP (*Transmission Control Protocol*), koji pozivajući aplikaciju nudi pouzdanu uslugu sa uspostavljanjem konekcije. Pri projektovanju mrežne aplikacije programer mora da navede jedan od ova dva transportna protokola. Kao što smo videli u odeljcima 2.6 i 2.7, programer bira protokol UDP ili TCP kada pravi soket.

Da bi terminologija bila jednostavnija, u kontekstu Interneta PDU-4 ćemo zvati segmentom. Pomenućemo, međutim, da se u internetskoj literaturi (na primer, u RFC-ovima) PDU za TCP takođe naziva segmentom, ali se PDU za UDP naziva datagramom. Ista ta internetska literatura koristi izraz datagram i za PDU mrežnog sloja! Smatramo da ćemo u uvodnoj knjizi o umrežavanju računara, kakva je ova, spreciti zabunu ako PDU za TCP i UDP zovemo segment, a ostavimo izraz datagram za PDU mrežnog sloja.

Pre nego što pređemo na kratak uvod u UDP i TCP, dodaćemo nekoliko reči o mrežnom sloju Interneta. (Mrežni sloj je detaljno obrađen u poglavljju 4.) Inter-netski protokol mrežnog sloja je IP što je skraćenica od Internet Protocol. IP obezbeđuje logičku komunikaciju između računara. Model usluge IP-a je najbolji

**mogući pokušaj isporuke.** To znači da IP čini „sve što može“ da isporuči segmente između računara, ali ne pruža nikakvu garanciju. Konkretno, ne garantuje isporuku segmenta, ne garantuje redosled isporuke segmenata i ne garantuje integritet podataka u segmentima. Zato se za IP kaže daje **nepouzdana usluga**. Pomenućemo takođe da svaki računar ima bar jednu adresu mrežnog sloja, takozvanu IP adresu. U poglavljiju 4 detaljno ćemo razmotriti IPadresiranje; ovde je jedino važno imati naumu da *savki računar ima IP adresu*.

Nakon ovog kratkog prikaza IP modela usluga, opisaćemo ukratko modele usluga koje pružaju UDP i TCP. Njihova osnovna uloga je da prošire IP uslugu ispo-ručivanja između dva krajnja sistema i na uslugu isporučivanja između dva procesa koji se izvršavaju na krajnjim sistemima. Proširivanje isporuke od računara do računara na isporuku od procesa do procesa naziva se multipleksiranje i demultipleksiranje transportnog sloja. Multipleksiranje i demultipleksiranje transportnog sloja predstavlja temu sledećeg odeljka. UDP i TCP takođe obezbeđuju proveravanje integriteta tako što u svojim zaglavljima sadrže polja za otkrivanje grešaka. Ove dve osnovne usluge transportnog sloja - isporuka podataka od procesa do procesa i pro-vera grešaka-jedine su usluge koje pruža UDP! Konkretno, kao i IP, UDP je nepouzdana usluga - on ne garantuje da će podaci koje šalje jedan proces, stići neoštećeni (ili stići uopšte!) od odredišnog procesa. UDP je detaljno opisan u odeljku 3.3.

S druge strane, TCP nudi aplikacijama nekoliko dodatnih usluga. Pre svega, on nudi pouzdan transfer podataka. Pomoću kontrole toka, rednih brojeva, potvrda prijema i tajmera (tehnika koje ćemo detaljno obraditi u ovom poglavljiju), TCP obezbeđuje da se podaci isporuče pravilno i u ispravnom redosledu od izvornog procesa do prijemnog procesa. Na ovaj način, TCP pretvara nepouzdanu uslugu IP između krajnjih sistema u pouzdanu uslugu **za** prenos podataka između procesa. TCP takođe obezbeđuje **kontrolu** zagušenja. Kontrola zagušenja je pre usluga za Internet kao celinu, usluga **za** opšte dobro, a ne toliko usluga koja se pruža aplikaciji koja je inicira. Slobodno govoreći, TCP kontrola zagušenja sprečava da neka TCP konekcija zatrpa preteranom količinom saobraćaja linkove i komutatore koji se nalaze između računara u komunikaciji. U principu, TCP obezbeđuje jednak deo propusnog opsega svim TCP konekcijama koje prelaze preko zagušenog mrežnog linka. Ovo se postiže regulisanjem brzine kojom TCP na strani pošiljaoca šalje saobraćaj u mrežu. Nasuprot tome, UDP saobraćaj nije regulisan. Aplikacija koja koristi UDP transport može da šalje proizvoljnom brzinom dokle god to hoće.

Protokol koji obezbeđuje pouzdan transfer podataka i kontrolu zagušenja mora da bude složen. Biće nam potrebno nekoliko odeljaka za opis principa pouzdanog transfera podataka i kontrole zagušenja, kao i dodatni odeljci za opis samog protokola TCP. Ove teme obradene su u odeljcima od 3.4 do 3.8. U ovom poglavljiju naizmenično izlažemo osnovne principe i njihovu primenu u protokolu TCP. Na primar, prvo opisuјemo pouzdan transfer podataka u opštem smislu, a zatim konkretan

način na koji TCP obezbeđuje pouzdan transfer podataka. Slično tome, prvo opisujući kontrolu zagušenja u opštem smislu, a nakon toga način na koji TCP kontroliše zagušenje. Ali pre nego što se upustimo u sve to, razmotrimo najpre multipleksiranje i demultipleksiranje transportnog sloja.

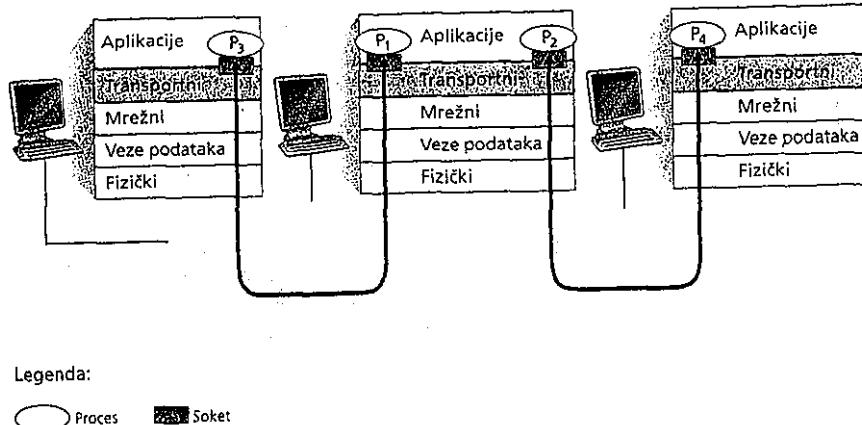
## 3.2 Multipleksiranje i demultipleksiranje

U ovom odeljku opisujemo multipleksiranje i demultipleksiranje transportnog sloja, to jest proširivanje usluge isporuke od računara do računara koju pruža mrežni sloj, na uslugu isporuke od procesa do procesa za aplikacije koje se izvršavaju na računarama. Da bi naše izlaganje ostalo konkretno, opisaćemo osnovnu uslugu transportnog sloja u kontekstu Interneta. Naglašavamo, međutim, daje usluga multipleksiranja i demultipleksiranja potrebna u svim računarskim mrežama.

U odredišnom računaru, transportni sloj prima segmente (tj. PDU transportnog sloja) od mrežnog sloja koji se nalazi neposredno ispod njega. Transportni sloj zadužen je da isporuči podatke iz tih segmenata odgovarajućem aplikacionom procesu koji i se izvršava na računaru. Pogledajmo jedan primer. Uzmimo da sedite pred računarom i da preuzimate veb stranice dok se izvršavaju jedna FTP sesija i dve Telnet sesije. Prema tome, izvršavaju se Četiri mrežna aplikaciona procesa - dva Telnet procesa, jedan FTP proces i jedan HTTP proces. Kada transportni sloj u vašem računaru prima podatke od mrežnog sloja nižeg nivoa, on mora da usmeri primljene podatke jednom od ova četiri procesa. Sada ćemo videti kako se to radi.

Pre svega, setite se iz odeljaka 2.6 i 2.7 da proces (kao deo mrežne aplikacije) ima soket koji predstavlja vrata kroz koja podaci iz mreže prolaze prema procesu i obratno. Prema tome, kako je prikazano na slici 3.2, transportni sloj u prijemnom računaru u suštini ne isporučuje podatke direktno procesu, već posredničkom soketu. Pošto u svakom pojedinačnom trenutku u prijemnom računaru postoji više soketa, svaki od njih ima jedinstven identifikator. Format identifikatora zavisi od toga da li je reč o UDP soketu ili TCP soketu, što ćemo obraditi malo kasnije.

Razmotrimo sada kako prijemni računar usmerava dolazni segment iz transportnog sloja u odgovarajući soket. Svaki segment transportnog sloja ima za ovu svrhu jedan skup polja u segmentu. Na prijemnom kraju, transportni sloj ispituje ova polja da bi odredio prijemni soket i zatim usmerava segment u taj soket. Ovaj zadatak isporučivanja podataka iz segmenta transportnog sloja u odgovarajući soket naziva se demultipleksiranje. Zadatak prikupljanja delova podataka u i2vomom računaru sa različitih soketa, enkapsuliranje svakog dela zaglavljem (koje će se kasnije koristiti za demultipleksiranje) da bi se napravio segment, i predavanje segmenata mrežnom sloju, naziva se multipleksiranje. Obratite pažnju na to da transportni sloj



**Slika 3.2** ♦ Multipleksiranje i demultipleksiranje u transportnom sloju

srednjem računaru na slici 3.2 mora da demultipleksira segmente koje dobija iz nižeg mrežnog sloja da bi ih predao procesu  $P_j$  ili  $P_3$  iznad njega; to se postiže usmeravanjem podataka dolaznog segmenta na odgovarajući soket za taj proces. Transportni sloj u srednjem računaru mora takođe da prikuplja odlazne podatke sa tih soketa, da formira segmente transportnog sloja i predaje te segmente naniže, mrežnom sloju.

Da bismo ilustrovali demultipleksiranje, setite se metafore sa domaćinstvom iz prethodnog odeljka. Svako dete prepoznamo po imenu. Kada Bil primi gomilu pošte od poštara, on je demulcipteksira tako što utvrđuje na koga su pisma adresirana i uručuje pisma svojoj braći i sestrama. Ana multipleksira poštu kada prikuplja pisma od braće i sestara i predaje ih poštaru.

Pošto smo shvatili uloge multipleksiranja i demultipleksiranja u transportnom sloju, pogledajmo kako se to zaista odvija u računaru. Iz prethodnog opisa nam je jasno daje za multipleksiranje u transportnom sloju potrebno da (1) soketi imaju jedinstvene identifikatore i da (2) svaki segment sadrži posebna polja koja određuju soket u koji treba taj segment isporučiti. Ta posebna polja su **polje broja izvornog porta** i **polje broja odredišnog porta** (slika 3.3). (UDP i TCP segmenti sadrže i druga polja o kojima ćemo govoriti u sledećim odeljcima ovog poglavlja.) Svi brojevi portova su 16-bitni, u rasponu od 0 do 65535. Brojevi portova od 0 do 1023 nazivaju se **dobro poznatim brojevima portova** i rezervisani su za dobro poznate aplikacijske protokole, kao što su HTTP (koji koristi broj porta 80) i FTP (koji koristi broj porta 21). Spisak dobro poznatih brojeva portova može se naći u RFC-u 1700, i

u njegovoj ažurnoj verziji na adresi <http://www.iana.org> [RFC 3232]. Kada razvijamo novu aplikaciju (kao sto je to učinjeno u odeljcima od 2.6 do 2.8), moramo dodeliti aplikaciji broj porta.

Sada bi trebalo da je jasno kako bi transportni sloj mogao da implementira uslugu demultipleksiranja: svakom soketu u računaru dodelio bi se broj porta, a kada segment stigne u računar transportni sloj bi ispitao broj odredišnog porta u segmentu i usmerio segment na odgovarajući soket. Podaci iz segmenta prošli bi kroz soket u odgovarajući proces. Kao što ćemo videti, UDP u osnovi tako i funkcioniše. Međutim, kao što ćemo takođe videti, multipleksiranje i demultipleksiranje u TCP-u ipak je složenije.

#### Multipleksiranje i demultipleksiranje bez uspostavljanja konekcije

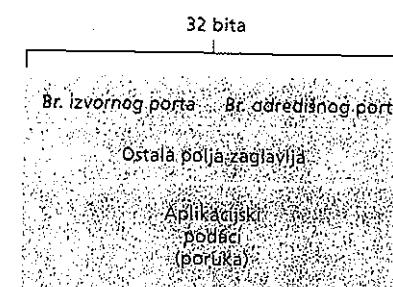
Setite se iz odeljka 2.7 da Java program koji se izvršava na računari može da napravi UDP soket pomoću naredbe:

```
DatagramSocket mvSocket = new DatagramSocket();
```

Kada se na ovaj način napravi UDP soket, transportni sloj automatski dodeljuje soketu broj porta. Konkretno, transportni sloj dodeljuje broj porta u rasponu od 1024 do 65535 koji trenutno ne koristi nijedan drugi UDP port u lačunaru. Java program bi takođe mogao da napravi soket pomoću naredbe:

```
DatagramSocket mvSocket = new DatagramSocket(19157);
```

U ovom slučaju, aplikacija dodeljuje UDP soketu određeni broj soketa - 19157. Ako programer piše kod za serverski deo „dobro poznatog protokola“, on mora da dodeli



**Slika 3.3** ♦ Polja sa brojevima izvornog i odredišnog porta u segmentu transportnog sloja

odgovarajući dobro poznati broj porta. Obična klijentska strana aplikacije dozvoljava da transportni sloj automatski (**i** transparentno) dodeli broj porta, dok se na serverskoj strani aplikacije dodeljuje tačno određeni broj porta.

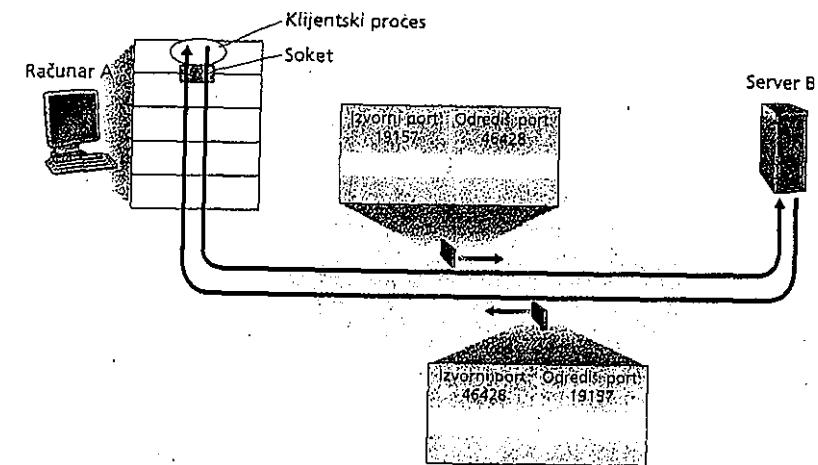
Pošto srao UDP soketima dodelili brojeve portova, možemo precizno da opišemo UDP multipleksiranje **i** demultipleksiranje. Uzmimo da, u računaru A, proces sa UDP soketom 19157 želi da pošalje aplikacijski podatak procesu sa UDP soketom 46428 u računaru B. Transportni sloj u računaru A napraviće segment transportnog sloja koji obuhvata aplikacijske podatke, izvorni broj porta (19157), odredišni broj porta (46428) **i** još dve vrednosti (koje ćemo kasnije opisati, a nisu važne za sadašnje izlaganje). Transportni sloj zatim predaje dobijeni segment mrežnom sloju. Mrežni sloj enkapsulira segment u IP datagram i daje sve od sebe da isporuči segment prijemnom računaru. Ako segment stigne prijemnom računaru B, ovaj ispituje broj odredišnog porta u segmentu (46428) i isporučuje segment u svoj soket određen portom 46428. Treba napomenuti da računar B možda izvršava više procesa od kojih svaki ima vlastiti UDP soket sa odgovarajućim brojem porta. Kako UDP segmenti stižu sa mreže, računar B usmerava (demultipleksira) svaki segment odgovarajućem soketu na osnovu broja odredišnog porta u segmentu.

Važno je primetiti da se UDP soket potpuno identificuje 2-torkom koja se sastoji od odredišne IP adrese **i** odredišnog broja porta. Prema tome, ako dva UDP segmenta imaju različite izvorne IP adrese **i**/ili različite izvorne brojeve portova, a istu *odredišnu* IP adresu **i** *odredišni* broj porta, ta dva segmenta će se usmeriti istom odredišnom procesu kroz isti odredišni soket.

Možda se sada pitate koja je svrha broja izvornog porta. Kao što je prikazano na slici 3.4, u segmentu A-prema-B broj izvornog porta služi kao deo „adrese za odgovor“ - kada B želi da vrati segment računaru A, odredišni port u segmentu B-prema-A uzima vrednost izvornog porta iz segmenta A-prema-B. (Potpuna adresa za odgovor sastoji se od IP adrese računara A **i** izvornog broja porta.) Kao primer za ovo može poslužiti program UDP servera koji smo radili u odeljku 2.7. U programu UDPServer.java, server koristi metod kojim vadi broj izvornog porta iz segmenta koji je primio od klijenta; zatim šaće klijentu novi segment u kojem se dobijeni broj izvornog porta koristi kao broj odredišnog porta u novom segmentu.

#### Multipleksiranje i demultipleksiranje pri uspostavljenoj konekciji

Da **bi** se shvatilo TCP demultipleksiranje, moramo bolje da razmotrimo TCP sokete **i** uspostavljanje TCP konekcije. Jedna suptilna razlika između TCP soketa **i** UDP soketa je u tome što se TCP soket identificuje pomoću 4-tvorke: izvoma IP adresa, izvorni broj porta, odredišna IP adresa **i** odredišni broj porta. Prema tome, kada TCP segment stigne sa mreže na računar, računar koristi sve četiri vrednosti da **bi** usmeno (demultipleksirao) segment na odgovarajući soket. Konkretno, a za razliku od



**Slika 3.4** ♦ Inverzija izvornog i odredišnog broja porta

UDP-a, dva dolazna TCP segmenta sa različitim IP adresama izvora, ili različitim izvornim brojevima portova, usmeriće se na dva različita soketa. Da bismo dobili još bolji uvid, razmotrimo ponovo programski primer sa TCP klijentom/serverom iz odeljka 2.6:

- ♦ Aplikacija TCP servera ima „soket dobrodošlice“ koji čeka zahteve za uspostavljanje konekcije koji stižu od TCP klijenata (slika 2.22) na broju porta 6789.
- ♦ TCP klijent proizvodi segment za uspostavljanje konekcije naredbom:

```
Socket clientSocket = new Socket {"serverHostName", 6789};
```

- ♦ Zahtev za uspostavljanje konekcije nije ništa drugo nego TCP segment sa brojem odredišnog porta 6789 u uključenim posebnim bitom za uspostavljanje konekcije u TCP zaglavju (opisan u odeljku 3.5). Segment takođe sadrži izvorni broj porta koji je izabran klijent. Prethodna naredba takođe proizvodi TCP soket za klijentski proces, kroz koji podaci mogu da ulaze u ovaj proces i da ga napuštaju.
- ♦ Kada server primi od klijenta dolazni segment sa zahtevom za konekciju, on o tome obaveštava serverski proces koji zatim pravi soket konekcije:

Socket connectionSocket = welcomeSocket.accept();

- ♦ Osim toga, server beleži sledeće četiri vrednosti iz segmenta za zahtevanje konekcije: (1) broj izvornog porta iz segmenta, (2) IP adresu izvornog računara, (3) odredišni broj porta iz segmenta i (4) vlastitu IP adresu. Novonapravljeni soket konekcije identifikuje se pomoću te Četiri vrednosti; svi sledeći dolazni segmenti Čiji se izvomi port, izvorna IP adresa, odredišni port i odredišna IP adresa poklapaju sa ove četiri vrednosti, biće demultiplesirani u ovom soketu. Postoje sada uspostavljena TCP konekcija, klijent i server mogu da šalju podatke jedan drugome.

Serverski računar može da podrži više istovremenih TCP soketa tako da svaki soket bude pridružen nekom procesu, a da se svaki soket identificuje vlastitom 4-torkom. Kada TCP segment stigne u računar, koriste se sva četiri polja (izvorna IP adresa, izvorni port, odredišna IP adresa, odredišni port) da bi se segment usmerio (demultiplesirao) u odgovarajući soket.

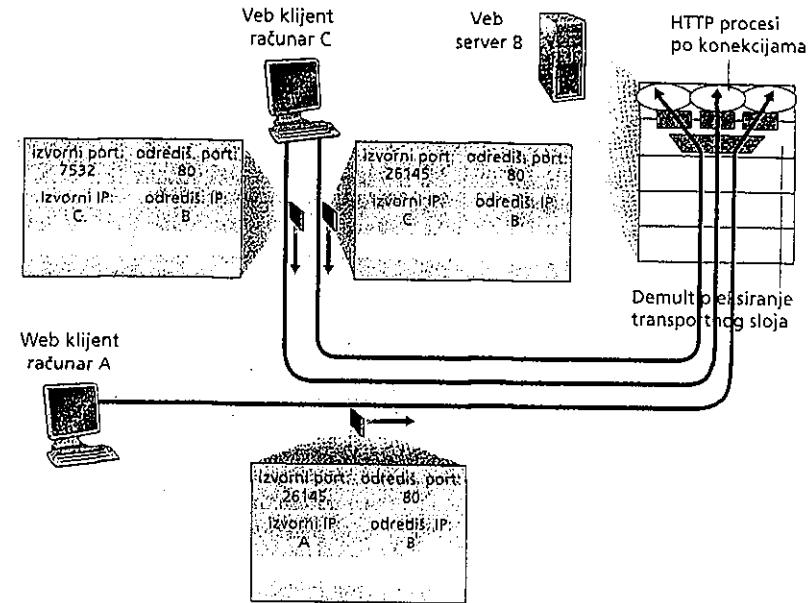
Ova situacija ilustrovana je na slici 3.5, na kojoj računar C inicira dve HTTP sesije na serveru B, a računar A inicira jednu HTTP sesiju na serveru B. Računari A i C i server B imaju svi vlastite IP adrese - A, C, odnosno B.

Računar C u svojim dvema HTTP konekcijama dodeljuje dva različita broja (26145 i 7S32) za izvorne portove. Pošto računar A bira brojeve izvornih portova nezavisno od računara C, on bi takođe mogao da dodeli svojoj HTTP konekciji izvomi port 26145. Bez obzira na to, server B će ipak moći pravilno da demultiplesira dve konekcije sa istim izvornim brojem porta pošto te konekcije imaju različite izvorne IP adrese.

#### Veb serveri i TCP

Pre zaključka treba reći još nešto o veb serverima i načinu na koji oni koriste brojeve portova. Uzmimo računar na kome se izvršava veb server, na primer Apache veb server na portu 80. Kada klijenti (na primer, Čitači) šalju serveru segmente, svi segmenti će imati odredišni port 80. Konkretno, i segment za početno uspostavljanje konekcije i segmenti koji sadrže poruke sa HTTP zahtevima imajuće odredišni port 80. Kao što smo upravo rekli, server razlikuje segmente od različitih klijenata po izvornim IP adresama i izvornim brojevima portova.

Veb serveri obično proizvode novi proces ili prave novu nit za svaku novu klijentsku konekciju. Na slici 3.5 prikazanje veb server koji pravi novi proces za svaku konekciju. Kao što se vidi na slici 3.5, svaki od ovih procesa ima vlastiti soket kroz koji stižu HTTP zahtevi i šalju se HTTP odgovori. Napominjemo, međutim, da ne postoji uvek odnos 1:1 između soketa i procesa. U stvari, današnji veb serveri visokih performansi često koriste šarao jedan proces, ali prave novu nit sa novim soketom za svaku novu klijentsku konekciju, (Nit se može smatrati potprocesom „lake



**Slika 3.5** ♦ Dva klijenta koriste isti broj odredišnog porta (80) za komunikaciju sa istom aplikacijom na veb serveru.

kategorije".) Ako ste uradili prvi programski zadatak u poglavljiju 2, napravili ste veb server koji upravo to radi. Za takav server, u jednom trenutku može da postoji mnogo soketa (sa različitim identifikatorima) za isti proces.

Ako klijent i server koriste postojani HTTP, tada će tokom cele postojane konekcije klijent i server razmenjivati HTTP poruke **kr02** isti serverski soket. Međutim, ako klijent i server koriste nepostojani HTTP, tada se za svaki zahtev i odgovor pravi i zatvara nova TCP konekcija, i zato se za svaki zahtev i odgovor pravi i kasnije zatvara novi soket. To često pravljenje i zatvaranje soketa može veoma *loše* da utiče na performanse opterećenog veb servera (mada se za izbegavanje ovog problema može koristiti niz trikova sa operativnim sistemom). Čitaoci koje zanimaju pitanja operativnog sistema u vezi sa postojanim i nepostojanim HTTP-om trebalo bi da pročitaju [Nielsen 1997].

Pošto smo opisali multipleksiranje i demultipleksiranje transportnog sloja, prelazimo na opis jednog od internetskih transportnih protokola, UDP. U sledećem odeljku videćemo da UDP dodaje gotovo jedino uslugu multipleksiranja i demultipleksiranja protokolu mrežnog sloja.

### 3.3 Prenos bez uspostavljanja konekcije; UDP

U ovom odeljku detaljnije razmatramo UDP, kako funkcioniše i šta radi. Bilo bi poželjno da Čitalac ponovo prouči odeljak 2. i gde se nalazi pregled modela usluge UDP-a, i odeljak 2.7 gde je opisano programiranje soketa preko UDP-a.

Da **bi** opis UDP-a bio konkretniji, prepostavimo da želimo da projektujemo gol transportni protokol, bez ikakvih dodataka. Kako bismo to uradili? Prvo razmotrimo korišćenje praznog transportnog protokola. Konkretno, na strani pošiljaoca poruke **bi** se uzimale od aplikacijskog procesa i predavale direktno mrežnom sloju, a na primajućoj strani **bi** se poruke dobijene od mrežnog sloja direktno predavale aplikacijskom procesu. Međutim, kao što smo videli u prethodnom odeljku, mora se uraditi nešto više od toga! Najmanje što transportni sloj mora da pruži jesu multipleksiranje i demultipleksiranje, da bi podatke dobijene iz mrežnog sloja preneo ispravnom procesu u aplikacijskom sloju,

UDP, definisan u RFC-u 768, izvršava minimum koji se očekuje od transportnog protokola. Osim funkcije multipleksiranja i demultipleksiranja i veoma ograničene provere grešaka, on ništa ne dodaje IP-u. U suštini, ako programer izabere UDP umesto TCP-a, aplikacija će se skoro direktno obraćati IP-u. UDP uzima poruke od aplikacijskog procesa, dodaje polja sa brojevima izvornog i odredišnog porta za multipleksiranje i demultiplesksiranje, dodaje još dva mala polja i predaje dobijeni segment mrežnom sloju. Mrežni sloj enkapsulira segmentu IP datagram i zatim na najbolji mogući način nastoji da isporuči segment prijemnom računaru. Ako segment stigne do prijemnog računara, UDP koristi broj odredišnog porta kako **bi** isporučio podatke iz segmenta ispravnom aplikacijskom procesu. Napominjemo da u UDP-u nema međusobne sinhronizacije između entiteta otpremnog i prijemnog transportnog sloja pre slanja segmenta. Zato se za UDP kaže da je on *bez uspostavljenje konekcije*.

DNS je primer protokola aplikacijskog sloja koji obično koristi UDP. Kada neka DNS aplikacija u računaru želi da postavi upit, ona pravi DNS upit i uručuje ga UDP-u. Bez prethodnog sinhronizovanja sa UDP entitetom koji se izvršava u odredišnom sistemu, UDP dodaje upitu polja zaglavljiva i predaje dobijeni segment mrežnom sloju. Mrežni sloj enkapsulira UDP segment u datagram i šalje ga serveru za imena. Zatim DNS aplikacija računara, koji je postavio upit, čeka odgovor na upit. Ako ne dobije odgovor (možda zato što je mreža izgubila upit ili odgovor), aplikacija pokušava da pošalje upit drugom serveru za imena ili obaveštava pozivnu aplikaciju da ne može dobiti odgovor.

Možda vas čudi zašto bi neki programer ikada gradio aplikaciju na UDP-u, a ne na TCP-u. Zar nije uvek bolje izabrati TCP, kad TCP obezbeđuje uslugu pouzdanog transfera podataka. Što UDP ne omogućava? Odgovor je da za mnoge aplikacije UDP više odgovara iz sledećih razloga:

- ◆ *Bolja kontrola na nivou aplikacije nad sadržajem i vremenom slanja.* Ako se koristi UDP, čim aplikacijski proces preda podatke UDP-u, UDP će spakovati podatke u UDP segment i odmah predati segment mrežnom sloju. Nasuprot tome, TCP sadrži mehanizam za kontrolu zagušenja koji zadržava TCP slanje u transportnom sloju, kada linkovi između izvorišnog i odredišnog računara postanu preterano zagušeni. TCP će takođe više puta da šalje isti segment podataka dokle god od odredišta ne primi potvrdu daje segment primljen, bez obzira na to koliko takva pouzdana isporuka traje. Pošto aplikacije u realnom vremenu često zahtevaju garantovanu frekvenciju slanja i ne žele preterano da odlažu prenošenje segmenta, a mogu da tolerišu mali gubitak podataka, model TCP usluge nije naročito pogodan za potrebe takvih aplikacija. Kao što ćemo objasniti, ove aplikacije mogu da koriste UDP i da svu dodatnu funkcionalnost koja im je potrebna povrh UDP-ove osnovne usluge (isporeuke segmenata od jednog procesa drugom), implementiraju kao deo aplikacije.
- ◆ *Nema uspostavljanja konekcije.* Kao što ćemo objasniti kasnije, TCP koristi sin-hronizovanje u tri koraka pre nego što počne da šalje podatke, UDP jednostavno kreće bez ikakvog formalnog uvoda. Zato kod UDP-a nema kašnjenja prilikom uspostavljanja konekcije. To je verovatno glavni razlog zbog kojeg se DNS izvršava preko UDP-a, a ne preko TCP-a. DNS bi inače bio mnogo sporiji. HTTP koristi TCP pošto je pouzdanost značajna za veb stranice sa tekstom. Ali, kako smo ukratko napomenuli u odeljku 2.2, odlaganje zbog uspostavljanja TCP konekcije za HTTP značajno doprinosi „sveukupnom čekanju na svetskoj mreži“.
- ◆ *Nema stanje konekcije.* TCP održava stanje konekcije u krajnjim sistemima. Ovo stanje konekcije obuhvata privremene memorije za primanje i slanje, parametre za kontrolu zagušenja i parametre za redne brojeve i brojeve potvrda. Videćemo u odeljku 3.5 da su ove informacije o stanju potrebne za primenu pouzdanog transfera podataka u TCP-i i za obezbeđenje kontrole zagušenja. Nasuprot tome, UDP ne prati stanje konekcije, a ni sve ove ostale parametre. Zato server namenjen određenoj aplikaciji može obično da podrži mnogo više aktivnih klijentata ako se aplikacija izvršava preko UDP-a, a ne preko TCP-a.
- ◆ *Manje dodatno opterećenje zaglavljem paketa.* TCP segment ima zaglavje od 20 bajtova, dok UDP ima samo 8.

Na slici 3.6 navedene su popularne Internet aplikacije i transportni protokoli koje one koriste. Kao što bismo i očekivali, e-pošta, pristup udaljenim terminalima, Web i transfer datoteka izvršavaju se preko TCP-a. Svim tim aplikacijama potrebna je usluga pouzdanog transfera podataka, koju on i obezbeđuje. Ipak, mnoge važne aplikacije se izvršavaju preko UDP-a, a ne preko TCP-a. UDP se koristi za ažuriranje RIP tabele rutiranja (pročitajte poglavje 4 o mrežnom sloju) pošto se ažuriranja šalju povremeno (obično svakih 5 minuta), pa se izgubljena ažuriranja zamjenjuju svežijim ažuriranjem. UDP se takođe koristi za prenos podataka za upravljanje mrežom (SNMP, poglavje 8). U ovom slučaju, UDP je bolji, jer aplikacije za upra-

vijanje mrežom. Često moraju da se izvršavaju kada je mreža u napregnutom stanju - upravo kada je teško postići pouzdan transfer podataka kontrolu zagrušenja. Osim toga, kako smo već pomenuli, DNS se izvršava preko UDP-a i tako izbegava kašnjenja za uspostavljanje TCP konekcije.

Kao što se vidi na slici 3.6, UDP se takođe danas često koristi za multimedijalne aplikacije, kao što su telefoniranje preko Interneta, video konferencije u realnom vremenu i protok audio i video datoteka. Ove aplikacije ćemo bliže razmotriti u poglavljiju 6. Sada samo napominjemo da sve te aplikacije mogu da tolerišu male gubitke paketa, pa pouzdani transfer podataka nije apsolutno neophodan za uspeh aplikacija. Štaviše, aplikacije u realnom vremenu, kao što su Internet telefoniranje i video konferencije, veoma loše podnose TCP-ovu kontrolu zagrušenja. Zato programeri multimedijalnih aplikacija često odlučuju da im se aplikacije izvršavaju preko UDP-a, a ne preko TCP-a.

Mada se danas često koristi, izvršavanje multimedijalnih aplikacija preko UDP-a predstavlja izvesnu kontroverzu. Kao što smo već pomenuli, UDP nema nikakvu kontrolu zagrušenja. Međutim, bez kontrole zagrušenja mreža bi došla u stanje u kojem se postiže sve manje korisnog rada. Kada bi svih pokrenuli protok video signala u realnom vremenu, koji ima veliku brzinu protoka, bez ikakve kontrole zagrušenja, na ruterima bi bilo toliko mnogo paketa da niko ne bi ništa mogao da

| Aplikacija                  | Protokol u aplikacijskom sloju | Transportni protokol |
|-----------------------------|--------------------------------|----------------------|
| Elektronska pošta           | SMTP                           | TCP                  |
| Pristup udaljenom terminalu | Telnet                         | TCP                  |
| Web                         | HTTP                           | TCP                  |
| Transfer datoteke           | FTP                            | TCP                  |
| Udaljeni server datoteke    | NFS                            | Obično UDP           |
| Protok multimedije          | vlastiti                       | Obična UDP           |
| Internet telefonija         | vlastiti                       | Obično UDP           |
| Upravljanje mrežom          | SNMP                           | Obično UDP           |
| Protokoli rutiranja         | RIP                            | Obično UDP           |
| Prevođenje imena            | DNS                            | Obično UDP           |

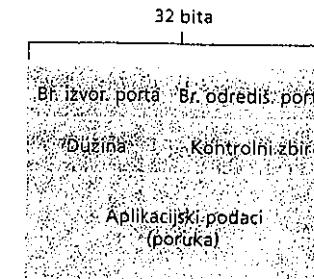
Slika 3.6 ♦ Popularne Internet aplikacije i transportni protokoli na kojima su zasnovane.

vidi. Prema tome, nedostatak kontrole zagrušenja u UDP-u predstavlja potencijalno veliki problem [Floyd 1999]. Mnogi istraživači predlažu nove mehanizme kojima bi se svim izvorima, pa i izvorima UDP-a, nametnula prilagodljiva kontrola zagrušenja [Mahdavi 1997; Floyd 2000].

Pre nego što predemo na opis strukture UDP segmenta, napominjemo da *jesti* moguće da aplikacija ima pouzdan transfer podataka, i ako koristi UDP. To se može postići ako se pouzdanost ugraditi u samu aplikaciju (na primer, dodavanjem mehanizama za potvrđivanje i ponovno slanje, koje ćemo videti u sledećem odeljku). Ali, ovo nije trivijalan zadatok i zadao bi programeru mnogo dodatnog posla. Ipak, ugradivanje pouzdanosti direktno u aplikaciju omogućava da ona zadrži „i jare i pare“. To jest, aplikacijski procesi mogu da ostvare pouzdanu komunikaciju ne podležući ograničenjima brzine slanja koje nameće TCP-ov mehanizam za kontrolu zagrušenja. Mnoge današnje aplikacije za protok signala u realnom vremenu upravo to i rade - izvršavaju se preko UDP-a, alt da bi se smanjio gubitak paketa, ugrađuju potvrdu prijema i ponavljanje slanja u samu aplikaciju; videti primer u [Rhee 1998].

### 3.3.1 Struktura TJDP segmenta

Struktura UDP segmenta, prikazana na slici 3.7, defmisana je u RFC-u 768. Aplikacijski podaci nalaze se u polju podataka UDP segmenta. Na primer, za DNS, polje podataka sadrži poruku upita ili poruku odgovora. Kod aplikacija sa protokom audio signala u realnom vremenu, polje podataka popunjava se audio uzorcima. UDP zaglavljive ima samo četiri polja od kojih svako zauzima dva bajta. Kao stoje rečeno u prethodnom odeljku, brojevi portova omogućavaju određenom računaru da prosledi aplikacijske podatke odgovarajućem procesu koji se izvršava na određenom krajinjem sistemu (tj. da obavi demultiplesksiranje). Kontrolni zbir služi prijemnom računari da proveri da li je u segmentu došlo do grešaka. U suštini, kontrolni zbir se izračunava i nad nekoliko polja IP zaglavljiva, a ne samo nad UDP segmentom. Ovaj



Slika 3.7 ♦ Struktura UDP segmenta

detalj ćemo zanemariti da nam ne bi drvo zaklonilo Sumu. Sada ćemo opisati izračunavanje kontrolnog zbir-a. Osnovni principi otkrivanja greške opisuju se u odeljku 5.1. Polje dužine navodi dužinu UDP segmenta u bajtovima, uključujući i zaglavlj-e.

### 3.3.2 UDP kontrolni zbir

UDP kontrolni zbir služi za otkrivanje greške. To jest, kontrolni zbir se koristi da bi se utvrdio da li su promenjeni bitovi u UDP segmentu (na primer, zbog šuma u lin-kovima ili čuvanja u ruteru), prilikom prenosa od izvora do odredišta. UDP na strani pošiljaoca izračunava komplement jedinice za sumu svih 16-bitnih reči u segmentu, gde se odbacuje svako prekoračenje do kojeg dođe prilikom sabiranja. Ovaj rezultat se stavlja u polje kontrolnog zbir-a UDP segmenta. Ovde dajemo jednostavan primer izračunavanja kontrolnog zbir-a. Detalje o efikasnom implementiraju izračunavanja možete naći u RFC-u 1071, a o performansama kada se radi sa stvarnim podacima u [Stone 1998; Stone 2000]. Kao primer, uzimimo da imamo sledeće tri 16-bitne reči;

```
0110011001100110
0101010101010101
000011100001111
```

Zbir prve dve od ovih 16-bitnih reči je:

```
0110011001100110
01010101010101
1011101110111011
```

Ako na ovu sumu dodamo treću reč, dobijamo:

```
1011101110111011
000011100001111
1100101011001010
```

Komplement jedinice dobija se pretvaranjem svih 0 u 1, a svih 1 u 0. Prema tome, komplement jedinice za sumu 1100101011001010 je: 0011010100110101 što se uzima kao kontrolni zbir. Na prijemnom kraju, sabiraju se sve četiri 16-bitne reči, uključujući i kontrolni zbir. Ako u paketu ne postoje greške, jasno je da će zbir kod primaoca biti 1UIII1111111UI. Ako je neki od bitova jednak 0, znamo da je u paketu došlo do greške.

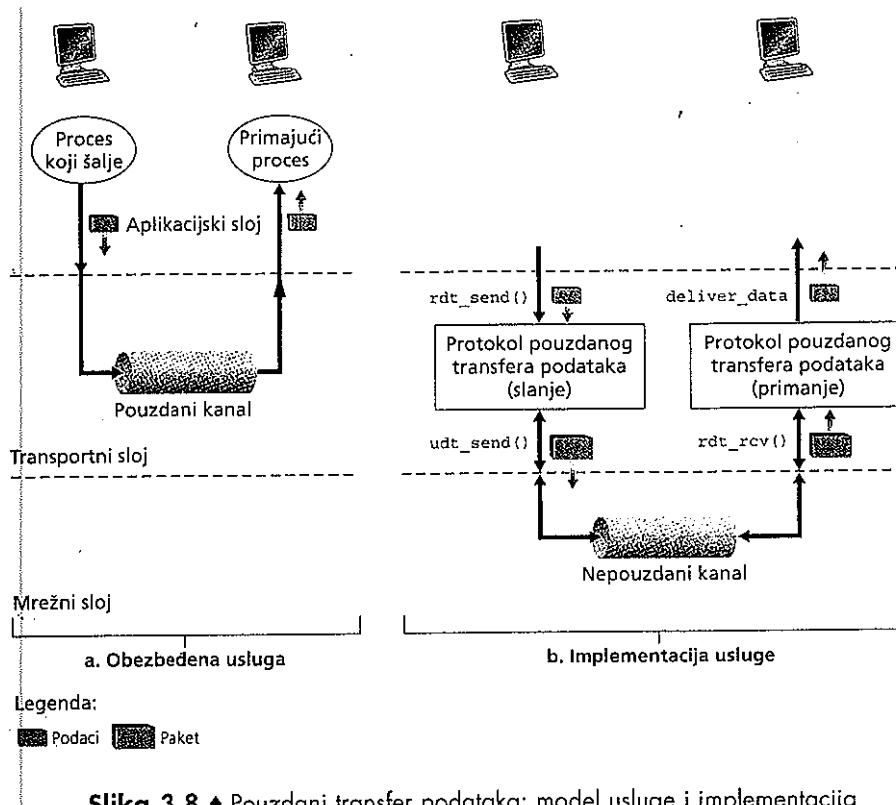
Možda vas čudi Što UDP uopšte koristi kontrolni zbir kada mnogi protokoli sloja veze sa podacima (uključujući i popularni protokol Ethernet) takođe sadrže proveru grešaka. Razlog leži u tome što ne postoji garancija da će svi linkovi od izvora do odredišta obezbediti proveru grešaka - neki od linkova možda koristi protokol koji ne sadrži proveru grešaka. Pošto se smatra da IP može da se izvršava povrh ma kog protokola sloja 2, korisno je da transportni sloj obezbedi proveru grešaka za svaki slučaj. Mada UDP sadrži proveru grešaka, on ništa ne preduzima da bi se greška ispravila. U nekim implementacijama UDP-a, oštećeni segment se jednostavno odbacuje; dok druge implementacije prosleđuju aplikaciji oštećeni segment uz upozorenje.

Ovim zaključujemo opis protokola UDP. Uskoro ćemo videti da TCP nudi svojim aplikacijama pouzdan transfer podataka kao i neke druge usluge koje UDP ne sadrži. Naravno, TCP je zato složeniji od UDP-a. Pre nego što predemo na opis protokola TCP, biće korisno da se vratimo za jedan korak i prvo opišemo osnovne principe pouzdanog transfera podataka pa ćemo to uraditi u sledećem odeljku. Nakon toga, istražićemp TCP u odeljku 3.5, pa ćemo videti da je TCP zasnovan na ovim osnovnim principima.

## 3.4 Principi pouzdanog transfera podataka

U ovom odeljku, razmotrićemo problem pouzdanog transfera podataka u opštem smislu. To je zato Što se problem implementirani a pouzdanog transfera podataka ne javlja samo u transportnom sloju već takođe i u sloju veze i u aplikacionom sloju. Taj problem je prema tome od ključnog značaja za umrežavanje. Zaista, kada bi trebalo napraviti spisak „prvih 10“ fundamentalno značajnih problema za celo umrežavanje, ovo bi bio kandidat za prvo mesto. U sledećem odeljku istražićemo TCP i konkretno pokazati da TCP primenjuje mnoge principe koje ćemo sada opisati.

Na slici 3.8 prikazanje radni okvir za našu studiju pouzdanog transfera podataka. Apstrakcija usluge u entitetima gornjeg sloja je pouzdani kanal kroz koji se mogu prenositi podaci. Ako postoji pouzdani kanal, ne dolazi do oštećenja prenetih bitova podataka (pretvorenih iz 0 u 1 ili obmuto) niti do gubitaka, i sve se isporučuje redosledom kojim je poslat. To je upravo model usluge koju TCP pruža Internet aplikacijama koje ga pozivaju.



Slika 3.8 ♦ Pouzdani transfer podataka: model usluge i implementacija

Protokol pouzdanog transfera podataka ima zadatak da implementira tu apstrakciju usluge. Zadatak je otežan činjenicom da sloj *ispod* protokola pouzdanog transfera podataka može da bude nepouzdan. Na primer, TCP je protokol pouzdanog transfera podataka koji se implementira povrh nepouzdanog (IP) mrežnog sloja s kraja nakraj. U stvari, sloj ispod dve krajnje tačke koje pouzданo komuniciraju može da se sastoji od jednog jedinog fizičkog linka (kao u slučaju protokola za prenos podataka u sloju veze), ili od globalne medumreže (kao u slučaju protokola transportnog sloja). Za sada je, međutim, dovoljno da taj sledeći niži sloj posmatramo jednostavno kao nepouzdan kanal od tačke do tačke.

U ovom odeljku postepeno ćemo razvijati otpremnu i prijemnu stranu pouzdanog protokola za transfer podataka, uzimajući u obzir sve složenije modele kanala koji se nalazi u sloju ispod. Na slici 3.8(b) prikazani su interfejsi našeg protokola za transfer podataka. Otpremna strana protokola za transfer podataka aktiviraće se odozgo pozivom `rdt_send()`. Podaci koji treba da se isporuče biće predati gornjem

sloju na prijemnoj strani. (Ovde *rdt* znači reliable data transfer ~ pouzdan transfer podataka, a `_send` znači da se poziva deo za slanje protokola *rdt*. U razvoju svakog protokola, prvi korak je biranje dobrog imenat.) Na prijemnoj strani, pozvaće se `rdt_rcv()` kada paket stigne od prijemne strane kanala. Kada protokol *rdt* želi da isporuči podatke gornjem sloju, to će uraditi pozivom `deliver_data()`. U sledećem tekstu ćemo umesto izraza segment za protokolsku jedinicu podataka koristiti izraz „paket“. Pošto se teorija koju razvijamo u ovom odeljku odnosi uopšteno na sve računarske mreže, a ne samo na transportni sloj Interneta, primereniji je opšći izraz „paket“.

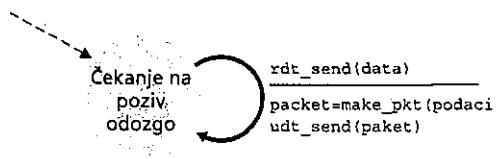
U ovom odeljku, razmatraćemo samo slučaj jednosmernog transfera podataka, tj. transfera podataka od otpremne do prijemne strane. Slučaj pouzdanog dvosmer-nog (puni dupleks) transfera podataka u suštini nije ništa teži, ali gaje znatno teže opisati. Mada razmatramo samo jednosmerni transfer podataka, važno je napomenuti da će otpremna i prijemna strana našeg protokola ipak morati da prenose pakete u *oba* smera, kao sto je naznačeno na slici 3.8. Uskoro ćemo videti da osim razmene paketa koji sadrže podatke koje treba preneti, otpremna i prijemna strana protokola *rdt* moraju takođe da razmenjuju i kontrolne pakete u *oba* smera. Obe strane protokola šalju pakete drugoj strani pozivanjem `udt_send()` (gde *udt* znači *unreliable data transfer* - nepouzdan transfer podataka).

### 3.4.1 Pravljenje protokola za pouzdan transfer podataka

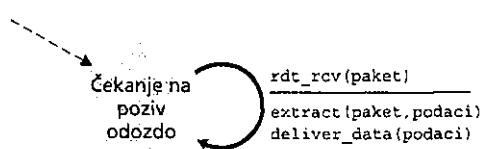
Sada ćemo razmotriti niz protokola, svaki složeniji od prethodnog, dok ne stignemo do besprekomo pouzdanog protokola za transfer podataka.

Pouzdan transfer podataka preko savršeno pouzdanog kanala: *rdtl.O*

Prvo razmotrimo najjednostavniji slučaj, kada je kanal u osnovi potpuno pouzdan. Sam protokol, koji ćemo nazvati *rdtl.0*, je trivijalan. Na slici 3.9 prikazane su definicije **konačnog automata** (*finite-state machine*, FSM) za otpremni i prijemni *rdtl.0*. *FSM* na slici 3.9(a) definije rad pošiljaoca, dok *FSM* 3.9(b) definije rad primaoca. Važno je primetiti da za primaoca i pošiljaoca postoje *zasebni* *FSM*-ovi. *FSM*-ovi pošiljaoca i primaoca na slici 3.9 imaju samo po jedno stanje. Strelice u *FSM* opisu označavaju prelaz protokola iz jednog stanja u drugo. (Pošto svaki *FSM* na slici 3.9 ima samo po jedno stanje, imamo prelaz iz jednog stanja nazad u to isto stanje; uskoro ćemo videti i složenije dijagrame stanja.) Dogadaj koji izaziva prelaz prikazan je iznad horizontalne linije u oznaci prelaza, a ispod horizontalne linije prikazana je aktivnost koja se preduzima kada dogadaj nastupi. Kada se povodom nekog dogadaja ne preduzima nikakva aktivnost, ili ako se aktivnost preduzima a dogadaj nije nastupio, koristiće se simbol A ispod, odnosno iznad, horizontalne linije da bi se eksplicitno naglasilo da nema aktivnosti ili dogadaja. Početno stanje *FSM*-a označeno je isprekidanim strelicom. Mada *FSM*-ovi na slici 3.9 imaju samo po jedno stanje, uskoro ćemo prikazati *FSM*-ove sa više stanja, pa će biti važno da se prepozna početno stanje svakog od njih.



a. rdt1.0: otpremna strana



b. rdt1.0: prijemna strana

**Slika 3.9** ♦ Protokol rdt1.0 za potpuno pouzdan kanal

Otpremna strana protokola rdt jednostavno prihvata podatke iz gornjeg sloja pomoću događaja, pravi paket koji sadrži te podatke (upotrebom funkcije make\_pkt (podaci)) i šalje paket u kanal. U praksi, događaj rdt\_send (podaci) bi potekao od poziva procedure (na primer, rdt\_send ()) u aplikaciji iz gornjeg sloja.

Na prijemnoj strani, rdt prima paket iz donjeg sloja putem događaja rdt\_rcv (paket), vadi podatke iz paketa (upotrebom funkcije extract (paket, podaci)) i predaje podatke gornjem sloju (pomoću funkcije deliver\_data (podaci)). U praksi, događaj rdt\_rcv (paket) bi potekao od poziva procedure (na primer, rdt\_rcv ()) iz protokola nižeg sloja.

U ovom jednostavnom protokolu, nema razlike između jedinice podataka i paketa. Osim toga, sav protokol paketa je od pošiljaoca prema primaocu; sa savršeno pouzdanim kanalom nema potrebe da prijemna strana šalje povratnu informaciju pošiljaocu pošto do greške ne može doći! Obratite pažnju na to da smo takođe prepostavili da je primalac u stanju da prihvata podatke istom brzinom kojom ih pošiljalac šalje. Prema tome, nema potrebe da primalac zahteva od pošiljaoca da „uspori”!

#### Pouzdan transfer podataka preko kanala sa greškama: rdt2 . 0

Realniji model kanala u osnovi bio bi kanal u kojem može doći do oštećenja bitova u paketu. Takve greške obično se javljaju u fizičkim komponentama mreže prilikom prenosa, umnožavanja ili privremenog memorisanja paketa. I dalje ćemo, za sada, pretpostaviti da su svi paketi primljeni onim redom kojim su i poslati (mada neki bitovi mogu biti oštećeni).

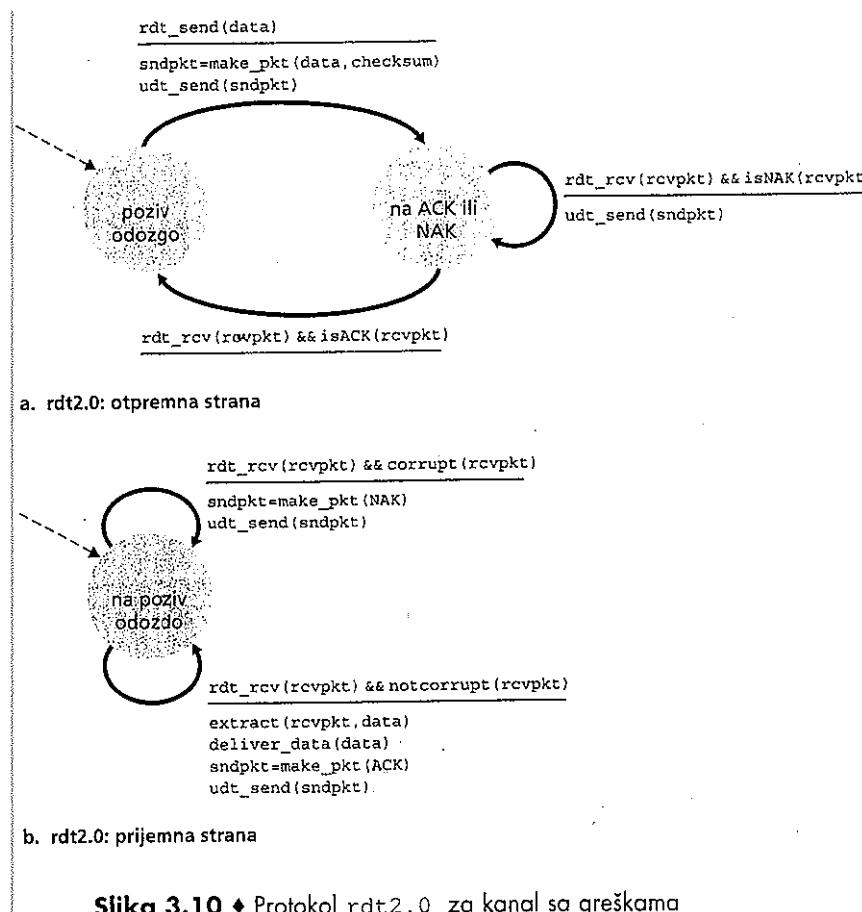
Pre nego što razvijemo protokol za pouzdanu komunikaciju preko takvog kanala, razmotrimo prvo kako bi ljudi postupili u takvoj situaciji. Zamislite da diktirate dugačku poruku preko telefona. U tipičnoj situaciji primalac poruke bi mogao da kaže „u redu“ nakon svake rečenice koju je čuo, razumeo i zapisao. Ako primalac čuje nerazgovetnu rečenicu, zatražiće daje ponovite. Ovaj protokol za diktiranje poruka sastoji se od **pozitivnih potvrda** („u redu“) i **negativnih potvrda** („molim, ponovite“). Te kontrolne poruke omogućavaju primaocu da saopšti pošiljaocu Staje primljeno pravilno, a staje primljeno nepravilno pa ga treba ponoviti. U računarskoj mreži, pouzdani protokoli za transfer podataka, zasnovani na takvom ponovnom slanju, poznati su kao **ARQ** (*Automatic Repeat reQuest*) protokoli.

U suštini, da bi se u ARQ protokolima uklonilo prisustvo grešaka zbog oštećenih bitova, potrebne su tri dodatne mogućnosti:

- ♦ *Otkrivanje grešaka.* Prvo, potreban je mehanizam kojim će primalac utvrditi da postoje greške. Setite se iz prethodnog odeljka da UDP koristi Internetovo polje kontrolnog zbiru baš u ovu svrhu. U poglavљу 5 detaljnije ćemo razmotriti tehnikе za otkrivanje i ispravljanje grešaka; te tehnikе omogućavaju primaocu da otkrije i eventualno ispravi greške u bitovima u paketu. Za sada, dovoljno je da znatno da je za te tehnikе potrebno od pošiljaoca poslati primaocu dodatne bitove (pored bitova podataka koje treba preneti); ti bitovi će se smestiti polje kontrolnog zbiru u paketu podataka protokola rdt 2 . 0.
- ♦ *Povratna informacija pošiljaocu.* Pošto se pošiljalac i primalac obično izvršavaju na različitim krajnjim sistemima i možda su razdvojeni hiljadama kilometara, jedini način da pošiljalac sazna kako primalac gleda na stvari (u ovom slučaju, da li je paket ispravno primljen) jeste da mu primalac pošalje eksplicitnu povratnu informaciju. Pozitivne (ACK) i negativne (NAK) potvrde iz malopredašnjeg primera (diktiranje poruke telefonom) su u stvari takve povratne informacije. Slično tome, naš protokol rdt2 . 0 će vraćati ACK i NAK pakete od primaoca ka pošiljaocu. U principu, ti paketi treba da sadrže samo jedan bit; na primer, vrednost 0 bi značila NAK, a vrednost 1 ACK.
- ♦ *Ponovno slanje.* Pošiljalac će ponovo poslati paket koji je kod primaoca primljen sa greškom.

Na slici 3.10 dat je FSM prikaz rdt 2 . 0 protokola za transfer podataka u kojem se koriste otkrivanje grešaka i pozitivne i negativne potvrde.

Otpremna strana protokola rdt 2 . 0 ima dva stanja. U stanju prikazanom na levoj strani, protokol za slanje čeka podatke iz gornjeg sloja, koje treba da pošalje dole. Kada nastupi događaj `rdt_send` (podaci), pošiljalac će napraviti paket (`sndpkt`) sa podacima koji se šalju i kontrolni zbir paketa (na primer, onako kako je u odeljku 3.3.2 opisano za slučaj UDP segmenta), a zatim će poslati taj paket operacijom `udt_send` (`sndpkt`). U stanju prikazanom na desnoj strani, protokol za slanje čeka od primaoca paket ACK ili NAK. Ako stigne paket ACK (na slici 3.10 ovom događaju odgovara notacija `rdt_recv(rcvpkt) && isACK(rcvpkt)`), pošiljalac zna da je poslednji preneti paket ispravno primljen, pa se protokol vraća u stanje čekanja podataka iz gornjeg sloja. Ako se primi NAK, protokol ponovo šalje



Slika 3.10 ♦ Protokol rdt2.0 za kanal sa greškama

poslednji paket i čeka da primalac vrati ACK ili NAK u odgovor na ponovljeno slanje paketa podataka. Važno je primetiti da kada je pošiljalac u stanju čekanja na ACK ili NAK, on ne može da primi još podataka za slanje iz gornjeg sloja; do toga će doći tek pošiljalac dobije ACK i napusti ovo stanje. Prema tome, pošiljalac neće poslati nove podatke dok se ne uveri daje primalac ispravno primio tekući paket. Zbog ovakvog ponašanja, protokoli kakav je rdt2 . 0, poznati su kao protokoli „stani i čekaj”.

FSM prijemne strane za rdt2 . 0 i dalje ima samo jedno stanje. Po prijemu paketa primalac odgovara sa ACK ili NAK, u zavisnosti od toga da lije primljeni paket oštećen ili ne. Na slici 3.10 iskaz `rdt_recv (rcvpkt) & & corru-pt (rcvpkt)` odgovara događaju kada se primi paket i utvrdi da on sadrži grešku.

Protokol rdt2 . 0 možda izgleda kao da funkcioniše ali, nažalost, on ima fatalni nedostatak. Konkretno, nismo uzeli u obzir mogućnost da ACK ili NAK paket može takođe da bude oštećen! (Pre nego što nastavimo, trebalo bi da razmislite o načinima da se ova greška ispravi.) Nažalost, ovaj mali previd nije tako bezopasan kao što možda izgleda. U najmanju ruku, moraćemo paketima ACK i NAK da dodamo kontrolni zbir da bismo otkrili takvu grešku. Mnogo je teže pitanje kako da se protokol oporavi od grešaka u ACK i NAK paketima. Problem je u tome što u slučaju greške u ACK ili NAK paketu, pošiljalac nema načina da sazna da li je primalac ispravno primio poslednji deo posleđenih podataka.

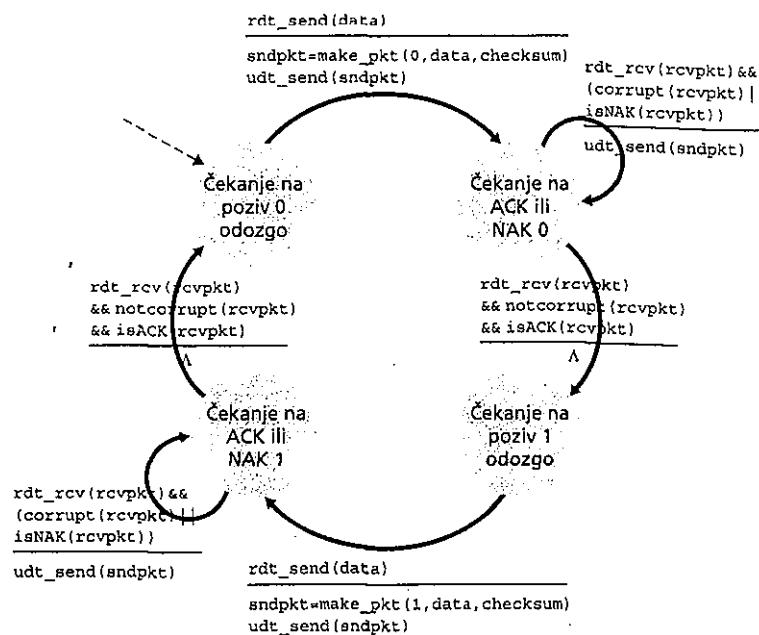
Pogledajmo tri mogućnosti za obradu neispravnih ACK ili NAK paketa:

- ♦ Kao prvo, da vidimo Šta bi čovek mogao da uradi u situaciji iz primera sa diktiranjem poruke. Ako govornik nije razumeo odgovor „u redu“ ili „molim ponovite“, on bi verovatno pitao „šta ste rekli?“ (i na taj način bi uveo u naš protokol novu vrstu paketa od pošiljaoca ka primaocu). Sagovorač bi zatim ponovio odgovor. Međutim, Šta ako se ošteti pitanje „šta ste rekli?“? Primalac, ne može da zna da li oštećeno pitanje spada u diktat ili je to zahtev da se ponovi poslednji odgovor, te bi verovatno rekao „šta ste vi rekli?“. Naravno, i taj odgovor može da bude oštećen. Jasno je, krenuli smo teškim putem.
- ♦ Druga alternativa je da se doda dovoljno bitova kontrolnog zbira tako da pošiljalac ne samo otkrije, već može i da ispravi bitske greške. Time se neposredno rešava problem za kanal u kojem može doći do oštećenja paketa, ali ne i do njihovog gubitka.
- ♦ Treći pristup je da pošiljalac jednostavno ponovo pošalje paket podataka ako primi oštećeni ACK ili NAK paket. Ovim pristupom se, međutim, u kanalu od pošiljaoca ka primaocu uvode duplikati paketa. Osnovni problem sa duplikatima paketa je u tome što primalac ne zna da lije poslati ACK ili NAK ispravno primljen kod pošiljaoca. Prema tome, on ne može *a priori* da zna da li paket koji stiže sadrži nove podatke ili ponovljene prethodne.

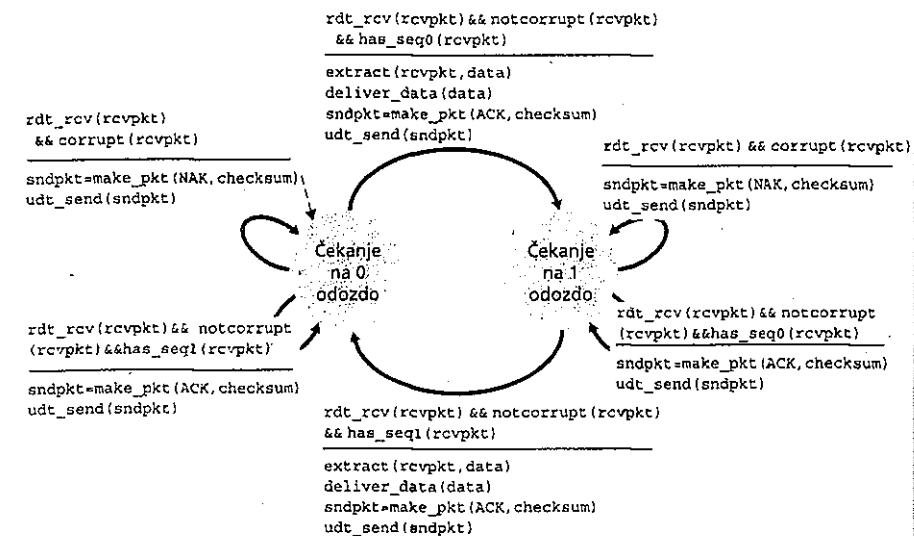
Jednostavno rešenje ovog novog problema (koje je prihvaćeno skoro u svim postojećim protokolima za transfer podataka, uključujući i TCP) jeste da se paketu

doda još jedno polje u kojem pošiljalac numeriše pakete tako što u ovo polje stavlja redni broj. Primalac zatim samo treba da proveri taj redni broj i utvrdi da li primljeni paket predstavlja ponavljanje. U ovom jednostavnom primeru protokola „stani i čekaj”, biće dovoljan redni broj od jednog bita, pošto će omogućiti primaocu da utvrdi da li pošiljalac ponovo šalje jednom već poslat paket (redni broj novopriznatog paketa je isti kao iredni broj poslednjeg paketa primljenog pre toga) ili je to nov paket (redni broj se promenio, nastavlja se aritmetikom modulo-2). Pošto trenutno koristimo pretpostavku da kanal ne gubi pakete, ACK i NAK paketi ne moraju sami da navode redni broj paketa čiji prijem potvrđuju. Pošiljalac zna daje primljeni ACK ili NAK paket (ispravan ili neispravan) nastao kao odgovor na poslednji predati paket podataka.

Na slikama 3.11 i 3.12 prikazanje FSM opis za rdt 2.1, našu fiksnu verziju protokola rdt2 . 0. U protokolu rdt2 . 1, FSM pošiljaoca i primaoca sada imaju dva puta više stanja nego ranije. Ovo je zato što stanje protokola sada mora da održava da li paket koji se trenutno šalje (iz pošiljaoca) ili očekuje (kod primaoca) treba da ima redni broj 1 ili 0. Primetićete da su aktivnosti u stanju kada se očekuje ili



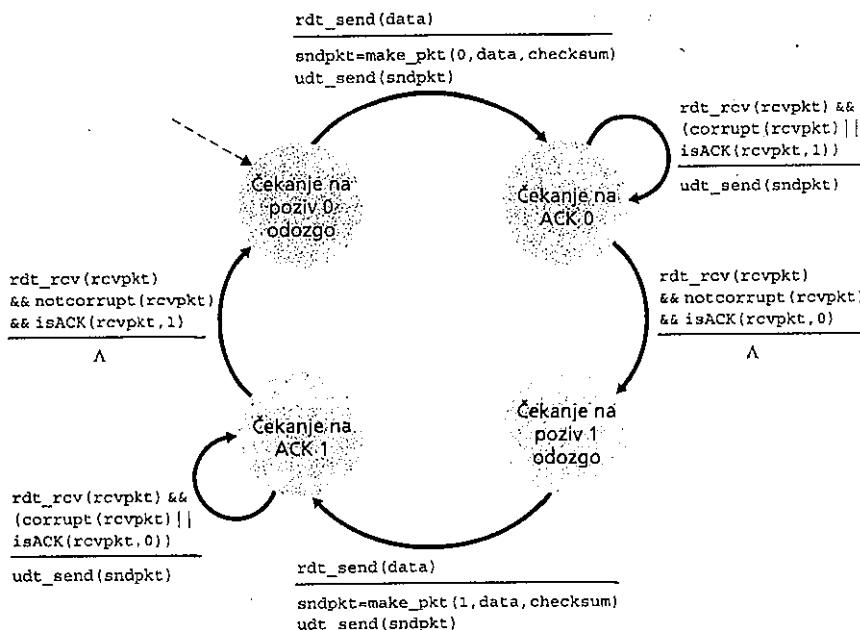
Slika 3.11 • Pošiljalac rdt2.1



Slika 3.12 • Primalac rdt2.1

šalje paket sa brojem 0, slika u ogledalu aktivnosti iz stanja kada se očekuje ili šalje paket sa brojem 1; jedina razlika se odnosi na rad sa rednim brojem.

Protokol rdt 2.1 koristi i pozitivne i negativne potvrde od primaoca ka pošiljaocu. Kada se primi paket izvan redosleda, pošiljalac šalje pozitivnu potvrdu za primljeni paket. Kada se primi oštećeni paket, pošiljalac šalje negativnu potvrdu prijema. Moguće je postići isti efekat ako se umesto NAK pošalje ACK za poslednji ispravno primljeni paket. Pošiljalac koji primi dva ACK-a za isti paket (tj. primi duplikat ACK-a) zna da primalac nije ispravno primio paket koji je usledio za paketom sa dve potvrde prijema. Ovakav pouzdani protokol za transfer podataka bez NAK-ova za kanal sa greškama u bitovima nazvali smo rdt2 . 2 i prikazali, ga na slikama 3.13 i 3.14. Suptilna razlika između protokola rdt2 . 1 i rdt2 . 2 jeste u tome što primalac sada mora da doda u ACK poruku i redni broj paketa čiji prijem potvrđuje (to se postiže uključivanjem argumenta ACK,0 ili ACK,1 u aktivnost make\_pkt () u FSM primaoca), a pošiljalac mora sada da proveri redni broj paketa čiji je prijem potvrđen primljenom ACK porukom (ovo se postiže dodavanjem argumenta 0 ili 1 u isACK {}) u FSM pošiljaoca).

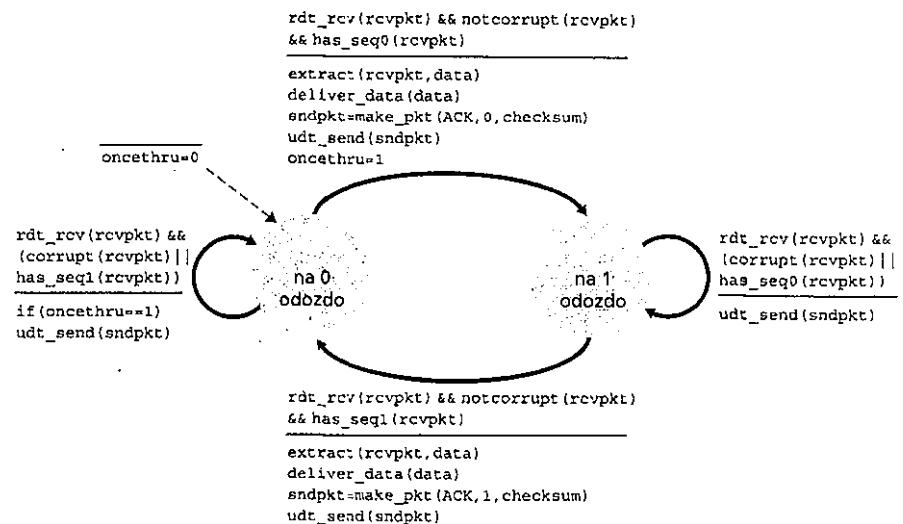


**Figure 3.13** ♦ Pošiljalac rdt2.2

Pouzdan transfer podataka preko kanala sa greškama u bitovima i gubicima paketa: rdt3 . 0

- Prepostavimo sada da, pored toga što oštećuje bitove, kanal kroz koji se vrši prenos podataka može i da *gubi* pakete, što u današnjim računarskim mrežama (uključujući i Internet) nije neuobičajeno. Sada u protokolu treba resiti još dva dodatna problema: kako otkriti gubitak paketa i šta uraditi u slučaju gubitka. Kontrolni zbirovi, redni brojevi, ACK paketi i ponovna slanja - tehnike razvijene za protokol rdt2.2
- omogućice nam da resimo ovo drugo pitanje. Rešavanje prvog pitanja zahtevaće da se doda jedan novi protokolski mehanizam.

Rešavanju gubitka paketa može se prići na više načina (u vežbama na kraju poglavlja videćemo ih još nekoliko). Ovdje ćemo za otkrivanje i rešavanje problema izgubljenih paketa zadužiti pošiljaoca. Prepostavimo da pošiljalac pošalje paket podataka i da se izgubi bilo sam paket, bilo primačevo ACK za taj paket. U oba slučaja pošiljalac neće od primaoca dobiti više nikakav odgovor. Ako pošiljalac dovoljno dugo sačeka da bi bio *siguran* da se paket izgubio, tada prosto može ponovo da pošalje isti paket podataka.



**Slika 3.14** ♦ Primalac rdt2.2

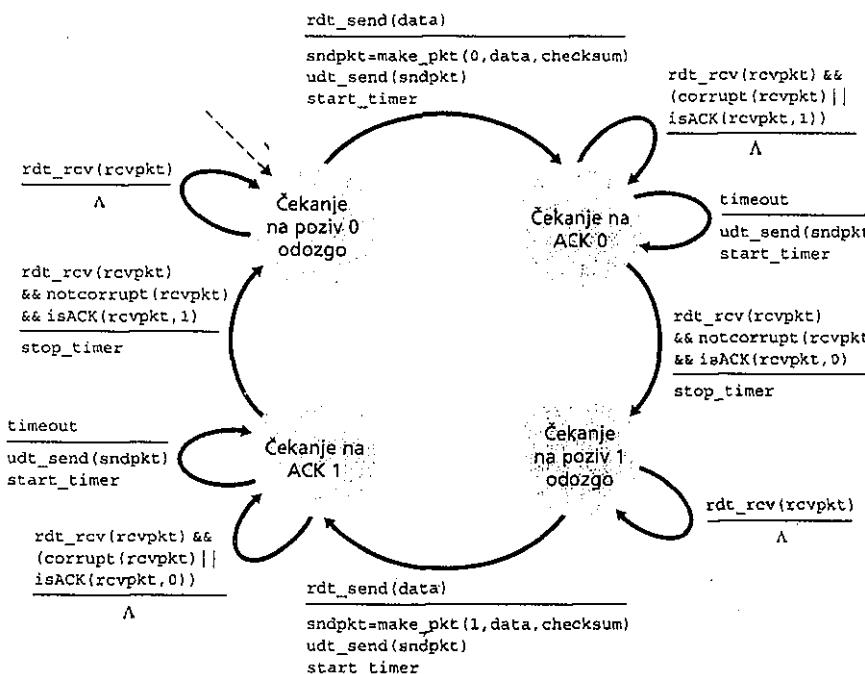
Koliko dugo treba pošiljalac da čeka da bi bio siguran da se nešto izgubilo? Jasno je da mora da čeka bar onoliko koliko traje put do pošiljaoca i nazad (gde može biti uključeno i privremeno čuvanje na usputnim ruterima), plus vreme potrebno za obradu paketa kod primaoca. U mnogim mrežama se maksimalno odlaganje u najgorem slučaju veoma teško procenjuje, a još teže precizno određuje. Štaviše, protokol bi u idealnom slučaju trebalo da se oporavi od gubitka paketa. Stope pre moguće; zato bi bilo preterano čekati sa početkom ispravljanja greške dok ne prode vreme potrebno u najgorem slučaju. U praksi je zato usvojen princip da pošiljalac izabere neko „razumno“ vreme u kojem je najverovatnije došlo do gubitka paketa, mada ne i stopostotno. Ako se ne primi ACK u tom vremenu, paket se šalje ponovo. Ako paket izuzetno mnogo kasni, pošiljalac će ga ponovo poslati iako nije izgubljen ni sim paket podataka, ni njegov ACK. Tako se u kanalu od pošiljaoca do primaoca javlja mogućnost dupliranih paketa podataka. Srećom, protokol rdt2.2 već sadrži funkcionalnost (redne brojeve) kojom se rešava slučaj dupliranih paketa.

Iz aspekta pošiljaoca, ponovno slanje je lek za sve. Pošiljalac ne zna da li je izgubljen paket podataka, ili je izgubljen ACK, ili jednostavno isuviše kasni paket ili ACK. U svim tim slučajevima rešenje je isto: ponovno slanje. Da bi se primenio mehanizam ponovnog slanja zasnovan na vremenu, biće potreban tajmer koji može da upozori pošiljaoca po isteku zadate količine vremena. Pošiljalac će morati da bude u stanju da (1) pokrene tajmer kad god šalje paket (bilo daje reč o prvom ili

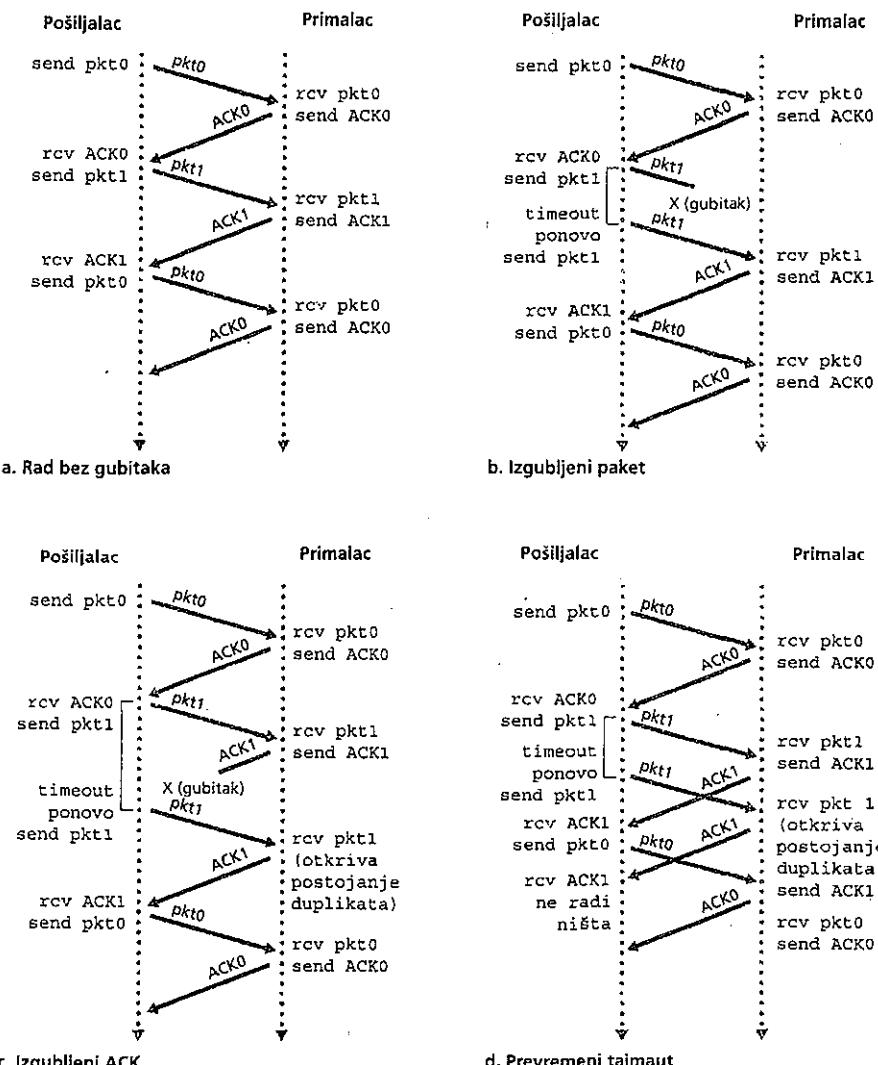
ponovljenom slanju), (2) da reaguje na upozorenje od strane tajmera (odgovarajućim aktivnostima) i (3) da zaustavi tajmer.

Postojanje duplih paketa koje pravi pošiljalac i gubitak paketa (podataka ili ACK) takođe usložnjava obradu ACK paketa koje prima pošiljalac. Ako stigne ACK, pitanje je kako će pošiljalac znati da li gaće primalac poslao kao odgovor na poslednji upućeni paket ili je to zakasneli ACK koji predstavlja odgovor na neko ranije slanje nekog drugog paketa? Rešenje ove dileme je da se ACK paket proširi poljem potvrde. Kada primalac pravi ACK, on će u ovo polje potvrde kopirati redni broj paketa podataka. Čiji prijem potvrđuje. Ispitivanjem sadržaja polja potvrde pošiljalac može da utvrdi redni broj paketa koji je pozitivno potvrđen.

Na slici 3.15 prikazanje FSM pošiljaoca za rdt3 . 0, protokol koji pouzdano prenosi podatke preko kanala koji može da ošteći i gubi pakete. Na slici 3.16 prikazano je kako protokol radi bez gubitka i kašnjenja paketa, tako postupa u slučaju gubitka paketa. Na slici 3.16 vreme se kreće od vrha dijagrama prema dnu; obratite pažnju na to da je vreme prijema paketa obavezno kasnije od vremena slanja zbog kašnjenja u prenosu i propagaciji. Na na slici 3.16, u ilustracijama odb. do d., uglaste



Slika 3.15 ♦ Pošiljalac rdt3.0



Slika 3.16 ♦ Funkcionisanje protokola rdt3.0 sa naizmeničnim bitovima

zgrade na strani pošiljaoca označavaju vreme na koje je postavljen tajmer i u kojem dolazi do tajm-auta. Malo suptilniji aspekti ovog protokola ispituju se u vežbama na kraju ovog poglavlja. Pošto redni brojevi paketa naizmenično imaju vrednost 0 i 1, protokol rdt3 . 0 se ponekad naziva protokol naizmeničnih bitova.

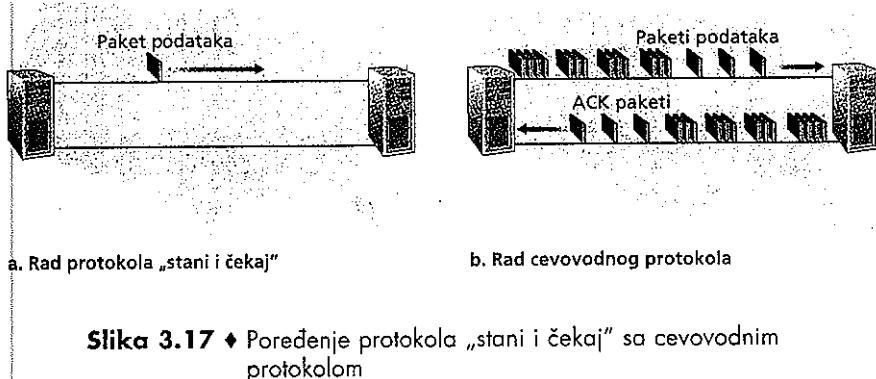
Sada smo sklopili ključne elemente protokola za transfer podataka. Kontrolni zbirovi, redni brojevi, tajmeri i pozitivne i negativne potvrde prijema paketa imaju ključnu i neophodnu ulogu u radu protokola. Dobili smo funkcionalan i pouzdan protokol za transfer podataka!

### 3.4.2 Cevovodni pouzdam protokoli za transfer podataka

Protokol rdt3 . 0 je funkcionalno ispravan protokol, ali malo je verovatno da bi iko bio zadovoljan njegovim performansama, pogotovo u današnjim mrežama velike brzine. Ključ problema performansi protokola rdt3 . 0 je Činjenica daje to protokol „stani i čekaj“.

Da biste shvatili do koje mere to stajanje i čekanje utiče na performanse, pogledajmo idealan slučaj sa dva krajnja računara od kojih se jedan nalazi na Zapadnoj obali SAD, a drugi na Istočnoj obali, kao sto je prikazano na slici 3.17. Vreme potrebno za propagaciju tamo i nazad brzinom svetlosti između ova dva krajnja sistema, RTT približno iznosi 30 milisekundi. Uzmimo da su povezani kanalom sa brzinom prenosa  $R$  od 1 Gb/s (IO<sup>9</sup> bitova u sekundi). Sa veličinom paketa  $L$ , od 1 000 bajtova (8 000 bitova) u svakom paketu, uključujući polja zaglavljiva i podatke, vreme potrebno da se paket zaista prenese linkom od 1 Gb/s je:

$$t_{trans} = \frac{L}{R} = \frac{8000 \text{ bitova/paket}}{10^9 \text{ bitova/paket}} = 8 \text{ mikrosekundi}$$



**Slika 3.17** ♦ Poređenje protokola „stani i čekaj“ sa cevovodnim protokolom

Na slici 3.18(a) prikazano je da sa našim protokolom „stani i čekaj“, ako pošiljalac počne slanje paketa u  $t = 0$ , tada će poslednji bit uči u kanalu na strani pošiljaoca za  $t - L/R = 8$  mikrosekundi. Paket zatim putuje kroz celu zemlju za 15 milisekundi, s tim što poslednji bit paketa stiže primaocu za  $t = RTT/2 + L/R = 15,008$  milisekundi.

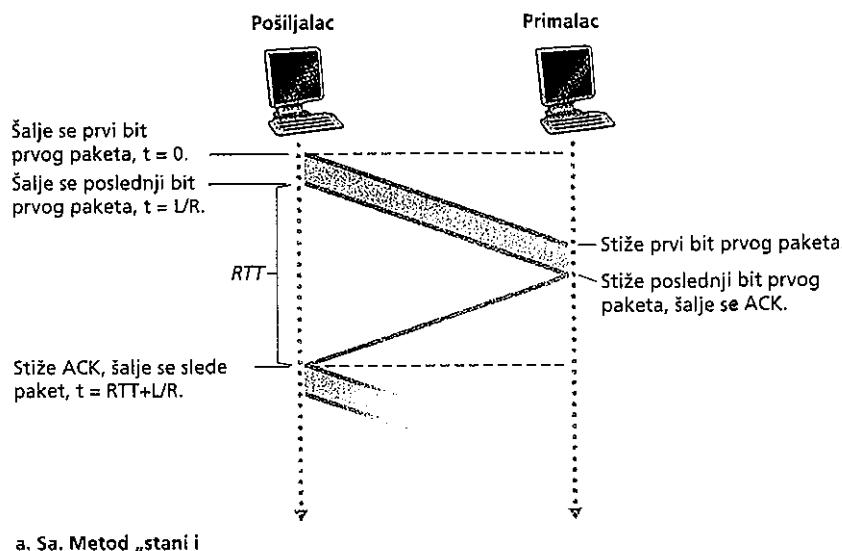
Radi jednostavnosti, pretpostavljamo da su ACK paketi izuzetno mali (da bismo mogli zanemariti vreme potrebno za njihov prenos) i da primalac može da pošalje ACK čim primi poslednji bit iz paketa podataka. U tom slučaju se ACK pojavljuje kod pošiljaoca nakon  $= RTT + L/R = 30,008$  milisekundi. U tom trenutku pošiljalac može da počne sa prenosom sledeće poruke. Prema tome, od 30,008 milisekundi pošiljalac se bavio slanjem samo tokom 0,008 milisekundi. Ako definisemo iskorišćenje pošiljaoca (ili kanala) kao procenat vremena u kome pošiljalac zaista šalje bitove kanalu, analiza slike 3.18(a) pokazuje da iskorišćenje protokola „stani i čekaj“ nije baš za pohvalu. Ono iznosi:

$$U_{pošiljalac} = \frac{L/R}{RTT + L/R} = \frac{0,008}{30,008} = 0,00027$$

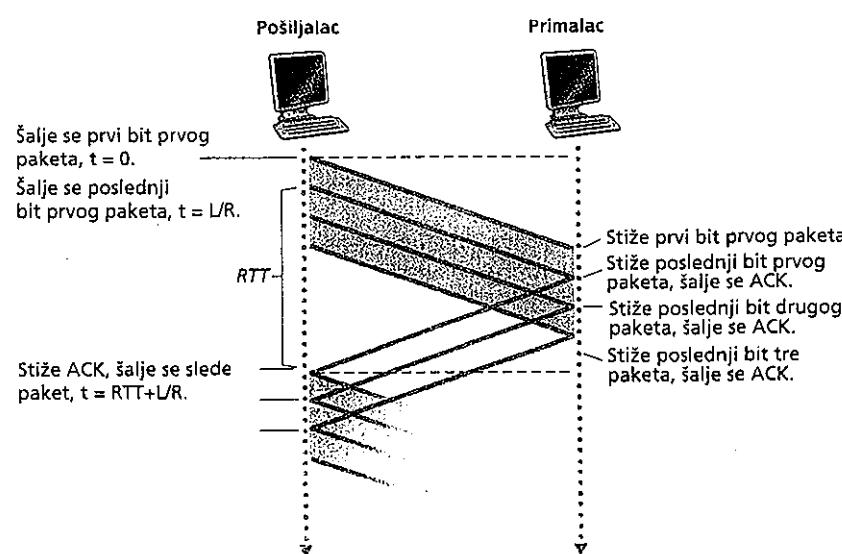
To jest, pošiljalac je uposlen samo 2,7 stotih delova jednog procenta vremena. Drugačije posmatrano, pošiljalac je bio u stanju da za 30,008 milisekundi pošalje samo 1 000 bajtova, što znači efektivno samo 267 kb/s - mada je na raspolažanju imao link od 1 Gb/s! Zamislite nesrećnog menadžera mreže koji je upravo platilo celo bogatstvo za link kapaciteta od 1 gigabita, a kroz njega uspeva da postigne protokol od samo 267 kb/s! Ovo je očigledan primer kako mrežni protokoli mogu da ograniči mogućnosti koje pruža hardverski sloj mreže. Osim toga, zanemarili smo vremena obrade protokola nižih slojeva kod pošiljaoca i primaoca, kao i kašnjenje zbog obrade i čekanja u redu do kojeg može doći na bilo kojem usputnom ruteru između pošiljaoca i primaoca. Da smo uključili i te efekte, ukupno kašnjenje bilo bi još veće a loše performanse još izraženije.

Rešenje ovog konkretnog problema sa performansama je sasvim jednostavno: umesto da radi po principu „stani i čekaj“, pošiljaocu se dozvoljava da šalje više paketa bez čekanja na potvrde, kako je prikazano na slici 3.17(b). Na slici 3.18(b) prikazano je da se iskorišćenje pošiljaoca u suštini utrostručava ako pošiljalac može da pošalje tri paketa bez čekanja na potvrde. Pošto mnoštvo paketa u tranzitu od pošiljaoca do primaoca može da se zamisli kao popunjavanje cevi, ova tehnika se naziva cevovodnom obradom. Cevovodna obrada ima sledeće posledice na protokole pouzdanog transfera podataka:

- ◆ Mora se povećati raspon rednih brojeva, pošto svaki paket u tranzitu (ne računajući ponovna slanja) mora da ima jedinstven redni broj, a može da postoji više tranzitnih paketa koji su još nisu potvrđeni.
- ◆ Na otpremnoj i prijemnoj strani protokola može se javiti potreba da se privremeno čuva više paketa. U najmanju ruku, pošiljalac će morati privremeno da čuva pakete koji su poslati, a da još nije potvrđen njihov prijem. Privremeno čuvanje primljenih paketa ponekad je potrebno i kod primaoca, kao što ćemo opisati u daljem izlaganju.



a. Sa. Metod „stani i



b. Cevovodna obrada

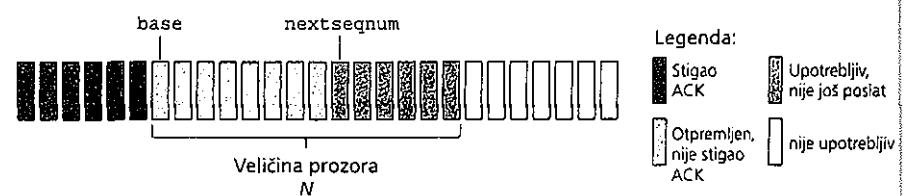
Slika 3.18 ♦ „Stani i čekaj“ i cevododno slanje

- Potreban raspon rednih brojeva i zahtevi privremenog čuvanja zavisiće **od** načina na koji protokol za transfer podataka reaguje na izgubljene i oštećene pakete, i na pakete koji previše kasne. Dva osnovna pristupa ispravljanju grešaka cevovodne obrade možemo nazvati: **Vrati-se-za-N** (Go-Back-N - GBN) i **selektivno ponavljanje**.

### 3.4.3 GBN

U GBN protokolu, pošiljaocu se dozvoljava da pošalje više paketa (ako su dostupni) bez čekanja na potvrdu, ali se ograničava maksimalan dozvoljeni broj  $N$  nepotvrđenih paketa u cevi. Na slici 3.19 prikazan je raspon rednih brojeva u GBN protokolu iz aspekta pošiljaoca. Ako definišemo base kao redni broj najstarijeg nepotvrđenog paketa, a nextseqnum kao najmanji neupotrebljeni redni broj (tj. redni broj sledećeg paketa za slanje), možemo uočiti četiri intervala u rasponu rednih brojeva. Redni brojevi u intervalu  $[0, \text{base}-1]$  odgovaraju paketima koji su već preneti i prijem potvrđen. Interval  $[\text{base}, \text{nextseqnum}-1]$  odgovara paketima koji su poslati, ali nisu još potvrđeni. Redni brojevi u intervalu  $[\text{nextseqnum}, \text{base}+N-1]$  koriste se za pakete koji mogu odmah da se šalju čim podaci stignu iz gornjeg sloja. Na kraju, redni brojevi veći ili jednaki  $\text{base}+N$  ne mogu se koristiti sve dok ne stigne potvrda prijema za neki nepotvrđeni paket koji se trenutno nalazi u cevi (konkretno, paket sa rednim brojem base).

Kako je nagovušeno na slici 3.19, raspon dozvoljenih rednih brojeva za otpremljene, a još nepotvrđene pakete može se smatrati „prozorom“ veličine nad rasponom rednih brojeva. Tokom rada protokola, ovaj prozor klizi unapred nad prostorom rednih brojeva. Zbog toga, A<sup>se</sup> često naziva **veličinom prozora**, a sam GBN protokol **kliznog prozora**. Možda se čudite zašto se broj nepotvrđenih paketa ograničava na neku vrednost  $N$ . Zašto se ne bi dozvolio neograničen broj takvih paketa? Videćemo u odeljku 3.5 daje kontrola toka jedan od razloga da se pošiljaocu nametne granica. Drugi razlog ćemo ispitati u odeljku 3.7 gde se opisuje TCP kontrola zagušenja,

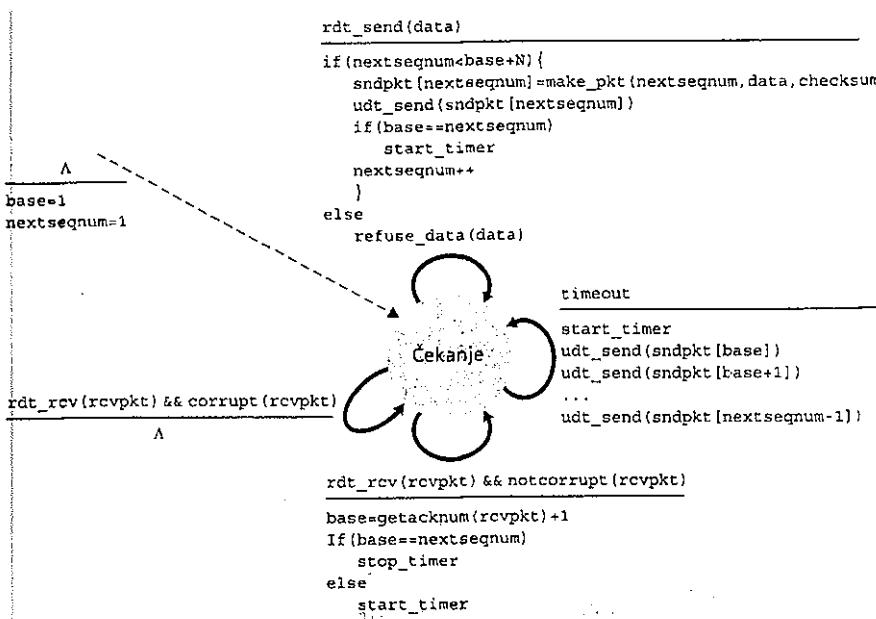


Slika 3.19 ♦ Redni brojevi u GBN iz aspekta pošiljaoca

U praksi, redni broj paketa stavlja se u polje fiksne dužine u zaglavljulu paketa. Ako je  $k$  broj bitova u polju rednog broja paketa, raspon rednih brojeva je  $[0,2^k-1]$ . Sa konačnim rasponom rednih brojeva, sve računanje vezano za redne brojeve mora se vršiti aritmetikom po modulu  $2^k$ . (To jest, prostor rednih brojeva može se posmatrati kao prsten veličine  $2^k$ , gde za rednim brojem  $2^k-1$  neposredno sledi redni broj 0.) Setite se da je protokol rdt3.0 imao redne brojeve od 1 bita i raspon rednih brojeva  $[0,1]$ . Nekoliko problema na kraju ovog poglavlja istražuje posledice konačnog raspona rednih brojeva. Videćemo u odeljku 3.5 da TCP ima 32-bitno polje rednog broja gde TCP redni brojevi numerišu bajtove u toku, a ne pakete.

Na slikama 3.20 i 3.23 dat je proširen FSM opis pošiljaoca i primaoca jednog GBN protokola zasnovanog na ACK potvrda i bez NAK potvrda. Ovaj FSM opis nazivamo proširem zato što smo dodali promenljive (slično promenljivama programskog jezika) za brojeve base i nextseqnum i takođe dodati operacije nad tim promenljivama i uslovne aktivnosti vezane za te promenljive. Primetićete da proširena FSM specifikacija počinje donekle da liči na specifikaciju u programskom jeziku. [Bochman 1984] sadrži izvrstan pregled dodatnih proširenja FSM tehnika kao i drugih tehnika za defmisanje protokola zasnovanih na programskim jezicima.

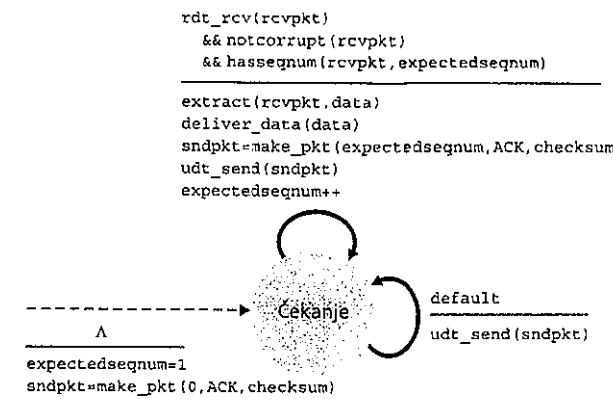
GBN pošiljalac mora da reaguje na tri vrste događaja:



Slika 3.20 ♦ Prošireni FSM opis GBN pošiljaoca

- ◆ *Poziv odozgo.* Kada se odozgo pozove rdt\_send(), pošiljalac prvo proverava da li je prozor pun, tj. da li postoji  $N$  nepotvrđenih paketa. Ako prozor nije pun, paket se pravi i šalje, i promenljive se ažuriraju na odgovarajući način. Ako je prozor pun, pošiljalac jednostavno vraća podatke gornjem sloju, što implicitno znači da je prozor pun. Prepostavka je da bi gornji sloj trebalo ponovo da pokuša kasnije. U stvarnim implementacijama, verovatnije je da će pošiljalac sačuvati te podatke u privremenoj memoriji (neće ih slati odmah), ili možda ima mehanizam sinhronizacije (na primer, semafor ili zastavicu) koji gornjem sloju dozvoljava da pozove rdt\_send() samo ako prozor nije pun.
- ◆ *Prijem ACK-a.* U našem GBN protokolu, potvrda za paket sa rednim brojem  $n$  smatraće se kumulativnom potvrdom, koja označava da su svi paketi sa rednim brojevima manjim od  $h$  i jednakim  $rt$  pravilno primljeni kod primaoca. Vratićemo se uskoro ovom pitanju prilikom opisa prijemne strane GBN protokola.
- ◆ *Dogadjaj lajm-aij.* Ime GBN protokola potiče od ponašanja pošiljaoca u slučaju izgubljenih paketa ili paketa koji previše kasne. Kao i u protokolu „stani i čekaj“, koristiće se tajmer za oporavak od gubitka paketa sa podacima ili paketa sa potvrdom. Ako nastupi tajm-aut, pošiljalac ponovo šalje sve pakete koji su prethodno poslati, a nisu još potvrđeni. Naš pošiljalac na slici 3.20 koristi samo jedan tajmer, koji možemo shvatiti kao tajmer najstarijeg predatog a još nepotvrđenog paketa. Ako stigne ACK, a ima još predatih i nepotvrđenih paketa, tajmer se ponovo pokreće. Ako nema više nepotvrđenih paketa, tajmer se zaustavlja.

Aktivnosti primaoca u GBN protokolu takođe su jednostavne. Ako se paket sa rednim brojem  $n$  primi ispravno i pravim redosledom (tj. poslednji podaci isporučeni gornjem sloju bili su iz paketa sa rednim brojem  $>i-1$ ), primalac šalje ACK za



Slika 3.21 ♦ Prošireni FSM opis GBN primaoca

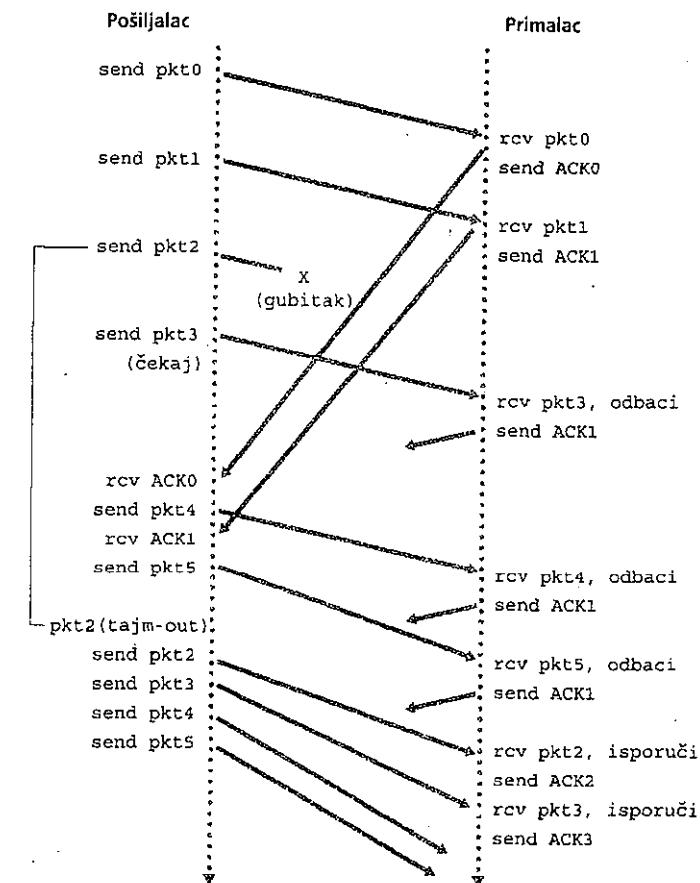
paket sa rednim brojem  $n$  i predaje podatke iz paketa gornjem sloju. U svim ostalim slučajevima, primalac odbacuje paketi ponovo šalje ACK za poslednji paket primljen ispravnim redosledom. Obratite pažnju na to da pošto se paketi isporučuju gornjem sloju jedan po jedan, ako je paket  $k$  primljen i isporučen, to znači i da su svi paketi sa rednim brojevima manjim od  $k$  takođe isporučeni. Prema tome, korišćenje kumulativnih potvrda je prirodno u GBN protokolu.

U našem GBN protokolu, primalac odbacuje pakete koji ne idu ispravnim redosledom. Mada izgleda glupo i rasipno da se odbaci paket koji je pravilno primljen (iako nije po redu), za takav postupak postoji opravdanje. Ne zaboravite da primalac mora redom da predaje pakete gornjem sloju. Pretpostavimo sada da se очekuje paket  $n$ , a stiže  $n+1$ . Pošto podaci moraju da se isporuče po redu, primalac *bi mogao* da sačuva paket  $n+1$  i isporuči ga gornjem sloju posle prijema i isporuke paketa  $n$ . Međutim, ako se paket  $n$  izgubio, i on i paket  $n+1$  biće kasnije ponovo poslati zbog GBN pravila ponovnog slanja na strani pošiljaoca. Prema tome, primalac može jednostavno da odbaci paket  $n+1$ . Prednost ovakvog pristupa je jednostavnije privremeno čuvanje kod primaoca - primalac ne mora privremeno da čuva *nijedan* paket izvan redosleda. Prema tome, dok pošiljalac mora da održava gornju i donju granicu svog prozora i položaj nextseqnum unutar tog prozora, jedino što primalac mora da održava je redni broj sledećeg očekivanog paketa. Ova vrednost čuva se u promenljivoj expectedseqnum prikazanoj u FSM opisu primaoca na slici 3.21. Naravno, nedostatak odbacivanja ispravno primljenog paketa jeste da i ponovno poslati paket može da se izgubi ili ošteti, što će zahtevati još ponovnih slanja.

Na slici 3.22 prikazan je rad GBN protokola za prozor veličine 4 paketa. Zbog ograničenja ove veličine prozora, pošiljalac šalje pakete od 0 do 3, ali zatim mora da čeka potvrdu prijema jednog ili više tih paketa, da bi mogao da nastavi sa slanjem. Sa prijemom svakog sledećeg ACK-a (na primer, ACK0 i ACK1), prozor klizi una-pred i pošiljalac može da prenese nove pakete (pkt4 odnosno pkt5). Na strani primaoca, paket 2 je izgubljen i zato se paketi 3,4 i 5 odbacuju, jer su van redosleda.

Na kraju opisa GBN protokola treba primetiti da bi implementacija ovog protokola u familiji protokola verovatno imala strukturu sličnu proširenom FSM opisu iz slike 3.20. Implementacija bi takođe verovatno bila u obliku različitih procedura za implementiranje aktivnosti koje treba preduzeti povodom različitih događaja koji nastupaju. U takvom **programiranju zasnovanom na dogadajima** različite procedure se pozivaju (aktiviraju) bilo iz drugih procedura u familiji protokola ili kao rezultat prekida. Kod pošiljaoca, ti dogadaji bi bili (1) poziv entiteta iz gornjeg sloja da se aktivira rdt\_send (), (2) tajmerski prekid i (3) poziv iz donjeg sloja da se pozove rdt\_rcv () kada stigne paket. Programerski zadaci na kraju poglavlja daće vam priliku da implementirate te rutine u simuliranom, ali realnom mrežnom okruženju.

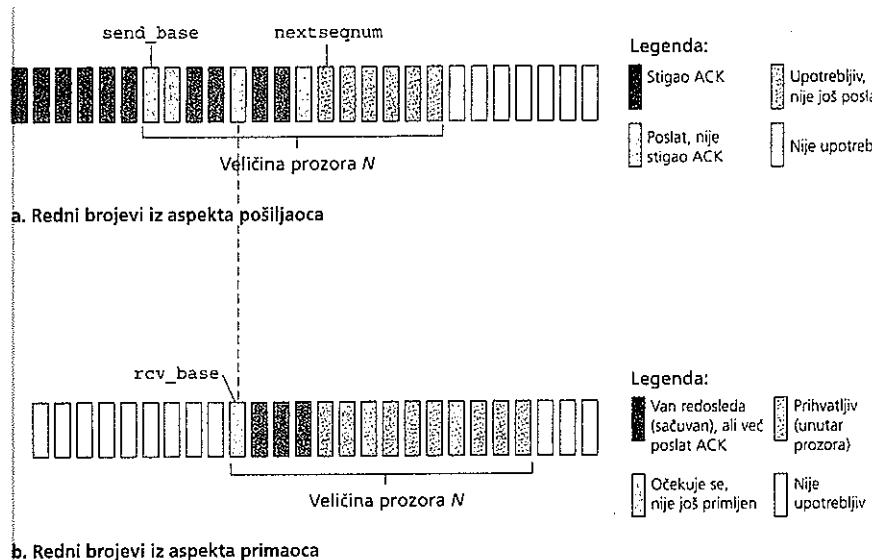
Ovde primećujemo da GBN protokol obuhvata skoro sve tehnike na koje ćemo naići u razmatranju komponenti pouzdanog transfera podataka u protokolu TCP, u odeljku 3.5. U ove tehnike spadaju upotreba rednih brojeva, kumulativnih potvrda, kontrolnih zbrojava i ponovnog slanja zbog tajm-auta.



**ilika 3.22 ♦ GBN u radu**

#### 3.4.4 Selektivno ponavljanje

GBN protokol dozvoljava pošiljaocu da potencijalno „popuni cevi” sa slike 3.17 paketima, i tako izbegne probleme nedovoljnog korišćenja kanala koje smo spominjali u vezi sa protokolom „stani i čekaj”. Međutim, ima situacija u kojima i sam GBN protokol ima probleme sa performansama. Konkretno, kada imamo veliku širinu prozora i kašnjenje zbog propusnog opsega, u cevi može da se nađe mnogo paketa. Jedna greška u paketu može da dovede do toga da GBN ponovo šalje velik broj paketa, od kojih mnoge nepotrebno. Sa povećanjem verovatnoće za nastajanje



**Slika 3.23 ♦ SR protokol – prostor rednih brojeva iz aspekta pošiljaoca i primaoca**

grešaka u kanalu, cev može da se zaguši tim nepotrebnim ponovnim slanjima. Zamislite da, u primeru sa diktiranjem poruke, za svaku nerazgovetnu reč treba ponoviti okolnih hiljadu reči (ako je, na primer, veličina prozora hiljadu reči). Diktiranje bi se usporilo zbog toliko ponavljanja.

Kao što ime govori, protokoli sa selektivnim ponavljanjem (*selective repeat*, SR) izbegavaju nepotrebna dodatna slanja tako što pošiljalac ponovo šalje samo one pakete za koje sumnja da su pogrešno primljeni (tj. izgubljeni ili oštećeni). Da bi se omogućilo pojedinačno ponovno slanje prema potrebi, primalac mora *pojedinačno* da potvrđuje ispravno primljene pakete. Veličina prozora A' i ovde se koristi da bi se ograničio broj zaostalih nepotvrđenih paketa u cevi. Međutim, za razliku od GBN-a, pošiljalac je već dobio ACK za neke pakete u prozoru. Na slici 3.23 prikazanje prostora rednih brojeva iz aspekta SR pošiljaoca. Na slici 3.24 opisane su različite aktivnosti koje preduzima SR pošiljalac.

SR primalac će potvrditi ispravno primljeni paket bez obzira na to da li je stigao u pravilnom redosledu. Paketi izvan redosleda se čuvaju dok ne stignu nedostajući

1. *Podaci primljeni odozgo.* Kada se odozgo prime; podaci, SR.pošiljalalaQprb-veravasledcíslobodni redni broj ža paket; Akoje<sup>A</sup> pošij'aoca,:pbdacj'se.palcuju:i šalju; inače se privremeno<sup>1</sup> čuVaju ili'vraćaju gornjem, sloj u'radi kasnjeg slanja'kao u GBN-u. : " " ;? '■:

2. *Tajmaut.* tajmeri se koriste kao zaštita odgubitča paketa; Medut<sup>A</sup> paket sada.mora da iina vlastiti; logički tajmer, pbš'6-če "se prilikom tajmauta slati samo jedan paket. Za imitiranje rada više logičkih tajmera.može še'koristi samo jedan hardverski tajmer [Varghes'e 1997]. " \

3. *Primljen je ACK.* Ako stigne ACK, Sli pošiljalac označava paket kao prim-. Ijen, ukoliko se oh nalazi u prozoru.Ukoliko je redni broj paketa jednak send\_base, početak prozora se pomera unapred do ncpotvrdehog paketa' sa najmanjim rednim brojem. Ako se prilikom.ppmjeranja prozora otkriju . neposlati paketi čiji se redni brojevi sada nalaze u prozoru.ti paketi.se šalju..

**Slika 3.24 ♦ Događaji i aktivnosti SR pošiljaoca**

paketi (tj. paketi sa nižim rednim brojevima), a tada se gornjem sloju redom isporučuje grupa paketa. Na slici 3.25 navedene su različite aktivnosti koje preduzima SR primalac. Na slici 3.26 dat je primer SR rada u slučaju izgubljenih paketa. Obratite pažnju na to da na slici 3.26 primalac prvo smešta pakete 3, 4 i 5 u privremenu memoriju , pa ih isporučuje gornjem sloju zajedno sa paketom 2 kada paket 2 konačno stigne.

Važno je primetiti da u koraku 2 na slici 3.25 primalac ponovo potvrđuje (ume-sto da zanemaruje) ranije primljene pakete sa rednim brojevima *manjim* od trenutne osnove prozora. Budite sigurni daje ovo ponovo potvrđivanje zaista potrebno. Ako uzmemo prostore rednih brojeva pošiljaoca i primaoca na slici 3.23, kad ne bi bilo ACK potvrde za paket send\_base koju primalac šalje pošiljaocu, pošiljalac bi kasnije poslao paket send\_base, mada je jasno (nama, ali ne i pošiljaocu!) daje primalac već dobio taj paket. Kada primalac ne bi potvrdio taj paket, prozor pošiljaoca ne bi se nikad pomerio! Ovaj primer ilustruje jedan važan aspekt SR protokola (kao i mnogih drugih protokola). Pošiljalac i primalac neće uvek imati identično saznanje o tome staje pravilno primljeno, a šta nije. Kod SR protokola, ovo znači da prozori pošiljaoca i primaoca neće uvek biti identični.

Neusklađenost između prozora pošiljaoca i primaoca ima značajne posledice u situaciji sa konačnim rasponom rednih brojeva. Pogledajte šta bi se dogodilo, na primer, sa konačnim rasponom od 4 redna broja (0, 1, 2, 3) i veličinom prozora 3. Pretpostavimo da su poslati paketi od 0 do 2 pravilno primljeni, i da ih je primalac potvrdio. U tom trenutku se prozor primaoca nalazi na četvrtom, petom i šestom paketu čiji su redni brojevi 3, 0 i 1. Uzmimo sada dva scenarija. U prvom scenariju, prikazanom na slici 3.27(a), ACK-ovi za prva tri paketa su se izgubili i pošiljalac

1. *Pravilno primljen paket sa rednim brojem u intervalu [ rcv\_base, rcv\_base+N-1 ].* U ovom slučaju, primljeni paket se nalazi unutar primaocvog prozora, pa se poslijecu šalje selektivni ACK paket. Ako paket nije ranije primljen, on se privremeno smešta u privremenu memoriju. Ako je redni broj paketa jednak osnovi primaocvog prozora ( rcv\_base na slići 3.22 ), ovaj paket, kao i eventualno paketi smešteni u privremenu memoriju ranije sa uzastopnim rednim brojevima koji zatim slede (oni koji počinju od rcv\_base ) isporučuju se gornjem sloju. Prozor primaoca se zatim pomera unapred za onoliko paketa koliko ih je isporučeno gornjem sloju. Proučite primer na slići 3.26. Kada stigne paket sa rednim brojem rcv\_base=2 , on sam i paketi 3, 4 i 5 mogu se isporučiti gornjem sloju.
  2. *Primljen je paket sa rednim brojem u intervalu [ rcv\_base-N, rcv\_base-1 ].* U ovom slučaju, potrebno je napraviti ACK mada je ovo paket koji je primalac već ranije potvrđio.
  3. *Inače. Zanemariti paket.*

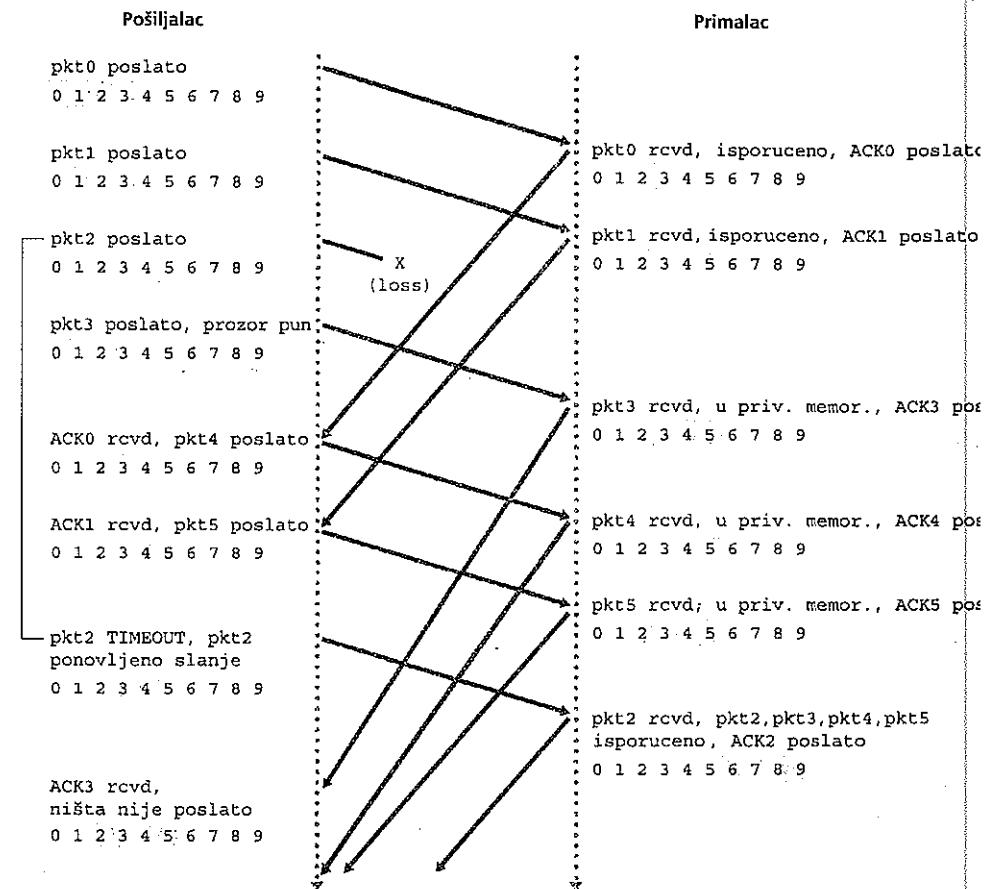
**Slika 3.25** ♦ Dogadjaji i aktivnosti SR primaoca

ih ponovo šalje. Primalac, znači, dobija paket sa rednim brojem 0 - kopiju prvog poslatog paketa.

U drugom scenariju prikazanom na slici 3.27(b), ACK-ovi za prva tri paketa su svih pravilno isporučeni. Pošiljalac, znači, pomera svoj prozor unapred i šalje četvrti, peti i šesti paket sa rednim brojevima 3, 0 i 1. Paket sa rednim brojem 3 se gubi, ali paket sa rednim brojem 0 stiže - paket koji sadrži *nove* podatke.

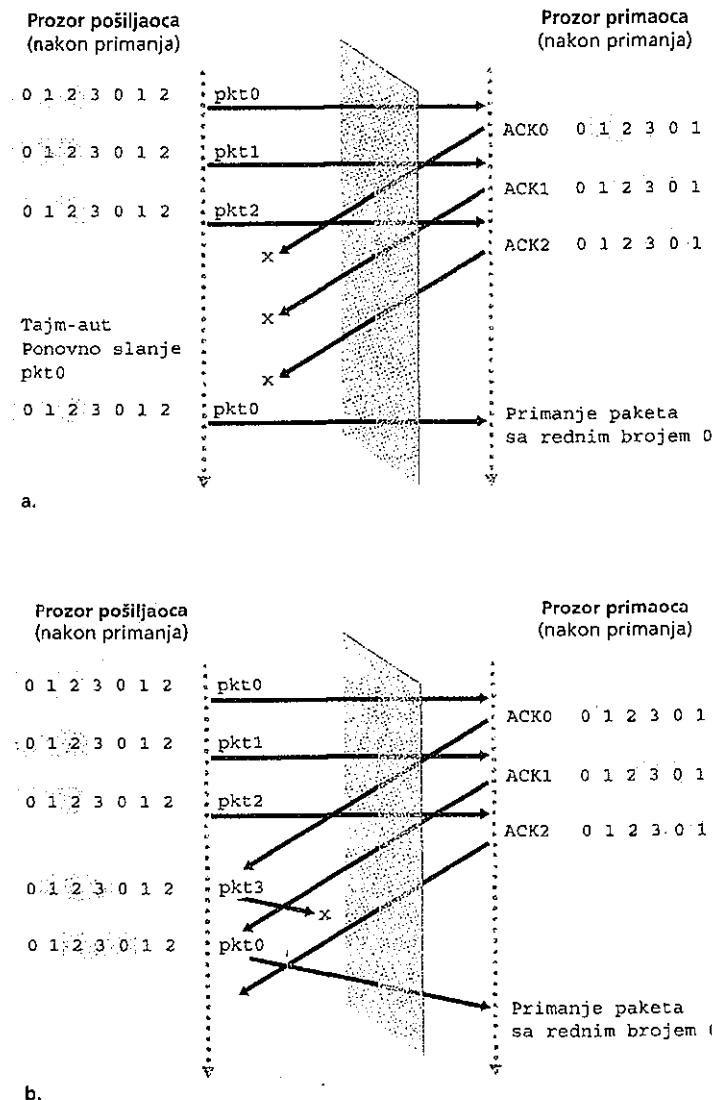
Na slici 3.27 razmotrićemo situaciju iz aspekta primaoca pred kojim se nalazi virtualna zavesa, pošto on ne može da „vidi“ šta se dešava na strani pošiljaoca. Primalac jedino vidi redosled poruka koje prima iz kanala i koje šalje u kanal. Iz njegovog aspekta, dva scenarija prikazana na slici 3.27 su *identična*. Ne postoji način da primalac razlikuje ponovno slanje prvog paketa od prvog slanja petog paketa. Jasno je da veličina prozora, koja je za jedan manja od veličine prostora rednih brojeva, rfije dobra. Kolika bi trebalo da bude veličina prozora? U jednom problemu na kraju poglavlja imaćete zadatak da pokažete da za SR protokole veličina prozora mora biti manja ili jednaka polovini veličine prostora rednih brojeva.

Ovim zaključujemo opis protokola za pouzdani transfer podataka. Obradili smo *dosta* i uveli niz mehanizama kojima se obezbeđuje pouzdani transfer podataka. U tabeli 3.1 nalazi se rekapitulacija ovih mehanizama. Pošto ste videli njihovu pri-menu i možete da razumete celinu, predlažemo da ponovo pregledate ovaj odeljak da biste videli kako smo ove mehanizme postupno dodavali da bismo objasnili sve složenije (i realnije) modele kanala koji povezuju pošiljaoca i primaoca, ili poboljšane performanse protokola.



**Slika 3.26** ♦ SR u radu

Zaključimo naš opis protokola za pouzdani transfer podataka razmatranjem jedine preostale prepostavke za model kanala preko koga se vrši prenos. Setite se da smo prepostavili da se unutar kanala između pošiljaoca i primaoca paketima ne može menjati redosled. To je uglavnom realna prepostavka kada su pošiljalac i primalac povezani jednim jedinim fizičkim kablom. Međutim, kada je „kanal“ u stvari mreža, može doći do izmenjenog redosleda paketa. Jedna manifestacija izmenje-nog redosleda paketa je pojava starog primerka paketa sa rednim brojem ili brojem potvrde  $x$ , iako prozori pošiljaca i primaoca ne sadrže  $x$ . Ako ima promene u redosledu paketa, tada će se primeti da je redosredjivanje potrebno.



**Slika 3.27** ♦ Dilema SR primaoca sa prevelikim prozorima: novi paket ili ponovno slanje?

| Mehanizam                 | Korišćenje, komentari                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kontrolni zbir            | Koristi se za otkrivanje bitskih grešaka u paketu koji se prenosi.                                                                                                                                                                                                                                                                                                                                              |
| Tajmer                    | Koristi se za tajmaut i ponovno slanje paketa zbog mogućnosti da je paket (ili njegov ACK) izgubljen u kanalu. Pošto do tajmarta može doći ako paket nije izgubljen, nego kasni (prevremeni tajmaut), ili kada je primaoc primio paket ali se izgubio ACK od primaoca ka pošiljocu, primaoc može da dobije duplike paketa.                                                                                      |
| Redni broj                | Koristi se za sekvenčalno numerisanje paketa podataka koji teku od pošiljoca do primaoca. Prekidi u rednim brojevima primljenih paketa omogućavaju primaocu da otkrije da je paket izgubljen. Paketi sa dupliranim rednim brojevima omogućavaju primaocu da otkrije postojanje duplih paketa.                                                                                                                   |
| Potvrda prijema           | Koristi je primaoc da bi obavestio pošiljocu da je pravilno primljen paket ili skup paketa. Potvrde obično sadrže redni broj paketa koji se potvrđuje. Potvrde mogu biti pojedinačne ili kumulativne, u zavisnosti od protokola.                                                                                                                                                                                |
| Negativna potvrda prijema | Koristi je primaoc da bi obavestio pošiljocu da paket nije pravilno primljen. Negativna potvrda obično sadrži redni broj paketa koji nije pravilno primljen.                                                                                                                                                                                                                                                    |
| Prozor, cevovodna obrada  | Pošiljalac se može ograničiti na slanje samo onih paketa čiji su redni brojevi naložene u datom rasponu. Ako se dozvoli slanje više paketa bez čekanja na potvrdu, iskorišćenost pošiljocu je bolja nego u režimu rada „stani i čekaj“. Uskoro ćemo videti da se veličina prozora može odrediti na osnovu primaoceve sposobnosti da prima i čuva poruke, ili na osnovu zagruženja mreže, ili i jednog i drugog. |

**Tabela 3.1** ♦ Rekapitulacija mehanizama za pouzdani transfer podataka i njihovog korišćenja

sledu paketa, može se smatrati da kana! privremeno smešta pakete u privremenu memoriju i spontano ih emituje u *bilo kom* kasnjem trenutku. Pošto redni brojevi mogu da se koriste više puta, potrebno je posebno paziti na takve duplike paketa. U praksi se obično kao zaštita koristi zabrana da se redni broj ponovo upotrebi sve dok pošiljalac ne bude „siguran“ da u mreži nema više nijednog ranije posla-tog paketa sa rednim brojem  $x$ . To se postiže pretpostavkom da paket ne može da „živi“ u mreži duže od nekog fiksнog maksimalnog vremena. U TCP proširenjima za mreže velike brzine [RFC 1323] prihvачен je maksimalni životni vek paketa od približno 3 minuta. U knjizi [Sunshine 1978] opisan je metod korišćenja rednih brojeva na takav način da se potpuno izbegni problemi izmenjenog redosleda.

## 3.5 Transport sa konekcijom: TCP

Pošto smo upoznali osnovne principe pouzdanog transfera podataka, preći ćemo na TCP - Internetov protokol transportnog sloja sa konekcijom. U ovom odeljku videćemo da se za obezbeđenje pouzdanog transfera podataka TCP oslanja na mnoge osnovne principe koje smo opisali u prethodnom odeljku, uključujući otkrivanje grešaka, ponovno slanje, kumulativne potvrde, tajmere i polja u zagлављу za redne brojeve paketa i redne brojeve potvrda. TCP je definisan u dokumentima RFC 793, RFC 1122, RFC 1323, RFC 2018 i RFC 2581.

### 3.5.1 TCP konekcija

Za TCP se kaže da je **sa konekcijom** zato što pre početka slanja podataka iz jedne aplikacije u drugu, ta dva procesa moraju prvo „da se upoznaju“ - tj. moraju jedan drugom da pošalju neke uvodne segmente da bi se uspostavili parametri transfera podataka koji sledi. Prilikom uspostavljanja TCP konekcije obe strane će postaviti početne vrednosti za mnoge promenljive TCP stanja (nekoliko njih opisuјemo u ovom odeljku i u odeljku 3.7) koje se odnose na TCP konekciju.

TCP „konekcija“ nije TDM ili FDM vod sa kraja na kraj kakvi postoje u mreži sa komutacijom vodova. Ona takođe nije ni virtuelno kolo (procitati poglavlje 1), pošto se stanje konekcije u potpunosti čuva u krajnjim sistemima. Protokol TCP se izvršava samo u krajnjim sistemima, a ne na usputnim elementima mreže (ruterima i mostovima). Usputni elementi mreže ne održavaju stanje TCP konekcije. U suštini, usputni ruteri uopšte nisu svesni TCP konekcije; oni vide samo datagrame.

TCP konekcija obezbuđuje transfer podataka **u punom dupleksu**. Ako postoji TCP konekcija između procesa A najednom računam i procesa B na drugom računam, tada podaci iz aplikacionog sloja mogu da teku od A do B a u isto vreme i od B do A. TCP konekcija je takođe uvek **od tačke do tačke** tj. između pojedinačnog pošiljaoca i pojedinačnog primaoca. Takođe „višezačno rutiranje“ (procitati odeljak 4.7>- transfer podataka od jednog pošiljaoca do više primalača u istoj operaciji slanja - nije moguća sa TCP-om. Za TCP su dva računara društvo, a treći je višak!

Pogledajmo sada kako se uspostavlja TCP konekcija. Uzmimo da proces koji se izvršava najednom računaru želi da uspostavi konekciju sa drugim procesom na drugom računaru. Verovatno se sećate da proces koji pokreće konekciju nazivamo klijentskim procesom, a drugi proces serverskim. Klijentski aplikacioni proces prvo obaveštava klijentski transportni sloj da želi da uspostavi konekciju sa procesom na serveru. U odeljku 2.6 naučili smo da Java klijentski program to postiže izdavanjem komande:

```
Socket clientSocket = new Socket("hostname", portNumber);
```

gde je hostname ime servera, a portNumber označava proces na serveru. Transportni sloj u klijentu zatim prelazi na uspostavljanje TCP konekcije sa TCP-om na serveru. Na kraju ovog odeljka opisaćemo relativno detaljno postupak uspostavljanja konekcije. Za sada je dovoljno da znamo da klijent prvo šalje poseban TCP segment; server odgovara drugim posebnim TCP segmentom; a na kraju klijent ponovo odgovara trećim posebnim segmentom. Prva dva segmenta ne sadrže nikakve „korisne podatke“, tj. nikakve podatke aplikacijskog sloja; treći segment već može da sadrži korisne podatke. Pošto se među računarima razmenjuju tri segmenta, ovaj postupak uspostavljanja konekcije često se naziva **sinhronizovanjem u tri koraka**.

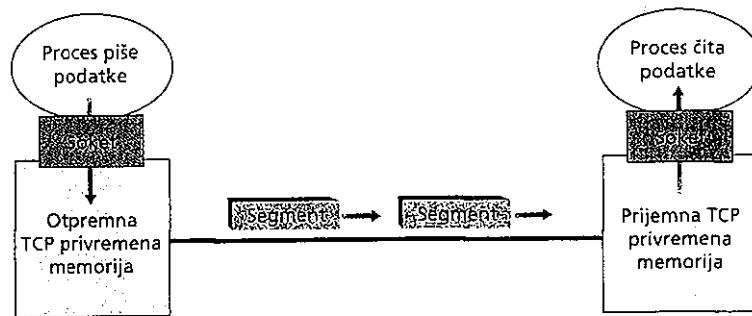
Pošto se TCP konekcija uspostavi, dva aplikaciona procesa mogu jedan drugom da šalju podatke. Razmotrimo slanje podataka od klijentskog procesa serverskom procesu. Klijentski proces šalje tok podataka kroz socket (vrata procesa), kao što je opisano u odeljku 2.6. Pošto podaci prođu kroz ta vrata, oni se nalaze u rukama TCP-a koji se izvršava na klijentu. Kao stoje prikazano na slici 3.28, TCP usmerava ove podatke u **otpremnu privremenu memoriju** te konekcije, a to je jedna od privremenih memorija koje se rezervišu tokom početnog sinhronizovanja u tri koraka. S vremenom na vreme, TCP će „zahvatati“ delove podataka iz otpremne privremene memorije. Zanimljivo je da u TCP specifikaciji [RFC 793] nema strogog pravila

### KRATAK OSVRT

#### VINTON SERF, ROBERT KAN I TCP/IP

Početkom sedamdesetih godina počele su da se množe mreže sa komutiranjem paketa, od kojih je samo jedna bila ARPAnet - preteča Interneta. Svaka od ovih mreža imala je vlastiti protokol. Dva istraživača, Vinton Serf i Robert Kan, uvideli su značaj međusobnog povezivanja. Hh mreža i izmisliла međumrežni protokol po imenu TCP/IP što je skraćenica za Transmission Control Protocol/Internet Protocol. Mada su Serf i Kan na početku posmatrali taj protokol kao celinu, on je kasnije podeljen na dva dela, TCP i IP, koji funkcionišu odvojeno. Serf i Kan objavili su izveštaj o protokolu TCP/IP u maju 1974 u IEEE Transactions on Communications Technology.

Protokol TCP/IP koji predstavlja osnovu današnjeg Interneta, smišljen je pre PC-ja i radnih stanica, pre širenja Etherнетa i drugih tehnologija za lokalne mreže, pre Weba, pre protoka audio signala u realnom vremenu i pre časnika. Serf i Kan uvideli su potrebu za mrežnim protokolom koji sa jedne strane obezbeđuje široku podršku za primene koje će se tek definisati, a s druge strane omogućava zajednički rad svih računara i protokola u sloju veze.



Slika 3.28 ♦ Otpremna i prijemna TCP privremena memorija

o lome kada bi TCP trebalo da pošalje podatke iz privremene memorije; kaže se da TCP treba da „Šalje podatke u segmentima prema vlastitom nahođenju“. Maksimalna količina podataka koja može da se zahvati i stavi u jedan segment ograničena je vrednosću promenljive MSS (*Maximum Segment Size*). MSS zavisi od TCP implementacije (koja zavisi od operativnog sistema) i često može da se konfiguriše; uobičajene vrednosti su 1 460 bajtova, 536 bajtova i 512 bajtova. (Ove veličine segmenta često se biraju tako da se izbegne IP fragmentiranje koje ćemo opisati u sledećem poglavljiju.) Obratite pažnju na to daje MSS maksimalna količina podataka aplikacionog sloja u segmentu, a ne maksimalna veličina TCP segmenta gde su uključena i zaglavlj. (Ovi nazivi malo zbnjuju, ali moramo ih prihvati jer su već veoma ustaljeni.)

TCP dopunjava svaki komad klijentskih podataka jednim TCP zaglavljem i tako pravi TCP segmenti. Segmenti se predaju naniže mrežnom sloju gde se pojedinačno enkapsuliraju u IP datagrame mrežnog sloja. IP datagrami se zatim šalju u 'mrežu'. Kad TCP na drugom kraju primi segment, podaci segmenta stavljuju se u prijemnu privremenu memoriju TCP konekcije, kao što je prikazano na slici 3.28. Aplikacija čita tok podataka iz ove privremene memorije. Svaka strana konekcije ima svoju otpremnu i svoju prijemnu privremenu memoriju. (Predlažemo čitaocu da pogleda onlajn aplet kontrole toka na adresi <http://wwwawi.com/kurose-ross> koji sadrži animaciju otpremne i prijemne privremene memorije.)

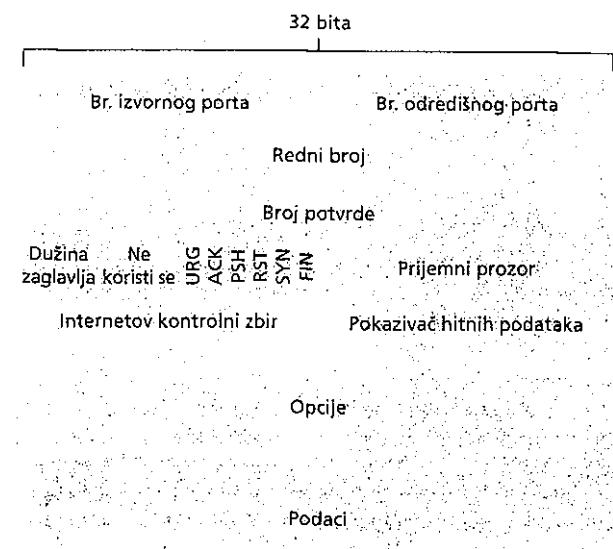
Iz ovog opisa vidimo da se TCP konekcija sastoji od privremene memorije, promenljivih i soketa konekcije za proces najednom računaru i druge grupe privremene memorije, promenljivih i soketa konekcije za proces na drugom računaru. Kao što je ranije spomenuto, u mrežnim elementima (ruterima, komutatorima i repetitorima) konekciji se ne dodeljuju nikakve privremene memorije niti promenljive.

### 3.5.2 Struktura TCP segmenta

Pošto smo ukratko pogledali TCP konekciju, proučimo sada strukturu TCP segmenta. TCP segment se sastoji od više polja zaglavja i jednog polja podataka. Polje podataka sadrži jedan komad aplikacijskih podataka. Kao što je već pomenuto, maksimalna veličina polja podataka u segmentu je MSS. Kada TCP šalje veliku datoteku, kao što je slika sa neke veb stranice, on obično deli datoteku na komade veličine MSS (osim poslednjeg komada, koji će često bili manji od parametra MSS). Interaktivne aplikacije, međutim, često prenose komade podataka koji su manji od parametra MSS; na primer, u aplikacijama za daljinsko prijavljivanje kao sto je Telnet, polje podataka u TCP segmentu često je dugačko samo 1 bajt. Pošto je TCP zaglavje obično dugačko 20 bajtova (12 bajtova više od UDP zaglavja), segmenti koje šalje Telnet ponekad su dugački samo 21 bajt.

Na slici 3.29 prikazana je struktura TCP segmenta. Kao i kod UDP segmenta, zaglavje sadrži broj izvornog i odredišnog porta, koji se koriste za multipleksiranje i demultipleksiranje podataka iz aplikacija gornjeg sloja i prema njima. Takođe, kao i u UDP segmentima, zaglavje sadrži i polje kontrolnog zbirka. Zaglavlj TCP segmenta sadrži još i sledeća polja:

- ♦ 32-bitno polje rednog broja i 32-bitno polje broja potvrde koje TCP pošiljalac i primalac koriste za implementiranje usluge pouzdanog transfera podataka što ćemo kasnije opisati.



Slika 3.29 ♦ Struktura TCP segmenta

- ◆ 16-bitno polje prijemnog prozora koristi se za kontrolu toka. Uskoro ćemo videti da se ono koristi da bi se saopštilo koliko bajtova primalac pristaje da prihvati.
- ◆ 4-bitno polje dužine zaglavlja navodi dužinu TCP zaglavlja u 32-bitnim recima. TCP zaglavlje može biti promenljive dužine zbog polja TCP opcija koje ćemo kasnije opisati. (Obično je polje opcija prazno, pa je dužina uobičajenog TCP zaglavlja jednaka 20 bajtova.)
- ◆ Opciono polje opcija promenljive dužine koje se koristi kada pošiljalac i primalac pregovaraju o maksimalnoj dužini segmenta (MSS) ili kao faktor za podešavanje prozora koji se koristi u mrežama velike brzine. Definisana je takođe i opcija vremenskog pečata. Više detalja možete naći u dokumentima RFC 845 i RFC 1323.
- ◆ Polje oznaka sadrži 6 bitova. Bit ACK, ukoliko je postavljen, označava da je vrednost broja potvrde važeća i daje u pitanju segment potvrde. Bitovi RST, SYN i FIN koriste se prilikom uspostavljanja i prekidanja konekcije što ćemo opisati na kraju ovog odeljka. Kada je bit PSH postavljen, to znači da bi primalac trebalo istog trenutka da prosledi podatke gornjem sloju. Na kraju, bit URG se koristi da bi se označilo da u ovom segmentu ima podataka koje je gornji sloj na strani pošiljaoca označio kao „hitne“ (urgente). Lokacija poslednjeg bajta ovih hitnih podataka određena je 16-bitnim poljem pokazivača hitnih podataka. TCP mora da obavesti entitet gornjeg sloja na prijemnoj strani da postoje hitni podaci i da mu predla pokazivač na kraj tih hitnih podataka. (U praksi se PSH, URG i pokazivač hitnih podataka ne koriste. Međutim, ta polja pominjemo da bi opis bio potpun.)

#### Redni brojevi i brojevi potvrda

Dva najvažnija polja u zaglavju TCP segmenta su redni broj i broj potvrde. Ta polja predstavljaju bitan deo TCP usluge pouzdanog transfera podataka. Ali, pre nego što opišemo na koji način ova polja obezbeđuju pouzdan transfer podataka, prvo ćemo objasniti šta tačno TCP stavlja u njih.

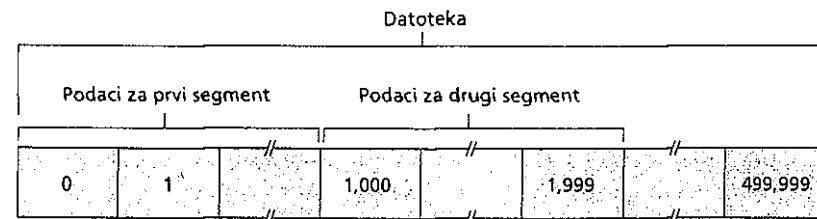
TCP posmatra podatke kao nestrukturisan, ali ureden tok bajtova. TCP tako i koristi redne brojeve, pa se oni odnose na tok prenetih bajtova, a ne na niz pre-netih segmenata. Redni broj za segment je, prema tome, redni broj prvog bajta u segmentu unutar toka bajtova. Pogledajmo jedan primer. Uzmimo da proces na računaru A želi preko TCP konekcije da pošalje tok podataka u proces na računaru B. TCP u računaru A implicitno će numerisati svaki bajt u toku podataka. Uzmimo da se tok podataka sastoji od jedne datoteke koja ima 500 000 bajtova, da MSS iznosi 1 000 bajtova, a da prvi bajt toka podataka ima redni broj 0. Kao što je prikazano na slici 3.30, TCP od tog toka podataka pravi 500 segmenata. Prvom segmentu dodeljuje se redni broj 0, drugom segmentu redni broj 1 000, trećem redni broj 2 000

i tako dalje. Redni broj se stavlja u polje rednog broja u zaglavju odgovarajućeg TCP segmenta.

Razmotrimo sada brojeve potvrda. Oni su nešto složeniji od rednih brojeva, Znate da je TCP potpuni dupleks, tako da računar A može da prima podatke od računara B u isto vreme dok i sam šalje podatke računaru B (u okviru iste TCP konekcije). Svaki segment koji stigne od računara B ima redni broj za podatke koji se šalju od B prema A. *Broj potvrde koji računar A stavlja u svoj segment je redni broj sledećeg bajta koji očekuje od računara B.* Dobro je pogledati nekoliko primera da bi se shvatilo šta se ovde događa. Uzmimo daje računar A primio od računara B sve bajtove numerisane od 0 do 535 i uzmimo da se spremi da pošalje jedan segment računaru B. Računar A očekuje bajt 536 i sve naredne bajtove iz toka podataka računara B. Zato računar A stavlja broj 536 u polje za broj potvrde sledećeg segmenta koji će poslati računaru B.

Kao drugi primer, uzmimo da je računar A primio od računara B jedan segment koji sadrži bajtove od 0 do 535 i drugi segment koji sadrži bajtove od 900 do 1 000. Iz nekog razloga računar A još nije primio bajtove 536 do 899. U ovom primeru, računar A i dalje čeka bajt 536 (i sledeće) da bi ponovo napravio tok podataka iz računara B. Prema tome, sledeći segment koji računar A šalje računaru B imaće u polju za broj potvrde vrednost 536. Pošto TCP potvrđuje samo bajtove do prvog nedostajućeg bajta u toku, za TCP se kaže da daje kumulativne potvrde.

U ovom poslednjem primeru pojavilo se jedno važno, ali suptilno pitanje. Računar A je primio treći segment (bajtove 900 do 1 000) pre nego što je primio drugi segment (bajtove 536 do 899). Prema tome, treći segment je stigao van redosleda. Suptilno pitanje glasi: Šta će računar uraditi kada u TCP konekciji primi segment van redosleda? Zanimljivo je da RFC dokumenti za TCP po ovom pitanju ne definišu nikakva pravila i prepustaju odluku programerima TCP implementacije. U osnovi postoje dve mogućnosti: ili će (1) primalac odmah odbaciti segmente primljene van redosleda (kao što smo rekli, to može da pojednostavi dizajn primaoca) ili će (2) primalac zadržati bajtove primljene van redosleda i Čekati da nedostajući bajtovi pristignu i popune praznine. Jasno je daje druga varijanta efikasnija što se tiče propusnog opsega mreže i zato je prihvaćena u praksi.



Slika 3.30 ♦ Deljenje podataka datoteke na TCP segmente

Na slici 3.30 prepostavili smo daje početni redni broj jednak 0. U praksi, obe sirane TCP konekcije biraju slučajni početni redni broj. To se radi da bi se na minimum svela verovatnoća da se segment iz neke ranije, već prekinute, konekcije između dva računara pogrešno ne shvati kao važeći segment u kasnijoj konekciji ta ista dva računara (gde se slučajno koriste isti brojevi portova kao u staroj konekciji) [Sunshine 1978].

Telnet: Kratak osvrt na redne brojeve i brojeve potvrda

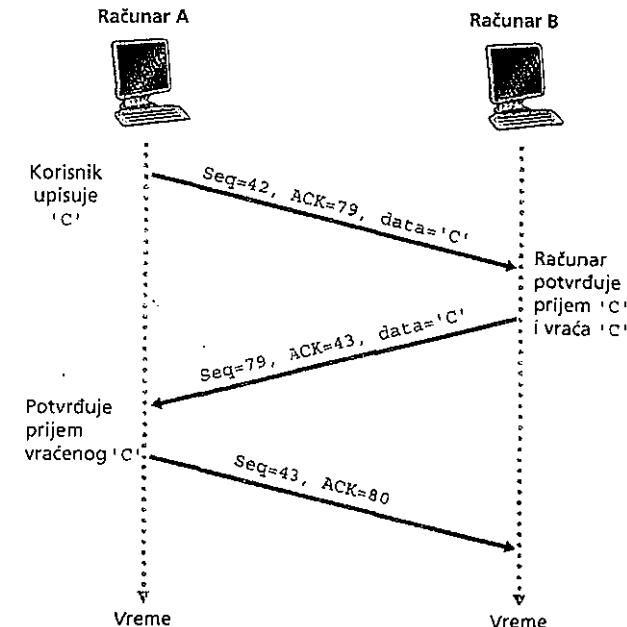
Telnet, definisan u RFC-u 854, je popularan protokol aplikacijskog sloja koji se koristi za daljinsko prijavljivanje. On se izvršava preko TCP-a i projektovan je za rad između bilo koja dva računara. Za razliku od aplikacija za paketski prenos podataka opisanih u poglavljju 2, Telnet je interaktivna aplikacija. Ovde ćemo opisati jedan primer Telneta, pošto on lepo ilustruje redne brojeve i brojeve potvrda u TCP-u. Napominjemo da mnogi korisnici sada radije koriste protokol ssh umesto Telneta, pošto podaci koji se šalju preko Telnet konekcije (uključujući i lozinke!) nisu šifro-vani, zbog čega je Telnet osetljiv na prisluškivanje (to ćemo opisati u odeljku 8.7).

Uzmimo da računar A pokrene Telnet sesiju sa računaram B. Pošto računar A pokreće sesiju on će se smatrati klijentom, a računar B serverom. Svaki znak koji korisnik otkuca (kod klijenta) poslaće se udaljenom računaru; udaljeni računar će vratiti kopiju znaka i ona će se prikazati na ekranu korisnika Telneta. Ovaj „vraćeni echo“ služi kao potvrda da su znakovi koje korisnik Telneta vidi već primljeni i obradeni na udaljenoj lokaciji. Svaki znak tako dva puta prelazi celu mrežu od trenutka kada korisnik pritisne taster do trenutka kada se znak pojavi na njegovom monitoru.

Uzmimo sada situaciju da korisnik upiše jedno jedino slovo, „C“, a zatim ode na kafu. Pogledajmo TCP segmente koji se šalju između klijenta i servera. Kao što se vidi na slici 3.31, mi prepostavljamo daje početni redni broj kod klijenta 42, a kod servera 79. Znate da redni broj u segmentu predstavlja redni broj prvog bajla u polju podataka. Prema tome, prvi segment koji klijent pošalje imaće redni broj 42; prvi segment koji pošalje server imaće redni broj 79. Znate daje broj potvrde redni broj sledećeg bajta podataka koji računar očekuje. Pošto se uspostavi TCP konekcija, a pre slanja podataka, klijent očekuje bajt 79, a server očekuje bajt 42.

Kao što je prikazano na slici 3.31, šalju se tri segmenta. Prvi segment se šalje od klijenta ka serveru, i u polju podataka sadrži jednobajtni ASCII prikaz slova „C“. Ovaj prvi segment sadrži još i vrednost 42 u polju rednog broja, kao što smo upravo opisali. Isto tako, pošto klijent još nije primio nikakve podatke od servera, ovaj prvi segment će u polju broja potvrde imati 79.

Drugi segment ide od servera ka klijentu. On ima dvostruku svrhu. Prvo, on služi kao potvrda podataka koje je server primio. Stavljanjem 43 u polje potvrde, server obaveštava klijenta da je uspešno primio sve do bajta 42 i da sada očekuje bajtove od 43 na dalje. Druga svrha ovog segmenta je da se vrati nazad slovo „C“. Prema tome, drugi segment ima u polju podataka ASCII prikaz slova „C“. Ovaj drugi segment nosi redni broj 79, početni redni broj toka podataka od servera ka



**Slika 3.31** ♦ Redni brojevi i brojevi potvrda u jednostavnoj Telnet aplikaciji preko TCP-a

klijentu u ovoj TCP konekciji, postoje u pitanju prvi bajt podataka koji šalje server. Obratite pažnju na to da se potvrda podataka koji su stigli od klijenta ka serveru šalje u istom segmentu koji prenosi podatke od servera ka klijentu; za takvu potvrdu kažemo da se šlepue na segmentu podataka od servera ka klijentu.

Treći segment se šalje od klijenta ka serveru. Njegova jedina svrha je da se potvrde podaci primljeni od servera. (Verovatno se sećate daje drugi segment sadržavao podatke ~ slovo „C“ - od servera ka klijentu.) Treći segment ima prazno polje podataka (tj. potvrda se ne šlepue ni na kakvim podacima od klijenta ka serveru). Segment sadrži 80 u polju broja potvrde zato što je klijent primio tok podataka sve do rednog broja 79 i sada očekuje bajtove od 80 na dalje. Možda vas čudi što ovaj segment takođe sadrži redni broj iako u njemu nema podataka, ali pošto TCP ima polje rednog broja u njemu mora da bude neki redni broj.

### 3.5.3 Procena vremena povratnog puta i tajm-aut

Uskoro ćemo videti da TCP, kao i naš protokol rdt iz odeljka 3.4, za oporavak od izgubljenih segmenata koristi mehanizam tajm-aut/ponovljeno slanje. Mada je koncepcija ovog mehanizma jednostavna, prilikom implementiranja u stvarnom protokolu kakav je TCP javljaju se mnoga suptilna pitanja. Možda je najočiglednije pitanje dužine intervala za tajm-aut. Jasno je da tajm-aut mora biti veći od vremena povratnog puta konekcije (RTT), tj. vremena od kad se segment pošalje dok ne stigne njegova potvrda. Inače bi dolazilo do nepotrebnih ponavljanja. Ali koliko veći treba da bude interval? Kako, pre svega, da se procent RTT? Da li bi svakom nepotvrđenom segmentu trebalo pridružiti zaseban tajmer? Toliko pitanja! Naš opis u ovom odeljku zasnovan je na tekstu o TCP-u u knjizi [Jacobson 1988] i važećim preporukama IETF-a za upravljanje TCP tajmerima [RFC 2988].

#### Procena vremena povratnog puta

Da bismo shvatili upravljanje TCP tajmerom najpre treba da proučimo način na koji TCP procenjuje vreme povratnog puta između pošiljaoca i primaoca. To se postiže na sledeći način. Uzorak vremena povratnog puta RTT za jedan segment, koji se naziva SampleRTT, jeste trajanje od trenutka slanja segmenta (tj. predavanja u IP) do prijema potvrde tog segmenta. Uglavnom se ne meri SampleRTT za svaki pre-neti segment. U većini TCP implementacija se u svakom trenutku pravi samo po jedan uzorak SampleRTT. To jest, u svakom trenutku se procenjuje samo jedan SampleRTT za jedan preneti ali trenutno još nepotvrđeni segment, tako da se dobija po jedna nova vrednost SampleRTT približno jedanput tokom svakog pojedinačnog povratnog puta. Osim toga, TCP nikad ne izračunava SampleRTT za segment koji se šalje ponovo; SampleRTT se meri samo za segmente koji su poslati jednom. (U jednom zadatku pri kraju poglavlja tražiće se da odgovorite zašto.) Očigledno je da će se vrednost SampleRTT menjati od segmenta do segmenta zavisno od zagušenja na ruterima i od različitog opterećenja krajnjih sistema. Zbog ovih razlika, svaka pojedinačna vrednost SampleRTT ne mora biti uobičajena. Da bi se procenio tipični RTT prirodno je da se uzme neki prosek dobijen od više vrednosti SampleRTT. TCP stalno izračunava jedan prosek izmerenih vrednosti SampleRTT po imenu EstimatedRTT. Čim dobije novi SampleRTT, TCP ažurira promenljivu EstimatedRTT prema sledećoj formuli:

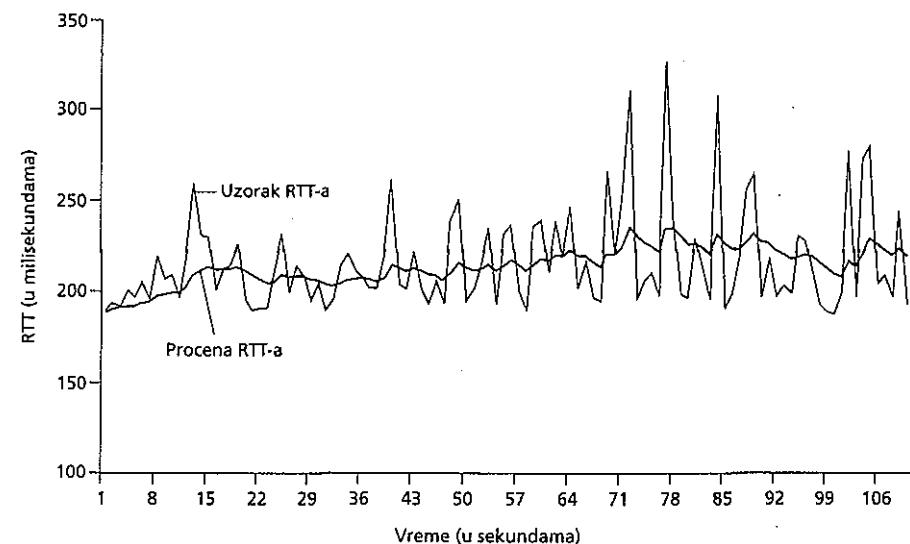
$$\text{EstimatedRTT} = (1 - a) \cdot \text{EstimatedRTT} + a \quad \blacksquare \quad \text{SampleRTT}$$

Gornja formula napisana je u obliku rekurzivne naredbe u programskom jeziku - nova vrednost za EstimatedRTT je ponderisana kombinacija prethodne vrednosti EstimatedRTT i nove vrednosti SampleRTT. Preporučena vrednost za  $a$  je 0,125 (tj. 1/8) [RFC 2988], pa u tom slučaju gornja formula postaje:

$$\text{EstimatedRTT} = 0.875 \cdot \text{EstimatedRTT} + 0.125 \cdot \text{SampleRTT}$$

Obratite pažnju na to da je EstimatedRTT ponderisani prosek vrednosti SampleRTT. Kao sto je opisano u domaćem zadatku na kraju poglavlja, ovaj ponderisani prosek daje veći značaj novijim uzorcima nego starijim. To je prirodno pošto noviji uzorci bolje odražavaju trenutno zagušenje mreže. U statistici se ovakav prosek naziva eksponencijalno ponderisani klizni prosek (*exponential weighted moving average, EWMA*). Ovaj prosek je „eksponencijalan“ zato što težinski faktor (ponder) pojedinačne vrednosti SampleRTT eksponencijalno opada sa svakim sledećim ažuriranjem. U domaćim zadacima će se tražiti da izvedete eksponencijalni izraz za EstimatedRTT.

Na slici 3.32 prikazane su vrednosti SampleRTT i EstimatedRTT za vrednost  $a = 1/8$  za jednu TCP konekciju između računara [gaia.cs.umass.edu](http://gaia.cs.umass.edu) (u gradu Amherst u Masacusetu) i računara [fantasia.eurecom.fr](http://fantasia.eurecom.fr) (na jugu Francuske). Jasno je da se varijacije u vrednosti SampleRTT izravnjavaju izračunavanjem vrednosti EstimatedRTT.



Slika 3.32 ♦ Uzorci i RTT procene

Osim procene trajanja povratnog puta, RTT, vredno je posedovati i kvantitativnu raeru varijacije RTT-a. [RFC 2988] definiše varijaciju vremena povratnog puta, DevRTT, kao procenu standardne devijacije SarapleRTT od EstimatedRTT:

$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot | \text{SampleRTT} - \text{EstimatedRTT} |$$

Obратite pažnju na to daje DevRTT eksponencijalno ponderisani klizni prošek razlike između vrednosti SampleRTT i EstimatedRTT. Ako vrednosti SampleRTT malo variraju, DevRTT će biti malo; a kada su ove fluktuacije veće, DevRTT će biti veliko. Preporučena vrednost za (3 je 0.25.

#### Upravljanje intervalom tajm-aut za ponovno slanje

Sada kada imamo vrednosti EstimatedRTT i DevRTT, koju bi vrednost trebalo uzeti kao TCP-ov interval za tajm-aut? Jasno je da bi interval trebalo da bude veći ili jednak vrednosti parametra EstimatedRTT, inače bi dolazilo do nepotrebnih ponovnih slanja. Ali, ne bi smeo da bude ni mnogo veći od parametra EstimatedRTT; inače, TCP ne bi dovoljno brzo ponovo poslao segment u slučaju

### PRINCIPI U PRAKSI

TCP obezbeđuje pouzdan transfer podataka pomoću tajmera i pozitivnih potvrda uglavnom no način koji smo opisali u odeljku 3.4. TCP potvrđuje podatke koji su primljeni pravilno, a ponovo šalje segmente kada se smatra da su segmenti ili njihove odgovarajuće potvrde izgubljene ili oštećene. Neke verzije TCP-a fakode sadrže i implicitni mehanizam NAK (negativna potvrda, prim. prev.), Tri ACK-a za isti segment igraju ulogu implicitnog NAK-a za sledeći segment, čime se pokreće ponovno slanje tog segmenta pre nego što nastupi lajm-aut. TCP koristi redne brojeve da bi primalač mogao da utvrdi do su segmenti izgubljeni ili dupli. Kao i u slučaju našeg protokola rdt3 zo pouzdani transfer podolako, TCP ne može sam pouzdano da utvrdi da li je segment ili njegov ACK izgubljen, oštećen ili on jednostavno mnogo kasni. TCP-ov odgovor će kod pošiljaoca biti isti: segment o kome je reč poslaće se ponovo.

TCP takođe koristi cevovodnu obradu, u kojoj pošiljalac može u istom trenutku da ima više poslatih, a još nepotvrđenih segmenata. Videli smo ranije da cevovodna obrada može značajno da unapredi propusnu moć sesije ako je veličina segmenta mala u odnosu na trajanje povratnog puta. Konkretan broj zaostalih nepotvrđenih segmenata koje pošiljalac može da drži utvrđuje se TCP-ovim mehanizmima za kontrolu toka i kontrolu zagruženja. TCP-ova kontrola toka opisana je no kraju ovog odeljka; TCP-ova kontrola zagruženja opisana je u odeljku 3.7. Za sada, dovoljno je da budemo svesni da TCP pošiljalac koristi cevovodnu obradu.

gubitka, čime bi se u aplikaciju uvelo značajno kašnjenje zbog transfera podataka. Zato je poželjno da interval za tajm-aut bude jednak EstimatedRTT plus neka rezerva. Rezerva bi trebalo da bude velika ako vrednosti parametra SampleRTT mnogo variraju, a mala u slučaju male varijacije. Zato ćemo ovde upotrebiti vrednost parametra DevRTT. Sva ova razmatranja uključena su u TCP-ov metod kojim se određuje interval tajm-auta za ponovno slanje:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$

### 3.5.4 Pouzdani transfer podataka

Verovatno znate da je Internetova usluga mrežnog sloja, EP, nepouzdana. IP ne garantuje isporuku datagrama, ne garantuje pravilan redosled datagrama i ne garantuje integritet podataka u datagramima. Kada se koristi usluga IP, datagrami mogu da prelju privremene memorije rutera i da nikada ne stignu na odredište; mogu da stignu izvan redosleda, a bitovi u datagramu mogu da se oštete (da se promene iz 0 u 1 i obratno). Pošto se segmenti transportnog sloja prenose preko mreže u IP datagramima, svi ti problemi mogu da se javi i u segmentima transportnog sloja.

TCP pruža uslugu pouzdanog transfera podataka preko IP-ove nepouzdane usluge najboljeg pokušaja. TCP-ova usluga pouzdanog transfera podataka obezbeđuje da tok podataka koji proces čita iz prijemne privremene memorije TCP-a bude neoštećen, bez prekida, bez duplikata i u pravilnom redosledu; tj. da tok bajtova bude tačno onakav kakav je poslao krajnji sistem na drugoj strani konekcije. U ovom pododeljku dajemo pregled načina na koji TCP obezbeđuje pouzdan transfer podataka. Videćemo da TCP-ova usluga pouzdanog transfera podataka koristi mnoge principe koje smo proučili u odeljku 3.4.

Kada smo razmatrali tehnike pouzdanog transfera podataka, bilo je najlakše prepostaviti da se svakom predatom, a još nepotvrđenom segmentu pridružuje zaseban tajmer. Mada je ovo teoretski odlično, upravljanje većim brojem tajmera predstavlja bi značajno programsko opterećenje. Zbog toga se u preporučenim procedurama za upravljanje tajmerima za TCP [RFC 2988] predlaže *samo jedan* tajmer za ponovno slanje, iako ima više poslatih, a još nepotvrđenih segmenata. Protokol TCP koji opisujemo u ovom odeljku zasnovan je na ovoj preporuci sa samo jednim tajmerom.

Način na koji TCP obezbeđuje pouzdan transfer podataka opisacemo u dva uzastopna koraka. Prvo predstavljamo krajnje pojednostavljeni opis TCP pošiljaoca gde se za oporavak od izgubljenih segmenata koristi samo tajmer; zatim dajemo potpuniji opis u kojem se osim tajmera koriste i duplikati potvrđivanja. U sledećem opisu, polazimo od prepostavke da se podaci šalju samo u jednom smeru, od računara A do računara B i da računar A šalje jednu veliku datoteku.

Na slici 3.33 prikazan je krajnji pojednostavljeni opis TCP pošiljaoca. Vidimo da u TCP pošiljaocu postoje tri glavna događaja u vezi sa inicijalnim i ponovnim sl-

njem podataka: stizanje podataka odozgo od aplikacije, tajm-aut i stizanje ACK-a. Kada nastupi prvi glavni događaj, TCP prima podatke od aplikacije, enkapsulira ih u segment i predaje segment IP-u. Obratite pažnju na to da svaki segment sadrži redni broj koji predstavlja broj prvog bajta podataka tog segmenta u toku bajtova, kao sto je opisano u odeljku 3.5.2. Takođe обратите pažnju na to da TCP pokreće tajmer kada preda segment u IP ako tajmer nije već uključen zbog nekog drugog segmenta.

```
/* Prepostavlja se da pošiljaoca ne ograničava TCP-ova kontrola toka niti kontrola zagušenja, da su podaci dobijeni od aplikacionog sloja manji od MSS-a i da se transfer podataka obavlja samo u jednom smeru */
```

```
NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber
```

```
loop (forever) (
 switch(event)
 event:podaci dobijeni odozgo od aplikacije
 create TCP segment with secuence number MextSeqNum if (timer currently not
running)
 start timer pass segment to IP
 NextSeqNum=NextSeqNum+length(data) break; event: timer
 timeout
 retransmit not-yet-acknowledged segment with
 smallest sequence number start timer
 break;
 event: primljen ACK, sa vrednošću y u polju ACK if (y > SendBase) {
 SendBase=y
 if (there are currently any not-yet-ackowl-edged segments) start timer
 1
 break;
 }
 } /* kraj beskonačne petlje */ Slika 3.33
```

Pojednostavljen TCP pošiljalac

(Može se zamisliti daje tajmer pridružen najstarijem nepotvrđenom segmentu.) Interval ovog tajmera je TimeoutInterval, koji se izračunava na osnovu vrednosti parametara EstimatedRTT i DevRTT, kao sto je opisano u odeljku 3.5.3.

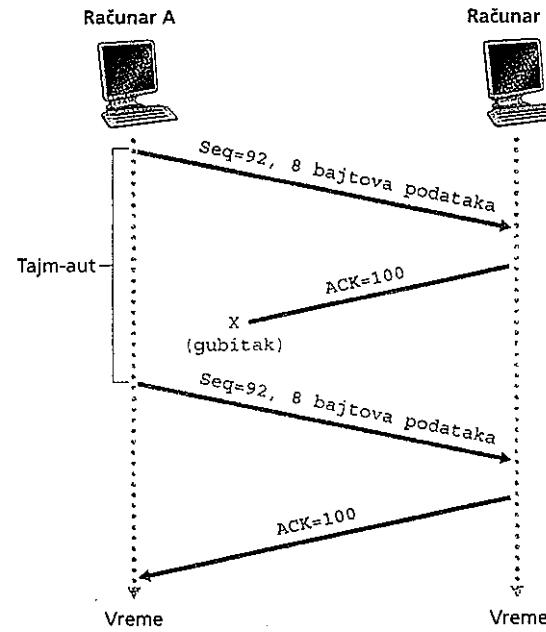
Dруги glavni događaj je tajm-aut. TCP reaguje na događaj tajm-aut tako što ponovo šalje segment koji gaje izazvao. TCP zatim ponovo pokreće tajmer. Treći važan događaj koji TCP pošiljalac mora da obradi je pristizanje segmenta potvrde (ACK) od primaoca (tačnije, segment koji sadrži važeću vrednost u polju ACK). Kada nastupi ovaj događaj, TCP poređi ACK vrednost y sa svojom promenljivom SendBase. Promenljiva TCP stanja SendBase je redni broj najstarijeg nepotvrđenog bajta. (Prema tome, SendBase-1 je redni broj poslednjeg bajta za koji se zna daje kod primaoca primljen pravilno i u ispravnom redosledu.) Kao što je ranije napomenuto, TCP koristi kumulativne potvrde, pa tako y potvrđuje prijem svih bajtova pre bajta broj y. Ako je y > .SendBase, taj ACK potvrđuje jedan ili više prethodno nepotvrđenih segmenata. Na taj način pošiljalac ažurira promenljivu SendBase; on takođe ponovo pokreće tajmer ako ima nekih još nepotvrđenih segmenata.

#### Nekoliko zanimljivih scenarija

Upravo smo opisali krajnje pojednostavljenu verziju načina na koji TCP obezbeđuje pouzdani transfer podataka. Ali, ta krajnje pojednostavljena verzija - iako uprošćena - ipak sadrži mnoge suptilnosti. Da biste bolje osetili kako ovaj protokol funkcioniše, pogledajmo nekoliko jednostavnih scenarija. Na slici 3.34 prikazan je prvi scenario u kojem računar A šalje jedan segment računaru B'. Uzmimo da segment ima redni broj 92 i da sadrži osam bajtova podataka. Pošto pošalje ovaj segment, računar A čeka od računara B segment sa brojem potvrde 100. Mada je segment iz A stigao u računar B, potvrda od B prema A se gubi. U ovom slučaju, nastupa događaj tajm-aut i računar A ponovo šalje isti segment. Naravno, kada računar B ponovo primi isti segment, on će iz rednog broja zaključiti da taj segment sadrži podatke koji su već primljeni. Prema tome, TCP u računaru B će odbaciti bajtove iz ponovo poslatog segmenta,

U drugom scenarioju, prikazanom na slici 3.35, računar A šalje dva segmenta jedan za drugim. Prvi segment ima redni broj 92 i osam bajtova podataka, a drugi segment ima redni broj 100 i 20 bajtova podataka. Uzmimo da oba segmenta stignu neoštetećena do računara B i da B šalje dve zasebne potvrde, za svaki od tih segmenata. Prvi broj potvrde je 100 a drugi 120. Uzmimo sada da nijedna od ovih potvrda ne stigne do računara A pre isteka tajm-autu. Kada nastupi događaj tajm-aut, računar A ponovo šalje prvi segment sa rednim brojem 92 i ponovo pokreće tajmer. Ako ACK za drugi segment stigne pre novog tajm-autu, drugi segment se neće ponovo poslati.

U trećem i konačnom scenarioju, uzmimo da računar A pošalje dva segmenta, isto kao u drugom primeru. Potvrda prvog segmenta se gubi u mreži, ali neposredno

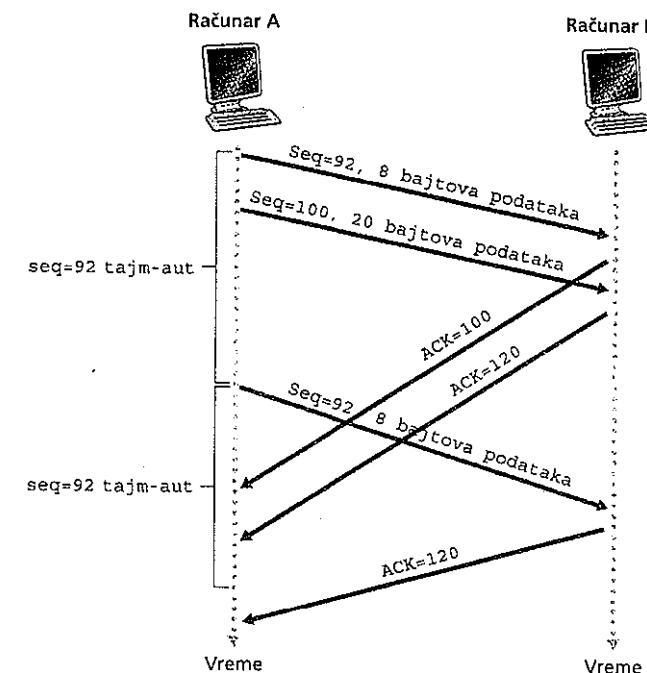


**Slika 3.34** ♦ Ponovno slanje zbog izgubljene potvrde

pre događaja tajm-aut, računar A prima potvrdu sa brojem 120. Računar A prema tome zna daje računar B primio sve zaključno sa bajtom 119, pa ne šalje ponovo nijedan od prva dva segmenta. Ovaj scenario prikazan je na slici 3.36.

#### Duplikiranje intervala za tajm-aut

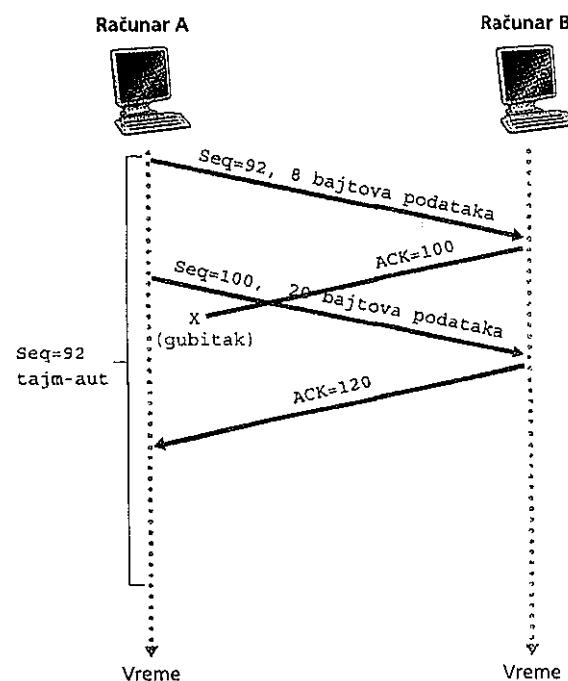
Sada ćemo opisati nekoliko modifikacija koje se koriste u većini TCP implementacija. Prva se odnosi na dužinu intervala za tajm-aut kada tajmeru istekne važnost. Kao što je gore opisano, kad god nastupi događaj tajm-aut, TCP ponovo šalje nepotvrđeni segment sa najmanjim rednim brojem. AH, svaki put kada TCP ponovo šalje neki segment, on udvostručava interval za tajm-aut u odnosu na prethodni umesto da ga ponovo izračuna od ranijeg EstimatedRTT i DevRTT (kao sto je opisano u odeljku 3.5.3). Na primer, uzimimo da TimeoutInterval pridružen najstarijem još nepotvrđenom segmentu iznosi 0,75 sekunde kada tajmeru prvi put istekne vreme. TCP će tada ponovo poslati ovaj segment i odrediti novi interval tajmera od 1,5 sekundi. Ako tajmer opet istekne nakon 1,5 sekundi, TCP će opet ponoviti slanje



**Slika 3.35** ♦ Segment 100 se ne šalje ponovo

tog segmenta, ali će sada odrediti da interval bude 3,0 sekunde. Na ovaj način, interval nakon svakog ponovnog slanja raste eksponencijalno. Međutim, ako se tajmer ponovo pokreće nakon bilo kojeg od druga dva događaja (tj. ako su podaci primljeni odozgo iz aplikacije ili je primljen ACK), vrednost TimeoutInterval izračunava se od najnovijih vrednosti parametara EstimatedRTT i DevRTT.

Ova modifikacija obezbeđuje ograničeni oblik kontrole zagušenja. (Potpunije oblike TCP kontrole zagušenja proučićemo u odeljku 3.7.) Do isteka tajmera došlo je najverovatnije zbog zagušenja na mreži, tj. prevelikog broja paketa koji stižu na jedan (ili više) redova za Čekanje u ruterima na putanji od izvora do odredišta, zbog čega se paketi odbacuju ili dugi čekaju u redovima. Ako tokom zagušenja izvori uporno nastave da ponavljaju slanje paketa, zagušenje može da postane još gore. Umesto toga, TCP postupa uviđavnije, tako što svaki pošiljalac ponavlja slanje u sve dužim i dužim intervalima. Kada budemo razmatrali CSMA/CD u poglavljju 5, videćemo da Ethernet koristi sličan pristup.



**Slika 3.36** ♦ Kumulativnom potvrdom se izbegava ponovno slanje prvog segmenta.

#### Brzo ponovno slanje

Jedan od problema ponovnog slanja koje se pokreće nakon tajm-aut-a je u tome stoji tajm-aut interval često prilično velik. Kada se segment izgubi, pošiljalac mora dugo da čeka pre nego što ponovo pošalje paket, pa tako produžava ukupno kašnjenje s kraja na kraj. Srećom, pošiljalac često može da uoči gubitak paketa mnogo pre isteka vremena tajmera kada primi takozvane duple ACK-ove. Dupli ACK je ACK koji ponovo potvrđuje segment za koji je pošiljalac već ranije primio potvrdu. Da bismo shvatili reagovanje pošiljaoca na dupli ACK, moramo da razmotrimo zašto primalac uopšte šalje dupli ACK. U tabeli 3.2 data je rekapitulacija politike po kojoj TCP primalac pravi ACK [RFC 1122, RFC 2581]. Kada TCP primalac dobije segment sa rednim brojem većim od sledećeg očekivanog rednog broja, on primeće prazninu u toku podataka - tj. primeće da neki segment nedostaje. Ova praznina može da potiče od toga što su segmenti izgubljeni u mreži ili od toga što im se poremetio redosled. Primalac ne može da vrati pošiljaocu eksplisitnu negativnu potvrdu zato

| Dogadjaj                                                                                                                                       | Postupak TCP primalaca                                                                                                                                                 |
|------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dolazak segmenta u ispravnom redosledu sa očekivanim rednim brojem. Svi prethodni podaci već su potvrđeni.                                     | Odgoditi ACK. Čekati do 500 ms na pristizanje još jednog segmenta u ispravnom redosledu. Ako za to vreme ne stigne sledeći segment u pravilnom redosledu, poslati ACK. |
| Pristizanje segmenta u ispravnom redosledu sa očekivanim rednim brojem. Jedan segment u ispravnom redosledu već čeka da se pošalje njegov ACK. | Odmah poslati kumulativni ACK čime se potvrđuju oba segmenta u ispravnom redosledu.                                                                                    |
| Pristizanje segmenta van redosleda, sa rednim brojem većim od očekivanog. Otkriven razmak.                                                     | Odmah poslati duplikat ACK-a sa rednim brojem sledećeg očekivanog bajta (tj. sa donjeg kraja razmaka).                                                                 |
| Pristizanje segmenta koji delimično ili potpuno popunjava razmak u primljenim podacima.                                                        | Odmah poslati ACK ako taj segment počinje na donjem kraju razmaka.                                                                                                     |

**Tabela 3.2** ♦ Preporuka TCP-a za pravljenje ACK-a [RFC 1122, RFC 2581].

što TCP ne koristi negativne potvrde. Umesto toga, on jednostavno ponovo potvrđuje (tj. pravi dupli ACK) za poslednji bajt podataka koji je primio u pravilnom redosledu. (Obratite pažnju na to daje u tabeli 3.2 prikazan slučaj kada primalac ne odbacuje segmente van redosleda.)

Pošto pošiljalac često šalje više segmenata jedan za drugim, kada se jedan segment izgubi verovatno je da će biti mnogo uzastopnih istovetnih ACK-ova. Ako TCP pošiljalac primi tri ACK-a za iste podatke, on to prihvata kao signal da se izgubio segment iza onog koji je tri puta potvrđen. (U problemima za domaći zadatak razmatramo pitanje zašto pošiljalac čeka da dobije tri istovetna ACK-a i ne reaguje na prvi duplirani ACK.) U slučaju da primi tri istovetna ACK-a, TCP preuzima brzo ponovno slanje [RFC 2581], tako što šalje nedostajući segment *pre* nego što istekne njegov tajmer. U TCP-u sa brzim ponovnim slanjem, sledeći odlomak koda zamenjuje dogadaj primljenog ACK-a sa slike 3.33:

```

event: primljen ACK, sa vrednošću y u polju ACK if (y > SendBase) {
 SendBase=y
 if (there are currently any not yet _acknowledged segments) start
 timer
 }
 else { /* dupli ACK za već potvrđeni segment *7 increment number of duplicate ACKs
 -
 received for y if (number of duplicate ACKS received =
 for y==3) {
 /* TCP brzo ponovno slanje */
 resend segment with sequence number y
 }
 break;
 }
}

```

Već smo primetili da se javlja mnogo suptilnih pitanja kada se u stvarnom protokolu kakav je TCP implementira mehanizam za tajm-aut i ponovno slanje. Gore pomenuti postupci koji su nastali kao rezultat više od 15 godina iskustva sa TCP-ovini tajmerima trebalo bi da ubede Čitaoca daje zaista tako!

#### GBN ili selektivno ponavljanje?

Naše proučavanje TCP-ovog mehanizma za oporavak od grešaka zaključićemo sledećim pitanjem: da li je TCP protokol tipa GBN ili SR? Znale da su TCP-ove potvrde kumulativne i da primalac ne potvrđuje pojedinačno segmente koji su primljeni ispravno, ali izvan redosleda. Zato, kao sto je prikazano na slici 3.33 (pogledajte takođe sliku 3.19), TCP pošiljalac mora da čuva samo najniži redni broj poslatog a nepotvrđenog bajta (SendBase) i redni broj sledećeg bajta koji treba da se pošalje (NextSeqNum). U tom pogledu, TCP dosta liči na GBN protokol, ali ipak postoje i velike razlike. U mnogim implementacijama TCP-a pravilno primljeni segmenti izvan redosleda ipak se čuvaju [Stevens 1994]. Uzmimo takođe situaciju kada pošiljalac pošalje niz segmenata 1, 2, N i svi oni stignu primaocu ispravni i u pravilnom redosledu. Prepostavite da se zatim izgubi potvrda za paket < A', a da preostalih N - 1 potvrda stigne pošiljaocu pre tajm-autu. U takvom slučaju, GBN protokol bi ponovo poslao ne samo paket n, već i sve naredne pakete n + 1, n + 2, N. S druge strane, TCP bi ponovo poslao najviše jedan segment, segment <. Štaviše, TCP ne bi poslao čak ni segment n ako bi potvrda za segment n + 1 stigla pre tajm-autu za segment n.

Predložena modifikacija protokola TCP, takođe selektivno potvrđivanje [RFC 2581], omogućava TCP primaocu da selektivno potvrđuje segmente primljene izvan redosleda umesto da kumulativno potvrđuje poslednji pravilan segment primljen u ispravnom redosledu. Kada se kombinuje sa selektivnim ponovnim slanjem

- gde se izostavlja ponovno slanje segmenata koje je primalac selektivno potvrdio
- TCP veoma liči na naš opšti protokol tipa SR. Prema tome, TCP-ov mehanizam za oporavak od grešaka bi najbolje bilo razvrstati kao mešavinu GBN protokola i protokola sa selektivnim ponavljanjem.

#### 3.5.5 Kontrola toka

Verovatno se sećate da računari na svakoj strani TCP konekcije rezervisti prijemne privremene memorije za tu konekciju. Kada TCP konekcija primi bajtove koji su ispravni i u pravilnom redosledu, ona ih stavlja u prijemnu privremenu memoriju. Pridruženi aplikacioni proces će čitati podatke iz ove privremene memorije, ali to ne mora biti čim podaci stignu. Zaista, prijemna aplikacija može da bude zauzeta nekim drugim zadatkom pa stoga i ne pokuša da čita podatke odmah po njihovom pristizanju. Ako aplikacija relativno sporo čita a pošiljalac suviše brzo šalje preveliku količinu podataka, prijemna privremena memorija konekcije može vrlo iako da se preplavi.

Da bi se eliminisala mogućnost da pošiljalac preplavi primaočevu privremenu memoriju, TCP svojim aplikacijama obezbeđuje uslugu kontrole toka. Kontrola toka je prema tome usluga uskladivanja brzine - jer uskladjuje brzinu kojom pošiljalac šalje sa brzinom kojom prijemna aplikacija čita. Kao što je već napomenuto, TCP pošiljalac može se usporiti i zbog zagušenja unutar IP mreže; ta vrsta kontrole naziva se kontrola zagušenja. Nju ćemo detaljno obraditi u odeljcima 3.6 i 3.7. Mada su aktivnosti koje preduzimaju kontrola toka i kontrola zagušenja slične (usporavanje pošiljaoca), one se očigledno preduzimaju iz veoma različitih razloga. Nažalost, mnogi autori mešaju ove izraze, pa bi mudar čitalac trebalo pažljivo da ih razluči. Sada ćemo opisati kako TCP pruža uslugu kontrole toka. Da ne bi drveće zaklonilo šumu, u celom ovom odeljku polazimo od pretpostavke da je TCP implementacija takva da TCP primalac odbacuje segmente koji nisu u pravilnom redosledu.

TCP omogućava kontrolu toka tako što se kod *pošiljaoca* održava promenljiva zvana prijemni prozor. Nezvanično objašnjenje je da se prijemni prozor koristi kako bi pošiljalac mogao da naslutи koliko ima mesta u privremenoj memoriji primaoca. Pošto TCP radi u punom dupleksu, pošiljalac na svakoj strani konekcije održava zaseban prijemni prozor. Razmotrimo prijemni prozor u kontekstu transfera datoteke. Uzmimo da računar A šalje računaru B veliku datoteku preko TCP konekcije. Računar B dodeljuje toj konekciji prijemnu privremenu memoriju; označite njenu veličinu sa *RcvBuf*. S vremenom na vreme, aplikacioni proces u računaru B čita iz ove privremene memorije. Definišite sledeće promenljive:

- ◆ *LastByteRead*: broj poslednjeg bajta u toku podataka koji je aplikacioni proces u računaru B pročitao iz privremene memorije.
- ◆ *LastByteRcvd*: broj poslednjeg bajta u toku podataka koji je stigao sa mreže i štavljenje u prijemnu privremenu memoriju računara B.

Pošto TCP ne srne da prelije dodeljenu privremenu memoriju, uvek mora biti:

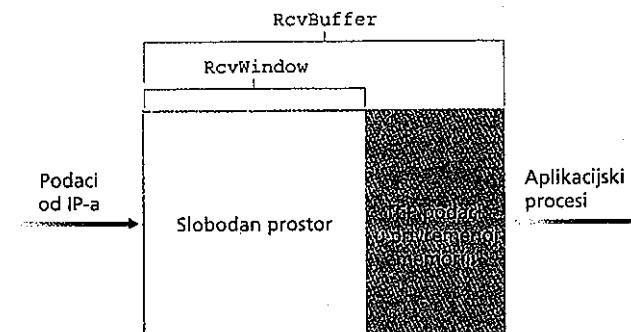
$$\text{LastByteRcvd} - \text{LastByteRead} < \text{RcvBuffer}$$

Prijemnom prozoru, označenom sa *RcvWindow*, dodeljuje se preostala količina prostora u privremenoj memoriji:

$$\text{RcvWindow} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

Pošto se preostali prostor menja u vremenu, *RcvWindow* je dinamičan. Promenljiva *RcvWindow* prikazana je na slici 3.37.

Kako konekcija koristi promenljivu *RcvWindow* u pružanju usluge kontrole toka? Računar B obaveštava računar A o slobodnom prostoru u privremenoj memoriji konekcije tako što stavlja trenutnu vrednost *RcvWindow* u polje prijemnog prozora svakog segmenta koji šalje računaru A. Na početku, računar B stavlja



**Slika 3.37** ♦ Prijemni prozor (RcvWindow) i prijemna privremena memorija (RcvBuffer)

RcvWindow = RcvBuf fer. Obratite pažnju na to da za uspeh ovakvog postupka računar B mora da prati nekoliko promenljivih vezanih za konekciju.

Računar A održava dve promenljive, LastByteSent (poslednji poslati bajt) i LastByteAcked (poslednji potvrđeni bajt). Obratite pažnju na to da je razlika između te dve promenljive, LastByteSent - LastByteAcked, jednaka količini nepotvrđenih podataka koje je A poslao u konekciju. Ako pazi da količina nepotvrđenih podataka ostane manja od vrednosti RcvWindow, računar A može biti siguran da ne preplavljuje prijemnu privremenu memoriju računara B. Prema tome, računar A tokom celog života konekcije obezbeđuje da bude

LastByteSent - LastByteAcked ^ RcvWindow

Kod ovog rešenja postoji jedan mali tehnički problem. Da biste ga sagledali, pretpostavimo da se prijemna privremena memorija računara B popuni tako da bude RcvWindow = 0. Posto se objavi računaru A da je RcvWindow = 0, pretpostavimo takođe da računar B nema više *ništa* da šalje računaru A. Pogledajmo sada šta se dogada. Pošto aplikacioni proces u B isprazni privremenu memoriju, TCP neće poslati računaru A novi segment sa novom vrednošću RcvWindow pošto on to radi samo ako ima podatke za slanje ili ako treba da pošalje neku potvrdu. Prema tome, računar A nikada ne bi saznao da se pojavio prostor u prijemnoj privremenoj memoriji računara B - računar A bi bio blokiran i ne bi mogao više da šalje podatke! Da bi se resio ovaj problem TCP specifikacija zahteva da ako je prijemni prozor računara B jednak nuli, računar A nastavi da šalje segmente sa po jednim bajtom podataka. Te segmente će primalač potvrditi. U jednom trenutku, privremena memorija će početi da se prazni, pa će potvrde preneti računaru A vrednost RcvWindow različitu od nule.

Onlajn lokacija na adresi <http://www.awl.com/kurose-ross> sadrži interaktivni Java aplet koji ilustruje rad prijemnog prozora TCP-a.

Pošto smo opisali TCP -ovu uslugu kontrole toka, ukratko ovde pomjenjemo da UDP ne obezbeđuje kontrolu toka. Da bi se shvatio problem, razmotrimo slanje niza UDP segmenata od jednog procesa u računaru A drugom procesu u računaru B. U uobičajenoj implementaciji UDP-a, UDP dodaje segmente u privremenu memoriju konačne veličine pre upotrebe odgovarajućeg soketa (tj. vrata prema procesu). Proces čita iz privremene memorije cele segmente, jedan po jedan. Ako proces ne čita segmente dovoljno brzo, privremena memorija će se prepliti i segmenti će biti izgubljeni.

### 3.5.6 Upravljanje TCP konekcijom

U ovom pododeljku preciznije ćemo razmotriti kako se TCP konekcija uspostavlja i raskida. Mada ova tema ne izgleda posebno uzbudljiva, ona je važna pošto uspostavljanje TCP konekcije može značajno da utiče na opaženo kašnjenje (na primer, prilikom šetanja po Vebu). Pogledajmo sada kako se uspostavlja TCP konekcija. Uzmimo da proces koji se izvršava najednom računaru (klijent) želi da pokrene konekciju sa drugim procesom na drugom računaru (server). Klijentski aplikacioni proces prvo obaveštava klijentski TCP da želi da uspostavi konekciju sa procesom na serveru. TCP u klijentu zatim kreće u uspostavljanje TCP konekcije sa TCP-om u serveru na sledeći način:

- ♦ Korak 1. TCP na klijentskoj strani prvo šalje poseban TCP segment TCP-u na serverskoj strani. Ovaj poseban segment ne sadrži podatke aplikacionog sloja, ali jedan od bitova oznaka u zaglavju segmenta (slika 3.29), bit SYN, ima vrednost 1. Zbog toga se ovaj poseban segment naziva segmentom SYN. Osim toga, klijent bira početni redni broj (client\_isn) i stavlja taj broj u polje rednog broja početnog TCP segmenta SYN. Ovaj segment se enkapsulira u IP datagram i šalje serveru. Pravilnom izboru slučajne vrednosti client\_isn posvećena je  
značajna pažnja da bi se izbegli neki bezbednosni rizici [CERT 2001-09].
- ♦ Korak 2. Kada IP datagram koji sadrži TCP segment SYN stigne serverskom računaru (ako uopšte stigne!), server vadi TCP segment SYN iz datagrama, dodeljuje TCP konekciji privremene memorije i promenljive i šalje klijentskom TCP-u segment odobrenja konekcije. Ovaj segment odobrenja konekcije takođe ne sadrži podatke aplikacionog sloja. Međutim, on u zaglavju sadrži tri značajne informacije. Prvo, bit SYN ima vrednost 1. Drugo, polje potvrde u zaglavju TCP segmenta ima vrednost client\_isn+1. Na kraju, server bira i vlastiti početni redni broj (server\_isn) i stavlja tu vrednost u polje rednog broja u zaglavju TCP segmenta. Ovaj segment odobrenja konekcije u suštini saopštava: „Primio sam vaš paket SYN za pokretanje konekcije sa vašim početnim rednim brojem

`client_isn`. Pristajem na uspostavljanje ove konekcije. Moj početni redni broj je `server_isn`."

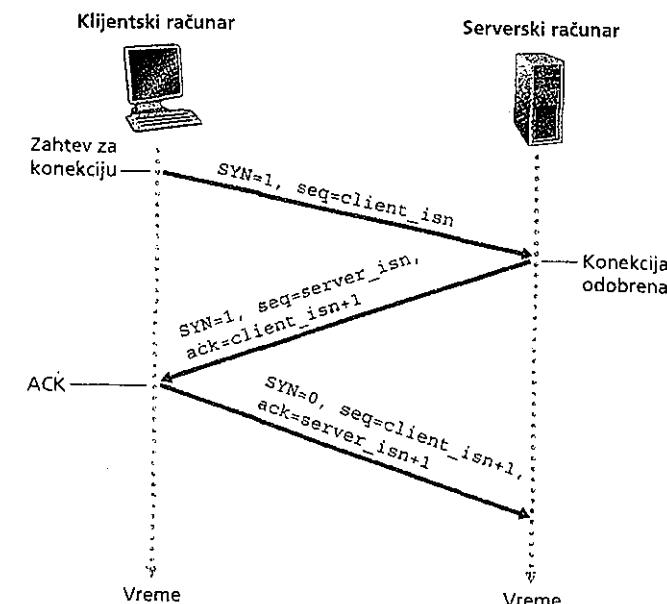
Segment odobrenja konekcije ponekad se naziva i segment SYNACK.

- ♦ Korak 3. Pošto primi segment SYNACK, klijent takođe dodeljuje konekciji privremene memorije i promenljive. Klijentski računar tada šalje serveru još jedan segment; ovaj poslednji segment potvrđuje serverov segment odobrenja konekcije (klijent to postiže stavljanjem vrednosti `server_isn+1` u polje potvrde u zaglavju TCP segmenta). Bit SYN ima vrednost 0, postoje konekcija uspostavljena.

Pošto se izvrše prethodna tri koraka, klijentski i serverski računar mogu jedan drugom da šalju segmente koji sadrže podatke. U svakom od ovih budućih segmentata, bit SYN imaće vrednost nula. Obratite pažnju na to da se za uspostavljanje konekcije između dva računara šalju tri paketa, kao što je prikazano na slici 3.38. Zbog toga se ova procedura uspostavljanja konekcije često naziva i sinhronizacija u tri koraka. U problemima za domaći zadatak istražuje se nekoliko aspekata TCP-ove sinhronizacije u tri koraka (Zašto su potrebni početni redni brojevi? Zašto je potrebna sinhronizacija u tri koraka, a ne u dva?). Zanimljivo je da alpinista i čuvar (koji stoji ispod alpiniste i čiji je posao da rukuje bezbednosnim uzetom alpiniste), da bi bili sigurni da su obojica spremni pre nego što alpinista počne da se penje, koriste komunikacioni protokol sa sinhronizacijom u tri koraka identičan TCP-u.

Sve dobre stvari se jednom završe, pa isto važi i za TCP konekciju. Bilo koji od dva procesa koji učestvuju u TCP konekciji može da prekine konekciju. Kada se konekcija završi, „resursi“ (tj. privremene memorije i promenljive) u računarima se oslobadaju. Na primer, pretpostavimo da klijent odluči da zatvori konekciju kao sto je prikazano na slici 3.39. Klijentski aplikacioni proces izdaje komandu zatvaranja. Tada klijentski TCP šalje serverskom procesu poseban TCP segment. Ovaj poseban segment sadrži vrednost 1 u bitu označe FIN u zaglavju segmenta (slika 3.39). Kada server primi ovaj segment, on vraća klijentu segment potvrde. Server zatim šalje vlastiti segment prekida u kome bit FIN ima vrednost 1. Na kraju, klijent potvrđuje serverov segment prekida. U tom momentu su svi resursi na oba računara oslobodeni.

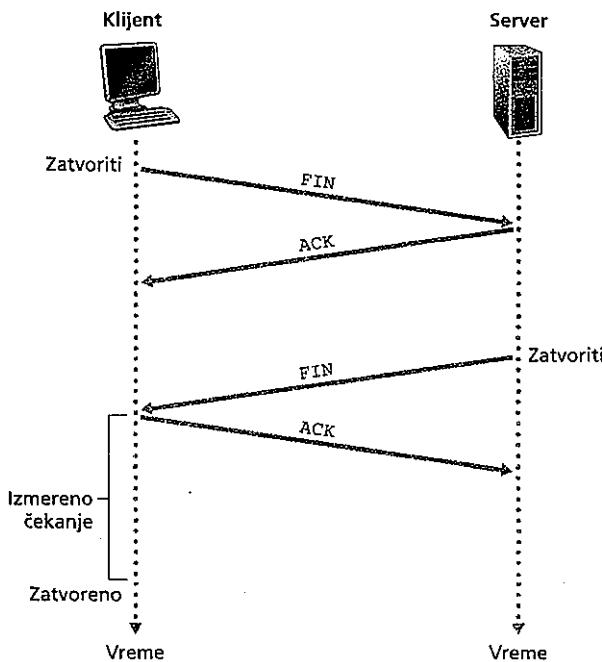
Tokom života TCP konekcije, protokol TCP koji se izvršava na svakom računaru prolazi kroz različita TCP stanja. Na slici 3.40 prikazanje uobičajen niz TCP stanja kroz koja prolazi TCP *klijent*. Klijentski TCP na početku se nalazi u stanju CLOSED. Aplikacija na klijentskoj strani inicira novu TCP konekciju (pravljenjem objekta Socket u našim Java primerima iz poglavlja 2). Zbog toga TCP u klijentu šalje TCP-u na serveru segment SYN. Pošto pošalje segment SYN, klijentski TCP prelazi u stanje SYN\_SENT. Dok je u stanju SYN\_SENT, klijentski TCP očekuje od serverskog TCP-a segment koji sadrži potvrdu za klijentov prethodni segment i u bitu SYN sadrži vrednost 1. Pošto primi takav segment, klijentski TCP prelazi u stanje



**Slika 3.38** ♦ TCP-ova sinhronizacija u tri koraka: razmena segmenata

nje ESTABUSHED. Dok je u stanju ESTABLISHED, klijentski TCP može da šalje i prima TCP segmente koji sadrže korisne podatke (tj. podatke iz aplikacije).

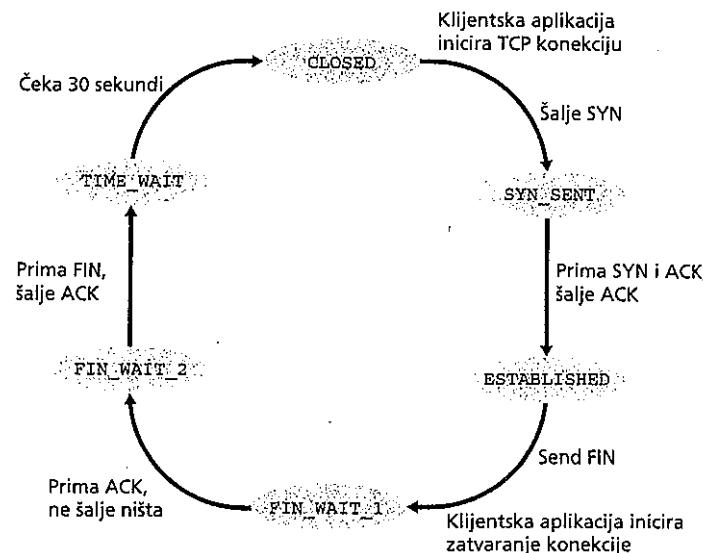
Uzmimo da klijentska aplikacija resi da zatvori konekciju (mada i server može da odluci da prekine konekciju). Tada klijentski TCP šalje TCP segment sa bitom FIN jednakim 1 i prelazi u stanje FIN\_WAIT\_1. Dok je u stanju FIN\_WAIT\_1, klijentski TCP očekuje od servera TCP segment sa potvrdom. Kada primi taj segment, klijentski TCP prelazi u stanje FIN\_WAIT\_2. Dok je u stanju FIN\_WAIT\_2, klijent čeka na drugi segment sa servera gde je bit FIN jednak 1; kada primi taj segment, klijentski TCP potvrđuje serverov segment i prelazi u stanje TIME\_WAIT. Stanje TIME\_WAIT omogućava TCP klijentu da ponovo pošalje konačnu potvrdu u slučaju da se ACK izgubi. Vreme provedeno u stanju TIME\_WAIT zavisi od implementacije, ali uobičajene vrednosti su 30 sekundi, 1 minut ili 2 minute. Nakon čekanja, konekcija se zvanično zatvara i svi resursi na klijentskoj strani (uključujući i brojove portova) se oslobadaju.



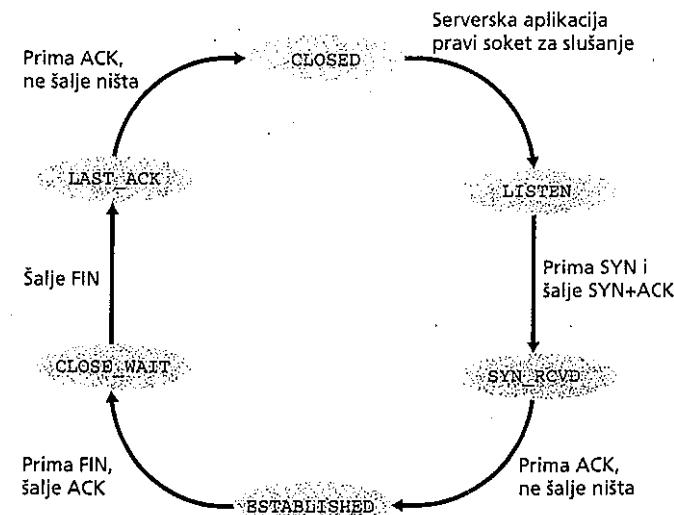
**Slika 3.39 ♦ Zatvaranje TCP konekcije**

Na slici 3.41 prikazan je niz stanja kroz koja prolazi TCP na serverskoj strani, pod pretpostavkom da klijent pokrene postupak za prekid konekcije. Prelasci iz stanja u stanje su očigledni. Na ova dva dijagrama prikazali smo samo kako se TCP konekcija normalno uspostavlja i prekida. Nismo opisali šta se događa u nekim neuobičajenim situacijama, na primer, kada obe strane jedne konekcije u isto vreme zahtevaju prekid. Ako vas zanima da proučite to i neka druga napredna pitanja u vezi sa TCP-om, predlažemo da pročitate Stivensovou sveobuhvatnu knjigu [Stevens 1994].

Pri nego što zaključimo ovaj odeljak, razmotrimo šta se događa kada računar primi prvi TCP segment. Čiji se broj porta ili izvorna IP adresa ne uklapaju ni sa jednim postojećim soketom u računaru. Na primer, prepostavite da računar primi TCP SYN paket sa određenim portom 80 ali da računar ne prihvata konekcije na portu 80 (tj. na portu 80 se ne izvršava nijedan web server). Računar će tada poslati izvoru poseban segment za resetovanje. Ovaj TCP segment sadrži postavljen bit RST (pročitajte odeljak 3.5.2). Kada računar pošalje takav segment, on poručuje izvoru „Ja nemam soket za taj segment. Molim nemojte ponavljati slanje segmenta.“ Kada računar primi UDP paket čiji se broj određenog porta ne uklapa ni sa jednim od aktivnih UDP soketa, poslaće poseban ICMP datagram kao što je opisano u poglavljiju 4.



**Slika 3.40 ♦ Uobičajeni niz TCP stanja kroz koja prolazi klijentski TCP**



**Slika 3.41 ♦ Uobičajeni niz TCP stanja kroz koja prolazi TCP na serverskoj strani**

Ovim zaključujemo uvod u kontrolu grešaka i kontrolu toka u TCP-u. U odeljku 3.7 vratimo se na TCP i donekle detaljno razmotriti njegovu kontrolu zagušenja. Pre toga se, međutim, vraćamo na opšti kontekst gde razmatramo pitanja kontrole zagušenja.

## 3.6 Principi kontrole zagušenja

U prethodnim odeljcima ispitali smo opšte principe i konkretnе mehanizme koje TCP koristi da bi obezbedio uslugu pouzdanog transfera podataka u situaciji kada se paketi gube. Već smo pomenuli da u praksi do tih gubitaka obično dolazi zbog prelivanja privremene memorije u ruterima kada je mreža zagušena. Ponovno slanje paketa leci simptom zagušene mreže (gubitak konkretnog segmenta u transportnom sloju) ali ne leći uzrok zagušenja mreže - previše izvora koji šalju podatke prevelikom brzinom. Da bi se uticalo na *uzrok* zagušenja potrebni su mehanizmi koji će da uspore pošiljaoce kada preti zagušenje mreže.

U ovom odeljku razmotrićemo problem kontrole zagušenja u opštem smislu. Pokušaćemo da shvatimo zašto je zagušenje „loše“, kako ono utiče na performanse aplikacija gornjeg sloja, kao i da proučimo različite pristupe izbegavanju zagušenja i reagovanju na zagušenje mreže. Ovo opšte izlaganje o kontroli zagušenja je važno jer predstavlja značajnu stavku na listi „prvih deset“ najznačajnijih problema umrežavanja ako želimo da postignemo pouzdani transfer podataka. Ovaj odeljak zaključujemo opisom kontrole zagušenja u usluzi ABR (*available bit-rate*) u ATM (*asynchronous transfer mode*) mrežama. Sledeci odeljak sadrži detaljan opis algoritma za kontrolu zagušenja u TCP-u.

### 3.6.1 Uzroci i posledice zagušenja

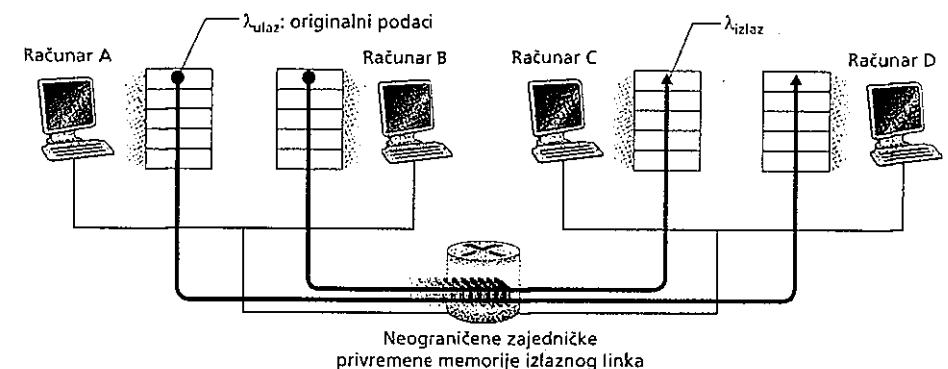
Opšti opis kontrole zagušenja počinjemo ispitujući redom tri sve složenija scenarija u vezi sa pojmom zagušenja. U svakom od ovih slučajeva videćemo zašto uopšte dolazi do zagušenja i koje su njihove posledice(u smislu resursa koji se nedovoljno koriste i loših performansi u krajnjim sistemima). Nećemo (za sada) razmatrati kako treba reagovati ili izbeći; zagušenje već samo želimo da objasnimo šta se dogada kada računari povećaju brzine prenosa, pa mreža postane zagušena.

Scenario 1: Dva pošiljaoca, jedan ruter sa beskonačnim privremenim memorijama

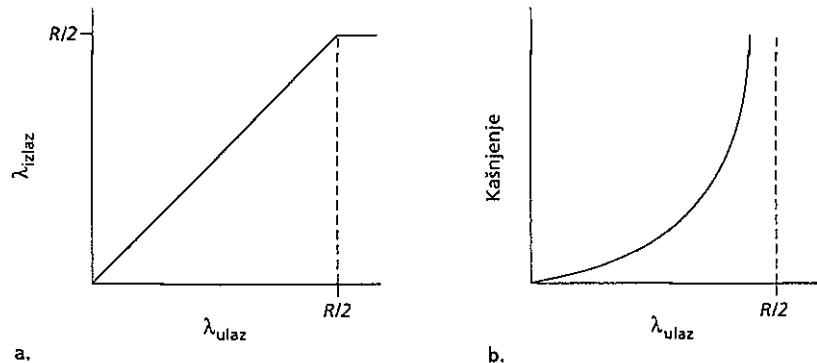
Počinjemo sa možda najjednostavnijim scenarijem zagušenja: dva računara (A i B) od kojih svaki ima konekciju sa jednim skokom između izvora i odredišta kao što je prikazano na slici 3.42.

Uzmimo da aplikacija u računaru A šalje podatke u konekciju (na primer, tako što predaje podatke protokolu transportnog sloja kroz soket) prosečnom brzinom  $\bar{K} = z$  bajtova/sekundi. Ti podaci su „originalni“ u smislu da se svaka jedinica podataka šalje u soket samo jednom. Protokol transportnog sloja ispod aplikacije je jednostavan. Podaci se enkapsuliraju i šalju; ne primenjuje se nikakvo ispravljanje grešaka (na primer, ponovo stanje), kontrola toka niti kontrola zagušenja. Ako zanemarimo dodatno opterećenje koje potiče od dodavanja infomacija u zaglavljaju transportnog sloja i nižih slojeva, brzina kojom računar A nudi saobraćaj ruteru iznosi  $K_{hz}$  bajtova/sekundi. Računar B funkcioniše na sličan način, pa radi jednostavnosti pretpostavljamo da i on šalje brzinom od  $X_{UVM}$  bajtova/sekundi. Paketi iz računara A i B prolaze kroz ruter i preko zajedničkog izlaznog linka kapaciteta  $R$ . Ruter ima privremene memorije koje mu omogućavaju čuvanje pristiglih paketa kada je brzina pristizanja veća od kapaciteta izlaznog linka. U ovom prvom scenariju pretpostavljamo da ruter ima beskonačnu količinu privremene memorije.

Na slici 3.43 prikazane su performanse konekcije računara A po tom prvom scenariju. Levi grafikon prikazuje propusnu moć po konekciji (broj bajtova u sekundi kod primaoca) u funkciji brzine slanja u konekciju. Za brzinu slanja između 0 i  $R/2$ , propusna moć kod primaoca jednak je brzini slanja kod pošiljaoca - što god pošiljalac pošalje prima se kod primaoca uz konačno kašnjenje. Međutim, ako je brzina slanja veća od  $R/2$ , propusna moć iznosi samo  $R/2$ . Ova gornja granica propusne moći potiče od toga što kapacitet linka dele dve konekcije. Link jednostavno ne može primaocu da isporuči pakete stalnom brzinom većom od  $R/2$ . Bez obzira na to koliko računari A i B povećavaju brzine slanja, nijedan od njih neće nikada postići propusnu moć veću od  $R/2$ .



**Slika 3.42** ♦ Scenario zagušenja 1: dve konekcije sa jednim zajedničkim skokom koji ima beskonačne privremene memorije

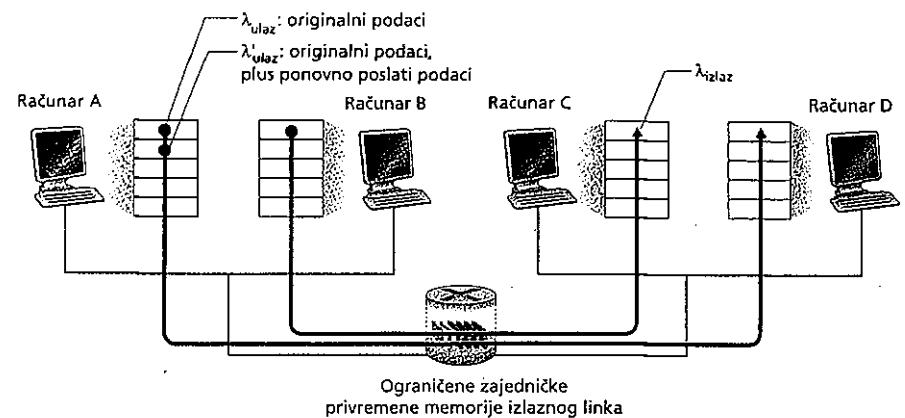


**Slika 3.43** ♦ Scenario zagušenja 1: propusna moć i kašnjenje u funkciji brzine slanja u računaru

Postizanje propusne moći od  $R/2$  po konekciji može čak da izgleda „uspešno” pošto je link potpuno iskorišćen isporukom paketa na njihova odredišta. Međutim, desni grafikon na slici 3.43 prikazuje šta se događa kada link radi blizu maksimalnog kapaciteta. Kako se brzina slanja približava vrednosti  $R/2$  (sa leve strane), prosečno kašnjenje postaje sve veće i veće. Kada brzina slanja pređe  $R/2$  prosecan broj paketa u redu za čekanje na ruteru je beskonačan, a prosečno kašnjenje od izvora do odredišta postaje neograničeno (pod pretpostavkom da konekcije rade pri istim brzinama slanja tokom beskonačnog vremenskog intervala). Prema tome, mada rad sa ukupnom propusnom moći blizu  $R$  može da bude idealan što se tiče same propusne moći, daleko je od idealnog što se tiče kašnjenja. *Čak i u ovom (krajnje) idealizovanom scenariju, već smo pronašli jednu posledicu zagušene mreže - kada se brzina pristizanja paketa približava kapacitetu linka, javlja se veliko kašnjenje u redovima za čekanje.*

Scenario 2: Dva pošiljaoca, jedan ruter sa konačnim privremenim memorijama

Sada ćemo malo promeniti prvi scenario na sledeća dva načina (slika 3.44). Prvo, uzećemo daje količinu privremene memorije na ruteru konačna. Posledica ove realne pretpostavke je da će se paketi odbacivati ako stignu u privremenu memoriju koja je već puna. Drugo, pretpostavljamo da je svaka konekcija pouzdana. Ako se kod rutera ispusti paket koji sadrži segment transportnog sloja, njega će pošiljalac kad-tad ponovo da pošalje. Pošto paketi mogu ponovo da se šalju, moramo sada pažljivije da koristimo izraz „brzina slanja“. Konkretno, označimo ponovo brzinu kojom aplikacija šalje originalne podatke u soketu sa  $>_{ulaz}$  bajtova/sekundi. Brzinu kojom transportni sloj šalje segmente (sa originalnim podacima *kao i* sa podacima

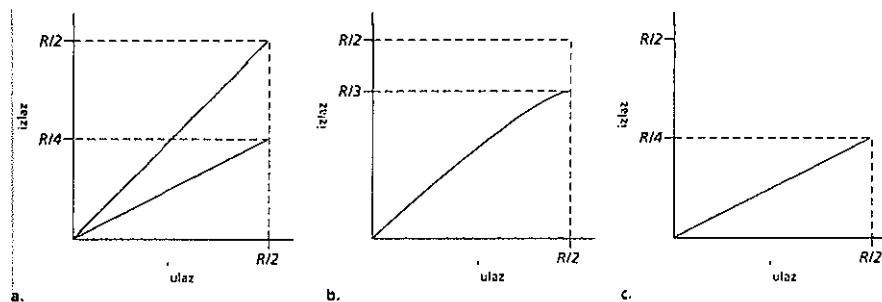


**Slika 3.44** ♦ Scenario 2: dva računara (sa ponovnim slanjima) i ruter sa konačnim privremenim memorijama

koji se ponovo šalju) u mrežu označićemo sa  $^*_{ulaz}$  bajtova/sekundi.  $X'_{ulaz}$  se ponekad naziva **ponudeno opterećenje** za mrežu.

Performanse ostvarene po scenariju 2 će sada u velikoj meri zavisiti od načina ponovnog slanja. Prvo, uzmiimo nestvarni slučaj da računar A bude u stanju da nekako (volješno!) utvrdi da li je privremena memorija u ruteru slobodna i da šalje paket samo kad je tako. U ovom slučaju ne bi bilo gubitaka,  $X_{ulaz}$  bilo bijednako  $A'_{ulaz}$ , a propusna moć konekcije bila bi jednaka  $A_{ulaz}$ . Ovaj slučaj prikazan je gornjom krivom na slici 3.45(a). Što se tiče propusne moći, performanse su idealne - Što god se pošalje biva i primljeno. Obratite pažnju na to da po ovom scenariju prosečna brzina slanja ne može preći  $R/2$ , pošto smo prepostavili da nikad ne dolazi do gubitka paketa.

Razmotrimo sledeći malo realniji slučaj kada pošiljalac ponovo šalje samo onaj paket za koji se pouzdano zna daje izgubljen. (I ova je pretpostavka malo nategnuta. Međutim, otpremni računar može da postavi dovoljno velik interval za tajm-aut pa će biti praktično siguran da se paket, koji u tom intervalu nije potvrđen, izgubio.) U ovom slučaju, performanse bi izgledale približno kao na slici 3.45(b). Da biste shvatili šta se ovde događa, razmotrite slučaj kada ponudeno opterećenje  $*_{ulaz}$  (brzina slanja originalnih podataka i ponavljanja), iznosi  $R/2$ . Prema slici 3.45(b), za tu vrednost ponudenog opterećenja, brzina kojom se podaci isporučuju prijemnoj aplikaciji iznosi  $R/3$ . Prema tome, od predatih  $0,5 R$  jedinica podataka, (prosečno)  $0,333 R$  bajtova/sekundi predstavlja originalne podatke, dok (prosečno)  $0,166 R$  bajtova/sekundi predstavlja ponovo poslatе podatke. *Ovde vidimo još jednu posledicu zagušene mreže - pošiljalac mora ponovo da šalje podatke da bi nadoknadio pakete ispuštene (izgubljene) zbog prelivanja privremene memorije.*



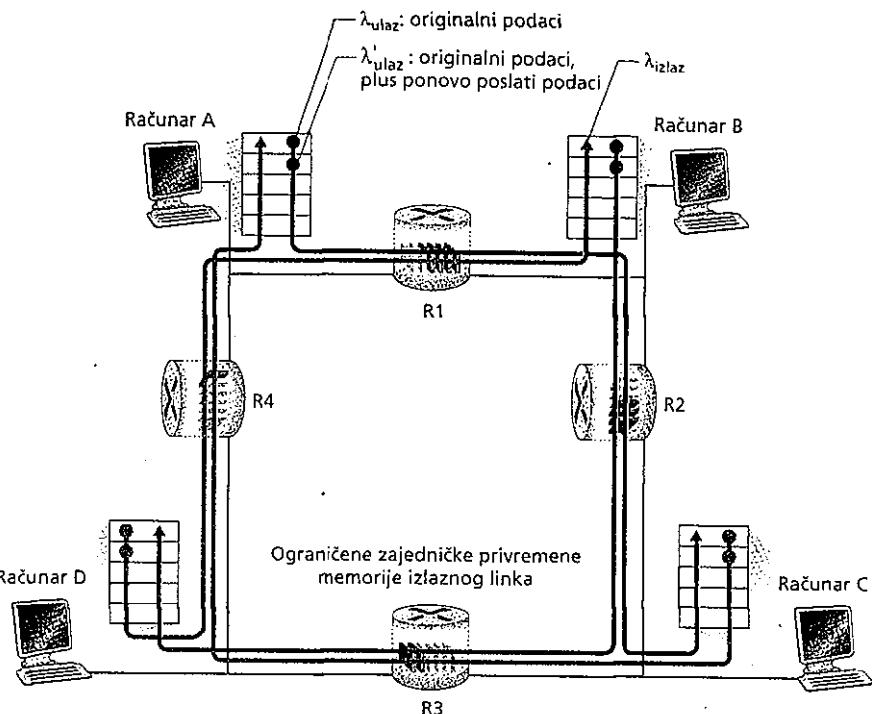
**Slika 3.45** ♦ Scenario zagušenja 2: performanse sa konačnim privremenim memorijama

Na kraju, razmotrimo slučaj kada kod pošiljaoca pre vremena nastupi tajm-aut i on ponovo pošalje paket koji kasni zbog čekanja u redu, a nije stvarno izgubljen. U tom slučaju, može se dogoditi da do primaoca stignu i originalni paket i paket koji je ponovo poslat. Naravno, primaocu treba samo jedan primerak ovog paketa, pa će on odbaciti paket koji je ponovo poslat. U ovom slučaju, „rad“ koji je ruter uložio u prosleđivanje ponovno poslate kopije originalnog paketa biće „uzaludan“ pošto je primalac već dobio originalni primerak ovog paketa. Ruter bi bolje iskoristio prenosni kapacitet linka daje umesto toga poslao neki drugi paket. *Ovde vidimo još jednu posledicu zagušene mreže - nepotrebno ponovno slanje izazvano velikim kašnjenjem dovodi do toga da ruter troši propusni opseg linka za prosleđivanja nepotrebnih kopija paketa.* Slika 3.45(c) prikazuje propusnu moć prema ponuđenom opterećenju ako se prepostavi da ruter svaki paket prosleđuje (u prošeku) dva puta. Pošto se svaki paket prosleđuje dva puta, ostvarena propusna moć može imati asimptotnu vrednost  $R/4$  kada ponuđeno opterećenje dostigne  $R/2$ .

### Scenarij 3: Četiri pošiljaoca, ruteri sa konačnim privremenim memorijama i putanja od više skokova

U našem završnom scenariju zagušenja, Četiri računara šalju pakete putanjama koje se preklapaju i od kojih svaka ima po dva skoka, kao što je prikazano na slici 3.46. Ponovo pretpostavljamo da u implementaciji usluge pouzdanog transfera podataka svi računari koriste mehanizam ponovnog slanja na osnovu tajm-aut-a, da svi raču-nari imaju istu vrednost  $\lambda_{\text{izlaz}}$ , a da svi linkovi rutera imaju kapacitet od  $R$  bajtova/ sekundi.

Razmotrimo konekciju od računara A do računara C koja prolazi kroz rutere R1 i R2. Konekcija A-C deli ruter R1 sa konekcijom D-B, a ruter R2 sa konekcijom B-D. Za ekstremno male vrednosti prelivanje privremene memorije je retko (kao u scenarijima zagušenja 1 i 2), a propusna moć je približno jednaka ponude-



**Slika 3.46** ♦ Četiri pošiljaoca, ruteri sa konačnim privremenim memorijama i putanje od više skokova

nom opterećenju. Kod nešto većih vrednosti  $X_{\text{izlaz}}$ , odgovarajuća propusna moć je takođe veća pošto se više originalnih podataka prenosi u mrežu i isporučuje na odredište, a prelivanja su takođe retka. Prema tome, kod malih vrednosti  $X_{\text{izlaz}}$  povećanje vrednosti  $X_{\text{izlaz}}$  dovodi do povećanja vrednosti  $X_{\text{izlaz}}$ .

Pošto smo razmotrili slučaj izuzetno malog saobraćaja, pogledajmo sada slučaj kada je  $\lambda_{\text{in}}$  (pa tako i  $\lambda'_{\text{in}}$ ) izuzetno veliko. Pogledajmo ruter R2: Saobraćaj A-C koji stiže do ruteru R2 (a dolazi do ruteru R2 tako stoje prosleden sa R1) može da pristiže brzinom od najviše  $J^*$ , koliko iznosi kapacitet linka od R1 do R2, bez obzira na vrednost  $X_{\text{izlaz}}$ . Ako je  $X_{\text{izlaz}}$  izuzetno veliko za sve konekcije (uključujući konekciju B-D), brzina pristicanja saobraćaja B-D na ruteru R2 može da bude mnogo veća od brzine saobraćaja A-C. Pošto saobraćaj A-C i saobraćaj B-D moraju da se takmiče na ruteru R2 za ograničeni prostor privremene memorije, količina saobraćaja A-C koja uspešno prođe kroz R2 (tj. koja se ne izgubi zbog prelivanja pri-

vremene memorije) postaje sve manja i manja kako raste ponudeno opterećenje od B~D. U graničnom slučaju, pošto ponudeno opterećenje teži beskonačnosti, prazna privremena memorija u R2 odmah se puni paketom B-D, a propusna moć konekcije A-C na ruteru R2 teži nuli. Ovo zatim dovodi do toga da propusna moć A-C sa kraja na kraj teži nuli u situaciji velikog saobraćaja. Ova razmatranja dovode nas do zaključka da ponudeno opterećenje ima negativne posledice na propusnu moć, kao što je prikazano na slici 3.47.

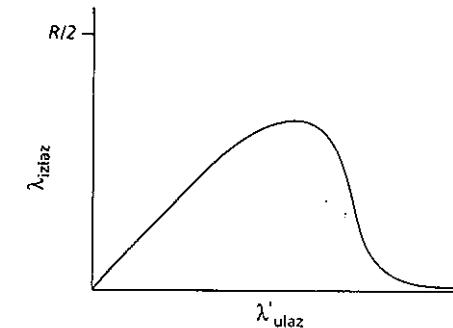
Razlog za opadanje propusne moći kod povećanja ponuđenog opterećenja očigledan je kada se razmotri količina uzaludnog „rada“ na mreži. U scenariju sa velikim saobraćajem koji smo upravo opisali, kad god se na ruteru prilikom drugog skoka ispusti paket, „rad“ koji je prvi ruter uložio u prosleđivanje paketa drugom ruteru na kraju ostaje „uzaludan“. Mreža bi isto tako dobro prošla (tačnije, isto bi tako loše prošla) daje prvi ruter jednostavno odbacio paket i ostao besposlen. U stvari, kapacitet prenosa koji je prvi ruter upotrebio za prosleđivanje paketa drugom ruteru mnogo bi korisnije bio upotrebljen daje prenet neki drugi paket. (Na primer, kada se bira paket za slanje, bilo bi bolje da ruter da prednost paketima koji su već prešli određen broj ruteru.) *Ovde vidimo još jednu posledicu ispuštanja paketa zbog zagušenja - kada se paket ispusti negde na putanji, ispada da je uzaludno utrošen kapacitet prenosa koji je upotrebljen za prosleđivanje paketa na svakom od uzvodnih ruta do momenta ispuštenja paketa.*

### 3.6.2 Pristupi kontroli zagušenja

U odeljku 3.7 ćemo veoma detaljno opisati kako TCP pristupa kontroli zagušenja.<sup>1</sup> Ovde ćemo ukazati na dva pristupa kontroli zagušenja koji se koriste u praksi i opisati konkretnе arhitekture mreže i protokole za kontrolu zagušenja koji oličavaju te pristupe.

d) najširem smislu, pristupe kontroli zagušenja možemo da razvrstamo na osnovu toga da li mrežni sloj daje transportnom sloju ikakvu eksplisitnu pomoć u svrhu kontrole zagušenja:

◆ **Kontrola zagušenja sa kraja na kraj.** U pristupu kontrole zagušenja sa kraja na kraj mrežni sloj ne daje transportnom sloju nikakvu eksplisitnu podršku u svrhu kontrole zagušenja. Čak i samo prisustvo zagušenja na mreži mora da se utvrdi u krajnjim sistemima jedino posmatranjem ponašanja mreže (na primer, gubljenje paketa i kašnjenje). U odeljku 3.7 videćemo da TCP obavezno mora da prihvati ovaj pristup sa kraja na kraj pošto sloj IP ne obezbeđuje krajnjim sistemima nikakvu povratnu informaciju o zagušenju mreže. Gubitak TCP segmenta (koji se otkriva pomoću tajm-auta ili trostrukih potvrda) uzima se kao znak da je došlo do zagušenja mreže, pa TCP na odgovarajući način smanjuje veličinu prozora. Videćemo takođe da se, u novim predlozima za TCP, kao znak da zagušenje mreže raste koriste povećane vrednosti kašnjenja povratnog puta.



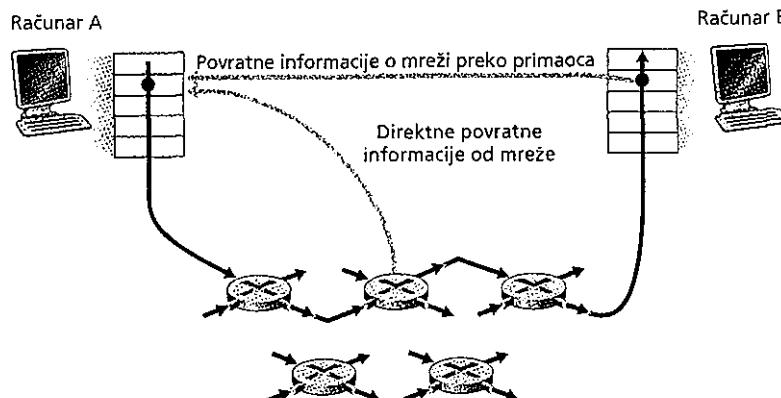
**Slika 3.47** ◆ Scenario 3: performanse sa konačnim privremenim memorijama i putanijama od više skokova

◆ **Kontrola zagušenja pomoću mreže.** Kod kontrole zagušenja pomoću mreže, mrežne komponente (tj. ruteri) daju pošiljaocu eksplisitne povratne informacije o stanju zagušenja na mreži. Ova povratna informacija može da bude jednostavno samo jedan bit koji ukazuje na zagušenje linka. Takav pristup bio je prihvaćen u ranim arhitekturama IBM SNA [Schvartz 1982] i DEC DECnet [Jain 1989; Ramakrishnan 1990], nedavno je predložen za TCP/IP mreže [Floyd TCP 1994; RFC 2481], a takođe se koristi i u kontroli zagušenja ATM ABR (available bit-rate), koju ćemo kasnije opisati. Moguće su i savršenije povratne informacije o mreži. Na primer, jedan oblik kontrole zagušenja ATM ABR koji ćemo uskoro proučiti omogućava ruteru da eksplisitno obavesti pošiljaoca o brzini prenosa koju ruter može da podrži na izlaznom linku.

Kod kontrole zagušenja pomoću mreže, informacija o zagušenju od mreže ka pošiljaocu obično se prosleđuje na jedan od dva načina prikazana na slici 3.48. Direktna povratna informacija se šalje sa mrežnog ruteru pošiljaocu. Ova vrsta obaveštavanja obično je u obliku paketa gušenja (koji u suštini kaže, „zagušen sam!“). Drugi oblik obaveštavanja je kada ruter u paketu koji putuje od pošiljaoca ka primaocu označava ili ažurira polje koje ukazuje na zagušenje. Kada primi označeni paket, primalac zatim obaveštava pošiljaoca daje dobio upozorenje o zagušenju. Obratite pažnju na to da ovaj poslednji oblik obaveštavanja zahteva najmanje vreme jednog punog povratnog puta.

### 3.6.3 Primer kontrole zagušenja pomoću mreže: Kontrola zagušenja ATM ABR

Ovaj odeljak zaključujemo kratkim opisom algoritma za kontrolu zagušenja u usluzi ATM ABR - protokol u kojem se za kontrolu zagušenja koristi pomoć mreže. Naglašavamo da nam ovde nije cilj da detaljno opišemo aspekte arhitekture ATM, već da



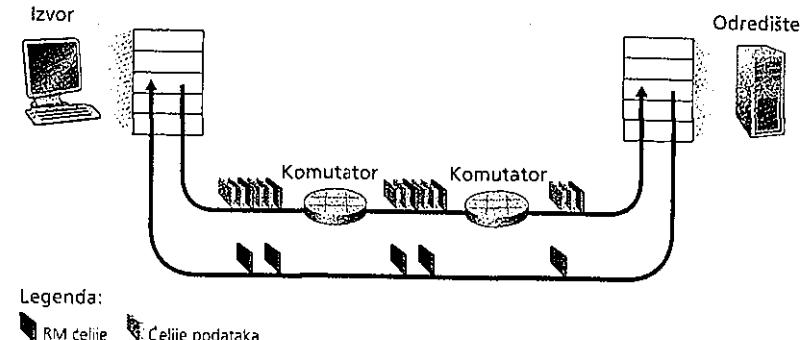
**Slika 3.48** ♦ Dve putanje povratnih informacija o zagušenju na koje upozorava mreža

prikažemo protokol sa značajno drugačijim pristupom kontroli zagušenja od inter-netskog protokola TCP. Prikazaćemo samo onih nekoliko aspekata ATM arhitekture koji su potrebni za shvatanje ABR kontrole zagušenja.

U osnovi, ATM za komutiranje paketa koristi princip virtualnih kola (VC). Sigurno se sećate iz poglavlja 1 da to znači da svaki komutator na putanji od izvora do odredišta vodi stanje virtuelnog kola od izvora do odredišta. To stanje virtuelnog kola omogućava komutatoru da prati ponašanje pojedinačnih pošiljalaca (npr. da prati njihovu prosečnu brzinu slanja) i da preduzima aktivnosti za kontrolu zagušenja zavisno od izvora (npr. da eksplisitno signalizira pošiljaocu da smanji brzinu kada komutator postaje zagušen). ATM je, zbog tih stanja virtualnih kola u mrežnim komutatorima, idealno prilagođen kontroli zagušenja pomoću mreže.

ABR je projektovan kao fleksibilna usluga transfera podataka koja podseća na TCP. Kada mreža nije opterećena ABR usluge bi trebalo da budu u stanju da iskoriste dostupan slobodan propusni opseg; kada je mreža zagušena on bi trebalo da priguši brzinu prenosa na neki unapred postavljeni minimum. Detaljan udžbenik za kontrolu zagušenja i upravljanje saobraćajem ATM ABR može se naći [Jain 1996].

Na slici 3.49 prikazano je radno okruženje za kontrolu zagušenja ATM ABR. U sledećem opisu držaćemo se ATM terminologije (na primer, koristićemo izraz „komutator“ umesto „ruter“ i „ćelija“ umesto „paket“). U usluzi ATM ABR, ćelije podataka prenose se od izvora do odredišta kroz niz usputnih komutatora. Između ćelija podataka nalaze se takozvane ćelije za upravljanje resursima, RM (Resource Management) ćelije; ubrzo ćemo videti da se te RM ćelije mogu upotrebiti za razmeđjivanje informacija o zagušenju među računarima i komutatorima. Kada se RM ćelija nade na odredištu ona se „okreće“ i vraća pošiljaocu (nakon što odredište even-



**Slika 3.49** ♦ Radno okruženje kontrole zagušenja za uslugu ATM ABR

tualno promeni njen sadržaj). I sam komutator može da napravi RM ćeliju i pošalje je direktno izvoru. Tako se RM ćelije koriste i za direktnе povratne informacije od mreže i za povratne informacije o mreži preko primaoca kao sto je prikazano na slici 3.49.

Kontrola zagušenja ATM ABR koristi pristup zasnovan na brzini. To jest, pošiljalac eksplisitno izračunava maksimalnu brzinu kojom može da šalje i tome se pri-lagodava. ABR predviđa tri mehanizma za signalizaciju zagušenja od komutatora prema primaocu:

- ♦ **EFCI bit.** Svaka ćelija podataka sadrži eksplisitnu indikaciju zagušenja unapred, EFCI (*explicitforwardcongestion indication*) bit. Zagušeni mrežni komutator može da uključi EFCI bit u ćeliji podataka na 1 i tako upozori odredišni računar na zagušenje. Odredište mora da proverava EFCI bit u svim primljenim ćelijama podataka. Kada na odredište stigne RM ćelija a u prethodno primljenoj ćeliji podataka EFCI bit je bio jednak 1, tada odredište uključuje bit za označavanje zagušenja (CI) u RM ćeliji na 1 i vraća RM ćeliju pošiljaocu. Pomoću EFCI bita u ćelijama podataka i CI bita u RM ćelijama pošiljalac može da se upozori na zagušenje u mrežnom komutatoru.
- ♦ **Bitovi CI i NI.** Kao što je već rečeno, RM ćelije od pošiljaoca ka primaocu nalaze se između ćelija podataka. Učestalost pojavljivanja RM ćelija može da se podešava parametrom Čija je podrazumevana vrednost jedna RM ćelija na svake 32 ćelije podataka. Te RM ćelije sadrže bitove koje može da uključi zagušeni mrežni komutator: bit indikacije zagušenja, CI (*congestion indication*) i bit „ne povećavaj“, NI (*no increase*). Konkretno, komutator može u slučaju blagog zagušenja da uključi na 1 bit NI u RM ćeliji koja prolazi, a u slučaju velikog zagušenja, bit CI. Kada odredišni računar primi RM ćeliju, on će je vratiti pošiljaocu sa nedirnutim bitovima CI i NI (osim što se CI može na odredištu promeniti u I kao rezultat gore opisanog mehanizma EFCI).

♦ **Postavljanje vrednosti ER.** Svaka RM celija takođe sadrži dvobajtno polje eksplisitne brzine ER (*explicit rate*). Zagušeni komutator može da smanji vrednost u polju ER svake RM celije u prolazu. Na taj način, polje ER će sadržati minimalnu brzinu koju mogu da podrže svi komutatori na putanji od izvora do odredišta.

ATM ABR izvor prilagođava brzinu kojom šalje celije zavisno od vrednosti u poljima CI, NI i ER u vraćenoj RM celiji. Pravila za to podešavanje brzine su dosta složena i pomalo dosadna za objašnjanje. Čitalac koga to zanima može naći detalje u knjizi [Jain 1996],

### 3.7 TCP kontrola zagušenja

U ovom odeljku vraćamo se poučavanju TCP-a. Kako smo saznali u odeljku 3.5, TCP obezbeđuje uslugu pouzdanog transporta između dva procesa koji se izvršavaju na različitim računarima. Još jedna ključna komponenta TCP-a je njegov mehanizam kontrole zagušenja. Kao što je napomenuto u prethodnom odeljku, TCP mora da koristi kontrolu zagušenja sa kraja na kraj, a ne kontrolu zagušenja pomoću mreže, pošto sloj IP ne daje krajnjim sistemima nikakve eksplisitne povratne informacije o zagušenju na mreži.

Pristup koji prihvata TCP je da se svaki pošiljalac primora da ograniči brzinu kojom šalje saobraćaj u svoju konekciju zavisno od uočenog zagušenja mreže. Ako TCP pošiljalac uoči daje na putanji između njega i odredišta zagušenje malo, on povećava brzinu slanja; ako uoči da postoji zagušenje, on je smanjuje. Ovaj pristup, međutim, otvara tri pitanja. Prvo, kako da TCP pošiljalac ograniči brzinu slanja saobraćaja u svoju konekciju? Drugo, kako da TCP pošiljalac prepozna da postoji zagušenje na putanji između njega i odredišta? I treće, koji bi algoritam pošiljalac trebalo da koristi za menjanje brzine slanja zavisno od uočenog zagušenja sa kraja na kraj? Sada ćemo ispitati ta tri pitanja u kontekstu algoritma TCP Reno za kontrolu zagušenja koji se koristi u većini savremenih operativnih sistema [Padhye 2001]. Da bi opis bio što konkretniji prepostavljamo da TCP pošiljalac šalje veliku datoteku.

Pogledajmo prvo kako TCP pošiljalac ograničava brzinu kojom šalje saobraćaj svoju konekciju. U odeljku 3.5 videli smo da se svaka strana TCP konekcije sastoji od prijemne privremene memorije, otpremne privremene memorije i nekoliko pro-menljivih (LastByteRead, RcvWindow itd). TCP mehanizam za kontrolu zagušenja zahteva od svake strane u konekciji da održava još jednu promenljivu, **prozor zagušenja**. Prozor zagušenja, označen sa CongWin (*congestion window*), nameće ograničenje za brzinu kojom TCP pošiljalac može da šalje saobraćaj mrežu. Konkretno, količina nepotvrđenih podataka kod pošiljaoca ne srne da pređe manju od vrednosti CongWin i RcvWindow, to jest:

LastByteSent - LastByteAcked <■ min{CongWin, RcvWindow}

Da bismo se usredosredili na kontrolu zagušenja (za razliku od kontrole toka), sada ćemo prepostaviti daje prijema TCP privremena memorija tako velika da se ograničenje prijemnog prozora može zanemariti; pa se tako količina nepotvrđenih podataka kod pošiljaoca ograničava jedino pomoću vrednosti CongWin.

Navedena granica ograničava količinu nepotvrđenih podataka kod pošiljaoca i tako indirektno ograničava njegovu brzinu slanja. Da biste to videli, razmotrite konekciju sa zanemarljivim gubicima i kašnjenjem u prenosu paketa. Tada na početku svakog povratnog puta (RTT) gornje ograničenje dozvoljava pošiljaocu da pošalje u konekciju približno CongWin bajtova podataka, a na kraju RTT-a stižu mu potvrde za te podatke. *Na taj način brzina slanja pošiljaoca iznosi približno CongWin/RTT-bajtova/sekundi. Prema tome, pošiljalac može da podesi brzinu slanja podataka u konekciju podešavanjem vrednosti CongNin.*

Razmotrimo sada kako TCP pošiljalac utvrđuje da postoji zagušenje na putanji između njega i odredišta. Definišimo „dogadjaj gubitka“ kod TCP pošiljaoca kao nastupanje tajm-aut ili primanje tri istovetna ACK-a od primaoca (dogadjaj tajm-aut opisali smo u odeljku 3.5.4 na slici 3.33 i kasnije dopunili taj opis na strani 245 dodavanjem brzog ponovnog slanja kada stignu tri istovetna ACK-a). Kada postoji izuzetno zagušenje, doći će do prelivanja jedne ili više privremenih memorija u rute-rima na putanji, što će dovesti do ispuštanja datagrama. Ispušteni datagram zatim dovodi do dogadjaja gubitka kod pošiljaoca - bilo da dode do tajm-aut ili stignu tri istovetna ACK-a - a pošiljalac će to uzeti kao znak da postoji zagušenje na putanji od njega do primaoca.

Pošto smo razmotrili kako se zagušenje otkriva, razmotrimo sada povoljniji slučaj kada u mreži nema zagušenja, tj. kada ne dolazi do gubitaka. U tom slučaju, TCP pošiljalac će primiti potvrde za prethodno nepotvrđene segmente. Kao što ćemo videti, TCP će prihvati prispeće ovih potvrda kao znak daje sve u redu - da se segmenti poslati u mrežu uspešno isporučuju na odredište - pa će iskoristiti potvrde za povećanje prozora zagušenja (pa tako i brzinu slanja). Obratite pažnju na to da će i povećanje prozora zagušenja biti relativno sporo ako potvrde stižu relativno sporo (npr. ako putanja sa kraja na kraj ima veliko kašnjenje ili sadrži link malog propusnog opsega). S druge strane, ako potvrde stižu brzo, prozor zagušenja će se povećavati brže. Pošto TCP koristi potvrde za pokretanje (ili za takt) povećanja prozora zagušenja, za TCP kažemo da ima **vlastiti takt**.

Sada možemo da razmotrimo algoritam koji TCP pošiljalac koristi da bi upravljao brzinom slanja u funkciji uočenog zagušenja. To je čuveni TCP **algoritam za kontrolu zagušenja**. Taj algoritam ima tri glavne komponente: (1) aditivno uvećanje, multiplikativno smanjenje, (2) sporo pokretanje i (3) reagovanje na dogadjaj tajm-aut.

### Aditivno uvećanje, multiplikativno smanjenje

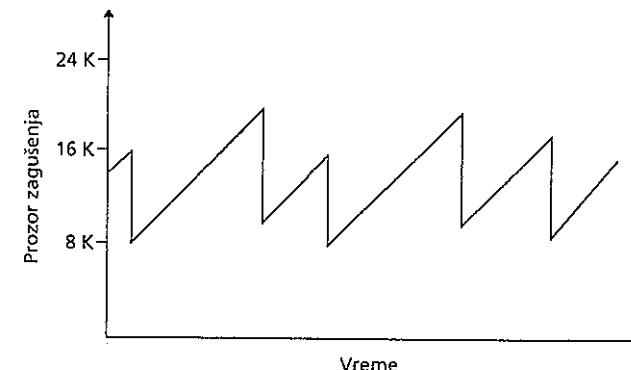
Osnovna zamisao na kojoj se zasniva TCP kontrola zagušenja jeste da pošiljalac smanji brzinu slanja (smanjenjem veličine prozora zagušenja, CongWin) kada dođe do gubitka. Pošto se ostalim TCP konekcijama koje prolaze kroz iste zagušene rutere verovatno takođe dogadaju gubici, i one će verovatno smanjiti svoje brzine slanja smanjenjem vlastitih vrednosti CongWin. Sveukupni efekat biće da izvori putanja koje prolaze kroz zagušene rutere smanje brzinu kojom šalju saobraćaj u mrežu što bi trebalo da smanji ukupno zagušenje tih ruta. Ali, za koliko bi trebalo da TCP pošiljalac smanji prozor zagušenja kada nastupi dogadjaj gubitka? TCP koristi takozvani pristup „multiplikativnog smanjenja“ tako što prepolovi trenutnu vrednost CongWin svaki put kada nastupi gubitak. Prema tome, ako je vrednost CongWin trenutno 20 kilobajtova, pa se otkrije gubitak, CongWin se smanjuje na pola, tj. na 10 kilobajtova. Ako dođe do još jednog gubitka, CongWin se dalje smanjuje na 5 kilobajtova. Vrednost CongVJin može i dalje da pada, ali se ne dozvoljava da padne ispod jednog MSS. (Ovo je „gruba slika“ načina na koji se prozor zagušenja menja nakon dogadjaja gubitka. U suštini, stvari su malo složenije. Kao što ćemo uskoro videti, vrednost CongWin nakon tajm-autu pada na 1 MSS, a zatim se brzo povećava do polovine svoje prethodne vrednosti.)

Pošto smo opisali kako TCP pošiljalac smanjuje brzinu slanja kada uoči zagušenje, prirodno je nakon toga razmotriti kako da TCP poveća brzinu slanja ako uoči da nema zagušenja, tj. ako ne dolazi do gubitaka. Obrazloženje za povećanje brzine je da ako nije uočeno zagušenje, znači da verovatno postoji dostupna (neupotrebljena) propusna moć koju bi TCP konekcija još mogla da upotrebii. U takvim uslovima, TCP polako povećava prozor zagušenja, obazirivo „istražujući“ da li postoji dodatna propusna moć na putanji sa kraja na kraj. TCP pošiljalac to postiže povećanjem vrednosti CongWin po malo svaki put kada primi ACK, tako da poveća CongWin približilo za po jedan MSS za svaki vrednosni interval povratnog puta [RFC 2581]. To može da se postigne na nekoliko načina; uobičajeno je da TCP pošiljalac poveća CongVJin za MSS-(MSS/ CongWin) bajtova kad god primi novu potvrdu. Na primer, ako MSS iznosi 1460 bajtova a CongVJin 14 600 bajtova, znači da se u jednom RTT-u šalje 10 segmenata. Svaki put kada stigne ACK (prepostavlja se po jedan ACK za svaki segment) prozor zagušenja povećava se za 1/10 MSS, tako da će nakon potvrda za svih 10 segmenata vrednost prozora zagušenja biti veća za 1 MSS, kao što se i želelo.

Sve u svemu, TCP pošiljalac aditivno uvećava brzinu kada uoči da na putanji s kraja na kraj nema zagušenja, a multiplikativno smanjuje brzinu kada uoči (po dogadjaju gubitka) da je putanja zagušena. Iz tog razloga, TCP kontrola zagušenja često se naziva algoritmom AIMD (*additive-increase, multiplicative-decrease*). U TCP-ovom protokolu kontrole zagušenja faza linearog povećanja poznata je kao izbegavanje zagušenja. Vrednost CongVJin stalno prolazi kroz cikluse tokom kojih se linearno povećava, a zatim naglo smanjuje na polovinu trenutne vrednosti (ako nastupi dogadjaj gubitka), pa u dugotrajnim TCP konekcijama prati šablon zubaca na testeru, kao sto je prikazano na slici 3.50.

### Sporo pokretanje

Na početku TCP konekcije, vrednost CongVJin obično se postavlja na jedan MSS [RFC 3390], što daje početnu brzinu slanja od približno MSS/RTT. Na primer, ako je MSS = 500 bajtova, a RTT = 200 ms, imaće se početnu brzinu slanja od približno svega 20 kb/s. Pošto propusni opseg dostupan konekciji može da bude mnogo veći od MSS/RTT, bilo bi šteta da se brzina uvećava samo linearno jer bi se tako previše čekalo da ona dođe do nekog znatnijeg nivoa. Zato u toj početnoj fazi umesto linearne povećanja brzine, TCP pošiljalac povećava brzinu eksponencijalno tako što za svaki RTT udvostručava vrednost CongWin. TCP pošiljalac nastavlja sa eksponencijalnim povećavanjem brzine slanja dok ne nastupi dogadjaj gubitka kada se CongVJin smanjuje na polovinu, a nakon toga raste linearno kao što smo ranije opisali. Prema tome, tokom ove inicijalne faze, koja se naziva sporim početkom (*slow start SS*), TCP pošiljalac počinje slanje malom brzinom (otuda naziv „*sporim početak*“), ali eksponencijalno povećava brzinu slanja. Pošiljalac postiže eksponencijalni rast tako što uvećava vrednost CongVJin za jedan MSS za svaku primljenu potvrdu poslatog segmenta. Konkretno, TCP šalje prvi segment u mrežu i čeka na potvrdu. Ako se taj segment potvrdi pre dogadjaja gubitka, TCP pošiljalac povećava prozor zagušenja za jedan MSS i šalje dva segmenta maksimalne veličine. Ako se ti segmenti potvrde pre dogadjaja gubitka, pošiljalac povećava prozor zagušenja za po jedan MSS za svaki potvrđeni segment, čime se dobija prozor zagušenja od četiri MSS-a pa se šalju četiri segmenta maksimalne veličine. Taj postupak se nastavlja sve dok potvrde stižu pre dogadjaja gubitka. Na taj način se vrednost CongVJin u fazi sporog početka udvostručava za svaki RTT.



**Slika 3.50** ♦ Kontrola zagušenja sa aditivnim uvećanjem, multiplikativnim smanjenjem

### Reagovanje na događaj tajm-aut

Do sada smo opisali ponašanje TCP-ovog prozora zagušenja tako što on od 1 MSS raste eksponencijalno (tokom sporog početka) dok ne nastupi događaj gubitka kada počinje šablon AIMD testere. Mada je taj prikaz prilično precizan, ne bi bilo u redu da izostavimo činjenicu kako TCP kontrola zagušenja različito reaguje na događaj gubitka koji nastupa zbog tajm-aut-a, a drugačije na gubitak otkriven prijemom tri istovetna ACK-a. Kada primi tri istovetna ACK-a, TCP se ponaša kao što smo upravo opisali - prozor zagušenja se prepolovi, a zatim se linearno povećava. Ali, ako nastupi događaj tajm-aut, TCP pošiljalac se vraća u fazu sporog početka - tj. podešava prozor zagušenja na 1 MSS, a zatim ga eksponencijalno povećava. Prozor i dalje eksponencijalno raste sve dok CongWin ne dostigne polovinu vrednosti koju je imao pre tajm-aut-a. Od tog trenutka, CongWin raste linearno, kao što bi nastavio nakon tri identična ACK-a.

TCP upravlja tom složenijom dinamikom pomoću jedne promenljive po imenu Threshold (prag) koja određuje veličinu prozora na kojoj će se prekinuti spori početak i početi izbegavanje zagušenja. Promenljivoj Threshold se daje velika početna vrednost (u praksi 65 kilobajtova [Stevens 1994]), tako da na početku nema efekta. Kad god nastupi događaj gubitka, Threshold poprima polovinu trenutne vrednosti CongMin. Na primer, ako je neposredno pre događaja gubitka prozor zagušenja bio 20 kilobajtova, vrednost Threshold postaje 10 kilobajtova i ostaje toliko sve do sledećeg događaja gubitka.

Pošto smo opisali promenljivu Threshold, sada možemo tačno da opišemo kako se ponaša CongWin nakon događaja tajm-aut. Kao stoje upravo napomenuto, nakon tajm-aut-a TCP pošiljalac prelazi u fazu sporog početka. Dok se nalazi u toj fazi on eksponencijalno povećava vrednost CongWin sve dok CongWin ne dostigne vrednost Threshold. Kada CongWin dostigne Threshold, TCP prelazi u fazu izbegavanja zagušenja kada CongWin raste linearno, kao što smo ranije opisali.

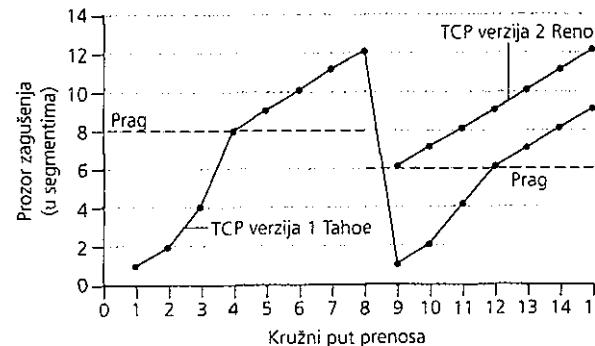
Sve u svemu, TCP-ov algoritam za kontrolu zagušenja ukratko je prikazan u tabeli 3.3. Sada je prirodno da se razmotri zašto se TCP-ova kontrola zagušenja drugačije ponaša nakon događaja tajm-aut nego nakon primanja tri istovetna ACK-a. Konkretno, zašto se TCP pošiljalac ponaša drastično nakon događaja tajm-aut kada smanjuje prozor zagušenja na 1 MSS, dok nakon primanja tri istovetna ACK-a smanjuje prozor zagušenja samo na polovinu? Zanimljivo je da u staroj verziji TCP-a, poznatoj kao **TCP Tahoe**, nakon svake vrste događaja gubitka, prozor zagušenja se bezuslovno smanjuje na 1 MSS i prelazi u fazu sporog početka. Kao što smo videli, u novoj verziji TCP-a, poznatoj kao **TCP Reno**, izbacuje se faza sporog početka nakon tri istovetna ACK-a. Obrazloženje za ukidanje sporog početka u tom slučaju jeste da mada je paket izgubljen, pristizanje tri istovetna ACK-a ukazuje na to da su neki segmenti (konkretno, ta tri dodatna segmenta nakon izgubljenog) ipak primljeni na strani pošiljaoca. Prema tome, za razliku od tajm-aut-a, mreža je u stanju da isporučuje bar neke segmente, iako su neki izgubljeni zbog zagušenja. Ovo izbacivanje faze sporog početka nakon tri istovetna ACK-a naziva se brzi oporavak.

| Stanje                     | Događaj                                     | Akcija kontrole zagušenja TCP pošiljaoca                                               | Komentar                                                                |
|----------------------------|---------------------------------------------|----------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Spori početak (SS)         | Stiže ACK za prethodno nepotvrđene podatke  | $CongWin = CongWin + MSS$ ,<br>if ( $CongWin > Threshold$ )<br>stanje $\rightarrow$ CA | CongWin se udvostručava za svaki RTT                                    |
| Izbegavanje zagušenja (CA) | Stiže ACK za prethodno nepotvrđene podatke  | $CongWin = CongWin + MSS - (MSS/CongWin)$                                              | Aditivno uvećanje, CongWin se za svaki RTT povećava za 1 MSS            |
| SS ili CA                  | Gubitak otkriven na osnovu trostrukog ACK-a | $Threshold = CongWin/2$<br>$CongWin = Threshold$<br>stanje $\rightarrow$ CA            | Brzi oporavak, multiplikativno smanjivanje. CongWin ne pada ispod 1 MSS |
| SS ili CA                  | Tajm-aut                                    | $Threshold = CongWin/2$<br>$CongWin = 1 MSS$<br>stanje $\rightarrow$ SS                | Prelazak na spori početak                                               |
| SS ili CA                  | Duplikat ACK-a                              | Povećanje brojača duplih ACK-ova za potvrđeni segment                                  | CongWin i Threshold se ne menjaju                                       |

**Tabela 3.3** ♦ Kontrola zagušenja TCP pošiljaoca [RFC 2581], pod pretpostavkom da početna vrednost CongWin iznosi MSS, da je početna vrednost Threshold velika (npr. 65 kilobajta [Stevens 1994]) i da TCP pošiljalac kreće iz stanja sporog početka. Prikazuje se stanje TCP pošiljaoca neposredno pre nastupanja događaja. Ostale detalje naći će se u [RFC 2581].

Na slici 3.51 prikazane su promene TCP prozora zagušenja u verzijama Reno i Tahoe. Na ovoj slici, prag na početku iznosi  $8 \cdot MSS$ . Prozor zagušenja se povećava eksponencijalno tokom sporog početka i dostiže prag pri četvrtom povratnom putu. Nakon toga, prozor zagušenja raste linearno dok ne dođe do tri istovetna ACK-a, malo nakon povratnog puta broj 8. Obratite pažnju na to daje u trenutku ovog gubitka prozor zagušenja jednak  $12 - MSS$ . Prag zatim dobija vrednost  $0.5 \cdot CongWin = 6 \cdot MSS$ . U verziji TCP Reno, prozor zagušenja dobija vrednost  $CongWin = 6 \cdot MSS$  i nakon toga raste linearno. U verziji TCP Tahoe, prozor zagušenja dobija vrednost  $1 MSS$  i raste eksponencijalno dok ne dostigne prag. Ovaj algoritam za kontrolu zagušenja uspostavio je V. Jacobson [Jacobson 1988]; niz modifikacija Jackbsonovog početnog algoritma opisan je u knjigama [Stevens 1994] i [RFC 2581].

Kao stoje gore napomenuto, većina TCP implementacija trenutno koristi algoritam Reno. Za algoritam Reno predloženo je mnogo varijanti [RFC 2582; RFC 2018]. Predloženi algoritam TCP Vegas [Brakmo 1995; Ahn 1995] pokušava da izbegne zagušenje, a da istovremeno održi dobra propusnu moć. Osnovna zamisao algoritma Vegas je da se (1) zagušenje u ruterima između izvora i odredišta otkrije



Slika 3.51 ♦ Promene TCP-ovog prozora zagušenja (Tahoe i Reno)

pre nego što dode do gubitka paketa i da se (2) brzina smanji linearno kada se otkrije da preti gubitak paketa. Gubitak paketa predviđa se posmatranjem trajanja povratnog puta. Što je veći RTT paketa, znači da je veće i zagušenje ruta.

#### Grubi opis TCP propusne moći

S obzirom na testerasti grafikon ponašanja TCP-a, prirodno je razmotriti kolika bi bila prosečna propusna moć (tj. prosečna brzina) dugotrajne TCP konekcije. U ovoj analizi zanemarićemo faze sporog početka do kojih dolazi nakon događaja tajm-aut. (Te su faze obično veoma kratke pošto pošiljalac izlazi iz njih eksponencijalnom brzinom.) Tokom pojedinačnog intervala povratnog puta brzina kojom TCP šalje podatke zavisi od prozora zagušenja i od trenutnog *RTT-a*. Kao što je već rečeno, kada prozor iznosi w u bajtova, a trenutno vreme povratnog puta je *RTT* sekundi, brzina prenosa TCP-a je približno  $w/RTT$ . TCP tada pokušava da upotrebi dodatni propusni opseg povećavanjem vrednosti  $w$  za jedan MSS kod svakog *RTT-a* dok se ne pojavi gubitak. Označite sa  $W$  vrednost  $w$  u trenutku gubitka. Pod pretpostavkom da su *RTT* i  $W$  približno konstantni tokom cele konekcije, brzina TCP prenosa kreće se od  $W/(2 \cdot RTT)$  do  $W/RTT$ .

Ove pretpostavke daju nam krajnje pojednostavljeni grubi model ponašanja TCP-ovog stacionarnog stanja. Mreža ispušta paket iz konekcije kada brzina dostigne  $W/RTT$ , brzina se tada smanjuje na pola i ponovo povećava za jedan  $MSS/RTT$  kod svakog *RTT-a* dok ponovo ne dostigne  $W/RTT$ . Ovaj proces se stalno ponavlja. Pošto se propusni opseg TCP-a (tj. njegova brzina) povećava linearno između dve krajnje vrednosti, imamo:

$$\text{prosečna propusna moć konekcije} = \frac{0,75 \cdot W}{RTT}$$

Pomoću ovog potpuno idealizovanog modela dinamike stacionarnog stanja TCP-a, možemo takođe da izvedemo jedan zanimljiv izraz koji povezuje učestalost gubitaka u jednoj konekciji sa njenim dostupnim propusnim opsegom [Mahdavi 1997]. Ovo izvođenje skicirano je u problemima za domaći zadatak. Jedan složeniji model koji se empirijski slaže sa izmerenim podacima je [Padhve 2000],

#### Budući TCP

Važno je shvatiti da se TCP kontrola zagušenja tokom godina razvijala i da se još uvek razvija. Prikaz TCP kontrole zagušenja kakva je bila krajem 1990-ih godina može se naći u knjizi [RFC 2581]; opis najnovijeg razvoja u domenu TCP kontrole zagušenja možete naći u knjizi [Floyd 2001]. To stoje bilo dobro za Internet dok je većina TCP konekcija prenosila SMTP, FTP i Telnet saobraćaj ne mora biti dobro za današnji Internet u kojem dominira HTTP ili za Internet u budućnosti koji će podržavati ko zna kakve vrste usluga.

Potrebu da se TCP stalno razvija možemo da ilustrijemo razmatranjem TCP konekcija velike brzine koje su potrebne aplikacijama u "grid" računarskim sistemima [Foster 2002]. Na primer, razmotrite TCP konekciju sa segmentima od 1500 bajtova gde RTT iznosi 100 ms i pretpostavimo da kroz tu konekciju hoćemo da šaljemo podatke brzinom od 10 Gb/s. S obzirom na [RFC 3649], primećujemo da prema gornjem obrascu za propusni opseg u TCP-u, da bi se postigao propusni opseg od 10 Gb/s, prosečna veličina prozora zagušenja bi morala da iznosi 83 333 segmenata. To je veoma mnogo segmenata pa je velika verovatnoća da se neki od njih ne izgubi. Šta bi se dogodilo u slučaju gubitka? Ili, drugačije rečeno, koji deo prenenih segmenata može da se izgubi a da TCP algoritam kontrole zagušenja opisan u tabeli 3.3 ipak omogući željenu brzinu od 10 Gb/s? U domaćim zadacima za ovo poglavљje izведен je obrazac za propusni opseg TCP konekcije u funkciji učestalosti gubitaka ( $l$ ), trajanja povratnog puta ( $RTT$ ) i maksimalne veličine segmenta (MSS):

$$\text{prosečni propusni opseg konekcije} = \frac{1.22 \cdot MSS}{RTT \sqrt{l}}$$

Iz ovog obrasca se vidi da za postizanje propusnog opsega od 10 Gb/s algoritam za kontrolu zagušenja današnjeg TCP-a ne može da toleriše verovatnoću gubitka segmenata veću od  $2 \cdot 10^{-10}$  (što odgovara jednom događaju gubitka na svakih 5 000 000 000 segmenata) a to je veoma mala verovatnoća. Ta činjenica je mnoge istraživače usmerila na istraživanje novih verzija TCP-a specifično projektovanih za takva okruženja sa veoma velikim brzinama; opise tih pokušaja možete pronaći u knjizi [Jin 2004; Kellz 2003].

#### 3.7.1 Fer ponašanje

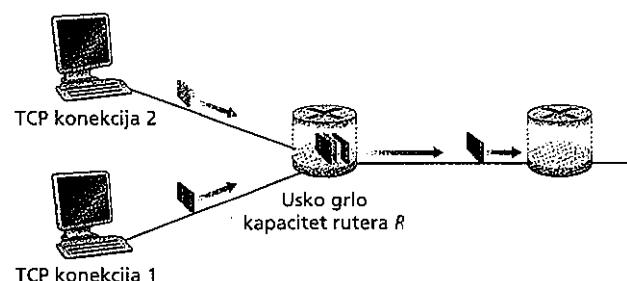
Razmotrimo KTCP konekcija od kojih svaka ima drugačiju putanju sa kraja na kraj, ali sve prolaze kroz link koji predstavlja usko grlo sa brzinom prenosa od  $R$  b/s. (Link smatramo uskim grlom ako taj link ima najmanji kapacitet prenosa za sve

konekcije, a svi ostali linkovi u poređenju sa njim nisu zagušeni i imaju dovoljan kapacitet prenosa.) Prepostavimo da svaka konekcija prenosi veliku datoteku i da kroz link koji predstavlja usko grlo ne prolazi nikakav UDP saobraćaj. Za mehanizam kontrole zagušenja kažemo daje fer ako je prosečna brzina prenosa svake konekcije približno  $R/K$ ; tj. ako svaka konekcija dobija podjednak deo propusnog opsega linka.

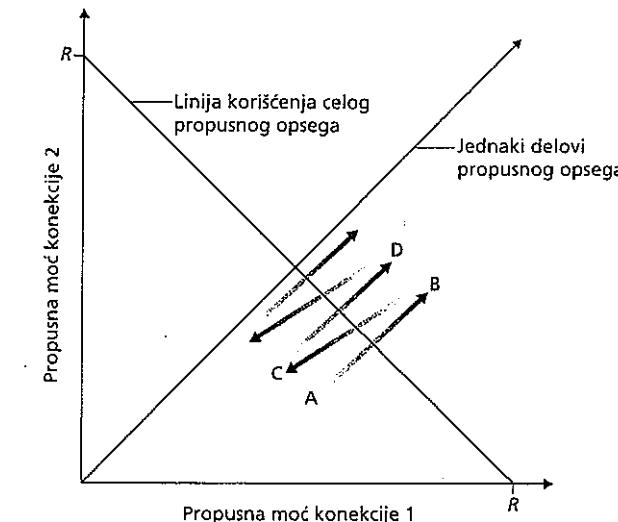
Da lije TCP-ov algoritam aditivnog uvećanja i multiplikativnog smanjenja fer, pogotovo ako uzmemo u obzir da se različite TCP konekcije pokreću u različito vreme i u nekom određenom trenutku mogu da imaju različite veličine prozora? Knjiga [Chiu 1989] sadrži elegantno i lako shvatljivo objašnjenje o tome kako TCP-ova kontrola zagušenja teži da svim TCP konekcijama koje se nadmeću za propusni opseg linka koji predstavlja usko grlo obezbedi jednak udio tog propusnog opsega.

Razmotrimo jednostavan slučaj kada dve TCP konekcije dele jedan jedini link sa brzinom prenosa  $R$ , kao što je prikazano na slici 3.52. Prepostavimo da obe konekcije imaju istu vrednost MSS i RTT (pa tako imaju istu veličinu prozora zagušenja i takođe istu propusnu moć), da treba da šalju velike količine podataka, a da kroz taj zajednički link ne prolaze druge TCP konekcije niti UDP datagrami. Osim toga, zanemarićemo TCP-ovu fazu sporog početka i prepostaviti da TCP konekcije stalno funkcionisu u režimu izbegavanja zagušenja (aditivno povećavanje, multiplikativno smanjivanje).

Na slici 3.53 učrtane su propusne moći koje su ostvarile dve TCP konekcije. Ako TCP treba da deli propusni opseg linka podjednako na obe konekcije, ostvarena propusna moć našla bi se oko simetrale od 45 stepeni (jednaki delovi propusnog opsega) koja počinje iz koordinatnog početka. U idealnom slučaju, zbir obe propusne moći trebalo bi da iznosi  $R$ . (U svakom slučaju, nije baš poželjno da obe konekcije dobiju podjednake delove kapaciteta linka a da ti delovi budu jednak nuli!) Prema tome, cilj bi trebalo da bude da se ostvarene propusne moći nadu negde blizu



Slika 3.52 ♦ Dve TCP konekcije sa zajedničkim linkom koji predstavlja usko grlo.



Slika 3.53 ♦ Propusna moć ostvarena u konekcijama 1 i 2

preseka linije „jednaki delovi propusnog opsega“ i linije „korišćenje celog propusnog opsega“ na slici 3.53.

Prepostavimo da su veličine TCP prozora takve da u datom trenutku konekcije 1 i 2 ostvaruju propusne moći označene tačkom A na slici 3.53. Postoje količina propusnog opsega linka koju ukupno troše te dve konekcije manja od  $R$ , neće doći do gubitaka i obe konekcije će, prema TCP-ovom algoritmu za izbegavanje zagušenja, povećati svoje prozore za 1 MSS kod svakog RTT-a. Prema tome, sveukupna propusna moć obe konekcije se kreće uz liniju od 45 stepeni (jednako povećanje kod obe konekcije), počevši od tačke A.. U jednom trenutku, propusni opseg linka koji ukupno troše obe konekcije preći će  $R$ , pa će doći do gubitka paketa. Uzmimo da u konekcijama 1 i 2 dođe do gubitka paketa kada one ostvare propusne moći označene tačkom B. Konekcije 1 i 2 zatim smanjuju svoje prozore za faktor 2. Tako dobijene ostvarene propusne moći označene su tačkom C, na polovini vektora koji počinje u tački B i završava se u koordinatnom početku. Pošto je ukupno utrošeni propusni opseg u tački C manji od  $R$ , obe konekcije ponovo povećavaju propusne moći duž linije pod uglom od 45 stepeni koja počinje od tačke C. U jednom trenutku, ponovo će doći do gubitka, na primer u tački D, pa će obe konekcije ponovo smanjiti prozore za faktor 2, i tako dalje. Propusni opseg koji ostvaruju obe konekcije u krajnjoj liniji kreće se duž linije jednakih delova propusnog opsega. Obe konekcije će težiti tom ponašanju bez obzira na to gde se one nalaze u dvodimenzionalnom prostoru! Mada

se ovaj scenario zasniva na nizu idealizovanih prepostavki, on ipak daje osećaj o tome kako TCP postiže da konekcije dobiju podjednake delove propusnog opsega.

U našem idealizovanom scenaru prepostavili smo da kroz link koji predstavlja usko grlo prolaze samo TCP konekcije, da konekcije imaju istu vrednost RTT i da je jednom paru računara i odredišta pridružena samo jedna TCP konekcija. U praksi se ti uslovi obično ne ostvaruju, pa aplikacije klijent/server mogu da dobiju veoma različite delove propusnog opsega linka. Konkretno, pokazalo se da kada više konekcija deli zajedničko usko grlo, sesije sa manjim RTT-om brže uspevaju da prigrabe dostupni propusni opseg na tom linku čim se on oslobodi (tj. brže otvaraju svoje prozore zagušenja) i tako ostvaruju veću propusnu moć od konekcija sa većim RTT-om [Lakshman 1997].

#### Fer ponašanje i UDP

Upravo smo videli kako TCP kontrola zagušenja uređuje brzinu prenosa putem mehanizma sa prozorom zagušenja. Mnoge multimedijalne aplikacije, kao što je Internetovo telefoniranje ili video konferencije, se upravo iz tog razloga ne izvršavaju preko TCP-a - one ne žele da se njihova brzina prenosa guši, čak i kada je mreža veoma zagušena. Umesto toga, ove aplikacije više vole da se izvršavaju preko UDP-a, koji nema ugradenu kontrolu zagušenja. Kada se izvršavaju preko UDP-a, aplikacije mogu da ubacuju audio i video konstantnom brzinom u mrežu i povremeno gube pakete, a ne da smanjuju brzinu na „fer“ nivo u trenucima zagušenja da ne bi gubile pakete. Iz perspektive TCP-a, multimedijalne aplikacije koje se izvršavaju preko UDP-a nisu fer - one ne saraduju sa drugim konekcijama niti podešavaju svoju brzinu prenosa na odgovarajući način. Danas veliko područje istraživanja predstavlja razvoj mehanizama za kontrolu zagušenja na Internetu kojim bi se sprečilo da UDP saobraćaj potpuno blokira propusnu moć Interneta [Floyd 1999; Floyd 2000; Kohler2004].

#### . Fer ponašanje i paralelne TCP konekcije

Ali, čak i kada bismo mogli da primoramo UDP saobraćaj na fer ponašanje, problem fer ponašanja još ne bi bio potpuno rešen zato što ništa ne sprečava TCP aplikacije da koriste više paralelnih konekcija. Na primer, Čitači Weba često koriste više paralelnih TCP konekcija za prenos više objekata jedne veb stranice. (U većini čitača se tačan broj višestrukih konekcija može konfigurisati.) Kada jedna aplikacija koristi više paralelnih konekcija, ona u zagušenom linku dobija veći deo propusnog opsega. Na primer, razmotrite link brzine  $R$  koji podržava devet aktivnih klijent/server aplikacija, tako da svaka aplikacija koristi jednu TCP konekciju. Ako se priključi jedna nova aplikacija i takođe koristi jednu TCP konekciju, svaka će aplikacija dobijati približno istu brzinu slanja od  $R/10$ . Ali, ako ova nova aplikacija koristi 11 paralelnih TCP konekcija, tada će njoj biti dodeljeno više od  $R/2$ , što nije fer. Pošto je veb saobraćaj veoma prisutan na Internetu, višestruke paralelne konekcije nisu neuobičajene.

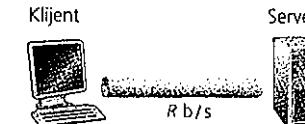
#### 3.7.2 Modeliranje TCP kašnjenja

Ovo poglavlje zaključujemo nekim jednostavnim modelima za izračunavanje vremena potrebnog da TCP pošalje neki objekat (kao što je slika, tekstualna datoteka ili MP3). Za dati objekat definiše se vreme čekanja kao vreme od trenutka kada klijent inicira TCP konekciju do trenutka kada primi zatraženi objekat u celosti. Modeli koji su ovde predstavljeni pružaju značajan uvid u ključne komponente vremena čekanja, uključujući početno TCP sinhronizovanje, TCP-ov spori početak i vreme za prenos objekta.

Ova jednostavna analiza počinje od prepostavke da mreža nije zagušena, tj. da TCP konekcija koja prenosi objekat ne mora da deli propusni opseg linka sa drugim TCP ili UDP saobraćajem. Isto tako, da se ne bi zamagljivala glavna pitanja, analizu vršimo u kontekstu jednostavne mreže sa jednim linkom, kao sto je prikazano na slici 3.54. (Ovaj link bi mogao biti model jednog uskog grla na putanji sa kraja na kraj. Pogledajte takođe probleme za domaći zadatak gde se nalazi eksplicitno proširenje na slučaj sa više linkova.)

Takođe polazimo od sledećih prepostavki koje pojednostavljaju slučaj:

- ◆ Količina podataka koju pošiljalac šalje ograničava se jedino veličinom njegovog prozora zagušenja. (Prema tome, prijemne TCP privremene memorije su velike.)
- ◆ Paketi se ne gube niti se oštećuju, pa nema ponovnih slanja.
- ◆ Sva dodatna preopterećenja protokolskim zaglavljima - uključujući zaglavljaja TCP-a, IP-a i sloja veze - zanemarljiva su pa ih ignorisemo.
- ◆ Objekat (tj. datoteka) koji treba preneti sastoji se od celog broja segmenata veličine MSS (maksimalna veličina segmenta).
- ◆ Jedini paketi čije se vreme prenosa ne zanemaruje su paketi koji prenose TCP segmente maksimalne veličine. Poruke sa zahtevima, potvrde i segmenti za uspostavljanje TCP konekcije su mali, pa imaju zanemarljivo vreme prenosa.
- ◆ Početni prag u TCP mehanizmu za kontrolu zagušenja ima veliku vrednost koju prozor zagušenja nikada ne dostiže.



**Slika 3.54** ◆ Jednostavna mreža od jednog linka koja povezuje klijenta i server.

Takođe uvodimo sledeće označavanje:

- ◆ Veličina objekta koji treba preneti je  $O$  bitova.
- ◆ MSS (maksimalna veličina segmenta) iznosi  $S$  bitova (na primer, 536 bajta).
- ◆ Brzina prenosa na linku od servera do klijenta iznosi  $R$  b/s.

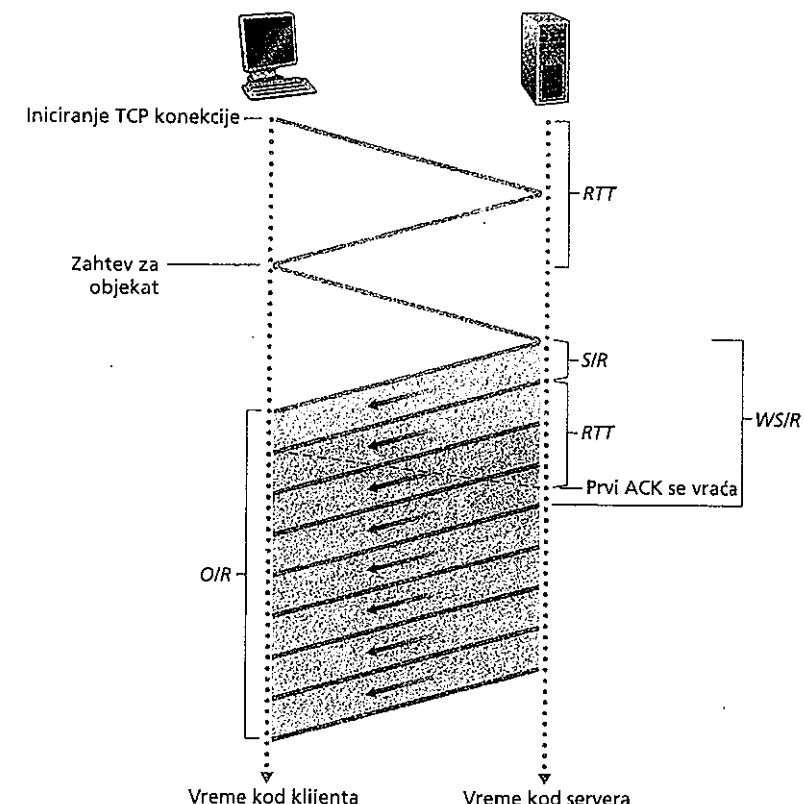
Pre nego što počnemo stvarnu analizu, pokušajmo donekle da ste knemo osećaj za problem. Razmotrimo koliko bi bilo vreme čekanja da nema ograničenja prozora zagušenja, tj. kada bi serveru bilo dozvoljeno da šalje segmente jedan za drugim sve dok ne pošalje ceo objekat. Da bismo odgovorili na ovo pitanje, prvo obratimo pažnju na to daje za iniciranje TCP konekcije potreban jedan RTT. Nakon toga, klijent šalje zahtev za objekat (koji se šlepaju na trećem segmentu TCP-ovog sinhronizovanja u tri koraka). Posle ukupno dva RTT-a, klijent počinje da prima podatke od servera. Klijent prima podatke od servera u periodu od  $O/R$ , vreme potrebno serveru da pošalje ceo objekat. Prema tome, u slučaju kada prozor zagušenja ne ograničava brzinu, ukupno vreme čekanja je  $2 \text{RTT} + O/R$ . Ovo predstavlja donju granicu; procedura sporog početka sa njenim dinamičkim prozorom zagušenja svakako će produžiti ovo vreme čekanja.

#### Statički prozor zagušenja

Mada TCP koristi dinamički prozor zagušenja, poučno je prvo analizirati slučaj sa statičkim prozorom zagušenja. Neka  $W$ , pozitivan ceo broj, označava statički prozor zagušenja fiksne veličine. Kod statičkog prozora zagušenja serveru se ne dozvoljava da ima više od  $W$  nepotvrđenih zaostalih segmenata. Kada server primi zahtev od klijenta, on odmah šalje klijentu  $W$  segmenata jedan za drugim. Server nakon toga šalje u mrežu po jedan segment za svaku potvrdu koju primi od klijenta. Server nastavlja da šalje po jedan segment za svaku potvrdu dok ne pošalje sve segmente iz  $W$  objekta. Potrebno je razmotriti dva slučaja:

1.  $WS/R > RTT + S/R$ . U ovom slučaju, server prima potvrdu za prvi segment prvog prozora pre nego što dovrši prenos prvog prozora.
2.  $WS/R < RTT + S/R$ . U ovom slučaju, server napre prenese količinu segmenata dovoljnu za prvi prozor pa tek potom prima potvrdu za prvi segment tog prozora.

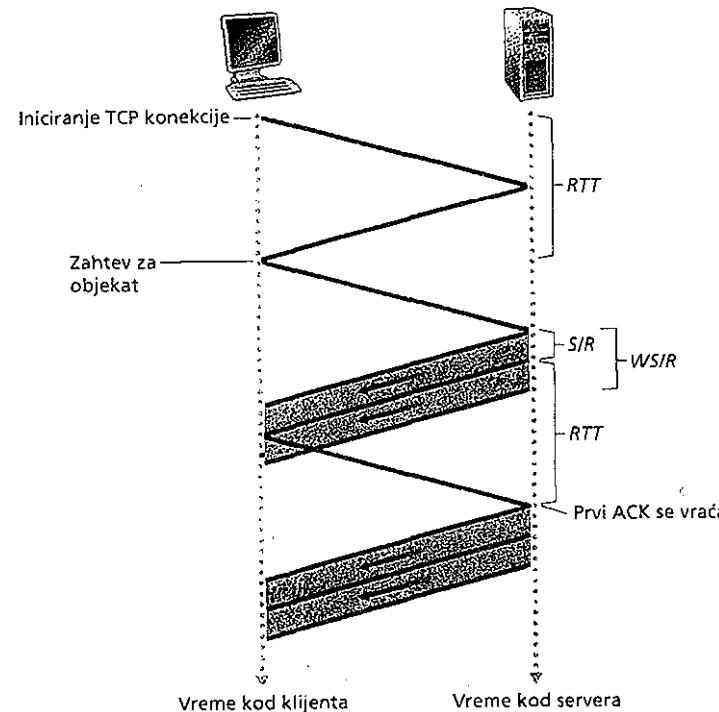
Razmotrimo najpre prvi slučaj, prikazan na slici 3.55. Na ovoj slici je veličina prozora  $W = 4$  segmenata. Jedan RTT je potreban za iniciranje TCP konekcije. Nakon toga, klijent šalje zahtev za objekat (koji se šlepaju na trećem segmentu TCP-ovog sinhronizovanja u tri koraka). Nakon ukupno dva RTT-a, klijent počinje da prima podatke od servera. Segmenti dolaze povremeno od servera svakih  $S/R$  sekundi, a



**Slika 3.55** ♦ Slučaj kada je  $WS/R > RTT + S/R$ .

klijent potvrđuje svaki segment koji primi. Kada server primi prvu potvrdu pre nego što dovrši slanje onoliko segmenata koliko staje u prozoru, server nastavlja da šalje segmente i nakon što popuni prvi prozor. S obzirom na to da potvrde stižu do servera svakih  $S/R$  sekundi od trenutka kada stigne prva potvrda, server neprekidno šalje segmente dok ne pošalje ceo objekat. Prema tome, pošto server počne da šalje objekat brzinom  $R$ , on i nastavlja da ga šalje istom brzinom dok ne prenese ceo objekat. Znači, vreme čekanja iznosi  $2 \text{RTT} + O/R$ .

Razmotrimo sada drugi slučaj koji je prikazan na slici 3.56. Na ovoj slici je veličina prozora  $W \sim 2$  segmenta. I opet, nakon ukupno dva  $RTT$ -a klijent počinje da prima segmente sa servera. Ti segmenti stižu svakih  $S/R$  sekundi, a klijent potvrđuje svaki segment koji primi od servera. Ali, sada server završava prenos prvog prozora pre nego što od klijenta stigne prva potvrda. Prema tome, pošto pošalje prozor, server mora da se zaustavi i sačeka neku potvrdu da bi nastavio sa slanjem. Kada potvrda na kraju stigne, server šalje klijentu novi segment. Nakon prve potvrde dolazi pun prozor potvrda u razmacima od po  $S/R$  sekundi. Za svaku od ovih potvrda server šalje tačno jedan segment. Prema tome, server se naizmenično nalazi u dva stanja: stanju slanja tokom kojeg prenosi  $W$  segmenata i stanju čekanja, kada ništa ne šalje.



**Slika 3.56** ♦ Slučaj kada je  $WS/R < RTT + S/R$ .

i čeka na potvrde. Vreme čekanja jednako je  $2 RTT + O/R + (K - 1) [S/R + RTT - WS/R]$ . Da bismo odredili vreme koje server provodi u stanju čekanja neka  $K$  bude broj prozora podataka koji pokrivaju objekat; tj.  $K = O/WS$  (ako  $O/IVS$  nije ceo broj treba zaokružiti  $K$  naviše do prvog celog broja). Server se nalazi u stanju čekanja između prenosa pojedinačnih prozora, tj.  $K - 1$  puta a svako čekanje traje  $RTT - (W-1)S/R$  (slika 3.56). Prema tome, za drugi slučaj je

$$\text{vreme čekanja} = 2RTT + O/R + (K - 1) [S/R + RTT - WS/R].$$

Sastavljanjem oba izraza dobijamo

$$\text{vreme čekanja} = 2RTT + O/R + (K - 1) [S/R + RTT - WS/R]^+$$

gde je  $f_{xy} = \max(A, 0)$ . Obratite pažnju na to da vreme čekanja ima tri komponente:  $2RTT$  za uspostavljanje TCP konekcije, za zahtev i početak primanja objekta;  $O/R$ , vreme potrebno serveru da prenese objekat; i, na kraju,  $(K - 1) [S/R + RTT - WS/R]^+$ , vreme koje server provede u čekanju.

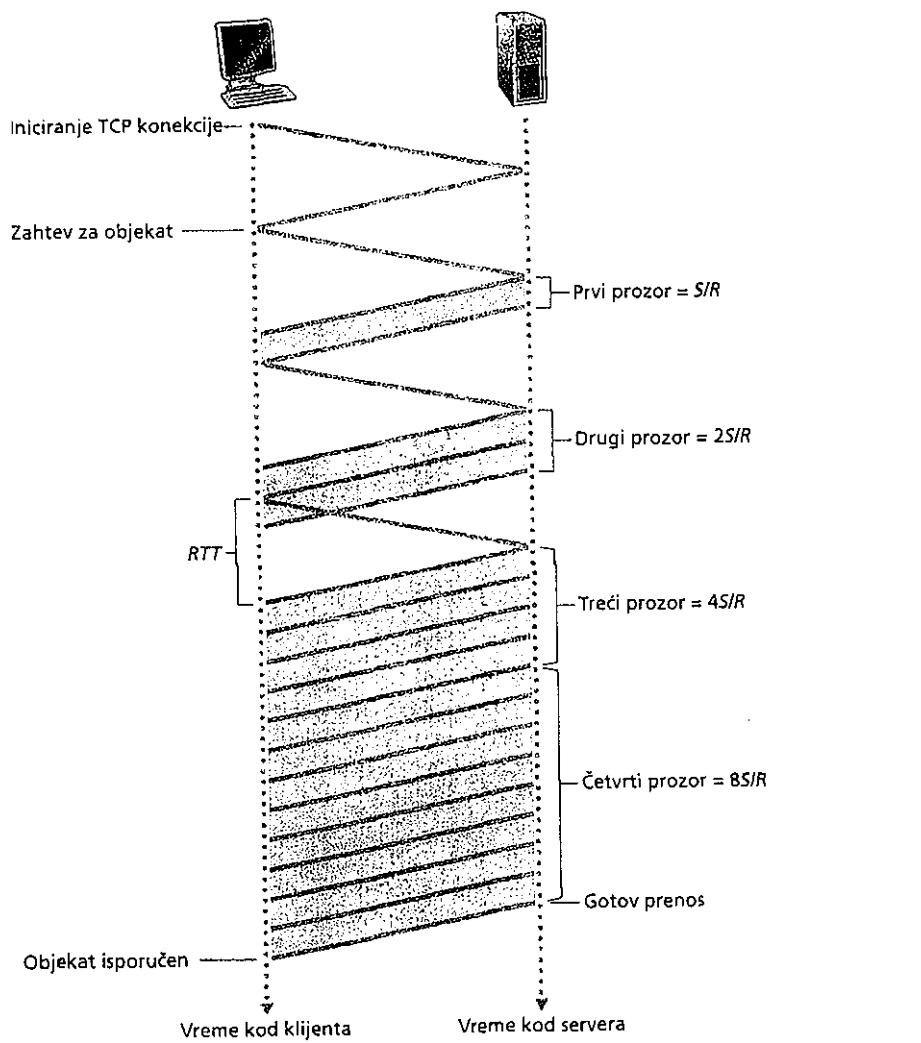
Ovim zaključujemo analizu statičkih prozora. Analiza za dinamičke prozore koja sledi je složenija, ali postoji paralela sa ovom analizom statičkih prozora.

#### Dinamički prozor zagušenja

Sada ćemo u modelu vremena čekanja uzeti u obzir TCP-ov dinamički prozor zagušenja. Znate da server počinje sa prozorom zagušenja od jednog segmenta i šalje klijentu jedan segment. Kada primi potvrdu za taj segment, on povećava prozor zagušenja na dva segmenta i šalje klijentu dva segmenta (u razmaku od  $S/R$  sekundi). Kada primi potvrde za ta dva segmenta, povećava prozor zagušenja na četiri segmenta i šalje klijentu četiri segmenta (opet u razmaku od  $S/R$  sekundi). Postupak se nastavlja tako što se prozor zagušenja udvostručava za svaki  $RTT$ . Dijagram vremenskog rasporeda za TCP prikazan je na slici 3.57.

Obratite pažnju na to daje  $O/S$  broj segmenata u objektu; u gornjem dijagramu je  $O/S = 15$ . Razmotrite broj segmenata u svakom prozoru. Prvi prozor sadrži jedan segment, drugi prozor sadrži dva segmenta, a treći prozor sadrži četiri segmenta. U opštem slučaju,  $k$ -ti prozor sadrži  $2^{k-1}$  segmenata. Neka  $K$  bude broj prozora koji pokrivaju objekat; u prethodnom dijagramu,  $K = 4$ . U opštem slučaju možemo izraziti  $K$  zavisno od  $O/S$  na sledeći način:

$$K = \min \{k : 2^0 + 2^1 + \dots + 2^{k-1} \geq O/S\}$$



**Slika 3.57** ♦ Vremenski raspored TCP-a tokom sporog početka

$$\begin{aligned}
 &= \min \{ k : 2^k - 1 > 0/S \} \\
 &= \min \{ k : k > \log_2(0/S + 1) \} \\
 &= T \log_2(0/S + 1)
 \end{aligned}$$

Pošto prenese pun prozor podataka, server može da stane (tj. prestane da prenosi) dok čeka na potvrdu. Na slici 3.57 server staje nakon što prenese prvi i drugi prozor, ali ne i nakon trećeg. Sada ćemo izračunati ukupno vreme stajanja nakon prenosa  $/c$ -tog prozora. Od trenutka kada server počne da prenosi  $/c$ -ti prozor do trenutka kada primi potvrdu za prvi segment u prozoru, protekne  $S/R + RTT$ . Vreme za prenos  $k$ -og prozora iznosi  $(S/R)2^{k-1}$ . Vreme stajanja je razlika između te dve vrednosti, to jest

$$\{S/R + RTT - 2^{k-1}(S/R)\}^+$$

Server može eventualno da stoji nakon prenosa svakog od prvih  $K$  prozora. (Server završava zadatak nakon prenosa AT-tog prozora.) Sada možemo izračunati vreme čekanja za prenos cele datoteke. Vreme čekanja ima tri komponente:  $2RTT$  za uspostavljanje TCP konekcije i zahtevanje datoteke;  $O/R$ , vreme za prenos objekta; i zbii svih vremena stajanja. Prema tome,

$$\text{vreme čekanja} = 2RTT + \frac{O}{R} + \sum_{k=1}^{K-1} \left[ \frac{S}{R} + RTT - 2^{k-1} \frac{S}{R} \right]^+$$

Citalac bi trebalo da uporedi gornju jednačinu sa jednačinom za vreme čekanja kod statičkog prozora zagušenja; svi elementi su potpuno jednaki, osim što se  $WS/R$  iz statičkog prozora zamjenjuje sa  $2^{k-1}(O/R)$  za dinamički prozor. Da bismo dobili sve-denijski izraz za vreme čekanja, neka  $Q$  predstavlja broj intervala stajanja na serveru kada bi se objekat sastojao od beskonačnog broja segmenata. Da bismo napravili izvođenje slično onome za  $K$  (pročitajte probleme za domaći zadatak), dobijamo:

$$Q = \left\lceil \log_2 \left( 1 + \frac{RTT}{S/R} \right) \right\rceil$$

Stvarni broj čekanja kod servera je  $P = \min \{ Q, K \}$ . Na slici 3.57 to je  $P = Q = 2$ . Kombinovanjem gornjih jednačina (pročitajte probleme za domaći zadatak) dobijamo sledeći izraz za vreme čekanja u zatvorenom obliku:

$$\text{vreme čekanja} = 2RTT + O/R + P[RTT + S/R] - (2^P - 1)S/R.$$

Prema tome, da bi se izračunato vreme čekanja, moramo jednostavno da izračunamo  $K \leq Q$ , odredimo  $P = \min\{Q, K-1\}$  i tako dobijeno  $P$  uvrstimo u gornji obrazac.

Zanimljivo je uporediti TCP vreme čekanja sa vremenom čekanja do kojeg bi došlo kada ne bi postojala kontrola zagušenja (tj. kada ne bi bilo ograničenja koje nameće prozor zagušenja). Ako nema kontrole zagušenja, vreme čekanja je  $2 RTT + O/R$ , koje definišemo kao *minimalno vreme čekanja*. Sada se lako može pokazati daje:

$$\text{vreme čekanja} / \text{minimalno vreme čekanja} \leq 1 + P/\{[(O/R)/RTT] + 2\}.$$

Iz gornjeg obrasca vidimo da TCP-ov spori početak neće značajno povećati vreme čekanja ako je  $RTT \ll O/R$  tj. ako je vreme povratnog puta mnogo manje od vremena potrebnog da se prenese objekat.

Pogledajmo sada neke primere scenarija. U svim primerima uzimamo daje  $S = 536$  bajtova što je uobičajena podrazumevana vrednost za TCP. Koristićemo RTT od 100 ms što nije neobična vrednost kašnjenja unutar kontinenta ili među kontinentima preko umereno zagušenih linkova. Prvo razmotrite slanje relativno velikog objekta veličine  $O = 100$  kilobajta. Broj prozora za ovaj objekat je  $K = 8$ . U sledećoj tabeli ispitujemo efekat koji mehanizam sporog početka ima na vreme čekanja za različite brzine prenosa.

| $R$      | $O/R$  | $P$ | Minimalno vreme čekanja: | Vreme čekanja sa |
|----------|--------|-----|--------------------------|------------------|
|          |        |     | $O/R + 2 RTT$            | sporim početkom  |
| 28 kb/s  | 28,6 s | 1   | 28,8 s                   | 28,9 s           |
| 100 kb/s | 8 s    | 2   | 8,2 s                    | 8,4 s            |
| 1 Mb/s   | 800 ms | 5   | 1 s                      | 1,5 s            |
| 10 Mb/s  | 80 ms  | 7   | 0,28 s                   | 0,98 s           |

Iz gornje tabele vidimo da za veliki objekat spori početak dodaje značajno kašnjenje samo kod velike brzine prenosa. Ako je brzina prenosa mala, potvrde se relativno brzo vraćaju i TCP ubrzano dostiže maksimalnu brzinu. Na primer, kada je  $i^* = 100$  kb/s broj čekanja je  $P=2$  dok je broj prozora koji treba preneti  $> 8$ ; prema tome, server stoji samo nakon prva dva od ukupno 8 prozora. S druge strane, kada je  $R = 10$  Mb/s server čeka nakon svakog prozora, što značajno produžava kašnjenje.

Razmotrimo sada slanje jednog malog objekta od  $O = 5$  kilobajta. Broj prozora za ovaj objekat je  $K = 4$ . U sledećoj tabeli ispitujemo efekat mehanizma sporog početka za različite brzine prenosa.

| $R$      | $O/R$  | $P$ | Minimalno vreme čekanja: | Vreme čekanja sa |
|----------|--------|-----|--------------------------|------------------|
|          |        |     | $O/R + 2 RTT$            | sporim početkom  |
| 28 kb/s  | 1,43 s | 1   | 1,63 s                   | 1,73 s           |
| 100 kb/s | 0,4 s  | 2   | 0,6 s                    | 0,76 s           |
| 1 Mb/s   | 40 ms  | 3   | 0,24 s                   | 0,52 s           |
| 10 Mb/s  | 4 ms   | 3   | 0,20 s                   | 0,50 s           |

I ovde spori početak dodaje značajno kašnjenje kada je brzina velika. Na primer, kada je  $R = 1$  Mb/s, server stoji nakon svakog prozora, pa je vreme čekanja veće od dvostrukog minimuma.

Za veći RTT, efekat sporog početka postaje značajan za male objekte kod malih brzina prenosa. U sledećoj tabeli ispitujemo efekat mehanizma sporog početka za  $RTT = 1$  sekund  $i = 5$  kilobajta ( $K=4$ ).

| $R$      | $O/R$  | $P$ | Minimalno vreme čekanja: | Vreme čekanja sa |
|----------|--------|-----|--------------------------|------------------|
|          |        |     | $O/R + 2 RTT$            | sporim početkom  |
| 28 kb/s  | 1,43 s | 3   | 3,4 s                    | 5,8 s            |
| 100 kb/s | 0,4 s  | 3   | 2,4 s                    | 5,2 s            |
| 1 Mb/s   | 40 ms  | 3   | 2,0 s                    | 5,0 s            |
| 10 Mb/s  | 4 ms   | 3   | 2,0 s                    | 5,0 s            |

Zaključak bi bio da spori početak može značajno da poveća vreme čekanja kada je veličina objekta relativno mala a RTT relativno veliko. Nažalost, to se često događa na Webu.

### Jedan primer: HTTP

Da bismo primenili analizu vremena čekanja, izračunajmo sada vreme odgovora za veb stranicu koja se šalje preko nepostojanog HTTP-a. Prepostavimo da se stranica sastoji od jedne osnovne HTML stranice i Mreferenciranih slika. Da bi primer ostao jednostavan, pretpostavimo da svaki od  $M+1$  objekata sadrži tačno  $O$  bitova.

Kada se koristi nepostojani HTTP, svaki objekat se prenosi nezavisno, jedan po jedan. Vreme odgovora za veb stranicu je, prema tome, zbir vremena čekanja pojedinačnih objekata. Prema tome,

$$\text{vreme odgovora} = (M+1) \left\{ 2RTT + \frac{O}{R} + P \left[ RTT + \frac{S}{R} \right] - (2^P - 1) \frac{S}{R} \right\}$$

Obratite pažnju na to da vreme odgovora za nepostojani HTTP ima oblik:

$$\text{Vreme odgovora} = (M + 1) \text{ O/R} + 2(M + !) \text{ RTT} + \text{vreme čekanja zbog sporog početka TCP-a za svaki od } M + 1 \text{ objekata.}$$

Jasno je da ako na veb stranici ima mnogo objekata i ako je RTT veliko, nepostojani HTTP će imati loše vreme odgovora. U problemima za domaći zadatak istražićemo vreme odgovora za druge transportne šeme HTTP-a. Čitalac bi mogao da pogleda i [Heidemann 1997; Cardwell 2000] gde se mogu naći odgovaraće analize.

## 3.8 Rezime

Ovo poglavlje počeli smo razmatranjem usluga koje protokoli transportnog sloja mogu da obezbede mrežnim aplikacijama. TJ jednom krajnjem slučaju, protokol transportnog sloja može biti veoma jednostavan i nuditi aplikacijama samo osnovne usluge, tj. multipleksiranje/demultipleksiranje za procese koji komuniciraju. Internetov protokol UDP je primer takvog osnovnog protokola transportnog sloja. U suprotnoj krajnosti, protokol transportnog sloja može aplikacijama da obezbedi niz garancija kao što su pouzdana isporuka podataka, garantovanje vremena čekanja i garantovanje propusnog opsega. I pored toga, usluge koje transportni protokol može da ponudi često su ograničene protokolom mrežnog sloja koji se nalazi ispod njega. Ako protokol mrežnog sloja ne može segmentima transportnog sloja da garantuje vreme čekanja ili propusni opseg, tada ni protokol transportnog sloja ne može da garantuje vreme čekanja ili propusni opseg za poruke koje razmenjuju procesi.

U odeljku 3.4 naučili smo da protokol transportnog sloja može da obezbedi pouzdan transfer podataka čak i kada mrežni sloj ispod njega nije pouzdan. Videli smo da obezbeđivanje pouzdanog transfera podataka sadrži mnoge suptilne elemente, ali da se zadatak može obaviti pažljivim kombinovanjem potvrda, tajmera, ponovnih slanja i rednih brojeva.

Mada smo u ovom poglavlju obradili pouzdani transfer podataka, trebalo bi imati na umu da pouzdani transfer podataka može da obezbedi i protokol sloja veze ili protokol mrežnog, transportnog ili aplikacijskog sloja. Bilo koji od navedena četiri sloja u familiji protokola može da implementira potvrde, tajmere, ponovna slanja i redne brojeve i tako obezbedi pouzdan transfer podataka za sloj iznad sebe, U stvari, tokom godina su naučnici i inženjeri računarstva nezavisno projektivali i implementirali protokole sloja veze, mrežnog, transportnog i aplikacijskog sloja koji obezbeđuju pouzdani transfer podataka (mada su mnogi od tih protokola neprimenito nestali).

U odeljku 3.5 detaljno smo pregledali TCP, Internetov pouzdan protokol transportnog sloja sa konekcijom. Naučili smo daje TCP složen, da obuhvata upravljanje konekcijom, kontrolu toka i procenu vremena povratnog puta, kao i pouzdani transfer podataka. TCP je u suštini složeniji od našeg opisa - namerno nismo opisali

šivali niz TCP ispravki, prepravki i poboljšanja koji su implementirani u različitim verzijama TCP-a. Sva ta složenost, međutim, sakrivena je za mrežnu aplikaciju. Ako klijent želi da sa jednog računara pouzdano pošalje podatke serveru na drugom računaru, on jednostavno otvara TCP soket prema serveru i ubacuje podatke u taj soket. Aplikacija klijent/server živi u blaženom neznanju o složenosti TCP-a.

U odeljku 3.6 ispitali smo kontrolu zagušenja u opštem slučaju, a u odeljku 3.7 smo pokazali kako TCP implementira kontrolu zagušenja. Naučili smo daje kontrola zagušenja obavezna za dobrobit mreže. Kad ne bi bilo kontrole zagušenja, na mreži bi lako nastao zastoj saobraćaja tako da bi malo ili nimalo podataka moglo da se prenese sa kraja na kraj. U odeljku 3.7 naučili smo da TCP implementira mehanizam kontrole zagušenja s kraja na kraj koji aditivno povećava brzinu prenosa kada se oceni da je putanja TCP konekcije prohodna, a multiplikativno smanjuje brzinu prenosa kada dođe do gubitaka. Taj mehanizam takođe teži da svakoj TCP konekciji koja prolazi kroz zagušeni link dodeli podjednak deo propusnog opsega linka. Takođe smo donekle detaljno proučili uticaj uspostavljanja TCP konekcije i sporog početka na ukupno vreme čekanja. Primetili smo da u mnogim važnim scenarijima samo uspostavljanje konekcije i spori početak značajno doprinose kašnjenju s kraja na kraj. Ponovo naglašavamo da iako se TCP kontrola zagušenja godinama razvijala, ona ostaje područje intenzivnog istraživanja i verovatno će se i dalje razvijati tokom sledećih godina.

U poglavlju 1 smo rekli da se računarska mreža može podeliti na „periferija mreže“ i na Jezgro mreže“. Rub mreže obuhvata sve što se događa u krajnjim sistemima. Pošto smo obradili aplikacijski sloj i transportni sloj, naš opis periferije mreže je završen. Vreme je da istražimo jezgra mreže! To putovanje počinje u sledećem poglavlju gde ćemo istraživati mrežni sloj, a nastavlja se u poglavlju 5 gde ćemo istražiti sloj veze.

## Domaći zadatak: problemi i pitanja Poglavlje 3

### Kontrolna pitanja ODEUCI 3.1-3.3

1. Uzmimo TCP konekciju između računara A i računara B. Pretpostavimo da TCP segmenti koji putuju od računara A prema računaru B imaju broj izvornog porta x, a broj odredišnog porta y. Koji su brojevi izvornog i odredišnog porta za segmente koji putuju od računara B prema računaru A?
2. Opišite zašto bi programer odlučio da će aplikacija da se izvršava preko UDP-a, a ne preko TCP-a.
3. Da li je moguće da aplikacija dobije pouzdan transfer podataka čak i kada se izvršava preko UDP-a? Ako jeste, na koji način?

## ODEUAK 3.5

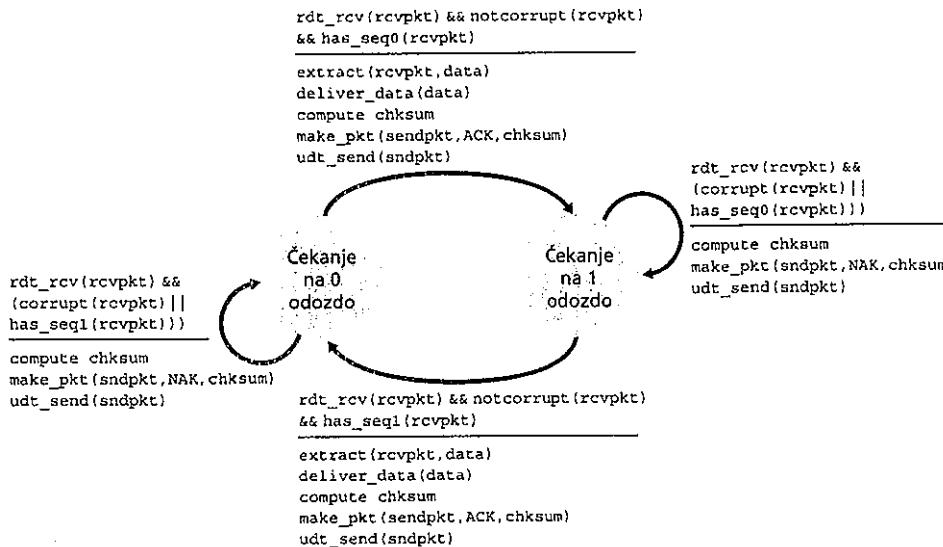
4. Tačno ili netačno?
  - a. Računar A šalje računaru B veliku datoteku preko TCP konekcije.  
Prepostavimo da računar B nema podataka koje bi slao računaru A.  
Računar B neće računaru A slati potvrde jer ne može da ih šlepne na podatke.
  - b. Veličina TCP prijemnog prozora RcvWindow se nikada ne menja tokom cele konekcije.
  - c. Prepostavimo da računar A šalje računaru B veliku datoteku preko TCP konekcije. Broj nepotvrđenih bajtova koje A šalje ne može preći veličinu prijemne privremene memorije.
  - d. Prepostavimo da računat A šalje veliku datoteku računam B preko TCP konekcije. Ako je redni broj jednog segmenta u ovoj konekciji  $m$ , tada će redni broj sledećeg segmenta obavezno biti  $m + 1$ .
  - e. TCP segment ima u svom zagлављu polje za RcvWindow.
  - f. Prepostavimo daje poslednji SampleRTT u jednoj TCP konekciji jednak 1 sekund. Tada će trenutna vrednost TimeoutInterval za tu konekciju svakako biti  $> 1$  sekund.
  - g. Prepostavimo da računar A šalje preko TCP konekcije računaru B jedan segment sa rednim brojem 38 i 4 bajta podataka. U tom segmentu broj potvrde mora biti 42.
5. Prepostavimo da računar A šalje računaru B dva TCP segmenta jedan za drugim preko TCP konekcije. Prvi segment ima redni broj 90; drugi ima redni broj 110.
  - a. Koliko podataka sadrži prvi segment?
  - b. Uzmimo da se prvi segment izgubi, ali da drugi segment stigne do B. Koji je broj potvrde koju će računar B poslati računani A?
6. Razmotrite primer sa Telnetom opisan u odeljku 3.5. Nekoliko sekundi nakon slova „C“ korisnik upiše slovo „R“. Koliko će se segmenata slati nakon upisivanja slova „R“ i šta se stavljaju u polja rednog broja i broja potvrde ovih segmenata?

## ODEUAK 3.7

7. Prepostavimo da u nekom linku koji predstavlja usko grlo sa brzinom  $R$  b/s postoje dve TCP konekcije. Svaka od njih šalje veliku datoteku (u istom smeru, preko linka koji predstavlja usko grlo). Prenos datoteka počinje u isto vreme. Kakvu brzinu prenosa bi TCP pokušao da dodeli svakoj od ovih konekcija?
8. Tačno ili netačno? Razmotrite kontrolu zagušenja u TCP-u. Kada kod pošiljaoca istekne tajmer, prag se postavlja na jednu polovicu njegove prethodne vrednosti.

## Problemi

1. Prepostavimo da klijent A inicira Telnet sesiju sa serverom S. Približno u isto vreme klijent B takođe inicira Telnet sesiju sa serverom S. Obezbedite moguće brojeve izvornih i odredišnih portova za:
  - a. segmente koji se šalju iz A prema S;
  - b. segmente koji se šalju iz B prema S;
  - c segmente koji se šalju iz S prema A;
  - d. segmente koji se šalju iz S prema B.
  - e. Ako su A i B različiti računari, da lije moguće da broj izvornog porta u segmentima iz A prema S bude isti kao u segmentima iz B prema S?
  - f. A kako stvar stoji ako su oni na istom računaru?
2. Analizirajte sliku 3.5. Koje su vrednosti izvornog i odredišnog porta u segmentima koji teku od servera prema klijentskim procesima? Koje su IP adrese u datagramima mrežnog sloja koji prenose segmente transportnog sloja?
3. UDP i TCP za svoje kontrolne zbirove koriste komplement jedinice. Prepostavimo da imate sledeća tri 8-bitna bajta: 01010101, 01110000, 01001100. Koji je komplement jedinice za zbir ovih 8-bitnih bajtova? (Obratite pažnju na to da iako UDP i TCP koriste 16-bitne reči kada izračunavaju kontrolni zbir, za ovaj problem se zahteva da radite sa sabirnim od 8-bitova.) Prikažite ceo postupak. Zašto UDP koristi komplement jedinice za zbir; tj. zašto jednostavno ne koristi sam zbir? Kako primalac otkriva greške u šemi sa komplementom jedinice? Da lije moguće da greška u jednom bitu ostane neprimеćena? A greška u dva bita?
4. Analizirajte motivaciju za izmenu protokola rdt2 .1. Pokažite da primalac, ako radi sa pošiljaocem prikazanim na slici 3.11, može dovesti pošiljaoca i primaoca u stanje mrtve petlje u kojoj svaki od njih čeka na dogadjaj koji nikada neće nastupiti.
5. U protokolu rdt 3 . 0, ACK paketi koji idu od primaoca ka pošiljaocu nemaju redne brojeve (mada imaju ACK polje koje sadrži redni broj paketa koji potvrduju). Zbog čega u ACK paketima nisu potrebni redni brojevi?
6. Skicirajte FSM za prijemnu stranu protokola rdt3 . 0.
7. Opišite rad protokola rdt 3.0 kada su paketi podataka i paketi potvrda oštećeni. Opis bi trebalo da bude sličan onome što vidite na slici 3.16.
8. Analizirajte kanal u kome može doći do gubitka paketa, ali je maksimalno kašnjenje poznato. Dopunite protokol rdt2 .1 tajm-autom kod pošiljaoca i ponovnim slanjem. Objasnite svojim recima zbog čega vaš protokol može pravilno da komunicira preko ovog kanala.



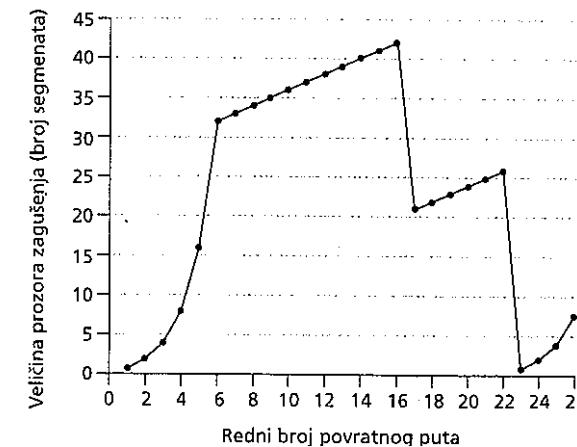
9. Strana pošiljaoca u protokolu rdt3.0 jednostavno zanemaruje (tj. ništa ne preduzima za) sve primljene pakete koji su pogrešni, ili ako je u paketu potvrde pogrešna vrednost u polju sa brojem potvrde. Prepostavimo da u takvom slučaju rdt3.0 ponavlja slanje trenutnog paketa podataka. Da li bi protokol i dalje funkcionišao? (Napomena: Ispitajte šta bi se dogodilo u slučaju da postoje samo greške sadržaja; da nema gubitaka paketa, ali da može doći do prevremenog tajm-autua. Razmotrite koliko puta bi se slao n-ti paket u graničnom slučaju kada  $n$  teži beskonačnosti.)
10. Razmotrite protokol naizmeničnih bitova (poznat kao „stani i čekaj“). Napravite dijagram u kojem će se videti da u slučaju kada mrežna konekcija između pošiljaoca i primaoca može da promeni redosled poruka (tj. da dve poruke koje se kreću po medijumu između pošiljaoca i primaoca mogu da zamene mesta), tada protokol naizmeničnih bitova neće pravilno funkcionišati (potrudite se da jasno pokažete u čemu se sastoji taj nepravilan rad). U dijagramu bi pošiljalac trebalo da bude na levoj, primalac na desnoj strani, a vremenska osa da ide vertikalno naniže. Prikažite razmenu podataka (D) i potvrda (A). Obavezno označite redni broj pridružen svakom segmentu podataka i potvrde.
11. Analizirajte protokol za pouzdani transfer podataka koji koristi samo negativne potvrde. Prepostavite da pošiljalac retko šalje podatke. Da li bi protokol samo sa NAK-ovima bio poželjniji od protokola koji koristi ACK? Zašto? Sada prepostavite da pošiljalac ima mnogo podataka za slanje, a da u konekciji sa

kraja na kraj retko dolazi do gubitaka. Da li bi u ovom drugom slučaju poželjniji bio protokol samo sa NAK-ovima od protokola koji koristi ACK? Zašto?

12. Analizirajte primer sa velikom razdaljinom priказан na slici 3.17. Kolika bi morala biti veličina prozora da bi iskorišćenost kanala bila veća od 90%?
13. Projektujte pouzdani protokol za transfer podataka sa cevovodnom obradom koji koristi samo negativne potvrde. Kojom brzinom bi vaš protokol reagovao na gubitak paketa kada je brzina pristizanja podataka do pošiljaoca mala? A kada je velika?
14. U opštem protokolu sa selektivnim ponavljanjem koji smo razmatrali u odeljku 3.4.4, pošiljalac prenosi poruku čim je ona dostupna (ako je unutar prozora) bez čekanja na potvrde. Prepostavimo sada da želimo protokol sa selektivnim ponavljanjem koji šalje poruke dve po dve. To jest, pošiljalac će poslati par poruka, a sledeći par će poslati tek kada sazna da su obe poruke iz prvog para pravilno primljene. Prepostavimo da kanal može da gubi poruke, ali da ih ne oštećuje niti im menja redosled. Projektujte protokol sa kontrolom grešaka za jednosmerni pouzdani transfer poruka. Napravite opis pošiljaoca i primaoca upotrebom konačnog automata. Opišite format paketa koji se šalju od pošiljaoca ka primaocu i obratno. Ako koristite pozive procedura osim onih iz odeljka 3.4 (na primer, udt\_send(), start\_timer(), rdt\_rcv() itd), jasno opišite njihove aktivnosti. Prikažite jedan primer (vremenski raspored kod pošiljaoca i primaoca) gde će se videti na koji način se vaš protokol oporavlja od gubitka paketa.
15. Analizirajte scenario u kojem računar A želi istovremeno da šalje poruke računarima B i C. Aje povezan sa B i C kanalom za difuzno emitovanje - kanal će preneti računarima B i C paket poslat iz računara A. Prepostavite da kanal za difuzno emitovanje koji povezuje A, B i C može nezavisno da gubi i oštećuje poruke (na primer, da poruka poslata iz A bude pravilno primljena u B, ali ne i u C). Projektujte protokol za kontrolu grešaka u stilu „stani i čekaj“ za pouzdani prenos paketa od A do B i C, takav da A neće prihvati novе podatke od gornjeg sloja, sve dok ne utvrdi da su i B i C pravilno primili trenutni paket. Napravite konačne automate za A i C. (Napomena: Konačni automat za B trebalo bi u osnovi da bude isti kao za C.) Osim toga, napravite i opis upotrebljenih formata paketa.
16. Uzmimo GBN protokol sa veličinom prozora pošiljaoca od 3 i rasponom rednih brojeva 1024. Prepostavimo da u trenutku  $t$  sledeći paket po redu koji primalac očekuje ima redni broj  $k$ . Prepostavite da medijima ne menja redosled poruka. Odgovorite na sledeća pitanja:
  - a. Koji su mogući skupovi rednih brojeva u prozoru pošiljaoca u trenutku  $t$ ? Objasnite odgovor.
  - b. Koje su sve moguće vrednosti polja ACK u svim mogućim porukama koje se u trenutku  $t$  vraćaju pošiljaocu? Objasnite odgovor.

17. Prepostavimo da imamo dva mrežna entiteta, A i B. B ima na raspolaganju poruke podataka koje će se slati prema A u skladu sa sledećim konvencijama: Kada A primi zahtev od gornjeg sloja da pribavi sledeću poruku sa podacima (D) od B, A mora da pošalje poruku zahteva (R) prema B kroz kanal od A do B. Tek kada B primi poruku R on može da pošalje poruku podataka (D) prema A kroz kanal od B do A. A bi trebalo da isporuči tačno jedan primerak svake poruke D gornjem sloju. Poruke R mogu da se izgube (ali ne i da se oštete) u kanalu od A do B; jednom poslate poruke D, uvek se pravilno isporučuju. Kašnjenje u oba kanala je nepoznato i promenljivo.
- Projektujte protokol (napravite konačni automat) koji obuhvata odgovarajuće mehanizme za kanal od A do B gde može doći do gubitaka i koji implementira predavanje poruka gornjem sloju u entitetu A, kao što je gore opisano. Upotrebite samo mehanizme koji su apsolutno neophodni.
18. Razmotrimo GBN protokole sa selektivnim ponavljanjem. Uzmimo daje prostor rednih brojeva veličine  $k$ . Odredite za svaki od ovih protokola najveći dozvoljeni prozor pošiljaoca koji će izbeći pojavljivanje problema kakvi se javljaju na slici 3.27.
19. Odgovorite na sledeća pitanja sa tačno ili netačno, ukratko obrazložite odgovor:
- Da li je u protokolu za selektivno ponavljanje moguće da pošiljalac primi ACK za paket van njegovog trenutnog prozora.
  - Da li je u GBN protokolu moguće da pošiljalac primi ACK za paket van njegovog trenutnog prozora.
  - Protokol sa naizmeničnim bitovima isto je što i protokol sa selektivnim ponavljanjem gde je veličina prozora pošiljaoca i primaoca jednaka 1.
  - Protokol sa naizmeničnim bitovima isto je što i GBN protokol gde je veličina prozora pošiljaoca i primaoca jednaka 1.
20. Razmotrite prenošenje ogromne datoteke od  $L$  bajtova sa računara A na računar B. Prepostavite da MSS iznosi 1460 bajtova.
- Koja je najveća vrednost  $i$  takva da se ne utroše svi TCP-ovi redni brojevi? Imajte na umu da polje iednih brojeva u TCP-u ima Četiri bajta.
  - Za  $L$  koji ste dobili pod (a), utvrđite koliko traje prenošenje datoteke. Prepostavite da se na svaki segment pre slanja dodaje ukupno 66 bajtova zaglavila transportnog, mrežnog i sloja veze podataka pre nego što se dobijeni paket pošalje preko linka od 10 Mb/s. Zanemarite kontrolu toka i kontrolu zagušenja kao da računar A može bez prekida da izbacuje segmente jedan za drugim.
21. Razmotrite TCP proceduru za procenu RTT-a. Prepostavite da je  $\alpha = 0.1$ . Neka  $\text{SampleRTT}_1$  bude najnoviji uzorak RTT-a,  $\text{SampleRTT}_2$  prvi prethodni uzorak RTT-a itd.

- Za daru TCP konekciju, prepostavite da su primljene četiri potvrde sa odgovarajućim uzorcima RTT-a:  $\text{SampleRTT}^1$ ,  $\text{SampleRTT}^2$ ,  $\text{SampleRTT}^3$  i  $\text{SampleRTT}^4$ . Izražite procenjeni EstimatedRTT pomoću četiri uzorka RTT-a.
  - Napravite obrazac za opšti slučaj sa  $n$  uzorka povratnih puteva.
  - Neka u obrascu (b)  $n$  teži beskonačnosti. Objasnite zašto se taj postupak za pronalaženje prošeka naziva eksponencijalni klizni prošek.
22. U odeljku 3.5.3 opisati smo TCP-ovo procenjivanje vremena povratnog puta. Šta mislite zbog čega TCP ne meri SampleRTT za ponovo poslate segmente?
23. Koji je odnos promenljive SendBase u odeljku 3.5.4 i promenljive LastByteRcvd u odeljku 3.5.5?
24. Koji je odnos promenljive LastByteRcvd u odeljku 3.5.5 i promenljive y u odeljku 3.5.4?
25. U odeljku 3.5.4 videli smo da TCP čeka da dobije tri identična ACK-a da bi pokrenuo brzo ponovno slanje. Šta mislite zbog čega su projektanti TCP-a odlučili da se brzo prenošenje ne pokreće već nakon prvog dupliranog ACK-a za segment?
26. Posmatrajte sliku 3.45(b). Ako  $\backslash_{\text{lim}}$  prede  $R/2$ , može li  $\backslash_{\text{az}}$  preći  $R/31$ ? Objasnite. Posmatrajte sada sliku 3.45(c). Ako  $X \backslash_{\text{az}}$  prede  $R/2$ , može li  $X \backslash_{\text{az}}$  da pređe  $R/4$  pod pretpostavkom da će se paket u prošeku dva puta proslediti od rutera ka primaocu? Objasnite.
27. Razmotrite sledeći dijagram veličine TCP prozora u funkciji vremena.



Pod pretpostavkom da se prikazano ponašanje događa u protokolu TCP Reno, odgovorite na sledeća pitanja (trebalo bi za svaki odgovor da destate kratko obrazloženje):

- a. Označite vremenske intervale u kojima TCP vrši spori početak.
- b. Označite vremenske intervale u kojima TCP izvršava izbegavanje zagušenja.
- c. Da li je gubitak segmenta nakon 16. povratnog puta otkriven na osnovu tri identična ACK-a ili na osnovu tajm-aut-a?
- d. Da li je gubitak segmenta nakon 22. povratnog puta, otkriven na osnovu tri identična ACK-a ili na osnovu tajm-aut-a?
- e. Koja je početna vrednost praga Threshold tokom prve povratne putanje?
- f. Koja je vrednost praga Threshold tokom 18. povratne putanje?
- g. Koja je vrednost praga Threshold tokom 24. povratne putanje?
- h. Tokom koje putanje se šalje 70. segment?
- i. Pod pretpostavkom da se gubitak paketa otkrije nakon 26. putanje na osnovu tri identična ACK-a, koje će biti vrednosti prozora zagušenja i praga Threshold?
28. Proučite sliku 3.53 na kojoj je prikazana konvergencija TCP-ovog algoritma aditivnog povećanja i multiplikativnog smanjenja. Prepostavite da umesto multiplikativnog smanjenja TCP umanjuje veličinu prozora za konstantan iznos. Da li bi za tako dobijeno aditivno povećanje, aditivno smanjenje težilo algoritmu jednakih delova? Obrazložite svoj odgovor dijagramom sličnim onom koji je dat na slici 3.53.
29. U odeljku 3.5.4 opisali smo udvostručavanje intervala za tajm-aut nakon događaja tajm-aut. Ovaj mehanizam predstavlja jednu vrstu kontrole zagušenja. Zastoje TCP-u pored ovog mehanizma „dvostrukog intervala za tajm-aut“ potreban i mehanizam kontrole zagušenja zasnovan na veličini prozora (kao stoje rečeno u odeljku 3.7)?
30. Računar A šalje ogromnu datoteku računaru B preko TCP koneksiјe. U toj koneksiјi nikad ne dolazi do gubitka paketa i tajmeri nikad ne isteknu. Označite sa  $R$  b/s brzinu prenosa linka koji povezuje računar A sa Internetom. Prepostavite da proces u računaru A može da šalje podatke u svoj TCP soket brzinom od  $S$  b/s, gde je  $5 = 10 \cdot R$ . Zatim prepostavite daje TCP-ova prijemna privremena memorija dovoljna da primi celu datoteku, a da otpremna privremena memorija može da primi samo jedan procenat datoteke. Kako se može sprečiti da proces u računaru A stalno predaje podatke u TCP soket brzinom od  $S$  b/s? TCP kontrolom toka? TCP kontrolom zagušenja? Nečim drugim? Objasnite.
31. Vratite se na idealizovani model za dinamiku stacionarnih stanja TCP-a. U vremenskom intervalu kada se brzina koneksiјe kreće od  $W/(2 \cdot RTT)$  do  $W/RTT$ , izgubio se samo jedan paket (pri samom kraju tog intervala).
- a. Pokažite da je učestalost gubitaka jednaka:
- $$L = \text{učestalost gubitaka} = \frac{1}{i} \cdot \frac{1}{8-4} = \frac{1}{4}$$

- b. Upotrebite dobijeni rezultat i pokažite da kada koneksiјa ima učestalost gubitaka paketa i tada njena prosečna brzina iznosi približno:

1,22-MSS

RTT-JI

32. U našem opisu budućeg TCP-a u odeljku 3.7 napomenuli smo da bi za postizanje propusnog opsega od 10 Gb/s TCP tolerisao verovatnoću gubitka segmenata od najviše  $2 \cdot 10^{-10}$  (što odgovara jednom događaju gubitka na svakih 5 000 000 000 segmenata). Pokažite kako se izvode vrednosti  $2 \cdot 10^{-10}$  i jedan da  $5 000 000 000$  polazeći od vrednosti RTT i MSS datih u odeljku 3.7. Koji bi bili podnošljivi gubici kada bi TCP morao da podrži koneksiјu od 100 Gb/s?
33. U našem opisu TCP kontrole zagušenja u odeljku 3.7 prepostavili smo da TCP pošiljalac uvek ima podatke za slanje. Razmotrite sada slučaj da TCP pošiljalac šalje veliku količinu podataka a zatim ostane besposlen (jer nema više podataka za slanje) u trenutku  $t_y$ . TCP ostaje besposlen relativno duže vreme a zatim hoće ponovo da šalje podatke u trenutku  $t_2$ . Kakve su prednosti i mane ako TCP, kada počne da šalje podatke u trenutku  $t_2$  upotrebi vrednosti CongWin i Treshold iz trenutka  $t_y$ ? Koju alternativu biste preporučili? Zašto?
34. Razmotrite slanje objekta veličine  $O = 100$  kilobajta od servera ka klijentu. Neka bude  $S = 536$  bajta i  $RTT = 100$  ms. Prepostavite da transportni protokol koristi statične prozore veličine  $W$ . (Odeljak 3.7.2.)
- a. Za brzinu prenosa od 28 kb/s odredite najmanje moguće vreme čekanja.  
Odredite minimalnu veličinu prozora kojom se postiže to vreme čekanja.
- b. Ponovite (a) za 100 kb/s,
- c. Ponovite (a) za 1 Mb/s,
- d. Ponovite (a) za 10 Mb/s.
35. Prepostavite da TCP tokom sporog početka kod svake primljene potvrde povećava prozor zagušenja za dva umesto za jedan. Tako bi se prvi prozor sastojao od jednog segmenta, drugi od tri, treći od devet segmenata itd. Postupkom prikazanim u odeljku 3.7.2:
- a. Izrazite  $K$  pomoću  $O$  i  $S$ .
- b. Izrazite  $Q$  pomoću  $RTT$ ,  $S$  i  $R$ .
- c; Izrazite vreme čekanja pomoću  $P - mm.(K - 1, Q)$ ,  $O$ ,  $R$  i  $RTT$ .
36. Uzmimo slučaj kada je  $RTT = 1$  sekund, a  $O = 100$  kbajta. Napravite dijagram (sličan dijagramima iz odeljka 3.7.2) u kojem se minimalno vreme čekanja ( $01 R = 2 RTT$ ) poređa sa vremenom čekanja uz spori početak za  $R = 28$  kb/s, 100 kb/s, 1 Mb/s i 10 Mb/s.
37. Tačno ili netačno?
- a. Ako se web stranica sastoji od tačno jednog objekta, tada nepostojane i postojane koneksiјe imaju tačno isto vreme odgovora.

- b. Uzmimo slanje jednog objekta veličine  $O$  od servera u čitač preko TCP-a. Ako je  $O > S$ , gde je  $S$  maksimalna veličina segmenta, server će bar jednom morati da čeka.
- c. Prepostavite da se veb stranica sastoji od deset objekata od kojih je svaki veličine  $O$  bitova. Za postojani HTTP, deo vremena odgovora koji otpada na  $RTT$  iznosi 20  $RTT$ .
- d. Prepostavite da se veb stranica sastoji od deset objekata od kojih je svaki veličine  $O$  bitova. Za nepostojani HTTP sa pet paralelnih konekcija, deo vremena odgovora koji otpada na  $RTT$  iznosi 12  $RTT$ .
38. U ovom problemu ćemo dopuniti neke od detalja u izvođenju obrasca za vreme čekanja iz odeljka 3.7.2. a. Izvedite formulu

$$Q = \left\lceil \log_2(1 + \frac{RTT}{S/R}) \right\rceil + 1$$

b. Upotrebite jednakost

$$\sum_{k=1}^p 2^{k-1} = 2^p - 1$$

za izvođenje obrasca

$$\text{Vreme čekanja} = 2RTT + \frac{O}{R} + P \left[ RTT + \frac{S}{R} \right] - (2^p - 1) \frac{S}{R}$$

39. U analizi dinamičkih prozora u odeljku 3.7.2 prepostavlja se da između servera i klijenta postoji jedan link. Ponovite analizu za  $T$  linkova između servera i klijenta. Prepostavite da na mreži nema zagušenja, tako da paketi nemaju kašnjenje zbog redova čekanja. Postoji, međutim, kašnjenje zbog čuvanja i prosleđivanja. Definicija za  $RTT$  ista je kao i ona koja je dala u odeljku o TCP kontroli zagušenja. (Napomena: Vreme od kad server pošalje prvi segment dok ne dobije potvrdu iznosi  $TS/R + RTT$ .)
40. Sećate li se diskusije na kraju odeljka 3.7.2 o vremenu odgovora za veb stranicu? Za slučaj nepostojanih konekcija utvrdite opšti izraz za *deo* vremena odgovora koji potiče od TCP-ovog sporog početka.

## Teze za diskusiju

- Razmotrite protok memorisanog audio sadržaja u realnom vremenu. Da li je logično da se aplikacija izvršava preko UDP-a ili TCP-a? Koji transportni protokol koristi RealNetworks? Zašto?

- Primenite prethodno pitanje na Microsoftove proizvode za protok signalu u realnom vremenu.
- U odeljku 3.7 napomenuli smo da klijent/server može „neuvidljivo“ da napravi mnogo istovremenih paralelnih konekcija. Šta se može učiniti da bi Internet bio zaista fer?
- Proučite u istraživačkoj literaturi pojam „TCP friendly“. Pročitajte takođe intervju sa Sali Flojd na kraju ovog poglavlja. Napišite na jednoj stranici rezime o pojmu TCP friendly.
- Na kraju odeljka 3.7.3 razmatrali smo činjenicu da aplikacija može da otvoriti više TCP konekcija i tako postigne veću propusnu moć (odnosno brži prenos podataka). Šta mislite, zašto više aplikacija ne pokušava da unapredi svoje performanse korišćenjem više konekcija? Šta bi se dogodilo kada bi sve aplikacije pokušale da unaprede svoje performanse korišćenjem više konekcija? Na koje teškoće bi naišao mrežni element koji bi pokušao da utvrdi da li neka aplikacija koristi više TCP konekcija?

## JU Programerski zadatak

### Implementiranje jednostavnog protokola za pouzdani transfer podataka

U ovom laboratorijskom programerskom zadatku pisaćete kod za pošiljaoca i primaoca transportnog sloja kojim se implementira jednostavan protokol za pouzdani transfer podataka. Postoje dve verzije ove vežbe, verzija protokola sa naizmeničnim bitovima i GBN verzija. Ova vežba će verovatno biti zanimljiva pošto će vaša implementacija veoma malo da se razlikuje od onog što bi se zahtevalo u realnoj situaciji.

Pošto verovatno nemate samostalne mašine (sa operativnim sistemom koji možete da menjate), vaš kod će morati da se izvršava u simuliranom hardver-sko-softverskom okruženju. Međutim, programski interfejs za vaše rutine, tj. kod koji poziva vaše entitete odozgo i odozdo veoma liči na ono što se zaista događa u okruženju UNIX. (Zaista, softverski interfejsi, opisani u ovom programerskom zadatku, mnogo su realniji od pošiljalaca i primalaca sa beskonačnim petljama koji se koriste u drugim tekstovima.) Simulira se takođe zaustavljanje i pokretanje tajmera, a tajmerski prekidi aktiviraće vašu rutinu za njegovu obradu.

## Laboratorija Ethereal: Istražite TCP

U ovoj vežbi ćete upotrebiti svoj čitač Weba da biste pristupili datoteci sa veb servera. Kao i u prethodnim Ethereal vežbama, koristićete Ethereal za hvatanje paketa koji stižu do vašeg računara. Za razliku od prethodnih vežbi, bićete takođe u stanju

da preuzmete trag paketa koji može da se čita u laboratoriji Ethereal sa veb servera sa kojeg ste preuzeeli datoteku. U tom serverskom tragu videćete pakete koji su nastali od vašeg pristupa veb serveru. Analiziraćete trage sa klijentske i server-ske strane i tako istražiti oba aspekta TCP-a. Posebno ćete proceniti performanse TCP konekcije između vašeg računara i veb servera. Pratićete ponašanje TCP-ovog prozora i izvoditi zaključke o gubljenju paketa, ponovnim slanjima, kontroli toka i kontroli zagušenja, kao i o procenjenom trajanju povratnog puta.

Kao i kod svih vežbi sa laboratorijom Ethereal, potpuni opis ove vežbe dostupan je na veb lokaciji ove knjige na adresi <http://www.awl.com/kurose-ross>.

## INTERVJU

### Sali Flojd

Sali Flojd je naučni istraživač u AT&T Centru za Internet istraživanja u ICSI (ACIRI), institutu posvećenom Internetu i pitanjima umrežavanja. Ona je poznata u računarskoj delatnosti po svom radu naprojektovanju intemetskih protokola, pogotovo pouzdanih i više znočnih, no projektovanju kontrole zagušenja [TCP], vremenskog rasporedivanja paketa (RED) i na analizi protokola. Sali je diplomirala sociologiju na Ka 1 informijskom univerzitetu Berkli, a magistrirala i doktorirala iz računarskih nauka na istom univerzitetu.



Kako ste odlučili da studirate računarske nauke?

Nakon diplome iz sociologije trebalo je da odlučim od Čega ću živeti: Na kraju sam dve godine učila elektroniku na lokalnom koledžu, a zatim deset godina radija u toj oblasti i u računarskoj nauci. To obuhvata i osam godina koje sam proveila kao sistemski inženjer za računare koji upravljaju železnicom Bay Area Rapid Transit. Kasnije sam odlučila da steknem više školskog znanja iz računarskih nauka i prijavila se na lokalne postdiplomske studije na Berkliju, na odeljenju za računarske nauke.

Zašto ste se odlučili na specijalizaciju iz oblasti umrežavanja?

Na postdiplomskim studijama sam se zainteresovala za teorijsku računarsku nauku. Prvo sam radila na analizi verovatnoće za algoritme, a kasnije na teoriji učenja računara. Takođe sam jedan dan mesečno radila u laboratoriji LBL (*Lawrence Berkeley Laboratory*), a moja kancelarija je bila preko puta kancelarije Vana Jakobsona koji je tada radio na TCP algoritmima za kontrolu zagušenja. Van me je pitao da li bih preko leta radila na analizama algoritama za neki mrežni problem vezan za synchronizaciju periodičnih poruka rutiranja. To me je zainteresovalo, pa sam tog leta to i radila.

Kada sam završila doktorsku tezu, Van mi je ponudio posao sa punim radnim vremenom na umrežavanju. Nisam tada planirala da 10 godina ostanem u umrežavanju, ali mi istraživanje mreža predstavlja veće zadovoljstvo od teorije računarskih nauka. Nalazim veće zadovoljstvo u svetu primene gde su posledice mog rada oplijivije.

Koji je bio vaš prvi posao u računarskoj industriji? Do čega je fo dovelo?

Moj prvi posao na računarima bio je u Železnicama BART (*Bay Area Rapid Transit*), od 1975. do 1982. gde sam radila na računarama koji upravljaju BART vozovima. Počela sam kao tehničar, održavala sam i popravljala različite distribuirane računarske sisteme vezane za upravljanje BART sistemom.

Tu je spadao jedan centralni računarski sistem i distribuirani sistemi mini-računara za kontrolu kretanja vozova; sistem DEC računara za prikazivanje reklama i određišta vozova na tablama u stanicama; kao i jedan sistem Modcomp računara za prikupljanje informacija sa naplatnih rampi. Poslednjih nekoliko godina u BART-u provela sam na zajedničkom projektu BART i laboratorija LBL na projektovanju i zamjeni zastarelog BART-ovog računarskog sistema za kontrolu vozova.

### Koji je najizazovniji deo vašeg posla?

Samо istraživanje predstavlja najizazovniji deo. Trenutno, radi se o projektovanju i istraživanju jednog novog mehanizma za kontrolu zagušenja sa kraja na kraj, na osnovu kontrole zagušenja zasnovane na jednačini. To je planirano ne kao zamena za TCP, već za jednoznačno emitovanje saobraćaja, kao sto je saobraćaj u realnom vremenu sa prilagodljivom brzinom kako bi se izbegle drastične promene brzine kada se brzina slanjanja preplovili zbog toga. Što je ispušten samo jedan paket. Kontrola zagušenja zasnovana na jednačini takođe je zanimljiva kao potencijalna osnova za kontrolu zagušenja višezačnog emitovanja. Više informacija se može naći na web stranici [http://www.psc.edu/networking/tcp\\_friendly.html](http://www.psc.edu/networking/tcp_friendly.html).

### Kako vi vidite budućnost umrežavanja i Interneta?

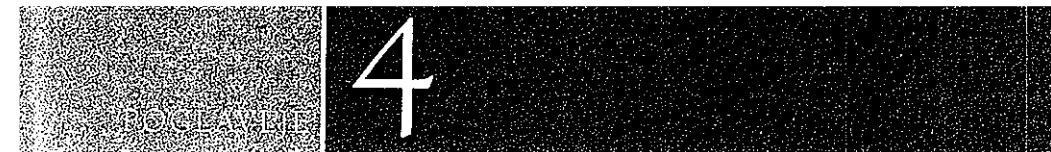
Jedna mogućnost je da ubočajeno zagušenje na koje nailazi saobraćaj na Internetu, postane manje izraženo kada se uključe mehanizmi sa cenama, a dostupni propusni opseg počne da raste brže od potraživanja. Mislim da je trend smanjenje zagušenja, mada ne izgleda nemoguće ni srednjoročno povećanje zagušenja sa povremenim kolapsima.

Budućnost samog Interneta ili Intemetove arhitekture nije mi sasvim jasna. Ima mnogo činilaca koji bi mogli da utiču na brze promene, pa je teško predvideti kako će se razvijati Internet ili njegova arhitektura, pa čak i predvideti koliko će taj razvoj moći uspešno da izbegne mnoge potencijalne zamke na tom putu.

### Koji su vas ljudi profesionalno inspirisali?

Ričard Karp, moj mentor na postdiplomskim studijama u suštini me je naučio šta je to istraživanje, a Van Jakobson, moj „voda grupe“ u LBL-u, zaslужan je za moj interes za umrežavanje i za veći deo mog shvatanja infrastrukture Interneta. Dejv Klark me je inspirisao jasnom vizijom arhitekture Interneta i svojom ulogom u razvoju te arhitekture putem istraživanja, pisanja i učešća u IETF-u i drugim javnim forumima. Debora Estrin me je inspirisala svojom koncentracijom i efikasnošću, kao i sposobnošću da svesno doneše odluku o tome što radi i zašto.

Jedan od razloga što sam uživala da radim na istraživanju mreža ovih 10 godina jeste to što ima toliko ljudi na tom polju koji mi se dopadaju, koje cenim i koji me inspirišu. Oni su inteligentni, vredno rade i veoma su posvećeni razvoju Interneta, obavljaju zadržavajući posao, a mogu da budu i dobro društvo za pivo i prijateljsko neslaganje (ili slaganje) na kraju dana punog sastanaka.



## Mrežni sloj

U prethodnom poglavlju smo videli da transportni sloj obezbeđuje različite vidove komunikacije od procesa do procesa oslanjajući se na uslugu komunikacije, od računara do računara koju obezbeđuje mrežni sloj. Naučili smo takođe da transportni sloj koristi uslugu mrežnog sloja ne znajući na koji način mrežni sloj implementira tu uslugu. Zato se možda pitate kakva je ta komunikacija od računara do računara, kako ona funkcioniše?

Predmet ovog poglavlja je upravo način na koji mrežni sloj implementira uslugu komunikacije od računara do računara. Videćemo da, za razliku od transportnog sloja, po deo mrežnog sloja postoji u svakom računaru i ruteru na mreži. Zato protokoli mrežnog sloja spadaju u najizazovnije (stoga i najzanimljivije!) u familiji protokola.

Mrežni sloj je ujedno jedan od naj složenijih slojeva u familiji protokola i zato je ovo obimna materija. Proučavanje počinjemo ispitivanjem mrežnog sloja i usluga koje on može da pruži. Zatim ćemo ponoviti dva šira pristupa strukturisanju isporuke paketa u mrežnom sloju - model datagrama i model virtualnih kola - koje smo već pomenuli u poglavlju 1 da bismo shvatili suštinsku ulogu adresiranja u isporuci paketa od izvora do odredišta.

U ovom poglavlju povlačimo značajnu razliku između dve funkcije mrežnog sloja: prosleđivanja i rutiranja. Prosleđivanje podrazumeva transfer paketa od ulaznog do izlaznog linka *istog* ruteru. Rutiranje podrazumeva *sve* mrežne ruteru, čijim

se zajedničkim radom pomoću protokola rutiranja određuju putanje (ili rute) kojima paketi putuju od izvornog do odredišnog čvora. Ako prilikom obrade ovog poglavlja imate uvek na umu ovu značajnu razliku, lakše ćete shvatiti mnoge teme.

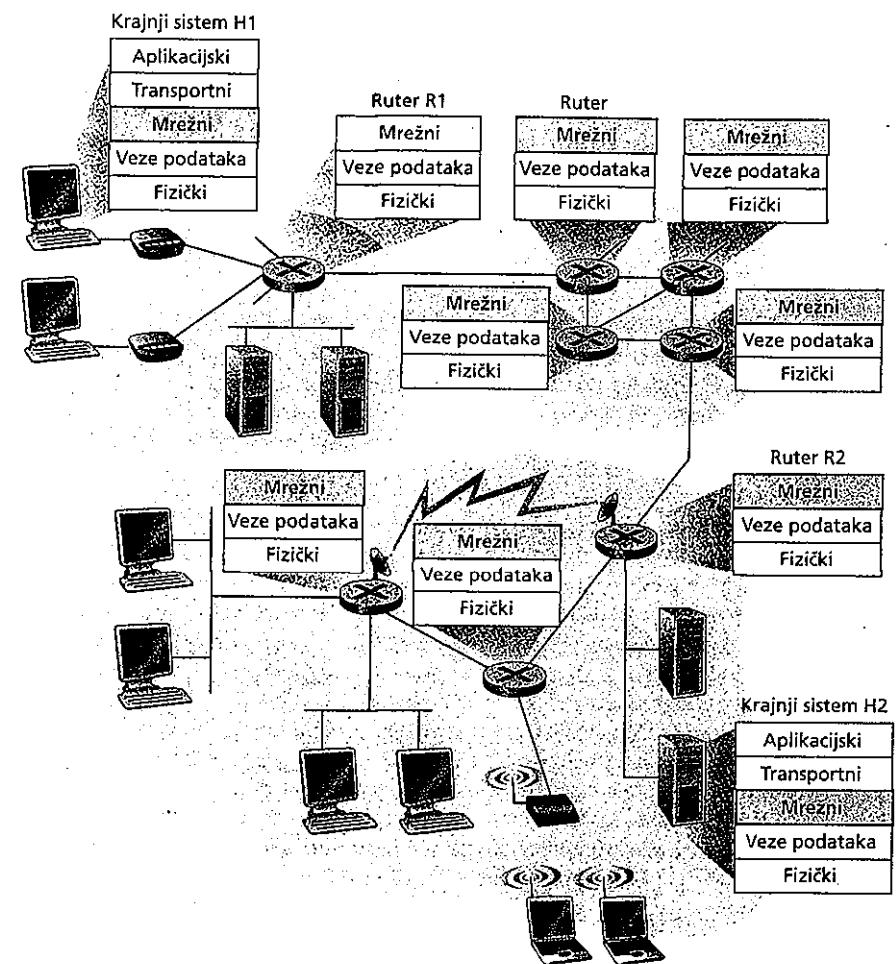
Da bismo produbili poznavanje prosledivanja paketa zavrićemo „unutar“ rutera - u njegovu hardversku arhitekturu i organizaciju. Zatim ćemo pogledati prosledivanje na Internetu uključujući i slavni IP (*Internet Protokol*). Razmotrićemo adresiranje u mrežnom sloju i format datagrama IPv4. Zatim istražujemo prevodenje mrežnih adresa (*network address translation, NAT*), fragmentaciju datagrama, inter-netski protokol za kontrolne poruke (*Internet Control Message Protocol, ICMP*) i IPv6.

Nakon toga pažnju posvećujemo funkciji rutiranja u mrežnom sloju. Videćemo da se posao rutiranja sastoji od pronalaženja dobrih putanja (odnosno, ruta) od pošiljalaca do primalaca. Prvo ćemo proučiti teoriju algoritama rutiranja, a koncentrišemo se na dve preovladajuće klase algoritama: rutiranje prema stanju linkova i rutiranje vektorima rastojanja. Pošto složenost algoritama rutiranja značajno raste sa porastom broja rutera u mreži, obradićemo i hijerarhijsko rutiranje. Zatim ćemo videti kako se ova teorija sprovodi u praksi kada predemo na protokole rutiranja među autonomnim sistemima (RIP, OSPF i IS-IS) i unutrašnji protokol rutiranja, BGP. Poglavlje zaključujemo opisom difuznog i višeznačnog rutiranja.

Sve u svemu, ovo poglavlje ima tri dela. Prvi deo, odeljci 4.1 i 4.2 sadrže opise funkcija i usluga mrežnog sloja. Drugi deo, odeljci 4.3 i 4.4 posvećeni su prosledi-vanju. Na kraju, u trećem delu, u odeljcima od 4.5 do 4.7 opisujemo rutiranje.

## 4.1 Uvod

Na slici 4.1 prikazana je jednostavna mreža sa dva računara, H1 i H2 i nekoliko rutera na putanji između računara H1 i H2. Pretpostavimo da H1 šalje informacije u H2 i razmotrimo ulogu mrežnog sloja u ovim računarima i u tranzitnim ruterima. Mrežni sloj u računaru H1 uzima segmente od transportnog sloja računara H1, enkapsulira svaki segment u datagram (tj. PDU mrežnog sloja) i zatim šalje datagrame na put prema njihovom odredištu; tj. šalje datagrame na najbliži ruter R1. U prijemnom računaru, H2, mrežni sloj prima datagrame od najbližeg ruteru (u ovom slučaju od ruteru R2). Vadi iz datograma segmente transportnog sloja i isporučuje ih naviše transportnom sloju u računaru H2. Prvenstvena uloga rutera je da „prosledjuje“ datagrame sa ulaznih linkova na izlazne. Obratite pažnju na to da su ruteri na slici 4.1 prikazani sa skraćenom familijom protokola tj. bez slojeva iznad mrežnog sloja, zato što (osim u svrhu kontrole) ruteri ne izvršavaju protokole aplikacijskog i transportnog sloja kakve smo razmatrali u poglavljima 2 i 3.



**Slika 4.1** ♦ Mrežni sloj

### 4.1.1 Prosledivanje i rutiranje

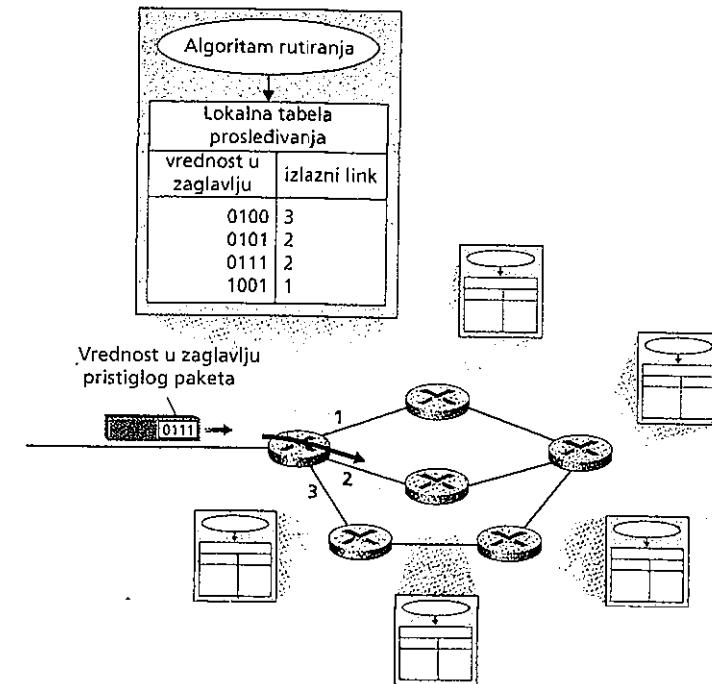
Uloga mrežnog sloja je naizgled jednostavna - da prenosi pakete od otpremnog do prijemnog računara. U tom poslu mogu se uočiti dve značajne funkcije mrežnog sloja:

- ♦ **Prosleđivanje.** Kada paket stigne na ruterov ulazni link, ruter mora da ga pre-mesti na odgovarajući izlazni link. Na primer, paket koji od računara H1 stigne u ruter R1 mora da se prosledi sledećem ruteru na putanji prema H2. U odeljku 4.3 zavirićemo u ruter i ispitati kako se paket u stvari prosleđuje od ruterovog ulaznog linka do izlaznog linka.
- ♦ **Rutiranje.** Mrežni sloj mora da utvrdi rutu ili putanju kojom paketi idu od pošiljaoca do primaoca. Algoritme koji izračunavaju te putanje nazivamo algoritmima rutiranja. Algoritam rutiranja bi, na primer, utvrdio putanju kojom paketi teku od računara H1 do H2.

Prilikom opisivanja mrežnog sloja autori često mešaju izraze *rutiranje* i *prosleđivanje*. Mi ćemo ih koristiti mnogo preciznije. *Prosleđivanje* znači lokalnu aktivnost ruteru prilikom prenosa datagrama sa interfejsa ulaznog linka u odgovarajući interfejs izlaznog linka. *Rutiranje* znači sveukupni proces određivanja putanje sa kraja na kraj kroz celu mrežu kojom će datagrami ići, od izvora do odredišta. Upo-trebimo analogiju iz automobilizma i vratimo se primeru putovanja od Pensilvanije do Floride iz odeljka 1.3.2. Tokom ovog putovanja naš vozač mora da pređe niz raskrsnica. Prosleđivanje možemo posmatrati kao prolazak kroz pojedinačnu raskrsnicu - automobil ulazi u raskrsnicu, dobija uputstvo kako da stigne do sledeće raskrsnice na putu prema konačnom odredištu i zatim se upućuje izlaznim drumom prema sledećoj raskrsnici. Rutiranjem možemo smatrati proces planiranja puta od Pensilvanije do Floride: pre nego što krene na put, vozač je proučio kartu i izabrao jednu od više mogućih putanja, gde se svaka putanja sastoji od niza segmenata koji se spajaju na raskrsnicama. U prvom delu ovog poglavlja usredsređujemo se na teme vezane za prosleđivanje; nakon toga se posvećujemo rutiranju.

Svaki ruter ima tabelu prosleđivanja. Ruter prosleđuje paket tako što ispituje vrednost jednog polja u zaglavljusu pristiglog paketa i tu vrednost koristi kao indeks za ruterovu tabelu prosleđivanja. Rezultat dobijen iz tabele prosleđivanja ukazuje na interfejs ruterovog linka na koji treba proslediti paket. Zavisno od protokola mrežnog sloja, ova vrednost iz zaglavljusa paketa može da bude adresa odredišta tog paketa ili indikacija konekcije kojoj paket pripada. Na slici 4.2 dalje jedan primer. U tom primeru, u ruter stiže paket koji u polju zaglavljusa ima vrednost 0111. Ruter koristi tu vrednost kao indeks za tabelu prosleđivanja i utvrđuje da je interfejs izlaznog linka za ovaj paket interfejs 2. Ruter tada internu prosleđuje paket interfejsu 2. U odeljku 4.3 zavirićemo u ruter i proučiti funkciju prosleđivanja daleko detaljnije.

Sada se možda pitate kako se konfigurišu tabele prosleđivanja u ruterima. Ovo je presudno pitanje iz kojeg se vidi značajno medudejstvo rutiranja i prosleđivanja. Kao što se vidi na slici 4.2, algoritam rutiranja određuje vrednosti koje se stavljaju u ruterske tabele prosleđivanja. Algoritam rutiranja može da bude centralizovan (kada se algoritam izvršava na jednoj centralnoj lokaciji a informacije za rutiranje se preuzimaju na svim ruterima) ili decentralizovan (kada se na svakom ruteru izvršava



**Slika 4.2** ♦ Algoritmi rutiranja određuju vrednosti u tabelama prosleđivanja.

deo distribuiranog algoritma rutiranja). U oba slučaja, ruter prima poruke protokola rutiranja koje se koriste za konfiguriranje tabele prosleđivanja. Zasebne i različite vrste funkcija prosleđivanja i rutiranja mogu se bolje ilustrovati još jednim hipotešičkim (nerealnim ali tehnički izvodljivim) primerom mreže u kojoj sve tabele prosleđivanja direktno konfigurišu mrežni operateri fizički prisutni kod ruteru. U tom slučaju ne bi bio potreban *nikakav* protokol rutiranja! Naravno, operateri bi morali da se dogovaraju oko konfigurisanja tabele prosleđivanja kako bi paketi stizali na svoja odredišta. Takođe je verovatno da bi takvo konfigurisanje bilo podložnije greškama nego protokol rutiranja i mnogo bi se sporije prilagođavalo promenama mrežne topologije. Prema tome, možemo da budemo zadovoljni što sve mreže imaju i funkciju prosleđivanja i funkciju rutiranja!

Dok se bavimo terminologijom, treba pomenuti još dva izraza koja se često mešaju a koja ćemo mi koristiti pažljivije. Izraz *komutator paketa* namenićemo

opštem uređaju za kom'utiranje paketa koji prenosvapaket iz interfejsa ulaznog linka u interfejs izlaznog linka, zavisno od vrednosti u polju zaglavlja paketa. Neki komutatori paketa, koje zovemo komutatori sloja veze (koji se ispituju u poglavlju 5), zasnivaju odluku na vrednosti u polju sloja veze. Drugi komutatori paketa, koje nazivamo ruterima, zasnivaju odluku o prosleđivanju na vrednosti u polju mrežnog sloja. (Da biste sasvim shvatili ovu važnu razliku, bilo bi dobro da ponovo pročitate odeljak 1.7.2 gde se opisuju datagrami mrežnog sloja i okviri sloja veze kao i njihovi odnosi.) Pošto se u ovom poglavlu bavimo mrežnim slojem, koristićemo izraz *ruter* umesno izraza *komutator paketa*. Izraz *ruter* ćemo koristiti čak i kada budemo govorili o komutatorima paketa u mrežama sa virtuelnim kolima (o kojima će uskoro biti reči).

#### Podešavanje konekcije

Upravo smo rekli da mrežni sloj ima dve važne funkcije, prosleđivanje i rutiranje. Ali uskoro ćemo videti da u nekim računarskim mrežama postoji i treća značajna funkcija mrežnog sloja, a to je podešavanje konekcije. Verovatno se sećate iz izlaganja o TCP-u da je potrebno sinhronizovanje u tri koraka pre slanja podataka od pošiljaoca ka primaocu. Tako pošiljalac i primalač podešavaju potrebne informacije

O stanju (na primer, redni broj i početnu veličinu prozora za kontrolu toka). Analogno tome, unekim arhitekturama mrežnog sloja (kao što su, ATM, Frame Relay, X.25, ali ne i Internet) obavezno je da se ruteri duž izabrane putanje od izvora do odredišta međusobno sinhronizuju i podese stanje, pre nego što paketi podataka mrežnog sloja počnu da teku. U mrežnom sloju se ovaj postupak naziva *podešavanje konekcije*. \* Podešavanje konekcije ćemo istražiti u odeljku 4.2.

#### 4.1.2 Modeli mrežne usluge

Pre nego što se detaljnije upustimo u mrežni sloj, prvo ćemo pogledati opštu sliku 1 razmotriti kakve sve vrste usluga nudi ovaj sloj. Kada transportni sloj u otpremnom računaru preda paket na mrežu (tj. prenese ga naniže mrežnom sloju u tom računaru) da li transportni sloj može da računa da će mrežni sloj isporučiti paket, na odredište? Kada se šalje više paketa, da li će oni biti isporučeni transportnom sloju prijemnog računara istim redom kako su poslati? Da li će razmak između slanja dva paketa biti jednak razmaku između njihovih primanja? Da li će mreža obezbediti ikakvu povratnu informaciju o zagruženju na mreži? Kakav je apstraktan prikaz (tj. koja su svojstva) kanala koji povezuje transportne slojeve u otpremnom i prijemnom računaru? Odgovori na ova pitanja! na mnoga druga zavise od modela usluge koju pruža mrežni sloj. Model mrežne usluge definiše karakteristike prenosa podataka sa jednog kraja periferije mreže na drugi, tj. između krajnjeg sistema koji šalje i onog koji prima.

Razmotrimo sada neke usluge koje bi mrežni sloj mogao da pruža. Na otpremnoj strani, kada transportni sloj preda paket mrežnom sloju, mrežni sloj bi mogao da nudi sledeće usluge:

- ◆ *Garantovana isporuka*. Ova usluga garantuje da će paket kad-tad stići na odredište.
- ◆ *Garantovana isporuka sa ograničenim kašnjenjem*. Ova usluga ne samo što garantuje isporuku paketa već i isporuku sa navedenim određenim kašnjenjem od računara do računara (npr. kroz najviše 100 ms).

Osim toga, u vezi sa *tokom paketa* između datog izvora i odredišta moglo bi se nuditi i sledeće usluge:

- ◆ *Isporuka u ispravnom redosledu*. Ova usluga garantuje da će paketi stizati na odredište onim redom kako se i šalju.
- ◆ *Garantovani propusni opseg*. Ova usluga mrežnog sloja emulira ponašanje pre-nosnog linka određene brzine (npr. 1 Mb/s) od otpremnog do prijemnog računara (iako stvarna putanja sa kraja na kraj može da obuhvata više fizičkih linkova). Sve dok otpremni računar predaje bitove (kao delove paketa) brzinom manjom od specifikovane, paketi se neće gubiti i svaki paket stiže od računara do računara u unapred određenom intervalu (npr. 40 ms).
- ◆ *Garantovana maksimalna promenljivost kašnjenja*. Ova usluga garantuje da će vreme između predaje dva uzastopna paketa kod pošiljaoca biti jednak vremenu između njihovog uzastopnog pristicanja kod primaoca (ili da taj razmak neće varirati više od neke navedene vrednosti).

Ovo je samo delimičan spisak usluga koje bi mrežni sloj mogao da nudi - tu su moguće bezbrojne varijacije.

Današnja arhitektura Interneta pruža samo jedan model usluge, takozvanu uslugu najboljeg pokušaja. Iz tabele 4.1 mogao bi se steći utisak da je *usluga najboljeg pokušaja* eufemizam za *bez ikakve usluge*. U usluzi najboljeg pokušaja ne garantuje se očuvanje vremenskog razmaka među paketima, ne garantuje se da će paketi stići redosledom kojim su poslati, niti se uopšte garantuje isporuka posla-tih paketa. Ako posmatramo samu definiciju, mreža koja ne bi isporučivala *nijedan* paket na odredište zadovoljila bi definiciju usluge najboljeg pokušaja. (Zaista, današnji zagušeni javni Internet ponekad izgleda kao primer baš takve mreže!) Međutim, kao što ćemo uskoro objasnitи, za tako minimalistički model usluge na mreži postoje valjani razlozi. Internetov model usluge najboljeg pokušaja trenutno se proširuje, pa će uključiti i takozvane integrisane usluge i diferencirane usluge. Te modele usluga koji se još razvijaju obradićemo kasnije u poglavljiju 7.

Druge mrežne arhitekture definisale su i implementirale modele usluge koji prevazilaze internetsku uslugu najboljeg pokušaja. Na primer, mrežna arhitektura ATM [ATM Forum 2004, Black 1995] predviđa više modela usluga, tj. da različite konekcije dobijaju na istoj mreži različite klase usluga. Objašnjenje načina na koji ATM mreža obezbeđuje takve usluge prevazilazi okvire ove knjige; ovde nam je cilj samo da napomenemo da za internetsku uslugu najboljeg pokušaja postoje altema-

| Mrežna arhitektura usluge | Model usluge     | Garantovani propusni opseg    | Garantovano bez gubitaka | Redosled    | Vremenski raspored | Upozorenje o zagušenju |
|---------------------------|------------------|-------------------------------|--------------------------|-------------|--------------------|------------------------|
| Internet                  | Najbolji pokušaj | Nema                          | Ne                       | Bilo koji   | Ne odrižava se     | Nema                   |
| ATM                       | CBR              | Garantovana konstantno brzina | Da                       | U redosledu | Održava se         | Nemo zagušenja         |
| ATM                       | ABR              | Garantovani minimum           | Ne                       | U redosledu | Ne odrižava se     | Postoji upozorenje     |

**Tabela 4.1** ♦ Modeli usluga Internet, ATM CBR i ATM ABR.

tive. Dva najvažnija modela ATM usluge su usluga konstantne bitske brzine i usluga dostupne bitske brzine:

♦ **ATM mrežna usluga konstantne bitske brzine (constant bit rate, CBR).** Ovo je bio prvi standardizovan model ATM usluge u kojem se ogleda tadašnja zain-teresovanost telefonskih kompanija za ATM i pogodnost CBR usluge za prenos audio i video saobraćaja u realnom vremenu konstantnom bitskom brzinom. Cilj CBR usluge je po zamisli jednostavan - postići da mrežna konekcija izgleda kao iznajmljena linija sa paricama ili optičkim vlaknima između pošiljaoca i primaoca. Sa CBR uslugom, ATM paketi (koji se u ATM žargonu nazivaju ćelijama) prenose se preko mreže tako da se garantuju maksimalne vrednosti kašnjenja ćelije sa kraja na kraj, promenljivosti kašnjenja ćelije sa kraja na kraj (koje se često naziva „treperenjem“) i procenat ćelija koje se izgube ili se prekasno isporuče. Otpremni računar i ATM mreža dogovaraju se o tim vrednostima prilikom uspostavljanja CBR konekcije.

4 **ATM mrežna usluga dostupne bitske brzine (available bit rate, ABR).** Kako Internet nudi takozvanu uslugu najboljeg pokušaja, najbolji opis za ABR bio bi „usluga neznatno bolja od najboljeg pokušaja“. Kao i u internetskom modelu usluge, u ABR usluzi ćelije mogu da se izgube. Međutim, za razliku od Interneta, ćelijama ne može da se promeni redosled (mada mogu da se izgube), a konekciji koja koristi ABR uslugu garantuje se minimalna brzina prenosa ćelije (*minimum cell transmission rate, MCR*). Ako u određenom trenutku mreža ima dovoljno slobodnih resursa, pošiljalac će moći uspešno da šalje saobraćaj brzinom *većom* od MCR-a. Osim toga, kao što smo videli u odeljku 3.6, ATM usluga ABR može pošiljaocu da obezbedi povratne informacije (u vidu bita upozorenja o zagušenju ili obaveštenja o manjoj brzini kojom treba slati) koje regulišu način na koji pošiljalac podešava brzinu između MCR-a i dozvoljene maksimalne brzine ćelija.

## 4.2 Mreže sa virtualnim kolima i sa datagramima

Sigurno se iz trećeg poglavlja sećate da transportni sloj može aplikacijama da ponudi uslugu bez konekcije ili uslugu sa konekcijom. Na primer, transportni sloj svakoj aplikaciji nudi izbor između dve usluge: UDP, usluge bez konekcije i TCP, usluge sa konekcijom. Slično tome, mrežni sloj može da ponudi uslugu bez konekcije i uslugu sa konekcijom. Usluge sa konekcijom i bez nje u mrežnom sloju imaju mnoge sličnosti sa uslugama sa konekcijom i bez nje u transportnom sloju. Na primer, usluga mrežnog sloja sa konekcijom počinje od sinhronizovanja između izvornog i odredišnog računara, a usluga mrežnog sloja bez konekcije nema nikakvog početnog usaglašavanja.

Iako usluge sa konekcijom i bez nje u mrežnom sloju imaju neke sličnosti sa uslugama sa konekcijom i bez nje u transportnom sloju, postoje i bitne razlike:

- ♦ U mrežnom sloju, to su usluge koje mrežni sloj pruža transportnom sloju od računara do računara. U transportnom sloju, reč je o uslugama koje transportni sloj pruža aplikacijskom sloju od procesa do procesa.
- ♦ U svim glavnim arhitekturama računarskih mreža do danas (Internet, ATM, Frame Relay itd.), mrežni sloj pruža uslugu sa konekcijom od računara do računara ili uslugu bez konekcije od računara do računara, ali ne i obe. Računarske mreže koje u mrežnom sloju daju samo uslugu sa konekcijom nazivamo **mreže sa virtuelnim kolima** (VC); računarske mreže koje u mrežnom sloju daju samo uslugu bez konekcije nazivamo **mreže sa datagramima**.
- ♦ Implementacije usluge sa konekcijom u transportnom sloju i usluge sa konekcijom u mrežnom sloju se bitno razlikuju. U prethodnom poglavlju smo videli da se usluga sa konekcijom u transportnom sloju implementira na periferiji mreže u krajnjim sistemima; uskoro ćemo videti da se usluga sa konekcijom u mrežnom sloju implementira u jezgru mreže kao i u krajnjim sistemima.

Mreže sa virtuelnim kolima i sa datagramima su dve osnovne klase računarskih mreža. One zasnivaju svoje odluke o posleđivanju na veoma različitim informacijama. Pogledajmo sada njihove implementacije.

### 4.2.1 Mreže sa virtuelnim kolima

Naučili smo da je Internet mreža sa datagramima. Međutim, mnoge alternativne mreže (ATM, Frame Relay i X.25) su mreže sa virtuelnim kolima i, prema tome, koriste konekcije u mrežnom sloju. Ove konekcije u mrežnom sloju nazivaju se **virtuelnim kolima** (*virtual circuit, VC*). Razmotrimo sada kako se u računarskoj mreži može implementirati usluga VC.

Virtuelno kolo (VC) sastoji se od (1) putanje (tj. niza linkova i ruteru) između izvornog i odredišnog računara, (2) VC brojeva, po jedan broj za svaki link na putanju i (3) stavki u tabeli prosledivanja svakog ruteru na putanji. Paket koji pripada jednom virtualnom kolu imaće u zaglavljtu VC broj. Pošto virtuelno kolo može u svakom linku da ima drugačiji VC broj, svaki ruter na putu mora u paketima koji kroz njega prolaze da zameni stari VC broj novim. Novi VC broj dobija se iz tabele prosledivanja.

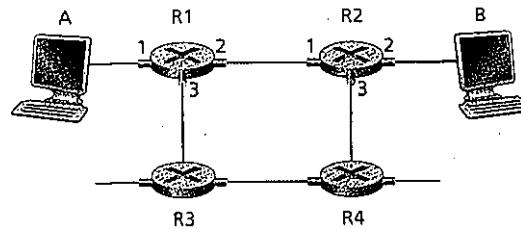
Za ilustrovanje ovog koncepta uzećemo mrežu prikazanu na slici 4.3. Brojevi pored linkova ruteru R1 na slici 4.3 su brojevi interfejsa linkova. Prepostavimo sada da računar A zatraži da mreža uspostavi VC između njega i računara B. Prepostavimo takođe da mreža izabere putanju A-RI-R2-B i dodeli brojeve 12, 22 i 32 linkovima u putanji ovog virtuelnog kola. U ovom slučaju, kada paket ovog virtuelnog kola napušta računar A, u njegovom zaglavljtu se u polju VC broja nalazi vrednost 12; kada napušta ruter R1, vrednost je 22; a kada napušta R2, vrednost je 32.

Kako ruter utvrđuje kojim brojem treba zameniti VC broj u paketu koji prolazi kroz ruter? U VC mreži tabela prosledivanja svakog ruteru sadrži prevodenje VC brojeva: na primer, tabela prosledivanja u R1 mogla bi da izgleda ovako:

| Ulazni interfejs | Dolazni VC broj | Izlazni interfejs | Odlazni VC broj |
|------------------|-----------------|-------------------|-----------------|
| 1                | 12              | 2                 | 22              |
| 2                | 63              | 1                 | 18              |
| 3                | 7               | 2                 | 17              |
| 1                | 97              | 3                 | 87              |
| ...              | ...             | ...               | ...             |

Kad god se kroz ruter uspostavi novo virtuelno kolo, dodaje se nova stavka u tabelu prosledivanja. Slično tome, kad god se neko virtuelno kolo završi, uklanjuju se odgovarajuće stavke iz svih tabela duž putanje.

Možda se pitate zašto paket prosto ne zadrži isti VC broj na svim linkovima duž putanje. Postoje dva razloga. Prvo, zamenom broja za svaki link smanjuje se potreba



Slika 4.3 ♦ Jednostavna mreža sa virtuelnim kolima

ban broj cifara u polju VC broja u zaglavljtu paketa. Drugo, još važnije, podešavanje VC brojeva je značajno jednostavnije kada se dozvole različiti VC brojevi na linko-vima duž putanje virtuelnog kola. Konkretno, kada se koristi više VC brojeva svaki link na putanji može da bira VC broj bez obzira na VC brojeve koje su izabrali drugi linkovi te putanje. Kada bi se zahtevao VC broj zajednički za sve linkove putanje, ruteri bi morali da razmene i obrade osetnu količinu poruka da bi se usaglasili oko zajedničkog VC broja za novu konekciju (tj. broja koji u tim ruterima trenutno ne koristi nijedno drugo virtualno kolo).

U VC mreži, mrežni ruteri moraju da održavaju **informacije** o stanju **konekcije** za aktivne konekcije. Konkretno, kad god se uspostavi nova konekcija, u tabelu pro-sledivanja svakog ruteru mora da se doda nova stavka konekcije; a svaki put kada se neka konekcija prekine, iz tabele treba izbaciti njenu stavku. Obratite pažnju na to da i kada nema prevodenja VC brojeva, ipak treba održavati informacije o stanju konekcije u kojima se VC broju pridružuje broj izlaznog interfejsa. Pitanje da li ruteri održavaju informacije o stanju konekcija ili ne je od presudnog značaja, i mi ćemo mu se vraćati više puta u ovoj knjizi.

U virtuelnom kolu prepoznaju se tri faze:

- ♦ **Podešavanje virtuelnog kola.** U fazi podešavanja, transportni sloj pošiljaoca kontaktira mrežni sloj, navodi adresu primaoca i čeka da mreža uspostavi virtuelno kolo. Mrežni sloj utvrđuje putanju između pošiljaoca i primaoca, tj. niz linkova i ruteru kroz koji će prolaziti svi paketi virtuelnog kola. Na kraju, mrežni sloj dodaje po jednu stavku u tabelu prosledivanja svakog ruteru na putanji. Tokom podešavanja virtuelnog kola mrežni sloj može takođe da rezerviše resurse (na primer, propusni opseg) za virtuelno kolo na putanji.
- ♦ **Transfer podataka.** Kao što se vidi na slici 4.4, pošto se virtuelno kolo uspostavi, kroz njega mogu da teku podaci.
- ♦ **Raskidanje virtuelnog kola.** Ovo počinje kada pošiljalac (ili primalac) obavesti mrežni sloj o svojoj želji da prekine virtuelno kolo. Mrežni sloj tada obično obaveštava krajnji sistem na drugoj strani mreže o prekidanju poziva i ažurira tabele prosledivanja u svim ruterima paketa na putanji da bi se znalo da virtuelno kolo više ne postoji.

Postoji jedna suptilna, ali važna razlika između uspostavljanja virtuelnog kola u mrežnom sloju i uspostavljanja konekcije u transportnom (na primer, TCP-ovo sinhronizovanje u tri koraka koje smo opisali u poglavljju 3). Uspostavljanje konekcije u transportnom sloju tiče se samo dva krajnja sistema. Dva krajnja sistema pristaju da komuniciraju i zajednički utvrđuju parametre (na primer, početni redni broj i veličinu prozora za kontrolu toka) njihove konekcije u transportnom sloju pre nego što podaci počnu da teku po njoj. Krajnji sistemi su svesni konekcije transportnog sloja, ali je ruter u mreži uopšte ne primećuju. S druge strane, u mrežnom sloju sa virtuelnim kolom, ruteri na putanji između dva krajnja sistema uključeni su u uspostavljanje virtuelnog kola i svaki ruter je svestan svih virtuelnih kola koja prolaze kroz njega.

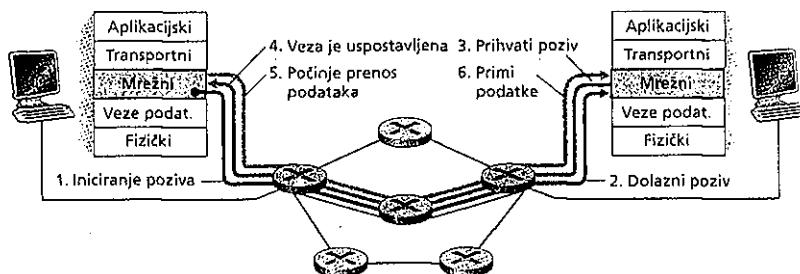
Poruke koje krajnji sistemi šalju na mrežu da bi ukazali na iniciranje ili raskidanje virtuelnog kola i poruke koje razmenjuju ruteri radi uspostavljanja virtuelnog kola (tj. za menjanje ruterskih tabela sa stanjima konekcija) poznate su kao poruke signalizacije a protokoli koji se koriste za razmenu tih poruka često se nazivaju protokolima signalizacije. Uspostavljanje virtuelnog kola prikazano je na slici 4.4. U ovoj knjizi ne obrađujemo protokole signalizacije; opšti opis signalizacije u mrežama sa konekcijama naći će se u knjizi [Black 1997] a specifikaciju ATM-ovog protokola signalizacije Q.2931 u knjizi [ITU-T Q.2931 1994].

#### 4.2.2 Mreže sa datagramima

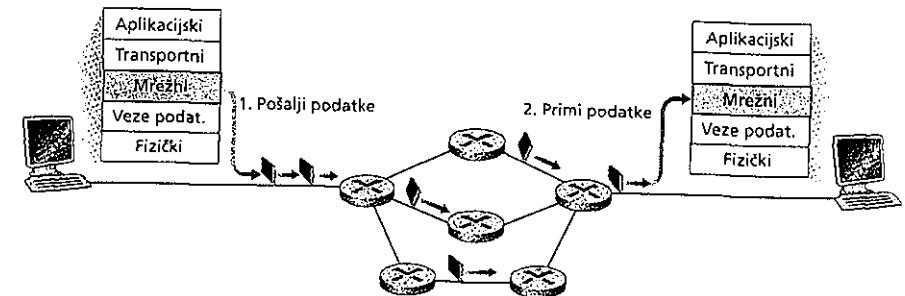
U mrežama sa datagramima, kad god krajnji sistem hoće da pošalje paket, on stavi u njega adresu krajnjeg odredišnog sistema i zatim ubaci paket u mrežu. Kao što se vidi na slici 4.5, to se postiže bez ikakvog uspostavljanja virtuelnog kola. Ruteri u mreži sa datagramima ne održavaju nikakve informacije o stanju virtuelnih kola (pošto i nema virtuelnih kola!).

Tokom prenosa od izvora do odredišta, paket prolazi kroz niz ruta. Svaki od ovih ruta koristi adresu odredišta u paketu za njegovo prosleđivanje. Konkretno, svaki ruter ima tabelu prosleđivanja u kojoj se adrese odredišta preslikavaju u interfejs linkova; kada paket stigne u ruter, ruter koristi adresu odredišta paketa da bi u tabeli prosleđivanja pronašao odgovarajući interfejs izlaznog linka. Ruter zatim prosleduje paket na taj interfejs izlaznog linka.

Da biste bolje shvatili postupak pretraživanja tabele, pogledaćemo jedan konkretni primer. Pretpostavite da sve adrese odredišta imaju 32 bita (što je slučajno upravo dužina odredišne adrese u IP datagramu). U gruboj implementaciji, tabela prosleđivanja bi imala po jednu stavku za svaku moguću adresu odredišta. Pošto



Slika 4.4 ♦ Podešavanje virtuelnog kola



Slika 4.5 ♦ Mreža sa datagramima

postoji više od 4 milijarde mogućih adresa, ta opcija nikako ne dolazi u obzir -tabela prosleđivanja bi bila stravična.

Prepostavimo zatim da naš ruter ima Četiri linka, mimerisana od 0 do 3 i da pakete treba proslediti na interfejs linkova na sledeći način:

| Raspon odredišnih adresa                              | Interfejs linka |
|-------------------------------------------------------|-----------------|
| 11001000 00010111 00010000 00000000<br>do<br><b>i</b> | o               |
| 11001000 00010111 00011000 00000000<br>do<br><b>i</b> | <i>i</i>        |
| 11001000 00010111 00011000 11111111<br>do<br><b>i</b> | 2               |
| 11001000 00010111 00011111 minu<br>do<br><b>i</b>     | 3<br>inače      |

Sasvim je jasno da u ovom slučaju nije potrebno imati 4 milijarde adresa prosleđivanja u ruterovoj tabeli prosleđivanja. Mogli bismo, na primer, da imamo tabelu prosleđivanja sa samo četiri stavke:

| Prefiks                  | Interfejs linka |
|--------------------------|-----------------|
| iioooioooooioin 00010    | 0               |
| iioooioooooioin 00011000 | i               |
| iioooioooooioin ooon     | 2               |
| inače                    | 3               |

Sa ovakvom vrstom tabele prosleđivanja, ruter među stawkama u tabeli traži prefiks određene adrese paketa; ako postoji jednakost, ruter prosleđuje paket na link pridružen toj vrednosti. Na primer, pretpostavimo da je određena adresa paketa 11001000 00010111 00010110 10100001; pošto je prefiks od 21 bita ove adrese jednak prvoj stavci u tabeli, ruter prosleđuje paket na interfejs linka 0. Ako prefiks nije jednak ni jednoj od prve tri stavke, ruter će proslediti paket na interfejs 3. Mada ovo izgleda prilično jednostavno, tu postoji jedna značajna suptilnost. Možda ste primetili daje moguće da određenoj adresi odgovara više stavki. Na primer, prvih 24 bita adrese 11001000 00010111 00011000 10101010 jednak je drugoj stavci tabele, a prvih 21 bitova je jednak trećoj stavci. U slučaju više jednakosti ruter primenjuje pravilo jednakosti najdužeg prefiksa; tj. pronalazi najdužu jednaku stavku u tabeli i prosleđuje paket na interfejs linka pridružen najdužem odgovarajućem prefiksu.

Naravno, da bi pravilo jednakosti najdužeg prefiksa moglo da se koristi, svaki interfejs izlaznog linka treba da bude zadužen za prosleđivanje velikog bloka susednih određenih adresa. U odeljku 4.4 videćemo da se internetske adrese obično dodjeljuju hijerarhijskim principom tako da u tabelama prosleđivanja većine rutera prevladjuje to svojstvo susednosti. I pored toga, u zajednici istraživača Interneta postoji briga da se u adresnom prostoru pojavljuje sve više i više prekida tako da neprekidni blokovi susednih adresa postaju sve manji i manji a tabele prosleđivanja sve veće. (Pročitajte [Maennel 2002], [RFC 3221] i opis Principi u praksi iz odeljka 4.4.)

Mada ruteri u mrežama sa datagramima ne održavaju nikakve informacije o stanju konekcija, oni ipak u svojim tabelama održavaju informacije o stanju prosleđivanja. Međutim, ove informacije o stanju prosleđivanja menjaju se relativno sporo. Zaista, algoritmi rutiranja ažuriraju tabele prosleđivanja u mreži sa datagramima, u intervalima od približno jednog do pet minuta. U VC mreži, tabela prosleđivanja u ruteru se menja kad god se uspostavi nova konekcija kroz taj ruter i kad god se raskine neka postojeća konekcija kroz njega. To se u ruteru okosnice nivoa 1 lako može događati u intervalima koji se mere mikrosekundama.

Pošto se tabele prosleđivanja mogu menjati u bilo kad, niz paketa koji se šalju od jednog krajnjeg sistema u drugi mogu da produ različitim putanjama kroz mrežu i mogu da stignu izvan redosleda. [Paxson 1997] i [Jaiswal 2003] predstavljaju jednu zanimljivu studiju merenja izmena u redosledima paketa i drugih fenomena u javnom Internetu.

### 4.1.3 Poreklo usluga datagrama i virtualnog kola

Razvoj modela usluga datagrama i virtuelnog kola ukazuje na njihovo poreklo. Pojam virtuelnog kola kao glavnog organizacionog principa potiče iz sveta telefonijske komunikacije u kojoj se koriste „realna“ kola. Kako se u mreži sa virtuelnim kolima uspostavljaju pozivi i stanje svakog poziva održava u ruterima unutar mreže, ona je znatno složenija od mreže sa datagramima (mada u knjizi [Molinero 2002] možete naći zanimljivo poređenje složenosti mreža sa komutacijom vodova i mreža sa komutacijom paketa). I to je naslede iz telefonije. U telefonskim mrežama, sama mreža je morala da bude složena pošto je kao krajnje sisteme povezivale glupe uređaje, kao što su telefoni sa rotacionim brojčanikom (za mlade Čitaocu koji možda ne znaju, to je analogni telefon koji nema dugmadi - ima samo kružni brojčanik).

S druge strane, Internetov model usluge sa datagramom, potekao je iz potrebe da se povezuju računari. Pošto su kao krajnje sisteme imali složenije uređaje, arhitekti Interneta su odlučili da u mrežnom sloju model usluge bude što jednostavniji. Kao što smo već videli u poglavljima 2 i 3, dodatna funkcionalnost (na primer, isporuka u redosledu, pouzdani transfer podataka, kontrola zagubljenja i DNS razrešavanja, itd.) se zatim implementira u višem sloju, u krajnjim sistemima. Ovaj model suprotnost telefonskoj mreži, pa imamo i neke zanimljive konsekvene:

- ◆ Model mrežne usluge na Internetu koji daje minimalne (nikakve!) garancije usluge (i zato mrežnom sloju postavlja minimalne zahteve), takođe olakšava *međusobno povezivanje* mreža koje koriste veoma različite tehnologije u sloju veze (na primer, satelitske mreže, Ethernet, optička vlakna ili radio veze) i koje imaju veoma različite brzine prenosa i karakteristike gubitaka. Međusobno povezivanje IP mreža detaljno ćemo obraditi u odeljku 4.4.
- ◆ Kao što smo videli u poglavljju 2, aplikacije kao što su e-pošta, VWebs, pa čak i usluga aplikacijskog sloja koju koristi mrežni sloj kao što je DNS, implementiraju se u računarima (serverima) na periferiji mreže. Mogućnost da se nova usluga doda prostim priključivanjem računara na mrežu i definisanjem novog protokola aplikacijskog sloja (kao stoje HTTP) omogućila je da se veoma brzo prihvate nove usluge na Internetu, na primer VWebs.

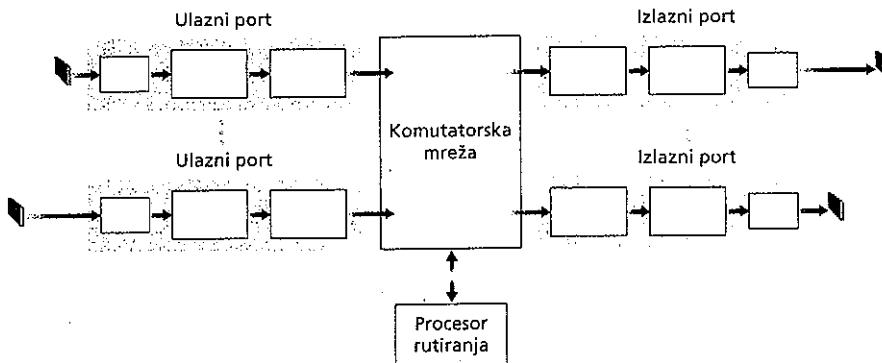
Međutim, kao što ćemo videti u poglavljju 7, u internetskoj zajednici se vode žučne rasprave o pravcu razvoja arhitekture Internetovog mrežnog sloja koja će moći da podrži usluge u realnom vremenu kao što je multimedija. U [Crowcroft 1995] daje zanimljivo poređenje ATM mrežne arhitekture sa virtuelnim kolima i predlaže se arhitektura za sledeću generaciju Interneta.

## 4.3 Šta ima u ruteru?

Pošto smo videli opšti pregled funkcija i usluga mrežnog sloja obratimo pažnju na funkciju prosleđivanja mrežnog sloja - samo prenošenje paketa iz ulaznog linka ruteru na odgovarajući izlazni link. U odeljku 4.2 već smo ukratko pomenuli nekoliko pitanja u vezi sa prosleđivanjem, tj. adresiranje i jednakost najdužeg prefiksa. U ovom odeljku razmotrićemo konkretnе arhitekture ruteru za prenošenje paketa iz ulaznih linkova na izlazne linkove. Prikaz smo morali da skratimo pošto je temeljna obrada dizajniranja ruteru oblast za sebe. Zato ćemo se ovde posebno potruditi da ukažemo na literaturu koja detaljnije pokriva ovu temu. Usput, ovde napominjemo da istraživači i praktičari umrežavanja računara Često naizmenično koriste izraze *prosleđivanje* i *komutiranje*. U ovoj knjizi koristićemo oba izraza.

Na slici 4.6. dat je opšti pregled ruterske arhitekture. Mogu se uočiti četiri komponente ruteru.

♦ *Ulazni port*. Ulazni port izvršava nekoliko funkcija. On izvršava funkcije fizičkog sloja (sasvim levi okvir ulaznog porta i sasvim desni okvir izlaznog porta na slici 4.6) završavajući fizički link koji ulazi u ruter. Dalje, on vrši funkcije sloja veze podataka (pričekane srednjim okvirom ulaznog i izlaznog porta), potrebne za saradnju sa funkcijama sloja veze podataka na drugoj strani ulaznog linka. On takođe izvršava funkciju pretraživanja tabele i prosleđivanja (sasvim desni okvir ulaznog porta i sasvim levi okvir izlaznog porta) tako da paket prosledjen kroz komutatorsku mrežu ruteru izade na odgovarajućem izlaznom portu. Kontrolni paketi (na primer, paketi koji prenose informacije protokola rutiranja) prosleđuju



Slika 4.6 ♦ Arhitektura ruteru

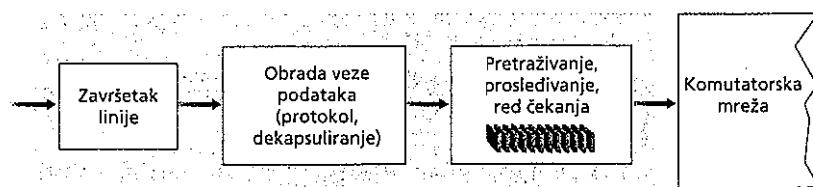
se od ulaznog porta prema procesoru rutiranja. U praksi se često više portova grupiše na jednu linijsku karticu u ruteru.

- ♦ *Komutatorska mreža*. Komutatorska mreža povezuje ulazne portove ruteru sa njegovim izlaznim portovima. Ta komutatorska mreža se celo nalazi u ruteru - mreža u mrežnom ruteru!
- ♦ *Izlazni portovi*. Izlazni port čuva pakete koji su mu prosledeni kroz komutatorsku mrežu, a zatim ih predaje na izlazni link. Izlazni port, znači, izvršava funkcije sloja veze podataka i fizičkog sloja inverzne u odnosu na ulazni port. Ako je link dvosmeran (tj. prenosi saobraćaj u oba smera) izlazni port tog linka obično će se nalaziti na istoj linijskoj kartici na kojoj je i ulazni port za taj link.
- ♦ *Procesor rutiranja*. Procesor rutiranja izvršava protokole rutiranja (na primer, protokole koje ćemo razmotriti u odeljku 4.6), održava informacije o rutiranju i tabele prosleđivanja i obavlja funkcije upravljanja mrežom u ruteru (procitati poglavlje 9).

U sledećirri odeljcima detaljnije ćemo razmotriti ulazne portove, komutatorsku mrežu i izlazne portove. Dokumenti [Chao 2002; Turner 1988; Giacopelli 1990; McKeown 1997a; Partridge 1998] sadrže opis nekih specifičnih arhitektura rutiranja. Dokument [McKeown 1997b] sadrži lepo napisan pregled modernih arhitektura ruteru u kojem se kao primer koristi ruter Cisco 12000. Da bi opis koji sledi bio konkretniji, uzećemo daje reč o paketskoj mreži i da se odluke o prosleđivanju zasnivaju na određenoj adresi paketa (a ne na VC broju u mreži sa virtuelnim kolima). Međutim, koncepti i tehnike su slične i u mreži sa virtuelnim kolima.

### 4.3.1 Ulazni portovi

Na slici 4.7 dat je detaljniji prikaz funkcija ulaznog porta. Kao stoje već rečeno, fizički sloj i sloj veze podataka implementirani su u ruteru električnim završetkom linije i obradom okvira ulaznog linka. Funkcija pretraživanja tabele i prosleđivanja u ulaznom portu ključna je za funkciju prosleđivanja u ruteru. U mnogim ruterima, ovde se utvrđuje izlazni port na koji će se pristigli paket proslediti kroz komutatorsku mrežu. Izbor izlaznog porta vrši se pomoću informacija iz tabele prosleđivanja. Mada tabelu prosleđivanja izračunava procesor rutiranja, njene kopije se obično čuvaju u svakom ulaznom portu i procesor rutiranja ih, po potrebi, ažurira. Pošto se lokalne kopije tabele prosleđivanja nalaze u svakom ulaznom portu, oni mogu da donose odluke o prosleđivanju bez pomoći centralizovanog procesora rutiranja. Takvim decentralizovanim prosleđivanjem izbegava se da nastane usko grlo na jednom mestu u ruteru.



**Slika 4.7** ♦ Obrada u ulaznom portu

Ako je ruter sa ograničenim mogućnostima obrade u ulaznom portu, on će jednostavno prosjetiti paket centralnom procesoru rutiranja koji će zatim pretražiti tabelu prosleđivanja i proslediti paket odgovarajućem izlaznom portu. Taj pristup se koristi kada radna stanica ili server služi kao ruter; tu je procesor rutiranja u stvari CPU radne stanice, a ulazni port je prosti kartica mrežnog interfejsa (na primer, Ethernet kartica).

### CISCO SISTEMI: DOMINACIJA JEZGROM MREŽE

Dok ovo pišemo (mart 2004) Cisco zapošljava više od 30.000 ljudi i ima dohodak od 150 milijardi dolara. Cisco trenutno dominira tržistem rutera za Internet, a tokom poslednjih godina prešao je na tržiste Internet telefonije gde se takmiči sa kompanijama za telefonsku opremu kao što su Lucent, Alcatel, Nortel i Siemens. Kako je nastala ova ogromna kompanija koja se bavi umrežavanjem? Sve je počelo 1984. godine [pre samo 20 godina] u dnevnoj sobi jednog stana U Silikonskoj dolini. Len Bosak i njegova žena Sandi Lerner radili su na univerzitetu Stanford kada im je palo na pamet da prave internetske rutere i prodaju ih istraživačkim i akademskim ustanovama. Sandi Lerner je predložila ime „Cisco“ (skraćenica za San Francisco), i takođe je projektovala zaštitni znak kompanije u vidu mosta. Uprava korporacije je prvo bila u njihovoj dnevnoj sobi, a projekat su na početku finansirali kreditnim karticama i honorarnim konsultantskim poslovima. Krajem 1986. godine Ciscovi prihodi dostigli su 250.000 dolara mesečno - što nije loše za preduzeće finansirano kreditnim karticama i bez ikakvog uloženog kapitala. Krajem 1987. godine Cisco je konačno uspeo da privuče ulaganje kapitala - 2 miliona dolara od Sequoia Capital za jednu trećinu vlasništva kompanije. U sledećih nekoliko godina Cisco je nastavio da raste i osvajao je sve veći deo Iržišta. Istovremeno, odnosi između bračnog para Bosak/Lerner i uprave Cisca su se kvarili. Cisco je počeo da se kotira na berzi 1990. godine, ali te iste godine su Lerner i Bosak napustili kompaniju.

Ako imamo tabelu prosleđivanja, njeno pretraživanje je u suštini jednostavno - samo se po tabeli pretraživanja traži stavka koja ima najduži prefiks mreže jednak odredišnoj adresi paketa, kao što je objašnjeno u odeljku 4.2.2. U praksi, međutim, život nije tako jednostavan. Verovatno je najznačajnija komplikacija to što ruteri okosnice moraju da rade velikom brzinom i da budu u stanju da izvrše na milione pretraživanja u sekundi. Zaista, poželjno je da obrada u ulaznom portu može da se obavlja brzinom linije, tj. da se pretraživanje izvrši u kraćem vremenu nego što je potrebno da se paket primi u ulazni port. U takvom slučaju obrada primljenog paketa može da se dovrši pre nego što se završi sledeća operacija prijema. Da biste stekli predstavu o zahtevima performansi za pretraživanje, uzmite u obzir da takozvani link OC48 funkcioniše brzinom od 2,5 Gb/s. Sa paketima dužine 256 bajtova to znači da brzina pretraživanja mora biti približno milion pretraživanja u sekundi.

Ako se uzmu u obzir današnje potrebe za linkovima velike brzine, linearno pretraživanje velike tabele prosleđivanja je nemoguće. Pametnije je stavke tabele prosleđivanja čuvati u strukturi stabla podataka. Za svaki nivo stabla može se smatrati da odgovara jednom bitu u odredišnoj adresi. Da bi se pronašla adresa, jednostavno se počinje od osnovnog čvora u stablu. Ako je prvi bit u adresi 0, stavku tabele prosleđivanja za odredišnu adresu sadrži levo podstablo; inače će adresa biti u desnem podstablu. Kroz odgovarajuće podstablo se zatim prelazi pomoću ostalih bitova u adresi - ako je sledeći bit u adresi 0, bira se levo podstablo prvog podstabla; inače se bira desno. Na taj način stavke tabele prosleđivanja pronalaze se u  $N$  koraka, gde je //broj bitova u adresi. (Čitalac će primetiti daje to u suštini binarno pretraživanje adresnog prostora veličine  $2^N$ .) Jedno unapređenje tehnika za binarno pretraživanje opisano je u [Srinivasan 1999], a opšti pregled algoritama za klasifikaciju paketa može se naći u [Gupta 2001].

AH, čak i za  $N = 32$  koraka (na primer, za 32-bitnu IP adresu), brzina binarnog pretraživanja nije dovoljna za zahteve rutiranja u današnjim okosnicama. Na primer, ako pretpostavimo daje potreban po jedan pristup memoriji u svakom koraku, sa vremenom pristupa memoriji od 40 ns, moglo bi se izvršiti manje od milion traženja adresa u sekundi. Zato je bilo potrebno da se istraži nekoliko tehniku da bi se dobila veća brzina pretraživanja. CAM (*Content Addressable Memories*) memorije omogućavaju da im se predstavi 32-bitna IP adresa a one zatim suštinski konstantnom brzinom vraćaju sadržaj tabele prosleđivanja za tu adresu. Ruter iz serije Cisco 8500 [Cisco 8500 1999] ima na svakom ulaznom portu CAM od 64 K.

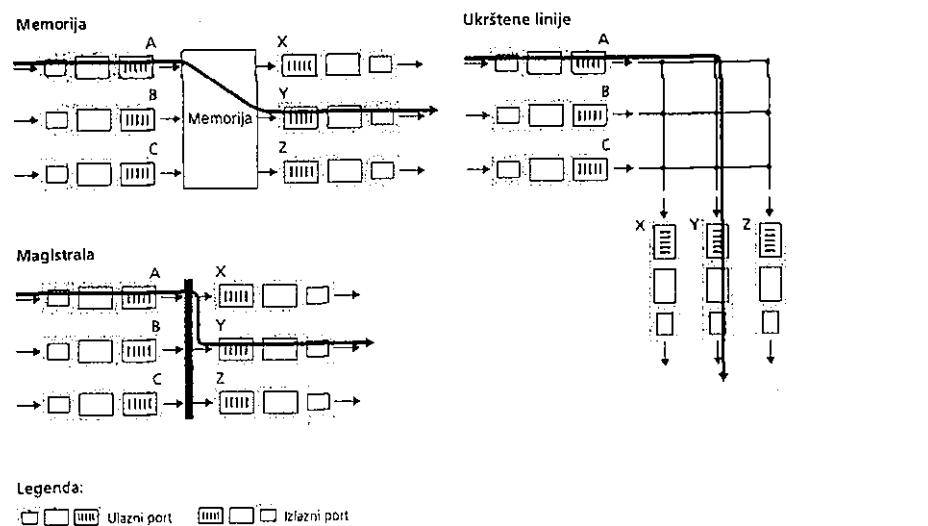
Drugi način da se ubrza pretraživanje je da se stavke iz tabele prosleđivanja kojima je nedavno pristupano čuvaju u kesu [Feldmeier 1988]. Potencijalni problem ovde predstavlja veličina kesa. Predložene su i brže strukture podataka koje omogućavaju da se stavke tabele prosleđivanja pronađu u  $\log(N)$  koraka [Waldvogel 1997] ili koje komprimuju tabelu pretraživanja na novi način [Brodnik 1997]. U [Gupta 1998] opisano je pretraživanje zasnovano na hardveru optimizirano za opšti slučaj u kojem tražena adresa ima najviše 24 značajna bita.

Kada se jednom utvrdi izlazni port paketa, on se prosleđuje u komutator. Međutim, paket može privremeno da se blokira pre ulaska u komutator (zbog toga što je on trenutno zauzet paketima iz drugih ulaznih portova). Blokirani paket zato mora da čeka u redu ulaznog porta i da kasnije pređe u komutatorsku mrežu. U odeljku 4.3.4 pobliže ćemo razmotriti blokiranje, čekanje u redu i rasporedavanje paketa (u ulaznim i izlaznim portovima) u ruteru.

### 4.3.2 Komutatorska mreža

Komutatorska mreža se nalazi u samom srcu rutera. Paketi se upravo kroz komutator zaista i komutiraju (tj. prosleđuju) iz ulaznog porta u izlazni. Komutiranje se može obaviti na više načina kao sto je prikazano na slici 4.8.

- ♦ **Komutiranje preko memorije.** Najjednostavniji prvi ruteri često su bili tradicionalni računari u kojima se komutiranje između ulaznih i izlaznih portova obavljalo pod direktnom kontrolom procesora (procesora rutiranja). Ulazni i izlazni portovi funkcionali su kao uobičajeni U/I uređaji u uobičajenom operativnom sistemu. Ulazni port je generisao prekid procesoru rutiranja uvek kada bi pn-



Slika 4.8 ♦ Tri tehnike komutiranja

stigao paket. Paket se tada kopirao iz ulaznog porta u procesorsku memoriju. Procesor rutiranja je zatim izdvajao određenu adresu iz zaglavlja, tražio odgovarajući izlazni port u tabeli prosledivanja i kopirao paket u privremene memorije izlaznog porta. Obratite pažnju na to da ako je memorijski propusni opseg takav da u memoriju može da se upiše ili da nje pročita B paketa u sekundi, tada će ukupna propusna moć komutatora (ukupna brzina kojom se paketi prenose iz ulaznih u izlazne portove) biti manja od S/2.

Mnogi savremeni ruteri takođe komutiraju putem memorije. Međutim, glavna razlika u odnosu na stare ruter je to što traženje određene adrese i smeštanje paketa na odgovarajuće mesto u memoriji izvršavaju procesori na ulaznoj linijskoj kartici. Na neki način, ruter koji komutiraju putem memorije veoma liče na multiprocesore sa deljenom memorijom jer procesori na linijskoj kartici smeštaju pakete u memoriju odgovarajućeg izlaznog porta. Cisco komutatori serije Catalyst 8500 [Cisco 8500 1999] i ruteri serije Bay Networks Accelar 1200 komutiraju pakete putem deljene memorije. Apstraktan model za proučavanje komutiranja putem memorije i poređenja sa ostalim vrstama komutiranja može se naći u [Iyer 2002].

- ♦ **Komutiranje putem magistrale.** Kod ovog rešenja ulazni portovi prenose paket direktno u izlazni port preko zajedničke magistrale, bez intervencije procesora rutiranja (obratite pažnju na to da kada se komutira preko memorije, paket takođe mora da prođe kroz sistemsku magistralu kada ulazi i kada izlazi iz memorije). Mada se procesor rutiranja ne uključuje u transfer na magistrali, pošto se magistrala deli po njoj se može prenosi samo jedan po jedan paket. Ako paket stigne na ulazni port dok je magistrala zauzeta transferom drugog paketa, on se blokira ispred komutatora i ostaje u redu čekanja ulaznog porta. Pošto svaki paket mora da prođe kroz tu jednu magistralu, propusni opseg komutiranja u ruleni ograničen je brzinom magistrale.

Pošto današnja tehnologija omogućava propusne opsege magistrale veće od jednog gigabita u sekundi, komutiranje putem magistrale često je dovoljno za ruter koji rade u mrežama za pristup i mrežama preduzeća (na primer, lokalnim i korporacijskim mrežama). Komutiranje putem magistrale koristi se u nizu današnjih ruteru uključujući i Cisco 1900 [Cisco Switches 1999] koji komutira pakete preko magistrale za razmenu paketa od jednog Gb/s, Sistem CoreBuilder 5000 kompanije 3Com [Kapoor 1997] međusobno povezuje portove koji se nalaze na različitim modulima komutatora preko magistrale podataka PacketChannel sa propusnim opsegom od 2 Gb/s.

- ♦ **Komutiranje putem višestruko povezane mreže.** Jedan od načina da se prevaziđe ograničenje na propusni opseg jedne jedine zajedničke magistrale je upotreba jedne složenije višestruko povezane mreže kakve su se u prošlosti koristile za međusobno povezivanje procesora i višeprocesorskoj arhitekturi računara. Komutator sa unakrsnim linijama je višestruko povezana mreža koja se sastoji od  $2n$  magistrala koje povezuju  $n$  ulaznih portova sa  $n$  izlaznih portova, kao što

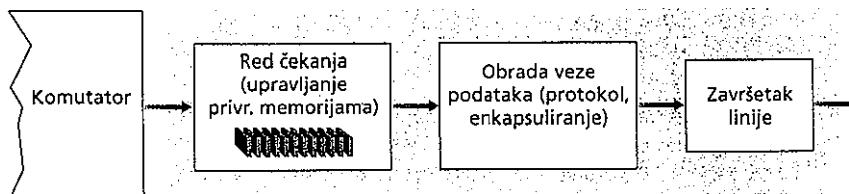
je prikazano na slici 4.8. Paket koji stigne na ulazni port putuje po horizontalnoj magistrali vezanoj za ulazni port, dok se ne ukrsti sa vertikalnom magistralom koja ide do željenog izlaznog porta. Ako je vertikalna magistrala koja ide ka izlaznom portu slobodna, paket se prenosi u izlazni port. Ako je vertikalna magistrala zauzeta transferom paketa iz nekog drugog ulaznog porta u ovaj isti izlazni port, pristigli paket se blokira i mora da ostane u redu čekanja ulaznog porta. Delta i Omega komutatorske mreže takođe se predlažu kao višestruko povezane mreže između ulaznih i izlaznih portova. U knjizi [Tobagi 1990] naći ćete pregled komutatorskih arhitektura. Komutatori familije Cisco 12000 [Cisco 12000 1998] koriste višestruko povezanu mrežu koja omogućava čak 60 Gb/s kroz komutatorsku mrežu. Trenutni trend u dizajnu višestruko povezanih mreža [Keshav 1998] je da se IP datagram promenljive dužine fragmentira u čelije fiksne dužine, pa se te čelije fiksne dužine označe i komutiraju kroz višestruko povezanu mrežu. Od tih čelija se zatim u izlaznom portu ponovo sklapa prvobitni datagram. Fiksna dužina čelije i označavanje značajno pojednostavljaju i ubrzavaju komutiranje paketa kroz višestruko povezanu mrežu.

### 4.3.3 Izlazni portovi

Obrada u izlaznom portu, prikazana na slici 4.9, uzima pakete koji su se čuvali u memoriji izlaznog porta i prenosi ih preko izlaznog linka. Obrada protokolom veze podataka i završetak linije su funkcije sloja veze i fizičkog sloja koje sarađuju sa ulaznim portom na drugom kraju izlaznog linka kao što je opisano u odeljku 4.3.1. Redovi čekanja i upravljanje privremenom memorijom potrebni su kada komutatorska mreža predaje pakete izlaznom portu brzinom većom od brzine izlaznog linka. Sada ćemo obraditi Čekanje u redu na izlaznom portu.

### 4.3.4 Gde dolazi do čekanja u redu?

Ako pogledamo funkcije ulaznih i izlaznih portova i konfiguraciju prikazanu na slici 4.8, očigledno je da se redovi paketa mogu stvoriti i na ulaznim i na izlaznim portovima. Važno je malo detaljnije razmotriti ove redove čekanja jer kako ti redovi rastu



Slika 4.9 ♦ Obrada u izlaznom portu

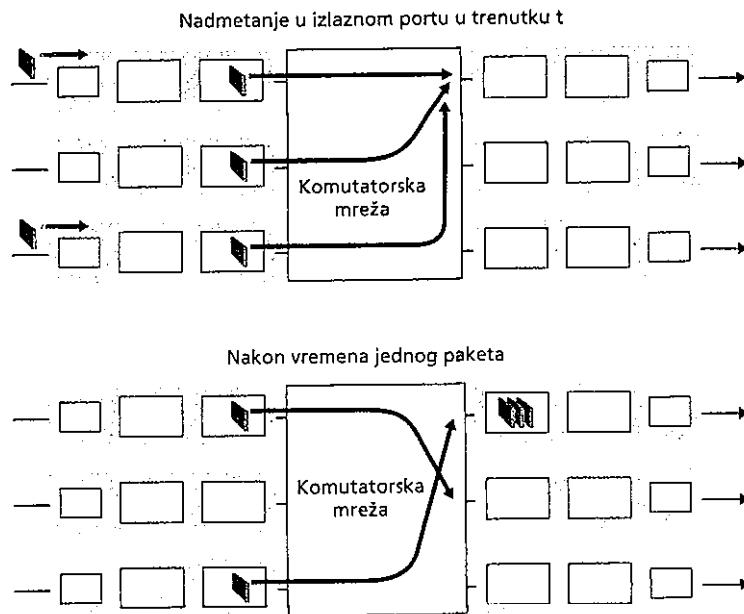
privremena memorija rutera će se na kraju potrošiti i doći će do **gubitka paketa**. Možda se sećate da smo u prethodnim opisima rekli da se paketi gube u mreži ili ispuštaju na ruteru. Upravo ovde u tim ruterovim redovima čekanja se takvi paketi ispuštaju i gube. Tačno mesto gde se paket gubi (u redu čekanja ulaznog porta ili u redu čekanja izlaznog porta) zavisiće od opterećenja saobraćajem, relativne brzine komutatorske mreže i brzine linija, što ćemo detaljnije opisati.

Uzmimo da su brzine ulazne i izlazne linije identične i da postoji  $n$  ulaznih i  $n$  izlaznih portova. **Brzinu komutatora** definisemo kao brzinu kojom komutatorska mreža može da prenese pakete od ulaznih do izlaznih portova. Ako je brzina komutatora bar  $n$  puta veća od brzine ulazne linije, u ulaznom portu neće doći do čekanja u redu. To je zato što Čak i u najgorem slučaju da svih  $n$  ulaznih linija prima pakete komutator će moći da prenese svih  $n$  paketa iz ulaznog porta u izlazni port u vremenu potrebnom da svaki od  $n$  ulaznih portova (istovremeno) primi *pojedinačni* paket. Ali, šta se može dogoditi na izlaznim portovima? Pretpostavimo opet daje komutatorska mreža bar  $n$  puta brža od linija. U najgorem slučaju, paketi koji su stigli na svaki od  $n$  ulaznih portova biće namenjeni u *isti* izlazni port. U tom slučaju, u vremenu potrebnom da se primi (ili pošalje) samo jedan paket, na izlazni port će stići  $n$  paketa. Pošto izlazni port može u tom vremenu da pošalje samo jedan paket (vreme prenosa paketa),  $n$  pristiglih paketa moraće da čeka u redu za prenos preko izlaznog linka. Tada, tokom vremena potrebnog da se prenese samo jedan od  $n$  paketa koji su već bili u redu za čekanje, može da stigne još  $n$  paketa. I tako dalje. U jednom trenutku, broj paketa u redu za čekanje može da postane tako veliki da se utroši sav memoriski prostor izlaznog porta i tada se paketi ispuštaju.

Čekanje u redu izlaznog porta prikazano je na slici 4.10. U trenutku / na svaki od ulaznih portova stigao je po jedan paket i svi su namenjeni za gornji izlazni port. Ako pretpostavimo da su brzine linija identične, a da komutator radi tri puta većom brzinom od linija, nakon jedne jedinice vremena (tj. u vremenu potrebnom da se primi ili pošalje jedan paket) sva tri prvobitna paketa su preneta na izlazni port i čekaju u redu da se prenesu. U sledećoj jedinici vremena jedan od ta tri paketa biće prenet preko izlaznog linka. U našem primeru, na ulaznu stranu komutatora stižu dva *nova* paketa; jedan od njih namenjen je za gornji izlazni port.

Jedna posledica čekanja u redu u izlaznom portu je da raspoređivač **paketa** u izlaznom portu mora da izabere jedan od paketa koji čekaju u redu za prenos. Taj izbor može biti sasvim jednostavan, kao sto je redosled pristizanja (*first-come-first-served*, FCFS) ili složenije raspoređivanje, kao što je ponderisano fer čekanje (*weighted fair queuing*, WFQ) u kojem se izlazni link deli na fer način između različitih konekcija sa kraja na kraj čiji paketi čekaju na prenos. Raspoređivanje paketa igra ključnu ulogu u obezbeđivanju **garancija kvaliteta usluge**. Ova tema biće opširno obradena u poglavljju 7. Opis disciplina u raspoređivanju paketa na izlaznom portu možete naći u [Cisco Queue 1995].

Slično tome, ako nema dovoljno memorije da se ulazni paket smesti u privremenu memoriju potrebno je doneti odluku da li da se ispušti paket koji stiže (politika poznata pod imenom **ispustiti poslednji**) ili da se ukloni jedan ili više paketa



**Slika 4.10** ♦ Red čekanja na izlaznom portu

iz reda čekanja i napravi mesto za paket koji upravo stiže. U nekim slučajevima korisnije je da se ispusti paket (ili stavi oznaka u zaglavlje) *pre* nego što se privremena memorija napuni da bi se pošiljaocu dao signal da postoji zagušenje. U dokumentima [Labrador 1999, HolJot 2002] predložen je i analiziran niz politika za ispuštanje i označavanje paketa (koje imaju zajednički naziv algoritma za aktivno upravljanje redovima Čekanja (*active queue management*, AQM)). Jedan od AQM algoritama koji je najčešće proučavan i implementiran je algoritam RED (Random Early Detection). Ako se koristi RED, izračunava se ponderisani prosek dužine izlaznog reda čekanja. Ako je u trenutku kada paket stiže prosečna dužina reda za čekanje manja od praga za minimum  $min_{th}$ , on se prima u red za čekanje. Obrnuto, ako je u trenutku kada paket stigne red za čekanje pun ili je prosečna dužina reda veća od praga za maksimum  $max_{th}$ , on se označava ili ispušta. Na kraju, ako paket stigne i zatekne prosečnu dužinu reda u intervalu  $[min_{th}, max_{th}]$ , on se označava ili ispušta po verovatnoći koja obično zavisi od prosečne dužine reda, od  $min_{th}$  i od  $max_{th}$ . Do sada je predložen niz funkcija za izračunavanje verovatnoće označavanja i ispuštanja, a različite verzije algoritma RED su analitički modelirane, simulirane i

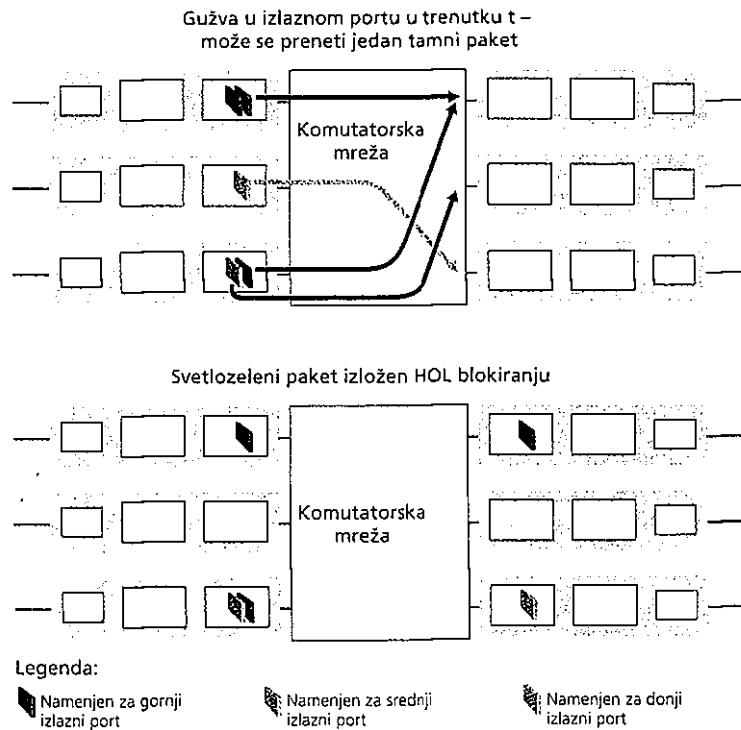
implementirane. U [Christiansen 2001] i [Floyd 2002] dati su pregledi i predloži za dodatno Čitanje.

Ako komutator nije dovoljno brz (u odnosu na brzinu ulazne linije) da bi se *svi* pristigli paketi preneli bez zastoja kroz komutator, doći će do redova čekanja i u ulaznim portovima gde će paketi čekati da budu preneti kroz komutator u izlazni port. Kao ilustraciju jedne važne posledice ovog čekanja u redu razmotrite komutator sa unakrsnim linijama i prepostavite (1) da su brzine svih linkova identične, (2) da se jedan paket može preneti iz bilo kog ulaznog porta do datog izlaznog porta u vremenskom intervalu potrebnom da se jedan paket primi na ulaznom linku i (3) da se paketi prenose iz datog ulaznog reda čekanja u željeni izlazni red čekanja po principu FCFS. Istovremeno prenošenje više paketa moguće je samo ako su im različiti izlazni portovi. Međutim, ako su na početku dva ulazna reda dva paketa namenjena za isti izlazni red čekanja, jedan od njih će se blokirati i moraće da čeka u ulaznom redu - komutator može u isto vreme da prenosi samo jedan paket u dati izlazni port.

Na slici 4.11 dat je primer u kojem su dva paketa (tamnozelena) na početku svojih ulaznih redova čekanja namenjena za isti gornji desni izlazni port. Prepostavimo da komutator odluči da prenese paket sa početka gornjeg levog reda čekanja. U tom slučaju tamnozeleni paket u donjem levom redu mora da čeka. Ali, ne samo što ovaj tamnozeleni paket mora da čeka, već mora da čeka i svetlozeleni paket koji se u donjem levom redu nalazi iza njega, iako nema *nikakvog* nadmetanja za srednji desni izlazni port (odredište svetlozelenog paketa). Ovaj fenomen poznat je kao HOL (head-of-the-line) blokiranje u komutatoru sa ulaznim redom čekanja - paket u ulaznom redu mora da čeka na prenos kroz komutator (čak i kada je njegov izlazni port sloboden) jer ga blokira drugi paket u redu ispred njega. Karol u knjizi [Karol 1987] objašnjava da zbog HOL blokiranja ulazni red čekanja dostiže neograničenu dužinu (to znači da će doći do značajnog gubitka paketa) pod određenim prepostavkama čim brzina pristizanja paketa na ulazne linkove dostigne samo 58 procenata njihovog kapaciteta. U knjizi [McKeown 1997b] opisan je niz rešenja za HOL blokiranje.

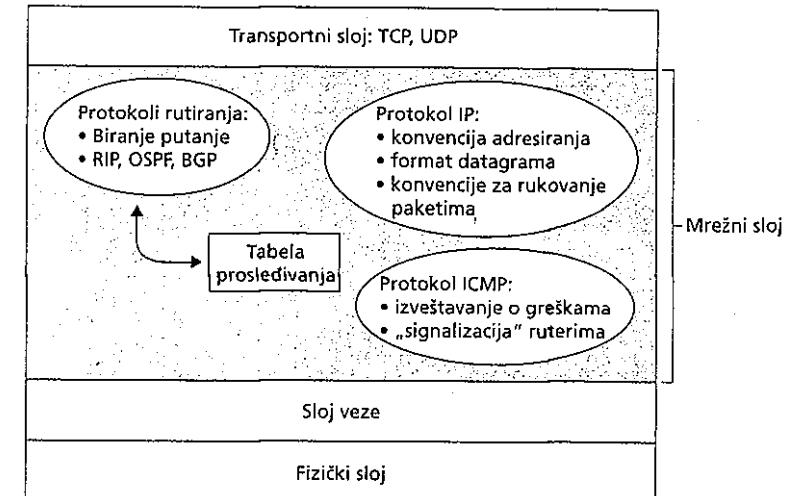
#### 4.4 Internet protokol (IP): Prosleđivanje i adresiranje na Internetu

Do sada smo u ovom poglavlju opisivali prosleđivanje i adresiranje mrežnog sloja ne pominjući nijednu konkretnu računarsku mrežu. U ovom odeljku obratićemo pažnju na to kako se prosleđivanje i adresiranje vrše na Internetu. Videćemo da su prosleđivanje i adresiranje značajne komponente Internet Protocola (IP). Danas su u upotrebi dve verzije protokola IP. Prvo ćemo ispitati opšteprihváćeni Internet protokol verzije 4, poznat kao IPv4 [RFC 791]. Na kraju odeljka ispitacemo IP verziju 6 [RFC 2373; RFC 2460] za koji se predlaže da tokom narednih godina zameni IPv4.



**Slika 4.11 ♦ HOL blokiranje u komutatoru sa ulaznim redom čekanja**

Ali pre nego što krenemo u pohod na IP, vratimo se za jedan korak i razmotrimo komponente od kojih se sastoji mrežni sloj Interneta. Kao što je prikazano na slici 4.13, mrežni sloj Interneta ima tri glavne komponente. Prva komponenta je protokol IP, tema ovog odeljka. Druga važna komponenta mrežnog sloja je komponenta za rutiranje koja utvrđuje putanje kojom će datagram ići od izvora do odredišta. Videli smo ranije da protokoli rutiranja izračunavaju tabele prosledivanja koje se koriste za prosledivanje paketa kroz mrežu. U odeljku 4.6 proučćemo Internetove protokole rutiranja. Poslednja komponenta mrežnog sloja je sredstvo za prijavljivanje grešaka u datagramima i reagovanje na zahteve za određenim informacijama u mrežnom sloju. U odeljku 4.4.3 obradićemo ICMP (*Internet Control Message Protocol*), Internetov protokol mrežnog sloja za izveštavanje o greškama i davanje informacija.

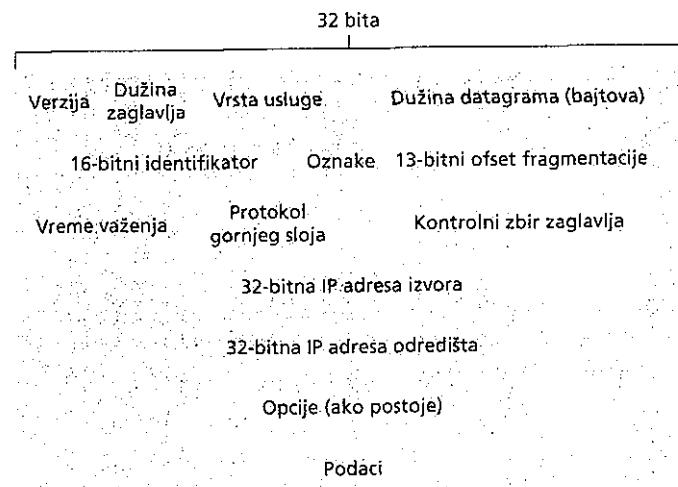


**Slika 4.12 ♦ Pogled u unutrašnjost mrežnog sloja Interneta**

#### 4.4.1 Format datagrama

Sigurno se sećate da se paket mrežnog sloja naziva *datagramom*. Proučavanje IP-a počinjemo pregledom sintakse i semantike datagrama IPv4. Možda mislite da nema ničeg suvremenijeg od sintakse i semantike bitova nekog paketa. I pored toga, datagram igra centralnu ulogu na Internetu - svaki student i profesionalac u umrežavanju mora da ga vidi, sagleda i savlada. Format datagrama IPv4 prikazan je na slici 4.13. Ključna polja 1IPv4 datagrama su sledeća:

- ♦ **Broj verzije.** Ova četiri bita navode verziju protokola IP za taj datagram. Prema broju verzije ruter može da odredi kako treba protumačiti ostatak IP datagrama. Različite verzije protokola IP koriste različite formate datagrama. Format datagrama za trenutnu verziju IP-a, IPv4, prikazan je na slici 4.13. Format datagrama za novu verziju IP-a (IPv6) opisan je pri kraju ovog odeljka.
- ♦ **Dužina zaglavљa.** Pošto datagram IPv4 može da sadrži promenljivi broj opcija (koje su uključene u zaglavje datagrama IPv4) ova četiri bita su potrebna da bi se odredilo mesto u IP datagramu gde počinju sami podaci. Većina IP datagrama ne sadrži opcije, pa tipični IP datagram ima zaglavљje od 20 bajtova.
- ♦ **Vrsta usluge.** Bitovi za vrstu usluge (*type of service*, TOS) uključeni su u IPv4 zaglavljive da bi se omogućilo prepoznavanje različitih „tipova“ IP datagrama (na primer, datagrami koji zahtevaju što manje kašnjenje, veliki propusni opseg ili pouzdanost). Na primer, moglo bi da bude korisno razlikovati datagrame u real-



Slika 4.13 ♦ Format datagrama IPv4

nom vremenu (na primer, iz neke aplikacije za IP telefoniju) od saobraćaja koji nije u realnom vremenu (na primer, FTP). Nedavno je jedan od glavnih proizvođača opreme za rutiranje (Cisco) uveo pravilo da prva tri bita u vrsti usluge definišu nivo usluga koje ruter može da obezbedi. Konkretno nivo usluge koja se pruža je pitanje politike koju utvrđuje administrator ruteru. Pitanje diferenciranih usluga detaljno ćemo obraditi u poglavljiju 7.

- ♦ **Dužina datagrama.** Ovo je ukupna dužina IP datagrama (zaglavje plus podaci), merena u bajtovima. Pošto je ovo polje dužine 16 bitova, teoretski maksimalna veličina IP datagrama je 65.535 bajtova. Međutim, retko se dešava da su datagrami veći od 1.500 bajtova.
- ♦ **Identifikator, oznake i offset fragmentacije.** Ova tri polja vezana su za takozvanu IP fragmentaciju, temu koju ćemo uskoro detaljno obraditi. Zanimljivo je da nova verzije IP-a, IPv6, ne dozvoljava da se fragmentiranje vrši u ruterima.
- ♦ **Vreme važenja.** Vreme važenja (*time-to-live, TTL*) je polje koje služi da bi se sprečilo večno kruženje datagrama u mreži (na primer, zbog dugotrajne petlje rutiranja). Ovo polje se smanjuje za 1 u svakom ruteru gde se datagram obraduje. Ako polje TTL dostigne vrednost 0, datagram mora da se odbaci.

♦ **Protokol.** Ovo polje se koristi samo kada IP datagram stigne do konačnog odredišta. Vrednost u ovom polju ukazuje na konkretni protokol transportnog sloja kojem bi trebalo predati podatke iz IP datagrama. Na primer, vrednost 6 znači da podatke treba predati protokolu TCP, dok vrednost 17 znači da podatke treba predati protokolu UDP. Spisak svih mogućih brojeva možete naći u knjigama [RFC 1700; RFC 3232]. Obratite pažnju na to da broj protokola u IP datagramu ima ulogu potpuno analognu broju porta u segmentu transportnog sloja. Broj protokola je lepkak koji vezuje mrežni i transportni sloj, dok je broj porta lepkak koji vezuje transportni i aplikacijski sloj. U poglavljiju 5 ćemo videti da okvir sloja veze takođe sadrži posebno polje koje povezuje sloj veze sa mrežnim slojem.

♦ **Kontrolni zbir zaglavlja.** Kontrolni zbir zaglavlja pomaže ruteru u otkrivanju biliških grešaka u primljenom IP datagramu. Kontrolni zbir zaglavlja izračunava se tako što se svaka dva bajta u zaglavljima uzmaju kao broj, pa se ti brojevi sabiju aritmetikom komplementajedinice. Kao što je opisano u odeljku 3.3, komplement jedinice za ovaj zbir, poznat kao Internetski kontrolni zbir, čuva se u polju kontrolnog zbirja. Ruter izračunava kontrolni zbir zaglavlja za svaki primljeni IP datagram i otkriva grešku ako kontrolni zbir u zaglavljima datagrama nije jednak izračunatom kontrolnom zbiru. Ruteri obično odbacuju datagrame u kojima otkriju grešku. Obratite pažnju na to da kontrolni zbir mora da se ponovo izračuna i sačuva u svakom ruteru zato što se polje TTL i eventualna polja opcija mogu promeniti. Zanimljiv opis brzih algoritama za izračunavanje Internetskog kontrolnog zbirja nalazi se u knjizi [RFC 1071]. Ovde se često postavlja pitanje zbog Čega TCP/IP vrši proveravanje grešaka i u transportnom i u mrežnom sloju? Ima mnogo razloga za to duplikiranje. Prvo, primetiće da se u IP sloju kontrolni zbir pravi samo za zaglavje, dok se kontrolni zbir TCP/UDP izračunava nad celim TCP ili UDP segmentom. Drugo, TCP/UDP i IP ne moraju biti iz iste familije protokola. TCP može u principu da se izvršava i preko drugih protokola (na primer, ATM-a), a IP može da prenosi podatke koji neće biti predati u TCP/UDP.

♦ **IP adrese izvora i odredišta.** Kada izvor napravi datagram, stavlja svoju IP adresu u polje IP adrese izvora i adresu krajnjeg odredišta u polje IP adrese odredišta. Često izvomi računar utvrđuje adresu odredišta pomoću DNS pretraživanja kao što je opisano u poglavljiju 2. IP adresiranje detaljno obradujemo u odeljku 4.4.2.

♦ **Opcije.** Polje opcija omogućava proširivanje IP zaglavlja. Namena je bila da se opcije zaglavlja koriste retko - otud i odluka da se smanji opterećenje tako što se informacije u poljima opcija ne uključuju u svako zaglavje datagrama. Međutim, već samo postojanje opcija komplikuje stvari - pošto zaglavja datagrama mogu da budu promenljive dužine, ne može se a priori utvrditi gde počinje polje podataka. Osim toga, pošto neki datagrami zahtevaju obradu opcija a drugi ne, vreme potrebno za obradu IP datagrama u ruteru može biti veoma različito. Ova razmatranja postaju posebno značajna za IP obradu u ruterima i računarima visokih performansi. Zbog tih i još nekih razloga, IP opcije su izbačene iz IPv6 zaglavlja, kao što se opisuje u odeljku 4.4.4.

♦ *Podaci (korisni podaci).* Na kraju, dolazimo do poslednjeg i najvažnijeg polja - razloga što datagram uopšte postoji! U većini slučajeva, polje podataka u IP datagramu sadrži segment transportnog sloja (TCP ili UDP) koji treba isporučiti na odredište. Međutim, polje podataka može da sadrži i druge vrste podataka, kao što su ICMP poruke (koje opisuјemo u odeljku 4.4.3).

Obratite pažnju na to da IP datagram sadrži ukupno 20 bajtova zaglavlja (pod pretpostavkom da nema opcija). Ako datagram prenosi TCP segment, tada svaki (nefragmentirani) datagram pored poruke iz aplikacijskog sloja prenosi ukupno 40 bajtova zaglavlja (20 bajtova IP zaglavlja i 20 bajtova TCP zaglavlja).

#### Fragmentacija IP datagrama

U poglavlju 5 ćemo videti da ne mogu svi protokoli sloja veze da prenose pakete iste veličine. Neki protokoli mogu da nose „velike“ pakete, dok drugi protokoli mogu da nose samo „male“. Na primer, Ethernet paketi ne mogu da prenesu više od 1.500 bajtova podataka, dok paketi mnogih regionalnih linkova ne mogu da prenose više od 576 bajtova. Maksimalna količina podataka koju može da prenosi paket sloja veze naziva se maksimalna jedinica za transfer (*maximum transfer unit*, MTU). Pošto se svaki IP datagram radi prenosa od jednog ruteru do sledećeg enkapsulira u paket sloja veze, MTU protokola u sloju veze postavlja strogo ograničenje na dužinu IP datagrama. Stroga granica za veličinu IP datagrama nije veliki problem. Problem je što linkovi na ruti između pošiljaoca i odredišta možda koriste različite protokole sloja veze i svaki od tih protokola može da ima drugačiji MTU.

Da biste bolje razumeli ovaj problem, zamislite da ste vi jedan ruter koji povezuje nekoliko linkova od kojih svaki izvršava drugačiji protokol sloja veze sa različitim MTU-ovima. Prepostavimo da ste primili IP datagram od jednog linka, proverili u svojoj tabeli prosleđivanja koji je izlazni link i utvrdili da taj izlazni link ima MTU manji od dužine primljenog IP datagrama. Nastupa panika - kako da stisnete veliki IP paket u polje korisnih podataka paketa sloja veze? Rešenje ovog problema je da se podaci iz IP datagrama „fragmentiraju“ na dva ili više manjih IP datagrama, a zatim se ti manji datagrami pošalju preko izlaznog linka. Svaki od ovih manjih datagrama naziva se fragment.

Fragmenti moraju ponovo da se sastave pre nego što se predaju transportnom sloju na odredištu. Zaista, i TCP i UDP očekuju da od mrežnog sloja prime kompletne nefragmentirane segmente. Projektanti protokola IPv4 smatrali su da bi ponovno sastavljanje (i eventualno ponovno fragmentiranje) datagrama u rutera uvelo značajnu komplikaciju u protokol i umanilo performanse rutera. (Da ste vi ruter, da li biste pored svog ostalog posla koji imate voleli još i da sastavljate fragmente?) Držeći se principa da mrežni sloj treba da ostane jednostavan, projektanti IPv4 odlučili su da zadatak ponovnog sastavljanja datagrama prepuste krajnjim sistemima, a ne mrežnim rutera.

Kada odredišni računar primi niz datagrama sa istog izvora, on mora da utvrdi da li su neki od tih datagrama fragmenti nekog prvobitno većeg datagrama. Ako utvrdi da su neki datagrami u stvari fragmenti, on mora da utvrdi kada je primio poslednji fragment i kako bi primljene fragmente trebalo ponovo sastaviti da bi se dobio prvobitni datagram. Da bi odredišni računar mogao da obavi te zadatke ponovnog sastavljanja, projektanti IP-a (verzija 4) stavili su u IP datagram polja *identifikacija*, *oznake* i *ofseta fragmentacije*. Kada se datagram pravi, otpremni računar mu osim izvorne i odredišne adrese dodeljuje i identifikacioni broj. Otpremni računar povećava identifikacioni broj za svaki poslati datagram. Kada ruter mora da fragmentira datagram, svaki novi datagram (tj. fragment) označava se izvornom adresom, odredišnom adresom i identifikacionim brojem prvobitnog datagrama. Kada odredište primi niz datagrama od istog otpremnog računara, on može da ispitava identifikacione brojeve datagrama i utvrdi koji su datagrami u stvari fragmenti jednog istog većeg datagrama. Postoje IP nepouzdana usluga, jedan ili više fragmenata možda neće ni stići na odredište. Zbog toga, da bi odredišni računar bio potpuno siguran daje primio poslednji fragment prvobitnog datagrama, poslednji fragment ima u bitu oznake vrednost 0, dok svi ostali fragmenti imaju vrednost 1. Osim toga, da bi odredišni računar mogao da utvrdi da li neki fragment nedostaje (i takođe da bi bio u stanju da ponovo sastavi fragmente u pravilnom redosledu), on koristi polje ofseta da odredi lokaciju fragmenta u prvobitnom IP datagramu.

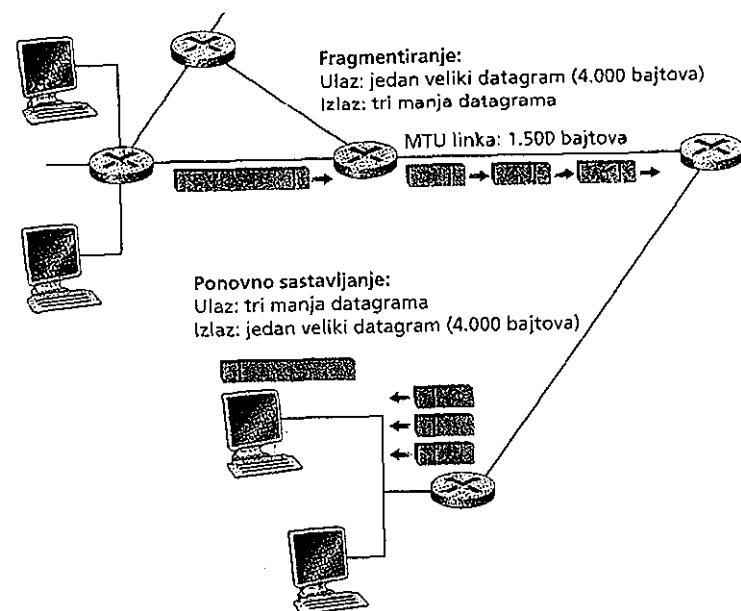
Na slici 4.14 dat je jedan primer. Datagram od 4.000 bajtova (20 bajtova IP zaglavlja plus 3.980 bajtova IP korisnih podataka) stiže na ruter i mora da se prosledi na link čiji MTU iznosi 1.500 bajtova. To znači da 3.980 bajtova podataka iz prvobitnog datagrama treba podeliti u tri fragmenta (od kojih je svaki takođe IP datagram). Pretpostavite da prvobitni datagram ima identifikacioni broj 777. Karakteristike tri dobi-jena fragmenta prikazane su u tabeli 4.2. Vrednosti u tabeli 4.2 odražavaju zahtev da broj bajtova prvobitnih korisnih podataka u svim fragmentima osim poslednjeg mora biti deljiv sa 8 i da se vrednost ofseta navodi u jedinicama od po 8 bajtova.

Korisni podaci iz datagrama predaju se transportnom sloju na odredištu tek pošto IP sloj potpuno rekonstruiše prvobitni IP datagram. Ako jedan ili više fragmenata ne stigne do odredišta, datagram se odbacuje i ne predaje se transportnom sloju. Ali, kao što smo naučili u prethodnom poglavljiju, ako se u transportnom sloju koristi TCP, tada će TCP obaviti opravak od ovog gubitka tako što će zahtevati da izvor ponovo pošalje podatke iz prvobitnog datagrama.

Na veb lokaciji ove knjige pripremili smo Java aplet koji pravi fragmente. Vi zadate veličinu pristiglog datagrama, MTU i identifikaciju pristiglog datagrama. Aplet vam automatski generiše fragmente. Potražite adresu <http://www.awl.com/kurose-ross>.

| Fragment | Veličina podataka                  | ID  | Offset                                                             | Oznaka                    |
|----------|------------------------------------|-----|--------------------------------------------------------------------|---------------------------|
| Prvi     | 1.480 bajtova                      | 777 | 0 (znači da podatke treba umetnuti od pozicije 0)                  | 1 (znači da ima još)      |
| Drugi    | 1.480 bajtova                      | 777 | 185 (znači da podatke treba umetnuti počev od pozicije 185-8=1480) | 1 (znači da ima još)      |
| Treći    | 1.020 bajtova (3980 - 1480 - 1480) | 777 | 370 (znači da podatke treba umetnuti počev od pozicije 380-8=2960) | 0 (znači da je poslednji) |

Tabela 4.2 ♦ IP fragmenti



Slika 4.14 ♦ IP fragmentiranje i ponovno sastavljanje

#### 4.4.2 IPv4 adresiranje

Sada skrećemo pažnju na adresiranje protokola IPv4. Mada adresiranje može da izgleda kao jednostavna i možda dosadna tema, nadamo se da ćete do kraja ovog poglavlja uvideti da adresiranje na Internetu nije samo sočna, suptilna i zanimljiva tema već i daje suštinski važno za Internet. IPv4 adresiranje odlično je obrađeno u [Semeria 1996] i prvom poglavlju knjige [Stewart 1999].

Međutim, pre nego što predemo na IP adresiranje, moramo nešto reći o načinu na koji su računari i ruteri povezani u mrežu. Računar obično ima samo jedan link prema mreži. Kada IP u tom računaru hoće da pošalje datagram, on će to uraditi preko ovog linka. Granica između računara i fizičkog linka naziva se interfejs. Razmotrite sada ruter i njegove interfejsе. Postoje ruterov zadatak da prima datagrame u jednom linku i prosledi ga na neki drugi link, ruter obavezno mora biti povezan sa dva ili više linkova. Granica između ruteru i bilo kojeg od njegovih linkova takođe se naziva interfejs. Ruter znači ima više interfejsa, po jedan za svaki link. Pošto svaki računar i ruter može da šalje i prima IP datagrame, IP zahteva da svaki interfejs računara i rutera ima vlastitu IP adresu. Prema tome, IP adresa je tehnički pridružena interfejsu, a ne računaru ili ruteru koji sadrži taj interfejs.

Svaka IP adresa dugačka je 32 bita (četiri bajta), pa tako postoji ukupno  $2^{32}$  mogućih IP adresa. Kako je  $2^{31}$  približno  $4,3 \cdot 10^9$ , evidentno je da postoji oko 4 milijardi mogućih IP adresa. Te adrese se obično pišu u takozvanoj decimalnoj notaciji sa tačkama, u kojoj se svaki bajt adrese zapisuje u decimalnom obliku a od ostalih bajtova u adresi se razdvaja tačkama. Uzmite kao primer IP adresu 193.32.216.9. Broj 193 je decimalna vrednost prvih osam bitova u adresi; broj 32 je decimalna vrednost drugih osam bitova u adresi itd. Prema tome, adresa 193.32.216.9 u binarnom obliku bi bila

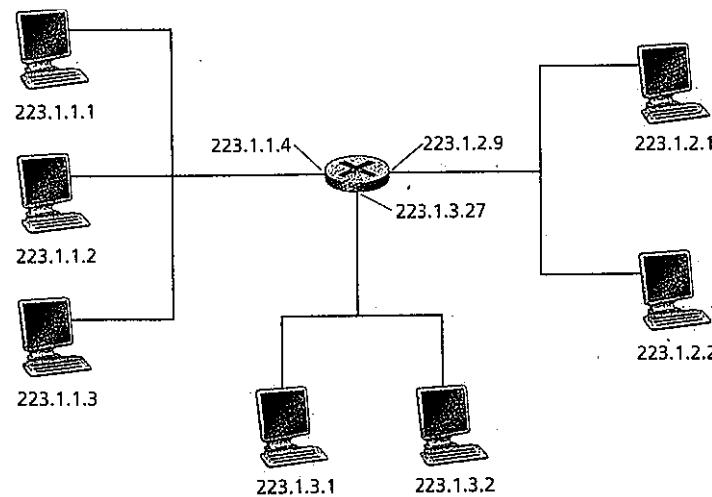
11000001 00100000 11011000 00001001

Svaki interfejs u svakom računaru i ruteru u globalnom Internetu mora da ima IP adresu koja je globalno jedinstvena (osim interfejsa iza NAT-ova koje opisuјemo na kraju ovog odeljka). Ove adrese, međutim, ne mogu da se biraju na proizvoljan način. Jedan deo IP adrese interfejsa određuje podmrežu na koju je povezan.

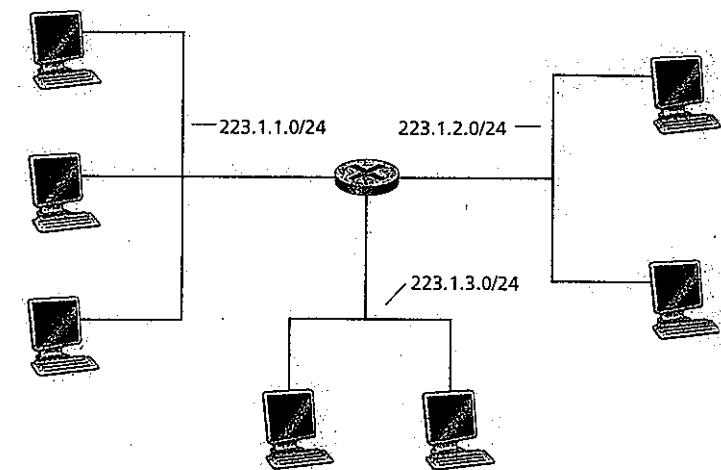
Na slici 4.15 imamo jedan primer IP adresiranja i interfejsa. Na ovoj slici, jedan ruter (sa tri interfejsa) povezuje sedam računara. Pogledajte bolje IP adrese dodeljene interfejsima računara i ruteru; treba obratiti pažnju na nekoliko stvari. Tri računara u gornjem levom delu slike 4.15 i interfejs ruteru sa kojim su povezani imaju IP adrese u obliku 223.1.1.xxx. To jest, svi imaju ista 24 bita IP adrese. Oni su takođe međusobno povezani jednim jedinim fizičkim linkom *bez ikakvih posrednih ruteru*. (Ta mreža bi mogla da bude, na primer, Ethernet LAN pa bi u tom slučaju interfejsi bili povezani Ethernet habom ili Ethernet komutatorom; videti poglavljje 5.) U IP žargonu, interfejsi li ta tri računara i gornji levi interfejs u mjeru Čine jednu podmrežu [RFC 950]. (Podmreža se u internetskoj literaturi naziva i *IP mrežom* ili prostо *mrežom*.) IP adresiranje dodeljuje adresu ovoj podmreži: 223.1.1.0/24, gde

oznaka „/24”, koja se ponekad naziva maska podmreže, znači da levih 24 bita 32 bitne veličine predstavljaju adresu podmreže. Podmreža 223.1.1.0/24 se znači sastoji od tri interfejsa računara (223.1.1.1, 223.1.1.2 i 223.1.1.3) i jednog interfejsa rutera (223.1.1.4). Svaki novi računar koji se poveže na podmrežu 223.1.1.0/24 morao bi da ima adresu u obliku 223.1.1.xxx. Na slici 4.15 prikazane su još dve podmreže: podmreža 223.1.2.0/24 i podmreža 223.1.3.0/24. Na slici 4.16 prikazane su tri IP podmreže sa slike 4.15.

IP definicija podmreže nije ograničena na Ethernet segmente koji povezuju više računara sa jednim interfejsom na ruteru. Da biste to bolje shvatili, razmotrite sliku 4.17 na kojoj su prikazana tri ruteru međusobno povezana linkovima od čvora do čvora. Svaki ruter ima tri interfejsa, po jedan za svaki link od čvora do čvora i jedan za link za difuzno emitovanje koji direktno povezuje ruter sa parom računara. Koje IP podmreže imamo ovde? Tri podmreže, 223.1.1.0/24, 223.1.2.0/24 i 223.1.3.0/24 slične su podmrežama koje smo videli na slici 4.15. Ali, obratite pažnju na to da u ovom primeru postoje još tri podmreže: jedna podmreža 223.1.9.0/24 za interfejs koji povezuju ruter R1 i R2; još jedna podmreža 223.1.8.0/24 za interfejs koji povezuju ruter R2 i R3; i treća podmreža 223.1.7.0/24 za interfejs koji povezuju ruter R3 i R1. Za opšti međusobno povezani sistem ruteru i računara možemo upotrebiti sledeće pravilo za definisanje podmreža u sistemu.



Slika 4.15 ♦ Adrese interfejsa i podmreže



Slika 4.16 ♦ Adrese podmreže

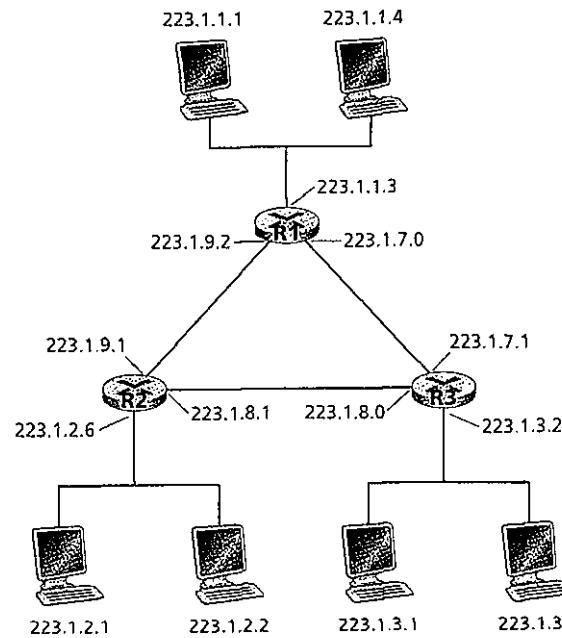
Da bi se odredile podmreže, prvo razdvajamo svaki interfejs od svog računara ili rutera i lako dobijamo ostrva izolovanih mreža na kom se interfejsi nalaze na krajevima izolovanih mreža. Svaku takvu izolovanu mrežu tada nazivamo podmrežom.

Ako primenimo ovaj postupak na međusobno povezani sistem sa slike 4.17, добићemo šest ostrva ili podmreže.

Iz gornjeg opisa je jasno da će organizacija (kompanija ili akademska ustanova) sa više Ethernet segmentima i linkovima od čvora do čvora imati više podmreža, tako da će svi uređaji u datoj podmreži imati istu adresu podmreže. U principu, različite podmreže mogu da imaju sasvim različite adrese podmreže. U praksi, međutim, njihove adrese podmreže imaju dosta zajedničkog. Da biste shvatili zašto, recimo nešto o adresiranju na globalnom Internetu.

Strategija dodeljivanja adresa na Internetu poznata je kao besklasno međudo-mensko rutiranje CIDR (*Classless Interdomain Routing* - akronim se izgovara „sai-der“) [RFC 1519]. CIDR uopštava pojam adresiranja podmreže. Kao kod adresiranja podmreže, 32-bitna IP adresa se deli na dva dela opet u decimalnom obliku sa tačkama  $a.b.c.d/x$ , gde  $x$  označava broj bitova u prvom delu adrese.

Najznačajnijih  $x$  bitova u adresi oblika  $a.b.c.d/x$ , čini mrežni deo IP adrese i često se naziva prefiksom (ili *mrežnim prefiksom*) adrese. Organizaciji se obično dodeljuje blok susednih adresa, tj. raspon adresa sa zajedničkim prefiksom (procitajte opis: Principi u praksi). U ovom slučaju, IP adrese uređaja u organizaciji imajuće



**Slika 4.17** ♦ Tri rute koji međusobno povezuju 6 podmreža.

zajednički prefiks. Kada u odeljku 4.6 budemo obrađivali Internetov protokol mtrira-nja BGP, videćemo da ruteri izvan mreže ove organizacije uzimaju u obzir samo tih vodećih  $x$  bitova prefiksa. To jest, kada ruter van organizacije prosledjuje datagram čija je odredišna adresa u organizaciji, dovoljno je da uzme u obzir samo vodećih  $x$  bitova adrese. To značajno smanjuje veličinu tabele prosleđivanja u ruterima, pošto će *satno jedna* stavka u obliku  $a.b.c.d/x$  biti dovoljna za prosleđivanje paketa u *svako odredište* unutar organizacije.

Preostalih  $32 - x$  bitova adrese može se posmatrati kao razlikovanje uređaja *unutar* organizacije, koji svi imaju isti mrežni prefiks. Ti će bitovi biti u igri kada se paketi prosleđuju u ruterima *unutar* organizacije. Ove bitove nižeg reda organizacija bi mogla (ali ne mora) dalje da deli postupkom poznatim kao podmreže. Na primer, prepostavite da prvih 21 bitova CIDR adrese  $a.b.c.d/11$  prvih 21 bitova znače mrežni prefiks organizacije i zajednički su za IP adrese svih računara te organizacije. Preostalih 11 bitova zatim određuju konkretnе računare u organizaciji. Unutrašnja struktura organizacije bi mogla da bude takva da se tih 11 desnih bitova upotrebni za

podmreže unutar organizacije, kao što je gore opisano. Na primer, prefiks  $a.b.c.d/24$  bi mogao da se odnosi na konkretnu podmrežu unutar organizacije.

Dok nije prihvaćen CIDR, mrežni deo IP adrese je morao da bude dugačak 8, 16 ili 24 bitova, po adresnoj šemi poznatoj kao adresiranje **zasnovano na klasama**, jer su se podmreže čije su adrese imale 8, 16 ili 24 bitova bile poznate kao mreže klase A, B, odnosno C. Zahtev da deo IP adrese koji se odnosi na mrežu bude tačno jedan, dva ili tri bajta pokazao se problematičnim za podršku sve većeg broja organizacija sa malim i srednjim podmrežama. Podmreža klase C (/24) mogla bi da sadrži samo  $2^E - 2 = 254$  računara (dve od  $2^A = 256$  adresa rezervišu se za posebnu upotrebu) - što nije dovoljno za mnoge organizacije. Međutim, podmreža klase B (/16), koja može da podrži do 65.634 računara je prevelika. U adresiranju zasnovanom na klasama organizacija koja je imala 2.000 računara obično je dobijala adresu podmreža klase B (/16). To je dovelo do naglog trošenja adresnog prostora klase B i loše iskorišće-nosti dodeljenog adresnog prostora. Na primer, organizaciji koja bi za svojih 2.000 računara koristila adresu klase B bio je dodeljen adresni prostor za 65.534 interfejsa - pa je ostalo neiskorišćeno 63.000 adresa koje su se mogle dodeliti drugim organizacijama.

Ali, ne bi bilo u redu da ne pomenemo jednu drugu vrstu IP adrese, naime, IP adresu za difuzno emitovanje 255.255.255.255. Kada računar emituje datagram sa adresom odredišta 255.255.255.255, poruka se isporučuje svim računarima u istoj podmreži. Ruteri opcionalno prosleđuju tu poruku u susednim [p podmrežama (mada to obično ne radi). U poglavljiju 5 ćemo videti primer kako se koristi difuzno emitovani IP paket kada budemo opisivali protokol DHCP.

Pošto smo detaljno proučili IP adresiranje, moramo da saznamo kako računar ili podmreža uopšte dobija IP adresu (ili adrese). Prvo ćemo pogledati kako organizacija dobija blok adresa za svoje uređaje, a zatim ćemo videti kako uređaj (na primer računar) dobija jednu od adresa iz bloka adresa koje su dobijene za organizaciju.

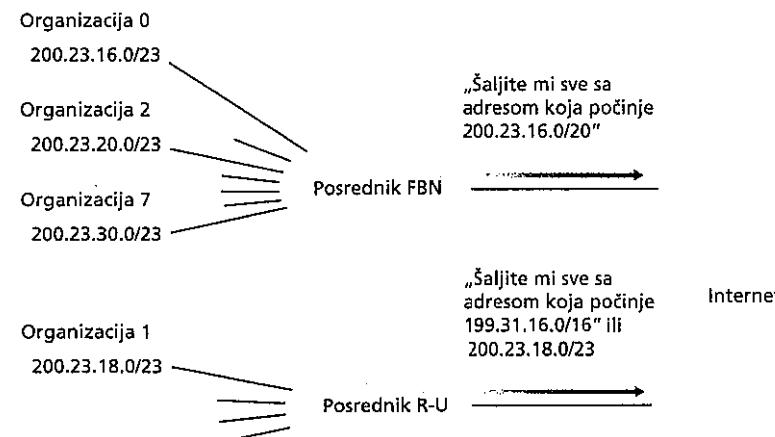
#### Dobijanje bloka mrežnih adresa

Da bi dobio blok IP adresa koje će se koristiti u podmreži organizacije, administrator mreže može prvo da pozove svog posrednika Internet usluga koji će mu obezbe-diti adrese iz jednog većeg bloka adresa koje su ranije dodeljene posredniku Internet usluga. Na primer, posredniku Internet usluga mogao je biti dodeljen blok adresa 200.23.16.0/20. Zatim bi on mogao da podeli svoj blok adresa na osam manjih blokova adresa jednakne veličine i pred po jedan od tih blokova svakoj od najviše osam organizacija koje on podržava kao što je prikazano dole u tekstu. (Podvukli smo mrežni deo ovih adresa.)

|                 |                |                                            |
|-----------------|----------------|--------------------------------------------|
| Posrednikovblok | 200.23.16.0/20 | <u>11001000 00010111 00010000 00000000</u> |
| Organizacija 0  | 200.23.16.0/23 | <u>11001000 00010111 00010000 00000000</u> |
| Organizacija 1  | 200.23.18.0/23 | <u>11001000 00010111 00010010 00000000</u> |
| Organizacija 2  | 200.23.20.0/23 | <u>11001000 00010111 00010100 00000000</u> |
| Organizacija 7  | 200.23.30.0/23 | <u>11001000 00010111 00011110 00000000</u> |

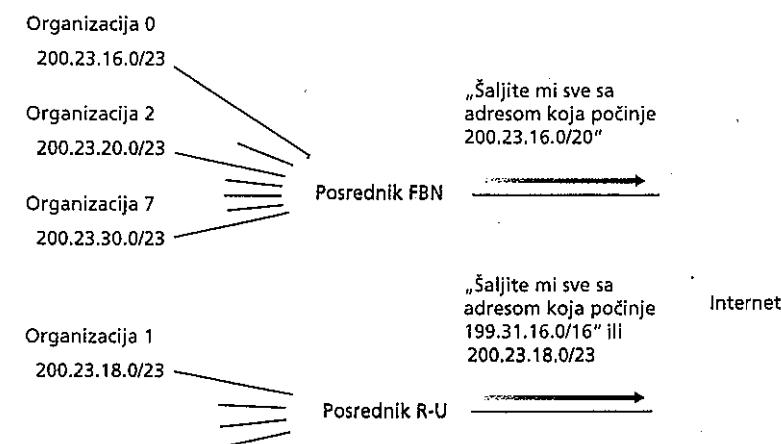
### PRINCIPI U PRAKSI

Ovaj primer posrednika za Internet usluge koji povezuje osam organizacija sa Internetom lepo ilustruje kako pažljivo dodeljene CIDR adrese olakšavaju rutiranje. Pretpostavimo, kao što je prikazano na slici 4.1.8, da posrednik (koga ćemo nazvati FBN) objavljuje spoljnom svetu da mu treba poslati sve datagrame čijih prvih 20 adresnih bitova glasi 200.23.16.0/20. Ostatak sveta ne mora da zna da se unutar bloka adresa 200.23.16.0/20 nalazi osam drugih organizacija od kojih svaka ima vlastitu podmrežu. Ova mogućnost da se koristi jedan mrežni prefiks za oglašavanje više mreža često se naziva **agregacija adresa** (a takođe i **agregacija ruta** ili **sumiranje ruta**).



Slika 4.18 ♦ Hjерархијско адресирање и агрегација рута

Agregacija adresa izuzetno dobro funkcioniše kada se blokovi adresa dodeljuju posrednicima za Internet usluge, a zatim od posrednika klijentskim organizacijama. Ali, šta se događa kada se adrese ne dodeljuju na takav hijerarhijski način? Šta će se, na primer, dogoditi ako Organizacija 1 poslane nezadovoljna lošim uslugama posrednika FBN i odluči da se prebaci kod drugog posrednika, na primer R-U? Kako je prikazano na slici 4.18, posrednik R-U je vlasnik bloka adresa 199.31.0.0/16, ali IP adrese Organizacije 1 su nažalost izvan tog bloka adresa. Šta bi sada trebalo uraditi? Naravno, Organizacija 1 bi mogla da prenumiši sve svoje rutere i računare na adrese unutar bloka adresa posrednika R-U, Ali, to je skupo rešenje, i Organizacija 1 bi već zbog toga ubuduće mogla da pređe od posrednika R-U kod nekog drugog. Obično se prihvata rešenje da Organizacija 1 zadrži svoje IP adrese u domenu 200.23.18.0/23. U tom slučaju, kao što je prikazano na slici 4.19, posrednik FBN i dalje objavljuje blok adresa 200.23.16.0/20, a posrednik R-U blok 199.31.0.0/16. Međutim, posrednik R-U sada objavljuje *još* / blok adresa za Organizaciju 1, 200.23.1.8.0/23. Kada ostali ruteri u većem Internetu naiđu na blokove adresa 200.23.16.0/20 (posrednika FBN) i 200.23.1.8.0/23 (posrednika, R-U), a žele da rufiraju prema nekoj adresi u bloku 200.23.1.8.0/23 oni će upolrebiri pravilo najdužeg prefiksa [videti odeljak 4.2.2] i ruti-rati prema posredniku R-U pošto on objavljuje duži (precizniji) adresni prefiks za traženu adresu odredišta.



Slika 4.19 ♦ Posrednik R-U има precizniju adresу за Organizaciju 1.

Dobijanje skupa adresa od posrednika internet usluga je jedan način da se dobije blok adresa, **ali** to nije jedini način. Jasno je, mora da postoji neki način na koji sam posrednik Internet usluga dobija blok adresa. Da li postoji globalno ovlašćeno telo sa najvišom odgovornošću za upravljanje prostorom IP adresa i dodeljivanje blokova adresa posrednicima Internet usluga i ostalim organizacijama? Zaista postoji! IP adresama upravlja Internet korporacija za dodeljena imena i brojeve (*Interne! Corporation for AssignedNames and Numbers*, ICANN) [ICANN 2004] na osnovu smemicu postavljenih u RFC 2050. Uloga neprofitne organizacije ICANN [NTIA 1998] nije samo dodeljivanje IP adresa, već takođe i upravljanje osnovnim DNS serverima. Ona takođe dodeljuje imena domena i razrešava sporove o imenima domena. ICANN dodeljuje adrese regionalnim Internet registrima (na primer, ARIN, RIPE, APNIC i LACNIC koji zajedno čine organizaciju za podršku adresa ICANN ASO-ICANN 2004) koji zatim dodeljuju adrese i upravljaju svaki u svom regionu.

#### Pribavljanje adrese računara

Kada organizacija pribavi blok adresa od svog posrednika Internet usluga, ona može da dodeli pojedinačne IP adrese računarskim i ruterskim interfejsima u svojoj organizaciji. Za ruterske adrese interfejsa, sistemski administrator ručno konfiguriše IP adresu u ruter (često to obavlja daljinski pomoću alatki za upravljanje mrežom). Računar može da dobije IP adresu na jedan od sledeća dva načina:

- ◆ *Ručno konfigurisanje.* Sistemski administrator ručno konfiguriše IP adresu u računaru (obično u datoteku).
- ◆ *Protokol za dinamičko konfigurisanje računara (DHCP, Dynamic Host Configuration Protocol)* [RFC 2131]. DHCP omogućava računaru da pribavi (automatski dodeljenu) IP adresu, kao i da sazna dodatne informacije kao što je adresa njegovog ruter za prvi skok i adresa njegovog DNS servera.

DHCP je u stanju da automatizuje mrežne aspekte povezivanja računara i on se često naziva plug-and-play protokolom. Zbog tog svojstva on je *veoma* privlačan mrežnim administratorima koji **bi** inače morali te zadatke da obave ručno! DHCP takođe ima široku primenu u kućnim mrežama za pristup Internetu i u bežičnim LAN-ovima u kojima računari često pristupaju mreži i napuštaju je.

Mrežni administrator može da konfiguriše DHCP tako da dati računar dobije postojanu IP adresu tj. da mu se svaki put kada pristupi mreži uvek dodeli ta ista IP adresa. Ali, mnoge organizacije i kućni posrednici Internet usluga nemaju dovoljno IP adresa za sve svoje računare. U takvom slučaju, DHCP se često koristi da bi se svakom računaru koji se povezuje dodelila privremena IP adresa. Na primer, razmotrite kućnog posrednika Internet usluga koji ima 2.000 korisnika, ali ih nikad nije istovremeno priključeno više od 400. Da bi zadovoljio svojih 2.000 korisnika, posredniku Internet usluga nije potreban blok od 2.000 adresa. Ako umesto toga,

upotrebi DHCP server za dinamičko dodeljivanje adresa, biće mu dovoljan blok od 512 adresa (na primer, blok u obliku 200.23.30.0/23). Kako računari dolaze i odlaze, DHCP server mora da ažurira svoju listu dostupnih IP adresa. Kad god se računar priključi, DHCP server mu dodeljuje proizvoljnu adresu iz svog raspoloživog skupa (iz tzv. pula) dostupnih adresa; kada se računar isključi, adresa se vraća u pul.

Drugi značajan razlog što DHCP ima tako široku primenu je pojava mobilnog računarstva. Uzmite, na primer, studenta koji nosi laptop iz spavaonice u biblioteku pa u učionicu. Vrlo je verovatno da će na svakom mestu student da se priključi na drugu mrežu pa će mu na svakom od tih mesta biti potrebna nova IP adresa. DHCP je idealan za takvu situaciju, pošto mnogi korisnici dolaze i odlaze, a adrese su potrebne samo za ograničeno vreme.

Protokol DHCP ustvari premošćuje granicu između mrežnog sloja i sloja veze u Internetovoj familiji protokola sa pet slojeva. Zato ćemo detaljan opis implementacije usluge DHCP odložiti do poglavlja 5 u kojem se obrađuje sloj veze.

#### Prevodioci mrežnih adresa (NAT-ovi)

S obzirom na ono što smo gore rekli, sada smo svesni da svaki uređaj sposoban za IP mora da ima IP adresu. Sa naglim širenjem takozvanih SOHO (*srna!! office, home office*) mreža, to bi naizgled značilo da bi kad god neki SOHO hoće da instalira LAN i međusobno poveže više mašina, posrednik za Internet usluge morao da mu dodeli raspon adresa za sve te mašine. Ako bi mreža kasnije porasla (na primer, kada deca kod kuće ne bi imala samo vlastite računare, već i PDA računare, telefone sposobljene za IP i umrežene Gamebov konzole) morao bi im se dodeliti veći broj adresa. A šta ako je posrednik u međuvremenu već drugima dodelio neprekidne delove trenutnog adresnog prostora SOHO mreže? I šta uopšte prosečan kućni korisnik hoće (ili treba) da zna o upravljanju IP adresama? Srećom, postoji jednostavniji način za dodeljivanje adresa, prevodenje mrežnih adresa (*Network Address Translation, NAT*) i on se u takvim situacijama sve više koristi [RFC 2663; RFC 3022].

Na slici 4.20 prikazanje NAT ruter. Kućni NAT ruter ima interfejs koji spada u kućnu mrežu na desnoj strani slike 4.20. Adresiranje unutar kućne mreže isto je kao što smo videli ranije - sva četiri interfejsa kućne mreže imaju istu mrežnu adresu, 10.0.0/24. Adresni prostor 10.0.0/8 je jedan od tri dela prostora IP adresa rezervi-sanih u [RFC 1918] za privatnu mrežu ili carstvo sa privatnim adresama, kao što je kućna mreža sa slike 4.20. *Carstvo sa privatnim adresama je* mreža čije adrese nešto znače samo uređajima unutar te mreže. Da biste videli zastoje to važno, razmotrite činjenicu da postoji na stotine hiljada kućnih mreža od kojih mnoge koriste isti adresni prostor, 10.0.0/24. Uredaji unutar date kućne mreže mogu jedan drugome da šalju pakete koristeći adresiranje 10.0.0/24. Međutim, jasno je da paketi koji se šalju *preko* granice kućne mreže u veći globalni Internet ne mogu da koriste te adrese (bilo kao adresu izvora ili kao adresu odredišta) pošto imaju na stotine hiljada mreža koje koriste taj blok adresa. To znači da adrese 10.0.0/24 jedino u datoj kućnoj mreži nešto znače.

Ali ako privatne adrese imaju značenje samo unutar date mreže, kako se sprovodi adresiranje ako se paketi šalju u globalni Internet ili se primaju iz Interneta gde su adrese obavezno jedinstvene? Odgovor se nalazi u NAT prevodenju,

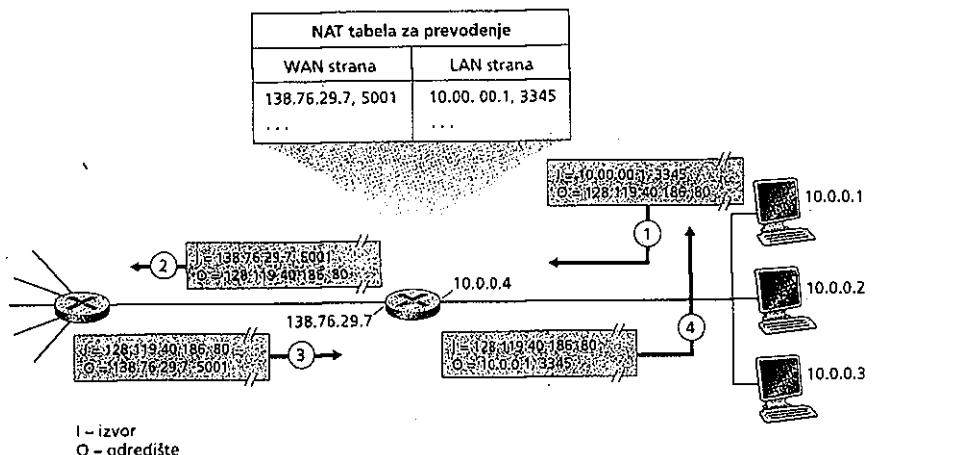
NAT ruter za spoljni svet ne *izgleda* kao ruter. Umesto toga, NAT ruter se prema spoljnom svetu ponaša kao *jedan* uređaj sa *jednom* IP adresom. Na slici 4.20 sav saobraćaj koji polazi od kućnog rutera prema većem Internetu ima izvornu IP adresu 138.76.29.7 i sav saobraćaj koji dolazi do kuće mora imati adresu odredišta 138.76.29.7. U suštini, NAT ruter krije detalje kućne mreže od spoljnog sveta. (Možda se pitate odakle računari u kućnoj mreži dobijaju adrese i odakle ruter dobija tu jednu IP adresu. Često je odgovor isti - od DHCP-a! Ruter dobija svoju adresu od DHCP servera kod posrednika Internet usluga, a isti kućni ruter izvršava DHCP server koji obezbeđuje adrese za računare u adresnom prostoru kućne mreže pod kontrolom NAT DHCP ruter.)

Sada se možda pitate kako taj ruter zna kojem internom računaru treba da prosledi dobijeni datagram ako svi datagrami koji do NAT rutera stižu iz regionalne mreže imaju istu IP adresu odredišta (konkretno, adresu interfejsa NAT rutera prema regionalnoj mreži)? Trik je u korišćenju NAT **tabele prevodenja** u NAT ruteru i u tome da se u stavke tabele osim IP adresa dodaju i brojevi portova.

Razmotrite primer prikazan na slici 4.20. Uzmite da korisnik sedi za računarom 10.0.0.1 u kućnoj mreži i zatraži veb stranicu od nekog veb servera (port 80) sa IP adresom 128.119.40.186. Računar 10.0.0.1 dodeljuje (proizvoljan) broj izvornog porta 3345 i šalje datagram u LAN. NAT ruter prima datagram, pravi novi broj izvornog porta 5001 za taj datagram, zamenjuje izvornu IP adresu svojom IP adresom za WAN 138.76.29.7 i zamenjuje prvo bitni broj izvornog porta 3345 sa 5001. Kada pravi novi broj izvornog porta NAT ruter može da izabere bilo koji broj izvornog porta koji se trenutno ne nalazi u NAT tabeli prevodenja. (Obratite pažnju na to da pošto polje za broj porta ima 16 bitova, protokol NAT može da podrži preko 60.000 istovremenih konekcija sa jednom samom IP adresom tog rutera za WAN!) NAT u ruteru takođe dodaje stavku u svoju NAT tabelu prevodenja. Veb server, u blaženom neznanju daje pristigli datagram koji sadrži HTTP zahtev prepravljen u NAT ruteru, odgovara datagramom čija je odredišna adresa IP adresa NAT rutera, a odredišni port mu je 5001. Kada taj datagram stigne u NAT ruter, ovaj pomoću odredišne IP adrese i broja porta u svojoj NAT tabeli prevodenja pronalazi odgovarajuću IP adresu (10.0.0.1) i odredišni broj porta (3345) za čitač u kućnoj mreži. Ruter tada u datagramu prepravlja odredišnu adresu i odredišni broj porta i posreduje datagram u kućnu mrežu.

NAT uživa Široku primenu poslednjih godina, ali trebalo bi pomenuti da mu se mnogi čistunci u IETF zajednici žučno protive. Prvo, oni prigovaraju da su brojevi portova namenjeni adresiranju procesa, a ne adresiranju računara. (Ovo kršenje zaista može dovesti do problema za servere koji se izvršavaju u kućnoj mreži pošto, kako smo videli u poglavljiju 2, serverski procesi čekaju dolazne pozive na dobro poznatim brojevima portova.) Drugo, oni prigovaraju da ruteri treba da obraduju pakete samo do sloja 3. Treće, oni prigovaraju da protokol NAT krši takozvani argument od kraja do kraja; tj. da bi računari trebalo direktno da komuniciraju, a ne da usputni Čvorovi menjaju IP adrese i brojeve portova. I četvrtro, oni prigovaraju da bi za rešenje nedostatka IP adresa trebalo upotrebiti IPv6 (videti odeljak 4.4.4), a ne lakomisleno krptiti problem pomoću štapa i kanapa kao što je NAT. Ali, sviđao vam se ili ne, NAT je postao značajna komponenta Interneta.

Još jedan veliki problem sa NAT-om je da on ometa P2P aplikacije, uključujući P2P aplikacije za deljenje datoteka i P2P aplikacije za govor preko IP-a. Verovatno se sećate iz poglavљa 2 da u P2P aplikaciji svaki ravnopravni učesnik A mora biti u stanju da inicira TCP konekciju sa bilo kojim drugim ravnopravnim učesnikom B. Suština problema je u tome da kad je učesnik B iza NAT-a on ne može da preuzme ulogu servera i prihvata TCP konekcije (osim ako je NAT posebno konfigurisan za P2P aplikacije). Kao što ćemo videti u domaćim zadacima, ovaj NAT problem može se izbeći ako učesnik A prvo kontaktira sa učesnikom B preko posrednog učesnika C koji nije iza NAT-a i sa kojim je B uspostavio aktivnu TCP konekciju. Učesnik A može tada da zatraži da B preko učesnika C inicira TCP konekciju direktno sa učesnikom A. Čim se uspostavi direktna P2P TCP konekcija između učesnika A i B, oni mogu da razmenjuju poruke ili datoteke. Ovaj trik, koji se naziva **preokret konekcije** zaista se koristi u mnogim P2P aplikacijama. Ali ako su i A i B svako iza



Slika 4.20 ♦ Prevodenje mrežnih adresa

svog NAT-a, tada je nemoguće uspostaviti TCP konekciju između ta dva učesnika, osim ako se NAT posebno ne konfiguriše za konkretnu aplikaciju.

Naš opis NAT-a je ovde morao da bude kratak. NAT ima mnoge druge značajne aspekte kao što su statički i dinamički NAT i uticaj NAT-a na protokole viših slojeva i na bezbednost. Više detalja kao i argumente za NAT i protiv njega naći ćete u knjigama [Cisco NAT 2004; Pliifer 2000].

#### 4.4.3 ICMP: Internet Control Message Protocol

Sigurno se sećate da mrežni sloj Interneta ima tri glavne komponente: IP protokol o kojem smo govorili u prethodnom odeljku; internetski protokoli rutiranja (uključujući RIP, OSPF i BGP) o kojima se govoru u odeljku 4.6 i ICMP, što je tema ovog odeljka.

ICMP, definisan u RFC-u 792, koriste računari i ruteri za međusobno prenošenje informacija o mrežnom sloju. Najtipičnija upotreba ICMP-a je izveštavanje o greškama. Na primer, kada se izvršava Telnet, FTP ili HTTP sesija, možda ste naišli na poruku o grešci kao što je „Destination network unreachable“ (odredišna mreža nedostupna). Ta poruka je potekla iz ICMP-a. U nekom trenutku, neki IP ruter nije mogao da pronađe putanju prema računaru navedenom u vašoj Telnet, FTP ili HTTP aplikaciji. Taj ruter je napravio i poslao vašem računaru ICMP poruku tipa 3 sa obaveštenjem o grešci.

ICMP se često smatra delom IP-a, ali u arhitekturi se nalazi neposredno iznad IP-a pošto se ICMP poruke prenose unutar IP paketa. To jest, ICMP poruke se prenose kao korisni podaci IP-a, isto onako kao što se prenose TCP ili UDP segmenti. Slično tome, kada računar primi IP paket u kome se kao protokol gornjeg sloja navodi ICMP, on demultiplexira paket ICMP-u, isto kao što bi ga demultiplexirao TCP-u ili UDP-u.

ICMP poruke imaju polje tipa i polje šifre a sadrže takođe zaglavljive i prvi osam bajtova IP datagrama koji je doveo do nastanka ICMP poruke (tako pošiljalac može da utvrdi koji je paket doveo do greške). Nekoliko ICMP poruka dato je na slici 4.21. Obratite pažnju na to da se ICMP poruke ne koriste samo za signalisanje grešaka.

Veoma poznati program ping šalje navedenom računaru ICMP poruku tipa 8 sa šifrom 0. Kada odredišni računar vidi zahtev za ehom, on vraća ICMP echo odgovor tipa 0 sa šifrom 0. Većina implementacija TCP/IP-a podržava ping server direktno u operativnom sistemu; tj. ovaj server nije proces. Izvomi oblik klijentskog programa ping možete naći u jedanaestom poglavljiju knjige [Stevens 1990]. Obratite pažnju na to da klijentski program mora biti u stanju da naredi operativnom sistemu da napravi ICMP poruku tipa 8 sa šifrom 0.

Još jedna zanimljiva ICMP poruka je poruka za suzbijanje izvora. Ta poruka se retko koristi u praksi. Prvobitna zamisao je bila da se tako kontroliše zagrušenje - da zagrušeni ruter pošalje nekom računaru ICMP poruku za suzbijanje izvora i primora taj računar da smanji brzinu slanja. Videli smo u poglavljju 3 da TCP ima vlastiti mehanizam za kontrolu zagrušenja koji funkcioniše u transportnom sloju bez korišćenja povratnih informacija mrežnog sloja kakva bi bila ICMP poruka za suzbijanje izvora.

U poglaviju 1 predstavili smo program Traceroute koji omogućava traširanje rute od jednog računara do bilo kog računara na svetu. Zanimljivo je da je Traceroute implementiran pomoću ICMP poruka. Da bi se utvrdila imena i adrese ruteru između izvora i odredišta, Traceroute u izvoru šalje na odredište niz običnih IP datagrama. Svaki od ovih datagrama nosi jedan UDP segment sa malo verovatnim brojem UDP porta. Prvi od ovih datagrama ima TTL = 1, drugi 2, treći 3 itd. Izvor takođe pokreće tajmre za svaki datagram. Kada »-ti datagram stigne do n-tog ruteru, n-ti ruter primeće da je TTL u datagramu upravo istekao. Prema pravilima protokola IP, ruter odbacuje datagram i šalje izvoru ICMP poruku upozorenja (tip 11 šifra 0). Ta poruka upozorenja sadrži ime ruteru i njegovu IP adresu. Kada se ta ICMP poruka vrati do izvora, izvor uzima vreme povratnog puta iz tajmera, aime i IP adresu n-tog ruteru iz ICMP poruke.

| ICMP tip | Šifra | Opis                                |
|----------|-------|-------------------------------------|
| 0        | 0     | echo odgovor (za ping)              |
| 3        | 0     | odredišna mreža nedostupna          |
| 3        | 1     | odredišni računar nedostupan        |
| 3        | 2     | odredišni protokol nedostupan       |
| 3        | 3     | odredišni port nedostupan           |
| 3        | 6     | odredišna mreža nepoznata           |
| 3        | 7     | odredišni računar nepoznat          |
| 4        | 0     | suzbiti izvor (kontrola zagrušenja) |
| 8        | 0     | echo zahtev                         |
| 9        | 0     | upozorenje ruteru                   |
| 10       | 0     | otkriće ruteru                      |
| 11       | 0     | TTL isteklo                         |
| 12       | 0     | loše IP zaglavlje                   |

**Slika 4.21** ♦ Tipovi ICMP poruka

Kako Traceroute na izvoru zna kada da prestane sa slanjem UDP segmenata? Sigurno se sećate da za svaki poslati datagram izvor povećava polje TTL. Jedan od datograma će kad stići do odredišnog računara. Pošto datagram sadrži UDP segment sa malo verovatnim brojem porta, odredišni računar će vratiti ICMP poruku da je odredišni port nedostupan (tip 3 šifra 3). Kada izvorni računar primi tu konkretnu poruku, on zna da ne treba više da šalje istraživačke pakete. (Standardni program Traceroute u stvari šalje grupe od po tri paketa sa istim TTL-om; njegov izveštaj tako sadrži po tri rezultata za svaki TTL.)

Na ovaj način, izvorni računar saznaće broj i identitet rutera koji se nalaze između njega i odredišnog računara kao i trajanje povratnog puta među njima. Obratite pažnju na to da klijentski program mora biti u stanju da naredi operativnom sistemu da napravi UDP datagrame sa određenim vrednostima TTL, a takođe da bude u stanju da primi obaveštenje operativnog sistema o prispeću ICMP poruka. Pošto sada znate kako Traceroute funkcioniše, možda ćete poželeti da se još malo igrate sa njim.

#### 4.4.4 IPv6

Početkom 1990-ih godina, IETF (*Internet Engineering Task Force*) je pokrenuo razvijanje naslednika za protokol IPv4. Osnovnu motivaciju predstavljalo je to što se uvidelo da 32-bitni adresni prostor IP-a počinje da se troši, a da še zapanjujućom brzinom povećava broj novih podmreža i IP Čvorova koji se povezuju sa Internetom (i dobijaju jedinstvene IP adrese). Da bi se zadovoljila ova potreba za velikim adresnim prostorom, razvijen je novi IP protokol, IPv6. Projektanti protokola IPv6 iskoristili su tu priliku da ubace i povećaju i druge elemente na osnovu prikupljenog operativnog iskustva sa protokolom IPv4.

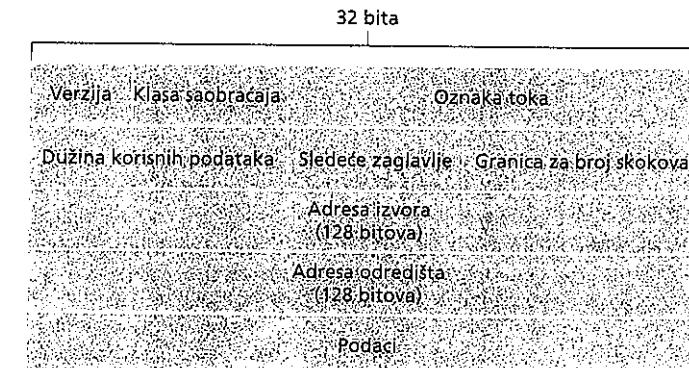
Dugo se raspravljalo o trenutku kada će sve IPv4 adrese da se potroše (i kada nijedna nova podmreža neće moći da se poveže sa Internetom). Na osnovu tadašnjih trendova u dodeljivanju adresa, procena dvojice voda radne grupe IETF Address Lifetime Expectations bila je da će se adrese istrošiti 2008 odnosno 2018 [Solensky 1996]. APN (American Registry for Internet Numbers) je 1996. godine izvestio da su već dodeljene sve IPv4 adrese klase A, 62 procenta adresa klase B i 37 procenata adresa klase C [ARIN 1996]. Mada su ove procene i izveštaji ukazivali na to da se ceo adresni prostor IPv4 neće tako brzo potrošiti, bilo je jasno da će biti potrebno dosta vremena da se nova tehnologija uvede na tako široko područje, pa je pokrenut projekat „Next Generation IP“ (IPng) [Bradner 1996; RFC 1752]. Rezultat tog projekta bila je specifikacija za IP verziju 6 (IPv6) [RFC 2460]. (Često se postavlja pitanje gde je nestao IPv5. Prvo se planiralo daće protokol ST-2 postati IPv5, ali se od ST-2 kasnije odustalo i umesto njega je prihvaćen protokol RSVP koji opisujemo u poglavljju 7.)

Odlični izvori informacija o protokolu IPv6 su matična stranica The IP Next Generation [Hinden 2002] i knjiga na tu temu Huitema [Huitema 1998].

#### Format datagrama IPv6

Format datagrama IPv6 prikazuje na slici 4.22. Najvažnije promene koje donosi IPv6 očigledne su iz formata datagrama:

- ◆ *Proširene mogućnosti adresiranja.* IPv6 produžava IP adresu sa 32 bita na 128 bitova. Tako je obezbedeno da se nikada ne potroše sve IP adrese. Sada svako zrno peska na svetu može da dobije svoju IP-adresu. Osim jednoznačnih i više-značnih IP adresa, IPv6 je uveo i novu vrstu adresa, adresu za jednostruko upućivanje (anycast), koje omogućavaju da se datagram koji se pošalje na takvu adresu isporuči samo jednom ali bilo kojem računaru iz neke grupe. (To svojstvo može, na primer, da se upotrebi da bi se poslao HTTP GET najbližoj od niza pre-slikanih lokacija koje sadrže dati dokument.)
- ◆ *Kompaktnije 40-bajtno zaglavljje.* Kao što ćemo kasnije opisati, niz polja iz protokola IPv4 je izbačeno ili je postalo opcionalno. Tako je dobijeno 40-bajtno zaglavje fiksne dužine koje omogućava bržu obradu IP datagrama. Novo kodiranje opcija obezbeđuje njihovu fleksibilniju obradu.
- ◆ *Označavanje i prioriteti toka.* IPv6 sadrži lukavu definiciju toka. RFC 1752 i RFC 2460 tvrde da to omogućava „označavanje paketa koji pripadaju konkretnom toku za koji pošiljalac zahteva posebno rukovanje kao što su usluge u realnom vremenu ili kvalitet usluge viši od podrazumevanog“. Na primer, audio i video prenosi mogli bi se tretirati kao tok. S druge strane, tradicionalnije aplikacije kao što su prenos datoteka i e-pošta možda se neće tretirati kao tokovi. Saobraćaj koji prenosi korisnik sa visokim prioritetom (na primer, neko ko plaća da za svoj saobraćaj dobije bolju uslugu) takođe je moguće tretirati kao tok. Jasno



Slika 4.22 ◆ Format datagrama IPv6

je međutim da projektanti protokola IPv6 predviđaju eventualnu potrebu da se pravi razlika među različitim tokovima. Čak iako još nije utvrđeno precizno značenje toka. Zaglavje IPv6 takođe sadrži 8-bitno polje za klasu saobraćaja. To polje, kao i polje TOS u protokolu IPv4, može se upotrebiti za davanje prednosti određenim paketima unutar toka ili za давање предности datagramima iz određenih aplikacija (na primer, ICMP paketima) nad datagramima iz drugih aplikacija (na primer, novosti).

Kako smo već napomenuli, poređenjem slike 4.22 sa slikom 4.13 otkriva se daje struktura datagrama IPv6 jednostavnija i kompaktnija. U protokolu IPv6 definisana su sledeća polja:

- ◆ *Verzija.* Ovo polje od četiri bita sadrži broj IP verzije. Jasno je da IPv6 u ovom polju sadrži vrednost 6. Obratite pažnju na to da ako u ovo polje stavite vrednost 4, nećete dobiti pravilan IPv4 datagram. (Kada bi to bilo tako, život bi bio mnogo jednostavniji - pročitajte u nastavku opis prelaska sa protokola IPv4 na IPv6.)
- ◆ *Klasa saobraćaja.* Svrha ovog 8-bitnog polja slična je polju TOS koje smo videli u protokolu IPv4.
- ◆ *Oznaka toka.* Kao što smo već opisali, ovo 20-bitno polje se koristi kao identifikacija toka datagrama.
- ◆ *Dužina korisnih podataka.* Ova 16-bitna vrednost koristi se kao ceo broj bez predznaka kojim se određuje broj bajtova u IPv6 datagramu iz 40-bajtnog zaglavљa fiksne dužine.
- ◆ *Sledeće zaglavље.* Ovo polje ukazuje na protokol (na primer, TCP ili UDP) kome će se isporučiti sadržaj (polje podataka) ovog datagrama. U ovom polju koriste se iste vrednosti koje se koriste u polju protokola u zaglavljvu IPv4.
- ◆ *Granica za broj skokova.* Sadržaj ovog polja umanjuje se za jedan u svakom ruteru koji prosledi datagram. Ako granica za broj skokova postane jednaka nuli, datagram se odbacuje.
- ◆ *Adrese izvora i odredišta.* Različiti formati 128-bitne adrese protokola IPv6 opisani su u RFC 2373.
- ◆ *Podaci.* Ovo je deo sa korisnim podacima u IPv6 datagramu. Kada datagram stigne na svoje odredište, korisni podaci će se izdvojiti iz IP datagrama i predati protokolu navedenom u polju za sledeće zaglavљe.

U prethodnom opisu navedene su svrhe polja iz datograma IPv6. Ako uporedimo format datograma IPv6 na slici 4.22 sa fonnatom datograma IPv4 koji smo videli ranije na slici 4.13, primetićemo da nekoliko polja koja su postojala u datagramu IPv4 više ne postoje u datagramu IPv6:

◆ *Fragmentacija/Ponovno sastavljanje.* IPv6 ne dozvoljava fragmentiranje i ponovno sastavljanje u usputnim ruterima; te operacije mogu se vršiti samo na izvoru i na odredištu. Ruter koji primi IPv6 datagram koji je previše velik da bi ga izlazni link prosledio, jednostavno će odbaciti datagram i vratiti pošiljaocu ICMP poruku „Packet Too Big“ (videti kasnije). Pošiljalac tada može ponovo da pošalje podatke u manjem IP datagramu. Fragmentiranje i ponovno sastavljanje troše mnogo vremena; oduzimanje tih funkcija od ruta i njihovo strogo zadržavanje u krajnjim sistemima znatno ubrzava IP prosledivanje unutar mreže.

- ◆ *Kontrolni zbir zaglavљa.* Pošto kontrolne zbirove prave i kontrolišu protokoli transportnog sloja (na primer, TCP i UDP) i sloja veze podataka (na primer, Ethernet), projektanti protokola IP verovatno su smatrali daje ta funkcija već dovoljno redundantna u mrežnom sloju i da se može ukloniti. I ovde je brza obrada TP paketa bila glavna briga. Možda se sećate iz opisa protokola IPv4 u odeljku 4.4.1 da kontrolni zbir zaglavљa IPv4 mora ponovo da se izračunava na svakom ruteru zato što IPv4 zaglavljče sadrži polje TTL (slično polju za granicu broja skokova u protokolu IPv6). I ta operacija je previše opterećivala IPv4, isto kao fragmentacija i ponovno sastavljanje.
- ◆ *Opcije.* Polje opcija više ne postoji u standardnom IP zaglavljvu. Međutim, opcije nisu uklonjene. Umesto toga, polje opcija postalo je jedno od mogućih sledećih zaglavљa na koje se pokazuje iz IPv6 zaglavljva. To jest, kao što zaglavljva protokola TCP ili UDP mogu da budu sledeća zaglavljva IP paketa to može da bude i polje opcija. Uklanjanjem polja opcija iz zaglavljva, dobilo se 40-bajtno IP zaglavljve fiksne dužine.

Možda se sećate iz našeg opisa u odeljku 4.4.3 da se protokol ICMP koristi među IP čvorovima za izveštavanje o greškama i za slanje ograničenih informacija (na primer, za echo odgovor na poruku ping) nekom krajnjem sistemu. Za IPv6 je u RFC 2463 definisana nova verzija ICMP-a. Osim što su postojeće definicije ICMP vrsta i šifara preuređene, ICMPv6 ima i nove vrste i šifre potrebne za nove funkcije protokola IPv6. Tu spada i nova vrsta „Packet Too Big“ i nova šifra greške „unrecognized IPv6 options“ (nepoznate IPv6 opcije). Osim toga, ICMPv6 obuhvata i funkcije protokola IGMP (Internet Group Management Protocol) koji ćemo proučiti u odeljku 4.7. IGMP, koji se koristi kada računar pristupa takozvanim grupama za više značno emitovanje i narušava ih. Prethodno je u protokolu IPv4, IGMP bio protokol odvojen od ICMP-a.

#### Prelazak sa protokola IPv4 **na** IPv6

Sada pošto smo videli tehničke detalje protokola IPv6, razmotrimo jedno veoma praktično pitanje: kako će javni Internet koji je zasnovan na protokolu IPv4 preći na IPv6? Problem je u tome što novi sistemi sposobljeni za upotrebu protokola IPv6 mogu da budu kompatibilni unazad tj. mogu da šalju, rutiraju i primaju IPv4 data-

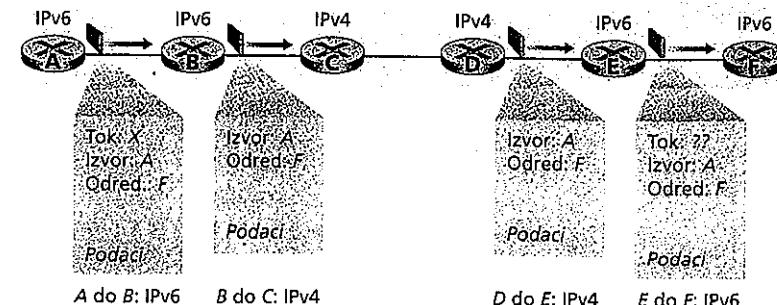
grame, ali već uvedeni sistemi sposobljeni za upotrebu protokola IPv4 ne mogu da rukuju datagramima IPv6. Postoji nekoliko rešenja.

Jedno rešenje bi bilo da se proglaši jedan granični dan - datum i vreme kada bi se sve mašine na Internetu isključile i nadogradile sa protokola IPv4 na IPv6. Poslednja velika promena tehnologije (kada se za uslugu pouzdanog transporta prešlo sa NCP na TCP) dogodila se pre skoro 20 godina. Čak i tada [RFC 801] kadaje Internet bio mali i kadaje njime upravljao mali broj „genijalaca“, znalo se da takav granični dan nije moguć. Granični dan koji bi uključio na stotine miliona mašina i milione mrežnih administratora i korisnika danas je još teže izvodljiv. RFC 2893 opisuje dva pristupa (koji mogu da se koriste odvojeno ili u kombinaciji) za postepeno uvođenje IPv6 računara i rutera u svet kojim vlada IPv4 (naravno, sa dugoročnim ciljem da svi IPv4 čvorovi jednog dana pređu na IPv6).

Verovatno najprostiji način da se uvedu čvorovi sposobni za IPv6 jeste pristup sa **dvostrukim stekom**, u kojem IPv6 čvorovi imaju takođe i kompletну implementaciju protokola IPv4. Takav čvor, koji se u RFC 2893 naziva IPv6/IPv4 sposobljen je za slanje i primanje obe vrste datograma. Kada saraduje sa IPv4 čvorom, Čvor IPv6/IPv4 može da koristi IPv4 datagrame; kada saraduje sa IPv6 čvorom, on može da koristi jezik IPv6. IPv6/IPv4 čvorovi moraju da imaju obe vrste adresa. Oni moraju čak da budu u stanju da odrede da li je drugi čvor sposobljen za IPv6 ili samo za IPv4. Taj problem se može resiti pomoću DNS-a (pročitati poglavje 2) koji može da vrati IPv6 adresu ako je traženi Čvor sposobljen za IPv6, a inače vraća IPv4 adresu. Naravno, ako je Čvor koji je izdao DNS zahtev sposoban samo za IPv4, DNS će vratiti samo IPv4 adresu.

U pristupu sa dvostrukim stekom, ako je bilo pošiljalac bilo primalac sposobljen samo za IPv4, mora se koristiti IPv4 datagram. Zbog toga je moguće da dva čvora sposobljeni za IPv6 na kraju u suštini jedan drugome šalju IPv4 datagrame. Ovo je prikazano na slici 4.23. Uzmimo da je čvor A sposobljen za IPv6 i želi da pošalje IP datagram čvoru F koji je takođe sposobljen za IPv6. Čvorovi A i B mogu da razmenjuju IPv6 datagrame. Međutim, čvor B mora da napravi IPv4 datagram da bi mogao da ga pošalje čvoru C. Naravno, polje podataka iz IPv6 datagrama može da se prekopira u polje podataka IPv4 datagrama i adrese mogu da se preslikaju na odgovarajući način. Međutim, prilikom konverzije iz protokola IPv6 u IPv4 postojeće polja specifična za IPv6 (na primer, identifikator toka) za koja nema odgovarajućih polja u protokolu IPv4. Informacije iz tih polja biće izgubljene. Na taj način, i pored toga što E i F mogu da razmenjuju IPv6 datagrame, IPv4 datagrami koji u čvor E stignu iz čvora D ne sadrže sva polja koja su postojala u prvobitnom IPv6 datagramu koji je poslat iz A.

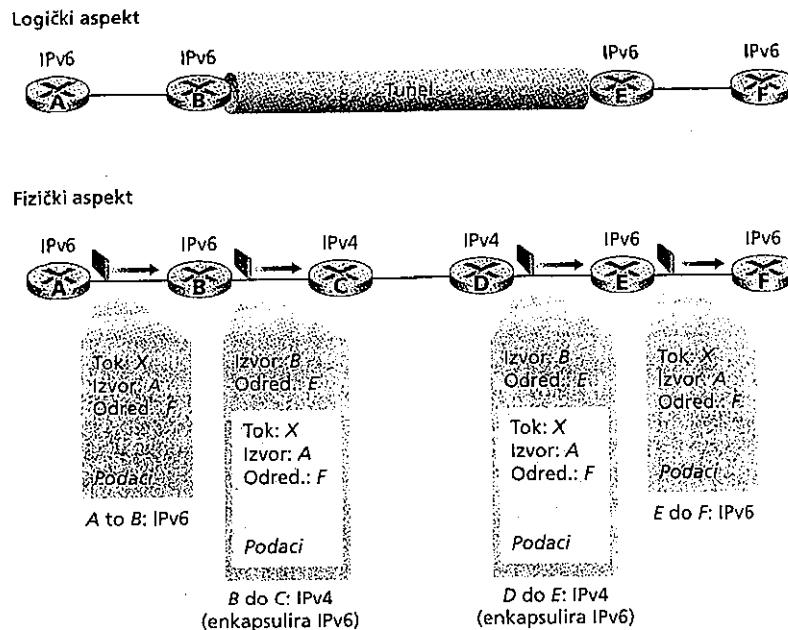
Jedna alternativa rešenju sa dvostrukim stekom koja je takođe opisana u RFC-u 2893, poznata je kao **tunelovanje**. Tunelovanje može da resi gore navedeni problem, jer omogućava, na primer, da E primi IPv6 datagram koji je nastao u čvoru A. Osnovna ideja tunelovanja je sledeća. Uzmimo da dva IPv6 čvora (na primer, B i E sa slike 4.23) žele da saraduju i koriste IPv6 datagrame ali su međusobno povezani preko usputnih IPv4 rutera. Usputne IPv4 rutere između dva IPv6 ruta nazivamo



Slika 4.23 ♦ Pristup sa dva steka

**tunelom**, kao sto je prikazano na slici 4.24. Kada se koristi tunelovanje, IPv6 čvor na otpremnoj strani tunela (na primer, B) uzima *ceo* IPv6 datagram i stavlja ga u polje podataka (korisnih podataka) IPv4 datagrama. Taj IPv4 datagram se zatim adresira IPv6 čvoru na prijemnoj strani tunela (na primer, čvoru E) i šalje prvom čvoru u tunelu (na primer, čvoru C). Usputni IPv4 ruteri u tunelu rutiraju ovaj IPv4 datagram među sobom, kao što bi radili sa bilo kojim datagramom, u blaženom neznanju da taj IPv4 datagram sadrži *ceo* IPv6 datagram. IPv6 čvor na prijemnoj strani tunela konačno prima IPv4 datagram (to i jeste odredište IPv4 datagrama!), utvrđuje da IPv4 datagram sadrži IPv6 datagram, izdvaja IPv6 datagram i zatim ga rutира isto kada ga je primio od direktnog povezanih IPv6 suseda.

Ovaj odeljak zaključujemo napomenom daje prihvatanje protokola IPv6 sporo počelo [Lawton 2001]. Sigurno sesećate da je jedan od glavnih razloga za uvođenje protokola IPv6 bilo to što su se dostupne IPv4 adrese potrošile. U odeljku 4.4.2 smo videli da su unapređenja kao što su CIDR adrese za IPv4, DHCP i NAT doprinela rešavanju ovog problema barem kratkoročno. Moguće je, međutim, da sve šira upotreba uređaja kao što su telefoni sposobljeni za IP i drugi prenosivi uređaji obezbede potreban pritisak za brže uvođenje verzije IPv6. Evropski program partnerstva treće generacije (*Europe's Third Generation Partnership Program*) [3GPP 2004] kao standardnu šemu adresiranja za mobilne multimedije navodi IPv6. Čak i ako IPv6 nije opšteprihvaćen tokom prvih 9 godina njegovog života, potrebno je posmatrati stvari dugoročno. I za prihvatanje današnjeg sistema telefonskih brojeva bilo je potrebno više desetina godina ali je on sada u funkciji već skoro pola veka i nema nagoveštaja da će se ukinuti. Slično tome, biće potrebno vreme da se prihvati IPv6, ali i on će nakon toga dugo biti na snazi. Brajan Karpenter, bivši predsedavajući IAB-a (Internet Architecture Board) [IAB 2004] i autor više RFC-a koji se odnose na IPv6 kaže „uvek sam to smatrao procesom od 15 godina koji počinje 1995“ [Lawton 2001]. Prema Karpenterovim datumima, približavamo se tački na dve trećine puta!



Slika 4.24 ♦ Tunelovanje

Jedna važna lekcija koju možemo naučiti iz iskustva sa protokolom IPv6 jeste da je užasno teško menjati protokole mrežnog sloja. Već od početka 90-tih godina niz novih protokola za mrežni sloj reklamirano je kao sledeća velika revolucija na Internetu, ali je većina njih do danas imala ograničen domet. Tu spadaju IPv6, protokoli za višečaćno upućivanje (odeljak 4.7) i protokoli za rezervisanje resursa (poglavlje 7). Zaista, uvođenje novih protokola u mrežni sloj slično je situaciji u kojoj bismo se našli kada bismo zamjenjivali temelje kuće - to se teško postiže bez rušenja cele građevine ili bar privremenog raseljavanja njenih stanovnika. S druge strane, Internet je doživeo naglo uvođenje više novih protokola u aplikacijskom sloju. Klasični prime-ri, naravno, jesu Web, trenutno slanje poruka i deljenje datoteka među ravnopravnim računarima. Još neki primjeri su protokol audio i video signala u realnom vremenu i distribuirane igre. Uvođenje novih protokola aplikacijskog sloja je kao krećenje kuće - relativno se lako izvodi i ako ste izabrali privlačnu boju, imitiraće vas i neki susedi. Da rezimiramo, u budućnosti se mogu očekivati promene u mrežnom sloju Interneta, ali će te promene verovatno biti mnogo sporije od promena u aplikacijskom sloju.

## 4.5 Algoritmi rutiranja

Do sada smo u ovom poglavlju uglavnom istraživali funkciju prosleđivanja u mrežnom sloju. Naučili smo da kada paket stigne u ruter, ruter indeksira tabelu prosleđivanja i utvrđuje na koji interfejs linka treba usmeriti paket. Takođe smo naučili da algoritmi rutiranja koji se izvršavaju u mrežnim ruterima razmenjuju i izračunavaju informacije potrebne za konfigurisanje tih tabela prosleđivanja. Odnos između algoritma rutiranja i tabele prosleđivanja prikazan je na slici 4.2. Pošto smo donekle istražili prosleđivanje, sada posvećujemo pažnju drugoj važnoj temi ovog poglavlja, kritičnoj funkciji mrežnog sloja - rutiranju. Bez obzira na to da li mrežni sloj pruža uslužu datagrama (kada između datog para izvora i odredišta različiti paketi mogu da idu različitim rutama) ili uslužu virtuelnog kola (kada svaki paket između datog izvora i odredišta idu uvek istom putanjom) mrežni sloj ipak mora da utvrdi putanju za svaki paket. Videćemo daje posao rutiranja da se utvrde dobre putanje (tj. rute) od pošiljalaca do primalaca kroz mrežu ruteru.

Obično se računar vezuje direktno na jedan ruter, takozvani podrazumevani **ruter** za taj računar (takođe se naziva i **ruter prvog skoka** za taj računar). Kad god računar emišuje paket, on se prenosi na njegov podrazumevani ruter. Podrazumevani ruter izvornog računara nazivamo **izvornim ruterom**, a podrazumevani ruter odredišnog računara **odredišnim ruterom**. Problem rutiranja paketa od izvornog do odredišnog računara jasno se svodi na problem mtiranja paketa od izvornog do odredišnog ruteru, a to je tema ovog odeljka.

Svrha algoritma rutiranja je jednostavna: ako je dat skup ruteru sa linkovima koji ih povezuju, algoritam rutiranja pronađi „dobni“ putanju od izvornog do odredišnog ruteru. Obično je dobra putanja ona koja ima najmanju cenu. Videćemo, međutim, da po koncepciji jednostavne i elegantne algoritme na kojima se zasniva rutiranje u današnjim mrežama, u praksi komplikuju realni problemi kao što su pitanja politike (na primer, pravilo kao stoje „ruter\* koji pripada organizaciji Z“) ne bi trebalo da prosleđuje pakete koji potiču iz mreže u vlasništvu organizacije Z“).

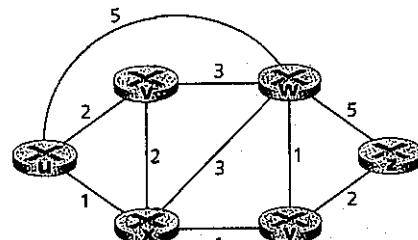
Za formulisanje algoritama rutiranja koristi se graf. Verovatno se sećate da je **graf**  $G = (N, \mathcal{E})$  skup od  $N$  čvorova i kolekcija  $\mathcal{E}$  ivica, gde je svaka ivica par čvorova iz skupa JV. U kontekstu rutiranja u mrežnom sloju, Čvorovi u grafu predstavljaju ruter - tačke na kojima se donose odluke o prosleđivanju paketa - a ivice (u terminologiji teorije grafova - „periferije“) koje povezuju te čvorove predstavljaju fizičke linkove između ruteru. Takva grafička apstrakcija računarske mreže prikazana je na slici 4.25. Grafove koji predstavljaju mape stvarnih mreža možete naći u knjigama [Dodge 1999, Cheswick 2000]; opis kvaliteta modeliranja različitih grafičkih modela Interneta naći ćete u knjigama [Zegura 1997, Faloutsos 1999].

Kao što se vidi na slici 4.25, ivica takođe ima vrednost koja predstavlja njenu cenu. Cena može da zavisi od fizičke udaljenosti kojom se prostire taj link (na primer, prekoceanski link može da ima veću cenu od kratkog zemaljskog linka), od brzine linka ili od novčanog iznosa cene pridružene tom linku. Za naše trenutne potrebe uzećemo cenu linka kao datu i nećemo brinuti o tome kako se ona utvrđuje. Za svaku ivicu  $(x,y)$  u  $E$  cenu ivice između čvorova  $x$  i  $y$  označićemo sa  $c(x,y)$ . Ako par  $(x,y)$  ne pripada  $E$  stavićemo daje  $c(x,y) \sim \infty$ . Takođe, sve vreme ćemo razmatrati neusmerene grafove (tj. grafove čije ivice nemaju smer), tako daje ivica  $(x,y)$  jednaka ivici  $(y,x)$  i daje  $c(x,y) = c(y,x)$ . Osim toga, za čvor  $y$  kažemo daje **susedan** Čvoru  $x$  ako  $(x,y)$  pripada kolekciji  $E$ .

Ako uzmemo da su ivicama ove grafičke apstrakcije dodeljene cene, prirodan cilj algoritma rutiranja je da se pronađu putanje sa najmanjom cenom od izvora do odredišta. Da bismo problem postavili preciznije, prisjetite se daje **putanja** u grafu

$G = (N, E)$  niz čvorova  $(*, x_1, x_2, \dots, x_p)$  takvih daje svaki od parova  $(x_i, x_j)$ ,  $i \neq j$ ,

$(x_i, x_j)$  ivica **iz** $E$ . Cena putanje  $(x_1, x_2, \dots, x_p)$  je prosto zbir cena svih ivica na putanji, tj.  $c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_p, x_1)$ . Za svaki par čvorova  $x$  i  $y$  obično postoji više putanja koje ih spajaju, a svaka od njih ima svoju cenu. Jedna ili više od ovih putanja su **putanje sa najmanjom cenom**. Problem najmanje cene je, dakle, jasan: pronaći putanju između izvora i odredišta koja ima najmanju cenu. Na primer, na slici 4.25 putanja sa najmanjom cenom od izvornog čvora  $u$  do odredišnog čvora  $w$  je  $(u, x, y, w)$  sa cenom putanje jednakom 3. Obratite pažnju na to da ako su cene svih linkova jednake, putanja sa najnižom cenom je takođe i **najkraća putanja** (tj. putanja koja od izvora do odredišta prelazi najmanji broj linkova).



Slika 4.25 ♦ Apstraktni model mreže

Počnite od jednostavne vežbe da pronađete putanje sa najmanjom cenom od čvora  $u$  do čvora  $z$  na slici 4.25 i razmislite malo o načinu na koji ste izračunali tu putanju. Ako razmišljate slično većini ljudi, utvrdili ste putanju od Čvora  $u$  do čvora  $z$  tako što ste posmatrali sliku 4.25, isprobali nekoliko putanja od  $u$  do  $z$  i na neki način došli do zaključka daje putanja koju ste izabrali najjeftinija od svih mogućih putanja. (Da li ste proverili svih 17 mogućih putanja između  $u$  i  $z$ ? Verovatno niste!) Takvo izračunavanje je primer centralizovanog algoritma rutiranja - algoritam rutiranja je izvršen najednom mestu, u vašoj glavi, sa kompletnim informacijama o celoj mreži. Jedan od načina na koji se algoritmi rutiranja klasificuju je po tome da li su globalni ili decentralizovani.

- ♦ Globalni algoritam rutiranja izračunava putanje sa najmanjom cenom od izvora do odredišta pomoću kompletног/globalnog znanja o celoj mreži. On upotrebljava podatke o povezanosti svih čvorova i cene svih linkova. Znači, algoritam mora nekako da pribavi te informacije pre samog izračunavanja. Samo izračunavanje može da se izvršava na jednoj lokaciji (centralizovani globalni algoritam rutiranja) ili da bude replicirano na više lokacija. Ključna karakteristika je, međutim, to da globalni algoritam ima **kompletne** informacije o povezanosti i o cenama linkova. U praksi, algoritmi sa globalnim informacijama o stanju često se nazivaju algoritmima prema stanju linkova (*link state*, LS), pošto algoritam mora biti upućen u cenu svakog linka u mreži. Algoritam globalnog stanja linkova proučićemo u odeljku 4.5.1.
- ♦ U decentralizovanom algoritmu rutiranja, izračunavanje putanje sa najmanjom cenom izvršava se na iterativni distribuirani način. Nijedan čvor nema kompletne informacije o cenama svih mrežnih linkova. Umesto toga, svaki čvor počinje samo od saznanja o cenama vlastitih direktno povezanih linkova. Zatim pomoću iterativnog procesa izračunavanja i razmene informacija sa susednim čvorovima (tj. čvorovima koji se nalaze na suprotnoj strani linkova sa kojima je čvor neposredno povezan) čvor postepeno izračunava putanje sa najnižom cenom do jednog odredišta ili skupa odredišta. Decentralizovani algoritam rutiranja koji ćemo proučiti u odeljku 4.5.2. poznat je kao algoritam sa vektorom rastojanja (*distance vector*, DV) zato što svaki čvor održava vektor procenjenih cena (rastojanja) do svih ostalih čvorova u mreži.

Još jedan način za grubu klasifikaciju algoritama rutiranja je prema tome da li su statički ili dinamički. Kod statičkih algoritama rutiranja rute se menjaju veoma retko, često ljudskom intervencijom (na primer, čovek ručno uređuje tabelu prosleđivanja u ruteru). U dinamičkim algoritmima rutiranja putanje rutiranja se menjaju kako se menja opterećenje mreže saobraćajem ili njena topologija. Dinamički algoritam može se izvršavati povremeno ili kao direktni odgovor na promene topologije ili cene linkova. Dok dinamički algoritmi bolje reaguju na promene u mreži, oni su takođe i osetljiviji na probleme kao što su petlje rutiranja i oscilacije na rutama.

Treći način da se klasifikuju algoritmi rutiranja je prema tome da li su osetljivi na opterećenje. Kod **algoritma osetljivog na opterećenje**, cene linkova se menjaju dinamički zavisno od trenutnog nivoa zagušenja određenog linka. Ako se linku koji je trenutno zagušen pridruži visoka cena, algoritam rutiranja će težiti da bira rute koje izbegavaju taj zagušeni link. Na početku su algoritmi rutiranja ARPAneta bili osetljivi na opterećenje [McQuillan 1980] ali se pojavio niz poteškoća [Huitema 1998]. Algoritmi rutiranja današnjeg Interneta (kao što su PJP, OSPF i BGP) **nisu osetljivi na opterećenje** pošto cena linka ne reflektuje eksplisitno trenutni (ili nedavni) stepen zagušenja.

#### 4.5.1 Algoritam rutiranja prema stanju linkova

Imajte na umu da su u algoritmu prema stanju linkova topologija mreže i cene svih linkova poznate, tj. dostupne kao podaci za algoritam stanja linkova. U praksi ovo se postiže tako što svaki čvor difuzno emituje identitet i cenu svojih povezanih linkova *svim* ostalim ruterima u mreži. U praksi (na primer pomoću Internetovog protokola rutiranja OSPF koji obraćujemo u odeljku 4.6.1) ovo se često postiže algoritmom za **difuzno emitovanje stanja linkova** [Perlman 1999]. Algoritme za difuzno emitovanje obradićemo u odeljku 4.7. Na kraju svih difuzno emitovanih stanja linkova svi će čvorovi imati identičan i kompletan pregled mreže. Tada svaki Čvor može da izvršava algoritam prema stanju linkova i da izračuna isti skup putanja sa najnižom cenom kao i svaki drugi Čvor.

Algoritam prema stanju linkova koji ćemo sada predstaviti poznat je kao Dijkstrin algoritam, prema svom pronalazaču. Primov algoritam je veoma sličan; u [Cormen 2001] nači ćete opšti opis grafovskih algoritama. Dijkstrin algoritam izračunava putanje sa najnižom cenom od jednog čvora (izvora, koji ćemo nazvati  $u$ ) do svih ostalih čvorova u mreži. Dijkstrin algoritam je iterativan i ima svojstvo da su nakon jfe-te iteracije algoritma poznate putanje sa najnižom cenom do  $k$  odredišnih čvorova, a među putanjama sa najnižom cenom do svih odredišnih čvorova; ili, ovih  $k$  putanja imaju  $k$  najnižih cena. Defmisaćemo sledeće označavanje:

- ◆  $D(y)$ : cena putanje od izvornog čvora do odredišta u koja trenutno (prema trenutnoj iteraciji algoritma) ima najmanju cenu.
- ◆  $p(y)$ : prethodni Čvor (sused čvora  $v$ ) na putanji sa trenutno najmanjom cenom od izvora do  $v$ .
- ◆  $N'$ : skup čvorova sa konačno utvrđenom najjeftinijom putanjom od izvora.

Globalni algoritam rutiranja sastoji se od jednog početnog koraka nakon kojeg sledi petlja. Broj ponavljanja petlje jednak je broju čvorova u mreži. Na kraju, algoritam će imati izračunate najkraće putanje od izvornog Čvora  $u$  do svakog drugog čvora u mreži.

Algoritam stanja linkova (Link State, LS) za izvorni čvor  $u$ :

```

1 Inicijalizacija:
2 N' = {u}
3 za sve čvorove v
4 ako je v sused u
5 then D(v) = c(u,v)
6 else D(v) = » 7
7 Loop
8 petlja w nije u N' tako da je D<w) minimum koji
9 w dodaje u N'
10 ažurira D(v) za sve v susedne w koji nisu N':
11 D(v) = min(D(v), D(w) + c(w,v))
12 /* nova cena do v je stara cena do v ili cena poznate
13 najkraće putanje do w plus cena od w do v */
14 until N' = N

```

Kao primer, razmotrimo mrežu na slici 4.25 i izračunajmo putanje sa najnižim cenama **od  $u$**  do svih mogućih odredišta. Tabelarni pregled izračunavanja po ovom algoritmu prikazan je u tabeli 4.3, gde svaki red tabele daje vrednosti promenljivih iz algoritma na kraju jedne iteracije. Razmotrimo detaljno prvi nekoliko koraka.

- ◆ U koraku inicijalizacije trenutno poznatim putanjama sa najmanjim cenama **od  $u$  do njegovih direktno povezanih suseda,  $v$ ,  $x$  i  $w$**  daju se početne vrednosti 2, 1 odnosno 5. Obratite posebnu pažnju na to da cena do  $w$  dobija vrednost 5 (mada ćemo uskoro videti da postoje putanja sa manjom cenom) postoje to cena direktnog linka **Gadnog skoka** **od  $u$  do  $w$** . Cenama **doyiz** daje se vrednost beskonačno zato što oni nisu direktno povezani sa  $u$ .
- ◆ U prvoj iteraciji pregledamo čvorove koje još nismo dodali skupu  $N'$  i pronalazimo čvor sa najmanjom cenom na kraju prethodne iteracije. Taj čvor je  $x$  sa cenom 1 i tako se  $x$  dodaje skupu  $N'$ . Tada se izvršava red 12 algoritma LS da bi se ažuriralo  $D(v)$  za sve Čvorove  $v$ , čime se dobijaju rezultati prikazani u drugom redu (korak 1) tabele 4.3. Cena putanje **do  $v$**  nije se promenila. Cena putanje **do  $w$**  (koja je na kraju inicijalizacije iznosila 5) ako se ide preko  $x$  ima cenu 4. Zato se bira ova putanja sa nižom cenom, a prethodni čvor  $\mathbf{v}$  na najkraćem putu **od  $u$  postaje  $x$** . Slično tome, cena **do  $y$**  (preko  $x$ ) iznosi 2, a tabela se ažurira u skladu sa tim.

- ♦ U drugoj iteraciji čvorovi  $v$  i  $y$  se pokazuju kao najjeftiniji (2), pa proizvoljno rešavamo nerešeni rezultat i u skup/ $w$  dodajemo  $y$  tako da  $A''$  sada sadrži  $u, x \setminus y$ . Cena do preostalih čvorova još se ne nalazi u  $N \setminus$  tj. čvorovi  $v, w, z$  ažuriraju se u 12. redu algoritma LS gde se dobijaju rezultati prikazani u trećem redu tabele 4.3.

- ♦ I tako dalje...

Kada se algoritam LS završi, za svaki čvor se zna prethodnik na putanji sa najnižom cenom od izvornog čvora. Za svakog prethodnika takođe je poznat *njegov* prethodnik, pa na taj način možemo da napravimo celu putanju od izvora do svih odredišta. Tabela prosledivanja u svakom čvoru, na primer u čvoru  $u$ , zatim može da se napravi od ovih informacija tako što se za svako odredište Čuva čvor za sledeći skok po putanji sa najnižom cenom od  $u$  do tog odredišta.

U čemu je računska složenost ovog algoritma? To jest, ako je dato  $n$  čvorova (ne računajući izvor), koliko izračunavanja treba izvršiti u najgorem slučaju da bi se pronaše najjeftinije putanje od izvora do svih odredišta? U prvoj iteraciji moramo da pretražimo svih  $n$  čvorova da bismo odredili čvor  $w$  koji ima najmanju cenu a nije u skupu  $A^7$ . U drugoj iteraciji moramo da proverimo  $n - 1$  čvorova da bismo utvrdili najmanju cenu; u trećoj iteraciji proveravamo  $n - 2$  čvorova itd. Sveukupno, ukupan broj čvorova koje moramo da ispitamo u svim iteracijama je  $n(n + 1)/2$ , pa tako kažemo da prethodna implementacija algoritma prema stanju linkova ima složenost najgoreg slučaja reda  $n$  na kvadrat:  $O(n^2)$ . (Pomoću jedne složenije implementacije ovog algoritma, u kojoj se koristi struktura podataka poznata kao hip, moguće je pronaći minimum već u devetom redu u logaritamskom umesto linearном vremenu, čime se složenost smanjuje.)

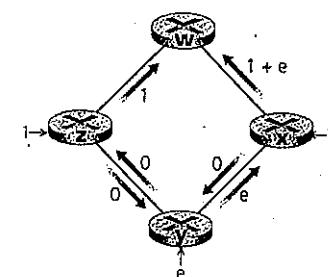
Pre nego što zaključimo opis algoritma LS, razmotrićemo nepoželjan slučaj do kojeg može doći. Na slici 4.26 prikazana je topologija jednostavne mreže gde su cene linkova jednake opterećenju koje se tim linkom prenosi, na primer, da reflektuju kašnjenje koje bi se pojavljivalo. U ovom primeru, cene linkova nisu simetrične; tj.  $c(u,v)$  jednak je  $c(y,u)$  samo ako su na linku  $(u, v)$  opterećenja u oba smera jednaka. U ovom primeru, iz čvora  $z$  polazi jedinica saobraćaja sa odredištem  $w$ , sa Čvora  $x$  ta-

| Korak | $N'$    | $D(v), p(v)$ | $D(w), p(w)$ | $D(x), p(x)$ | $D(y), p(y)$ | $D(z), p(z)$ |
|-------|---------|--------------|--------------|--------------|--------------|--------------|
| 0     | $u$     | $2, u$       | $5, u$       | $1, u$       | $\infty$     | $\infty$     |
| 1     | $ux$    | $2, u$       | $4, x$       |              | $2, x$       | $\infty$     |
| 2     | $uy$    | $2, u$       | $3, y$       |              | $4, y$       |              |
| 3     | $uyw$   |              | $3, y$       |              | $4, y$       |              |
| 4     | $uyvw$  |              |              |              | $4, y$       |              |
| 5     | $uyvwz$ |              |              |              |              |              |

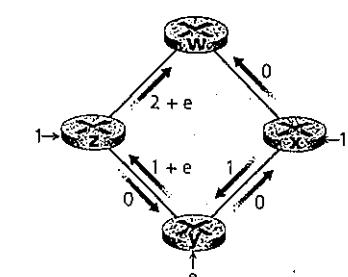
Tabela 4.3 ♦ Izvršavanje algoritma stanja linka na mreži sa slike 4.25.

kode kreće jedinica saobraćaja sa odredištem  $w$ , a čvor  $y$  ubacuje količinu saobraćaja jednaku  $e$ , takođe prema odredištu  $w$ . Početno rutiranje prikazano je na slici 4.26(a) gde cene linkova odgovaraju količini prenetog saobraćaja.

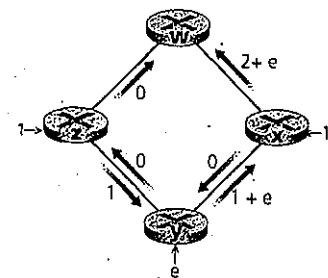
Kod sledećeg izvršavanja algoritma LS, čvor  $x$  utvrđuje (na osnovu cena linkova prikazanih na slici 4.26(a)) daje putanja do  $w$  u smeru kazaljke na satu ima cenu 1, dok putanja do istog odredišta u smeru suprotnom od kazaljke na satu koja se dotele koristila) ima cenu  $1 + e$ . Zbog toga je putanja do  $w$  sa najnižom cenom sada u smeru kazaljke na satu. Slično tome,  $x$  utvrđuje da je njegova putanja sa najnižom cenom do  $w$  takođe u smeru kazaljke na satu, pa dobijamo cene kao na slici 4.26(b). Pri sledećem izvršavanju algoritma LS, čvorovi  $x, y, z$  i svi primećuju putanju sa cenom 0 prema  $w$  u smeru suprotnom od kazaljke na satu i svi rutiraju svoj saobraćaj rutama suprotnim od kazaljke na satu. Kod sledećeg izvršavanja algoritma LS,  $x, y, z$  svi rutiraju svoj saobraćaj u smeru kazaljke na satu.



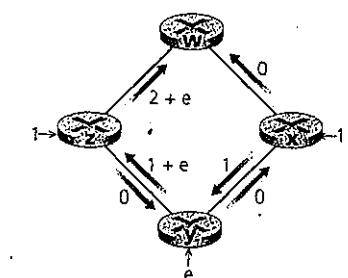
a. Početno rutiranje



b.  $x, y$  i  $z$  otkrivaju bolju putanju prema  $w$ , u smeru kazaljke na satu



c.  $x, y, z$  otkrivaju bolju putanju prema  $w$ , suprotno od kazaljke na satu



d.  $x, y, z$  otkrivaju bolju putanju prema  $w$ , u smeru kazaljke na satu

Slika 4.26 ♦ Oscilacije sa rutiranjem prema stanju linkova.

Šta da se uradi da bi se sprečile takve oscilacije (do kojih može doći ne samo u algoritmu LS već u svakom algoritmu koji koristi merenja zagušenja ili kašnjenja na linkovima)?

Jedno rešenje bilo bi da se propiše da cene linkova ne smeju zavisiti od količine prenetog saobraćaja - to rešenje nije prihvatljivo pošto je jedan od ciljeva rutiranja upravo izbegavanje veoma zagušenih linkova (na primer, onih sa velikim kašnjenjem). Drugo rešenje je da se spreči istovremeno izvršavanje algoritma LS u svim ruterima. Ovo izgleda prihvatljivije, pošto bismo se nadali da čak i kada ruteri izvršavaju algoritam LS u istim vremenskim intervalima istanca algoritma ne bi bila ista u svim čvorovima. Zanimljivo je da su istraživači nedavno utvrdili da ruteri na Internetu mogu sami da se međusobno sinhronizuju [Floyd Synchronization 1994]. To jest, čak i kada na početku izvršavaju algoritam u istim vremenskim intervalima ali u različito vreme, primerak istanca u ruterima vremenom postaje i ostaje sinhronizovan. Jedan od načina da se izbegne takva samosinhronizacija jeste da svaki ruter slučajno bira vreme kada će da pošalje objavu linka.

Pošto smo proučili algoritam prema stanju linkova, sada ćemo proučiti drugi značajan algoritam rutiranja koji se koristi u današnjoj praksi - algoritam rutiranja sa vektorom rastojanja.

#### 4.5.2 Algoritam rutiranja sa vektorom rastojanja

Dok algoritam LS koristi globalne informacije, algoritam **sa vektorom rastojanja (distance vector, DV)** je iterativan, asinhron i distribuiran. On je *distribuiran* po tome što svaki čvor prima odredene informacije od jednog svog *direktno povezanog suseda* ili od više njih, izračunava, a zatim distribuira rezultate svog izračunavanja tim istim susedima. On je *iterativan* po tome što se ovaj postupak ponavlja sve dok ima informacija za razmenu među susedima. (Zanimljivo je, videćemo da se ovaj algoritam sam zaustavlja - ne postoji signal da izračunavanje treba prekinuti; ono se prosti završi.) Algoritam je *asinhron* po tome što nije neophodno da svi čvorovi rade usaglašeno. Videćemo daje asinhroni, iterativni, distribuirani algoritam koji se sam prekida daleko zanimljiviji i zabavniji od centralizovanog algoritma!

Pre nego što predstavimo algoritam DV, biće korisno da opišemo jedan važan odnos koji postoji između cene i-putanja sa najnižom cenom. Neka  $d_{ij}$  bude cena najjeftinije putanje od čvora  $i$  do čvora  $j$ . Najniže cene vezane su slavnom Belman-Fordovom jednačinom:

$$d_x(y) = \min_w \{ c(x, w) + d_w(y) \}, \quad (4.1)$$

Izraz  $\min_w$  u jednačini predstavlja minimum svih direktno povezanih suseda čvora  $x$ . Belman-Fordova jednačina je intuitivno jasna. Zaista, ako nakon puta od  $x$  do  $y$  krenemo putanjom sa najnižom cenom od  $v$  do  $y$ , cena putanje biće  $c(x, v) + d_v(y)$ . Pošto

putovanje moramo da počnemo prelaskom do nekog suseda  $v$ , najmanja cena od  $x$  do  $y$ , izračunavaće se kao zbir minimuma cena putanja  $c(x, v) + d_v(y)$  za svakog suseda  $v$ .

Ali zbog onih koji možda sumnjuju u validnost ove jednačine, proverimo je za izvorni čvor  $u$  i odredišni čvor  $z$  sa slike 4.25. Izvorni čvor  $u$  ima tri suseda:  $v$ ,  $x$  i  $w$ . Ispitivanjem različitih putanja grafa, lako je videti daje  $d_u(z) = 5$ ,  $d_v(z) = 3$  i  $d_x(z) = 3$ . Ako ove vrednosti ubacimo u jednačinu 4.1, uz cene  $c(u, v) = 2$ ,  $c(u, x) \sim 1$  i  $c(u, w) = 5$  dobijamo

$d_u(z) = \min\{2 + 5, 5 + 3, 1 + 3\} = 4$ , što je očigledno tačno, a to je upravo rezultat koji za ovu mrežu daje i Dijkstrin algoritam. Ova brza provera trebalo bi da ukloni sve eventualne sumnje.

Belman-Fordova jednačina nije samo intelektualni kuriozitet. Ona zaista ima značajnu praktičnu vrednost. Konkretno, rešenje Belman-Fordove jednačine obezbeđuje stavke za tabelu prosledivanja čvora  $x$ . Da bismo to videli, neka  $v^*$  bude susedni čvor za koji je postignut minimum u jednačini 4.1. Zatim, ako čvor  $x$  želi da pošalje paket čvoru  $y$  po najjeftinije putanji, trebalo bi prvo da prosledi paket do čvora  $v^*$ . Prema tome, u tabeli prosledivanja čvora  $x$  kao ruter prvi skok za krajnje odredište  $y$  bio bi naveden čvor  $v^*$ . Još jedan značajan praktičan doprinos Belman-Fordove jednačine jeste to što ona nagoveštava oblik komunikacije među susedima do koje će doći u algoritmu DV.

Osnovna zamisao je sledeća. Svaki čvor počinje od  $D_x(y)$ , procenjene cene najjeftinije putanje od njega do čvoray, za sve Čvorove iz  $N$ . Neka  $D_x = [D^x(y): y \in N]$  bude vektor rastojanja čvora  $x$ , a to je vektor procenjenih troškova od  $x$  do svih ostalih čvorova,  $y$ , iz  $N$ . U algoritmu DV, svaki čvor  $x$  održava sledeće podatke rutiranja:

- ◆ Za svakog suseda  $v$ , cenu  $c(x, v)$  od  $v$  do direktno povezanog suseda  $v$ .
- ◆ Vektor rastojanja čvorax, tj.  $D_x = [D_x(y): y \in N]$ , koji sadrži procenjene troškove do svih odredišta  $y$ , iz  $N$ .
- ◆ Vektore rastojanja svih suseda, tj.  $= [D_y(y): y \in N]$  svih suseda  $v$  čvora  $v$ .

U distribuiranom asinhronom algoritmu, s vremenom na vreme svaki čvor šalje svoj ažurirani vektor rastojanja svim svojim susedima. Kada čvor\* primi novi vektor rastojanja od nekog svog suseda  $v$ , on memorise taj vektor rastojanja i zatim pomoću Belman-Fordove jednačine ažurira vlastiti vektor rastojanja na sledeći način:

$$D_x(y) = \min_w \{ c(x, w) + D_w(y) \} \text{ za svaki čvor } y \in N$$

Ako se vektor rastojanja čvora  $x$  prilikom ovog ažuriranja promeni, čvor  $x$  će zatim poslati primerak svog vektora rastojanja svim svojim susedima koji zatim mogu da ažuriraju vlastite vektore rastojanja. Čudesno je to da, ako svi čvorovi stalno razmenjuju svoje vektore rastojanja na asinhroni način, svi procenjeni troškovi  $D_x(y)$  teže iznosu  $d_x(y)$ , stvarnoj ceni najjeftinije putanje od čvora  $x$  do čvora  $y$  [Bersekas 1992]!

Algoritam sa vektorom rastojanja Na svakom čvoru,  $x$ :

```

1 Initialization:
2 for ali destinations y in N:
3 $D_x(y) = c(x,y)$ /* ako y nije sused, tada je c
4 $(x,y) = - */$
5 for each neighbor w
6 $D_x(w) = \infty$ for ali destinations y in N
7 for each neighbor w
8 send distance vector $D_x = [D_x(y) : y \in N]$ to w
9 loop
10 wait (until I see a link cost change to neighbor w
11 or until I receive a distance vector from neighbor w)
12
13 for each y in N:
14 $D_x(y) = \min_v \{ c(x,v) + D_v(y) \}$
15 if $D_x(y)$ changed for any destination y
16 send distance vector $D_x = [D_x(y) : y \in N]$ to all neighbors
17 forever
18
19

```

U algoritmu DV (*distance vector*) vidimo da Čvor  $x$  ažurira svoj vektor sa proce-nama rastojanja kada primeti pramenu cene na nekom od svojih direktno povezanih linkova ili kada od nekog suseda primi ažurirani vektor rastojanja. Ali da bi vlastitu tabelu prosleđivanja ažurirao za dato odredište  $y$ , Čvoru  $x$  nije potrebno najkraće rastojanje do  $y$  već susedni čvor  $v^*(y)$  tj. ruter prvog skoka na najkraćoj putanji do  $y$ . Kao što biste i očekivali, ruter prvog skoka  $v^*(y)$  je sused u koji ostvaruje minimum u 14. redu algoritma DV. (Ako više suseda vostvaruju isti minimum, tada  $v^*(y)$  može da bude bilo koji od njih.) Prema tome, u redovima 13-14 za svako odredište  $y$ , Čvor  $x$  takođe utvrđuje  $v^*(y)$  i ažurira svoju tabelu prosleđivanja za odredište  $y$ .

Imajte na umu da je algoritam LS globalan u smislu da svaki čvor mora prvo da pribavi kompletну mapu mreže pre nego što primeni Dijkstrin algoritam. Algoritam DV je decentralizovan i ne koristi takve globalne informacije. Zaista, jedine informacije u svakom čvoru su cene linkova prema direktno povezanim susedima i informacije koje prima od tih suseda. Svaki čvor Čeka na ažuriranje od bilo kog suseda (redovi 16-17), izračunava nov vektor rastojanja kada primi ažuriranje (red 14) i distribuira susedima svoj novi vektor rastojanja (redovi 16-17). Algoritam DV se u praksi koristi u mnogim protokolima rutiranja uključujući Internetove protokole RIP, i BGP, ISO IDRP, Novellov IPX i prvobitni ARPAnet.

Na slici 4.27 prikazanje rad algoritma DV za jednostavnu mrežu od tri čvora prikazanu na vrhu slike. Rad algoritma prikazan je na sinhroni način gde svi čvorovi istovremeno dobiju vektore rastojanja od svojih suseda, izračunavaju svoje nove vektore rastojanja i obaveštavaju svoje susede ako je došlo do promene vektora rastojanja. Pošto proučite ovaj primer, trebalo bi da uvidite da algoritam pravilno funkcioniše i na asinhroni način, gde se izračunavanja u čvorovima kao i pravljenje i primanje ažurnih informacija dogada u bilo kom trenutku.

U sasvim levoj koloni prikazane su tri početne tabele rutiranja sva tri čvora. Na primer, tabela u gornjem levom uglu je početna tabela rutiranja čvora  $x$ . Unutar odredene tabele rutiranja, svaki red je jedan vektor rastojanja - konkretno, tabela rutiranja svakog čvora sadrži njegov vlastiti vektor rastojanja i vektore rastojanja njegovih neposrednih suseda. Prema tome, prvi red početne tabele rutiranja čvora  $x$  je  $D_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$ . Drugi i treći red ove tabele su poslednji primljeni vektori rastojanja od čvorova  $y$  i  $z$  odnosno  $z$ . Pošto u trenutku inicijalizacije čvor  $x$  nije još ništa primio od Čvorova  $y$  i  $z$ , stavke u drugom i trećem redu imaju početnu vrednost beskonačno.

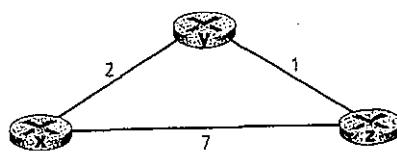
Nakon inicijalizacije, svaki čvor šalje svoj vektor rastojanja svakom od svoja dva suseda. Ovo je na slici 4.27 prikazano strelicama koje polaze iz prve kolone tabela prema drugoj koloni tabela. Na primer, čvor  $x$  šalje svoj vektor rastojanja  $D_x = [0, 2, 7]$  u oba čvora  $y$  i  $z$ . Pošto primi ažuriranja, svaki čvor ponovo izračunava vlastiti vektor rastojanja. Na primer, čvor  $y$  izračunava:

$$\begin{aligned} D_y(x) &= 0 \\ D_y(y) &= \min\{c(x,y) + D_x(y), c(x,z) + D_x(z)\} = \min\{2+0, 7+1\} = 2 \\ D_y(z) &= \min\{c(x,z) + D_x(z), c(y,z) + D_x(z)\} = \min\{2+1, 7+0\} = 3 \end{aligned}$$

U drugoj koloni su, dakle, za svaki čvor prikazani njegov vlastiti vektor rastojanja i vektori rastojanja koji su upravo primljeni od suseda. Obratite, na primer, pažnju na to da Čvor  $x$  procenjuje da se najjeftinija putanja do čvora  $z$ ,  $D_x(z)$ , promenila od 7 na 3. Takođe obratite pažnju na to da za čvorove  $y$  i  $z$  minimum predstavlja čvor  $y$ . Prema tome, u ovoj fazi algoritma ruteri prvog skoka su  $v^*(y) = y$  i  $takode v^*(z) = y$ .

Pošto čvorovi ponovo izračunaju vlastite vektore rastojanja, oni ponovo šalju svoje ažurirane vektore rastojanja susedima (ukoliko je došlo do promene). Ovo je na slici 4.27 prikazano strelicama iz druge kolone tabela u treću. Primetićete da samo Čvorovi  $x$  i  $z$  šalju ažuriranja: vektor rastojanja čvora  $y$  nije se promenio i zato on ne šalje ništa. Pošto prime ažuriranja, čvorovi ponovo izračunavaju svoje vektore rastojanja i ažuriraju svoje tabele rutiranja koje su prikazane u trećoj koloni.

Postupak primanja novih cena od suseda, ponovnog izračunavanja stavki u tabeli rastojanja i obaveštavanja suseda o promenjenim cenama najjeftinije putanje do nekog odredišta nastavlja se sve dok ima poruka. U tom trenutku, pošto se ne šalju nikakve poruke sa novim vrednostima, tabele rastojanja se vise neće preračunavati



Tabela

|   |  | cena do |   |   |
|---|--|---------|---|---|
|   |  | x       | y | z |
|   |  | 0       | 2 | 7 |
| x |  | 0       | 2 | 7 |
| y |  | 2       | 0 | 1 |
| z |  | 7       | 1 | 0 |

|   |  | cena do |   |   |
|---|--|---------|---|---|
|   |  | x       | y | z |
|   |  | 0       | 2 | 3 |
| x |  | 0       | 2 | 3 |
| y |  | 2       | 0 | 1 |
| z |  | 7       | 1 | 0 |

|   |  | cena do |   |   |
|---|--|---------|---|---|
|   |  | x       | y | z |
|   |  | 0       | 2 | 3 |
| x |  | 0       | 2 | 3 |
| y |  | 2       | 0 | 1 |
| z |  | 7       | 1 | 0 |

|   |  | cena do |   |   |
|---|--|---------|---|---|
|   |  | x       | y | z |
|   |  | 0       | 2 | 1 |
| x |  | 0       | 2 | 1 |
| y |  | 2       | 0 | 1 |
| z |  | 7       | 1 | 0 |

|   |  | cena do |   |   |
|---|--|---------|---|---|
|   |  | x       | y | z |
|   |  | 0       | 2 | 3 |
| x |  | 0       | 2 | 3 |
| y |  | 2       | 0 | 1 |
| z |  | 7       | 1 | 0 |

|   |  | cena do |   |   |
|---|--|---------|---|---|
|   |  | x       | y | z |
|   |  | 0       | 2 | 1 |
| x |  | 0       | 2 | 1 |
| y |  | 2       | 0 | 1 |
| z |  | 7       | 1 | 0 |

Slika 4.27 ♦ Algoritam vektora rastojanja (DV)

i algoritam ulazi u stanje mirovanja; tj. svi čvorovi izvršavaju čekanje u redovima 10-11 algoritma DV. Algoritam ostaje u stanju mirovanja dok ne dođe do promene cene nekog linka, što ćemo sada opisati.

#### Algoritam vektora rastojanja: promene cena linkova i otkazivanje linkova

Kada čvor koji izvršava algoritam DV otkrije promenu u ceni linka između sebe i suseda (redovi 10-11), on ažurira svoju tabelu rastojanja (redovi 13-14) i ako je došlo do promene cene najjeftinije putanje, obaveštava svoje susede (redovi 16-17). Na slici 4.28(a) prikazano je ovo ponašanje u situaciji kada se cena linka od  $y$  prema  $x$  promenila od 4 na 1. Ovde ćemo se usredosrediti samo na stavke u tabelama rastojanja čvorova  $y$  i  $z$  do odredišta (reda)  $x$ . Algoritam DV dovodi do sledećeg niza dogadaja:

- ♦ U trenutku  $t_0$ ,  $y$  otkriva promenu u ceni linka (cena se promenila od 4 na 1), ažurira svoj vektor rastojanja i obaveštava svoje susede o toj promeni pošto se promenio vektor rastojanja.
- ♦ U trenutku  $t_1$  prima obaveštenje od  $y$  i zatim ažurira svoju tabelu. Pošto izračuna novu najnižu cenu do  $x$  (smanjila se od 5 na 2), on šalje svojim susedima novi vektor rastojanja.
- ♦ Tj. u trenutku  $t_2$ ,  $y$  prima obaveštenje od čvora  $z$  i ažurira svoju tabelu rastojanja. U čvoru  $y$  se nisu promenile najniže cene, pa zato  $y$  ne šalje nikakvu poruku čvoru  $z$ . Algoritam dolazi u stanje mirovanja.

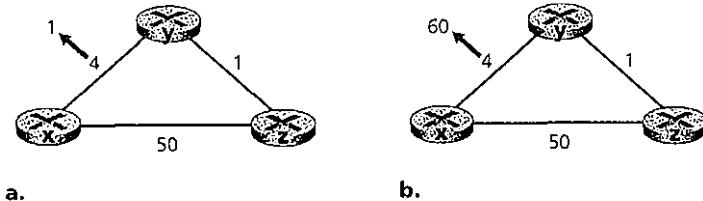
Potrebne su, znači, samo dve iteracije algoritma DV da bi se dostiglo stanje mirovanja. Radosne vesti o smanjenoj ceni između  $x$  i  $y$  brzo su se propagirale kroz mrežu.

Razmotrimo sada do čega može doći ako cena linka *porasle*. Uzmimo da cena linka između  $x$  i  $y$  poraste od 4 na 60 kao što je prikazano na slici 4.28(b).

- Pre promene cene linka imamo  $D_y(x) = 4$ ,  $D_y(z) = 1$ ,  $D_z(y) = 1$  i  $D_z(x) = 5$ . U trenutku  $t_0$ ,  $y$  otkriva promenu u ceni linka (cena se promenila od 4 na 60). Cvor  $y$  izračunava da nova najjeftinija putanja do  $x$  iznosi

$$D_y(x) = \min\{c(yj) + D_x(j), c(y, z) + D_z(z)\} = \min\{60+0, 1+5\} = 6$$

Naravno, sa našim globalnim znanjem o mreži mi vidimo da ta nova cena preko *nije tačna*. Ali, čvor jedino ima informaciju da direktna cena do\* iznosi 60, a daje zadnje obaveštenje koje je dobio od čvora  $z$  glasilo da se do\* može stići za 5. Zato, da bi stigao do  $x$ ,  $y$  bi sada išao preko  $z$  smatrajući da se od  $z$  do\* može stići za 5. U trenutku  $t_1$  imamo petlju rutiranja: da bi se stiglo do  $x$ ,  $y$  rutira preko  $z$ , a  $z$  rutira preko  $y$ . Petlja rutiranja je kao crna rupa - paket sa odredištem  $x$  koji u trenutku  $t_1$  stigne do  $y$  ili  $z$  šetače između ta dva čvora zauvek (ili dok se njihove tabele prosledivanja ne promene).



**Slika 4.28** ♦ Promena cene linka

2. Kako je čvor  $y$  izračunao novu minimalnu cenu prema  $x$ , on o tome obaveštava  $z$  u trenutku
3. Nakon trenutka  $t_1$ ,  $z$  prima novi vektor rastojanja  $\langle z \rangle_y$  sa novom najmanjom cenom iz čvora  $y$  do  $x$ , koja sada iznosi 6.  $z$  zna da može stići u čvory po ceni 1, pa tako izračunava da nova najniža cena do  $x$  iznosi  $D_z(x) = \min\{50+0, 1+6\} = 7$ . Kako se u čvoru  $z$  najniža cena do  $x$  povećala, on o tome obaveštava  $y$  u trenutku  $t_1$ .
4. Na sličan način, pošto primi novi vektor rastojanja iz  $z$ ,  $y$  tada utvrđuje da je  $D_y(x) = \infty$ , pa šalje svoj vektor rastojanja čvoru  $z$ , zatim utvrđuje da je  $D_z(x) = 9$  i šalje svoj vektor rastojanja čvoru  $v$  itd.

Dokle će taj proces da traje? Trebalо bi da uvidite da će petlja imati 44 iteracije (razmena poruka između  $y$  i  $z$ ) - sve dok  $z$  u jednom trenutku ne izračuna da je cena njegove putanje preko  $y$  veća od 50. U tom trenutku,  $z$  će (konačno!) utvrditi da njegova najjeftinija putanja do  $x$  ide preko njegove direktnе veze sa  $x$ .  $y$  će od tada putanje koje idu prema čvoru  $x$  rutirati preko čvora  $z$ . Videli smo da su loše vesti o povećanju cene linka zaista sporo putovali! Šta bi se dogodilo da se cena linka  $c(yj)$  promenila od 4 na 10.000, a daje cena  $c(zj)$  bila 9.999? Zbog takvih situacija se problem koji smo sada videli ponekad naziva problemom brojanja do beskonačnosti.

#### Algoritam sa vektorom rastojanja: dodavanje lažnog rastojanja

Specifična situacija sa petljom koju smo upravo opisali može da se izbegne pri-menom jedne tehnike poznate kao *lažno rastojanje*. Zamisao je jednostavna - ako  $z$  rutira preko  $y$  da bi se stiglo do odredišta  $x$ , tada će  $z$  obavestiti  $y$  da je njegovo rastojanje do  $x$  beskonačno, tj.  $z$  će obavestiti  $y$  da je  $D_z(x) = \infty$  (iako  $z$  zna daje u stvari  $D_z(x) = 5$ ).  $z$  će ostati pri toj maloj laži sve dok rutira putanje prema odredištu  $x$  preko  $y$ . Kako  $y$  veruje da od  $z$  nema putanje prema  $x$ ,  $y$  neće nikad pokušati da rutira putanje čije je odredište  $x$  preko  $z$ , sve dok  $z$  nastavlja da rutira putanje prema  $x$  preko  $y$  (i pretvara se da to ne radi).

Da pogledamo sada kako lažno rastojanje rešava konkretni problem petlje koji smo imali na slici 4.28(b). Zahvaljujući lažnom rastojanju, tabela rastojanja u čvoru  $y$  ukazuje daje  $D_z(x) = \infty$ . Kada se cena linka  $(x,y)$  promeni od 4 na 60 u trenutku  $t_0$ ,  $y$  ažurira svoju tabelu i nastavlja da rutira direktno prema  $x$ , iako je cena 60 veća, a obaveštava  $z$  o toj novoj ceni prema  $x$  tj.  $D_y(x) = 60$ . Pošto primi to obaveštenje u trenutku  $z$  odmah prebacuje rutu prema  $x$  direktno preko linka  $(z,x)$  po ceni 50. Pošto je ovo nova putanja sa najnižom cenom prema  $x$  i pošto putanja više ne prolazi preko  $y$ ,  $z$  sada obaveštava  $y$  da je  $D_z(x) = 50$  u trenutku  $t_2$ . Pošto primi obaveštenje od  $z$ ,  $y$  ažurira svoju tabelu rastojanja sa  $D(x) = 51$ . Isto tako, pošto se  $z$  sada nalazi na putanji najniže cene od  $y$  prema  $x$ ,  $y$  blokira suprotnu putanju od  $z$  prema  $x$  tako što obaveštava  $z$  u trenutku  $t_3$  da  $D_y(x) = \infty$  (iako  $y$  zna daje u stvari  $D(x) = 51$ ).

Da li lažno rastojanje rešava problem brojanja do beskonačnosti u opštem slučaju? Ne. Trebalо bi analizom da se uverite da petlje u kojima učestvuje tri ili više čvorova (a ne samo dva neposredno susedna čvora) ne mogu da se otkriju tehnikom lažnog rastojanja.

#### Poređenje algoritama rutiranja sa stanjem linkova i sa vektorom rastojanja

Algoritmi DV i LS imaju komplementarne pristupe izračunavanju rutiranja. U algoritmu DV (sa vektorom rastojanja) svaki čvor razgovara *jedino* sa svojim direktno povezanim susedima, ali ih snabdeva procenjenim najjeftinijim putanjama od njega do *svih* ostalih čvorova u mreži (za koje zna). U algoritmu LS (sa stanjem linkova), svaki čvor razgovara sa *svim* ostalim čvorovima (putem difuznog emitovanja) ali im saopštava *jedino* cene do svojih direktno povezanih suseda. Proučavanje algoritama sa stanjem linkova i sa vektorima rastojanja zaključićemo jednim brzim poređenjem nekih njihovih atributa.

- ♦ **Složenost poruka.** Videli smo da je za LS potrebno da svaki čvor zna cenu svih linkova u mreži. Da bi se to postiglo, potrebno je da se pošalje  $O(|N| |E|)$  poruka. Isto tako, kad god se promeni cena nekog linka, nova cena mora da se pošalje svim čvorovima. Algoritam DV pri svakoj iteraciji zahteva razmenu poruka među direktno povezanim susedima. Videli smo da vreme potrebno da se algoritam iscrpi zavisi od mnogo činilaca. Kada se promeni cena linka, algoritam DV propagira rezultate te promene samo ako nova cena linka menja najnižu cenu putanje prema nekom od čvorova vezanih za taj link.
- ♦ **Brzina izračunavanja.** Videli smo da naša implementacija algoritma LS raste sa kvadratom broja čvorova i da je potrebno  $O(|N|^2 |E|)$  poruka. Algoritam DV može da se izračunava sporo i može da ima petlje rutiranja dok se algoritam ne završi. U algoritmu DV takođe može doći do problema brojanja do beskonačnosti.
- ♦ **Robusinost.** Sta može da se dogodi ako ruter otkaze, pogrešno se ponaša ili je napadnut? Ako se koristi LS, ruter može da emituje neispravnu cenu nekog od svojih povezanih linkova (ali ne i drugih linkova). Čvor takođe može da ošteći

ili ispusti neke LS pakete o stanju linkova koje je primio. Ali, LS čvor izračunava samo vlastite tabele prosleđivanja; ostali čvorovi vrše slična izračunavanja sami za sebe. To znači da su u sistemu LS izračunavanja ruta donekle razdvojena što im daje određeni stepen robustnosti. U sistemu DV, čvor može da objavi neispravne najjeftinije putanje do bilo kojeg ili do svih odredišta. (Zaista, 1997. godine pokvareni ruter u malom posredniku za Internet usluge predao je ruterima državne okosnice pogrešne informacije o rutiranju. Zbog toga su drugi ruteri preplavili neispravni ruter saobraćajem i doveli do toga da veliki delovi Interneta čak nekoliko sati ostanu nepovezani [Neumann 1997].) Uopšteno govoreći, pri-mećujemo da se u sistemu DV pri svakoj iteraciji izračunavanja koja je izvršio jedan čvor predaju njegovom susedu, a zatim u sledećoj iteraciji indirektno i susedu njegovog suseda. Na taj način, neispravan rezultat iz jednog čvora može u sistemu DV da se proširi kroz celu mrežu.

Na kraju, nijedan od ovih algoritama nije bolji od drugog; zaista, u Internetu se koriste oba.

#### Ostali algoritmi rutiranja

Algoritmi LS i DV koje smo upravo obradili nisu samo u širokoj upotrebi, već su u suštini *jedini* algoritmi rutiranja koji se danas u praksi koriste. Ipak, tokom poslednjih 30 godina istraživači su predlagali mnoge algoritme rutiranja, od krajnje jednostavnih do veoma složenih i komplikovanih. Jedna velika klasa algoritama rutiranja posmatra saobraćaj paketa kao tokove između izvora i odredišta u mreži. U takvom pristupu, problem rutiranja može matematički da se formuliše kao problem ograničene optimizacije poznat kao problem mrežnog toka [Bertsekas 1991]. Pome-nućemo ovde još jedan skup algoritama rutiranja koji potiče iz sveta telefonije. Ovi algoritmi rutiranja sa komutiranjem vodova važni su za umrežavanje sa komutiranjem paketa u slučajevima kada za svaku konekciju koja se rutira preko nekog linka treba rezervisati resurse na tom linku (na primer, pomoćne memorije ili deo propusnog opsega linka). Dok formulacija problema rutiranja možda izgleda sasvim različita od formulacije rutiranja prema najnižoj ceni koju smo videli u ovom poglavljvu, postoji niz sličnosti bar u pogledu algoritma za pronađenje putanje (algoritma rutiranja). Čitalac može naći detaljan prikaz ovog područja istraživanja u [Ash 1998; Ross 1995; Girard 1990].

#### 4.5.3 Hjерархијско rutiranje

U proučavanju algoritama LS i DV posmatrali smo mrežu jednostavno kao kolekciju međusobno povezanih ruteru. Ruteri se nisu međusobno razlikovali jer su svi izvršavali isti algoritam rutiranja za izračunavanje putanja rutiranja kroz celu mrežu. U praksi je ovaj model i njegov pojam homogenog skupa ruteru koji svi izvršavaju isti algoritam rutiranja previše uproščen bar iz dva značajna razloga:

- ♦ *Skaliranje.* Kako broj ruteru raste, preopterećenje koje potiče od izračunavanja, čuvanja i saopštavanja informacija o rutiranju (na primer, ažurnog stanja linkova ili promena najnižih cena putanja) postaje preterano. Današnji javni Internet se sastoji od stotina miliona računara. Jasno je da bi za čuvanje informacija o rutiranju prema svakom od njih bile potrebne ogromne količine memorije. Emitovanje ažurnog stanja linkova svim ruterima u javnom Internetu utrošilo bi sav propusni opseg, pa ništa ne bi ostalo za slanje paketa podataka! Algoritam sa vektorom rastojanja koji bi iterirao preko tako velikog broja ruteru sigurno se nikada ne bi završio! Jasno je da treba nešto uraditi da bi se smanjila složenost izračunavanja ruta u tako velikim mrežama kakav je javni Internet.
- ♦ *Administrativna autonomija.* Mada inženjeri često zanemaruju želje kompanija da koriste ruteru po vlastitoj želji (na primer, da same biraju algoritme rutiranja), ili da kriju 'neke' aspekte interne organizacije svoje mreže od spoljnog sveta, ovo su značajna pitanja. Idealno bi bilo da organizacija koristi i administrira svoju mrežu kako sama hoće, a da ipak može da je povezuje sa ostalim spoljašnjim mrežama.

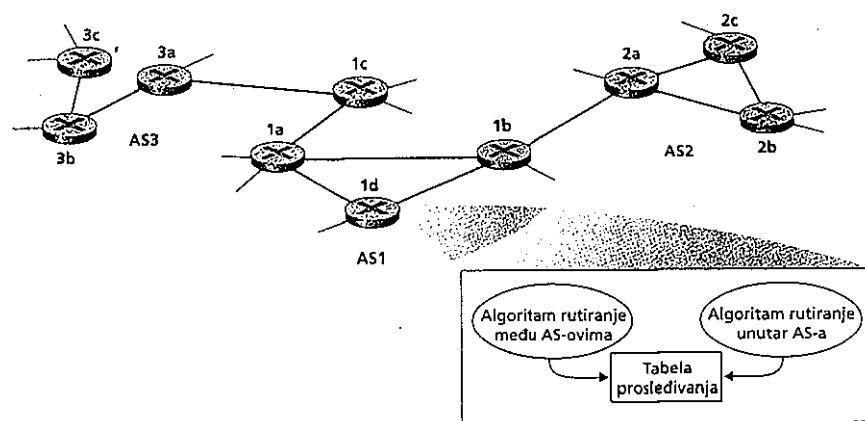
Oba ova problema mogu da se rese tako što se ruteri organizuju u autonomne sisteme (AS) i to tako što je svaki autonomni sistem koji se sastoji od grupe ruteru obično pod zajedničkom administrativnom kontrolom (npr. imaju istog posrednika za Internet usluge ili pripadaju istoj organizaciji). Svi ruteri u jednom AS-u izvršavaju isti algoritam rutiranja (na primer, algoritam LS ili DV) i imaju informacije jedni o drugima - tačno kao u našem idealizovanom modelu iz prethodnog odeljka. Algoritam rutiranja koji se izvršava unutar autonomnog sistema nazivamo unutrašnji protokol rutiranja autonomnog sistema. Biće naravno potrebno povezati AS-ove međusobno, pa će jedan ili više ruteru u AS-u imati dodatni zadatci da prosleđuje pakete na odredišta van AS-a; ti ruteri se nazivaju ruterima mrežnog prolaza.

Na slici 4.29 prikazan je jednostavan primer sa tri autonomna sistema - AS1, AS2 i AS3. Na ovoj slici, deblje linije predstavljaju direktnе linkove između parova ruteru. Tanje linije koje se spuštaju od ruteru predstavljaju podmrežu direktno povezane sa ruterima. AS1 ima četiri ruteru - la, lb, lc i ld - koji izvršavaju unutrašnji protokol rutiranja autonomnog sistema AS1. Prema tome, svaki od ova četiri ruteru zna kako da rutira pakete po optimalnoj putanji do svakog odredišta u autonomnom sistemu AS1. Slično tome, autonomni sistemi AS2 i AS3 imaju svaki po tri ruteru. Obratite pažnju na to da protokoli rutiranja unutar AS-ova koji se izvršavaju u sistemima AS1, AS2 i AS3 ne moraju biti isti. Obratite takođe pažnju na to da su ruteri lb, lc, 2a i 3a ruteri mrežnog prolaza,

Sada bi trebalo da je jasno kako ruteri u AS-ovima utvrđuju putanje rutiranja za parove izvor-odredište unutar autonomnog sistema. Ali još uvek nedostaje jedan veliki deo misterije rutiranja sa kraja na kraj. Kako će ruter unutar jednog AS-a znati kako da rutira paket na odredište izvan tog AS-a? Problem nije težak ako AS ima samo jedan mrežni prolaz koji se povezuje sa samo jednim drugim AS-om. U tom slučaju, postoje unutrašnji algoritam rutiranja utvrđio najjeftiniju putanju od svakog

unutrašnjeg ruteru do mrežnog prolaza, svaki unutrašnji ruter zna kako da prosledi paket. Kada primi paket, ruter mrežnog prolaza prosleduje paket na onaj link koji vodi iz autonomnog sistema. Autonomni sistem na suprotnoj strani tog linka zatim preuzima odgovornost za rutiranje paketa prema njegovom konačnom odredištu. Kao primer, prepostavite da ruter 2b sa slike 4.29 primi paket čije se odredište nalazi izvan sistema AS2, Ruter 2b će tada proslediti paket bilo ruteru 2a ili ruteru 2c, što zavisi od tabele prosleđivanja u ruteru 2b koja je konfigurisana unutrašnjim protokolom rutiranja autonomnog sistema AS2. Paket će na kraju stići do ruteru mrežnog prolaza 2a, koji će proslediti paket ruteru 1b. Pošto paket napusti ruter 2a, autonomni sistem AS2 više nema ništa sa ovim paketom.

Znači pitanje nije teško ako izvorni AS ima samo jedan link koji vodi van AS-a. Ali šta ako izvorni AS ima dva ili više linkova (kroz jedan ili više mrežnih prolaza) koji vode van AS-a? Tada je mnogo teže utvrditi gde treba da se prosledi paket. Na primer, razmotrite ruter iz sistema AS 1 i prepostavite da on primi paket čije se odredište nalazi izvan AS-a. Jasno je da ruter treba da prosledi paket nekom od dva mrežna prolaza, 1b ili 1c, ali kojem? Da bi se resio ovaj problem, AS 1 mora da (1) sazna koja su odredišta dostupna putem autonomnog sistema AS2 a koja putem sistema AS3 i (2) da propagira te informacije o dostupnosti svim ruterima u AS1 da bi svaki ruter konfigurisao svoju tabelu prosleđivanja i pravilno tretirao odredišta van AS-a. Ova dva zadatka - pribavljanje informacija o dostupnosti od susednih AS-ova i propagiranje informacija o dostupnosti svim ruterima unutar AS-a - obavlja protokol rutiranja.



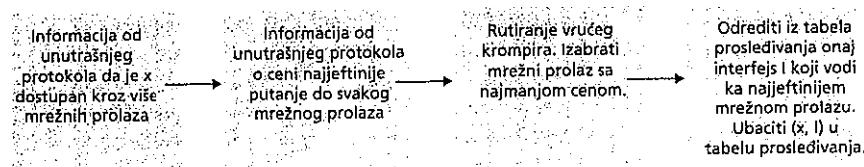
Slika 4.29 ♦ Primer međusobno povezanih autonomnih sistema

**ranja medu autonomnim sistemima.** Pošto protokol rutiranja medu autonomnim sistemima uključuje komunikaciju između dva autonomna sistema, ta dva autonomna sistema moraju da izvršavaju isti protokol rutiranja medu autonomnim sistemima. U stvari, na Internetu svi AS-ovi izvršavaju isti protokol rutiranja medu autonomnim sistemima, po imenu BGP4, koji opisuјemo u sledećem odeljku. Kao što se vidi na slici 4.29, svaki ruter prima informacije od jednog unutrašnjeg protokola rutiranja autonomnog sistema i jednog protokola rutiranja medu autonomnim sistemima i konfiguriše svoju tabelu prosleđivanja pomoću informacija iz oba ta protokola.

Uzmimo kao primer jednu podmrežu  $x$  (sa odredenom CIDR adresom) i uzimimo da AS1 sazna od protokola rutiranja medu autonomnim sistemima da je podmreža  $x$  dostupna iz sistema AS3 ali da nije dostupna iz sistema AS2. AS1 zatim propagira ovu informaciju svim svojim ruterima. Kada ruter Id sazna daje mrežu  $x$  dostupna iz sistema AS3, pa prema tome preko mrežnog prolaza 1c, on tada iz informacija do-bijenih od unutrašnjeg protokola rutiranja određuje ruterski interfejs na najjeftinijej putanji iz ruteru Id do ruteru mrežnog prolaza 1c. Recimo da je to interfejs  $I$ . Ruter Id može u svoju tabelu prosleđivanja da doda stavku ( $x, I$ ). (Ovaj primer, kao i ostali iz ovog Odeljka, imaju za cilj da objasne glavne principe ali uprošćeno prikazuju ono što se zaista događa na Internetu. U sledećem odeljku daćemo detaljniji, mada složeniji, opis protokola BGP.)

Nađovezujući se na prethodni primer, prepostavimo sada da se AS2 i AS3 povezuju sa drugim AS-ovima koji nisu prikazani na slici. Prepostavimo takođe da AS 1 sazna od protokola rutiranja medu autonomnim sistemima da je podmreža  $x$  dostupna i iz sistema AS2 preko mrežnog prolaza 1b i iz sistema AS3 preko mrežnog prolaza 1c. AS1 bi zatim propagirao ovu informaciju svim svojim ruterima uključujući i Id. Da bi konfigurisao svoju tabelu prosleđivanja, ruter Id bi trebalo da odredi u koji ruter mrežnog prolaza (1b ili 1c) treba da usmeri pakete sa odredištem u  $x$ . Jedan pristup, koji se često koristi u praksi, je da se primeni rutiranje vrućeg krompira. U rutiranju vrućeg krompira ruter pokušava da se oslobodi paketa (vrućeg krompira) što pre može (tačnije, što jeftinije). To se postiže slanjem paketa onom ruteru mrežnog prolaza koji od svih mrežnih prolaza koji imaju putanju prema odredištu imaju najnižu cenu od ruteru do mrežnog prolaza. U kontekstu trenutnog primera, rutiranje vrućeg krompira koje bi se izvršavalo u prolazu Id bi na osnovu informacija dobijenih od unutrašnjeg protokola rutiranja utvrdio cene putanja do 1b i 1c i zatim izabralo jeftiniju od te dve putanje. Pošto se izabere putanja, ruter Id dodaje stavku za podmrežu  $x$  u tabelu prosleđivanja. Na slici 4.30 prikazana je rekapitulacija aktivnosti u ruteru Id potrebnih za dodavanje nove stavke za  $x$  u tabelu prosleđivanja.

Kada AS sazna od susednog AS-a za neko odredište, on može da oglasi tu informaciju za rutiranje nekim od svojih ostalih susednih AS-ova. Na primer, prepostavimo da AS 1 sazna od sistema AS2 da je podmreža  $x$  dostupna preko sistema AS2. AS 1 bi mogao da obavesti AS3 da je  $x$  dostupna preko sistema AS 1. Na ovaj način, ako AS3 treba da rutira paket sa odredištem u  $x$ , AS3 bi prosledio paket u AS1, koji bi zatim prosledio paket u AS2. Kao što ćemo videti u opisu BGP-a, AS prilično fle-



**Slika 4.30** ♦ Koraci u dodavanju odredišta izvan AS-a u ruterovu tabelu prosleđivanja

ksibilno odlučuje o tome da li da obavesti susedne AS-ove o nekom odredištu. To su odluke *politike*, koje zavise više od ekonomskih nego od tehničkih argumenata.

Iz odeljka 1.5 se možda sećate da se Internet sastoji od hijerarhije međusobno povezanih posrednika za Internet usluge. Kakav je dakle odnos posrednika i AS-a? Mogli biste pomisliti da ruteri jednog posrednika i linkovi koji ih povezuju čine jedan AS. Mada je često tako, mnogi posrednici dele svoju mrežu na više AS-ova. Na primer, neki posrednici prvog reda koriste jedan AS za celu svoju mrežu dok je drugi dele desetine međusobno povezanih AS-ova.

Sve u svemu, problemi skaliranja i administrativne kontrole rešavaju se definisanjem autonomnih sistema. Unutar autonomog sistema svi ruteri izvršavaju isti unutrašnji protokol rutiranja. Autonomni sistemi među sobom izvršavaju isti protokol rutiranja među autonomnim sistemima. Problem skaliranja je rešen pošto unutrašnji protokol rutiranja treba da zna samo za rutere u svom autonomnom sistemu. Problem administrativne kontrole je rešen pošto svaka organizacija može da izvršava unutrašnji protokol koji god hoće; međutim, svaki par povezanih AS-ova mora da izvršava isti protokol rutiranja među autonomnim sistemima kako bi mogli da razmenjuju informacije o dostupnosti.

U sledećem odeljku ispitacemo dva unutrašnja protokola rutiranja (RIP i OSPF) i jedan protokol rutiranja među autonomnim sistemima (BGP) koji se koriste u današnjem Internetu. Ovi primeri će lepo upotpuniti naša razmatranje hijerarhijskog rutiranja.

## 4.6 Rutiranje na Internetu

Pošto smo proučili internetsko adresiranje i protokol IP, sada ćemo se posvetiti Internetovim protokolima rutiranja; njihov zadat je da utvrde putanje kojom će datagram stići od izvora do odredišta. Videćemo da Internetovi protokoli rutiranja primenjuju mnoge principe koje smo proučili ranije u ovom poglavljiju. Za način rutiranja na današnjem Internetu ključni su algoritmi stanja linkova i vektora rastojanja koje smo proučili u odeljcima 4.5.1 i 4.5.2 i pojma autonomnih sistema (AS) koji smo razmatrali u odeljku 4.5.3.

Verovatno se sećate iz odeljka 4.5.3 da se kolekcija ruteru pod istom administrativnom i tehničkom kontrolom i koja koristi isti zajednički protokol rutiranja naziva autonomnim sistemom (AS). Svaki AS, zatim, obično sadrži više podmreža (ovde koristimo izraz podmreža u preciznom adresnom smislu iz odeljka 4.4.2).

### 4.6.1 Unutrašnji protokoli rutiranja autonomnih sistema na Internetu: RIP

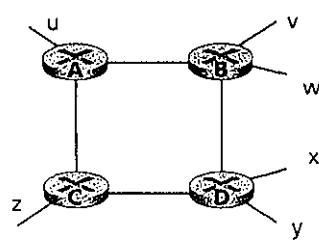
Unutrašnji protokol rutiranja AS-a služi da bi se odredilo kako se vrši rutiranje unutar tog autonomnog sistema. Unutrašnji protokoli rutiranja AS-a poznati su i kao unutrašnji protokoli mrežnih prolaza. Na Internetu su za rutiranje unutar autonomnih sistema bila u Širokoj upotrebi dva protokola rutiranja: najpre RIP (*Routing Information Protokol*) i zatim OSPF (*Open Shortest Path First*). Protokol rutiranja blisko povezan sa OSPF-om je protokol IS-IS [RFC 1142, Periman 1999]. Prvo opisujemo RIP a zatim OSPF.

RIP je bio jedan od prvih unutrašnjih protokola rutiranja AS-a na Internetu i danas je još uvek u širokoj upotrebi. Njegovi koren i ime potiču iz arhitekture XNS (*Xerox Nenvork Svstems*). RIP je toliko rasprostranjen pre svega zato što je bio uključen u verziju UNIX-a koji je podržavao TCP/IP i koji je distribuiran u paketu BSD (*Berkeley Sopware Distribution*) 1982. godine. Prva verzija RIP-a definisana je u dokumentu RFC 1058, a druga verzija, kompatibilna unazad, definisana je u dokumentu RFC 2453.

RIP je protokol vektora rastojanja koji funkcioniše veoma slično idealizovanom protokolu koji smo opisali u odeljku 4.5.2. Verzija RIP-a opisana u dokumentu RFC 1058 određuje cene prema broju skokova; tj. svaki link ima cenu 1. U algoritmu DV u odeljku 4.5.2 smo zbog jednostavnosti navodili cene između parova ruteru. U RIP-u (a takođe i u OSPF-u) cene se navode od izvornog ruteru do neke odredišne podmreže. RIP koristi izraz *skok* što je u stvari broj podmreža kroz koje se prolazi duž najkratče putanje od izvornog ruteru do odredišne podmreže, uključujući odredišnu podmrežu. Na slici 4.31 prikazan je AS sa šest podmreža grana. Tabela na stici prikazuje broj skokova od izvora A do svih podmreža grana.

Maksimalna cena putanje ograničena je na 15, pa se zato RIP ograničava na autonomne sisteme sa manje od 15 skokova u prečniku. Verovatno se sećate da kod protokola vektora rastojanja, susedni ruteri međusobno razmenjuju informacije o rutiranju. Vektor rastojanja u svakom ruteru sadrži trenutno procenjena najmanja rastojanja od tog ruteru do podmreža u AS-u. U RIP-u, ažurne podatke o rutiranju susedi razmenjuju približno svakih 30 sekundi u takozvanim porukama RIP odgovora. Poruka odgovora koju šalje ruter ili računar sadrži spisak od najviše 25 odredišnih podmreža unutar AS-a kao i udaljenost pošiljaoca do svake od tih podmreža. Poruke odgovora nazivaju se takođe i RIP objave.

Pogledajmo jedan jednostavan primer gde ćemo videti kako funkcionišu RIP objave. Razmotrite deo AS-a prikazan na slici 4.32. Na ovoj slici linije koje povezuju rutere predstavljaju podmreže. Radi bolje preglednosti označeni su samo izabrani

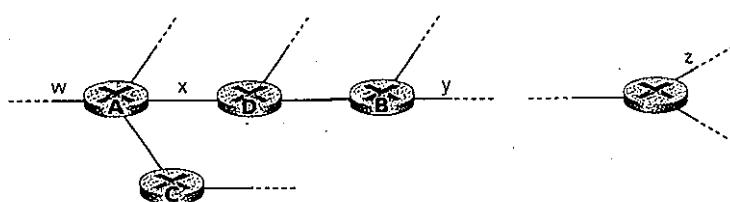


| Odredište | Broj skokova |
|-----------|--------------|
| u         | 1            |
| v         | 2            |
| w         | 2            |
| x         | 3            |
| y         | 3            |
| z         | 2            |

Slika 4.31 ♦ Broj skokova od izvornog ruteru A do različitih podmreža

ruteri (*A, B, C i D*) i podmreže (*w, x, y i z*). Isprekidane linije ukazuju na to da se AS produžava i da u njemu ima mnogo više ruter i linkova nego stoje prikazano,

Svaki ruter održava RIP tabelu koja se naziva tabela rutiranja. Tabela rutiranja jednog ruteru sadrži njegov vektor rastojanja i njegovu tabelu prosleđivanja. Na slici 4.33 prikazana je tabela rutiranja u ruteru *D*. Obratite pažnju na to da tabela rutiranja ima tri kolone. Prva kolona je za odredišnu podmrežu, druga kolona sadrži sledeći ruter na najkraćoj putanji prema odredišnoj mreži, a u trećoj se nalazi broj skokova (tj. broj podmreža koje treba preći, uključujući odredišnu podmrežu) da bi se najkraćim putem stiglo u odredišnu podmrežu. U ovom primeru tabela pokazuje da za slanje datagrama iz ruteru *D* u odredišnu podmrežu *w*, datagram treba prvo proslediti susednom ruteru *A*; tabela takođe pokazuje da do odredišne podmreže *w* po najkraćoj putanji treba dva skoka. Slično tome, tabela pokazuje daje podmreža z udaljena sedam skokova i to preko ruteru *B*. U principu, tabela rutiranja ima po jedan red za svaku podmrežu unutar AS-a, mada RIP verzije 2 dozvoljava agregaciju ruta prema podmrežama pomoću tehnika agregacije kakve smo ispitali u odeljku 4.4. Tabela na slici 4.33 i sledeće tabele su zbog toga samo delimično popunjene.



Slika 4.32 ♦ Deo jednog autonomnog sistema

| Odredišna mreža | Sledeći ruter | Broj skokova do odredišta |
|-----------------|---------------|---------------------------|
| w               | A             | 2                         |
| y               | B             | 2                         |
| z               | B             | 7                         |
| x               | —             | 1                         |
| ...             | ...           | ...                       |

Slika 4.33 ♦ Tabela prosleđivanja u ruteru *D* pre primanja objave od ruteru *A*

Prepostavimo sada da nakon 30 sekundi ruter *D* primi od ruteru *A* objavu prikazanu na slici 4.34. Obratite pažnju na to da ova objava sadrži samo informacije iz tabele rutiranja ruteru *A*! Ova informacija kazuje, konkretno, daje podmreža z udaljena samo 4 skoka od ruteru *A*. Kada ruter *D* primi ovu objavu, on kombinuje objavu (slika 4.34) sa starom tabelom rutiranja (slika 4.33). Konkretno, ruter *D* saznaće da do podmreže z sada postoji putanja preko ruteru *A* kraća od putanje preko ruteru *B*. Prema tome, ruter *D* ažurira svoju tabelu rutiranja i uključuje kraću najkraću putanju, kao što je prikazano na slici 4.35. Možda se pitate kako to daje najkraću putanju do podmreže *z* postala kraća? Možda se decentralizovani algoritam vektora rastojanja tek izračunavao (videti odeljak 4.5.2) ili su možda u AS dodati novi linkovi i/ili ruteri pa su se tako promenile najkraće putanje u AS-u.

Pogledajmo zatim nekoliko aspekata u implementaciji RIP-a. Sigurno se sećate da RIP ruteri razmenjuju objave približno svakih 30 sekundi. Ako ruter nema vesti od svog suseda bar svakih 180 sekundi, smatraće da taj sused više nije dostupan; ili je sused otkašao ili se prekinuo link koji ih povezuje. U tom slučaju, RJP menja lokalnu tabelu rutiranja i zatim propagira tu informaciju tako što šalje objave svojim suse-dnim ruterima (onima koji su još uvek dostupni). Ruter može takođe od svog suseda

| Odredišna mreža | Sledeći ruter | Broj skokova do odredišta |
|-----------------|---------------|---------------------------|
| z               | C             | 4                         |
| w               | —             | 1                         |
| x               | —             | 1                         |
| ...             | ...           | ...                       |

Slika 4.34 ♦ Objava iz ruteru A

| Odredišno mreža | Sledeći ruter | Broj skokova do odredišta |
|-----------------|---------------|---------------------------|
| W               | A             | 2                         |
| Y               | B             | 2                         |
| Z               | A             | 5                         |
| ....            | ....          | ....                      |

**Slika 4.35** ♦ Tabela prosleđivanja u ruteru D nakon primanja objave od ruteru A

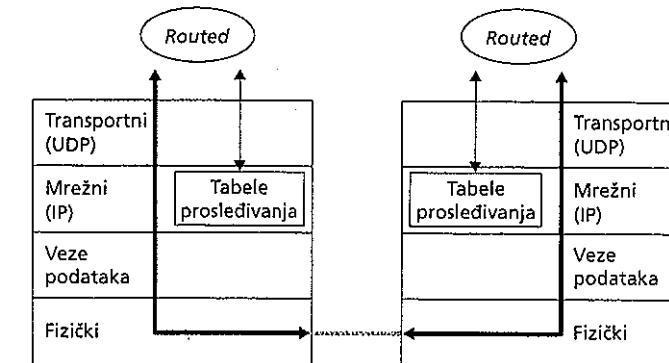
da zahteva informaciju o ceni do datog odredišta pomoću poruke RIP zahteva. Ruteri šalju jedni drugima poruke RIP zahteva i RIP odgovora preko UDP-a i pri tom koriste broj porta 520. UDP paket se između ruta prenosi u standardnom IP datagramu. Činjenica da RIP koristi protokol transportnog sloja (UDP) preko protokola mrežnog sloja (IP) da bi implementirao funkciju mrežnog sloja (algoritam rutiranja) može da izgleda prilično uvrnuto (i jeste!). Kada malo detaljnije razmotrimo način na koji se RIP implementira biće i ovo jasnije.

Na slici 4.36 data je skica uobičajenog implementiranja RIP-a u UNIX sistemu, na primer, u UNIX radnoj stanicu koja služi kao ruter. Proses po imenu routed (koji se izgovara „rut di“) izvršava protokol RIP, tj. održava informacije o rutiranju i razme-njuje poruke sa routed procesima koji se izvršavaju u susednim rutaferima. Pošto je RIP implementiran kao proces aplikacijskog sloja (iako veoma poseban proces koji može da manipuliše tabelama rutiranja u UNIX-ovom kemu), on može da šalje i prima poruke preko standardnog soketa i koristi standardni transportni protokol. Prema tome, RIP je implementiran kao protokol aplikacijskog sloja (pročitati poglavlje 2) koji se izvršava preko UDP-a.

## 4.6.2 Unutrašnji protokoli rutiranja autonomnih sistema na Internetu: OSPF

Kao i RIP, OSPF (otvorena prvo najkraća putanja) rutiranje se koristi unutar AS-ova. OSPF i njegov blizak srodnik IS-IS obično se uvodi kod posrednika Internet usluga višeg reda dok se RIP uvodi kod posrednika nižeg reda i u mrežama preduzeća. „Otvorena“ u skraćenici OSPF znači da je specifikacija protokola rutiranja javno dostupna (za razliku, na primer, od Ciscovog protokola EIGRP). Najnovija verzija OSPF-a, verzija 2 definisana je u javnom dokumentu RFC 2328.

OSPF je zamišljen kao naslednik RIP-a, pa kao takav ima niz naprednih svojstava. U suštini, međutim, OSPF je protokol sa stanjem linkova koji koristi priliv in-



**Slika 4.36** ♦ Implementiranje RIP-a kao daemona routed

formacija o stanju linkova i jedan Dijkstrin algoritam za najjeftiniju putanju. Kada se koristi OSPF, ruter izgrađuje kompletну topološku mapu (tj. orientisani graf) celog autonomnog sistema. Ruter zatim lokalno izvršava Dijkstrin algoritam za najkraću putanju kako bi utvrdio stablo najkraćih putanja prema svim podmrežama i pri tom sebe postavlja za osnovni čvor. Mrežni administrator konfiguriše cene pojedinačnih linkova (pročitajte Principe u praksi: Podešavanje pondera OSPF linkova). Administrator može svim linkovima da stavi cenu 1, čime postiže rutiranje prema minimalnom broju skokova a može i da ponderiše linkove tako da im cene budu obrnuto srazmene kapacitetu da bi podstakao saobraćaj na linkovima sa većim propusnim opsegom. OSPF ne propisuje politiku za ponderisanje linkova (to je zadatak mrežnog administratora), već umesto toga obezbeđuje mehanizme (protokol) kojim se utvrđuje rutiranje najjeftinijih putanja na osnovu pondera za linkove.

Kada se koristi OSPF, ruter difuzno emituje informacije o rutiranju *svim* ostalim ruterima u autonomnom sistemu, a ne samo svojim susednim ruterima. Ruter difuzno emituje informacije o stanju linkova kad god dode do promene stanja nekog linka (na primer, do promene cene ili promene statusa uključenja/isključenja). On takođe povremeno difuzno emituje stanje linkova (najmanje svakih 30 minuta). Čak i kad nema promena u stanju linkova. RFC 2328 napominje da „povremeno ažuriranje objava stanja linkova povećava robustnost ovog algoritma“. OSPF objave stavljuju se u OSPF poruke koje direktno prenosi IP sa protokolom gornjeg sloja 89, za OSPF. Prema tome, protokol OSPF mora sam da implementira funkcije kao što su pouzdani transfer poruka i difuzno emitovanje stanja linkova. Protokol OSPF takođe proverava da li linkovi funkcionišu (pomoću poruke HELLO koja se šalje povezanom suse-du) i omogućava OSPF ruteru da preko svog susednog ruteru pribavi bazu podataka o stanju linkova u celoj mreži.

Evo nekoliko unapredjenja koje donosi OSPF:

- ◆ **Bezbednost.** Proverava se autentičnost kompletne razmene između OSPF rutera (na primer, ažuriranje stanja linkova). To znači da u OSPF protokolu unutar domena mogu da učestvuju samo ruteri od poverenja, čime se sprečava da zlo-namemi uljezi (ili studenti umrežavanja koji svoje novostećeno znanje koriste za šale) ubace netačne informacije u routerske tabele. OSPF paketi među ruterima se podrazumevano ne proveravaju pa bi moglo doći do falsifikovanja. Moguće je konfigurisati dve vrste bezbednosti - jednostavnu ili MD5 (u poglavljju 8 možete naći opšti opis provere autentičnosti i posebno MD5). Ako se koristi jednostavna provera autentičnosti u sve ruterse se konfiguriše ista lozinka. Kada ruter pošalje OSPF paket, on uključuje lozinku kao čisti tekst. Jasno je da jednostavna provera autentičnosti nije bezbedna. Provera autentičnosti MD5 zasniva se na zajedničkim tajnim ključevima koji se konfigurišu u sve ruterse. Svaki ruter izračunava MD5 heš za svaki OSPF paket na osnovu sadržaja paketa i konfigurisanog tajnog ključa. Dobijenu vrednost heša uključuje u OSPF paket. Prijemni ruter će pomoći unapred konfigurisanog ključa izračunati MD5 heš primljenog paketa i uporediti dobijeni rezultat sa heš vrednošću iz paketa. Čime proverava autentičnost paketa. Kao zaštita od napada ponavljanjem koriste se i redni brojevi u MD5 proveri autentičnosti.
- ◆ **Više putanja sa istom cenzom.** Kada do jednog odredišta više putanja ima istu cenu, OSPF dozvoljava da se i koristi više putanja (tj. ako postoji više putanja sa istom cenzom, ne mora se za sav saobraćaj birati ista putanja).
- ◆ **Integrirana podrška za jednoznačno i više značno rutiranje.** Više značni OSPF [RFC 1584] sadrži jednostavna proširenja za OSPF koja omogućavaju više značno rutiranje (ova tema je detaljnije obrađena u odeljku 4.7.2). Više značni OSPF (*Multicast OSPF*, MOSPF) koristi postojeću OSPF bazu podataka o linkovima i dodaje jednu novu vrstu objave stanja linkova u postojeći mehanizam za difuzno emitovanje stanja linkova.
- ◆ **Podrška za hijerarhiju unutar istog domena rutiranja.** Možda je najznačajniji napredak u OSPF-u mogućnost da se autonomni sistem strukturiše hijerarhijski. U odeljku 4.5.3 već smo uvideli mnoge prednosti hijerarhijskih struktura za rutiranje. U ostaku ovog odeljka obradićemo implementaciju OSPF hijerarhijskog rutiranja.

OSPF autonomni sistem može se konfigurisati u područja. Svako područje izvršava vlastiti OSPF algoritam rutiranja prema stanju linkova, tako što svaki ruter u području difuzno emituje svoje stanje linkova svim ostalim ruterima u tom području. Unutrašnji detalji područja tako ostaju nevidljivi za sve ruterse izvan područja. Rutiranje unutar područja uključuje samo ruterse iz istog područja.

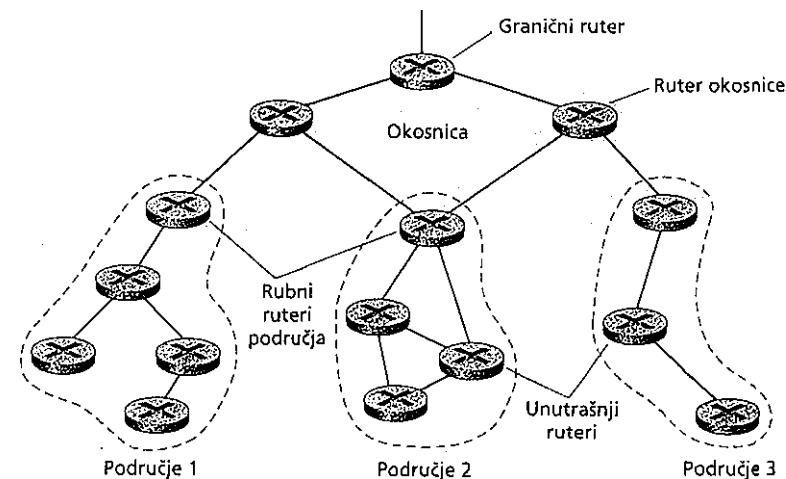
U svakom području je jedan rubni ruter područja ili više njih zaduženo za rutiranje paketa van područja. Tačno jedno OSPF područje u autonomnom sistemu konfiguriše se kao područje okosnice. Prvenstvena uloga područja okosnice je da rutira saobraćaj između ostalih područja u autonomnom sistemu. Okosnica uvek sadrži

sve rubne ruterse područja u autonomnom sistemu, a može da sadrži i ruterse koji nisu rubni. Rutiranje među područjima unutar autonomnog sistema zahteva da se paket prvo rutira rubnom ruteru područja (rutiranje unutar područja), zatim kroz okosnicu do rubnog ruteru područja koji se nalazi u odredišnom području, a tek nakon toga do konačnog odredišta.

Na slici 4.35 dat je dijagram hijerarhijski strukturisane OSPF mreže. Na ovoj slici vidimo četiri vrste OSPF rutera.

- ◆ **Unutrašnji ruteri.** Ovi ruteri su u područjima koja ne spadaju u okosnicu i izvršavaju samo rutiranje unutar autonomnog sistema.
- ◆ **Rubni ruteri područja.** Ovi ruteri pripadaju i području i okosnici.
- ◆ **Ruteri okosnice /ruteri koji nisu J-ubni.** Ovi ruteri vrše rutiranje unutar okosnice, ali sami nisu rubni ruteri područja. Unutrašnji ruteri saznavaju da postoje rute prema drugim područjima van okosnice na osnovu informacija (u suštini objave o stanju linka koja, međutim, objavljuje cenu rute prema drugom području, a ne cenu linka) difuzno emitovanih unutar područja iz njegovih ruteru okosnice.
- ◆ **Granični ruteri.** Granični ruter razmenjuje informacije o rutiranju sa ruterima koji pripadaju drugim autonomnim sistemima. Taj ruter može, na primer, da koristi BGP za rutiranje medu AS-ovima. Kroz takav granični ruter ostali ruteri saznavaju za putanje prema spoljnjim mrežama.

OSPF je relativno složen protokol i naš je opis morao da bude skraćen; dodatne detalje možete naći u knjigama [Huitema 1998; Moy 1998; RFC 2328].



**Slika 4.37** ♦ Hijerarhijski strukturisan OSPF AS sa 4 područja

## PRINCIPI U PRAKSI

### PODEŠAVANJE PONDERA OSPF LINKOVA

Nos opis ruliranjem prema slanju linkova polazio je od prepostavke do su ponderi linkova određeni, da se izvršava algoritam ruliranja kao što je OSPF i da saobraćaj teče u skladu sa tabelama rutiranja koje je izračunao algoritam LS. U smislu uzroka i posledice, ponderi linkova su doti (tj. oni su prvi), a iz njih proizlaze (pomoću Dijkstrinog algoritma) putanje rutiranja prema najnižim cenama. Iz te perspektive, ponder linka reflektuje cenu koju treba platiti za korišćenje linka {npr. ako su ponderi linkova obratno proporcionalni kapacitetu, tada bi korišćenje linkova sa velikim kapacitetom imalo niži ponder pa bi bilo privlačniji je što se tiče rutiranja} a Dijkstrin algoritam služi za pronaalaženje što niže ukupne cene.

U praksi su uzročno posledični odnosi između pondera linkova i putanja rutiranja ponekad obrnuti, jer mrežni operateri konfigurisu pondere linkova da bi se dobole putanje rutiranja koje će ostvariti određene ciljeve saobraćajnih inženjera [Forlz 2000, Fortz 2002]. Na primer, prepostavite da mrežni operater procenu toku saobraćaja koji ulazi u mrežu na svakoj ulaznoj točki sa odredištem na drugoj izlaznoj tački. Operater bi mogao da uredi specifično rutiranje tokova od ulaza ka izlazu lako da maksimalno opterećenje svih linkova u mreži bude što manje. Ali sa algoritmom rutiranja kakav je OSPF, ponderi linkova predstavljaju glavne operaterove „ručice“ za usmeravanje tokova kroz mrežu. Prema tome, da bi postigao svoj cilj da maksimalno opterećenje svih linkova u mreži bude što manje, opeialer mora do pronade skup pondera zo linkove koji će laj cilj i ostvariti. Ovo je suprotan uzročno posledični odnos - poznalo je željeno rutiranje tokova a treba pronaći takve pondere OSPF linkova do OSPF algoritam rutiranja proizvede željeno rutiranje tokova.

### 4.6.3 Rutiranje među autonomnim sistemima: BGP

Upravo smo naučili kako posrednici za Internet usluge koriste RIP i OSPF za utvrđivanje optimalnih putanja za parove izvor - odredište unutar istog AS-a. Razmotrimo sada kako se utvrđuju optimalne putanje za parove izvor - odredište iz različitih AS-ova. Protokol **BGP** (*Border Gateway Protocol*) verzija 4, definisan u RFC-u 1771 (videli takođe [RFC 1772; RFC 1773]), je u današnjem Internetu *defacto* standardni protokol za rutiranje medju domenima, odnosno medju autonomnim sistemima. Obično se pominje kao BGP4 ili jednostavno kao **BGP**. Kao protokol rutiranja medju autonomnim sistemima (videti odeljak 4.5.3), BGP svakom AS-u omogućava da:

1. pribavi od susednih AS-ova informacije o dostupnosti podmreža;
2. propagira informacije o dostupnosti svim ruterima unutar AS-a;
3. odredi „dobre“ rute do podmreža na osnovu informacija o dostupnosti i politike AS-a.

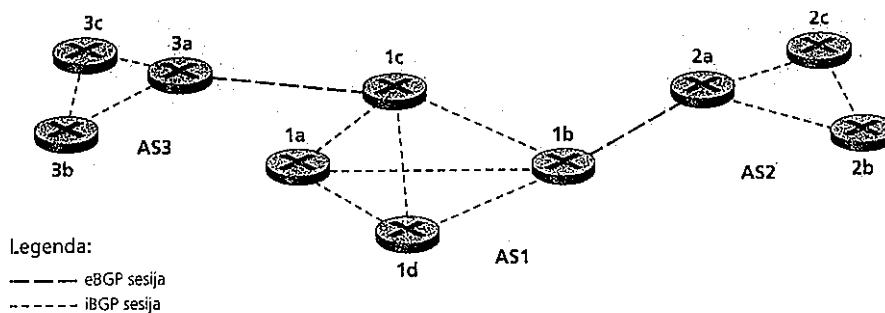
Pogotovo, BGP omogućava svakoj podmreži da oglosi svoje postojanja svima na Internetu. Podmreža viće „Ja postojim, ja sam ovde“, a BGP će se pobrinuti da svaki AS na Internetu sazna za tu podmrežu i kako se do nje stiže. Da nema BGP-a, svaka podmreža bi ostala izolovana - sama i nepoznata ostalima na Internetu.

### Osnove BGP-a

BGP je izuzetno složen; cele knjige su posvećene toj temi. Štaviše, čak pošto se pročitaju te knjige i RFC-iji teško je potpuno ovladati BGP-om pre nego što se mese-cima (ako ne godinama) praktično koristi BGP u ulozi projektanta ili administratora kod posrednika Internet usluga višeg reda. I pored toga, pošto je BGP apsolutno suštinski protokol na Internetu - to je upravo protokol koji drži celu stvar na okupu - moramo da naučimo bar osnove njegovog funkcionisanja. Na početku opisujemo kako bi BGP mogao da radi u kontekstu primera jednostavne mreže sa slike 4.29. Ovim opisom nadovezujemo se na obradu hijerarhijskog rutiranja iz odeljka 4.5.3; bilo bi dobro da ga ponovo pročitate.

Kada se koristi BGP, parovi ruteru razmenjuju informacije o rutiranju putem polutrajnih TCP konekcija na portu 179. Polutrajne TCP konekcije za mrežu sa slike 4.29 prikazane su na slici 4.38. Obično postoji po jedna takva BGP TCP konekcija za svaki link koji direktno povezuje dva ruteru iz različitih AS-ova; prema tome, na slici 4.38 imamo jednu TCP konekciju između nitera mrežnih prolaza 3a i 1c i drugu TCP konekciju između ruteru mrežnih prolaza 1b i 2a. Postoje takođe polutrajne BGP TCP konekcije između ruteru unutar administrativnog sistema. Konkretno, na slici 4.38 prikazana je uobičajena konfiguracija jedne TCP konekcije za svaki par ruteru unutar AS-a tako da imamo preplet TCP konekcija u svakom AS-u. Za svaku TCP konekciju dva ruteru na krajevima konekcije nazivamo **BGP ravnopravnim** učesnikom, a TCP konekcija sa svim BGP porukama koje se šalju po njoj je **BGP sesija**. Osim toga, BGP sesija koja povezuje dva AS-a naziva se **eksternom BGP (e-BGP) sesijom**, a BGP sesija između ruteru iz istog AS-a naziva se **internom BGP (iBGP) sesijom**. Na slici 4.38 eBGPsesije su prikazane dužim crticama; iBGP sesije su prikazane kraćim crticama. Obratite pažnju na to da linije BGP sesija na slici 4.38 ne odgovaraju uvek fizičkim linkovima na slici 4.29.

BGP dozvoljava da svaki AS sazna koja su određišta dostupna preko njegovih susednih AS-ova. Kada se koristi BGP, određišta nisu računari već CIDR **prefksi**, gde svaki prefiks predstavlja podmrežu ili kolekciju podmreža. Tako, na primer, prepostavimo da su za AS2 vezane Četiri podmreže: 138.16.64/24, 138.16.65/24, 138.16.66/24 i 138.16.67/24. AS2 bi tada mogao da sakupi prefiks te četiri podmreže i upotrebi BGP za objavu jedinstvenog prefiksa 138.16.64/22 autonomnom sistemu AS 1. Uzmimo kao drugi primer prepostavku da se samo prve tri podmreže nalaze u AS-u2 a daje četvrta, 138.16.67/24, u AS-u3. Tada, kao sto je objašnjeno u tekstu pod naslovom Principi u praksi, odeljka 4.4.2, pošto ruteri za prosledivanje datagrama koriste pravilo najdužeg prefiksa, AS3 bi mogao da dojavi AS-ul konkretniji prefiks 138.16.67/24, a AS2 bi *ipak* objavio zbirni prefiks 138.16.64/22. Svaku



Slika 4.38 ♦ eBGP i iBGP sesije

objavu prefiksa možemo da posmatramo kao obećanje. Kada jedan AS objavi neki prefiks drugom AS-u, on mu obećava da će proslediti datagram sa odredištem u tom prefiksu po putanji prema tom prefiksu.

Ispitajmo sada kako bi BGP distribuirao informacije o dostupnosti prefiksa preko BGP sesija prikazanih na slici 4.38. Kao što bi se očekivalo, pomoću eBGP sesije između mrežnih prolaza 3a i 1c AS3 šalje u AS 1 listu prefiksa dostupnih iz AS-a3; a AS1 šalje AS3 listu prefiksa dostupnih iz AS-al. Slično tome, AS1 i AS2 razme-njuju informacije o dostupnosti prefiksa preko svojih ruteru mrežnog prolaza 1b i 2a. Takođe prema očekivanju, kada ruter mrežnog prolaza (u bilo kom AS-u) primi prefikse koji stignu preko eBGP-a, on pomoću svoje iBGP sesije distribuiru prefikse ostalim ruterima u AS-u. Na taj način, ne samo da će ruteri iz AS-al van eBGP sesije saznati za prefikse iz AS-a3, već će za njih saznati i ruter mrežnog prolaza 1b. Ruter mrežnog prolaza 1b (iz AS-al) može zatim da objavi autonomnom sistemu AS2 prefikse iz AS-a3. Kada ruter (bez obzira na to da li je ruter mrežnog prolaza ili nije) sazna za neki novi prefiks, on u svojoj tabeli prosleđivanja napravi stavku za taj prefiks kao što je opisano u odeljku 4.5.3.

#### Atributi putanja i BGP

Pošto sada imamo uvodna saznanja o BGP-u, krenimo malo dublje (ipak ćemo i dalje gurati pod tepih neke manje važne detalje!). Kada se koristi BGP, autonomni sistem se prepoznaće po svom globalno jedinstvenom broju autonomnog sistema (*autonomous system number*, ASN) [RFC 1930]. (Nije sasvim tačno da svaki AS ima svoj ASN. Konkretno, takozvani završni AS koji prenosi isključivo saobraćaj čiji je on ili izvor ili odredište, obično nema ASN; ovaj detalj ćemo zanemariti u

daljem opisu da nam drvo ne bi zaklonilo šumu.) AS brojevi, kao i IP adrese, dodeljuju se u regionalnim registrima ICANN [ICANN 2002].

Kada ruter kroz BGP sesiju objavljuje neki prefiks, on uz prefiks stavlja i određen broj BGP atributa. U BGP žargonu, prefiks zajedno sa svojim atributima naziva se rutom. Prema tome, ravnopravni BGP učešnici objavljuju jedni drugima rute. Dva najvažnija atributa su AS-PATH i NEXT-HOP:

- ♦ **AS-PATH.** Ovaj atribut sadrži eksplicitan spisak svih AS-ova kroz koje je prošla objava za prefiks o kojem je reč. Kada se prefiks preda u AS, taj AS dodaje svoj ASN u atribut AS-PATH. Na primer, razmotrite sliku 4.38 i prepostavite daje prefiks 138.16.64/24 prvo objavljen iz AS-a2 u AS1; ako AS1 zatim objavi taj prefiks u AS3, atribut AS-PATH bi bio AS2 AS1. Ruteri koriste atribut AS-PATH za otkrivanje i sprečavanje petlji oglašavanja; konkretno, ako ruter vidi da se njegov AS već nalazi na spisku, on će odbaciti objavu. Kao što ćemo uskoro objasniti, ruteri koriste atribut AS-PATH i prilikom biranja jedne od više putanja za isti prefiks.
- ♦ **NEXT-HOP.** Par AS-ova, na primer AS A i AS B, može da ima više fizičkih linkova koji ih direktno povezuju. Zato, kada se prosleđuje paket iz AS-a A u AS B, on bi mogao da se pošalje bilo kojim od ovih linkova koji direktno povezuju AS-ove. Kada ruter mrežnog prolaza iza AS-a B šalje objavu ruteru mrežnom prolazu u AS-u A, on u objavu uključuje svoju IP adresu (konkretni)e, IP adresu interfejsa koji ide do ruteru mrežnog interfejsa u AS-u A). Ruter u AS-u A može putem eBGP-a i iBGP-a da primi više ruta za isti prefiks, a da svaka prolazi kroz drugi ruter prvog skoka. Kada konfiguriše svoju tabelu prosleđivanja, ovaj ruter bi morao da izabere jednu od tih putanja.

BGP takođe uključuje atribut koji omogućavaju ruterima da rutama odrede ponder u izabranim mernim jedinicama i jedan atribut koji označava na koji načinje prefiks stigao BGP u početnom AS-u. Potpuni opis atributa ruta možete naći u [Griffm 2002; Stewart 1999; Halabi 2000; Feamster 2004].

Kada ruter mrežnog prolaza dobije objavu ruteru, on koristi svoju politiku uvoza pri odlučivanju da li da filtrira ili prihvati rutu i da li da podesi odredene atributte kao što su izabrane mene jedinice ruteru. Politika uvoza može da filtrira rutu zato što AS ne želi da šalje saobraćaj preko nekog od AS-ova iz spiska AS-PATH. Ruter mrežnog prolaza može takođe da filtrira rutu zato što već zna za bolju rutu prema istom prefiksu.

#### Izbor BGP ruta

Kao stoje u ovom odeljku već rečeno, BGP koristi eBGP i iBGP za distribuciju ruta svim ruterima unutar AS-a. Iz distribuiranih informacija ruter može da sazna za više ruta prema istom prefiksu pa u tom slučaju mora da se odluči za jednu od mogućih ruta. U ovom postupku selekcije ruter polazi od svih ruta za koje je saznao i koje je

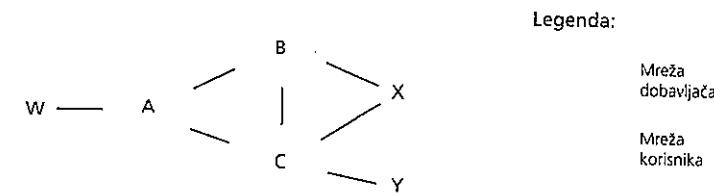
prihvatio. Ako postoje dve ili više ruta do istog prefiksa, BGP redom poziva sledeća pravila eliminacije dok ne ostane samo jedna ruta za dati prefiks,

- ◆ Jedan od atributa svake rute je vrednost lokalnog prioriteta. Lokalni prioritet rute mogao je da odredi ruter ili je mogao da ga sazna od drugog rutera u istom AS-u. Ovo je odluka politike i prepusta se mrežnom administratoru AS-a. (Uskoro ćemo detaljnije opisati pitanja BGP politika.) Biraju se rute sa najvišim vredno-stima lokalnog prioriteta.
- ◆ Od preostalih ruta (svih sa jednakom vrednošću lokalnog prioriteta) bira se ruta sa najkrćim atributom AS-PATH. Daje ovo jedino pravilo za biranje ruta, BGP bi za određivanje putanja koristio algoritam DV, gde je jedinica mera za razdaljinu broj AS skokova a ne broj skokova ruta.
- ◆ Od preostalih ruta (svih sa jednakom vrednošću lokalnog prioriteta i jednakim dužinama atributa AS-PATH) bira se ruta sa najbližim NEXT-HOP ruterom. Ovde se najbližim smatra ruter za koga je utvrđena najmanja cena po najefтинijoj putanji. Ovaj postupak se često naziva rutiranjem vruećeg krompira.
- ◆ Ako još uvek ostaje više ruta, za izbor se koriste BGP identifikatori [Stewart 1999].

Pravila eliminacije su složenija nego Stoji to gore opisano. Da bi se izbegle noćne more o BGP-u, bolje gaje učiti u manjim dozama!

#### Politika rutiranja

Neke od osnovnih koncepata BGP rutiranja ilustrovaćemo jednim jednostavnim pri-merom. Na slici 4.39 prikazano je šest međusobno povezanih autonomnih sistema: A, B, C, W, X i Y. Obratite pažnju na to da su A, B, C, W, X i Y AS-ovi, a ne ruteri. Pretpostavimo da su autonomni sistemi W, X i Y završne mreže, a da su A, B i C mreže posrednika za uslugu okosnice. Sav saobraćaj koji ulazi u završnu mrežu mora imati odredište u toj mreži i sav saobraćaj koji napušta tu mrežu mora da potiče iz nje. Jasno je da su W i Y takve mreže. X je tako zvana višedoma završna mreža, postoje povezana sa ostatkom mreže preko dva različita posrednika (situacija koja je u praksi sve češća). Međutim, kao i W i Y, i sam X mora da bude odredište, odnosno izvor svog saobraćaja koji stiže u X ili ga napušta. Ali, kako se implementira i nameće takvo ponašanje završne mreže? Kako sprečiti X da prosleđuje saobraćaj između B i C? To se može lako postići kontrolisanjem načina na koji se objavljuju BGP rute. Konkretno, X će funkcionišati kao završna mreža ako (svojim susedima B i C) objavi da ne poseduje putanje hi za jedno odredište osim njega samog. To jest, iako X zna za putanju, na primer XCY, kojom se stiže do mreže Y, on neće objaviti tu putanju svom susedu B. Pošto B ne zna da X ima putanju prema Y, B neće nikad proslediti saobraćaj čije je odredište Y (ili C) preko X. Ovaj jednostavan primer



**Slika 4.39** ◆ Jednostavni BGP scenario

ilustruje kako se za implementiranje odnosa između korisnika i posrednika može upotrebiti selektivno objavljuvanje ruta.

Predimo sada na mrežu posrednika, na primer, AS B. Uzmimo da je B saznao (od A) da A ima putanju AW prema W. B znači može da instalira rum BAWV u svoju bazu informacija za rutiranje. Jasno je, B takođe hoće da objavi putanju BAW svom korisniku X da bi X znao da se prema W može nitirati preko B. Međutim, da li bi B trebalo da oglasi putanju BAVV i drugom korisniku C? Ako to uradi, tada bi C mogao da rutira saobraćaj prema W preko CBAW. Ako su A, B i C posrednici okosnice, tada bi B opravdano mogao da smatra da on ne treba da deli teret (i troškove!) prenošenja tranzitnog saobraćaja između posrednika A i C. B može opravdano da smatra da je obaveza posrednika A i C da pri rutiranju koriste svoju direktnu vezu (i trošak!) između A i C. Trenutno među posrednicima za Internet usluge iz okosnice ne postoje zvanični standardi za međusobno rutiranje. Međutim, grubo pravilo kojeg se drže komercijalni posrednici zai Internet usluge jeste da saobraćaj koji teče njihovom okosnicom mora imati izvor ili odredište (ili obe) u mreži nekog njihovog korisnika; inače taj saobraćaj koristi besplatnu vožnju u mreži tog posrednika. Pojedinačni partnerski ugovori (iz gore pomenute oblasti) obično su predmet pregovaranja samih posrednika i smatraju se njihovom tajnom; u [Huston 1999a] nalazi se zanimljiv opis partnerskih ugovora. Detaljniji opis načina na koji politike rutiranja odražavaju komercijalne odnose među posrednicima naći ćete u [Gao 2001].

Kao što smo napomenuli, BGP je *de facto* standard za rutiranje među AS-ovima u javnom Internetu. Ako vas zanima sadržaj različitih BGP tabela rutiranja (velike su!) iz mtera posrednika prvog reda, pogledajte <http://www.routeviews.org>. BGP tabele rutiranja često sadrže na desetine hiljada prefiksa i odgovarajućih atributa. Statistike o veličini i karakteristikama BGP tabela rutiranja predstavljene su u [Huston 2001].

Ovim zaključujemo kratak uvod u BGP. Važno je razumeti BGP jer on igra centralnu ulogu u Internetu. Savetujemo čitaocima da pregledaju reference u [Griffin 2002; Stewart 1999; Labovitz 1997; Halabi 2000; Huitema 1998; Gao 2001; Feam-ster 2004] i nauče više o BGP-u.

## PRINCIPI U PRAKSI

### ZAŠTO SE UNUTAR 1 IZMEĐU AS-OVA KORISTE RAZLIČITI PROTOKOLI RUTIRANJA?

Pošto smo proučili detalje konkretnih protokola rutiranja koji se na današnjem Internetu koriste unutar i između AS-ova, možemo zaključiti razmatranje ovim možda najfundamen-lalnijim pitanjem koje se uopšte može postaviti o tim protokolima (nadam se da ste se sve vreme to pitali i da nisle zbog drveta izgubili iz vida šumu!): Zašto se koriste različiti protokoli rutiranja unutar i između AS-ova?

Odgovor na ovo pitanje ukazuje na osnovnu razliku među ciljevima rutiranja unutar i između AS-ova:

- ◆ *Politika.* Između AS-ova preovladaju pitanja politike. Ponekad je važno da se saobraćaju koji potiče iz datog AS-a nikako ne dozvoli prolaz kroz drugi konkretni AS. Slično tome, neki AS možda želi da kontrolise vrstu tranzitnog saobraćaja koji prenosi između drugih AS-ova. Videli smo da BGP prenosi atribute putanje i omogućava kontro-lisanu distribuciju informacija i rutiranju, pa se mogu donositi takve odluke zasnovane na politici. Unutar AS-a se sve formalno nalazi pod istom administrativnom kontrolom, pa pitanja politike imaju daleko manji značaj kada sa biraju tule unutar AS-a.
- ◆ *Skaliranje.* Za rutiranje među AS-ovima kritična je sposobnost algoritma'rutiranja i njegovih struktura podataka da se skaliraju kako bi se omogućilo rutiranje prema velikom broju mreža i između njih. Unutar AS-a skaliranje nije toliko kritično. Među ostalim, ako administrativni domen previše poraste uvek ga je moguće podeliti na dva AS-a, pa između njih koristiti rutiranje među AS-ovima. [Verovatno se sećate da OSPF dozvoljava da se podelom AS-a u „područja“ izgrade takve hijerarhije.]
- ◆ *Performanse.* Pošto politika toliko utiče na rutiranje među AS-ovima, kvalitet (na primer, performanse) upotrebljenih ruta cesto nije toliko značajan (tj. ponekad će se izabrati duži i skuplja ruta koja zadovoljava određene kriterijume politike umesto kraće rute koja te kriterijume ne zadovoljava). Zaista, videli smo da se rutama među AS-ovima ne pridružuje čak ni pojam cene (osim broja AS skokova). Unutar AS-a, međutim, pitanja politike nisu tako značajna, pa se rutiranje više bavi performansama koje se mogu postići na rutu.

## 4.7 Difuzno i višeznačno rutiranje

U ovom poglavlju smo se do sada bavili protokolima rutiranja koji podržavaju jednoznačnu komunikaciju (tj. od tačke do tačke) u kojoj jedan izvorni čvor šalje paket samo jednom odredišnom čvoru. U ovom odeljku obraćamo pažnju na protokole za difuzno i višeznačno rutiranje. U **difuznom rutiranju** mrežni sloj pruža

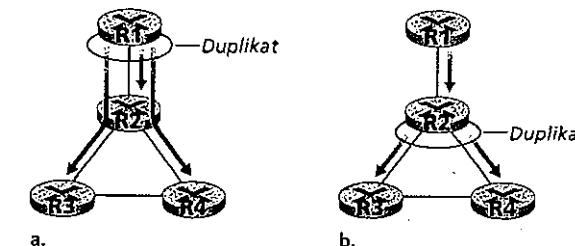
uslužu isporuke paketa koji se **iz** jednog izvornog čvora šalje svim ostalim čvorovima u mreži; **višeznačno rutiranje** omogućava da jedan izvorni čvor pošalje po primerak paketa nekom podskupu ostalih čvorova u mreži. U odeljku 4.7.1 razmo-tričemo algoritme difuznog rutiranja i njihovo otelotvorenje u protokolima rutiranja. Višeznačno rutiranje istražićemo u odeljku 4.7.2.

### 4.7.1 Algoritmi difuznog rutiranja

Možda je najprostiji način da se postignu komunikacije sa difuznim emitovanjem da otpremni čvor pošalje zaseban primerak paketa na svako odredište, kao što je prikazano na slici 4.40(a). Ako je dato  $N$  odredišnih čvorova, izvorni čvor jednostavno napravi  $N$  kopija paketa, adresira svaki primerak na drugo odredište a zatim pošalje  $N$  primeraka na odredišta pomoću jednoznačnog rutiranja. Ovaj pristup sa **iV-tostrukim jednoznačnim** difuznim emitovanjem je jednostavan - nije potreban nikakav nov protokol rutiranja mrežnog sloja, dupliranje paketa, niti prosleđivanje paketa. Taj pristup, međutim, ima nekoliko nedostataka. Prvi nedostatak je njegova neefikasnost. Ako je izvomi Čvor vezan za ostatak mreže samo jednim linkom, kroz taj link će proći primeraka (istog) paketa. Jasno je da bi bilo efikasnije da se preko prvog skoka pošalje samo jedan primerak a da zatim čvor na drugoj strani prvog skoka napravi i prosledi potreban broj dodatnih kopija. To jest, bilo bi efikasnije kada bi sami mrežni Čvorovi (a ne samo izvorni čvor) pravili duplike paketa. Na primer, na slici 4.40(b) link R1-R2 prelazi samo jedan primerak paketa. Taj paket se zatim duplira u R2 pa se linkovima R2-R3 i R2-R4 šalje po jedan primerak.

Još jedan nedostatak pristupa sa **AMostrukim jednoznačnim** emitovanjem je možda suptilniji ali nije manje značajan. Implicitna pretpostavka AMostrukog jednoznačnog emitovanja je da pošiljalac zna sve primaocu i njihove adrese. Ali kako da se pribave te informacije? Najverovatnije bi bili potrebni dodatni mehanizmi protokola (npr. protokol za difuzno emitovani poziv na učlanjenje ili registrovanje odredišta).

Pravlenie/brenenie duplikata



Slika 4.40 ♦ Dupliranje na izvoru i dupliranje u mreži

To bi dovelo do dodatnog preopterećenja, a što je još važnije, dodatnog usložnjavanja protokola koji je na početku izgledao sasvim jednostavan. Konačni nedostatak pristupa sa A<sup>+</sup>-tostrukim jednoznačnim emitovanjem odnosi se na svrhu kojoj difuzno emitovanje treba da služi. U odeljku 4.5 saznali smo da protokoli rutiranja prema stanju linkova koriste difuzno emitovanje za obznanjivanje informacija o stanju linkova koje se koriste za izračunavanje jednoznačnih ruta. Jasno je da ne bi bilo mudro (u najmanju ruku!) koristiti infrastrukturu za jednoznačno rutiranje u situaciji u kojoj se difuzno emituju informacije za pravljenje i ažuriranje jednoznačnih ruta.

S obzirom na nekoliko nedostataka pristupa sa N-tostrukim jednoznačnim emitovanjem, značajni su pristupi u kojima sami mrežni čvorovi igraju aktivnu ulogu u dupliranju paketa, prosleđivanju paketa i izračunavanju difuznih ruta. Ispitaćemo nekoliko takvih pristupa i pri tom koristiti notaciju grafova koju smo uveli u odeljku 4.5. Ponovo modeliramo mrežu kao graf,  $G = (N, E)$ , gde je  $H$  skup čvorova i kolekcija  $E$  ivica gde je svaka ivica par čvorova iz skupa  $N$ . Kada to ne bude dovodilo do zabune bićemo malo nedosledni sa notacijom pa ćemo koristiti  $N$  da bismo označili i skup čvorova kao i njegovu kardinalnost ( $|N|$ ) odnosno veličinu.

#### Nekontrolisano plavljenje

Najočiglednija tehnika da se ostvari difuzno emitovanje je **plavljenje** u kojem izvorni čvor šalje primerak paketa svim svojim susedima. Kada čvor primi difuzno emitovani paket, on ga duplira i prosleđuje svim svojim susedima (osim suseda od kojeg je primio paket). Jasno je da će, ako je graf povezan, ovom šemom kad tad svaki čvor u grafu primiti primerak paketa. Mada je ova šema jednostavna i elegantna, ona ima jednu fatalnu grešku (pre nego što nastavite sa čitanjem, pokušajte da otkrijete tu fatalnu grešku): ako graf sadrži cikluse, jedan ili više primeraka svakog difuzno emitovanog paketa će kružiti beskonačno. Na primer, na slici 4.40 će R2 plaviti prema R3, R3 će plaviti prema R4, R4 će plaviti prema R2, a R2 će plaviti (ponovo!) prema R3 i tako dalje. Ovaj jednostavan scenario dovodi do beskonačnog ■ kruženja dva difuzno emitovana paketa, jednog u smeru kretanja kazaljke na satu a drugog u smeru koji je suprotan smeru kretanja kazaljke na satu. Međutim, može se javiti jedna još razomjera fatalna greška: kada je čvor povezan sa više od dva druga čvora, on će napraviti i proslediti više primeraka difuzno emitovanog paketa od kojih će svaki napraviti više primeraka (u drugim čvorovima sa više od dva suseda) i tako dalje. Takva **oluja difuznog emitovanja** koja potiče od beskonačnog umnožavanja difuzno emitovanih paketa na kraju bi proizvela toliko difuzno emitovanih paketa da bi mreža postala neupotrebljiva. (Među domaćim zadacima na kraju ovog poglavlja nalazi se problem u kojem se analizira brzina kojom takva oluja difuznog emitovanja raste.)

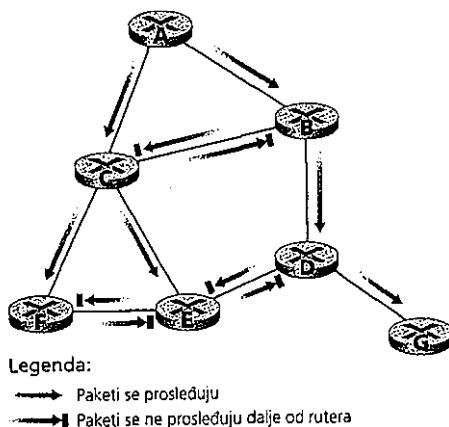
#### Kontrolisano plavljenje

Ključ za izbegavanje oluje difuznog emitovanja sastoji se u tome da čvor mudro proceni kada da plavi a kada ne (npr. ako je već primio i prosledio raniji primerak istog paketa). U praksi se to postiže na jedan od nekoliko načina.

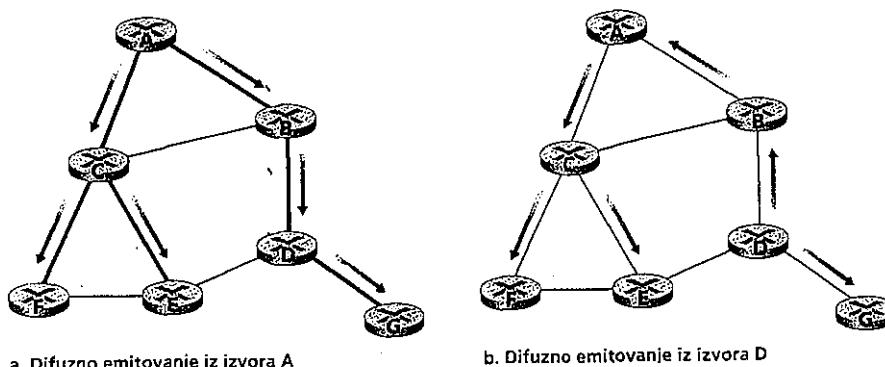
U **plavljenju pod kontrolom rednih brojeva** izvorni čvor stavlja u paket za difuzno emitovanje svoju adresu (ili neki drugi jedinstveni identifikator) kao i **redni, broj** difuznog emitovanja i zatim šalje paket svojim susedima. Svaki čvor održava listu izvornih adresa i rednih brojeva za sve difuzno emitovane pakete koje je već primio, duplirao i prosledio. Kada primi difuzno emitovani paket, čvor pro-verava da li se paket već nalazi na listi. Ako se nalazi na listi, paket se odbacuje; ako ga nema, paket se duplira i prosleđuje svim susedima (osim suseda od kojeg je paket primljen). U protokolu Gnutella, opisanom u poglavljiju 2, koristi se plavljenje pod kontrolom rednih brojeva za difuzno emitovanje poruka u preklapljenoj mreži. (Kada se koristi protokol Gnutella, dupliranje i prosleđivanje poruka vrši se u aplikacijskom a ne u mrežnom sloju.)

Drugi pristup rešavanju problema kontrolisanog plavljenja je algoritam za **prosleđivanje inverznom putanjom /reverse path forwarding, RPF**. [Dalal 1978] koji se ponekad naziva i difuzno emitovanje inverznom putanjom (*reversepath bro-adcasting, RPB*). Zamisao na kojoj se zasniva prosleđivanje inverznom putanjom je jednostavna, ali ipak elegantna. Kada ruter primi difuzno upućeni paket sa datom adresom izvora, on predaje paket na sve svoje izlazne linkove (osim onog na kojem je paket primljen) samo ako je paket stigao linkom koji se nalazi na njegovoj vlastitoj najkraćoj jednoznačnoj putanji prema pošiljaocu. Inače, ruter jednostavno odbacuje pristigli paket i ne prosleđuje ga ni na jedan od svojih izlaznih linkova. Takav paket može da se ispusti pošto ruter zna daje na linku koji se nalazi na njegovoj najkraćoj putanji prema pošiljaocu već primio ili će primiti primerak istog paketa. (Mogli biste proveriti da li će se to zaista i dogoditi i da neće doći do kruženja niti do oluje.) Obratite pažnju na to da prosleđivanje inverznom putanjom ne koristi jednoznačno rutiranje za isporuku paketa na određiste niti zahteva da ruter zna celu najkraću putanju između njega i izvora. Dovoljno je da zna za sledeći skok na njegovoj najkraćoj jednoznačnoj putanji prema pošiljaocu jer koristi identitet tog suseda samo za odlučivanje da li da prosledi primljeni difuzno upućeni paket.

Na slici 4.41 prikazanje RPF. Prepostavite da linkovi prikazani debelim zelenim linijama predstavljaju najjeftinije putanje od primalaca do izvora ( $A$ ). Čvor  $A$  na početku difuzno upućuje paket iz izvora  $A$  čvorovima  $C$  i  $B$ . Čvor  $B$  će paket iz izvora  $A$  koji je primio od  $A$  (pošto se  $A$  nalazi na njegovoj najjeftinije putanji prema  $A$ ) prosleđiti i u  $C$  i u  $D$ .  $B$  će zanemariti (ispustiti bez prosleđivanja) pakete iz izvora  $A$  koje primi iz svih ostalih ruta (na primer, od ruta  $C$  ili  $D$ ). Razmotrimo sada Čvor  $C$  koji će primiti paket iz izvora  $A$  direktno od  $A$ , ali i od  $B$ . Pošto se  $B$  ne nalazi na najkraćoj putanji od  $C$  do  $A$ ,  $C$  će zanemariti (ispustiti) sve pakete iz izvora  $A$  koje primi od  $B$ . Sa druge strane, kada  $C$  primi paket iz izvora  $A$  direktno od  $A$ , on će ga prosleđiti ruterima  $B, E \setminus F$ .



Slika 4.41 ♦ Prosleđivanje inverznom putanjom



Slika 4.42 ♦ Difuzno emitovanje sveobuhvatnim stablom

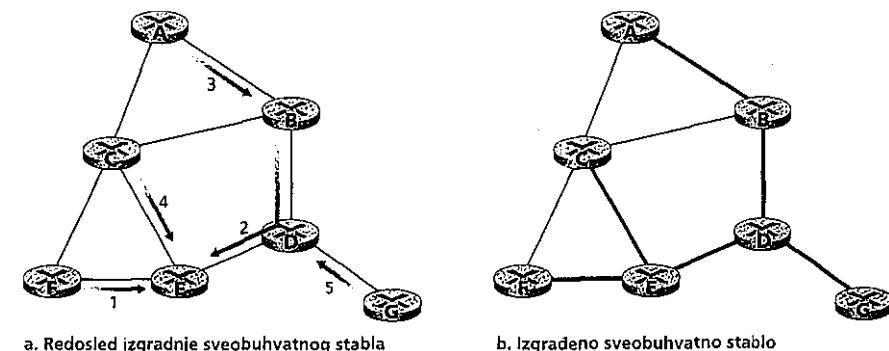
#### Difuzno emitovanje sveobuhvatnim stablom

Mada plavljenje pod kontrolom rednih brojeva i RPF izbegavaju oluje difuznog emitovanja, oni ne izbegavaju u potpunosti prenošenje redundantnih difuzno emitovanih paketa. Na primer, na slici 4.42 čvorovi *B*, *C*, *D* i *E* primiče jedan ili dva redundantna paketa. Idealno bi bilo da svaki čvor primi samo jedan primerak difuznog emitovanog paketa.

Ako ispitate stablo koje se sastoji od čvorova povezanih debljim linijama na slici 4.42(a) vidite da kada bi se difuzno emitovani paketi prosleđivali samo duž linkova ovog stabla, svaki čvor u mreži bi primio tačno jedan primerak difuzno emitovanog paketa - upravo rešenje kakvo tražimo! Ovo stablo je primer sveobuhvatnog stabla - stabla koje sadrži sve čvorove u grafu. Formalnije rečeno, sveobuhvatno stablo grafa  $G = (N, E) \setminus Q$  gde  $Q = (N, E')$  takav da je  $E'$  podskup od  $E$ ,  $G$  je povezan,  $G'$  ne sadrži cikluse i  $G'$  sadrži sve prvobitne čvorove iz  $G$ . Ako svaki link ima pridruženu cenu a cena stabla je suma cena linkova, tada se sveobuhvatno stablo čija je cena najmanja od svih sveobuhvatnih stabala tog grafa naziva **minimalno sveobuhvatno stablo**.

Prema tome, još jedan pristup obezbeđivanju difuznog emitovanja jeste da se prvo napravi sveobuhvatno stablo. Kada izvorni čvor hoće difuzno da emituje paket, on ga šalje na sve primerke linkova koji pripadaju sveobuhvatnom stablu. Čvor koji primi difuzno emitovani paket zatim prosleđuje paket svim svojim susedima u sveobuhvatnom stablu (osim suseda od kojeg je primio paket). Ne samo što sveobuhvatno stablo eliminše redundantne difuzno emitovane pakete, već kad se jednom uspostavi to stablo može koristi svaki čvor koji počinje difuzno emitovanje kao što je prikazano na slikama 4.41(a) i 4.42(b). Obratite pažnju na to da čvor ne mora biti svestran celog stabla; on jednostavno treba da zna koji od njegovih suseda iz  $G$  jesu susedi u sveobuhvatnom stablu.

Glavna složenost vezana za pristup sa sveobuhvatnim stablom jeste izgradnja i održavanje sveobuhvatnog stabla. Razvijeno je više distribuiranih algoritama sa sveobuhvatnim stablom [Gallager 1983, Gartner 2003]. Ovde ćemo razmotriti samo jedan jednostavan algoritam. U **pristupu sa centrom** za izgradnju sveobuhvatnog stabla definiše se centralni čvor (ponekad se naziva **tačkom sastanka** ili **centrom**).'



Slika 4.43 ♦ Izgradnja sveobuhvatnog stabla od centra

Čvorovi zatim tom centralnom čvoru upućuju jednoznačno adresirane poruke o pristupanju stablu. Poruka o pristupanju stablu se prosleduje jednoznačnim iritiranjem prema centru dok ne stigne do Čvora koji već pripada sveobuhvatnom stablu ili do centra. U svakom slučaju, putanja kojom je išla poruka o pristupanju definiše granu u stablu između centra i rubnog čvora koji je pokrenuo poruku o pristupanju. Može se smatrati da je ta nova putanja priključena postojećem stablu.

Na slici 4.43 prikazana je izgradnja sveobuhvatnog stabla. Pretpostavite da se Čvor *E* izabere za centar stabla. Stablu prvo pristupa Čvor *F* koji poruku o pristupanju prosleduje čvoru *E*. Link *EF* postaje početno stablo. Zatim stablu pristupa Čvor *B* tako što svoju poruku o pristupanju šalje čvoru *E*. Pretpostavimo da putanja jednoznačnog upućivanja iz *B* u *E* ide preko *D*. U tom slučaju, poruka pristupanja dodaje na stablo putanju *BDE*. Čvor *A* pristupa grupi prosleđivanjem svoje poruke o pristupanju prema *E*. Ako je putanja za jednoznačno upućivanje iz *A* prema *E* bila preko *B*, tada pošto je *B* već pristupio stablu, dolazak poruke o pristupanju iz *A* u *B* doveće do trenutnog priključivanja linka *AB* na stablo. Sledeći čvor koji pristupa je *C* prosleđivanjem svoje poruke o pristupanju direktno u *E*. Na kraju, pošto jednoznačno rutiranje iz *G* u *E* mora biti preko *D*, kada *G* pošalje svoju poruku o pristupanju u *E*, link *GD* se priključuje stablu u čvoru *D*.

#### Algoritmi za difuzno upućivanje u praksi

Protokoli za difuzno upućivanje se u praksi koriste i u aplikacijskom i u mrežnom sloju. Kao stoje rečeno u odeljku 2.6, Gnutella [Gnutella 2004] koristi difuzno upućivanje u aplikacijskom sloju za difuzno upućivanje upita o sadržaju među Gnutella učesnicima. Ovde je link između dva ravnopravna procesa distribuirane obrade u aplikacijskom sloju u stvari jedna TCP konekcija. Gnutella koristi jedan vid plavljenja pod kontrolom rednih brojeva u kojem se jedan 16-bitni identifikator i jedan 16-bitni opisivoč korisnih podataka (koji definišu tip Gnutella poruke) koriste za otkrivanje da li je primljena difuzno upućena ponika već primljena, duplirana i pro-sleđena. Kao stoje već rečeno u odeljku 2.6, Gnutella takođe koristi polje TTL (rok trajanja) da bi se ograničio broj skokova prilikom plavljenja. Kada proces Gnutella primi i duplira poruku, osmanjuje polje TTL pre nego što prosledi upit. Na taj način, prosleđena Gnutella poruka će stići samo do onih ravnopravnih učesnika koji su od pokretača upita udaljeni manje od datog broja (početne vrednosti TTL) skokova aplikacijskog sloja. Gnutellin mehanizam plavljenja se zato ponekad naziva *plavljenje sa ograničenim dometom*.

Još jedan vid plavljenja pod kontrolom rednih brojeva imamo i kod difuznog emitovanja objava o stanju linkova (*Link State Advertisement*, LSA) u algoritmu rutiranja OSPF [RFC 2328, Perlman 1999] i u algoritmu rutiranja JS-IS [RFC 1142, Perlman 3999]. Za identifikaciju objava o stanju linkova OSPF koristi 32-bitni redni broj kao i 16-bitno polje starosti. Verovatno se sećate da OSPF čvor povremeno difuzno emituje LSA-ove za povezane linkove kada se promeni cena linka ili kada se link uključi/isključi. Redni brojevi LSA-ova koriste se za otkrivanje duplikata ali za još

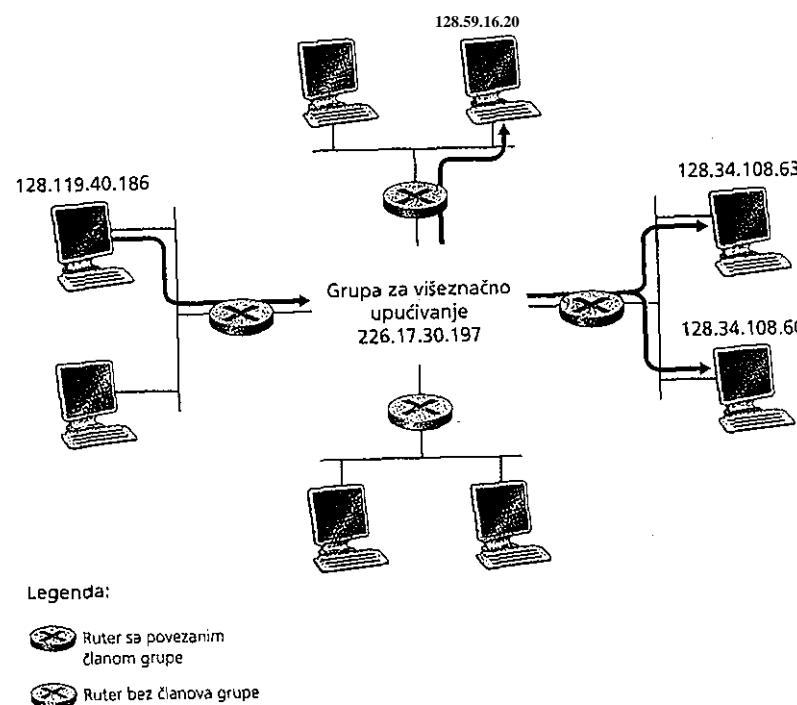
jednu važnu funkciju u OSPF-u. S obzirom na plavljenje, moguće je da LSA koji je napravljen na izvoru u trenutku / stigne *nakon* novijeg LSA-a koji je na istom izvoru napravljen u trenutku / + *S*. Redni brojevi koje pravi izvorni čvor omogućavaju da se stariji LSA razlikuje od novijeg. Polje starosti ima funkciju sličnu polju TTL. Početna vrednost polja starosti je nula a povećava se pri svakom skoku plavljenja, ali se povećava i dok u memoriji rutera čeka da bude prosleden. Mada smo samo ukratko opisali LSA algoritam plavljenja, primećujemo da projektovanje LSA protokola za difuzno emitovanje može zaista da bude veoma složeno, U [RFC 789; Perlman 1999, odeljak 12.2.3.3] opisan je incident kada je neispravno prenet LSA iz dva rutera u kvaru doveo do toga da jedna od početnih verzija algoritma LSA plavljenja skoro zaustavi ceo ARPAnet.

#### 4.7.2 Višeznačno upućivanje

U prethodnom odeljku smo videli da se sa uslugama difuznog emitovanja, paketi isporučuju svakom čvoru u mreži. U ovom odeljku obraćićemo pažnju na uslugu višeznačnog upućivanja u kojoj se višeznačno upućeni paket isporučuje samo *podskupu* mrežnih Čvorova. Niz novih mrežnih aplikacija zahteva isporuku paketa od jednog ili više pošiljalaca jednoj grupi primalaca. Te aplikacije služe za paketsko slanje podataka (na primer, transfer softverske nadogradnje od programera ka korisnicima kojima je potrebna nadogradnja), striming kontinualnih medija (na primer, transfer audia, videa i teksta živog predavanja za skup distribuiranih učesnika u predavanju), aplikacije sa deljenim podacima (na primer, elektronska tabla za tele-konferencije koja je zajednička za više distribuiranih učesnika), dodatni podaci (na primer, stanje berze, ažuriranje veb kesa i interaktivne igre (na primer, distribuirana interaktivna virtuelna okruženja ili igre sa više igrača kao što je Quake).

Kada se koristi višeznačna komunikacija, odmah nailazimo na dva problema - kako prepoznati primaocve višeznačno upućenog datagrama i kako adresiran datagram koji se šalje tim primaocima. U slučaju jednoznačne komunikacije, IP adresa primaoca (odredišta) nalazi se u svakom jednoznačnom upućenom IP datagramu i identificuje jedinog primaoca; u slučaju difuznog upućivanja, *svi* Čvorovi treba da prime difuzno emitovani paket pa nisu potrebne nikakve adrese odredišta. Ali, u slučaju višeznačnog upućivanja sada imamo više primalaca. Ima li smisla da svaki višestruko upućeni datagram sadrži IP adrese svih tih primalaca? Mada bi za mali broj primalaca to rešenje bilo izvodljivo, ne bi bilo dobro za slučaj stotina ili hiljada primalaca; količina adresnih informacija u datagramu bila bi veća od podataka koji se prenose u polju korisnih podataka. Da bi pošiljalac mogao eksplicitno da navede sve primaoce, on bi morao da zna identitete i adrese svih primalaca. Ubrzo ćemo videti da postoje slučajevi kada bi takav zahtev bio nepoželjan.

Iz tih razloga, u arhitekturi Interneta (a takođe i u arhitekturi ATM [Black 1995]), višeznačno upućeni datagram adresira se pomoću indirektnih adresa. To jest, za grupu primalaca se koristi jedan identifikator, a primerak datograma koji je adresiran na grupu pomoću tog identifikatora isporučuje se svim primaocima koji su pridruženi



**Slika 4.44** ♦ Grupa za višeznačno upućivanje: datagram adresiran na grupu isporučuje se svim članovima grupe

toj grupi. Jedan identifikator koji u Internetu predstavlja grupu primalaca je adresa za višeznačno upućivanje klase D. Grupa primalaca vezana za adresu klase D naziva se grupa za višeznačno upućivanje. Pojam grupe za višeznačno upućivanje prikazan je na slici 4.44. Ovde su četiri računara (osenečena zeleno) pridružena adresi grupe za višeznačno upućivanje 226.17.30.197 i primiče sve datagrame adresirane na tu adresu 2a višeznačno upućivanje. Problem koji ostaje da se resi je činjenica da svaki računar ima jedinstvenu IP adresu za jednoznačno upućivanje koja je potpuno nezavisna od adrese grupe za višeznačno upućivanje u kojoj učestvuje.

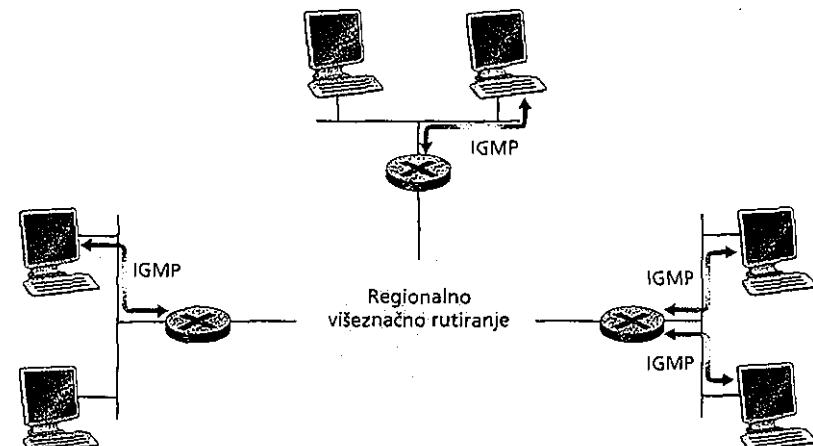
Mada je pojam grupe za višeznačno upućivanje jednostavan, on otvara niz novih pitanja. Kako se grupa osniva i kako prestaje da postoji? Kako se bira adresa grupe? Kako se u grupu dodaju novi računari (bilo kao pošiljaoci ili kao primaoci)? Može li svako da se priključi grupi (i zatim šalje ili prima datagrame tz grupe) ili se članstvo u grupi ograničava, i ako je tako ko ga kontroliše? Da li u okviru protokola mrežnog sloja Članovi grupe znaju identitet ostalih Članova? Kako mrežni ruteri saraduju

isporučivanju višeznačno upućenog datagrama svim Članovima grupe? Za Internet, odgovori na sva ova pitanja uključuju protokol za upravljanje grupama na Internetu (*Internet Group Management Protocol*, IGMP) [RFC 3376]. Zato ćemo sada razmotriti IGMP, pa ćemo se nakon toga vratiti na ova Šira pitanja.

#### IGMP

Protokol za upravljanje grupama na Internetu (*Internet Group Management Protocol*, IGMP) verzija 3 [RFC 3376], izvršava se između računara i njegovog direktno povezanog ruteru (nezvanično, direktno povezani ruter možete zamisliti kao ruter prvog skoka koji računar vidi na putanji prema bilo kojem drugom računaru van vlastite lokalne mreže ili kao ruter poslednjeg skoka svake putanje prema ovom računaru), kao što je prikazano na slici 4.45. Na slici 4.45 prikazana su tri ruteru prvog skoka za višeznačno upućivanje od kojih je svaki povezan sa svojim računarima preko jednog izlaznog lokalnog interfejsa. Taj lokalni interfejs je u ovom primeru povezan sa LAN-om i mada svaki LAN ima više povezanih računara, oni obično ne učestvuju svi u isto vreme u istoj grupi za višeznačno upućivanje.

IGMP obezbeđuje način na koji računar obaveštava svoj povezani ruter da aplikacija koja se izvršava na računaru želi da pristupi određenoj grupi za višeznačno upućivanje. Kako je IGMP ograničen na računar i njegov direktno povezani ruter, jasno je da je potreban drugi protokol koji će služiti za koordinaciju višeznačnih



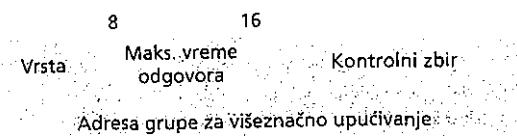
**Slika 4.45** ♦ Dve komponente višeznačnog upućivanja u mrežnom sloju: IGMP i protokoli za višeznačno rutiranje

rutera (uključujući i povezane rutere) u čelom Internetu da bi se višezačno upućeni datagrami rutirali do konačnog odredišta. Ova poslednja funkcija postiže se pomoću algoritama za višezačno rutiranje u mrežnom sloju kao što su PIM, DVMRP i MOSPF. Višezačno upućivanje u mrežnom sloju Interneta sastoji se od dve komplementarne komponente: IGMP-a i protokola za višezačno rutiranje.

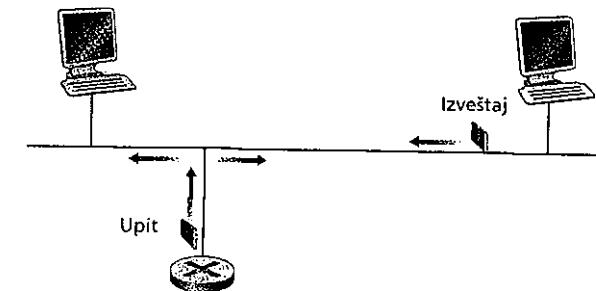
Iako se IGMP naziva „protokolom članstva u grupama“ taj naziv nije sasvim tačan pošto se IGMP izvršava *lokalno*, između računara i jednog povezanog ruter. Bez obzira na taj naziv, IGMP nije protokol koji se izvršava među svim računarima koji su pristupili jednoj grupi za višezačno upućivanje. Zaista, u mrežnom sloju ne postoji nijedan protokol članstva u grupi za višezačno upućivanje koji se izvršava na svim Internet računarima jedne grupe. Ne postoji, na primer, nijedan protokol mrežnog sloja koji bi omogućio računaru da utvrdi identitet svih ostalih računara u celoj mreži koji su pristupili nekoj grupi za višezačno upućivanje. (Pročitajte probleme za domaći zadatak gde će se dodatno istraživati posledice ove projektantske odluke.)

IGMP ima samo tri vrste poruka. Fornat IGMP poruke prikazan je na slici 4.46. Kao i za ICMP, IGMP poruke se prenose (enkapsulirane) unutar IP datagrama sa brojem IP protokola 2. Ruter šalje poruku opšti upit o članstvu (engl. member-ship\_query message) svim računarima na priključenom interfejsu (na primer, svim računarima lokalne mreže) da bi utvrdio skup svih grupa za višezačno upućivanje kojima su pristupili računari sa tog interfejsa. Ruter može takođe da utvrdi da li je bilo koji računar sa priključenog interfejsa pristupio nekoj konkretnoj grupi za višezačno upućivanje ako uputi konkretni upit o članstvu. Kao stoje prikazano na slici 4.47, konkretni IGMP upit o članstvu će u polju za adresu grupe imati adresu za višezačno upućivanje tražene grupe. Na poruku za upit o članstvu računari odgovaraju IGMP porukom izveštajem o članstvu (engl. membership\_report message) kao stoje prikazano na slici 4.47. Poruka sa izveštajem o članstvu takođe se pravi u računaru kada aplikacija prvi put pristupa grupi za višezačno upućivanje, pa ne čeka od rutera poruku sa upitom o članstvu.

Poslednja vrsta IGMP poruke je poruka o napuštanju grupe (engl. leave\_group message). Zanimljivo je daje ova poruka opcionalna! Ako je ona opcionalna, kako će ruter otkriti da na priključenom interfejsu više nema računara koji pripadaju određenoj grupi za višezačno upućivanje? Odgovor na ovo pitanje leži u načinu na koji se koristi IGMP poruka za upit o članstvu. Kada na poruku sa upitom o članstvu za



Slika 4.46 ♦ Format IGMP poruke



Slika 4.47 ♦ IGMP upit i izveštaj o članstvu

određenu adresu grupe za višezačno upućivanje ne odgovori nijedan računar, ruter *zaključuje* da nema računara u dатој grupi. Ovo je primer onog što se ponekad naziva meko stanje internetskog protokola. U protokolu sa mekim stanjem, stanje (u slučaju IGMP-a to je činjenica da postoje računari koji pripadaju dатој grupi za višezačno upućivanje) se prekida putem tajm-auta (u ovom slučaju, pomoću povremene poruke sa upitom o članstvu iz ruter) ako se eksplisitno ne obnovi (u ovom slučaju porukom sa izveštajem o članstvu iz priključenog računara). Postoji mišljenje daje protokole sa mekim stanjem jednostavnije kontrolisati od protokola sa čvrstim stanjima koja moraju eksplisitno da se uspostavljaju i uklanjaju, a moraju još da imaju i mehanizme za oporavak od situacije kada otkaze ili se pre vremena prekine entitet zadužen za uklanjanje stanja [Sharma 1997]. Izvrstan opis mekog stanja može se naći i u knjigama [Raman 1999; Ji 2003] pročitajte i dodatak *Principi u praksi* u odeljku 7.9.

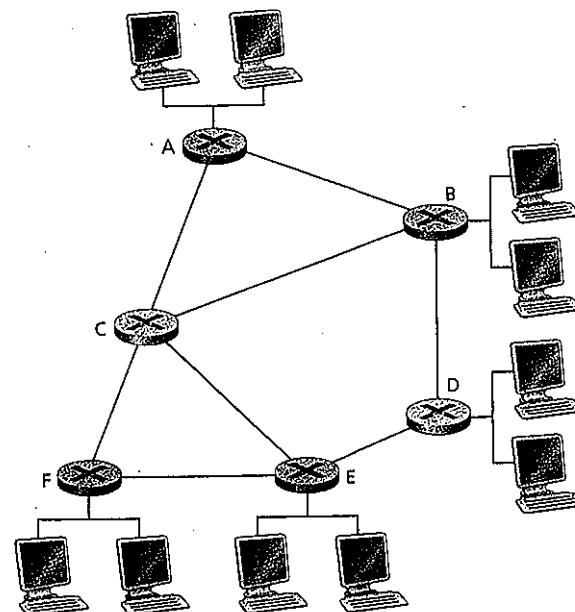
Pošto smo proučili protokol za pristupanje grupi za višezačno upućivanje i za njeno napuštanje, sada ćemo lakše razumeti važeći model internetske usluge višezačnog upućivanja koja se zasniva na radu Štiva Diringa [RFC 1132, Deering 1990]. U tom modelu usluge, svaki računar može da pristupi grupi za višezačno upućivanje u mrežnom sloju. Računar jednostavno izda svom priključenom ruteru IGMP poruku sa izveštajem o članstvu. Taj ruter će, u saradnji sa drugim ruterima na Internetu, uskoro početi tom računaru da isporučuje višezačno upućene datagrame. Prema tome, pristupanje grupi za višezačno upućivanje inicira primalac. Pošiljalac ne mora da brine o eksplisitnom dodavanju primalaca u grupu za višezačno upućivanje, ali ne može ni da kontroliše ko pristupa grupi, pa prema tome ni ko prima datagrame koji se grupi šalju. Slično tome, u početnim verzijama IGMP-a primalac nije mogao da odredi skup izvora iz kojih želi (ili ne želi) da prima višezačno upućene pakete; IGMPv3 omogućava da se to odredi.

Postojeći model usluge višezačnog upućivanja na Internetu zasniva se po mnogo čemu na istoj filozofiji kao i model usluge za jednoznačno upućivanje - krajnje jednostavan mrežni sloj, a dodatne funkcije (kao stoje članstvo u grupama) obezbeđuju se u protokolima viših slojeva u računarima na periferiji mreže. Ova filozofija je neosporno uspešna u slučaju jednoznačnog upućivanja. Da li će minimalistički

mrežni sloj biti isto tako uspešan za model usluge višezačnog upućivanja, ostaje još uvek otvoreno pitanje. Jedan alternativni model predstavljen je u [Holbrook 1999, RFC 3569]. Zanimljiv opis važećeg modela usluge višezačnog upućivanja na Internetu i pitanja vezanih za njegovu realizaciju naći ćete u [Diot 2000].

#### Algoritmi za višezačno rutiranje

Na slici 4.48 prikazana je instalacija za problem višezačnog rutiranja. Razmotrimo jednu grupu za višezačno upućivanje i prepostavimo da svaki ruter kome je priključen računar koji je pristupio toj grupi može da šalje ili da prima saobraćaj adresiran na tu grupu. Na slici 4.48 su računari koji su pristupili grupi za višezačno upućivanje osenčeni u boji, a takođe i njihov direktno povezani ruter. Kao što se vidi na slici 4.48, višezačno upućeni saobraćaj potreban je samo jednom podskupu u populaciji ruta za višezačno upućivanje (onima čiji su priključeni računari pristupili toj grupi za višezačno upućivanje). Na slici 4.48 to su ruteri A, B, E \ F. Pošto nijedan od računara priključenih na ruter D nije pristupio grupi za višezačno upućivanje, a ruter C nije priključen nijedan računar, ni C ni D nemaju potrebu da im se šalje saobraćaj grupe za višezačno upućivanje.

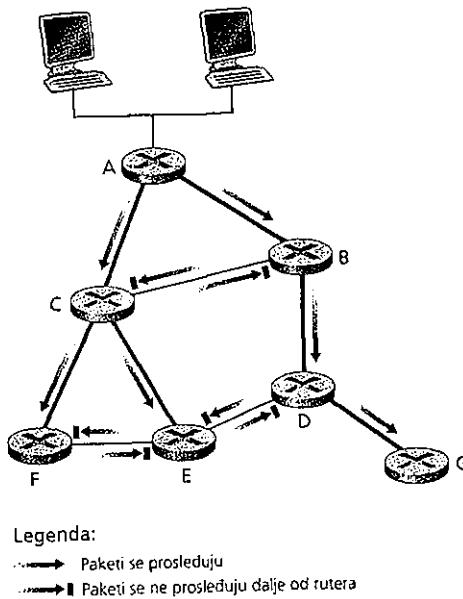


**Slika 4.48** ♦ Računari sa višezačnim upućivanjem, njihovi priključeni ruteri i drugi ruteri

Cilj višezačnog rutiranja je da se pronade stablo linkova koje povezuje sve ruterne koji imaju priključene računare pripadnike grupe za višezačno upućivanje. Višezačno upućeni paketi će se zatim rutirati po tom stablu od pošiljaoca do svih računara koji pripadaju stablu za višezačno upućivanje. Naravno, u stablu se mogu naći i ruteri bez priključenih računara koji pripadaju grupi za višezačno upućivanje (na primer, na slici 4.48 ne mogu se povezati ruteri A, B, E \ F \ x stablo, a da se ne uključi ruter C ili ruter D ili oba).

U praksi se za utvrđivanje stabla višezačnog rutiranja koriste dva pristupa. Ti pristupi se razlikuju po tome da li se koristi jedno stablo zajedničko za grupu za distribuciju saobraćaja od svih pošiljalaca u grupi ili se za svakog pojedinačnog pošiljaoca pravi zasebno stablo rutiranja.

- ♦ **Višezačno rutiranje pomoću stabla zajedničkog za grupu.** Kao u slučaju difuznog emitovanja sveobuhvatnim stablom, višezačno rutiranje pomoću stabla zajedničkog za grupu zasniva se na izgradnji stabla koje obuhvata sve rubne ruterne tako što priključeni računari koji pripadaju grupi šalju (jednoznačne) poruke pristupanja adresirane na centralni čvor. Kao i kod difuznog emitovanja poruka o pristupanju se pomoću jednoznačnog rutiranja prosljeđuje prema centru dok ne stigne u sam centar ili do ruteru koji već pripada stablu za višezačno upućivanje. Svi ruteri sa putanje kojom je išla poruka o pristupanju će zatim pro-sleđivati primljene višezačno upućene pakete prema rubnom ruteru koji je inicirao pristupanje. Osnovno pitanje kod stabla za višezačno rutiranje sa centrom je postupak kojim se bira centar. Algoritmi za biranje centra opisuju se u [Wall 1980; Thaler 1997; Estrin 1997].
- ♦ **Višezačno rutiranje pomoću stabla od izvora.** Dok je algoritam višezačnog rutiranja sa stablom grupu služio za izradu jednog zajedničkog stabla rutiranja koje se koristi za rutiranje paketa od svih pošiljalaca, u drugoj velikoj klasi algoritama za višezačno rutiranje se za svaki izvor u grupi za višezačno upućivanje pravi zasebno stablo. U praksi se RPF algoritam (sa izvornim čvorom x) koristi za izgradnju stabla za višezačno upućivanje datagrama koji polaze od izvora\*. Algoritam RPF koji smo proučili treba malo prilagoditi da bi se koristio za višezačno upućivanje. Da biste shvatili zašto, razmotrite šta se događa u ruteru D na slici 4.49. Ako se koristi RPF za difuzno emitovanje, on će proslediti pakete ruteru G iako ruter G nema nijedan povezani računar koji je pristupio grupi za višezačno upućivanje. Mada to nije tako strašno u slučaju kada D ima samo jedan nizvodni ruter, G, zamislite šta bi se dogodilo kada biste nizvodno od D imali na hiljade ruta! Svaki od tih ruta primio bi neželjene višezačno upućene pakete. (Taj scenario nije sasvim nemoguće. Prvobitni MBone [Casner 1992; Macedonia 1994], prva globalna mreža za višezačno upućivanje imala je na početku upravo taj problem.) Rešenje problema sa primanjem neželjenih višezačno upućenih paketa pod RPF-om naziva se orezivanje. Višezačni ruter koji primi višezačno upućene pakete i nema priključene računare koji su pristupili



**Slika 4.49** ♦ Prosleđivanje inverznom putanjom, višeznačno upućivanje

toj grupi poslaće uzvodnom rutcu poruku orezivanja. Ako ruter primi poruke za orezivanja od svih svojih nizvodnih ruta on može da prosledi poruku orezivanja uzvodno.

#### Višeznačno rutiranje na Internetu

Prvi protokol za višeznačno rutiranje koji se koristio na Internetu i algoritam za višeznačno rutiranje sa najširim podrškom je **protokol za višeznačno usmeravanje sa vektorom rastojanja** (*Distance Vector Multicast Routing Protocol, DVMRP*) [RFC 1075]. DVMRP implementira stablo od izvora, prosleđivanje inverznom putanjom i orezivanje. DVMRP koristi algoritam vektora rastojanja koji omogućava da svaki ruter izračuna izlazni link (sledeći skok) na svojoj najkraćoj putanji prema svakom mogućem izvoru. Ta informacija se zatim koristi u algoritmu RPF koji smo malo pre opisali. Javno dostupan primerak DVMRP softvera može se dobiti u [mrouted 1996].

Pored izračunavanja informacija za sledeći skok, DVMRP takođe izračunava spisak zavisnih nizvodnih ruta koji je potreban za orezivanje. Kada ruter primi poruke orezivanja za datu grupu od svih svojih zavisnih nizvodnih ruta, on će pro-

pagirati poniku orezivanja uzvodno prema ruteru od koga prima višeznačno upućeni saobraćaj za tu grupu. DVMRP poruka orezivanja sadrži Životni vek orezivanja (čija je podrazumevana vrednost 2 sata) koji označava koliko će orezana grana ostati isključena do ponovnog automatskog uključenja. Ruter šalje DVMRP poniku priključenja svom uzvodnom susedu da bi prethodno odsečenu granu ponovo dodaо na stablo višeznačnog upućivanja.

Drugi protokol za višeznačno upućivanje u širokoj upotrebi na Internetu je **protokol rutiranja za višeznačno upućivanje nezavisno od protokola** (*Protocol Independent Multicast, PIM*) [Deering 1996; RFC 2362; Estrin 1998b] koji eksplicitno predviđa dva različita scenarija višeznačne distribucije. U takozvanom **gustom režimu**, članovi grupe za višeznačno upućivanje su gusto locirani; tj. mnoge rute ili većinu ruta u području treba uključiti u rutiranje višeznačno upućenih datagrama. U **retkom režimu** broj ruta sa priključenim članovima grupe je mali u odnosu na ukupan broj ruta; članovi grupe su široko rasejeni. PIM se prilagođava dihotomiji retko-gusto tako što nudi dva eksplicitna režima rada: gusti i retki režim. PIM Dense Mode je tehnika prosleđivanja inverznom putanjom sa slanjem i orezivanjem slična po koncepciji sa protokolom DVMRP. PIM Sparse Mode je pristup na osnovu centra sličan starijem protokolu za višeznačno rutiranje CBT (core-based tree) [RFC 2201; RFC 2189]. Jedno novije svojstvo PIM-a je sposobnost da se nakon pristupanja zajedničkoj tački prebací sa zajedničkog stabla grupe na stablo sa određenim izvorom. Stablo konkretnog izvora ponekad bolje odgovara jer se smanjuje koncentracija saobraćaja kada se koristi više stabala sa konkretnim izvorima (proučite probleme za domaći zadatak). Ako se koristi PIM Sparse Mode, ruter koji od jednog svog priključenog računara primi datagram za slanje jednoznačno će uputiti datagram zajedničkoj tački sastanka. Zajednička tačka (*rendezvous point, RP*) zatim višeznačno upućuje datagram preko zajedničkog stabla grupe. RP obaveštava pošiljaoca da prekine slanje ako nema ruta priključenih na stablo (tj. kada niko ne sluša!).

U gornjem opisu implicitno smo pretpostavili da svi ruti izvršavaju isti protokol za višeznačno rutiranje. Kao što smo videli i kod jednoznačnog rutiranja to se obično događa unutar jednog autonomnog sistema (AS). Međutim, različiti AS-ovi mogu da izvršavaju različite protokole za višeznačno rutiranje. Za glavne Internet-ske protokole za višeznačno rutiranje definisana su pravila interoperabilnosti [RFC 2715]. (Pravila su posebno zamršena zbog veoma različitih pristupa višeznačnom rutiranju u protokolima za retki i gusti režim.) Još uvek, međutim, nedostaje protokol za višeznačno rutiranje među AS-ovima da bi se rutirali višeznačno upućeni datogrami između različitih AS-ova.

Do sada je DVMRP služio kao *de facto* protokol za višeznačno rutiranje među AS-ovima.

Na kraju, pomenućemo da višeznačno upućivanje nije na Internetu uzeo mnogo maha. Mnoge kompanije za prenos (streaming) video tokova i kompanije za distribuciju sadržaja umesto višeznačnog upućivanja koriste preklapanje i prave mreže za višeznačnu distribuciju u aplikacijskom sloju. Kao i IPv6, višeznačni IP se borio - i još uvek se bori - za značajniji prodor na Internet.

## 4.8 Rezime

U ovom poglavlju krenuli smo u istraživanje jezgra mreže. Naučili smo da mrežni sloj obuhvata sve računare i rutere u mreži. Zbog toga su protokoli mrežnog sloja najsloženiji u familiji protokola.

Saznali smo da ruter možda mora istovremeno da obradi na milione tokova paketa između različitih parova izvora i odredišta. Da "bi" ruter mogao da obradi tako veliki broj tokova, projektanti su tokom godina zaključili da bi zadaci rutera trebalo da ostanu što jednostavniji. Da bi se pojednostavio posao rutera, moguće je pre-duzeti različite mere uključujući korišćenje mrežnog sloja sa datagramima umesto mrežnog sloja sa virtuelnim kolima, upotrebu ujednačenog zaglavljaka fiksne dužine (kao za IPv6), eliminisanje fragmentacije (isto kao IPv6) kao i pružanje samo usluge najboljeg pokušaja. Ovde je možda najvažniji trik u tome da se *ne vodi računa* o pojedinačnim tokovima, već da se odluke o rutiranju zasnivaju samo na hijerarhijski strukturisanim odredišnim adresama u paketima. Zanimljivo je primetiti da poštanska služba godinama koristi upravo taj trik.

U ovom poglavlju, razmotrili smo takođe osnovne principe algoritama rutiranja. Saznali smo da projektanti algoritama rutiranja apstrahuju mrežu računara kao graf sa Čvorovima i granama. Polazeći od takve apstrakcije moguće je iskoristiti bogatu teoriju pronađenja najkraćih putanja u grafovima koja se u zajednici operacionih istraživanja i algoritama razvija već 40 godina. Videli smo da postoje dva šira pristupa, jedan centralizovani (globalni) pristup u kojem svaki čvor dobija kompletну mapu mreže i nezavisno primenjuje algoritam za rutiranje najkraće putanje; i jedan decentralizovani pristup u kojem pojedinačni čvorovi imaju samo parcijalnu sliku cele mreže, a čvorovi zajednički rade na tome da se paketi isporuče po najkraćim rutama. Videli smo kako se hijerarhija koristi u rešavanju problema skaliranja tako što se velike mreže dele u nezavisne administrativne domene po imenu autonomni sistemi (AS-ovi). Svaki AS nezavisno rutira svoje datagrame kroz AS, kao što svaka zemlja nezavisno rutira svoju poštu po državi. Saznali smo kako se centralizovani, decentralizovani i hijerarhijski pristup primenjuju u glavnim protokolima rutiranja na Internetu: RIP, OSPF i BGP. Proučavanje algoritama rutiranja zaključili smo razmatranjem difuznog i viševeznog rutiranja.

Zaključujemo proučavanje mrežnog sloja i prelazimo korak dalje niz familiju protokola, u sloj veze. Isto kao i mrežni sloj, sloj veze takođe spada u jezgro mreže. Ali, u sledećem poglavlju videćemo da sloj veze ima znatnije lokalizovan zadatak prenošenja paketa među čvorovima na istom linku ili LAN-u. Mada ovaj zadatak može naizgled da se čini trivijalnim u poređenju sa zadacima mrežnog sloja, videćemo da sloj veze postavlja pred nas niz značajnih i fascinantnih pitanja, koja će nas dugo zaokupljati.

## | Domaći zadatak: problemi i pitanja

### Poglavlje 4 Kontrolna pitanja

#### ODEUCU.I-4.2

- Obnovimo terminologiju koja se koristi u ovoj knjizi. Verovatno se sećate da se paket transportnog sloja naziva *segment* a daje paket sloja veze *okvir*. Kako se naziva paket mrežnog sloja? Sigurno se sećate da se i ruteri i komutatori sloja veze nazivaju *komutatorima paketa*. Koja je osnovna razlika između rutera i komutatora sloja veze? Imajte na umu da izraz *ruteri* koristimo i u mrežama sa datagramima i u mrežama sa virtuelnim kolima.
- Koje su dve glavne funkcije mrežnog sloja u mrežama sa datagramima? Koje dodatne funkcije ima mrežni sloj u mrežama sa virtuelnim kolima?
- Kakva je razlika između rutiranja i prosleđivanja?
- Da li ruteri koriste tabele prosleđivanja i u mrežama sa datagramima i u mrežama sa virtuelnim kolima? Ako je tako, opišite tabele prosleđivanja za obe klase mreža.
- Opišite neku zamišljenu uslugu koju bi mrežni sloj mogao da pruža pojedinačnom paketu. Uradite to isto za tok paketa. Da li neke od vaših zamišljenih usluga postoje u mrežnom sloju Interneta? Da li postoje u ATM-ovom modelu usluge CBR? Da li postoje u ATM-ovom modelu usluge ABR?
- Navedite neke aplikacije koje bi imale koristi od ATM-ovog modela usluge CBR.

#### ODELJAK 4.3

- Objasnite zašto svaki ulazni port u ruteru velike brzine čuva kopiju tabele prosleđivanja?
- U odeljku 4.3 opisuju se tri vrste komutatorske mreže. Navedite koje su i ukratko ih opišite.
- Opišite na koji način se gubitak paketa može dogoditi u ulaznom portu. Opišite kako se može eliminisati gubitak paketa u ulaznom portu (ako se ne koriste beskonačne pomoćne memorije).
- Opišite na koji način se gubitak paketa može dogoditi u izlaznim portovima,
- Sta je to HOL blokiranje? Da li se ono dešava u ulaznim ili u izlaznim portovima?

#### ODEUAK 4.4

- Da li ruteri imaju IP adrese? Ako imaju, koliko ih imaju?
- Koji je 32-bitni binarni ekvivalent IP adrese 223.1.3.27?

14. Posetite računar koji koristi DHCP za dobijanje IP adrese, mrežne maske, podrazumevani niter i IP adrese lokalnog DNS servera. Navedite te vrednosti.
15. Prepostavite da između izvornog i odredišnog računara postoje tri ruter. Ako zanemarimo fragmentaciju, preko koliko interfejsa će preći IP segment koji se pošalje od izvornog do odredišnog računara? Koliko tabela pretraživanja će se indeksirati da bi se datagram preneo od izvora do odredišta?
16. Prepostavimo da aplikacija svakih 20 ms proizvodi komade od po 40 bajtova podataka i da se svaki takav komad enkapsulira u TCP segment, a zatim u IP datagram. Koji procenat takvog datagrama pripada dodatnom opterećenju, a koji aplikacijskim podacima?
- i 17. Prepostavite da računar A šalje računaru B jedan TCP segment enkapsuliran u IP datagram. Kada računar B primi datagram, kako mrežni sloj računara B zna da treba taj segment (tj. korisne podatke datagrama) da preda TCP-u a ne UDP-u ili nekom drugom?
- \ 18. Prepostavite da ste kupili bežični ruter i povezali ga sa kablovskim modemom. Prepostavite takođe da vaš posrednik Internet usluga dinamički dodeli vašem povezanim uređaju (tj. bežičnom ruteru) jednu IP adresu. Takođe prepostavite da u kući imate pet PC-ja koji koriste 802.11 za bežično povezivanje sa bežičnim ruterom. Kako se IP adrese dodeljuju PC-jima? Da li bežični ruter koristi NAT? Zašto?
- j 19. Opишite razliku između polja zagлавja kod protokola IPv4 i IPv6. Da li postoje neka zajednička polja?
- \ 20. Kaže se da kada se IPv6 tuneluje kroz IPv4 rutere, IPv6 posmatra IPv4 tunele kao protokole sloja veze. Da li se slažete sa tim tvrdnjem? Zašto?

## ODELJAK 4.5

- j 21. Navedite razlike algoritama rutiranja prema stanju linkova i prema vektorima rastojanja.
- i 22. Objasnite kako je hijerarhijsko organizovanje Interneta pomoglo pri skaliranju na milione korisnika.
- ) 23. Da li je neophodno da svi autonomni sistemi koriste isti algoritam rutiranja unutar AS-a? Zašto?

## ODEUAK 4.6

- j 24. Razmotrite sliku 4.31. Uzmite prvo bitnu tabelu u D i prepostavite da D primi od .4 sledeću objavu:

| Odredišna mreža | Sledeći ruter | Broj skokova do odredišta |
|-----------------|---------------|---------------------------|
| Z               | C             | 10                        |
| W               | -             | 1                         |
| X               | -             | 1                         |
| ...             | ...           | ...                       |

Da li će se tabela u D promeniti? Ako hoće, kako?

25. Opišite razliku među objavama protokola RIP i protokola OSPF.
26. Popunite praznu liniju: RIP objave obično objavljaju broj skokova do različitih odredišta. Sa druge strane, BGP ažuriranja objavljaju \_\_\_\_\_ do različitih odredišta.
27. Zašto se na Internetu unutar i između AS-ova koriste različiti protokoli?
28. Zašto su pitanja politike isto toliko značajna u unutrašnjim protokolima AS-ova (kao što su OSPF i RIP) koliko i u protokolima između AS-ova (kakav je BGP)?
29. Definišite sledeće pojmove i opišite razlike između njih: *podmreža, prefiks i BGP ruta.*
30. Na koji način BGP koristi atribut NEXT-HOP? Kako koristi atribut AS-PATH?
31. Opišite na koji način mrežni administrator kod posrednika višeg reda implementira politike kada konfiguriše BGP.

## ODELJAK 4.7

32. Koja je značajna razlika između implementiranja difuznog upućivanja pomoću više jednoznačnih upućivanja i jednogjedinog difuznog upućivanja koje podržava mrežni ruter?
33. Da li su sledeće izjave tačne ili netačne za sve tri vrste implementiranja difuznog upućivanja (nekontrolisano plavljenje, kontrolisano plavljenje i difuzno emitovanje po sveobuhvatnom stablu)? Možete poći od pretpostavke da se paketi ne gube zbog prekoračenja privremenih memorija i da se svi paketi isporučuju na Sinku onim redosledom kojim su poslati.
- a. Čvor će primiti više kopija istog paketa.
- b. Čvor može da prosledi više kopija jednog paketa preko istog izlaznog linka.
34. Kada računar pristupa grupi za višezačno upućivanje da li mora da promeni svoju IP adresu grupe za višezačno upućivanje kojoj pristupa?
35. Koje uloge, igraju protokol IGMP i protokol za višezačno rutiranje u regionalnoj mreži?
36. Koja je razlika između zajedničkog stabla grupe i stabla od izvora u kontekstu višezačnog rutiranja?

## Problemi

1. Razmotrimo razloge za mreže sa datagramima i sa virtuelnim kolima i razloge protiv njih.

- Prepostavimo da se u mrežnom sloju ruteri nalaze u stresnim uslovima zbog kojih češće mogu da otkazu. Koje aktivnosti bi trebalo izvršiti na višem nivou u slučaju otkazivanja rutera? Da li je za taj slučaj bolje okruženje sa datagramima i sa virtuelnim kolima?
- Prepostavimo da bi u svrhu garantovanja nivoa performansi (na primer, kašnjenja) za celu putanju od izvora do odredišta mreža zahtevala da pošiljalac objavi svoju maksimalnu brzinu saobraćaja. Ako su prijavljena maksimalna brzina saobraćaja i postojeća objavljena brzina saobraćaja takve da nije moguće preneti saobraćaj od izvora do odredišta u zahtevanim granicama kašnjenja izvoru se ne dozvoljava da pristupi mreži. Da li bi se takva politika lakše postigla u okruženju sa datagramima ili sa virtuelnim kolima?

2. Razmotrite mrežu sa virtuelnim kolima. Prepostavite daje VC broj 16-bitno polje.

- Koji je maksimalan broj virtuelnih kola koja mogu da se prenesu preko linka?
- Prepostavite da centralni čvor određuje putanje i VC brojeve prilikom uspostavljanja konekcije. Prepostavite da se na svakom linku duž VC putanje koristi isti VC broj. Opisite kako bi centralni čvor utvrdio VC broj prilikom uspostavljanja konekcije. Da li je moguće da bude manje aktivnih virtuelnih kola od maksimuma definišanog pod (a) a da ipak nema slobodnih VC brojeva?
- Prepostavite da se dozvole različiti VC brojevi na svakom linku duž VC putanje. Prilikom uspostavljanja konekcije, pošto se utvrdi putanja sa kraja na kraj, opisite kako linkovi mogu decentralizovano da biraju VC brojeve i konfigurišu svoje tabele prosledivanja bez oslanjanja na centralni čvor.
- Osnovna tabela prosledivanja u mreži sa virtuelnim kolima ima četiri kolone. Šta znače vrednosti u tim kolonama? Osnovna tabela prosledivanja u mrežu sa datagramima ima dve kolone. Šta znače vrednosti u tim kolonama?
- Razmotrite mrežu sa virtuelnim kolima sa 2-bitnim poljem za VC broj. Prepostavite da mreža hoće da uspostavi virtuelno kolo preko Četiri linka: A, B, C i D. Prepostavite da svaki od ovih linkova trenutno već održava po dva virtuelna kola sa sledećim VC brojevima:

| Link A | Link B | Link C | Link D |
|--------|--------|--------|--------|
| 00     | 01     | 10     | 11     |
| 01     | 10     | 11     | 00     |

U odgovorima na sledeća pitanja imajte na umu da svako od postojećih virtuelnih kola prolazi samo kroz po jedan od navedenih linkova.

- Ako svako virtuelno kolo treba da koristi isti VC broj na svim linkovima duž putanje, koji VC broj bi mogao da se dodeli novom kolu?
- Ako svako virtuelno kolo može da ima različite VC brojeve na raznim linkovima duž putanje (pa tabela prosledivanja mora da obavlja prevođenje VC brojeva), koliko različitih kombinacija četiri VC broja (po jedan za svaki od četiri linka) bi se moglo upotrebiti?
- U tekstu smo koristili izraz *usluga sa konekcijom* za opise u transportnom sloju a *usluga konekcije* za mrežni sloj. Zbog čega postoji ta suptilna razlika u terminologiji?
- U odeljku 4.3 smo napomenuli da ne može biti čekanja u redu na ulazu ako je komutatorska mreža  $n$  puta brža od ulazne linije, pod pretpostavkom da  $n$  ulaznih linija ima identične brzine. Objasnite (recima) zastoje tako?
- Zamislite mrežu sa datagramima u kojoj se koristi 32-bitno adresiranje. Prepostavite da ruter ima četiri linka, numerisana od 0 do 3, a da pakete treba proslediti interfejsima linkova na sledeći način:

| Raspont odredišnih adresa                                                                              | Interfejs linka |
|--------------------------------------------------------------------------------------------------------|-----------------|
| <b>n</b> <b>inoooo</b> 00000000 00000000 00000000<br>do<br><b>11100000 lliiuu milili milili</b>        | 0               |
| <b>11100001</b> 00000000 00000000 <b>oooooooo</b><br>do<br><b>11 101001 00000000 l1I1l111 11111111</b> | 1               |
| <b>11101001</b> 00000001 00000000 00000000<br>do<br><b>1110100L tlIIUU UIIIU muni</b>                  | 2               |
| Inače                                                                                                  | 3               |

- Napravite tabelu prosledivanja sa četiri stavke koja će koristiti pravilo najdužeg prefiksa i prosledjivati pakete na pravilne interfejsne linkove.
- Opisite na koji način vaša tabela određuje odgovarajući interfejs linka za datagrame sa sledećim odredišnim adresama:

|          |          |          |          |
|----------|----------|----------|----------|
| 11001000 | 10010001 | 01010001 | 01010101 |
| 11100001 | 00000000 | 11000011 | 00 U1100 |
| 11100001 | 10000000 | 00010001 | 0U10111  |

- Zamislite mrežu sa datagramima u kojoj se koristi 8-bitno adresiranje. Prepostavite da ruter koristi pravilo najdužeg prefiksa i sledeću tabelu prosledivanja:

| Prefiks | Interfejs |
|---------|-----------|
| 00      | 0         |
| 01      | 1         |
| 10      | 2         |
| 11      | 3         |

Navedite raspon odredišnih adresa računara i broj adresa u rasponu za svaki od navedenih interfejsa. 9. Zamislite mrežu sa datagramima u kojoj se koristi 8-bitno adresiranje. Prepostavite da ruter koristi pravilo najdužeg prefiksa i sledeću tabelu prosleđivanja:

| Prefiks | Interfejs |
|---------|-----------|
| 1       | 0         |
| 11      | 1         |
| 111     | 2         |
| Inače   | 3         |

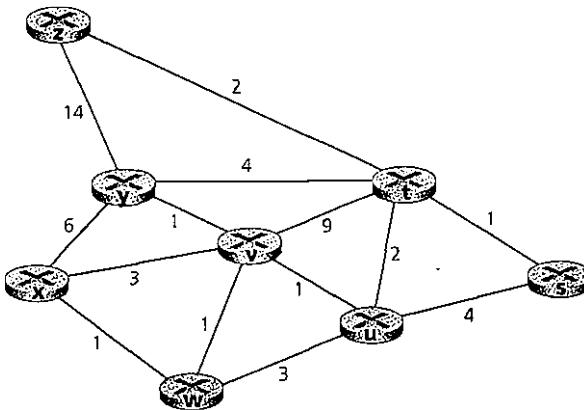
Navedite raspon odredišnih adresa računara i broj adresa u rasponu za svaki od navedenih, interfejsa.

10. Zamislite ruter koji povezuje tri podmreže: Podmrežu 1, Podmrežu 2 i Podmrežu 3. Prepostavite da svi interfejsi u tim podmrežama moraju da imaju prefiks 223.1.17/24. Takođe prepostavite da Podmreža 1 mora da podrži najviše 125 interfejsa a Podmreže 2 i 3 treba da podrži najviše po 60 interfejsa. Odredite tri mrežne adrese (oblika a.b.c/x) koje će zadovoljiti postavljene zahteve.
11. 11 odeljku 4.2.2 dat je primer tabele prosleđivanja (sa korišćenjem pravila najdužeg prefiksa). Prepišite tu tabelu ali u obliku a.b.c/x umesto u binarnom obliku.
12. U problemu 7 se zahtevalo da napravite tabelu prosleđivanja (sa upotreborom pravila najdužeg prefiksa). Prepišite tu tabelu ali u obliku a.b.c/x umesto u binarnom obliku.
13. Zamislite podmrežu sa prefiksom 101.101.101.64/26. Navedite primer adrese (oblika xxx.xxx.xxx.xxx) koja se može dodeliti u ovoj mreži. Prepostavite da jedan posrednik za Internet usluge poseduje blok adresa oblika 101.101.12S/17. Prepostavite da on hoće od tog-bloka da napravi četiri podmreže sa podjednakim brojem IP adresa. Koji će biti prefiksi (oblika a.b.c.d/x) te četiri podmreže.
14. Razmotrite topologiju sa slike 4.17. Označite tri podmreže sa računarima (počnite u smeru kazaljke na satu od 12:00) kao Mreže A, B i C. Označite podmreže bez računara kao Mreže D, E i F.
  - a. Dodelite mrežne adrese svakoj od ovih podmreža uz sledeća ograničenja: Sve adrese treba dodeliti iz 214.97.254/23; Podmreža A treba da ima

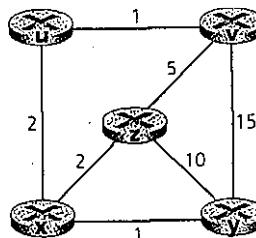
dovoljno adresa da podrži 250 interfejsa; Podmreža B treba da ima dovoljno adresa da podrži 120 interfejsa; Podmreža C treba da ima dovoljno adresa da podrži 120 interfejsa. Naravno, podmreže D, E i F moraju biti u stanju da podrže po dva interfejsa. Za svaku mrežu dodeljivanje treba dati u obliku a.b.c.d/x ili a.b.c.d/x - e.f.g.h.v.

- b. Na osnovu odgovora pod (a), napravite tabele prosleđivanja (uz pravilo najdužeg prefiksa) za svaki od tri ruteru.
15. Razmotrite slanje datagrama od 3000 bajtova u link čiji je MTU jednak 500 bajtova. Prepostavite da originalni datagram ima identifikaciju 422. Koliko fragmenata treba napraviti? Kakve su im karakteristike?
16. Prepostavite da su između izvornog računara A i odredišnog računara B datagrami ograničeni na 1500 bajtova (uključujući zaglavje). Ako prepostavimo daje zaglavje 20 bajtova, koliko bi datagrama bilo potrebno da se pošalje jedan MP3 od 4 miliona bajtova?
17. Razmotrite raspored mreže sa slike 4.20. Prepostavite daje posrednik dodelio ruteru adresu 126.13.89.67, a daje mrežna adresa matične mreže 192.168/16.
  - a. Dodelite adrese svim interfejsima u matičnoj mreži.
  - b. Prepostavite da svaki računar ima TCP konekcije u toku, sve sa portom 80 računara 128.13.9.40.86. Pripremite šest odgovarajućih stavki za NAT tabelu prevodenja,
18. U ovom problemu treba da analiziramo uticaj NAT-a na P2P aplikacije. Prepostavite da ravnopravni učesnik Arnold upitom otkrije da učesnik po imenu Bernard ima datoteku koju želi da preuzme. Prepostavite takođe da se-Bernard nalazi iza NAT-a a Arnold ne. Neka adresa NAT-a prema WAN-u bude 138.76.29.7 a interna IP adresa za Bernarda 10.0.0.1. Prepostavimo da NAT nije posebno konfiguriran za tu P2P aplikaciju.
  - a. Objasnite zašto učesnik Arnold ne može da pokrene TCP konekciju sa Bernardom iako zna daje WAN adresa NAT-a 138.76.29.7.
  - b. Prepostavite sada daje Bernard uspostavio TCP konekciju sa drugim učesnikom, Sindi, koja nije iza NAT-a, Prepostavite takođe daje Arnold saznao od Sindi da Bernard ima željenu datoteku i da Arnold može da uspostavi (ili je već uspostavio) TCP konekciju sa Sindi. Opišite na koji način Arnold može da upotrebi dve TCP konekcije (jednu od Bernarda do Sindi i drugu od Arnolda do Sindi) i obavesti Bernarda da pokrene direktnu TCP konekciju (tj. mimo Sindi) sa Arnoldem. Ta se tehnika ponekad naziva *okretanje konekcije*. Obratite pažnju na to da i pored toga Stoji Bernard iza NAT-a, Arnold može da upotrebi ovu direktnu TCP konekciju da zatraži datoteku a Bernard može tom konekcijom da isporuči datoteku.
19. Nadovežimo se na prethodni problem i prepostavimo da su i Arnold i Bernard iza NAT-ova. Pokušajte da osmislite tehniku kojom bi Arnold uspostavio TCP konekciju sa Bernardom bez posebnog konfiguriranja sanja NAT-a za određenu aplikaciju. Ako imate teškoće sa defmisanjem takve tehnike, objasnite zašto.

20. Na slici 4.25 navedite putanje od  $u$  do  $z$  koje ne sadrže petlje.
21. Razmotrite sledeću mrežu. Sa navedenim cenama linkova primenite Dijkstrin algoritam najkraće putanje i izračunajte najkraću putanju od  $x$  do svih ostalih mrežnih čvorova. Prikažite rad algoritma pomoću tabele slične tabeli 4.3.



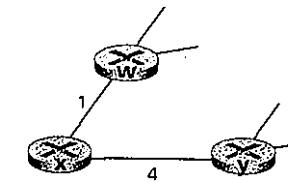
22. Razmotrite mrežu prikazanu u problemu 21. Pomoću Dijkstrinog algoritma u obliku sličnom tabeli 4.3,
- izračunajte najkraću putanju od  $s$  do svih ostalih mrežnih čvorova.
  - izračunajte najkraću putanju od  $i$  do svih ostalih mrežnih čvorova.
  - izračunajte najkraću putanju od  $u$  do svih ostalih mrežnih čvorova.
  - izračunajte najkraću putanju od  $v$  do svih ostalih mrežnih čvorova.
  - izračunajte najkraću putanju od  $w$  do svih ostalih mrežnih čvorova.
  - izračunajte najkraću putanju  $o \& y$  do svih ostalih mrežnih čvorova.
  - izračunajte najkraću putanju od  $z$  do svih ostalih mrežnih čvorova.
23. Razmotrite sledeću mrežu i prepostavite da svaki čvor na početku zna za cenu do svakog od svojih suseda. Uzmite algoritam vektora rastojanja i prikažite stavke za tabelu rastojanja u čvoru  $z$ .



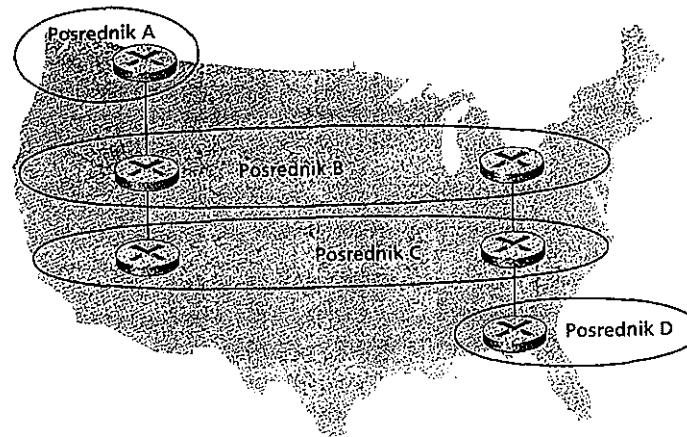
24. Razmotrite jednu opštu topologiju (tj. nemojte uzeti konkretnu gore prikazanu mrežu) i jednu sinhronu verziju algoritma vektora rastojanja. Prepostavite

da pri svakoj iteraciji čvor razmenjuje svoje vektore rastojanja sa susedima i prihvata njihove vektore rastojanja. Ako prepostavimo da algoritam u svakom Čvoru počinje od saznanja jedino o cennama do svojih neposrednih suseda, koji je maksimalan broj iteracija potreban da se distribuirani algoritam završi? Dokažite svoj odgovor.

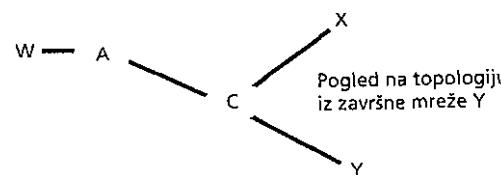
25. Razmotrite dole prikazani fragment mreže.  $x$  ima samo dva priključena suseda,  $w \setminus y$ .  $w$  ima najjeftiniju putanju do odredišta  $u$  (nije na slici) koja iznosi 5, a  $y$  ima najjeftiniju putanju do  $u$  koja iznosi 6. Nismo prikazali kompletne putanje iz  $w$  i  $y$  do  $u$  (i između  $w$  i  $y$ ). Sve cene linkova u mreži imaju isključivo celo-brojne pozitivne vrednosti.



- Navedite stavke vektora rastojanja od  $x$  do odredišta  $w$ ,  $y$  i  $u$ .
  - Navedite promenu cene linka za  $c(x, xv)$  ili  $c(x, y)$  takvu da  $x$  obavesti svoje susede o postojanju nove najjeftinije putanje prema  $u$  do koje je došlo nakon izvršavanja algoritma vektora rastojanja.
  - Navedite promenu cene linka za  $c(x, w)$  ili  $c(x, y)$  takvu da  $x$  neće obavestiti svoje susede o postojanju nove najjeftinije putanje prema  $u$  nakon izvršavanja algoritma vektora rastojanja.
26. Razmotrite topologiju sa tri čvora prikazanu na slici 4.27. Umesto cena linkova prikazanih na slici 4.27, cene linkova su  $c(x,y) = 5$ ,  $c(y,z) = 6$ ,  $c(z,x) = 2$ . Izračunajte tabele-rastojanja nakon koraka inicijalizacije i nakon svake iteracije jedne sinkrone verzije algoritma vektora rastojanja (kao što smo ranije uradili prilikom opisa slike 4.27).
27. Opišite kako se u BGP-u mogu otkriti petlje u putanjama.
28. Razmotrite mrežu koja je prikazana na početku sledeće strane. Posrednik B pruža usluge državne okosnice regionalnom posredniku A. Posrednik C pruža usluge državne okosnice regionalnom posredniku D. B i C su međusobno ravnopravno povezani na dva mesta i koriste BGP. Razmotrite saobraćaj koji se kreće od A do D. B će radije predati saobraćaj posredniku C na Zapadnoj obali (pa će C morati da snosi troškove prenošenja saobraćaja kroz celu zemlju), dok bi C radije preneo saobraćaj preko ravnopravne tačke na Istočnoj obali (tako da bi C snosio troškove saobraćaja kroz celu zemlju). Koji bi BGP mehanizam mogao da upotrebi C da bi predao saobraćaj iz A za D u zajedničkoj tački sa B na Istočnoj obali? Da biste odgovorili na ovo pitanje, inoraćete dobro da proučite specifikaciju BGP.



- 29 Na slici 4.39 razmotrite informacije o putanjama koje stižu u završne mreže W, X i Y. Na osnovu informacija dostupnih u W i X, kakav je pogled na topologiju mreže iz svakog od njih? Objasnite svoj odgovor. Pogled na topologiju iz mreže Y je sledeći.



30. Razmotrite mrežu sa osam Čvorova (gde su čvorovi označeni slovima od s do 2) iz problema 21. Prikažite stablo minimalnih cena koje počinje od s, a uključuje (kao završne računare) čvorove u, v, w i y. Svojim recima objasnite zašto vaše stablo jeste stablo minimalnih cena.
31. Razmotrite dva u osnovi različita načina za ostvarivanje difuznog upućivanja: emulacija pomoću jednoznačnog upućivanja i difuznog upućivanje pomoću mrežnog sloja (tj. pomoću ruteru) pod pretpostavkom da se za difuzno upućivanje pomoću mrežnog sloja koristi difuzno emitovanje po sveobuhvatnom stablu. Uzmimo slučaj sa jednim pošiljaocem i 32 primaoca. Prepostavite da je pošiljalac povezan sa primaocima jednim binarnim stablom ruteru. Kakva je za ovu topologiju cena slanja difuzno upućenog paketa u slučaju emulacije pomoću jednoznačnog upućivanja i u slučaju difuznog upućivanja pomoću mrežnog sloja? Ovde, kad god se paket (ili kopija paketa) šalje preko jednog linka nastupa „trošak“. Kakva topologija međusobnog povezivanja pošiljaoca,

primalaca i rutera dovodi do najveće razlike između ove dve cene? Možete uzeti koliko god hoćete ruteru.

32. Razmotrimo funkcionisanje algoritma inverznog prosleđivanja (RPF) sa slike 4.41. Pomoću iste topologije pronađite skup putanja od svih Čvorova do izvornog čvora A (i označite te putanje na grafu deblje osenčenim linijama kao na slici 4.41) tako da ako bi te putanje bile najjeftinije putanje, tada bi pod RPF-om čvor B primio od A, od C i od D po jedan primerak difuzne poruke iz A.
33. Razmotrite topologiju prikazanu na slici 4.41, gde svi linkovi imaju jedinične cene i prepostavite daje čvor E izvor difuznog upućivanja. Pomoću strelica kakve su prikazane na slici 4.41, ukažite na linkove preko kojih će se paketi prosleđivati pomoću RPF-a i linkove preko kojih se paketi neće prosleđivati ako se uzme da čvor E predstavlja izvor.
34. Razmotrite topologiju prikazanu na slici 4.43. i prepostavite da svi linkovi imaju jedinične cene. Prepostavite daje čvor C izabran za centar u algoritmu za višezačno rutiranje prema centru. Pod pretpostavkom da kada šalje poruku pristupanja u C svaki priključeni ruter koristi svoju najjeftiniju putanju do čvora C, nacrtajte dobijeno stablo za višezačno rutiranje prema centru. Da li je dobijeno stablo jedno stablo minimalnih cena? Objasnite svoj odgovor.
35. U odeljku 4.5.1 smo poučili Dijkstrin algoritam rutiranja prema stanju linkova koji izračunava jednoznačne putanje koje su pojedinačno najjeftinije od izvora do svih odredišta. Unija tih putanja mogla bi se posmatrati kao **stablo najjeftinijih jednoznačnih putanja** (ili stablo najkratkih jednoznačnih putanja, ako su cene svih linkova identične). Izgradnjom suprotnog primera dokažite da stablo najjeftinijih putanja *nije* uvek isto što i stablo sa minimalnim sveukupnim cenama.
36. Zamislite mrežu u kojoj su svi Čvorovi povezani sa tri druga čvora. U jednom vremenskom koraku čvor može da primi difuzno emitovane pakete od svojih suseda, duplira pakete i pošalje ih svim svojim susedima (osim čvora od kojeg je primio konkretan paket). U sledećem koraku, susedni čvorovi mogu da ih prime, dupliraju, proslede itd. Prepostavite da se u takvoj mreži za difuzno emitovanje koristi nekontrolisano plavljenje. Koliko će primeraka difuzno emitovanog paketa biti preneto u koraku  $t$ , pod pretpostavkom da je u koraku 1 prenet samo jedan difuzno emitovani paket iz izvornog čvora do njegova tri suseda.
37. Videli smo u odeljku 4.7 da ne postoji protokol mrežnog sloja koji bi se mogao upotrebiti za prepoznavanje računara koji učestvuju u grupi za višezačno upućivanje. Polazeći od toga, kako mogu aplikacije za višezačno upućivanje da saznaju identitet računara koji učestvuju u grupi za višezačno upućivanje?
38. Projektujte protokol (napravite opis u obliku pseudo koda) na aplikacijskom nivou koji evidentira adrese računara za sve računare koji učestvuju u jednoj grupi za višezačno upućivanje. Konkretno navedite mrežnu uslugu (jednoznačnu ili višezačnu) koju vaš protokol koristi i navedite da li vaš protokol šalje poruke unutar ili van normalnog toka (u odnosu na tok aplikacijskih podataka medu učesnicima u grupi za višezačno upućivanje), i zašto to čini.

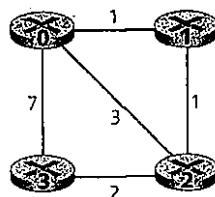
39. Koja je veličina adresnog prostora za višezačno upućivanje? Prepostavite sada da dve različite grupe za višezačno upućivanje biraju adresu na slučajan način. Kolika je verovatnoća da će izabrati istu adresu? Prepostavite sada da 1000 grupa za višezačno upućivanje istovremeno bira adresu grupe na slučajan način. Kolika je verovatnoća da dođe do preklapanja?

## Teze za diskusiju

1. Pronadite tri kompanije koje trenutno prodaju ruterske proizvode velike brzine. Uporedite proizvode.
2. Upotrebite uslugu whois u američkom registru za Internet brojeve (<http://www.arin.net/whois>) i utvrđite blokove IP adresa za tri univerziteta. Može li se usluga whois iskoristiti za pouzdano utvrđivanje lokacije konkretnе IP adrese?
3. Da li je moguće napisati klijentski program ping (sa ICMP porukama) u Javi? Zašto?
4. U odeljku 4.7 napomenuli smo da uvođenje protokola IPv6 za sada ide sporo. Zašto ide sporo? Šta bi bilo potrebno da se to uvođenje ubrza?
5. Opišite neke probleme koje NAT-ovi postavljaju pred IPsec bezbednost (videti [Phifer 2000]).
6. Prepostavite da AS-ovi X i Z nisu direktno povezani, već da su povezani preko AS-a Y. Nakon toga prepostavite da X ima ugovor o ravnopravnosti sa Y, a da Y ima isti takav ugovor sa Z. Na kraju, prepostavite da Z prihvata tranzit svakog saobraćaja iz Y, ali ne prihvata tranzit saobraćaja iz X. Da li BGP omogućava da Z implementira takvu politiku?
7. U odeljku 4.7 prikazali smo niz aplikacija višezačnih upućivanja. Koja od tih aplikacija je primerena za minimalistički Internetov model usluge višezačnog upućivanja? Zašto? Koje aplikacije nisu posebno primerene za ovaj model usluge?

## Programerski zadatak

U ovom programerskom zadatu pisaćete „distribuirani“ skup procedura koje implementiraju distribuirano asinhrono rutiranje pomoću vektora rastojanja za sledeću mrežu.



Treba da napišete sledeće rutine koje će se „izvršavati“ asinhrono u pripremljenom okruženju za ovaj zadatak. Za čvor 0 treba da napišete rutine:

- ◆ *rtinit0()*. Ova rutina će se pozvati jednom na početku emulacije. Rutina *rtinit0* nema argumenata. Ona treba da inicializuje vašu tabelu rastojanja u čvoru 0 direktnim cenama 1, 3 i 7 do čvorova 1, 2 odnosno 3. Na gornjoj slici su svi linkovi dvosmerni, a cene u oba smera identične. Nakon inicializiranja tabele rastojanja i ostalih struktura podataka potrebnih za rutine vašeg čvora 0, trebalo bi direktno povezanim susedima (u ovom slučaju, 1, 2 i 3) poslati cene najjeftinijih putanja do svih ostalih čvorova u mreži. Ove informacije o minimalnim cenama šalju se susednim čvorovima u paketu za ažuriranje rutiranja tako što se pozove rutina *tolayer2Q* kao sto je opisano u potpunom opisu zadatka. Format paketa za ažuriranje rutiranja takođe se nalazi u potpunom opisu zadatka.
- ◆ *rtupdate0(struct rtpkt \*rcvdpkt)*. Ova rutina će se pozivati kada čvor 0 primi paket rutiranja od nekog direktno povezanog suseda. Parametar *\*rcvdpkt* je pokazivač na primljeni paket. Rutina *rtupdate0O* predstavlja „srž“ algoritma vektora rastojanja. Vrednosti u paketu za ažuriranje mrfiranja dobijenom iz nekog drugog čvora / predstavljaju cene trenutno najjeftinijih putanja od čvora / do svih ostalih čvorova u mreži. Rutina *rtupdate0Q* koristi primljene vrednosti za ažuriranje vlastite tabele rastojanja (kao sto je navedeno u algoritmu vektora rastojanja). Ako se zbog ovog ažuriranja promeni njegova minimalna cena do nekog drugog čvora, čvor 0 o toj promeni minimalne cene obaveštava svoje direktno povezane susede tako što im šalje paket rutiranja. Sigurno se sećate da u algoritmu vektora rastojanja samo direktno povezani čvorovi razmenjuju pakete rutiranja. Prema tome, čvorovi 1 i 2 će međusobno komunicirati, a čvorovi 1 i 3 neće.

Slične rutine definisane su za čvorove 1, 2 i 3. Prema tome, ukupno ćete napisati osam procedura: *rtinitQ()*, *rtinitl()*, *rtinit2()*, *rtinit3()*, *rtupdateO()*, *rtupdate1()*, *rtupda(e2())* i *rtupdate3()*. Te rutine će sve zajedno da implementiraju distribuirano, asinhrono izračunavanje tabele rastojanja za topologiju i cene prikazane na slici sa prethodne strane.

Potpuni opis programskega zadatka, kao i kod u jeziku C koji će vam biti potreban za pravljenje simuliranog hardversko/softverskog okruženja možete naći na adresi <http://www.awl.com/kurose-ross>. Dostupna je takođe verzija ovog zadatka u jeziku Java.

## Laboratorijski zadaci

Na veb lokaciji koja pripada ovom udžbeniku: <http://www.awl.com/kurose-ross> nači ćete dve Ethereal vežbe za ovo poglavlje. U prvoj vežbi ispitujete se rad protokola IP, a posebno format IP datagrama. U drugoj vežbi istražujete se protokol ICMP u komandama- ping i traceroute.

## INTERVJU

### Vinton G. Serf

Vihton G. Serf je viši potpredsednik odeljenja za arhitekturu i tehnologiju u VWorldCom. Poznal je kao jedan od projektnata protokola TCP/IP i arhitekture Interneta. Kao potpredsednik u MCI Digital Information Services od 1982. do 1986. godine, on je predvodio uspostavljanje sistema MCI Mail, prve komercijalne usluge za e-poštu povezane sa Internetom. Dok je od 1976. do 1982. godine bio u agenciji za napredne istraživačke projekte ministarstva odbrane SAD (*Defense Advanced Research Projeds Ageocy*, DARPA), njegova uloga je bila do vodi razvoj Interneta i tehnologija bezbednosti i paketa podataka vezanih za Internet. Vinton je magistriro matematiku na univerzitetu Stanford, a doktorirao računarske nauke na univerzitetu UCLA.

Šta vas je navelo na specijalizaciju *h oblasti umrežavanja?*

Krajem '60-tih godina radio sam kao programer na univerzitetu UCLA. Moje postavljenje podržavala je Agencija za napredne istraživačke projekte ministarstva odbrane SAD (koja se tada nazivala ARPA, a sada DARPA). Radio sam u laboratoriji profesora Leonarda Klajnroka u Centru za mrežna merenja novostvorenog ARPANET-a. Prvi čvor mreže ARPANET instaliran je na univerzitetu UCLA septembra 1969. godine. Ja sam bio zadužen za programiranje jednog računara koji se koristio za evidentiranje informacija o performansama ARPANET-a i za izveštavanje o tim informacijama radi poređenja sa matematičkim modelima i predviđenim performansama mreže.

Nekoliko drugih apsolvenata i ja bili smo zaduženi za rad na takozvanim „protokolima za nivo računara“ ARPANET-a - na procedurama i formatima koji bi omogućili saradnju različitih vrsta računara u mreži. To je bilo očaravajuće istraživanje (za mene) novog sveta distribuiranog računarstva i komunikacije.

Kada ste projekovali protokol IP, da li ste zamišljali da će jednog dana biti toliko sveprisutan?

Kada smo Bob Kan i ja počeli na tome da radimo 1973. godine, mislim da smo uglavnom razmišljali o glavnom pitanju: kako da obezbedimo saradnju različitih paketskih mreža pod pretpostavkom da ne možemo da menjamo same mreže. Nadali smo se da ćemo pronaći način koji će omogućiti proizvoljnom skupu paketskih mreža da saraduju na transparentan način tako da će računari moći da komuniciraju sa kraja na kraj bez potrebe da učestvuju u bilo kakvim prevodenjima. Mislim da smo znali da je reč o moćnoj tehnologiji koja će moći da se širi, ali sumnjam da smo imali predstavu kakav će biti svet sa milionima računara povezanih u Internet.

Kakvo je vaše mišljenje o budućnosti umrežavanja i Interneta? Po vašem mišljenju, koji su glavni izazovi i prepreke u njihovom daljem razvoju?

Ja verujem da će sam Internet, i mreže uopste, i dalje da se šire. Već postoje ubedljivi dokazi da će na Internetu postojati milijarde uređaja sposobljenih za Internet, uključujući sprave kao što su mobilni telefoni, frižideri, PDA-ovi, kućni serveri, televizori, kao i ubičajeni laptopovi, serveri itd. Velike izazove predstavljaju podrška za mobilnost, trajanje baterija, kapacitet linkova za pristup mreži i mogućnost proširenja optičkog umrežavanja bez ograničenja. Određivanje međuplanetarnog proširenja Interneta je projekat na kojem sam intenzivno angažovan u laboratoriji Jet Propulsion. Moraćemo da predemo sa IPv4 (32-bk-nog adresiranja) na 3PvC (128 bitova). Spisak je dugačak!

Ko vas je profesionalno inspirisao?

Moj kolega Bob Kan, moj mentor Džerald Estrin, moj najbolji prijatelj Stiv Kroker (sreli smo se u gimnaziji i on mi je prvi pokazao računare 1960!); i hiljade inženjera koji danas nastavljaju da razvijaju Internet.

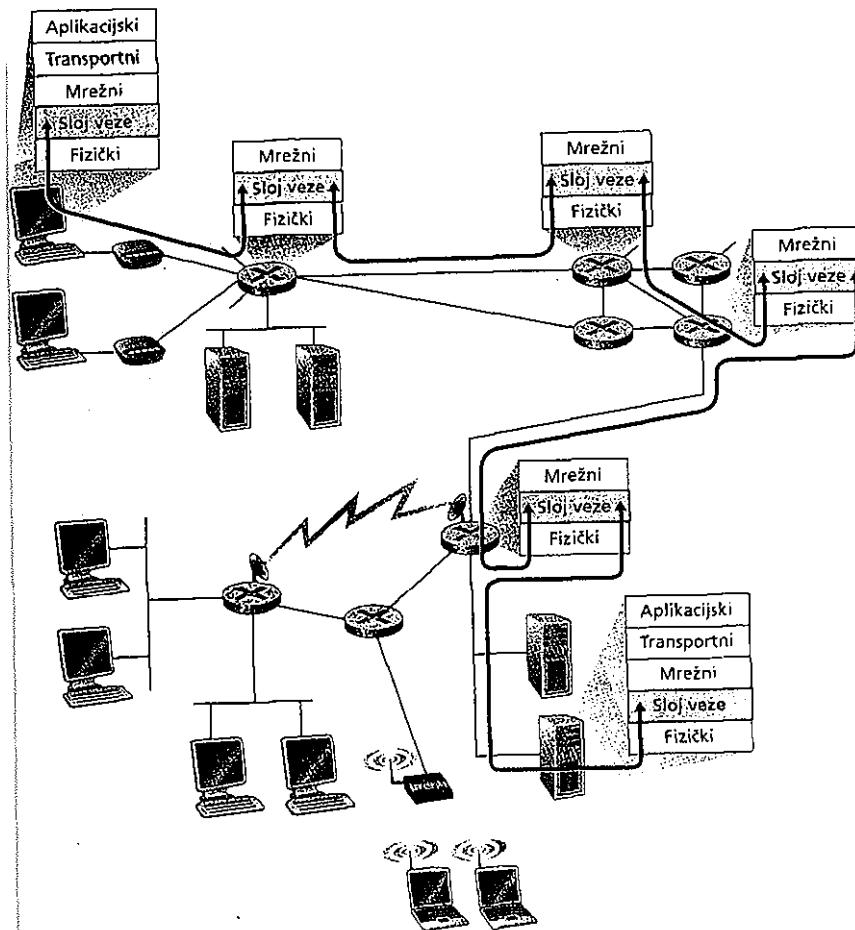
Da li imate neki savet za studente koji danas ulaze u domen umrežavanja i Interneta?

Razmišljajte izvan ograničenja postojećih sistema - zamislite šta bi moglo da se ostvari; a zatim obavite težak posao smišljanja kako da se to dostigne polazeći od trenutnog stanja stvari. Usudujte se da sanjate: nekoliko kolega i ja u laboratoriji Jet Propulsion radimo na projektu međuplanetarnog proširenja zemaljskog Interneta. Ta implementacija može da potraje na desetine godina, ali da parafraziram: „Čovek treba da stremi dalje od onog sto doseže, jet čemu inače služe nebesa.“

# Sloj veze podataka i lokalne mreže računara

U prethodnom poglavlju saznali ste da sloj mreže obezbeđuje komunikacionu uslugu između dva matična računara. Kao što je prikazano na slici 5.1, ova komunikaciona putanja sastoji se od niza komunikacionih linija koji počinje od izvornog računara, prolazi kroz niz rutera i završava se na odredišnom računaru. Kako nastavljamo da prolazimo kroz familiju protokola, od sloja mreže do sloja veze podataka,:prirodno je da se zapitamo kako se paketi šalju preko pojedinačnih linija, unutar komunikacione putanje od tačke do tačke. Kako se datagrami mrežnog sloja enkapsuliraju u okvire sloja veze podataka za prenos preko jedne linije? Da li protokoli sloja veze podataka obezbeđuju pouzdan prenos podataka od jednog do drugog rutera? Mogu li različiti protokoli sloja veze podataka da se upotrebe u različitim linijama duž komunikacione putanje? Na ova, kao i na druga važna pitanja, odgovoricećemo u ovom poglavlju.

Objašnjavajući sloj veze, otkrićemo da postoje dve fundamentalno različite vrste kanala. Prva vrsta sastoji se od difuznih kanala, koji su uobičajeni u lokalnim računarskim mrežama (LAN), bežičnim LAN-ovima, satelitskim mrežama i mrežama sa pristupom preko hibridnih optičkih kablova (HFC). Kod difuznog-kanala,,Veći broj računara povezan je na istu komunikacionu liniju i potreban je tzv. protokol za



**Slika 5.1** • Sloj veze podataka

pristup medijumu da bi se uskladili prenosi i izbegle kolizije. Druga vrsta kanala je komunikaciona linija od tačke do tačke, kao na primer između dva rutera ili »zmedju modema« stanu i rutera posrednika za Internet usluge (ISP). Uskladivanje pristupa u liniji od tačke do tačke je jednostavno, ali ipak postoje mnoga pitanja, koja obuhvataju okvire, pouzdan prenos podataka, otkrivanje grešaka i kontrolu toka.

U ovom poglavlju ćemo istražiti više značajnih tehnologija sloja veze podataka. Detaljnije ćemo razmotriti Ethernet, daleko najrasprostranjeniju tehnologiju lokalnih računarskih mreža. Pogledaćemo takođe PPP, protokol pogodan za kućne računare.

Iako Wi-Fi, i uopšte bežični LAN-ovi, svakako spadaju u tematiku u vezi sa slojem veze podataka, njih ne objašnjavamo u ovom poglavlju. Razlog leži u tome što je Wi-Fi značajna tema - VVi-Fi revolucija dramatično menja način kako ljudi pristupaju i koriste Ethernet. Zato je Wi-Fi detaljno objašnjen u poglavlju 6, koje je posvećeno bežičnim računarskim mrežama i mobilnosti.

## 5.1 Sloj veze podataka: uvod i usluge

Hajde da počnemo sa nekom korisnom terminologijom. U ovom poglavlju ćemo računare i rutere jednostavno zvati **čvorovi** (*nodes*), i zato nam, kao što ćemo uskoro videti, neće biti posebno značajno da lije čvor ruter ili računar. Takođe, komunikacione kanale koji povezuju susedne čvorove duž komunikacione putanje, zvaćemo **linkovi**. Dakle, da bi se datagram preneo od izvornog do odredišnog računara, on mora da pređe preko svakog *pojedinačnog linka* u putanji od tačke do tačke. Na datom linku, predajni čvor enkapsulira datagram u okvir i prenosi okvir u link; prijemni čvor prima okvir i iz njega izvlači datagram.

### 5.1.1 Usluge koje obezbeđuje sloj veze

Protokol sloja veze podataka koristi se za prenos datagrama duž pojedinačnog linka. **Protokol sloja veze** definiše format paketa koji se razmenjuju između čvorova na krajevima linka, kao i aktivnosti koje ti čvorovi preduzimaju prilikom predaje i prijema tih paketa. Setite se iz poglavlja 1 da se jedinice podataka koje se razmenjuju pomoću protokola sloja veze podataka zovu **okviri** i daje uobičajeno da svaki okvir ovog sloja enkapsulira jedan datagram mrežnog sloja. Kao što ćemo uskoro videti, aktivnosti koje preduzima protokol sloja veze podataka prilikom slanja i prijema okvira obuhvataju otkrivanje greške, ponovni prenos, kontrolu toka i slučajan pristup. Primere protokola sloja veze podataka možete naći u mrežama kao što su Ethernet, bežične lokalne računarske mreže 802.11 (poznate kao i Wi-Fi), Token Ring i PPP; u mnogim situacijama i ATM može da se posmatra kao protokol sloja veze podataka. Navedene protokole ćemo detaljnije objasniti u drugoj polovini ovog poglavlja.

Dok mrežni sloj treba da prenosi segmente transportnog sloja od tačke do tačke, od izvornog do odredišnog računara, protokol sloja veze podataka treba da prenosi datagrame mrežnog sloja od čvora do čvora, preko *jednog linka* u putanji. Značajna karakteristika sloja veze podataka je da datagram može da se prenosi različitim protokolima veze podataka na različitim linkovima duž putanje. Na primer, datagram može da se prenosi pomoću Etherneta na prvom linku, pomoću PPP-a na posle-dnjem linku i pomoću WAN protokola sloja veze na svim linkovima između njih.

Važno je zapaziti da usluge koje pružaju različiti protokoli veze podataka i same mogu da budu različite. Na primer, protokol sloja veze može ili ne može da obezbedi pouzdanu isporuku. Prema tome, mrežni sloj mora da bude sposoban da izvrši svoj zadatak od tačke do tačke, u prisustvu heterogenog skupa pojedinačnih usluga sloja veze.

Da bismo stekli uvid u sloj veze i u to kakav je njegov odnos prema mrežnom sloju, hajde da razmotrimo analogiju sa prevozom. Zamislite putničku agenciju koja planira turističko putovanje od Prinstona, u Nju Džerziju, do Lozane u Švajcarskoj. Pretpostavite da agencija odluči daje za turistu najpogodnije da ide limuzinom od Prinstona do aerodroma JFK, zatim avionom od aerodroma JFK do aerodroma u Ženevi i najzad vozom od aerodroma u Ženevi do železničke stanice u Lozani. Kada je putnička agencija napravila te tri rezervacije, odgovornost kompanije za iznajmljivanje limuzina u Prinstonu je da preveze turistu od Prinstona do aerodroma JFK; na vazduhoplovnoj kompaniji je da ga preveze turistu od JFK-a do Ženeve; a odgovornost švajcarske železnice je da on stigne od Ženeve do Lozane. Svaki od tri segmenta putovanja je „direktan“ između dve „susedne“ lokacije. Zapazite da svakim od tri segmenta putovanja upravlja različita kompanija i da koristi sasvim različit način prevoza (limuzina, avion i voz). Mada su načini prevoza različiti, svaki od njih pruža osnovnu uslugu prenosa putnika od jedne do druge susedne lokacije. U ovoj analogiji sa prevozom, turista je analogan sa datagramom, svaki segment prevoza je analogan sa komunikacionim linkom, način prevoza je analogan sa protokolom sloja veze podataka, a putnička agencija koja planira putovanje je analognog sa protokolom rutiranja.

Iako je osnovna funkcija usluge svakog sloja veze podataka da „prenosi“ datagram od jednog do drugog susednog čvora preko jednog komunikacionog linka, detalji usluge zavisiće od specifičnog protokola koji se koristi. Protokol sloja veze podataka može da pruži sledeće usluge:

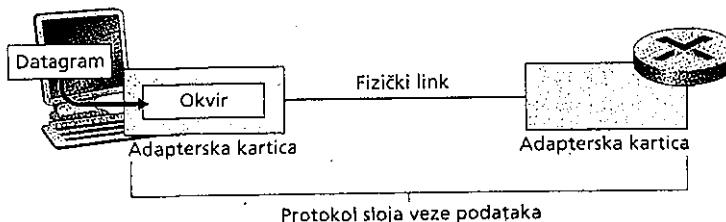
- ◆ *Pravljenje okvira.* Gotovo svi protokoli sloja veze podataka enkapsuliraju svaki datagram mrežnog sloja u okvir pre njegovog slanja na link. Okvir se sastoji od polja za podatke, u koje se stavlja datagram mrežnog sloja i izvesnog broja polja zaglavljiva. (Okvir može takođe da uključi završna polja; međutim, mi ćemo i polja zaglavljiva i završna polja zvati poljima zaglavljiva.) Strukturu okvira određuje protokol veze podataka. Kada u drugoj polovini ovog zaglavljiva budemo ispitivali određene protokole sloja veze podataka, videćemo više različitih formata okvira.
- ◆ *Pristup linku (LinkAccess).* Protokol Media Access Control (MAC) definiše pravila po kojima se okvir prenosi na link. Za linkove od tačke do tačke, koji imaju jednog pošiljaoca u jednom kraju i jednog primaoca na drugom kraju, protokol MAC je jednostavan (ili čak ine postoji) - pošiljalac može da šalje okvir kad god je link slobodan. Zanimljiviji slučaj nastaje kada više čvorova deli jedan difuzni link - to je tzv. problem višestrukog pristupa. Ovde MAC služi da koordinira prenose okvira iz više čvorova; protokole višestrukog pristupa detaljno ćemo objasniti u odeljku 5.3.

- ◆ *Pouzdana isporuka.* Kada protokol sloja veze podataka obezbeđuje pouzdanu uslugu isporuke, on garantuje da će preneti svaki datagram mrežnog sloja preko linka bez greške. Setite se da izvesni protokoli transportnog sloja (kao stoje TCP) takođe obezbeđuju pouzdanu uslugu isporuke. Slično usluži pouzdane isporuke transportnog sloja, usluga pouzdane isporuke sloja veze podataka postiže se pomoću potvrda i ponovnih prenosa (pročitajte odeljak 3.4). Usluga pouzdane isporuke obično se koristi za linkove koji su skloni velikoj učestanosti grešaka, kao što su bežični linkovi, u cilju lokalnog ispravljanja grešaka - na linku gde se greška pojavljuje - radije nego da se prisilno ponavlja prenos od tačke do tačke pomoću protokola transportnog ili aplikacijskog sloja. Međutim, usluga pouzdane isporuke može da se posmatra kao nepotrebno dodatno opterećenje za linkove sa malim greškama, kao što su optički, koaksijalni i mnogi linkovi sa upredenim paricama bakarnih provodnika. Iz tog razloga, na mnogim ožičenim linkovima nije obezbeđena usluga pouzdane isporuke.
- ◆ *Kontrola toka (flow control).* Čvorovi na svakom kraju linka imaju ograničen kapacitet privremenog smeštanja okvira. To je potencijalni problem, zato što prijemni čvor može da prima okvire brzinom većom od one kojom je u stanju da ih obraduje. Bez kontrole toka, privremena memorija prijemnika mogla bi da se prepuni, a okviri koji zatim dolaze da se izgube. Slično transportnom sloju, protokol sloja veze podataka obezbeđuje kontrolu toka kako bi sprečio predajni čvor najednom kraju linka da prepuni prijemni čvor na drugom kraju linka.
- ◆ *Otkrivanje greške.* Prijemnik čvora može pogrešno da odluči da je bit u nekom okviru nula kada je poslat kao jedinica i obratno. Do takvih grešaka dolazi usled slabljenja signala i elektromagnetskih smetnji. S obzirom na to da nije potrebno da se prosleduje datagram sa greškom, mnogi protokoli sloja veze podataka imaju mehanizam za otkrivanje prisustva jedne greške ili više grešaka. To se radi tako što predajni čvor postavlja u okvir i bitove za otkrivanje greške, a prijemni čvor proverava te bitove, tj. da traga za greškama. Otkrivanje greške je uobičajena usluga među protokolima sloja veze podataka. Setite se iz poglavlja 3 i 4, da transportni i mrežni slojevi Interneta takođe obezbeđuju ograničen oblik otkrivanja greške. Otkrivanje greške u sloju veze podataka obično je složenije i implementirano je u hardveru.
- ◆ *Ispravljanje greške.* Ispravljanje greške slično je njenom otkrivanju, izuzev što prijemnik može ne samo da otkrije da lije došlo do greške u okviru, nego takođe i da tačno odredi gde se u okviru ona pojavila (pa samim tim i da ima mogućnost da je ispravi). Neki protokoli (kao što je ATM) obezbeđuju ispravljanje grešaka samo u zaglavljiju paketa, a ne za ceo paket. Otkrivanje i ispravljanje grešaka objasnićemo u odeljku 5.2.
- ◆ *Poludupleks i puni dupleks.* Kod punog dupleksnog prenosa, svaki čvor, na bilo kom kraju linka, može da predaje i prima pakete isto vreme. Kod poludupleksnog prenosa, čvor ne može da predaje i prima u isto vreme.

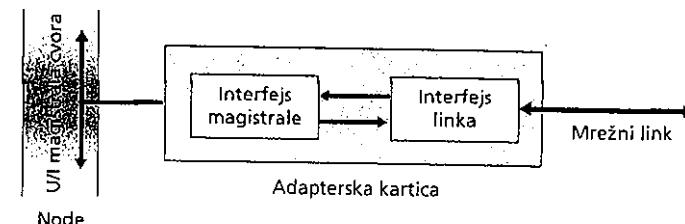
Kao Stope ranije rečeno, mnoge usluge koje obezbeđuje sloj veze podataka paralelne su sa uslugama koje obezbeđuje transportni sloj. Na primer, i sloj veze podataka i transportni sloj mogu da obezbede pouzdanu isporuku. Iako su mehanizmi koji se koriste za obezbeđivanje pouzdane isporuke u ta dva sloja slični (procitajte odeljak 3.4), usluge pouzdane isporuke ipak nisu iste. Transportni protokol obezbeđuje pouzdanu isporuku između dva procesa od tačke do tačke; pouzdan protokol sloja veze podataka obezbeđuje uslugu pouzdane isporuke između dva Čvora povezana jednim linkom. Slično tome, protokoli i sloja veze podataka i transportnog sloja, mogu da obezbede kontrolu toka i otkrivanje grešaka; opet, kontrola toka u protokolu transportnog sloja obezbeđuje se od tačke do tačke, dok se u protokolu sloja veze podataka ona obezbeđuje od čvora do susednog čvora.

### 5.1.2 Komuniciranje adaptera

Za dati komunikacioni link, protokol sloja veze podataka je u većini slučajeva implementiran u adapteru. Adapter je ploča (ili PCMCIA kartica) koja obično sadrži RAM, DSP čipove, interfejs magistrale računara i interfejs linka. Uobičajeno je da se adapteri zovu mrežne interfejsne kartice ili NIC. Kao stope prikazano 'na slici 5.2, mrežni sloj u prednjem čvoru (odnosno računaru ili ruteru) prosleđuje datagram adapteru, koji upravlja prednjom stranom komunikacionog linka. Adapter enkapsulira datagram u okvir i zatim predaje okvir na komunikacioni link. Na drugoj strani, prijemni adapter prima ceo okvir, izvlači datagram i prosleđuje ga mrežnom sloju. Ako protokol sloja veze podataka obezbeđuje otkrivanje greške, predajni adapter postavlja bitove za otkrivanje greške, a prijemni adapter proverava da li ima grešaka. Ako protokol sloja veze podataka obezbeđuje pouzdanu isporuku, tada su mehanizmi za pouzdanu isporuku (na primer, sekvenca brojeva, tajmer i potvrde) potpuno implementirani u adapterima. Ako protokol sloja veze podataka obezbeđuje slučajan pristup (procitajte odeljak 5.3), tada je i protokol slučajnog pristupa potpuno implementiran u adapterima.



**Slika 5.2** ♦ Protokol sloja veze podataka za komunikacioni link implementiran je u adapterima na dva kraja linka.



**Slika 5.3** ♦ Adapter je poluautonomna jedinica.

Adap-ter je poluautonomna jedinica. Na primer, adapter može da primi okvir, odredi da li je on pogrešan i odbaci ga, a da o tome ne obavesti čvor koji ga je poslao. Kada primi okvir, adapter će poslati signal prekida svom računari samo ako želi da prosledi datagram mrežnom sloju. Slično tome, kada čvor prosleđuje datagram adapteru naniže, on mu poverava sav posao oko prenosa datograma preko tog linka. Iako je adapter poluautonoman, on nije potpuno autonomna jedinica. Mada smo na slici 5.3 prikazali adapter kao posebnu „ kutiju”, on je obično smešten u istom fizičkom kućištu sa ostalim komponentama čvora, deli sa njim napajanje i magistrale i, na kraju, nalazi se pod kontrolom čvora.

Kao stope prikazano na slici 5.3, glavni sastavni delovi adaptera su interfejs magistrale i interfejs linka. Interfejs magistrale odgovoran je za komunikaciju sa čvorom u kome se adapter nalazi, prenoseći podatke i upravljačke informacije. Interfejs linka odgovoran je za implementiranje protokola sloja veze podataka. Pored stavljanja datograma u okvir i njihovog izdvajanja iz okvira, on može da obezbedi otkrivanje greške, da implementira protokol slučajnog pristupa, kao i da obezbedi i druge potrebne funkcije. On takođe obuhvata i elektronska kola za predaju i prijem električnih signala sa linka. Kod popularnih tehnologija, kao stope Ethernet, interfejs linka implementiran je pomoću skupa čipova koji se mogu kupiti na tržištu široke potrošnje. Iz tog razloga, adapteri za Ethernet i Wi-Fi su neverovatno jeftini - često koštaju manje od 20 US\$. Ako posetite web stranicu za adaptore firme 3COM, možete saznati više o arhitekturi adaptera za Ethernet od 10, 100 i 1000 Mb/s i za ATM od 155 Mb/s [3Com 2004].

## 5.2 Tehnike za otkrivanje i ispravljanje grešaka

U prethodnom odeljku napisali smo da su otkrivanje i ispravljanje grešaka na nivou bita - otkrivanje i ispravljanje oštećenja bitova unutar okvira sloja linka poslatog iz jednog čvora u drugi sa njim fizički povezani čvor - dve usluge koje često obezbeđuje sloj veze podataka. U poglavljiju 3, videli smo da se usluge otkriva-

nja i ispravljanja grešaka često nude i u transportnom sloju. U ovom odeljku, ispitaćemo nekoliko najjednostavnijih tehnika koje mogu da se upotrebe za otkrivanje i, u nekim slučajevima, ispravljanje grešaka. Potpuna teretska obrada i implementacija ove tehnike je sama po sebi predmet mnogih radova (na primer, [Schvartz 1980] ili [Bertsekas 1991]), pa ćemo sa na njoj ovde samo kratko zadržati. Naš cilj je da razvijemo intuitivan osećaj za mogućnosti koje imaju tehnike za otkrivanje i ispravljanje grešaka i da vidimo kako rade jednostavne tehnike i kako se one koriste u praksi.

Na slici 5.4 ilustrovana je postavka koju ćemo koristiti u našem proučavanju. Na strani prednjog čvora, podatak  $D$  koji treba da se zaštiti od grešaka bitova, proširen je je bitovima za otkrivanje i ispravljanje grešaka ( $EDC$ ). Podatak koji treba da se zaštiti po pravilu ne uključuje samo datagram koji je prosleden naniže iz mrežnog sloja, nego i informaciju adresiranja na nivou linka, brojke sekvence i druga polja u zaglavju okvira veze podataka. 1  $D$  i  $EDC$  šalju se ka prijemniku unutar okvira linka. Kod prijemnog čvora primaju se sekvence bitova  $D'$  i  $EDC'$ . Zapazite da  $D'$  i  $EDC$  mogu da se razlikuju od originalnih  $D$  i  $EDC$ , kao rezultat promene bitova u prenosu.

Izazov za prijemnik je da otkrije da li je  $D'$  isti kao originalni  $D$  ili nije, imajući u vidu da je primio samo  $D'$  i  $EDC$ . Precizno izražavanje po pitanju prijemnikove odluke na slici 5.4 je značajno (Mi pitamo da li je otkrivena greška, a ne da li se greška dogodila!). Tehnike za otkrivanje i ispravljanje grešaka dozvoljavaju prije-

mniku da ponekad, ali ne uvek, otkrije da su se pojavile greške bitova. To znači da, Čak i uz upotrebu bitova za otkrivanje grešaka, postojeće i dalje mogućnost da se pojave neotkrivene greške bitova, odnosno da prijemnik nije u stanju da otkrije da sadržaj primljene informacije ima pogrešne bitove. Posledica toga je da prijemnik isporučuje neispravan datagram mrežnom sloju, ili da zaključuje daje sadržaj nekog drugog polja u zaglavju okvira oštećen. Mi zato želimo da odaberemo šemu otkrivanja grešaka, tako daje verovatnoća ovakvih dogadaja mala. Uopšteno govoreći, složenije tehnike za otkrivanje i ispravljanje grešaka (odnosno one koje rade uz manju verovatnoću pojave neotkrivenih grešaka bitova) zahtevaju da se obavi više izračunavanja i prenese veći broj bitova u komunikaciji.

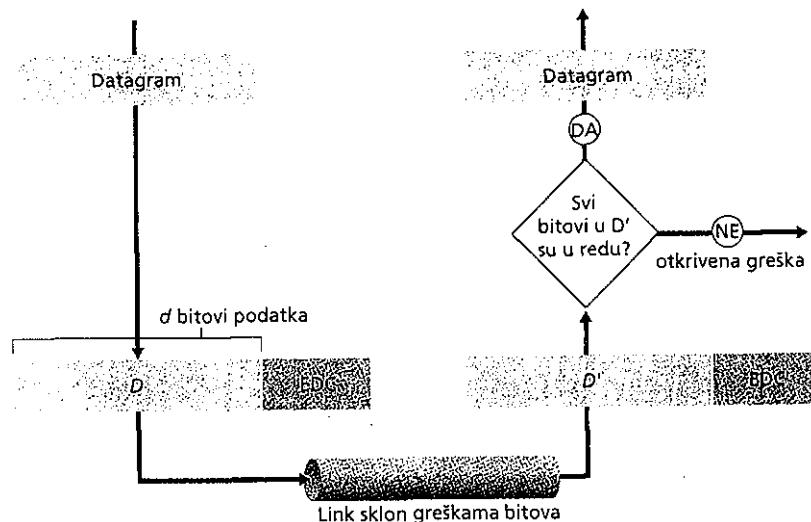
Hajde da sada ispitamo tri tehnike za otkrivanje grešaka u podacima - ispitivanja pamosti (da bismo ilustrovali osnovne ideje o otkrivanju i ispravljanju grešaka), metode kontrolnog zbirka (koje se češće primenjuju u transportnom sloju) i ciklične provere redundantnosti (za koje je uobičajeno da se primenjuju u sloju veze podataka u adapterima).

### 5.2.1 Provere parnosti

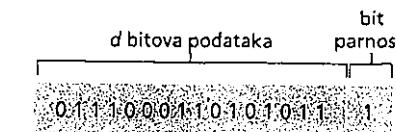
Upotreba jednog bita parnosti je možda najjednostavniji oblik otkrivanja greške. Pretpostavite da informacija koja treba da se pošalje,  $D$  na slici 5.4, ima  $d$  bitova. U parnoj šemi parnosti, pošiljalac jednostavno uključuje jedan dodatni bit i bira njegovu vrednost tako da ukupan broj "jedinica" u  $d+1$  bitu (originalna informacija plus bit parnosti) bude paran. Kod neparnih šema parnosti, vrednost bita parnosti bira se tako da bude neparan broj Jedinica". Na slici 5.5 ilustrovana je parna šema parnosti, sa jednim bitom pamosti smeštenim u posebnom polju.

Operacija prijemnika sa jednim bitom pamosti je takođe jednostavna. Prijemnik treba samo da prebroji koliko ima "jedinica" u primljenih  $d+1$  bitova. Ako se u parnoj šemi pamosti pronade neparan broj bitova koji imaju vrednost 1, prijemnik zna da se dogodila najmanje jedna greška na bitu. Tačnije, on zna da se dogodio neki neparan broj grešaka na bitovima.

Ali, šta se dešava ako se dogodio paran broj grešaka na bitovima? Rezultat bi bio neotkrivena greška. Ako je verovatnoća grešaka na bitovima mala fako se za greške može pretpostaviti da se dogadaju nezavisno od jednog bita do sledećeg, vero-



Slika 5.4 ♦ Scenario za otkrivanje i ispravljanje grešaka

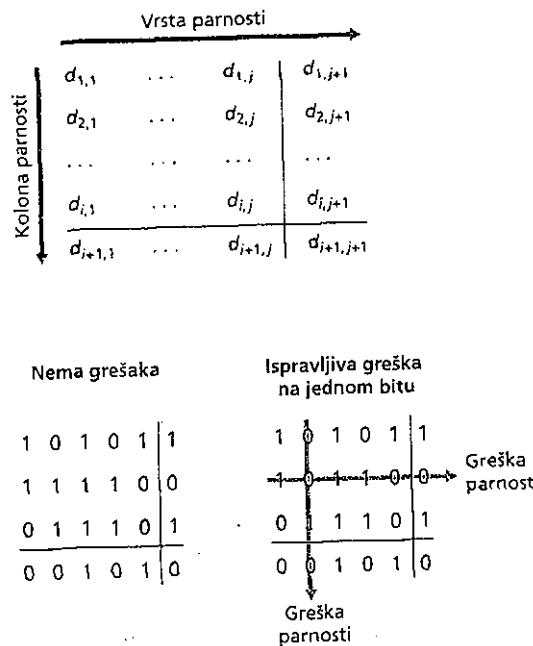


Slika 5.5 ♦ Jednobitna parna parnost

vatnoća višestrukih grešaka u paketu bila bi ekstremno mala. U tom slučaju, jedan bit pamosti bi zadovoljavao. Međutim, merenja su pokazala da se greške javljaju zajedno, u „grupama”, pre nego da se događaju nezavisno. U uslovima grupnih grešaka, verovatnoća neotkrivenih grešaka unutar okvira zaštićenog pomoću pamosti sa jednim bitom može da se približi vrednosti od 50 procenata [Spragins 1991]. Jasno je daje potrebna robustnija šema za otkrivanje grešaka (i ona se, na sreću, koristi u praksi). Ali, pre ispitivanja šema za otkrivanje grešaka koje se koriste u praksi, hajde da razmotrimo jednostavnu generalizaciju jednobitne pamosti koja će nam obezbediti uvid u tehnike ispravljanja grešaka.

Na slici 5.6 prikazana je dvodimenzionama generalizacija jednobitne šeme. par-nosti. U ovom slučaju,  $d$  bitova u  $D$  podeljeni su u / vrsta i  $j$  kolona. Vrednost par-nosti se računa za svaku vrstu i svaku kolonu. Rezultujući ('+/-| bitova pamosti obuhvataju bitove za otkrivanje greške unutar okvira veze podataka.

Sada pretpostavite da se u originalnih c/bitova informacije pojavila greška jednog bita. Kod ove **dvodimenzionalne šeme parnosti**, pamost i kolone i vrste koje sadrže pogrešan bit biće pogrešne. Prijemnik zato može ne samo da *otkrije* činjenicu da se



Slika 5.6 • Dvodimenzionalna parna parnost

dogodila jedna greška, nego i da upotrebi indekse vrste i kolone sa greškama pamosti da bi stvarno identifikovao oštećeni bit i *ispravio* tu grešku! Na slici 5.6 prikazan je primer u kome je bit sa vrednošću 1 na poziciji (2,2) oštećen i obrnut na 0 - stoje greška koju prijemnik može da otkrije i ispravi. Mada je naša rasprava usredsredena na originalnih  $d$  bitova informacije, jedna greška u samim bitovima pamosti takođe se može otkriti i ispraviti. Dvodimenzionalna pamost takođe može da otkrije (ali ne i da ispravi!) bilo kakvu kombinaciju dve greške u paketu. Ostale osobine dvodimenzionalne šeme ispitane su u problemima na kraju poglavlja.

Sposobnost prijemnika da otkrije i da ispravi greške poznata je kao ispravljanje grešaka unapred (*forward error correction*, FEC). Ove tehnike se **obično** koriste za skladištenje zvuka i reprodukciju sa uređaja **kao** Što su audio CD-ovi. U formiranju mreža, tehnike FEC mogu da se upotrebe same po sebi, ili zajedno sa tehnikama ARQ koje smo izložili u poglavlju 3. Vrednost tehnika FEC je što one mogu da smanje broj ponovnih prenosa od strane pošiljaoca. **Stoje** možda još važnije, one dozvoljavaju trenutno ispravljanje grešaka u prijemniku. Time se izbegava Čekanje u toku vremena povratnog puta, koje je prijemniku potrebno da bi primio NAK paket i da bi paket koji je ponovo poslat stigao do prijemnika - što može da bude značajna prednost za aplikacije mreža u realnom vremenu [Rubenstein 1998], Noviji radovi koji ispituju upotrebu tehnike FEC u protokolima sa kontrolom greške obuhvataju [Miersack 1992; Nonnenmacher 1998; Byers 1998; Shacham 1990].

## 5.2.2 Metode kontrolnog zbir-a

U tehnikama kontrolnog zbir-a, sa  $d$ bitova sa slike 5.4 postupa se **kao** sa sekvencom  $\wedge$ -bitnih celih brojeva. Jednostavna metoda kontrolnog zbir-a je da se prosto saberi ti /r-bitni celi brojevi i bitovi rezultata upotrebe za otkrivanje greške. Takozvani Internet kontrolni zbir zasniva se na ovom pristupu - sa bajtovima podataka postupa se **kao sa** 16-bitnim celim brojevima i oni se sabiraju. Jedinični komplement ovog zbir-a formira Internet kontrolni zbir koji se prenosi u zaglavlu segmenta. Kao Stope objašnjezno u odeljku 3.3, prijemnik proverava kontrolni zbir tako što uzima jedinični komplement zbir-a primljenih podataka (uključujući tu i sam kontrolni zbir) i ispituje da li su svi bitovi rezultata 1. Ako bilo koji bit ima vrednost 0, to ukazuje na grešku. RFC 1071 detaljno objašnjava Internet algoritam kontrolnog zbir-a i njegovu implementaciju. U protokolima TCP i UDP, Internet kontrolni zbir se proračunava preko svih polja (uključujući zaglavla i polja sa podacima). U drugim protokolima, na primer XTP [Strayer 1992], jedan kontrolni zbir se računa za zaglavje, a drugi **za** ceo paket.

Metode kontrolnog zbir-a zahtevaju relativno malo proširivanje paketa u vidu dodavanja novih bitova. Na primer, kontrolni zbirovi za TCP i UDP koriste samo 16 bitova. Međutim, oni obezbeđuju relativno slabu zaštitu od grešaka u poređenju sa cikličnim prverama redundantnosti (CRC), koje se objašnjavaju dalje u tekstu i koje se često koriste u sloju veze podataka. Ovde se prirodno postavlja pitanje: zašto se kontrolni zbir koristi u transportnom sloju, a CRC u sloju veze podataka? Setite

se da se transportni sloj obično implementira u softveru računara, kao deo njegovog operativnog sistema. S obzirom na to daje otkrivanje grešaka transportnog sloja implementirano u softveru, važno je da šema za otkrivanje grešaka bude jednostavna i brza, baš kao stoje to metoda kontrolnog zbirka. S druge strane, otkrivanje grešaka u sloju veze podataka implementirano je u hardveru adaptera, koji brzo može da izvrši i složenije CRC operacije.

Osnovni razlog zbog kojeg se kontrolni zbir koristi u transportnom sloju, a jače metode CRC u sloju linka je što se kontrolno sabiranje lako implementira u softveru.

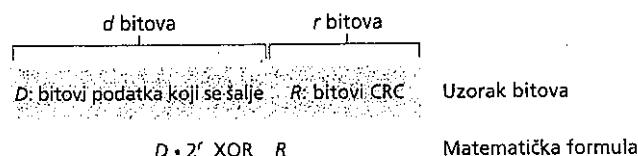
McAulev [McAulev 1994] opisuje poboljšane kodove težinskog kontrolnog zbirka koji su pogodni za softverske implementacije velike brzine, a Feldheimer [Feldheimer 1995] prikazuje tehnikе brzih softverskih implementacija, ne samo za kodove težinskog kontrolnog zbirka, nego i za CRC (pročitajte dalje u tekstu), kao i za druge kodove.

### 5.2.3 Ciklična provera redundantnosti (CRC)

Tehnika koja se danas široko koristi u računarskim mrežama zasnovana je na **kodovima sa cikličnom proverom redundantnosti (CRC)**. CRC kodovi su takođe poznati kao **polinomski kodovi**, zato što se na niz bitova koji treba da se pošalju može gledati kao na polinom čiji koeficijenti imaju vrednosti 0 i 1 u tom nizu, sa operacijama koje se interpretiraju kao polinomska aritmetika nad nizom bitova.

CRC kodovi rade na sledeći način. Zamislite  $d$ -bitni podatak,  $D$ , koji predstavlja čvor želi da pošalje prijemnom čvoru. Pošiljalac i primalac moraju prvo da se spojaju u razumeju o uzorku od  $r + 1$  bita, poznatom kao **generator**, koji će označiti sa  $G$ . Zahvaljujući da bit najveće težine (krajnji levi) u  $G$  bude 1. Ključna ideja na kojoj se zasniva CRC prikazana je na slici 5.7. Za dati podatak,  $D$ , pošiljalac će odabrat  $r$  dodatnih bitova,  $R$ , i priključiti ih na kraj niza  $D$  tako da rezultujući uzorak  $d + r$  (interpretiran kao binarni broj) bude tačno deljiv sa  $G$  koristeći aritmetiku modulo 2. Na ovaj način, proces proveravanja grešaka je jednostavan: primalac deli primljeni  $d + r$  sa  $G$ . Ako ostatak nije nula, primalac zna da se dogodila greška; u suprotnom, podatak se prihvata kao ispravan.

Sva CRC izračunavanja vrše se u aritmetici modulo 2, bez prenosa u sabiranju ili pozajmljivanju u oduzimanju. To znači da su sabiranje i oduzimanje identični i da su oba ekvivalentni bitskom operandu isključivom ILI (XOR). Na primer,



Slika 5.7 ♦ Kodovi CRC

$$1011 \text{ XOR } 0101 = 1110 \text{ 1001 XOR}$$

$$1101 = 0100 \text{ Takođe, slično tome}$$

$$1011 - 0101 = 1110 \text{ 1001} - 1101 =$$

$$0100$$

Množenje i deljenje su isti kao u aritmetici sa osnovom 2, izuzev što se sva zahtevana sabiranja ili oduzimanja vrše bez prenosa i pozajmljivanja. Kao u običnoj binarnoj aritmetici, množenje sa  $2^k$  pomera bitove u levo za  $k$  mesta. Dakle, uz date  $D$  i  $R$ , izraz  $D \bullet 2^r \text{ XOR } R$  daje bitove  $d + r$  prikazane na slici 5.7. U našoj daljoj diskusiji, koristićemo ovu algebarsku karakterizaciju bitova  $d + r$ .

Okrenimo se sada odlučujućem pitanju, kako pošiljalac proračunava  $R$ . Setite se da želimo da pronađemo  $R$ , tako da postoji takvo  $n$  da je

$$D - 2^r \text{ XOR } R = nG$$

To znači, želimo da odaberemo  $R$  koje  $G$  deli sa  $\bullet 2^r \text{ XOR } R$  bez ostatka. Ako izvršimo isključivo-ILI (odnosno, saberemo po modulu 2 bez prenosa)  $R$  sa obe strane gornje jednačine, dobijamo

$$D - 2^r = nG \text{ XOR } R$$

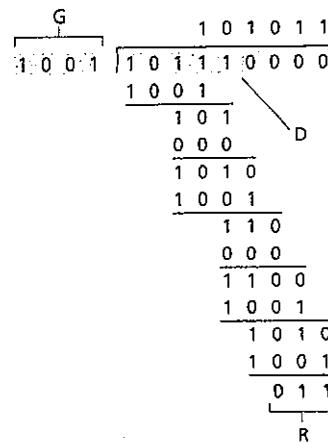
Ova jednačina nam govori da ako podelimo  $D \bullet 2^r$  sa  $G$ , vrednost ostatka je tačno  $R$ . Drugim rečima, možemo da izračunamo  $R$  na sledeći način

$$\underline{D - 2^r R} = \\ \text{ostatak od } Q$$

Na slici 5.8 prikazan je ovaj proračun za slučaj  $D = 101110$ ,  $d = 6$ ,  $G = 1001$  i  $r = 3$ . Devet bitova koji se prenose u ovom slučaju su 101110011. Možete dprove-rite ovaj proračun, a takođe da se uverite da je  $D \bullet 2^r = 101011 \bullet G \text{ XOR } R$ .

Definisani su međunarodni standardi za 8-, 12-, 16- i 32-bitne generatore,  $G$ . S-bitni CRC se koristi da zaštiti 5-bajtno zaglavje u ATM celijama. 32-bitni standard CRC-32, koji je usvojen u brojnim IEEE protokolima sloja linka, koristi generator

$${}^{\circ}\text{CRC-32} = 100000100110000010001110110110111$$



**Slika 5.8** ♦ Primer proračuna CRC

Svaki od CRC standarda može da otkrije grupu grešaka koja se javlja na manje od  $r + 1$  bitova. (To znači da sve uzastopne greške bitova na rastojanju  $/m$  ili manje bitova mogu da se otkriju.) Pored toga, pod odgovarajućim pretpostavkama, grupna greška veća od  $r + 1$  bitova može da se otkrije sa verovatnoćom od  $1 - 0,5^r$ . Takođe, svaki od CRC standarda može da otkrije svaki neparan broj grešaka bitova. U tekstu [Schvartz 1908], dat je odličan uvod u ovu temu.

### 5.3 Protokoli višestrukog pristupa

U uvodu ovog poglavlja, napisali smo da postoje dve vrste mrežnih linkova: linkovi od tačke do tačke i difuzni linkovi. Link od tačke do tačke sastoji se od jednog pošiljaoca najednom kraju linka, i jednog primaoca na drugom kraju linka. Mnogi protokoli sloja linka projektovani su kao linkoviod tačke do tačke; PPP (*point-to-point protocol*) i HDLC su dva takva protokola, koja ćemo objasniti kasnije u ovom poglavlju. Druga vrsta linka, difuzni link, može da ima više predajnih i više prijemnih čvorova koji svi dele jedan, isti difuzni kanal. Termin „difuzni“ ovde se koristi zato što, kada bilo koji čvor šalje okvir, kanal difuzno prenosi okvir i svaki čvor prima kopiju. Primeri za tehnologije difuznog sloja linka su Ethernet i bežični LAN-ovi. U ovom odeljku ćemo se vratiti korak unazad od specifičnih protokola sloja linka i prvo ispitati problem koji je najznačajniji za sloj veze podataka: kako da se uskladi pristup više predajnih i prijemnih čvorova deljenom difuznom kanalu

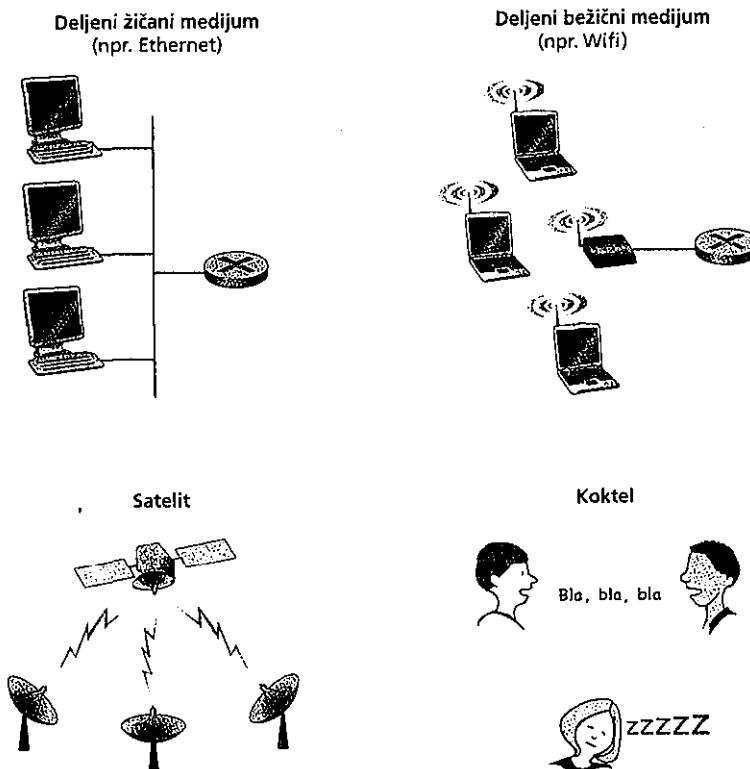
- takozvani problem višestrukog pristupa. Difuzni kanali se često koriste u lokalnim računarskim mrežama (LAN-ovima), mrežama koje su geografski koncen-trisane u jednoj zgradi (ili korporaciji ili univerzitetskom naselju). Zato ćemo na kraju ovog poglavlja takođe videti kako se kanali sa višestrukim pristupom koriste u LAN-ovima.

Nama je svima poznato značenje difuznog prenosa, zato što ga televizija koristi od kada je pronađena. Ali, tradicionalna televizija je jednosmerni difuzni prenos (odnosno, jedan fiksni čvor prenosi mnogim prijemnim čvorovima), dok čvorovi na difuznom kanalu računarske mreže mogu i da predaju i da primaju. Pogodnija analogija difuznog kanala je koktel prijem, gde se mnogo ljudi skuplja u velikoj prostoriji (vazduh je difuzni medijum), da bi pričali i slušali. Druga dobra analogija je nešto sa čime bi mnogi čitaoci trebalo da budu upoznati - učionica gde nastavnik (nastavnici) i student (studenti) takođe dele jedan difuzni medijum. Centralni problem u ova scenarija je da se odredi ko i kada priča (odnosno predaje na kanalu). Kao ljudi, mi smo razradili skup protokola za deljenje difuznog kanala:

„Dajte svakom priliku da govori.“ „Ne pričajte kada vam se obraćaju.“ „Ne monopolisite razgovor.“ „Podignite ruku kada imate pitanje.“ „Ne prekidajte nekoga kada govori.“ „Nemojte zaspati kada neko govori.“

Računarske mreže imaju slične protokole - to su takozvani protokoli višestrukog pristupa - pomoću kojih čvorovi regulišu svoj prenos na deljenom difuznom kanalu. Kao stoje prikazano na slici 5.9, protokoli sa višestrukim pristupom su potrebni u širokom skupu postavki mreža, uključujući tu ozičene i bežične lokalne mreže računara, kao i satelitske mreže. Mada, tehnički posmatrano, svaki čvor pristupa difuznom kanalu preko svog adaptera, u ovom odeljku mi ćemo se pozivati na čvor kao na predajni i prijemni uređaj. U praksi, stotine ili čak hiljade čvorova mogu direktno da komuniciraju preko difuznog kanala.

S obzirom na to da su svi čvorovi u stanju da šalju okvire, to više od dva Čvora mogu da rade istovremeno. Kada se to dogodi, svi čvorovi primaju više okvira u isto vreme; znači da dolazi do kolizije okvira u prenosu, na svim primaocima. Obično se dešava da, kada postoji kolizija, nijedan od prijemnih čvorova ne može da uradi sa poslatim okvirima ništa što bi imalo smisla; signali okvira u koliziji postaju međusobno nerazumljivo zamršeni. Zato su svi okviri u koliziji izgubljeni, a difuzni kanal neupotrebljiv u vremenskom intervalu kolizija. Jasno je da, ako mnogo Čvorova želi često da predaje okvire, mnogi prenosi će za rezultat imati kolizije i veliki deo propusnog opsega difuznog kanala će biti neiskorišćen.



**Slika 5.9** ♦ Različiti kanali sa višestrukim pristupom

Da bi se obezbedilo da difuzni kanal korisno radi kada je aktivno više Čvorova, potrebno je da se nekako usklade prenosi aktivnih čvorova. To je posao protokola sa višestrukim pristupom. U toku 30 godina, hiljade članaka i stotine doktorskih disertacija napisano je o protokolima sa višestrukim pristupom; iscrpan pregled ovog rada dat je u [Rom 1990]. Štaviše, aktivno istraživanje u oblasti protokola sa višestrukim pristupom se nastavlja, zahvaljujući stalnom pojavljivanju novih vrsta linkova, posebno novih bežičnih linkova.

Tokom godina, implemenirano je više desetina protokola sa višestrukim pristupom u različitim tehnologijama sloja linka. Međutim, skoro svaki protokol sa višestrukim pristupom možemo klasifikovati da pripada jednoj od sledeće tri kategorije: **protokoli sa deljenjem kanala**, **protokoli sa slučajnim pristupom** i **protokoli sa pristupom „na koga je red“**. U sledeća tri pododeljka objasnićemo ove kategorije protokola sa višestrukim pristupom.

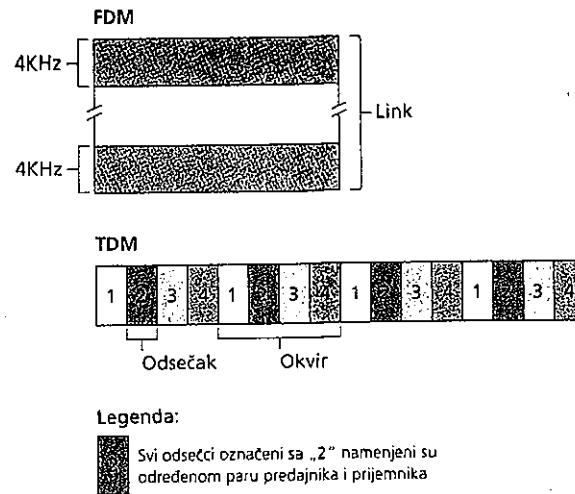
Zaključićemo ovaj pregled time da bi, u idealnom slučaju, protokol sa višestrukim pristupom za difuzni kanal brzine  $R$  bitova u sekundi, trebalo da ima sledeće karakteristike:

1. Kada samo jedan čvor ima podatke za slanje, on ima propusnu moć od  $R$  bitova u sekundi.
2. Kada  $M$  čvorova imaju podatke za slanje, svaki od njih ima propusnu moć od  $R/M$  bitova u sekundi. To ne znači da svaki od  $M$  čvorova obavezno uvek ima brzinu prenosa od  $R/M$ , nego da bi svaki od  $M$  čvorova trebalo da ima brzinu prenosa od  $R/M$  u nekom pogodno defmisanom vremenskom intervalu.
3. Protokol je decentralizovan; to znači, da nema centralnih čvorova koji mogu da otkazuju sistem.
4. Protokol je jednostavan, tako da nije skup za implementaciju.

### 5.3.1 Protokoli sa deljenjem kanala

Setite se ranijih objašnjenja u odeljku 1.3 da su vremensko (*time-division multiple-xing*, TDM) i frekventno multipeksiranje (*frequency-division mtddiplexing*, FDM) dve tehnike koje mogu da se koriste za deljenje propusnog opsega difuznog kanala između svih čvorova koji dele taj kanal. Kao primer, prepostavite da kanal podržava  $N$  čvorova i da je brzina prenosa kanala  $R$  bitova u sekundi. TDM deli vreme na **vremenske okvire** (*time frames*), a dalje deli svaki vremenski okvir na **N vremenskih odsečaka** (*Ume slots*). (Vremenski prozor TDM ne bi trebalo da se pomeša sa PDU-om sloja linka, koji se takode naziva okvir. Da bi se smanjila mogućnost zabune, u ovom pododeljku na PDU sloja linka ćemo se pozivati kao na paket.) Svaki vremenski odsečak se onda dodeljuje jednom od  $N$  čvorova. Kad god čvor ima paket koji treba da pošalje, on vrši predaju bitova paketa za vreme svog dodeljenog vremenskog odsečka koji se aktivira tokom rotirajućeg TDM prozora. Uobičajeno je da se veličine vremenskih odsečaka biraju tako da jedan paket može da se prenese u okviru jednog vremenskog odsečka. Na slici 5.10 prikazan je jednostavan primer TDM-a od Četiri čvora. Ako se vratimo na našu analogiju, koktel prijem regulisan pomoću TDM-a dozvolio bi jednom učesniku da govori u fiksnom vremenskom periodu, a zatim drugom učesniku da govori jednako dugo i tako dalje. Kada su svi imali svoju priliku da govore, postupak se ponavlja.

Tehnika TDM je privlačna, zato što eliminiše kolizije i savršeno je pravedna: svaki čvor ima brzinu  $R$  tokom korišćenja svog okvira, a brzinu 0 izvan toga, pa je srednja brzina prenosa po čvoru  $R/N$  bitova u sekundi za vreme svakog vremenskog okvira. Međutim, ona ima dva velika nedostatka. Prvo, čvor je ograničen na prosečnu brzinu od  $R/N$  bitova u sekundi čak i kada je jedini čvor sa paketima za slanje. Drugi nedostatak je to što čvor uvek mora da čeka na svoj red u sekvenci prenosa - opet, čak i kada je jedini čvor sa paketom za slanje. Zamislite učesnika na prijemu koji jedini ima nešto da kaže (i zamislite daje to čak i reda okolnost u kojoj svako na prijemu želi da čuje šta ta jedna osoba ima da kaže). Jasno, tehnika TDM bi bila loš izbor za protokol sa više pristupa na tom određenom prijemu.



**Slika 5.10** ♦ Primeri TDM i FDM sa četiri čvora

Dok TDM deli difuzni kanal u vremenu, FDM deli kanal brzine  $R$  bitova u sekundi na različite frekvencije (od kojih svaka ima propusni opseg od  $R/N$ ) i dodeljuje svaku frekvenciju jednom od  $V$  čvorova. FDM tako stvara  $N$  manjih kanala od  $R/N$  bitova u sekundi od jednog, većeg kanala brzine od  $R$  bitova u sekundi. FDM deli i prednosti i nedostatke TDM-a. FDM takođe deli glavni nedostatak sa TDM-om - čvor je ograničen na propusni opseg od  $R/N$ , čak i kada je jedini čvor sa paketom za slanje.

Treći protokol za deljenje kanala je višestruki pristup sa deljenjem koda (*Code Division Multiple Access*, CDMA). Dok TDM i FDM dodeljuju čvorovima vremenske odsečke i frekvencije, respektivno, CDMA svakom čvoru dodeljuje različit kod. Svaki čvor onda koristi jedinstveni kod da bi kodovao bitove podataka koje šalje. Ako se kodovi pažljivo izaberu, mreže CDMA imaju izvanrednu osobinu da različiti čvorovi mogu da prenose *istovremeno*, a da njihovi odgovarajući primaoci ipak tačno prime bitove kodovanih podataka (prepostavljajući da primalac zna kod pošiljaoca), uprkos uticaju prenosa drugih čvorova. Izvesno vreme CDMA se koristio u vojnim sistemima (zbog svoje otpornosti na ometanje), a sada počinje da se široko nalazi i u civilnoj upotrebi, posebno u bežičnim kanalima sa višestrukim pristupom. S obzirom daje upotreba CDMA tako tesno povezana sa bežičnim kanalima, uzdržaćemo se od diskusije o tehničkim detaljima CDMA do poglavlja 6. Za sada, biće dovoljno da znamo da CDMA kodovi, kao vremenski odsečci u TDM-u i frekvencije u FDM-u, mogu da se dodeljuju korisnicima kanala sa višestrukim pristupom..

### 5.3.2 Protokoli sa slučajnim pristupom

Druga Široka klasa protokola sa višestrukim pristupom su takozvani protokoli sa slučajnim pristupom. U protokolu sa slučajnim pristupom, predajni čvor uvek predaje punom brzinom kanala, odnosno  $R$  bitova u sekundi. Kada postoji kolizija, svaki čvor koji je njime obuhvaćen ponovo šalje svoj okvir (odnosno paket), sve dok okvir ne prode bez kolizija. Ali, kada čvor doživi koliziju, on ne šalje ponovo okvir istog trenutka. Umesto toga, on čeka da prode vreme slučajnog trajanja dok ponovo ne pošalje okvir. Svaki čvor obuhvaćen kolizijom bira nezavisno slučajno kašnjenje. S obzirom na to da se slučajna kašnjenja biraju nezavisno, moguće je da će jedan od čvorova odabrat kašnjenje koje je dovoljno manje od kašnjenja ostalih čvorova u koliziji i da će zato moći da provuče svoj okvir kroz kanal bez kolizije.

Postoje desetine, ako ne i stotine protokola sa slučajnim pristupom koji su opisani u literaturi [Rom 1990; Bertsekas 1991]. U ovom odeljku, opisacemo nekoliko protokola sa slučajnim pristupom koji se najviše koriste - protokole ALOHA [Abramson 1970; Abramson 1985] i protokole sa prepoznavanjem nosioca sa slučajnim pristupom (*carrier sense multiple access*, CSMA) [Kleinrock 1975b]. Kasnije, u odeljku 5.5., objasnićemo detalje Ethemeta [Metcalfe 1976], popularnog i široko primenjivanog CSMA protokola.

#### ALOHA sa odsečcima

Hajde da započnemo našu studiju protokola sa slučajnim pristupom sa jednim od najjednostavnijih, takozvanim protokolom ALOHA sa odsečima. U našem opisu ALOHE sa odsečcima, pretpostavićemo sledeće:

- ♦ Svi okviri se sastoje od tačno  $L$  bitova.
- ♦ Vreme se deli na odsečke veličine  $L/R$  sekundi (odnosno, odsečak je jednak vremenu prenosa jednog okvira).
- ♦ Čvorovi počinju da predaju okvire samo na počecima odsečaka.
- ♦ Čvorovi su sinhronizovani, tako da svaki od njih zna kada odsečak počinje.
- ♦ Ako dode do kolizije dva ili više okvira u odsečku, svi čvorovi detektuju koliziju pre nego što se odsečak završi.

Neka je  $p$  verovatnoća, odnosno broj između 0 i 1. Operacija ALOHE sa odsečcima u svakom čvoru je jednostavna:

- ♦ Kada čvor treba da pošalje novi okvir, on čeka do početka sledećeg odsečka i predaje celokupni okvir tom odsečku.
- ♦ Ako nema kolizije, čvor je uspešno predao svoj okvir i zato ne treba da razmatra njegovo ponovno prenošenje. (Čvor može da pripremi novi okvir za prenos u sledećem okviru, ako ga ima.)

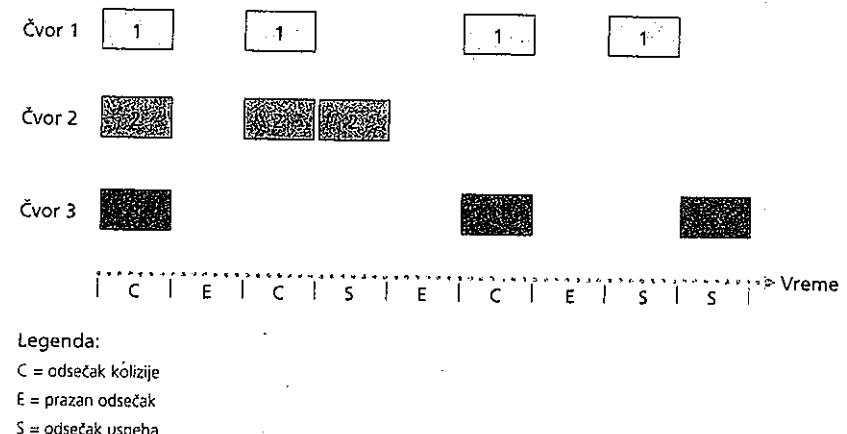
- ♦ Ako postoji kolizija, čvor je detektuje pre kraja odsečka. Čvor ponovo predaje svoj okvir u svakom sledećem odsečku sa verovatnoćom  $p$  dok se okvir ne prenese bez kolizije.

Pod ponovnim prenošenjem sa verovatnoćom  $p$ , podrazumevamo da čvor u stvari baca novčić sa sledećim ishodima: događaj „glava“ odgovara ponovnom pre-nosu i ima verovatnoću  $p$ . Događaj „pismo“ znači „preskoči odsečak i baci novčić ponovo u sledećem odsečku“; to se dešava sa verovatnoćom  $(1 - p)$ . Svi čvorovi obuhvaćeni kolizijom bacaju svoje novčiće nezavisno.

Izgleda da ALOHA sa odsečcima ima mnogo prednosti. Za razliku od deljenja kanala, ALOHA sa odsečcima dozvoljava čvoru da stalno predaje punom brzinom,  $R$ , kadaje taj Čvor jedini aktivan. (Za čvor kažemo daje aktivan kada ima okvira za slanje.) Protokol ALOHA sa odsečcima je i veoma decentralizovan, zato što svaki čvor detektuje kolizije i nezavisno odlučuje kada da ponovi predaju. (Međutim, ALOHA sa odsečcima zahteva da odsečci budu sinhronizovani u čvorovima; uskoro ćemo objasniti verziju protokola ALOHA bez odsečaka, kao i protokole CSMA, od kojih nijedan ne zahteva takvu sinhronizaciju i zato su potpuno decentralizovani.) ALOHA sa odsečcima je takođe izuzetno jednostavan protokol.

Protokol ALOHA sa odsečcima dobro radi kada postoji samo jedan aktivan čvor, ali koliko je on jednostavan kada ima više aktivnih čvorova? U vezi sa efikasnošću može se razmišljati o dve stvari. Prvo, kao stoje prikazano na slici 5.11, kada ima više aktivnih čvorova, izvestan deo odsečaka će biti u *koliziji* i zato će biti „odbačen“. Drugi problem je što će drugi deo odsečaka biti *prazan*, zato što se svi aktivni čvorovi uzdržavaju od prenosa zbog probabilističke politike prenosa. Jedini „iskorišćeni“ odsečci će biti oni u kojima tačno jedan Čvor predaje. Odsečak u kome samo jedan čvor predaje zove se odsečak uspeha. Efikasnost protokola sa višestrukim pristupom i odsečcima definije se kao deo odsečaka uspeha tokom dugog perioda, u slučaju kada postoji veliki broj aktivnih čvorova, od kojih svaki ima veliki broj okvira za slanje. Zapazite da bi, ako nije upotrebljen nikakav oblik kontrole pristupa i ako bi svaki čvor odmah ponovo prenosio posle svake kolizije, efikasnost bila ravna nuli. Jasno je da ALOHA sa odsečcima povećava efikasnost iznad nule, ali koliko?

Sada ćemo nastaviti sa izvođenjem maksimalne efikasnosti protokola ALOHA sa odsečcima. Da bi ovo izvođenje bilo jednostavno, hajde da malo promenimo protokol i pretpostavimo da svaki Čvor uvek ima okvir za slanje i da Čvor predaje novi okvir sa verovatnoćom  $p$ . (To znači da pretpostavljamo da svaki Čvor uvek ima okvir za slanje i da čvor šalje novi okvir sa verovatnoćom  $p$ , kao i okvir koji je već u koliziji.) Najpre pretpostavite da postoji  $N$  Čvorova. Tada je verovatnoća da dati odsečak predstavlja odsečak uspeha jednaka verovatnoći da jedan od čvorova predaje, a da ostalih  $N - 1$  čvorova ne predaju. Verovatnoća da jedan Čvor predaje je  $p$ ; verovatnoća da ostali čvorovi ne predaju je  $(1 - p)^{N-1}$ . Zato je verovatnoća da je odsečak uspešan jednaka  $p(1 - p)^{N-1}$ . S obzirom na to da ima  $N$  čvorova, verovatnoća da neki proizvoljan čvor obavi uspešan prenos iznosi  $Np(1 - p)^{N-1} \sim$



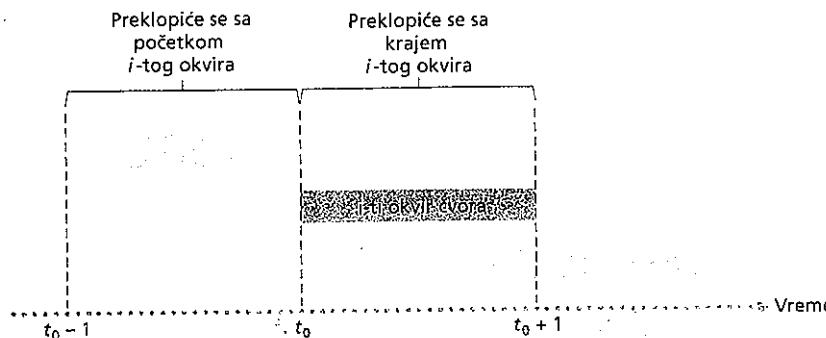
**Slika 5.11** ♦ Čvorovi 1, 2 i 3 se sudaraju u prvom odsečku. Čvor 2 konačno uspeva u četvrtom odsečku, čvor 1 u osmom odsečku, a čvor 3 u devetom odsečku.

Dakle, kada postoji  $N$  aktivnih čvorova, efikasnost protokola ALOHA sa odsečcima jednaka je  $A^N(1 - p)^{N-1}$ . Da bi se dobila maksimalna efikasnost za  $N$  aktivnih čvorova, treba da pronademo koje maksimizuje taj izraz. (Pogledajte probleme za domaće zadatke u kojima je opisan opšti postupak tog izvođenja.) A da bismo dobili maksimalnu efikasnost za veliki broj aktivnih čvorova, uzećemo granicu  $N p(1 - p)^{N-1}$  kada se  $N$  približava beskonačnosti. (I to pogledajte u problemima za domaće zadatke.) Kada izvršimo te proračune, otkrićemo da je maksimalna efikasnost protokola data izrazom  $1/e = 0,37$ . To znači da, kada veliki broj čvorova ima mnogo okvira za predaju, onda (u najboljem slučaju) samo 37% čvorova radi nešto korisno. Dakle, efektivna brzina prenosa kanala nije  $R$ , nego samo  $0,37 R$  bitova u sekundi! Slična analiza takođe pokazuje da je 37 procenata odsečaka prazno i da se u 26 procenata odsečaka javljaju kolizije. Zamislite sirotog administratora mreže koji je kupio sistem ALOHA sa odsečcima, od 100 Mb/s, očekujući da će moći da koristi mrežu za prenošenje podataka između velikog broja korisnika zajedničkom brzinom od, recimo, 80 Mb/s! Iako kanal može da prenosi dati okvir punom brzinom od 100 Mb/s, u dužem vremenu realna propusna moć tog kanala će biti manja od 37 Mb/s u sekundi.

## ALOHA

Protokol ALOHA sa odsećima zahteva da svi čvorovi sinhronizuju svoje prenose tako da započnu na početku odsečka. Prvi protokol ALOHA [Abramson 1970] bio je stvarno bez odsečaka, potpuno decentralizovan protokol. U takozvanoj čistoj ALOHI, kada okvir stigne prvi put (odnosno, mrežni sloj prosledi datagram naniže iz mrežnog sloja u čvor pošiljaoca), čvor odmah predaje ceo okvir na difuzni kanal. Ako predati okvir dode u koliziju sa jednim ili više prenosa, Čvor će odmah (pošto je predao ceo svoj okvir u koliziji) ponovo predati okvir sa verovatnoćom  $p$ . U suprotnom, čvor čeka u intervalu koji je jednak vremenu potrebnom za prenos jednog okvira. Posle tog čekanja, on predaje okvir sa verovatnoćom  $p$ , ili čeka (ostajući besposlen) na drugi okvir sa verovatnoćom  $(1-p)$ .

Da bismo odredili efikasnost čiste ALOHE, usredsredićemo se na jedan individualni Čvor. Napravićemo iste pretpostavke kao i u našoj analizi ALOHE sa odsećima i uzećemo da vreme prenosa okvira bude jedinica vremena. U bilo kom datom trenutku, verovatnoća da čvor prenosi okvir je  $p$ . Pretpostavimo da prenos tog okvira počinje u trenutku  $t_0$ . Kao Stope prikazano na slici 5.12, da bi prenos tog okvira bio uspešan, nijedan drugi okvir ne može da započne svoj prenos u intervalu vremena  $[t_0 - 1, t_0]$ . Takav prenos bi se preklopio sa početkom prenosa okvira  $/$ -tog čvora. Verovatnoća da nijedan drugi ne počinje prenos u tom intervalu je  $(1 - p)^N$ . Slično tome, nijedan drugi čvor ne može da počne prenos dok čvor  $i$  prenosi, jer bi se takav prenos preklopio sa krajem prenosa  $/$ -tog čvora. Verovatnoća da nijedan od svih drugih čvorova ne počinje prenos u tom intervalu je takođe  $(1 - p)^N$ . Prema tome, verovatnoća da dati čvor izvrši uspešan prenos je  $?(1 - p)^N$ . Izračunavanjem granice kao u slučaju ALOHE sa odsećima, dolazimo do toga daje efikasnost čiste ALOHE samo  $1/(2e)$  - tačno polovina one koju ima ALOHA sa odsećima. To je cena koja mora da se plati za potpuno decentralizovani protokol ALOHA.



Slika 5.12 ♦ Ometajući prenosi u čistoj ALOHI



## KRATAK OSVRT

## NORM ABRAMSON I ALOHANET

Dr Norm Abramson, inženjer, strasno je voleo jahanje na talasima i interesovala ga je komutacija paketa. Ta kombinacija interesovanja dovela ga je na Univerzitet na Havajima 1969. godine. Havaji se sastoje od mnogo planinskih ostrva, što otežava postavljanje i rad zemaljskih mreža. Kada ne bi jahao na talasima, Abramson je razmišljao kako da projekfuje mrežu koja bi vršila komutaciju paketa preko radija. Mreža koju je projekfovala imala je jedan centralni računar i više sekundarnih čvorova, razbacanih po Havajskim ostrvima. Mreža je imala dva kanala, od kojih je svaki koristio različit frekventni opseg. Kanal za predaju naniže je difuzno predavao pakete iz centralnog računara do sekundarnih čvorova; kanal za predaju naviše je slao pakete iz sekundarnih računara u centralni računar. Pored predaje informacionih paketa, centralni računar je kanalom za predaju naniže lakoće slao i potvrdju za svaki paket koji je uspešno primio tz sekundarnih računara.

S obzirom na to da su sekundarni računari slali pakete na decentralizovan način, neizbežno je dolazio do kolizija na kanalu za predaju naviše. To posmatranje je dovelo Abramsona do izuma čistog protokola ALOHA, kao što je opisano u ovom poglavljiju. Godine 1970. uz trajnu podršku ARPA-e, Abramson je povezao svoju mrežu ALOHAner so mrežom ARPAnet. Abramsonov rad značajan je ne samo zato što je to bio prvi primer radiomreže sa komutacijom paketa, nego i zato što je nadahnuo Boba Metkalfa. Nekoliko godina pošto ga je Abramson izmislio, Metkalf je modifikovao protokol ALOHA, i tokom su nastali protokol CSMA/CD i LAN Ethernet.

## CSMA - slučajan pristup sa prepoznavanjem nosioca

U oba protokola ALOHA, u onom sa odsećima kao i u onom bez njih, odluka čvora da predaje donosi se nezavisno od aktivnosti drugih čvorova priključenih na difuzni kanal. Posebno, čvor nikada ne obraća pažnju na to da li neki drugi čvor predaje kad on počinje sa predajom, niti zaustavlja predaju ako drugi čvor počinje da ga ometa svojom predajom. U našoj analogiji sa koktel prijemom, protokoli ALOHA su kao dosadan učešnik na prijemu koji nastavlja da brblja bez obzira na to da li drugi ljudi nešto govore. Kao ljudi, mi imamo ljudske protokole koji nam omogućavaju ne samo da se ponašamo civilizovanje, nego i da smanjimo vreme koje provodimo „u kolizijama“ međusobno u svojim razgovorima i, shodno tome, da povećamo vreme u kome stvarno razgovaramo. Posebno, postoje dva važna pravila za vođenje uljudnog razgovora:

- ♦ *Slušaj pre nego što progovoriš.* Ako neki drugi govore, sačekaj dok oni završe. U mrežnom svetu, to se zove prepoznavanje nosioca (*carrier sensing*) - čvor sluša kanal pre nego što predaje. Ako se okvir iz drugog čvora trenutno prenosi

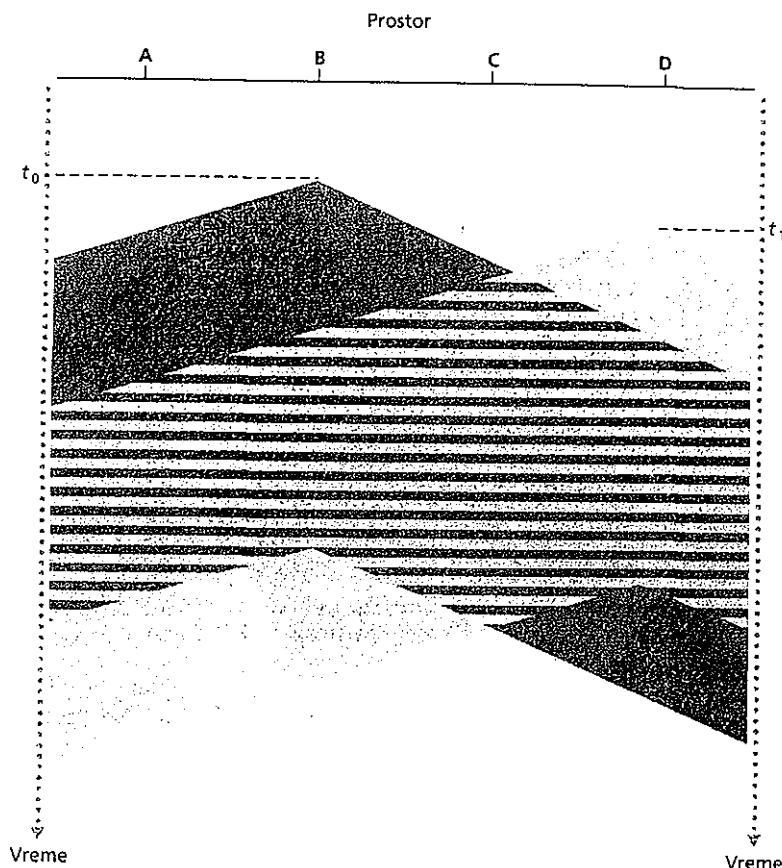
po kanalu, Čvor onda Čeka („odstupa“) slučajno vreme, a zatim opet sluša kanal. Ako oseti da je kanal slobodan, čvor počinje sa predajom okvira. U suprotnom, Čvor čeka drugo slučajno vreme i zatim ponavlja ovaj proces. ♦ *Ako neko počne da govori u isto vreme, prekini da govorиш.* U mrežnom svetu, to se zove otkrivanje kolizija (*collision detection*) - Čvor koji predaje istovremeno sluša kanal. Ako otkrije da drugi čvor predaje okvir koji ga ometa, on zaustavlja predaju i koristi neki protokol da odredi kada bi sledeći put trebalo da pokuša da predaje.

Ova dva pravila ugrađena su u familiju protokola CSMA (*carrier sense multiple access*, otkrivanje nosioca sa slučajnim pristupom) i CSMA/CD (CSMA sa otkrivanjem kolizija) [Kleinrock 1975b; Metcalfe 1976; Lam 1980; Rom 1990]. Predložene su mnoge varijacije CSMA i CSMA/CD. Čitalac može da pročita tri reference radi detaljnijih informacija o ovim protokolima. Mi ćemo u odeljku 5.5 detaljno proučiti šemu CSMA/CD koja se koristi u Ethernetu. Ovde ćemo razmotriti najvažnije i osnovne karakteristike protokola CSMA i CSMA/CD.

Prvo pitanje koje bi se moglo postaviti o protokolu CSMA glasi: Ako svi čvorovi izvršavaju prepoznavanje nosioca, zašto uopšte dolazi do kolizija? Na kraju krajeva, čvor će odustati od predaje kad god oseti da neki drugi čvor predaje. Odgovor na to pitanje može najbolje da se ilustruje upotrebom prostorno-vremenskih dijagrama [Molle 1987]. Na slici 5.13 prikazanje prostorno-vremenski dijagram četiri Čvora (A, B, C, D) koji su priključeni na linearu difuznu magistralu. Horizontalna osa prikazuje položaj svakog čvora u prostoru. Vertikalna osa prikazuje vreme.

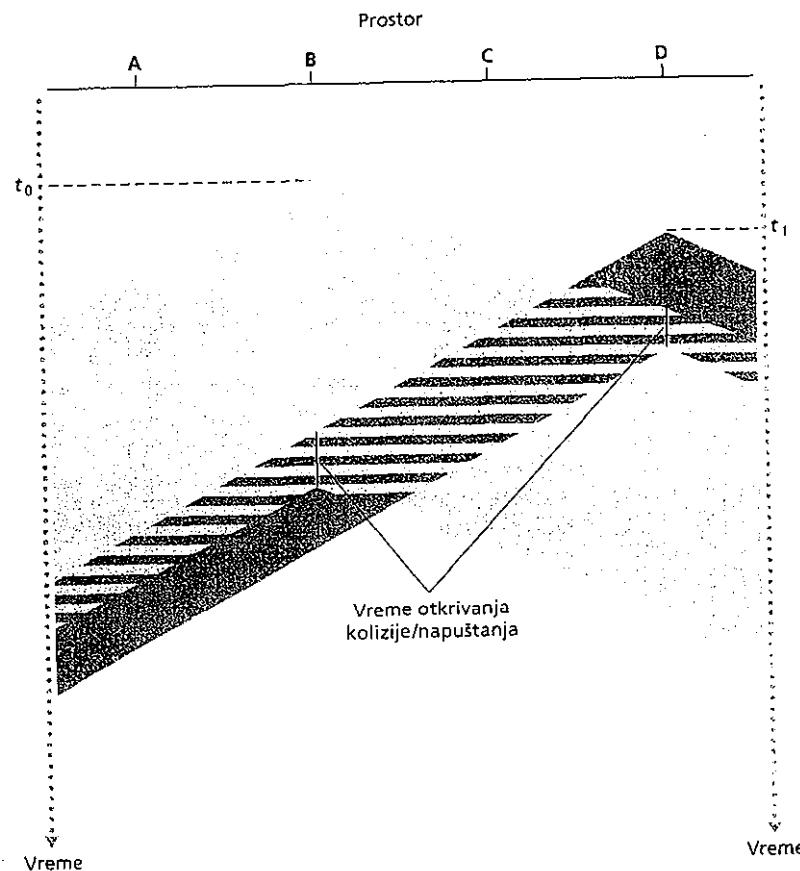
U trenutku  $t_0$ , Čvor B oseća da je kanal slobodan, jer nijedan drugi čvor trenutno ne predaje. Zato Čvor B počinje sa predajom, a njegovi bitovi se propagiraju u oba pravca po difuznom medijumu. Propagacija naniže bitova sa čvora B na slici 5.13, sa porastom vremena pokazuje daje potrebno vreme veće od nule da bi zaista bilo propagacije bitova B (iako skoro brzinom svjetlosti) duž difuznog medijuma. U trenutku  $t_1$ , ( $t_1 > t_0$ ), čvor D ima okvir za slanje. Mada čvor B u trenutku  $t_1$  predaje, bitovi koje predaje B još nisu stigli do D, pa zato D oseća kanal kao slobodan. U skladu sa protokolom CSMA, D zato počinje da predaje svoj okvir. Kratko vreme posle toga, predaja Čvora B počinje da ometa predaju D na položaju D. Na slici 5.15 jasno se vidi da će kašnjenje usled propagacije po kanalu od kraja do kraja difuznog kanala - vreme koje je potrebno da se signal prostire od jednog do drugog čvora - imati odlučujuću ulogu u određivanju njegove performanse. Što je kašnjenje propagacije veće, to je veća verovatnoća da čvor koji otkriva nosioca neće moći da oseti predaju koja je već počela u drugom čvoru mreže.

Na slici 5.13, čvorovi ne otkrivaju kolizije; i B i D nastavljaju da predaju svoje okvire u celini, čak iako se dogodila kolizija. Kada čvor otkriva koliziju, on prekida prenos čim otkrije koliziju. Na slici 5.14 prikazanje isti scenario kao na slici 5.13, izuzev što oba čvora obustavljaju svoje prenose odmah posle otkrivanja kolizija.



**Slika 5.13** ♦ Prostorno-vremenski dijagram dva CSMA čvora sa prenosima u koliziji

Jasno je da će dodavanje otkrivanja kolizija protokolu sa višestrukim pristupom poboljšati performansu protokola time što se neće prenositi beskoristan (ometanjem okvirom drugog čvora) oštećen okvir u celini. Protokol Ethernet koji ćemo proučiti u odeljku 5.5 je CSMA protokol koji koristi otkrivanje kolizija.



**Slika 5.14** ♦ CSMA sa otkrivanjem sudara

### 5.3.3 Protokoli tipa „na koga je red“

Setite se da su dve poželjne osobine protokola sa višestrukim pristupom da (1) kada je samo jedan čvor aktivan, on ima propusnu moć od  $R$  bitova u sekundi i (2) kada je  $M$  čvorova aktivno, onda svaki čvor ima propusnu moć od skoro  $R/M$  bitova u sekundi. Protokoli ALOHA i CSMA imaju prvu osobinu, ali ne i drugu. To je moti-visalo istraživače da stvore drugu klasu protokola - **protokole tipa „na koga je red“** (*laking-tumsprotocol.i*). Kao kod protokola sa slučajnim pristupom, postoje na desetine protokola sa ovakvim pristupom i svaki od njih ima mnogo varijacija. Ovde ćemo objasniti dva od najvažnijih protokola. Prvi je **protokol sa prozivanjem**.

Pro-

tokol sa prozivanjem zahteva da jedan od Čvorova bude označen kao glavni. Glavni čvor **proziva** svaki od čvorova na kružni način. Preciznije, glavni **čvor** prvo šalje poruku čvoru 1, kojom mu kaže da može da predaje do nekog maksimalnog broja okvira. Kada čvor 1 prenese nekoliko okvira, glavni Čvor saopštava čvoru 2 da sada on može da prenese do nekog maksimalnog broja okvira. (Glavni čvor može da odredi kada je čvor završio sa slanjem svojih okvira posmatrajući nedostatak signala na kanalu.) Postupak se nastavlja, tako što glavni čvor proziva svaki od Čvorova na cikličan način.

Protokol sa prozivanjem odstranjuje kolizije i prazne odsečke od kojih pate protokoli sa slučajnim pristupom. To omogućava mnogo veću efikasnost. Ali, on takođe ima i nekoliko nedostataka. Prvi nedostatak je to što protokol uvodi kašnjenje prozivanja - vreme potrebno da se obavesti svaki Čvor da on može da predaje. Ako je, na primer, samo jedan čvor aktivan, onda će on prenosi brzinom manjom od  $R$  bitova u sekundi, zato što glavni čvor mora da prozove svaki od neaktivnih čvorova redom, svaki put kada je aktivni čvor poslao svoj maksimalni broj okvira. Drugi nedostatak, koji je možda ozbiljniji, je to što ako glavni čvor otkaže, ceo kanal postaje neoperativan.

Drugi protokol sa pristupom „na koga je red“ je **protokol sa prosleđivanjem žetona** (*token passing protocol*). U tom protokolu ne postoji glavni čvor. Mali okvir posebne namene koji se zove **žeton** (*token*), razmenjuje se između čvorova po nekom fiksnom redu. Na primer, čvor 1 bi mogao uvek da šalje žeton Čvoru 2, čvor 2 bi mogao uvek da šalje žeton čvoru 3, čvor  $N$  bi mogao uvek da šalje žeton čvoru 1. Kada čvor primi žeton, on se na njemu zadržava samo ako ima okvira za slanje; u suprotnom, on odmah prosleđuje žeton sledećem čvoru. Prosleđivanje žetona je decentralizovano i ima veliku efikasnost. Ali ni ono nije bez problema. Na primer, otkaz jednog čvora može da uništi ceo kanal. Ili, ako čvor slučajno propusti da osloboди žeton, tada moraju da se pozovu izvesni postupci obnavljanja, kako bi se žeton vratio u cirkulaciju. Tokom godina, razvijeni su mnogi proizvodi za prosleđivanje žetona i svaki od njih je pokušavao da reši ovo i druga nezgodna pitanja; dva od tih protokola, FDDI i IEEE 802.5, spomenućemo u sledećem odeljku.

### 5.3.4 Lokalne računarske mreže (LAN-ovi)

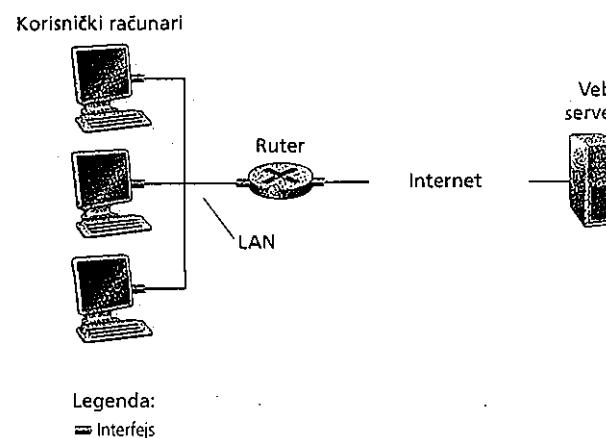
Protokoli sa višestrukim pristupom upotrebljavaju se zajedno sa mnogo različitih vrsta difuznih kanala. Oni su se koristili za satelitske i bežične kanale, čiji čvorovi su prenosili preko zajedničkog spektra frekvencija. Oni se danas koriste u uzvodnom kanalu za kablovski pristup Internetu (pročitajte odeljak 1.5), a široko se upotrebljavaju u lokalnim računarskim mrežama (LAN-ovima).

Setite se: da je LAN računarska mreža koja je koncentrisana u geografskom području, kao što je zgrada ili univerzitetsko naselje. Kada korisnik pristupa Internetu iz univerzitetskog ili korporativnog naselja, pristup je gotovo uvek putem LAN-a. U ovoj vrsti pristupa Internetu, korisnikov računar je Čvor LAN-a, a LAN obezbeđuje pristup Internetu preko rutera, kao sto je prikazano na slici 5.15. Brzina

prenosa  $R$  većine LAN-ova je veoma velika. Čak i početkom 80-ih godina, bili su uobičajeni LAN-ovi od 10 Mb/s. Danas su uobičajeni LAN-ovi od 100 Mb/s, a na raspolaganju su i LAN-ovi od 1 i 10 Gb/s.

U 80-im i početkom 90-ih godina, bile su popularne dve tehnologije LAN-ova. Prvu klasu činili su Ethernet LAN-ovi (takođe poznati kao LAN-ovi 802.3 [Spur-geon 2002]), koji su zasnovani na slučajnom pristupu. Druga klasa tehnologija za LAN sastoji se od tehnologija prosleđivanja žetona, uključujući **Token Ring** (takođe poznat i kao IEEE 802.5) i FDDI (poznata takođe kao interfejs optički distribuiranih podataka [Jain 1994]). S obzirom na to da ćemo detaljnije istražiti tehnologije Ethernet u odeljku 5.5, ovde ćemo se usredosrediti na LAN-ove sa prosleđivanjem žetona. Naše objašnjanje tehnologije prosleđivanja žetona je namerno kratko, zato što su one postale relativno sporedan igrač u sučeljavanju sa neumornom konkurenčijom Etherneta. I pored toga, da bismo obezbedili primere tehnologije prosleđivanja žetona i da bismo svemu ovome dali izvesnu istorijsku perspektivu, korisno je da kažemo neku reč i o tehnologiji Token Ring.

U Token Ring LAN-u,  $N$  čvorova LAN-a (računari i ruteri) povezani su u prsten direktnim linkovima. Token ring topologija definiše redosled prosleđivanja žetona. Kada čvor dobije žeton i pošalje okvir, okvir se propagira kroz ceo prsten, stvarajući tako virtualni difuzni kanal. Odredišni čvor čita okvir sa medijuma sloja linka dok okvir prolazi. Čvor koji šalje okvir ima odgovornost da ukloni okvir sa prstena.



**Slika 5.15** • Korisnički računari pristupaju Internetovom veb serveru preko LAN-a. Difuzni kanal između korisničkog računara i rutera sastoji se od jednog „linka“.

FDDI je projektovan za geografski veće LAN-ove, uključujući tu i takozvane **gradske računarske mreže** (*metropolitan area network, MAN*). Za geografski velike mreže (koje se prostiru na više kilometara), ne bi bilo efikasno da se okvir propagira nazad otpremnom čvoru po prolasku kroz odredišni čvor. Kod kanala FDDI odredišni čvor uklanja okvir sa prstena. (Strogo govoreći, FDDI nije pravi difuzni kanal, zato što svaki čvor ne prima svaki poslati okvir.)

## 5.4 Adresiranje sloja linka

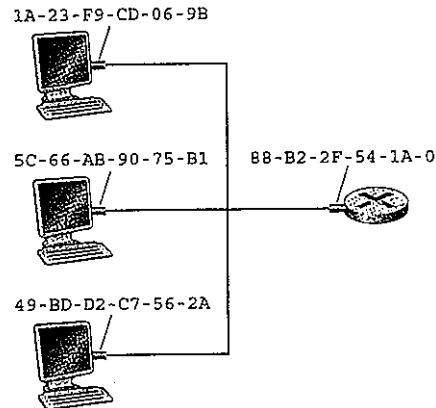
Čvorovi - odnosno, računari i ruteri - imaju adrese sloja linka. To bi vas moglo iznenaditi, ako se setite iz poglavlja 4 da čvorovi imaju i adrese mrežnog sloja. Mogli biste se zapitati, zašto je potrebno da imamo adrese i u mrežnom sloju i u sloju linka? Pored opisivanja sintakse i funkcije adresa sloja linka, u ovom odeljku se nadamo da ćemo vam objasniti zašto su dva sloja adresa korisni i, u stvari, neophodni.

Pored toga, u ovom odeljku ćemo objasnitи dva važna pojma koji se odnose na adrese. Prvi je protokol razrešavanja adresa (ARP), koji obezbeđuje prevodenje IP adresa u adresu sloja linka. Drugi je protokol za dinamičko konfigurisanje glavnog računara (DHCP). O usluzi DHCP govorili smo u poglavlju 4; ovde ćemo prime-niti naše znanje o adresama sloja linka da bi opisali kako se implementira usluga DHCP.

### 5.4.1 MAC adrese

U stvarnosti, nema čvor (odnosno, računar ili ruter) adresu sloja linka; umesto toga njegov adapter ima adresu sloja linka. To je ilustrovano na slici 5.16. Adresa sloja linka se drugačije zove **LAN adresa**, **fizička adresa**, ili **MAC adresa** (*Media Access Control*, kontrola pristupa medijumima). Kako je MAC adresa izgleda najpopularniji termin, mi ćemo se od sada na adresu sloja linka pozivati kao na MAC adrese. Za većinu LAN-ova (uključujući tu Ethernet i bežične LAN-ove 802.11), MAC adresa je dužine šest bajtova, što daje  $2^{48}$  mogućih MAC adresa. Kao što je prikazano na slici 5.16, uobičajeno je da se ove šestobajtne adrese izražavaju u heksadecimalnoj notaciji, gde je svaki bajt izražen kao par heksadecimalnih brojeva. Značajna činjenica o MAC adresama je da su one stalne - kada se adapter proizvede, LAN adresa se upiše u njegovu ROM memoriju.

Zanimljiva osobina LAN adresa je da ne postoje dva adaptora koja imaju iste adrese. To bi moglo da izgleda iznenadjuće, s obzirom na to da se adaptori proizvode u mnogo različitim zemaljama i kompanija. Kako su kompanije koje proizvode adaptore u Tajvanu sigurne da koriste adrese koje su različite od onih koje koriste kompanije koje proizvode adaptore u Belgiji? Odgovor je da IEEE upravlja prostorom fizičkih adresa. Posebno, kada kompanija želi da proizvodi adaptore, ona kupuje deo adresnog prostora od  $2^{24}$  adresa, za iznos nominalne članarine. IEEE dodeljuje deo od  $2^{24}$  adresa, fiksirajući prva 24 bita fizičke adrese i dozvoljavajući kompaniji da pravi jedinstvene kombinacije od poslednja 24 bita za svaki adapter.



**Slika 5.16** ♦ Svaki adapter povezan u LAN ima jedinstvenu LAN adresu.

MAC adresa adaptora ima linearnu strukturu (za razliku od hijerarhijske strukture) i ne menja se bez obzira na to gde se adapter nalazi. Prenosni računar sa Ethernet karticom ima istu MAC adresu, bez obzira na to gde je računar. PDA sa interfejsom 802.11 ima uvek istu MAC adresu, bez obzira gde je PDA. Setite se da, za razliku od toga, IP adresa ima hijerarhijsku strukturu (odnosno, deo za mrežu i deo za računar) i da IP adresa čvora mora da se menja kada se računar premešta. MAC adresa adaptora analogna je nečijem broju socijalnog osiguranja, koji takođe ima linearnu strukturu i ne menja se bez obzira na to gde ide njegov vlasnik. IP adresa je analogna nečijoj poštanskoj adresi koja je hijerarhijska i mora da se promeni kad god se ta osoba preseli. Kao što je nekog korisno da ima i poštansku adresu i broj socijalnog osiguranja, tako je za čvor korisno da ima i adresu mrežnog sloja i MAC adresu.

Kao što smo opisali na početku ovog odeljka, kada adapter želi da pošalje okvir nekom odredišnom adaptoru u okviru istog LAN-a, adapter pošiljaoca umeće MAC adresu adaptora odredišta u okvir i onda šalje okvir u LAN. Ako je LAN difuzan (kao što su 802.11 i mnogi Ethernet LAN-ovi), okvir primaju i obraduju svi drugi adapteri u LAN-u. Posebno, svaki adapter koji prima okvir proveriće da vidi da li MAC adresa odredišta odgovara njegovoj sopstvenoj MAC adresi. Ako postoji podudarnost, adapter izdvaja priloženi datagram i prosleduje ga protokolu naviše, svom nadređenom Čvoru. Ako nema podudarnosti, adapter odbacuje okvir, bez prosleđivanja datagrama mrežnom sloju. Dakle, samo će adapter u odredišnom čvoru poslati signal prekida svom nadređenom procesoru u trenutku kada prima okvir.

## PRINCIPI U PRAKSI

### NEZAVISNI SLOJEVI

Ima više razloga zbog kojih čvorovi imaju LAN adrese pored adresa mrežnog sloja Prvo, LAN-ovi su projeklovi za proizvoljne protokole mrežnog sloja, a ne samo za IP i Internet. Ako bi adapteri trebalo da dobiju dodeljene IP adrese, a ne „neutralne“ MAC adrese, onda oni ne bi mogli iako da podržavaju druge protokole mrežnog sloja (na primer IPX ili DECnet). Drugo, ako bi adapteri trebalo da koriste adrese mrežnog sloja umesto MAC adresa, adresa mrežnog sloja bi trebalo da bude uskladištena u RAM memoriji adaptora i ponovo konfigurisana svaki put kada bi se adopter premestio (ili uključilo napajanje). Druga opcija bi bila da se ne koriste nikakve adrese u adapterima i da svaki adapter prosleđuje podatke (po pravilu, IP datagram) svakog primljenog okvira čvoru kome pripada. Ovaj čvor bi onda mogao da proveri da li mu odgovara adresa mrežnog sloja. Problem sa ovom opcijom je u tome što bi čvor bio prekidan svakim okvirom posla-tim u LAN-u, uključujući tu i okvire koji su namenjeni drugim čvorovima na istom difuznom ■ LAN-u. Sve u svemu, da bi slojevi bili široko nezavisni gradivni blokovi u mrežnoj arhitekturi, mnogi od njih treba do imaju sopstvenu adresnu šemu. Sada možemo da vidimo tri različite vrste adresa: imena čvorova za aplikacijski sloj, IP adrese za mrežni sloj i LAN adrese za sloj veze podataka.

Međutim, ponekad predajni adapter *zaista* želi da svi drugi adapteri u LAN-u prime i obrade okvir koji on šalje. U tom slučaju, adapter umeće specijalnu difuznu MAC adresu u polje adrese odredišta okvira. Kod LAN-ova koji koriste šestobajtne adrese (kao što su Ethernet i LAN-ovi sa prosleđivanjem žetona), difuzna adresa je niz od 48 susednih jedinica (odnosno, FF-FF-FF-FF-FF-FF u heksadecimainoj notaciji).

### 5.4.2 Protokol razrešavanja adresa (ARP)

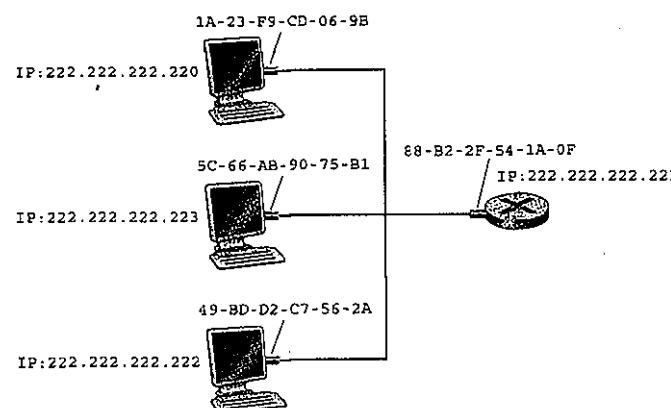
S obzirom na to da postoje i adrese mrežnog sloja (na primer, Internet IP adrese) i adrese sloja linka (odnosno, MAC adrese), potrebno je da se izvrši prevođenje između njih. U slučaju Interneta, to je zadatak protokola za razrešavanje adresa (*address resolution protocol*, ARP) [RFC 826].

Da biste razumeli potrebu za protokolom kao što je ARP, pogledajte mrežu prikazanu na slici 5.17. U ovom jednostavnom primeru, svaki čvor ima IP adresu, a svaki adapter čvora MAC adresu. Kao i obično, IP adrese su prikazane u decimalnoj notaciji sa tačkom, a MAC adrese u heksadecimainoj notaciji. Sada prepostavite da

čvor sa IP adresom 222.222.222.220 želi da pošalje IP datagram čvoru 222.222.22-2.222. (Na primer, odredišni čvor 222.222.222.222 može da bude veb server, a predajni Čvor 222.222.222.220 je mogao da utvrdi IP adresu veb servera iz DNS-a). U ovom primeru, i izvorni i odredišni čvor su na istoj mreži (LAN), u adresnom smislu iz odeljka 4.4.3. Da bi poslao datagram, predajni čvor mora da obezbedi svom adapteru ne samo IP datagram, nego i MAC adresu čvora 222.222.222.222. Uz dati IP datagram i MAC adresu, adapter predajnog čvora će konstruisati okvir sloja veze podataka koji sadrži MAC adresu prijemnog čvora i poslaće okvir u LAN.

Važno pitanje koje se postavlja u ovom odeljku je kako predajni čvor određuje MAC adresu za odredišni čvor sa IP adresom 222.222.222.222? Kao što možete da pogodite, on koristi ARP. ARP modul u predajnom čvoru kao ulaz uzima bilo koju IP adresu na istom LAN-u i враћa odgovarajuću MAC adresu. U našem primeru, predajni čvor 222.222.222.220 obezbeđuje svom ARP modulu IP adresu 222.222.2-22.222, a ARP modul vraća odgovarajuću MAC adresu 49-BD-D2-C7-56-2A.

Dakle, vidimo da ARP prevodi IP adresu u MAC adresu. To je dosta analogno DNS-u (razmatranom u odeljku 2.5), koji prevodi imena računara u IP adrese. Međutim, značajna razlika između ova dva prevodioca je u tome što DNS prevodi imena računara za računare bilo gde na Internetu, dok ARP prevodi IP adrese samo za čvorove u okviru istog LAN-a. Ako je čvor u Kaliforniji pokušavao da upotrebi ARP kako bi preveo IP adresu čvora u Misisipiju, ARP bi vratio grešku.



**Slika 5.17** ♦ Svaki čvor u LAN-u ima IP adresu, a svaki adapter čvora MAC adresu.

| IP Address      | MAC Address       | TTL      |
|-----------------|-------------------|----------|
| 222.222.222.221 | 88-B2-2F-54-1A-0F | 13:45:00 |
| 222.222.222.223 | 5C-66-AB-90-75-B1 | 13:52:00 |

**Slika 5.18** ♦ Moguća ARP tabela u čvoru 222.222.222.220

Sad kada smo objasnili šta radi ARP, hajde da pogledamo kako on to radi. Modul ARP u svakom čvoru ima tabelu u svojoj RAM memoriji koja se zove ARP tabela. Ova tabela sadrži preslikavanja IP adresa u MAC adrese. Na slici 5.18 prikazano je kako bi mogla da izgleda ARP tabela u čvoru 222.222.222.220. Za svako preslikavanje adresa, tabela takođe sadrži i stavku roka trajanja (*time to live*, TTL), koja pokazuje kada će adresa biti izbrisana iz tabele. Po pravilu, vreme isteka važnosti je 20 minuta od kada se stavka postavi u ARP tabelu.

Sada prepostavite da čvor 222.222.222.220 želi da pošalje datagram sa odredišnom IP adresom drugog čvora na istom LAN-u. Predajni Čvor treba da dobije MAC adresu čvora odredišta, na osnovu date IP adrese tog čvora. Taj zadatak je jednostavan, ako ARP tabela predajnog čvora ima stavku za Čvor odredišta. Ali, šta ako ARP tabela trenutno nema stavku za čvor odredišta? Konkretno, prepostavite da čvor 222.222.222.220 želi da pošalje datagram čvoru 222.222.222.222. U tom slučaju, predajni čvor koristi protokol ARP da bi odredio adresu. Prvo, predajni čvor konstruiše specijalan paket koji se zove ARP paket. ARP paket ima nekoliko polja, uključujući tu i prijemne IP i MAC adrese. Paketi i za ARP upit i za odgovor imaju isti format. Namena ARP upitnog paketa je da se pošalje upit svim drugim čvorovima u LAN-u sa namerom da se odredi MAC adresa koja odgovara traženoj IP adresi.

Ako se vratimo našem primeru, čvor 222.222.222.220 prosleduje ARP upitni paket adapteru, zajedno sa indikacijom da bi adapter trebalo da pošalje paket na difuznu LAN adresu, odnosno na FF-FF-FF-FF-FF-FF Adapter enkapsulira ARP paket u okvir sloja veze podataka, koristi difuznu adresu kao adresu odredišta tog okvira i predaje okvir u LAN. Podsetivši se na našu analogiju sa brojem socijalnog osiguranja/poštanskom adresom, zapazite daje ARP upit ekvivalentan osobi koja više u prostoriji neke kompanije (na primer, AnvCorp) prepunoj boksova: „Koji je broj socijalnog osiguranja osobe čija poštanska adresa je boks 13, soba 112, AnvCorp, Palo Alto, CA?“ Okvir koji sadrži ARP upit primaju svi drugi adapteri u LAN-u, i (zbog difuzne adrese), svaki adapter prosleduje ARP paket unutar okvira naviše do svog računarskog čvora. Svaki čvor proverava da li njegova IP adresa odgovara IP adresi odredišta u ARP paketu. Samo čvor u kom postoji podudarnost vraća odzivni ARP paket čvoru koji je pitao. Upitni čvor (222.222.222.220) onda može da ažurira svoju ARP tabelu i pošalje svoj IP datagram.

Ima nekoliko zanimljivih stvari koje treba zapaziti u vezi sa ARP protokolom. Prvo, upitna ARP poruka se šalje unutar difuznog okvira, dok se odzivna ARP poruka šalje unutar standardnog okvira. Pre nego što nastavite sa čitanjem, trebalo bi da razmislite zastoje to tako. Drugo, ARP je „priključi i radi“ (*plug and play*); to znači da se ARP tabela čvora pravi automatski - ne treba da je konfiguriše administrator sistema. A ako se, na kraju, čvor isključi iz LAN-a, njegova stavka se konačno briše iz tabele čvorova koji ostaju u podmreži.

#### Slanje datagrama čvoru izvan LAN-a

Sada bi trebalo da bude jasno da ARP funkcioniše kada čvor želi da pošalje datagram drugom Čvoru *u istoj podmreži*. (Podmreže su precizno definisane u odeljku 4.4.2). Ali, hajde da sada pogledamo složeniju situaciju, kada čvor u podmreži želi da pošalje datagram mrežnog sloja čvoru *izvan podmreže* (odnosno, preko rutera u drugu podmrežu). Hajde da govorimo o tom pitanju u kontekstu slike 5.19, gde je prikazana jednostavna mreža koja se sastoji od dve podmreže, međusobno povezane pomoću rutera.

Ima nekoliko zanimljivih stvari koje treba zapaziti u vezi sa slikom 5.19. Prvo, postoje dve vrste čvorova: računari i routeri. Svaki računar ima tačno jednu IP adresu i jedan adapter. Ali, kao što smo govorili u poglavlju 4, ruter ima IP adresu za *svaki* od svojih interfejsa. Svaki interfejs rutera takođe ima sopstveni ARP modul (u ruteru) i adapter. S obzirom na to da ruter na slici 5.19 ima dva interfejsa, on ima dve IP adrese, dva ARP modula i dva adaptera. Naravno, svaki adapter u mreži ima svoju sopstvenu MAC adresu.

Takođe zapazite da podmreža 1 ima mrežnu adresu 111.111.111/24, a da podmreža 2 ima mrežnu adresu 222.222.222/24. Dakle, svi interfejsi priključeni na podmrežu 1 imaju adrese u obliku 111.111.111.xxx, a svi interfejsi priključeni na podmrežu 2 imaju oblik 222.222.222.xxx.

Hajde da sada ispitamo kako bi računar na podmreži 1 poslao datagram računaru na podmreži 2. Posebno, pretpostavite da računar 111.111.111.111 želi da pošalje IP datagram računaru 222.222.222.222. Kao i obično, računar pošiljalac prosledjuje datagram svom adapteru. Ali, računar pošiljalac mora takođe da ukaže svom adapteru odgovarajuću MAC adresu odredišta. Koju bi MAC adresu adapteru trebalo da upotrebi? Mogli bismo da se usudimo da nagadamo kako je odgovarajuća MAC adresa ona koju ima adapter za računar 222.222.222.222, odnosno 49-BD-D2-C7-56-2A. Međutim, to nagadjanje je pogrešno. Ako bi predajni adapter upotrebio tu MAC adresu, nijedan od adaptera unutar podmreže 1 ne bi prosledio IP datagram naviše u svoj mrežni sloj, zato što adresa odredišta okvira ne bi odgovarala MAC adresi bilo kog adaptora u podmreži 1. Datagram bi samo preminuo i otisao u raj za datagrame<sup>1</sup>.

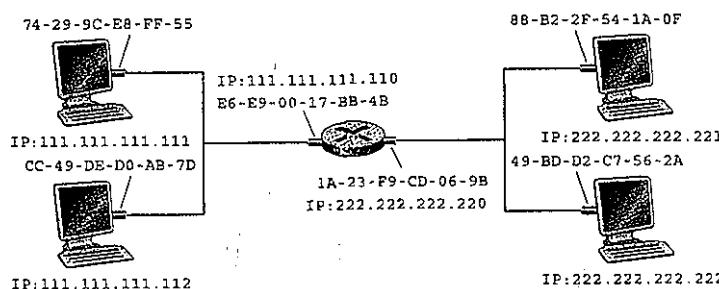
Ako pažljivo pogledamo sliku 5.19 vidimo da, kako bi datagram iz 111.111.111.111.11 otisao u Čvor u podmreži 2, on mora prvo da se pošalje interfejsu rutera 111.111.111.110. Dakle, odgovarajuća MAC adresa za okvir je adresa adaptora za interfejs rutera 111.111.111.110, odnosno E6-E9-00-17-BB-4B. Kako čvor pošiljaoca dolazi do MAC adrese U 1.111.II 1.110? Naravno, koristeći ARP! Jednom kada predajni adapter ima tu MAC adresu, on pravi okvir i šalje ga u podmrežu 1. Adapter rutera na podmreži 1 vidi daje okvir sloja linka podataka adresiran njemu i zato prosledjuje okvir mrežnom sloju rutera. Ura! IP datagram je uspešno prenet od izvornog računara do ruteru! AH, nismo završili. Još uvek treba da prenesemo datagram od rutera do odredišta. Ruter sada treba da odredi tačan interfejs na koji treba da se prosledi datagram. Kao što smo objasnili u poglavlju 4, to se radi tako što se pogleda u tabelu prosledivanja u ruteru. Tabele prosledivanja kaže ruteru da datagram treba da se prosledi preko ruterovog interfejsa 222.222.222.220. Taj interfejs onda prosleduje datagram svom adapteru, koji ga enkapsulira u novi okvir i šalje okvir u podmrežu 2.

Ovog puta, MAC adresa odredišta je stvarno MAC adresa krajnjeg odredišta. A kako ruter dobija tu MAC adresu odredišta? Naravno, od ARP-a!

ARP za Ethernet definisan u RFC-u 826. Lep uvod u ARP dat je u uputstvu za TCP/IP, RFC 1180. Mi ćemo detaljnije ispitati ARP u problemima za domaće zadatke.

#### 5.4.3 Protokol za dinamičko konfigurisanje glavnog računara

Kada smo u poglavlju 4 govorili o IP adresama, ukratko smo razmotrili uslugu koju pruža DHCP, protokol koji se dosta koristi u korporativnim, univerzitetskim i kućnim LAN-ovima za dinamičko dodeljivanje IP adresa računarima. Pošto smo uslugu opisali u poglavlju 4, sada ćemo upotrebiti naše novousvojeno znanje o MAC adresama da bismo opisali kako DHCP stvarno radi.



**Slika 5.19** ♦ Dve podmreže međusobno povezane ruterom

DHCP je protokol između klijenta i servera. Klijent je obično novodošavši računar koji želi da dobije informacije o konfiguraciji mreže, uključujući i sopstvenu IP adresu. U najprostijem slučaju, svaka podmreža (adresnom smislu opisanom u odeljku 4.4.2) će imati DHCP server. Ako nema servera u podmreži, potreban je agent za prenos DHCP (obično ruter) koji zna adresu DHCP servera za tu mrežu. Na slici 5.20 prikazan je DHCP server priključen na podmrežu 223.1.2/24, sa ruterom koji služi kao prenosni agent za dolazeće klijente koji su priključeni na podmreže 223.1.1/24 i 223.1.3/24.

Za novoprispeli računar, protokol DHCP je proces u četiri koraka:

- ◆ **Otkrivanje DHCP-a.** Prvi zadatak novoprispelog računara je da pronađe DHCP server sa kojim treba da stupi u interakciju. To se radi upotrebom **DHCP poruke za otkrivanje**, koju klijent šalje unutar UDP paketa na port 67. UDP paket se enkapsulira u IP datagramu. Ali, kome bi taj datagram trebalo da se pošalje? Računar čak ne zna ni IP adresu mreže na koju je priključen, što je mnogo manje od DHCP servera za tu mrežu. U takvim uslovima, DHCP klijent stvara IP datagram koji sadrži njegovu DHCP poruku za otkrivanje, uz difuznu odre-

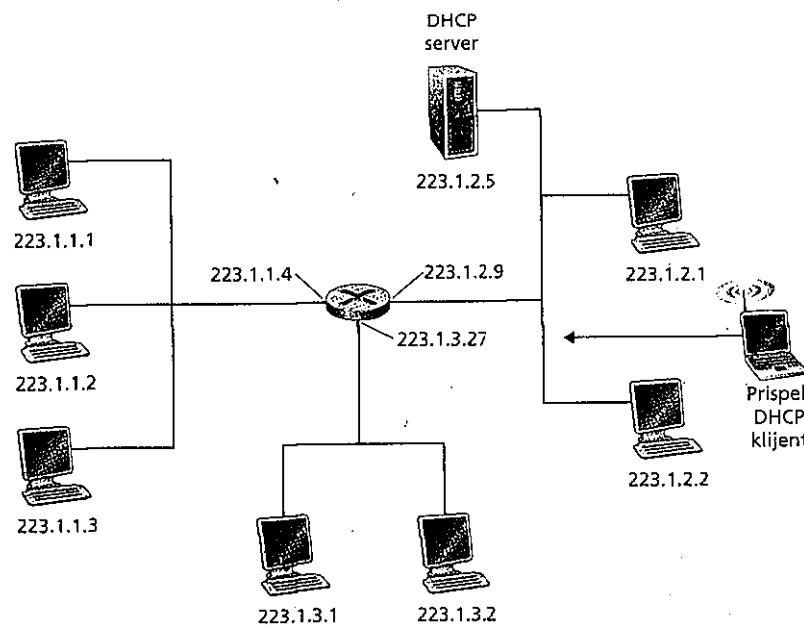
dišnu IP adresu 255.255.255.255 i izvornu [P adresu „ovog računara“ 0.0.0.0. DHCP klijent prosleduje IP datagram svom adapteru, koji enkapsulira datagram u okvir sloja linka. Taj okvir sloja linka obuhvata difuznu MAC adresu (FF-FF-FF-FF-FF-FF) u polju za adresu odredišta. DHCP klijent onda šalje u mrežu difuzni okvir, koji sadrži poruku za otkrivanje. Taj difuzni okvir će primiti svi adapteri u mreži. Ako je DHCP server priključen na istu podmrežu, on će obraditi enkapsuliranu poruku za otkrivanje (pogledajte dalje u tekstu); ako je DHCP prenosni agent priključen na podmrežu, on će proslediti okvir mreži sa DHCP serverom. (Taj preneseni okvir će imati različitu izvornu MAC adresu.) Poruka za otkrivanje sadrži ID transakcije koji dozvoljava sledećim odzivima da budu uskladeni sa zahtevom za otkrivanje.

- ◆ **Ponuda (e) DHCP servera.** DHCP server koji prima DHCP poruku za otkrivanje odgovara klijentu **DHCP porukom za ponudu**. Kako u mreži može da bude prisutno više DHCP servera, klijent može da se nade u zavidnom položaju daje u stanju da bira između više ponuda. Svaka poruka servera za ponudu sadrži ID transakcije primljene poruke za otkrivanje, predloženu IP adresu za klijenta, mrežnu masku i **vreme zakupa IP** adrese - vreme za koje će IP adresa biti važeća. Uobičajeno je da server postavi vreme zakupa na više sati ili dana (Droms 1999). Okvir sloja linka koji sadrži IP datagram koji sadrži UDP segment koji sadrži DHCP poruku za ponudu se onda šalje prispelom klijentu. (Ako vam se od ove ugnježdene enkapsulacije vrti u glavi, vratite se korak natrag i ponovo pročitajte odeljak 1.7.)
- ◆ **DHCP zahtev.** Novoprispeli klijent će izabrati između jedne ili više ponuda servera i odgovorice na izabranu ponudu DHCP porukom za zahtev, vraćajući nazad parametre konfiguracije.
- ◆ **DHCP A CK.** Server odgovara na DHCP poruku za zahtev porukom DHCP ACK, potvrđujući zahtevane parametre.

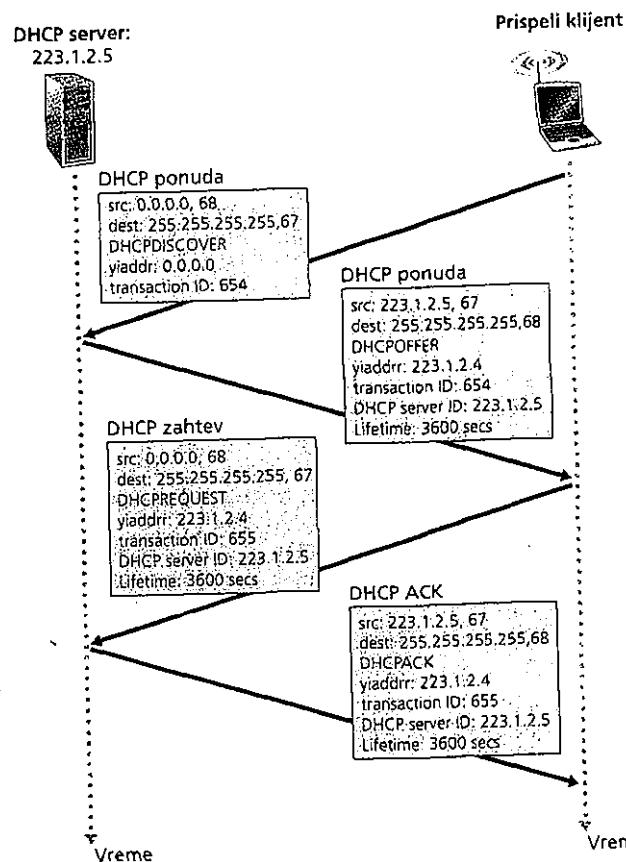
Jednom kada klijent primi DHCP ACK, interakcija je upotpunjena i klijent može da koristi IP adresu dodeljenu od DHCP za dato vreme zakupa. Kako klijent može poželeti da koristi svoju adresu i posle isteka zakupa, DHCP takođe obezbeđuje mehanizam koji dopušta klijentu da obnovi svoj zakup IP adrese.

Jednostavna DHCP interakcija klijent-server prikazana je na slici 5.21, za postavku mreže sa slike 5.20. Na toj slici, yiaddr („vaša Internet adresa“) pokazuje adresu koja se dodeljuje novoprispelom klijentu.

Vrednost DHCP *plug-and-play* mogućnosti je jasna. Zamislite studenta koji prelazi iz učionice u spavaću sobu sa prenosnim računarcem, pridružuje se novoj podmreži i tako dobija novu IP adresu na svakoj lokaciji. Nezamislivo je da bi administrator sistema morao da rekonfiguriše prenosne računare na svakoj lokaciji, a malo studenata (izuzev onih koji pohađaju nastavu iz računarskog umrežavanja!) bi imalo znanja da ručno konfiguriše svoje prenosne računare. Međutim, sa aspekta pokretljivosti, DHCP zaista ima nedostatke. Kako se nova IP adresa dobija od DHCP



**Slika 5.20** ♦ DHCP scenario klijent-server



Slika 5.21 ♦ DHCP interakcija klijent-server

svaki put kada se čvor poveže na novu podmrežu, veza sa udaljenom aplikacijom ne može da se održava kada se pokretni čvor kreće između podmreža. U poglavlju 6 ćemo ispitati pokretni IP - nedavno proširenje IP infrastrukture koje dozvoljava pokretnom čvoru da koristi jednu stalnu adresu dok se kreće između podmreža.

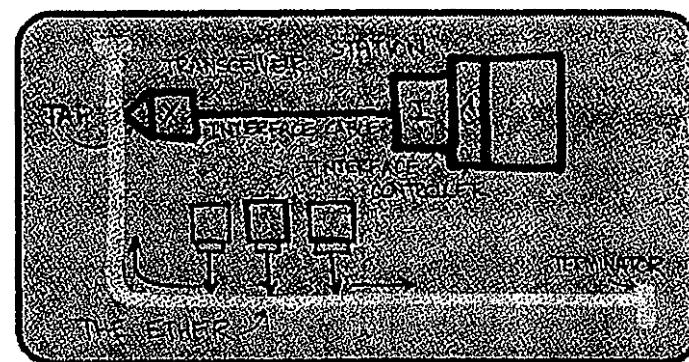
Dodatni detalji o DHCP mogu da se pronađu u [Drooms 1999] i [dhc 2004]. Implementacija DHCP u izvornom kodu raspoloživa je na Internet Systems Consortium [ISC 2004].

## 5.5 Ethernet

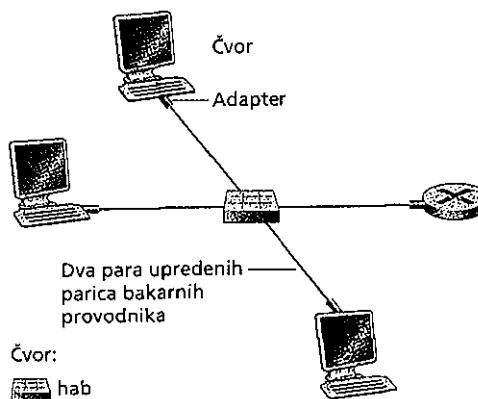
Ethernet je u priličnoj meri preuzeo tržište LAN-a. U 1980-im i početkom 1990-ih godina, Ethernet se suočavao sa mnogobrojnim izazovima od strane drugih tehnologija LAN, uključujući tu Token Ring, FDDI i ATM. Neke od tih drugih tehnologija uspele su da zauzmu deo tržišta za nekoliko godina. Ali, od kada je pronađen sredinom 1970-ih godina, Ethernet je nastavio da se razvija i raste i zadržao je svoj većinski deo tržišta. Danas je Ethernet daleko preovlađujuća tehnologija za LAN i verovatno će to ostati u doglednoj budućnosti. Moglo bi se reći da je Ethernet za lokalno umrežavanje računara ono stoje Internet bio za globalno umrežavanje.

Ima mnogo razloga za Ethernetov uspeh. Prvo, Ethernet je bio prvi široko pri-menjivani brzi LAN. S obzirom na to daje Ethernet odavno u upotrebi, administratori mreža su se blisko sa njim upoznali - sa njegovim čudima i dosetkama - pa su nerado prelazili na druge LAN tehnologije kada bi one stupale na scenu. Drugo, Token'Ring, FDDI i ATM bili su složeniji i skuplji od Etherneta, što je još više obeshrabriло administratore mreža da uđu u neke promene. Treće, najjači razlog za prelazak na drugu LAN tehnologiju (kao što su FDDI i ATM) obično je bila veća brzina nove tehnologije; međutim, Ethernet je uvek uzvraćao udarac, proizvodeći verzije koje su radile jednakom ili još većom brzinom. Komutirani Ethernet je takođe predstavljen početkom 1990-ih godina, što je još više povećalo njegovu efektivnu brzinu podataka. Najzad, zbog velike popularnosti Etherneta, njegov hardver (posebno adapteri, habovi i komutatori) je postao roba široke potrošnje i izvanredno je jeftin.

Originalni Ethernet LAN su pronašli Bob Metcalf i David Boggs, sredinom 1970-ih godina. Na slici 5.22 prikazana je Metcalfova šema Etherneta.



Slika 5.22 ♦ Originalna Metcalfova konstrukcija dovela je do standarda Ethernet 10Base5, koji je obuhvatao interfejsni kabl za povezivanje Ethernet adaptora sa spoljašnjim primopredajnikom.



**Slika 5.23 ♦ Topologija zvezde za Ethernet. Čvorovi su međusobno povezani pomoću haba.**

Na ovoj slici ćete zapaziti daje originalni Ethernet LAN koristio magistralu za međusobno povezivanje čvorova. Ta topologija magistrale se zadržala tokom 1980-ih i najvećeg dela 1990-ih godina; posebno, tehnologija Ethernet 10Base2, koja je koristila tanki koaksijalni kabl za magistralu, bila je izuzetno popularna u 1990-im godinama. Međutim, izuzev retkih nasledenih, gotovo sve Ethernet instalacije danas koriste topologiju zvezde, kao što je prikazano na slici 5.23. U središtu topologije zvezde je hab ili komutator. Uskoro ćemo detaljnije govoriti o habovima i komutatorima. Izvanredan izvor onlajn informacija o Ethernetu je Spurgeon-nova Ethernet veb stranica [Spurgeon 2004].

### 5.5.1 Struktura Ethernetovog okvira

Ethernetov okvir je prikazan na slici 5.23. Jednom kad razumemo Ethernetov okvir, znademo već mnogo o Ethernetu. Da bismo našu diskusiju o Ethernetovom okviru stavili u opipljiv kontekst, hajde da razmatramo slanje IP datagrama od jednog do drugog računara, gde su oba računara u istom Ethernet LAN-u (na primer, Ethernet LAN na slici 5.23). Mada je koristan teret našeg Ethernet okvira IP datagram, zapazimo, ipak, da Ethernet može da prenosi i druge pakete mrežnog sloja. Neka predajni adapter, adapter A, ima MAC adresu AA-AA-AA-AA-AA-AA, a prijemni adapter, adapter B, ima MAC adresu BB-BB-BB-BB-BB-BB. Predajni adapter enkapsulira IP datagram unutar Ethernetovog okvira i prosledjuje okvir na fizički sloj. Prijemni adapter prima okvir od fizičkog sloja, izdvaja IP datagram i prosledjuje ga mrežnom sloju. U tom kontekstu, hajde da ispitamo šest polja Ethernetovog okvira, kao stoje prikazano na slici 5.24;



**Slika 5.24 ♦ Struktura Ethernetovog okvira**

- ♦ **Polje za podatke (46 do 1500 bajtova).** Ovo polje nosi IP datagram. Maksimalna jedinica prenosa (MTU) Ethernta je 1500 bajtova. To znači da ako je IP datagram veći od 1500 bajtova, računar mora da ga podeli, kao sto je objašnjeno u odeljku 4.4.4. Minimalna veličina polja za podatke je 46 bajtova. To znači da ako je datagram manji od 46 bajtova, polje za podatke mora da bude „popunjeno“ do 46 bajtova. Kada se koristi popunjavanje, podaci koji se prosleđuju mrežnom sloju sadrže i popunu u IP datagram. Mrežni sloj koristi polje za dužinu u zaglavju IP datograma, da bi uklonio popunu.
- ♦ **Adresa odredišta (6 bajtova).** Ovo polje sadrži LAN adresu adaptora odredišta, u našem primeru BB-BB-BB-BB-BB-BB. Kada adapter B primi Ethernet okvir čija je adresa odredišta BB-BB-BB-BB-BB-BB ili difuzna MAC adresa, on prosledjuje sadržaj polja za podatke okvira u mrežni sloj; ako primi okvir sa bilo kojom drugom MAC adresom, on ga odbacuje.
- ♦ **Adresa izvora (6 bajtova).** Ovo polje sadrži LAN adresu adaptora koji predaje okviru LAN, u našem primeru AA-AA-AA-AA-AA-AA.
- ♦ **Polje za tip (2 bajta).** Polje tipa dozvoljava Ethernetu da multipleksira protokole mrežnog sloja. Da bismo razumeli ovu zamisao, treba da imamo na umu da računari pored IP-a mogu da koriste i druge protokole mrežnog sloja. U stvari, dati računar može da podržava višestruke protokole mrežnog sloja, koristeći različite protokole za različite aplikacije. Iz tog razloga, kada Ethernet okvir stigne u adapter B, adapter B treba da zna kom protokolu mrežnog sloja treba da prosledi (odnosno demultipleksira) sadržaj polja za podatke. IP i drugi protokoli mrežnog sloja (na primer, Novell IPX ili AppleTalk) imaju svaki svoj, standardizovani broj koji određuje tip. Pored toga, ARP protokol (objašnjen u prethodnom odeljku) ima sopstveni broj tipa. Zapazite daje polje tipa analogno polju protokola u datagramu mrežnog sloja i poljima broja porta u segmentu transportnog sloja; svako od tih polja služi da spoji protokol jednog sloja sa slojem koji se nalazi iznad njega.
- ♦ **Ciklička provera redundantnosti (CRC) (4 bajta).** Kao stoje objašnjeno u odeljku 5.2.3, svrha polja CRC je da dozvoli prijemnom adaptoru, adaptoru B, da otkrije da li je nastala bilo kakva greška u okviru, odnosno, da li su bitovi u okviru promenili vrednost. Uzroci grešaka bitova obuhvataju slabljenje snage signala i elektromagnetne smetnje iz okoline koje prodiru u Ethernet kablove i interfejsne kartice. Greške se otkrivaju na sledeći način. Kada računar A konstruiše Ether-

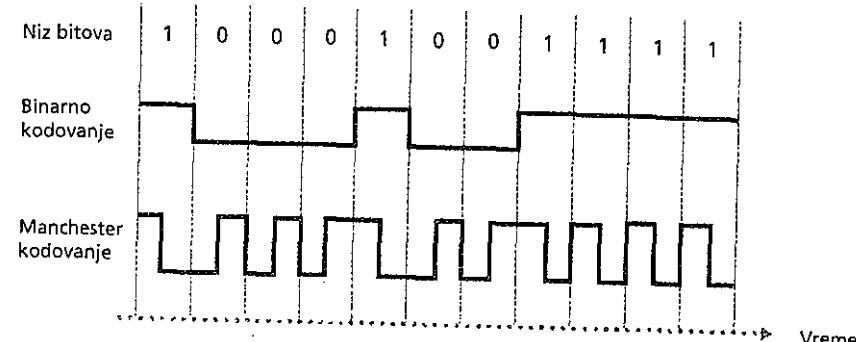
net okvir, on izračunava polje CRC, koje se dobija iz svih ostalih bitova unutar okvira, izuzev bitova preambule. Kada računar B prima okvir, on primenjuje iste matematičke operacije na okvir i proverava da li je rezultat jednak onome što se već nalazi u polju CRC. Ta operacija se zove CRC provera. Ako CRC provera ne uspe (odnosno, rezultat izračunavanja nije jednak sadržaju polja CRC), onda računar B zna da postoji greška u okviru.

♦ *Preamble (8 bajtova).* Ethernet okvir počinje osmabajtnim poljem preambule. Svaki od prvih sedam bajtova preambule ima vrednost 10101010; poslednji bajt je 10101011. Prvih sedam bajtova preambule služe da „probude” prijemne ada-ptere i da sinhronizuju njihove generatore takta sa generatorom takta pošiljaoca. Zašto bi generatori takta bili van sinhronizacije? Imajte na umu da adapter A namerava da prenosi okvir na 10 Mb/s, 100 Mb/s ili 1 Gb/s, zavisno od vrste Ethernet LAN-a. Međutim, zato što ništa nije apsolutno savršeno, adapter A neće prenosi okvir tačnom ciljnom brzinom; uvek će biti izvesnog *odstupanja* od ciljne brzine, koje nije *a priori* poznato ostalim adapterima u LAN-u. Prijemni adapter može da se sinhronizuje sa generatorom takta adaptera A vezujući se jednostavno na bitove u prvih sedam bajtova preambule. Poslednja dva bita osmog bajta preambule (prve dve susedne jedinice) obaveštavaju adapter B da će uskoro da nađe neka „važna stvar”. Kada računar B detektuje dve susedne jedinice, on zna da su sledećih šest bajtova adresa odredišta. Adapter saopštava kada je našao kraj okvira jednostavnom detekcijom odsustva signala.

Ethernet koristi prenos u osnovnom opsegu; odnosno, adapter šalje digitalni signal direktno u difuzni kanal. Kartica interfejsa ne premešta signal u drugi frekventni opseg, kao što se to radi u ADSL i sistemima kablovskih modema. Mnoge Ethernet tehnologije (na primer, IOBaseT) takođe koriste Manchester kodovanje, kao što je prikazano na slici 5.25. Kod Manchester kodovanja, svaki bit sadrži prelaz; 1 ima prelaz odozgo na dole, dok 0 ima prelaz odozdo na gore. Razlog za Manchester kodovanje je što generatori takta u predajnom i prijemnom adapteru nisu savršeno sinhronizovani. Uključivanjem prelaska u sredini svakog bita, prijemni računar može da sinhronizuje svoj generator takta prema onome u predajnom računaru. Jednom kada se generator takta prijemnog adaptera sinhronizuje, prijemnik može da iscrta svaki bit i utvrdi da li je on 1 ili 0. Manchester kodovanje je operacija fizičkog sloja, a ne operacija sloja veze podataka; međutim, mi smo ga ukratko opisali ovde, zato što se ono koristi isključivo u Etherneru.

#### Nepouzdana usluga bez konekcije

Sve Ethernet tehnologije obezbeđuju mrežnom sloju uslugu bez konekcije. To znači da, kada adapter A želi da pošalje datagram adapteru B, onda adapter A enkapsulira datagram u Ethernet okvir i pošalje okvir u LAN, bez prethodne „sinhronizacije” (potvrđivanja spremnosti za prenos) sa adapterom B. Ova usluga bez konekcije slo-ja-2 analogna je usluzi datagrama IP sloja-3 i UDP usluzi sloja-4.



**Slika 5.25 ♦ Manchester kodovanje**



#### KRATAK OSVRT

##### BOB METKALF 1 ETHERNET

Kao doktorant na Univerzitetu Harvard početkom 1970-ih godina. Bob Melkolf je radio na ARPAnetu na MIT-u. Za vreme studija, on je takođe imao uvid u Abramsonov rod na ALOHAnetu protokolima sa slučajnim pristupom. Pošlo je završio doktorsku disertaciju, baš pre početka rada u Xeroxovom istraživačkom centru u Palo Alto (Xerox PARC), posebno je Ambramsona i njegove kolege sa Univerziteta u Havajima u toku tri meseca, gde je dobio uvid iz prve ruke u ALOHAnetu. U Xerox PARC-u, na Metkalfu su ostavili utisak računari Alto, koji su u mnogo čemu bili prethodnici personalnih računara iz 1980-ih godina. Metkalf je sagledao potrebu da umreži te računare na jeftin način. Tako je, naoružan svojim znanjem o ARPAnetu, ALOHAnetu i protokolima sa višestrukim prislušnjacima, Metkalf - zajedno sa kolegom Davidom Bogsom - pronašao Ethernet.

Melkalfovi i Bogsov Ethernet prvobitno je radio na 2,94 Mb/s i povezivao do 256 računara, na udaljenosti do jedne milje. Melkalf i Bogsov su uspeli da većinu istraživača u Xerox PARC-u komuniciraju preko svojih računara Alto. Metkalf je zatim napravio savez između Xeroxa, Digitala i Intela sa ciljem da uspostave Ethernet na standardu 10 Mb/s, koji je ratifikovala organizacija IEEE. Xerox nije pokazao mnogo zanimanja za komercijalizaciju Elhernefa. Godine 1979, Metkalf je osnovao sopstvenu kompaniju, 3Com, koja je razvila i komercijalizovala mrežnu tehnologiju, uključujući i tehnologiju Etherneta. Posebno, u 3Com su razvijene i dote na tržište kartice Ethernet početkom 1980-ih godina, za veoma popularne IBM-ove PC računare. Metkalf je napustio 3Com 1990. godine, kada je imao 2000 ljudi i dobit od 400 miliona USD. Početkom januara 2002. godine, 3Com zapošljava preko 8000 ljudi.

Sve Ethernet tehnologije obezbeđuju nepouzdanu uslugu mrežnom sloju. Posebno, kada adapter B prima okvir od adaptera A, on propušta okvir kroz CRC proveru, ali ne šalje potvrdu kada okvir prođe kroz CRC proveru (niti šalje negativnu potvrdu kada CRC provera ne uspe). Adapter A nema ni najmanju ideju da li je preneti okvir prošao CRC proveru. Kada CRC provera okvira ne uspe, adapter B jednostavno odbaci okvir. Ovaj nedostatak pouzdanog transporta (na sloju veze podataka) čini Ethernet jednostavnim i jeftinim. Ali, to takođe znači da niz datagrama prosledenih mrežnom sloju može da ima propuste.

Ako ima propusta zbog odbačenih Ethernet okvira, da li propuste vidi i aplikacija u računaru B? Kao što smo saznali u poglavljiju 3, to zavisi samo od toga da li aplikacija koristi UDP ili TCP. Ako aplikacija koristi UDP, onda će ona u računaru B zaista imati propuste u podacima. S druge strane, ako aplikacija koristi TCP, onda TCP u računaru B neće potvrditi odbačene podatke, što će prouzrokovati da TCP u računaru A ponovo započne prenos nepotvrđenog segmenta. Zapazite da kada TCP ponovo šalje podatke, oni će dolaziti preko Ethernet adaptera u kome su biti odbačeni. Dakle, u tom smislu, Ethernet može ponovo da prenosi podatke. Ali, trebalo bi da imamo na umu da Ethernet nije svestan da ponovo prenosi iste podatke. Ethernet misli da prima potpuno nov datagram sa potpuno novim podacima, čak iako taj datagram sadrži podatke koji su najmanje jednom već bili prenošeni.

### 5.5.2 CSMA/CD: Ethernetov protokol sa višestrukim pristupom

Kada su čvorovi međusobno povezani habom (za razliku od komutatora sloja veze podataka), Ethernet LAN je pravi difuzni LAN - odnosno, kada adapter predaje okvir, primaju ga svi adapteri u LAN-u. Imajući u vidu da Ethernet može da koristi difuziju, potreban mu je protokol sa višestrukim pristupom. Setite se iz odeljka 5.3 da CSMA/CD radi sledeće:

1. Adapter može da počne da predaje u bilo kom trenutku; to znači, ne koriste se nikakvi vremenski odsečci.
2. Adapter nikada ne predaje okvir kada oseti da neki drugi adapter predaje; to znači, on koristi prepoznavanje nosioca.
3. Pre nego što pokuša sa ponovnim prenosom, adapter Čeka tokom nasumičnog vremenskog perioda, koji je obično mali u poređenju sa trajanjem prenosa okvira.

Ovi mehanizmi daju CSMA/CD u okruženju LAN-a mnogo bolju performansu od ALOHE sa odsečcima. U stvari, ako je maksimalno kašnjenje usled propagacije između stanica veoma malo, efikasnost CSMA/CD može da se približi vrednosti od 100 procenata. Ali, zapazite da gore navedeni drugi i treći mehanizam zahtevaju da svaki Ethernet adapter može (1) da oseti kada neki drugi adapter predaje i (2) da otkrije koliziju dok sam prenosi svoje okvire, Ethernet adapteri izvršavaju ove zadatke mereći nivoe napona pre i posle prenosa.

Svaki adapter izvršava protokol CSMA/CD bez eksplisitne koordinacije sa drugim adapterima u Ethernetu. Unutar određenog adaptora, protokol CSMA/CD radi na sledeći način:

1. Adapter dobija datagram mrežnog sloja iz svog nadredenog čvora, priprema Ethernets okvir i stavlja okvir u svoju privremenu memoriju.
2. Ako adapter oseti da je kanal slobadan (odnosno, da nema energije signala koji ulazi u adapter iz kanala u vremenu trajanja 96 bitova), on počinje da prenosi okvir. Ako adapter oseti da je kanal zauzet, on čeka sve dok ne detektuje da nema signala (plus 96 vremena trajanja bita) i tada počinje da prenosi okvir.
3. Dok prenosi, adapter nadgleda da li su prisutni signali koji dolaze od drugih adaptera. Ako adapter prenese ceo okvir bez otkrivanja signala od drugih adaptera; on je završio sa okvirom.
4. Ako adapter otkrije signal od drugih adaptera tokom prenosa, on zaustavlja svoj prenos okvira i umesto njega šalje 48-bitni signal zagušenja.
5. Posle prekida (odnosno slanja signala zastoja), adapter ulazi u fazu eksponencijalnog odstupanja. Posebno, kada prenosi dati okvir, posle n-te kolizije za taj okvir, adapter na slučajan način bira vrednost za  $K$  iz  $\{0, 1, 2, \dots, 2^{m_A}\}$  gde je  $m_A = \min(n, 10)$ . Adapter onda čeka T-512 vremenskih intervala trajanja jednog bita, a zatim se vraća na postupak 2.

Nekoliko komentara o CSMA/CD-u. Svrha signala zastoja je da se osigura da ostali adapteri koji predaju postanu svesni kolizije. Hajde da pogledamo jedan primer. Pretpostavite da adapter A počinje da prenosi okvir i baš pre nego što signal iz A stigne do adaptora B, adapter B počinje da predaje. Tako će B predati samo nekoliko bitova pre nego što zaustavi svoj prenos. Tih nekoliko bitova će se zaista preneti do A, ali oni možda nemaju dovoljno energije da bi A otkrio koliziju. Da bi se obezbedilo da A otkrije koliziju (tako da bi takođe mogao da se zaustavi), B predaje 48-bitni signal zastoja.

Razmotrite sada algoritam eksponencijalnog odstupanja. Prva stvar koju ovde treba zapaziti je daje trajanje bita (odnosno, vreme koje je potrebno da bi se preneo jedan bit) veoma kratko; kod Etherneata od 10 Mb/s, ono iznosi 0,1 mikrosekunde. Pretpostavite da adapter prvi put pokušava da prenese okvir i, dok prenosi, otkriva koliziju. Tada adapter bira  $K = 0$  sa verovatnoćom 0,5 i  $K = 1$  sa verovatnoćom 0,5. Ako adapter izabere  $K = 0$ , on odmah posle slanja signala zastoja prelazi na postupak 2. Ako adapter izabere  $K = 1$ , on čeka 51,2 mikrosekunde posle slanja signala zastoja. Posle druge kolizije, i? se sa podjednakom verovatnoćom bira iz  $\{0, 1, 2, 3\}$ . Posle tri kolizije,  $K$  se sa podjednakom verovatnoćom bira iz  $\{0, 1, 2, 3, 4, 5, 6, 7\}$ . Posle deset ili više kolizija,  $K$  se sa podjednakom verovatnoćom bira iz  $\{0, 1, 2, 1023\}$ . Dakle, veličina skupa iz koga se bira hraste eksponencijalno sa brojem kolizija (do  $n = 10$ ); to je razlog zašto se Ethernetov algoritam odstupanja zove „eksponencijalnim odstupanjem“.

Ethernet standard nameće ograničenja u rastojanju između dva čvora. Ova ograničenja obezbeđuju da, ako adapter A izabere manju vrednost za K od svih drugih adaptera koji su obuhvaćeni kolizijom, tada će on biti sposoban da pošalje svoj okvir bez dozivljavanja nove kolizije. Ovu osobinu ćemo detaljnije istražiti u problemima za domaće zadatke.

Zašto se koristi eksponencijalno odstupanje? Zašto da, na primer, ne izaberemo  $K$  iz  $\{0, 1, 2, 3, 4, 5, 6, 7\}$  posle svake kolizije? Razlog je u tome što, kada adapter doživi svoju prvu koliziju, on nema nikakvu ideju o tome koliko je adaptora obuhvaćeno tom kolizijom. Ako postoji samo mali broj adaptora u koliziji, ima smisla birati  $K$  iz malog skupa malih vrednosti. S druge strane, ako je mnogo adaptora obuhvaćeno kolizijom, ima smisla birati  $K$  iz većeg, šireg skupa vrednosti (zašto?). A povećavanjem veličine skupa posle svake kolizije, adapter se na odgovarajući način prilagođava tim različitim scenarijima.

Ovde takođe zapažamo da svaki put kada adapter priprema novi okvir za prenos, on izvršava algoritam CSMA/CD koji je gore objašnjen. Adapter ne uzima u obzir nikakve kolizije koje su se mogle dogoditi u nedavnoj prošlosti. Tako je moguće da adapter sa novim okvirom bude odmah u stanju da obavi uspešan prenos, dok se više drugih adaptora nalazi u stanju eksponencijalnog odstupanja.

### Efikasnost Etherнетa

Kada samo jedan čvor ima okvir za slanje, on može da vrši prenos punom brzinom Ethernet tehnologije (10 Mb/s, 100 Mb/s ili 1 Gb/s). Međutim, ako mnogo čvorova ima okvire za slanje, efektivna brzina prenosa kanala može da bude mnogo manja. Efikasnost Etherнетa definišemo kao dugoročni deo vremena u kome se okviri prenose po kanalu bez kolizija kada postoji veliki broj aktivnih čvorova, gde svaki čvor ima veliki broj okvira za slanje. Da bismo predstavili aproksimaciju efikasnosti Etherнетa, neka  $t_{pns}$  označava maksimalno vreme koje je potrebno energiji signalu da se prenese između dva adaptora. Neka je  $t_{trans}$  vreme za prenos Ethernet okvira maksimalne veličine (približno 1,2 ms za Ethernet od 10 Mb/s). Izvođenje efikasnosti je izvan domena ove knjige (pročitajte [Lam 1980] i [Bertsekas 1991]). Ovde jednostavno dajemo sledeću aproksimaciju:

$$\text{Efficiency} = \frac{t_{trans}}{1 + 5^{\frac{t_{pns}}{t_{trans}}}}$$

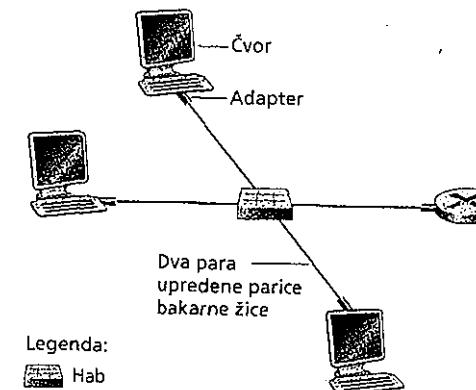
Iz ove formule vidimo da, kako se  $t_{pns}$  približava vrednosti 0, efikasnost se približava vrednosti 1. Ovo odgovara našem intuitivnom zaključku da ako je kašnjenje propagacije nula, čvorovi u koliziji će odmah odustati, bez uzaludnog trošenja vremena na kanalu. Takođe, ako  $t_{pns}$  je veoma veliko, efikasnost se približava vrednosti 1. To je takođe intuitivno, zato što kada okvir zauzme kanal, on će ga držati veoma dugo vreme; dakle, kanal će raditi koristan posao u najvećem delu vremena.

### 5.5.3 Ethernet tehnologije

U 2004. godini, najčešće Ethernet tehnologije su 10BaseT i 100BaseT, koje koriste upredene parice bakarnih provodnika u topologiji zvezde i imaju brzinu prenosa od 10 Mb/s i 100 Mb/s, respectivno. Ove Ethernet tehnologije su standardizovane od strane radnih grupa IEEE 802.3. To je razlog zašto se na Ethernet LAN često poziva kao na 802.3 LAN.

Na slici 5.26 ilustrovane su tehnologije 10BaseT i 100BaseT. Svaki adapter u svakom čvoru ima direktnu vezu od tačke do tačke sa habom. Ova veza se sastoji od dva para upredenih parica bakarnih provodnika, jedan za predaju a drugi za prijem. Na kraju veze je konektor RJ-45 koji liči na konektor RJ-11 za obične telefone. Ono „T“ u 10BaseT i 100BaseT znači „upredena parica“ (*twisted-pair*). 1 za 10BaseT i 100BaseT, maksimalna dužina veze između adaptora i haba je 100 metara; dakle, maksimalno rastojanje između bilo koja dva čvora iznosi 200 metara. Kao što ćemo objasniti u sledećem odeljku, to maksimalno rastojanje može da se poveća korišćenjem niza habova, mostova, komutatora i linkova na bazi optičkih vlakana.

Hab je uređaj fizičkog sloja koji deluje na pojedinačnim bitovima, a ne na okvirima. On ima dva ili više interfejsa. Kada bit, koji predstavlja jedinicu ili nulu, stigne iz jednog interfejsa, hab prosti ponovo napravi bit, pojača njegovu energiju i predi bit na sve druge interfejsse. Važno je imati na umu da habovi ne primenjuju prepoznavanje nosioca niti bilo koji drugi deo CSMA/CD; hab ponavlja dolazni bit na svim odlaznim interfejsima, čak i kad postoji energija signala na nekom od interfejsa. Zato što habovi predaju bitove, svaki adapter u Ethernetu 10/100BaseT može da (1) oseti kanal kako bi odredio da li je slobodan i da (2) otkrije koliziju tokom prenosa.



Slika 5.26 ♦ Topologija zvezde za 10BaseT i 100BaseT

Habovi takođe obezbeđuju svojstva upravljanja mrežom. Na primer, ako neki adapter radi neispravno i stalno šalje Ethernet okvire (takođevani „brbljivi adapter“), tada će mreža 10/100BaseT nastaviti da funkcioniše, zato što će hab otkriti problem i interno isključiti adapter koji je neispravan. Uz ovo svojstvo, administrator mreže ne mora da ustaje iz kreveta i vraća se natrag na posao da bi resio problem. Takođe, većina habova može da prikuplja informacije i dostavlja ih računaru koji je povezan direktno sa habom. Kao što se govori u poglavljiju 9, ovaj računar za nadzor obezbeđuje grafički interfejs koji prikazuje statističke podatke i grafikone, kao što su iskorišćenje propusnog opsega, učestanost kolizija, prosečne veličine okvira itd. Administratori mreže mogu da koriste ove informacije, ne samo da bi otkrivali i rešavali probleme, nego i da bi planirali kako bi LAN trebalo da se razvija u budućnosti.

Mnogi današnji Ethernet adapteri su za 10/100 Mb/s. To znači da oni mogu da se koriste i za Ethernet IOBaseT i za Ethernet I/OBaseT. I/OBaseT obično koristi upredene parice kategorije 5 (visokokvalitetan par upredenih žica sa mnogo zavoja). Za razliku od IOBase2 i IOBaseT, I/OBaseT ne koristi Manchester kodovanje, nego umesto njega mnogo efikasnije kodovanje koje se zove 4B5B: svaka grupa od pet perioda generatora takta koristi se za slanje četiri bita da bi bilo dovoljno promena logičkih nivoa, kako bi se obezbedila sinhronizacija generatora takta.

Na ovom mestu ukratko pominjemo da obe Ethernet tehnologije, 10 Mb/s i 100 Mb/s, mogu da koriste linkove od optičkih vlakana. Link od optičkih vlakana često se koristi da poveže habove koji se nalaze u različitim zgradama u okviru istog naselja. Optička vlakna su skupa, zbog cene njihovih konektora, ali imaju odličnu otpornost na šumove. Standardi IEEE 802 dozvoljavaju da LAN ima veći geografski domet kada se optička vlakna koriste pri povezivanju čvorova na okosnici.

#### Gigabit Ethernet i Ethernet 10 Gb/s

Gigabit Ethernet je proširenje veoma uspešnih standarda Ethernet 10 Mb/s i 100 Mb/s. Nudеći brzinu podataka od 1000 Mb/s, Gigabit Ethernet održava punu kompatibilnost sa ogromnom bazom instalirane opreme za Ethernet. Standard za Gigabit Ethernet, koji se zove IEEE 802.3z, radi sledeće:

- ◆ Koristi standardni format Ethernet okvira (slika 5.24) i unazad je kompatibilan sa tehnologijama IOBaseT i I/OBaseT. To dopušta laku integraciju Gigabit Ether-neta sa postojećom instaliranom bazom Ethernet opreme.
- ◆ Dopušta linkove od tačke do tačke, kao i deljene difuzne kanale. Linkovi od tačke do tačke koriste komutatore (pročitajte odeljak 5.6), dok difuzni kanali koriste habove, kao sto je ranije opisano za IOBaseT i I/OBaseT. U žargonu Gigabit Etherneta, habovi se zovu „distributeri sa privremenom memorijom“.

- ◆ Koristi CSMA/CD za deljene difuzne kanale. Da bi imao prihvatljuvu efikasnost, maksimalno rastojanje između čvorova mora da bude strogo ograničeno.
- ◆ Dopušta rad u punom dupleksu na 1000 Mb/s u oba pravca za kanale od tačke do tačke.

Kao i IOBaseT i I/OBaseT, Gigabit Ethernet ima topologiju zvezde, sa habom ili komutatorom u centru. (Komutatori za Ethernet će biti objašnjeni u odeljku 5.6.) Gigabit Ethernet često služi kao okosnica za međusobno povezivanje više Ethernet LAN-ova 10 Mb/s i 100 Mb/s. U početku radeći preko optičkih vlakana, Gigabit Ethernet je sada u stanju da radi preko kablova UTP kategorije 5.

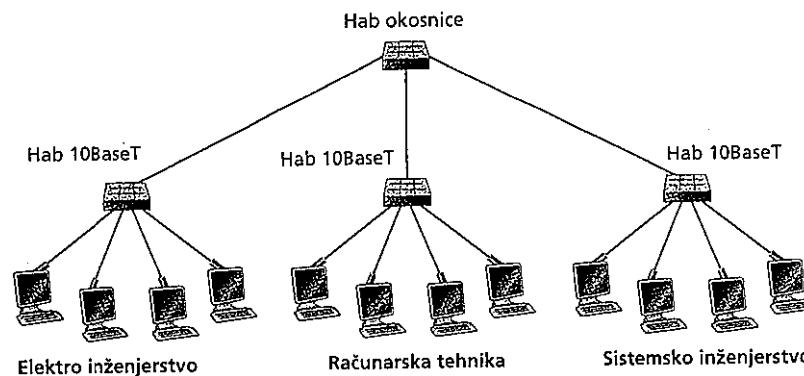
Sa proizvodima koji su se pojavili 2001. godine, 10 Gigabit Ethernet dalje proširuje popularnu Ethernet tehnologiju. Pored toga, standard 10 Gigabit Ethernet, 802.3ae, proširuje Ethernet tehnologiju na linkove od tačke do tačke regionalnih računarskih mreža (*wide area network, WAN*). Mnogo korisnih informacija i linkova o Gigabit i 10 Gigabit Ethernetu, možete naći na Spurgeonovoj veb lokaciji o Ethernetu [Spurgeon 2004].

## 5.6 Međusobne veze: habovi i komutatori

Institucije - kompanije, univerziteti i visoke škole - obično se sastoje od mnogo odeljenja, gde svako odeljenje ima sopstveni Ethernet LAN i upravlja njime. Naravno, institucija želi da njena odeljenja međusobno povezu svoje odeljenjske LAN segmente. U ovom odeljku ćemo razmotriti tri različita pristupa povezivanju LAN-ova: habove, mostove i komutatore. Danas je upotreba sva tri pristupa široko rasprostranjena.

### 5.6.1 Habovi

Najjednostavniji način da se međusobno povezu LAN-ovi jeste da se koriste habovi. Na slici 5.27 prikazana su tri akademска odeljenja na univerzitetu koja hoće da povezu svoje LAN-ove. Svako od tri odeljenja na slici ima Ethernet IOBaseT koji obezbeđuje mrežni pristup fakultetu, nastavnom osoblju i studentima odeljenja. Svaki računar u odeljenju ima vezu od tačke do tačke sa odeljenjskim habom. Četvrti hab, koji se zove hab okosnice, ima veze od tačke do tačke sa odeljenjskim labovima, međusobno povezujući LAN-ove tri odeljenja. Konstrukcija prikazana na slici 5.27 je konstrukcija haba sa više nivoa (*multi-tier hub design*), zato što su habovi raspoređeni u hijerarhiji. Mogu se takođe napraviti konstrukcije sa više od dva nivoa - na primer, jedan nivo za odeljenja, jedan nivo za škole unutar univerziteta (na primer, inženjerska škola, poslovna škola itd.) i jedan nivo za najviši univerzitetski nivo.



**Slika 5.27** ♦ Tri odeljenjska Etherneta međusobno povezana pomoću haba

U konstrukciji sa više nivoa, celu međusobno povezanu mrežu zovemo LAN, a svaki od odeljenjskih delova LAN-a (odnosno, odeljenjski hab i računare koji su sa njim povezani) zovemo **segmentom LAN-a**. Važno je zapaziti da svi segmenti LAN-a na slici 5.27 pripadaju istom **domenu kolizije**; znači, kad god dva ili više čvorova u segmentu LAN-a predaju u isto vreme, doći će do kolizija i svi čvorovi koji predaju učiće eksponencijalno odstupanje.

Medusobno povezivanje odeljenjskih LAN-ova pomoću haba okosnice ima mnogih prednosti. Prvo i pre svega, ono obezbeđuje međuodeljenjsku komunikaciju između računara u različitim odeljenjima. Drugo, ono povećava maksimalno rastojanje između bilo kog para čvorova u LAN-u. Na primer, kod 10BaseT, maksimalno rastojanje između čvora i njegovog haba je 100 metara; dakle, u jednom segmentu LAN-a, maksimalno rastojanje između bilo kog para čvorova je 200 metara. Međusobnim povezivanjem habova, to maksimalno rastojanje može da se poveća, zato što rastojanje između direktno povezanih habova može takođe da bude 100 metara kada se koriste upredene parice (a i više ako se koriste optička vlakna). Treća korist je što konstrukcija sa više nivoa obezbeđuje izvestan stepen tolerancije pri mogućem otkazu. Preciznije, ako bilo koji od odeljenjskih habova počne da radi neispravno, hab okosnice može da otkrije problem i isključi neispravan hab sa LAN-a; na taj način, preostala odeljenja mogu da nastave sa radom i komuniciraju dok se neispravan odeljenjski hab ne popravi.

Mada je hab okosnice koristan uređaj za međusobno povezivanje, on ima tri ozbiljna ograničenja koja otežavaju njegovu primenu. Prvo i možda najvažnije, kada su odeljenjski LAN-ovi međusobno povezani pomoću haba (ili repetitora), onda se nezavisni domeni kolizije odeljenja pretvaraju u jedan veliki, zajednički domen kolizije. Istražimo to pitanje u kontekstu slike 5.27. Pre međusobnog povezivanja tri

odeljenja, svaki odeljenjski LAN imao je maksimalnu propusnu moć od 10 Mb/s, pa je maksimalna združena propusna moć tri LAN-a bila 30 Mb/s. Ali, jednom kada su tri LAN-a međusobno povezana pomoću haba, svi računari u tri odeljenja pripadaju istom domenu kolizije, a maksimalna združena propusna moć je smanjena na 10 Mb/s.

Druge ograničenje je to što, ako različita odeljenja koriste različite Ethernet tehnologije, može biti nemoguće da se međusobno povezu odeljenjski habovi sa habom okosnice. Na primer, ako neka odeljenja koriste 10BaseT, a ostala odeljenja 100BaseT, onda je nemoguće povezati sva odeljenja bez stavljanja okvira u privremenu memoriju u tački međusobnog povezivanja; kako su habovi u suštini repetitori i ne stavljuju okvire u privremenu memoriju, oni ne mogu da međusobno povezuju segmente LAN-a koji rade na različitim brzinama.

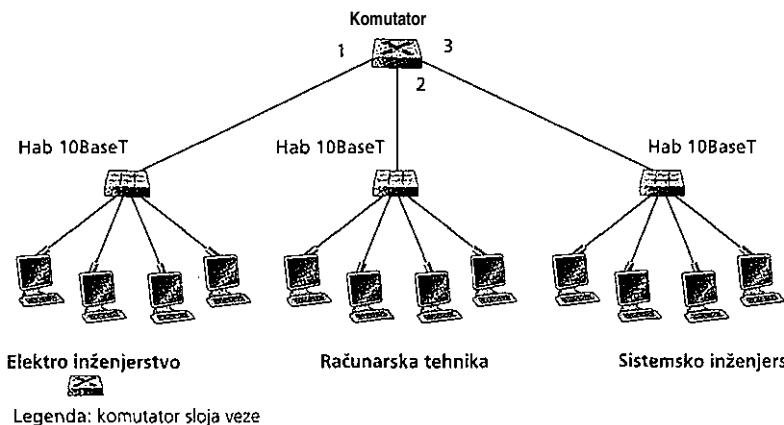
Treće ograničenje je to što svaka Ethernet tehnologija (10Base2, 10BaseT, 100BaseT itd.) ima ograničenje maksimalnog dozvoljivog broja čvorova u domenu kolizije, maksimalnog rastojanja između dva računara u domenu kolizije i maksimalnog dozvoljivog broja nivoa u LAN-u sa više nivoa, kao i geografskog dometa LAN-a sa više nivoa.

### 5.6.2 Komutatori sloja veze podataka

Za razliku od habova koji su uređaji fizičkog nivoa, komutatori sloja veze podataka - zvani prosto **komutatori** - rade na Ethernet okvirima i zato su uređaji sloja-2. U stvari, kao potpuno osposobljeni komutatori paketa, oni prosledjuju okvire koristeći LAN adresu odredišta. Kada okvir dode na interfejs komutatora, komutator ispituje adresu odredišta sloja-2 okvira i pokušava da prosledi okvir na interfejs koji vodi ka odredištu.

Na slici 5.28 prikazano je kako tri akademска odeljenja iz našeg prethodnog primera mogu da se povezu pomoću komutatora. Tri broja do komutatora su brojevi komutatorskih interfejsa. Kada se odeljenja međusobno povezu pomoću komutatora, kao na slici 5.28, opet celu međusobno povezанu mrežu zovemo LAN i opet svaki od odeljenjskih delova mreže zovemo segmentom LAN-a. Ali, za razliku od konstrukcije haba sa više nivoa na slici 5.27, svaki segment LAN-a je sada izolovani domen kolizije.

Komutatori mogu da prevaziđu mnoge probleme koji postoje kod habova. Prvo, komutatori dozvoljavaju međuodeljenjsku komunikaciju, dok istovremeno održavaju izolovane domene kolizija za svako odeljenje. Drugo, komutatori mogu međusobno da povezuju različite LAN tehnologije, uključujući tu Ethernet od 10 Mb/s, 100 Mb/s i Gigabit Ethernet. Treće, kada se za međusobno povezivanje segmenata LAN-a koriste komutatori, nema ograničenja koliki LAN može da bude; u teoriji, ako se koriste komutatori, moguće je izgraditi LAN koji se prostire preko cele zemaljske kugle. Dakle, kao što ćemo diskutovati na kraju ovog odeljka, komutatori rade u punom dupleksu i obezbeđuju komutaciju prečicom (*cut-through*).

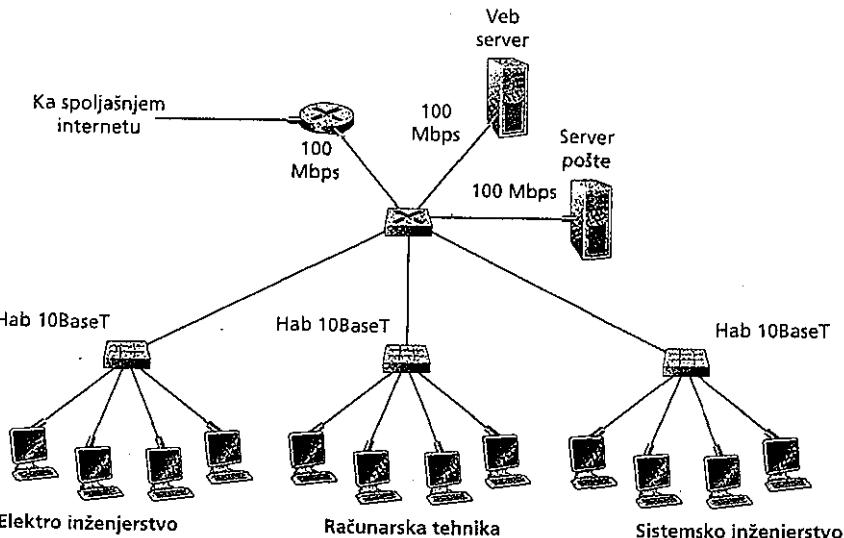


**Slika 5.28** ♦ Tri odeljenjska LAN-a međusobno povezana pomoću komutatora

Na slici 5.29 prikazano je kako institucija sa više odeljaka i više kritičnih servera može da razmesti kombinaciju habova, komutatora i ruteru. Na slici 5.29 svako od tri odeljenja ima svoj sopstveni Ethernet segment od 10 Mb/s, sa svojim sopstvenim habom. Zato što svaki<sup>1</sup> odeljenjski hab ima vezu sa komutatorom, celokupan međodeljenjski saobraćaj je ograničen na Ethernet segment odeljenja. I web i poštanski serveri imaju namenski pristup komutatoru od 100 Mb/s. Najzad, ruter koji vodi na Internet, ima pristup komutatoru od 100 Mb/s. Zapazite da ovaj komutator ima najmanje tri interfejsa od 10 Mb/s i tri interfejsa od 100 Mb/s.

#### Prosleđivanje i filtriranje komutatora

**Filtriranje** je sposobnost komutatora da odredi da li bi okvir trebalo da se prosledi nekom njegovom interfejsu, ili samo da se odbaci. **Prosleđivanje** je sposobnost da se odrede interfejsi ka kojima bi okvir trebalo da se usmeri, a onda da se okvir i usmeri ka tim interfejsima. **Filtriranje** i **prosleđivanje** komutatora se rade pomoću **tabele komutatora**. Tabela komutatora sadrži stavke za neke, ali ne obavezno i za sve čvorove u LAN-u. Stavka u tabeli komutatora sadrži (1) MAC adresu čvora, (2) interfejs komutatora koji vodi do čvora i (3) vreme kadaje stavku za čvor upisana u tabelu. Primer tabele komutatora za LAN sa slike 5.28 prikazan je na slici 5.30. Iako **ovaj** opis prosleđivanja okvira može da zvuči slično objašnjenju prosleđivanja datagrama dalom u poglavljiju 4, ubrzo ćemo videti da postoje značajne razlike. Ovde zapažamo da su adrese koje koriste komutatori MAC adrese, a ne adrese mrežnog



**Slika 5.29** ♦ Institucionalna mreža koja koristi kombinaciju habova, Ethernet komutatora i ruteru.

slaja. Uskoro ćemo takođe videti da se tabela komutatora konstruiše na različit način od tabele rutiranja.

Da bismo razumeli kako rade filtriranje i prosleđivanje komutatora, prepostavimo da okvir sa adresom odredišta DD-DD-DD-DD-DD-DD stiže na komutator na interfejsu x. Komutator indeksira svoju tabelu sa MAC adresom

| Adresa            | Interfejs | Vreme |
|-------------------|-----------|-------|
| 62-FE-F7-11-89-A3 | 1         | 9:32  |
| 7C-BA-B2-B4-91-10 | 3         | 9:36  |
| ....              | ....      | ....  |

**Slika 5.30** ♦ Deo tabele mosta za LAN na slici 5.28

DD-DD-DD-DD-DD-DD i pronalazi svoj odgovarajući interfejs y za koji se zna da vodi do adrese odredišta DD-DD-DD-DD-DD-DD. (Uskoro ćemo videti Šta se dešava ako adresa DD-DD-DD-DD-DD-DD nije u tabeli.)

- ◆ Ako je x jednako y, onda okvir dolazi iz segmenta LAN-a koji sadrži adapter DD-DD-DD-DD-DD-DD. Nema potrebe da se okvir prosleduje na bilo koji drugi interfejs, komutator obavlja funkciju filtriranja odbacujući okvir.
- ◆ Ako x nije jednako y, onda okvir treba da se prosledi segmentu LAN-a priključenom na interfejs y. Komutator obavlja svoju funkciju prosleđivanja stavljajući okviru izlaznu privremenu memoriju koja prethodi interfejsu y.

Ova jednostavna pravila dopuštaju komutatoru da održi razdvojene domene kolizija za svaki od različitih segmenata LAN-a koji su priključeni na njegove interfejsе. Pravila takođe dozvoljavaju da dva skupa čvorova na različitim segmentima LAN-a komuniciraju istovremeno, bez međusobnog ometanja.

Hajde da prođemo kroz ova pravila za mrežu na slici 5.28 i njenu tabelu komutatora na slici 5.30. Pretpostavite da okvir sa adresom odredišta 62-FE-F7-11-89-A3 stiže u komutator sa interfejsa 1. Komutator ispituje svoju tabelu i vidi da je odredište na LAN segmentu povezanim sa interfejsom 1 (odnosno LAN-om Elektro inž-e-njerstva). To znači daje okvir već bio difuzno poslat u segment LAN-a koji sadrži odredište, Komutator zato filtrira (odnosno, odbacuje) okvir. Sada pretpostavite da okvir sa istom adresom odredišta stiže sa interfejsa 2. Komutator ponovo ispituje svoju tabelu i vidi daje odredište u pravcu interfejsa 1; on, dakle, prosleduje okvir na izlaznu privremenu memoriju koja prethodi interfejsu 1. Iz ovog primera trebalo bi da bude jasno da, dok je tabela komutatora potpuna i tačna, komutator izoluje podejenske domene kolizija, dok istovremeno dopušta odeljenjima da komuniciraju.

#### Habovi u odnosu na komutatore

Setite se da kada hab prosleduje okvir u link, on samo šalje bitove na link, ne trudeći se da oseti da li se na njemu trenutno odvija neki drugi prenos. Za razliku od toga, kada komutator želi da prosledi okvir na link, on pokreće algoritam CSMA/CD koji je objašnjen u odeljku 5.3. Posebno, komutator odustaje od prenosa ako oseti da je u toku prenos sa nekog drugog čvora u segmentu LAN-a u koji on želi da pošalje okvir; pored toga, komutator koristi eksponencijalno odstupanje kada jedan od njegovih prenosa za rezultat ima koliziju. Na taj način, komutatori se u velikoj meri ponašaju kao adapteri čvorova. Ali, govoreći tehnički, oni nisu adapteri čvorova, zato što ni komutator ni njegovi interfejsi nemaju MAC adresu. Setite se da adapter Čvora uvek umeće svoju MAC adresu u adresu izvora svakog okvira koji prenosi. S druge strane, komutator ne menja izvornu adresu okvira.

Značajna karakteristika komutatora je da mogu da se upotrebe za kombinovanje Ethernet segmenata koji koriste različite Ethernet tehnologije. Na primer, ako

na slici 5.28 Elektro inženjerstvo ima Ethernet 10Base2, Računarska tehnika Ethernet IOBaseT, a Sistemsko inženjerstvo Ethernet IOBaseT, onda može da se kupi komutator koji može da poveže ta tri LAN-a. Pomoću komutatora Gigabit Ethernet, moguće je imati dodatnu vezu od 1 Gb/s sa ruterom, koji je sa svoje strane povezan sa širom univerzitetskom mrežom. Kao što smo ranije pomenuh, ova osobina da se međusobno povezuju različite brzine linkova nije raspoloživa kod habova.

Takođe, kada se kao uredaji za međusobno povezivanje koriste komutatori, nema teorijskog ograničenja za geografski obim LAN-a. U teoriji, možemo da izgradimo LAN koji se prostire na celoj zemaljskoj kugli, međusobno povezujući habove u dugoj, linearnoj topologiji, gde je svaki par susednih habova međusobno povezan pomoću komutatora. Kod ovakve konstrukcije, svaki hab ima sopstveni domen kolizije, a nema ograničenja koliko LAN može da bude dugačak. Međutim, uskoro ćemo videti da nije poželjno graditi veoma velike mreže koristeći isključivo komutatore kao uredaje za međusobno povezivanje - velikim mrežama su takođe potrebni ruteri.

#### Samoobučavanje

Komutator ima izvanrednu osobinu (posebno za već prezaposlenog administratora mreže) da se njegova tabela pravi automatski, dinamički i autonomno - bez ikakve intervencije od strane administratora mreže ili protokola konfiguracije. Drugim recima, komutatori se sami obučavaju. Ova sposobnost se ostvaruje na sledeći način:

1. Tabela komutatora je u početku prazna.
2. Kada okvir stigne na jedan od interfejsa, a adresa odredišta okvira nije u tabeli, komutator prosleduje kopije okvira u privremenu memoriju koje prethode svim drugim interfejsima. (Na svakom od ovih drugih interfejsa, okvir se prenosi u segment LAN-a koristeći CSMA/CD).
3. Za svaki primljeni okvir, komutatoru svojoj tabeli skladišti (1) MAC adresu u *polju za adresu izvora okvira*, (2) interfejs sa koga je okvir stigao, (3) tekuće vreme. Na taj način, komutator zapisuje u svojoj tabeli segment LAN-a u kome se nalazi čvor pošiljalac. Ako svaki čvor u LAN-u na kraju pošalje okvir, onda će svaki čvor na kraju biti zapisan u tabeli.
4. Kada čvor stigne na jedan od interfejsa a adresa odredišta okvira je u tabeli, onda komutator prosleduje okvir na odgovarajući interfejs.
5. Komutator briše adresu u tabeli ako se posle određenog vremenskog perioda (vreme stareњa, *aging time*) ne primi nijedan okvir sa tom adresom kao adresom izvora. Na taj način, ako se PC zameni drugim PC-jem (sa različitim adapterom), LAN adresa prvobitnog PC-ja će na kraju biti uklonjena iz tabele komutatora.

Hajde da prođemo kroz osobinu samoobučavanja za mrežu na slici 5.28 i njenu odgovarajuću tabelu komutatora na slici 5.30. Pretpostavite da u vreme 9:39 okvir

| Adresa            | Interfejs | Vreme |
|-------------------|-----------|-------|
| 01-12-23-34-45-56 | 2         | 9:39  |
| 62-FE-F7-11-89-A3 | 1         | 9:32  |
| 7C-BA-B2-B4-91-10 | 3         | 9:36  |
| ....              | ....      | ....  |

**Slika 5.31** ♦ Komutator saznaće o lokaciji adaptora sa adresom 01-12-23-34-45-56

sa adresom izvora 01-12-23-34-45-56 stiže sa interfejsa 2. Prepostavite da la adresa nije u tabeli komutatora. Tada komutator dodaje novu stavku u tabeli, kao što je prikazano na slici 5.31.

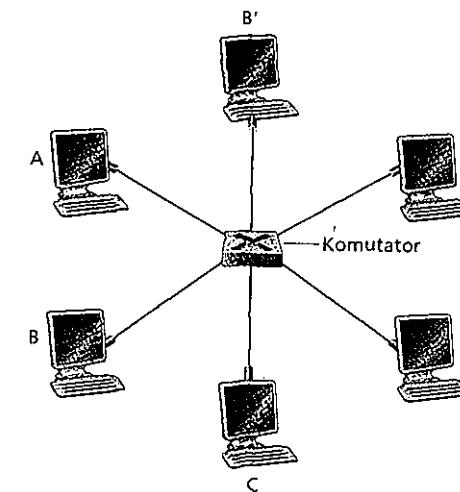
Nastavljajući sa ovim istim primerom, prepostavimo daje vreme zasta-revanja za ovaj komutator 60 minuta i da niješ okvir sa izvornom adresom 62-FE-F7-11-89-A3 ne stiže u komutator između 9:32 i 10:32. Onda u trenutku 10:32 komutator briše tu adresu iz svoje tabele.

Komutatori su *plug-and-play* uređaji zato što ne zahtevaju nikakvu intervenciju od strane administratora mreže ili korisnika. Administrator mreže koji želi da instalira komutator ne radi ništa više od priključivanja segmenata LAN-a na interfejs komutatora. Administrator mreže ne treba da konfiguriše tabele komutatora u vreme instalacije ili kada se računar ukloni iz jednog od segmenata LAN-a.

#### Namenski pristup i puni dupleks

Komutator sa velikim brojem interfejsa omogućava direktnе konekcije između računara i komutatora. Kada računar ima direktnu konekciju sa komutatorom (a ne deljenu LAN konekciju), kaže se da ima namenski pristup. Na slici 5.32, komutator omogućava namenski pristup za šest računara.

Komutator i sa njim direktno povezani računari rade u režimu punog dupleksa. Hajde da vidimo kako se to radi. Da bismo ovu diskusiju konkretizovali, pretpo-siavimo da svaka konekcija na slici 5.32 korisli dva paTa upredenih parica bakarnih provodnika (kao u 10BaseT i 100BaseT), jedan par za predaju od računara do komutatora, a drugi par za predaju od komutatora do računara. Zbog namenskog pristupa, kada računar A predaje okvir na svom uzvodnom paru provodnika, nema mogućnosti da će okvir doći u koliziju sa predajom iz nekog drugog računara ili iz komutatora. Slično tome, zato što komutatori "memorišu i prosledjuju", komutator će predavati najviše jedan okvir istovremeno na bilo kome od nizvodnih parova provodnika. Dakle, sa direktnim uzvodnim i nizvodnim konekcijama, nisu potrebni ni detekcija kolizija ni prepoznavanje nosioca. U stvari, svaki link postaje link od tačke do tačke, čime se izbegava potreba za bilo kakvim protokolom za pristup medijumu! Dakle, onemogućavanje u svakom adapteru prepoznavanja nosioca, detekcije kolizija i vraćanja predavanih podataka na ulaz prijemnika, stvara se kanal punog dupleksa.



**Slika 5.32** ♦ Komutator obezbeđuje namenski Ethernet pristup za šest računara

ksa između svakog računara i komutatora. Na primer, na slici 5.32, računar A može da pošalje datoteku u A' dok B šalje datoteku u B', a C šalje datoteku u C'. Ako svaki računar ima adaptersku karticu za 10 Mb/s, onda će ukupna propusna moć za vreme tri simultana prenosa datoteka biti 30 Mb/s. Ako A i A' imaju adaptore od 100 Mb/s, a preostali računari imaju adaptore od 10 Mb/s, onda će ukupna propusna moć za vreme tri simultana prenosa datoteka biti 120 Mb/s.

#### Komutacija prečicom

Pored velikog broja interfejsa, podrške za mnoštvo vrsta fizičkih medijuma, brzina prenosa i privlačnih svojstava upravljanja mrežama, proizvođači komutatora često mame kupce izjavavama da njihovi komutatori koriste komutaciju prečicom (*cut-through*), a ne komutaciju paketa sa memorisanjem i prosleđivanjem, koju koriste ruteri i komutatori sloja veze. Razlika između memorisanja i prosleđivanja i komutacije prečicom je u nijansi. Da biste razumeli tu razliku, posmatrajte paket koji se prosleđuje kroz komutaciju paketa (odnosno ruter ili komutator). Kao stoje objašnjeno u odeljku 4.4, paket stiže u komutator preko *ulaznog porta*, i napušta komutator preko *izlaznog porta*. Na izlaznom portu može da bude ili da ne bude drugih paketa koji čekaju u izlaznoj privremenoj memoriji izlaznog porta. Kada ima paketa u izlaznoj privremenoj memoriji, nema apsolutno nikakve razlike između komutacije sa memorisanjem i prosleđivanjem i komutacije prečicom. Dve tehnike komutacije razlikuju se samo kada je izlazna privremena memorija prazna.

Setite se iz poglavlja 1 da, kada se paket prosleduje kroz komutator paketa sa memorisanjem i prosleđivanjem, on se prvo prikupi i memorise u celosti, pre nego što komutator počne da ga predaje na izlaznom linku. U slučaju kada se izlazna privremena memorija isprazni pre nego što je ceo paket stigao u komutator, ovo prikupljanje stvara kašnjenje memorisanja i prosleđivanja u komutatoru - kašnjenje koje doprinosi ukupnom kašnjenju od jednog do drugog kraja (procitajte odeljak 1.6). Gornja granica tog kašnjenja je  $L/R$ , gde je  $L$  dužina paketa, a  $R$  brzina prenosa *ulaznog* linka. Zapazite da paketu preti kašnjenje memorisanja i prosleđivanja samo ako se izlazna privremena memorija isprazni pre nego što ceo paket stigne u komutator.

Kod komutacije prećicom, ako se privremena memorija isprazni pre nego što ceo paket stigne, komutator može da počne da predaje prednji deo paketa dok njegov zadnji deo nastavlja da dolazi. Naravno, pre predaje paketa na izlazni link, mora prvo da stigne deo paketa koji sadrži adresu odredišta. (Ovo malo kašnjenje je neizbežno za sve vrste komutacije, zato što komutator mora da odredi odgovarajući izlazni link.) Sve u svemu, sa komutacijom prećicom paket ne mora da bude potpuno memorisan pre nego što se prosledi; umesto toga, paket se prosleđuje kroz komutator kada je izlazni link sloboden. Ako je izlazni link mreža sa višestrukim pristupom koja se deli sa drugim računarima (na primer, izlazni link se povezuje sa habom), onda komutator takođe mora da oseti da li je link sloboden pre nego što „direktno prosledi“ paket.

Da bismo dobili izvestan uvid u razliku između komutacije sa memorisanjem i prosleđivanjem i komutacije prećicom, hajde da se podsetimo na analogiju sa karavanom, koja je uvedena u odeljku 1.6. U toj analogiji, postoji autoput sa povremenim naplatnim kioscima, gde u svakom kiosku postoji po jedan službenik. Na autoputu se nalazi karavan od deset vozila koja putuju zajedno, svako istom konstantnom brzinom. Vozila u karavanu su jedina vozila na autoputu. Svaki naplatni kiosk opslužuje vozila konstantnom brzinom, tako da kada vozila napuste naplatni kiosk, ona su na podjednakom međusobnom rastojanju. Kao i ranije, možemo da razmišljamo o karavanu kao daje paket, o svakom vozilu u karavanu kao daje bit, a o brzini naplatnog kioska kao o brzini prenosa linka. Sada razmotrite šta rade vozila iz karavana kada stignu do naplatnog kioska. Ako svako vozilo nastavlja direktno do naplatnog kioska sve do pristizanja, onda je naplatni kiosk-prećicu. Ako, s druge strane, svako vozilo čeka na ulasku dok sva ostala vozila karavana ne stignu, onda je naplatni kiosk sa memorisanjem i prosleđivanjem. Jasno je da naplatni kiosk sa memorisanjem i prosleđivanjem zadržava karavan više od naplatnog kioska-prećice.

Komutator prećicom može da smanji kašnjenje paketa od kraja do kraja, ali za koliko? Kao što smo ranije pomenuli, maksimalno kašnjenje memorisanja i prosleđivanja je  $L/R$ , gde je  $L$  veličina paketa, a  $R$  brzina ulaznog linka. Maksimalno kašnjenje je približno 1,2-ms za Ethernet od 10 Mb/s i 0,12 ms za Ethernet od 100 Mb/s (odgovarajuće za Ethernet paket maksimalne veličine). Dakle, komutator prećicom smanjuje kašnjenje za samo 0,12 do 1,2 ms, a do tog smanjenja dolazi samo kadaje izlazni link malo opterećen. Koliko je značajno to kašnjenje? Verovatno ne mnogo u praktičnim primenama.

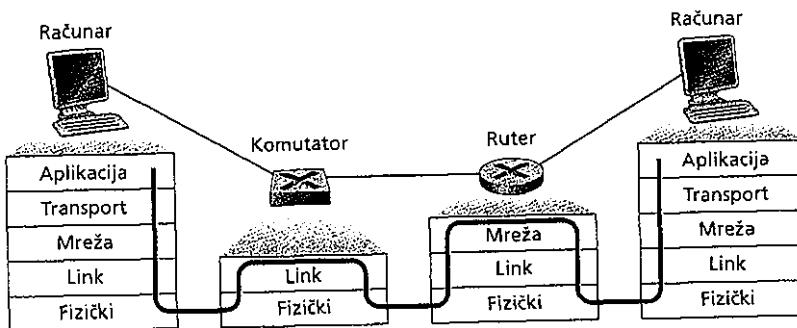
#### Komutatori u odnosu na rutere

Kao što smo naučili u poglavlju 4, ruteri su komutatori za memorisanje i prosleđivanje paketa, koji svoj posao obavljaju koristeći adrese mrežnog sloja. Mada je komutator takođe komutator za memorisanje i prosleđivanje paketa, on se suštinski razlikuje od rutera u tome što prosleđuje pakete koristeći MAC adrese. Dok je ruter komutator paketa sloja-3, komutator komutira pakete na sloju-2.

Čak iako su komutatori i ruteri suštinski različiti, administratori mreže često moraju da biraju koji od njih će da upotrebe kada instaliraju uređaj za međusobno povezivanje. Na primer, za mrežu na slici 5.28, administrator mreže bi mogao lako da upotrebti ruter umesto komutatora. Zaista, ruter bi takođe držao tri domena kolizija razdvojeno, dok bi dozvoljavao komunikaciju između odeljenja. Imajući u vidu da su i komutatori i ruteri kandidati za uređaje međusobnog povezivanja, koji su razlozi za i protiv ova dva pristupa?

Razmotrimo prvo razloge za i protiv u raspravi o primeni komutatora. Kao stoje ranije pomenuto, komutatori su plug-and-play. Što je osobina koju cene svi prezaposleni administratori mreže na svetu. Komutatori takođe imaju relativno velike brzine filtriranja i prosleđivanja paketa - kao što je prikazano na slici 5.33, komutatori treba samo da obrade pakete naviše ka sloju 2, dok ruteri moraju da obraduju okvire naviše ka sloju 3. S druge strane, topologija komutirane mreže je ograničena na obuhvatno stablo. Takođe, velika komutirana mreža bi zahtevala velike ARP tabele u Čvorovima i generisala bi značajan ARP saobraćaj i obradu. Pored toga, komutatori ne nude nikakvu zaštitu protiv difuznih oluja - ako jedan od računara poludi i počne da predaje beskrajn niz difuznih Ethemet okvira, komutatori će prosleđivati sve te okvire, što dovodi do pada Čitave mreže.

Razmotrimo sada razloge za i protiv u raspravi o primeni rutera. Zato stoje mrežno adresiranje često hijerarhijsko (a ne linearno kao MAC adresiranje), paketi obično ne kruže kroz rutere. Čak i kada mreža ima redundantne putanje. (Međutim, paketi mogu da kruže kada su tabele rutera pogrešno konfigurisane; ali, kao što smo naučili u poglavlju 4, IP koristi specijalno polje zaglavljia datagrama da ograniči kruženje.) Dakle, paketi nisu ograničeni na obuhvatno stablo i mogu da koriste najbolju putanju između izvora i odredišta. S obzirom na to da ruteri nemaju ograničenje obuhvatnog stabla, oni su dozvolili da se Internet izgradi sa bogatom topologijom koja uključuje, na primer, višestruke aktivne linkove između Evrope i Severne Amerike. Drugo svojstvo rutera je da oni obezbeđuju zaštitu pomoću mrežne barijere od difuznih oluja sloja-2. Međutim, možda je najveći nedostatak rutera što nisu plug-and-play, pa oni i računari koji su na njih priključeni zahtevaju da se konfigurišu IP adrese. Takođe, ruteri često imaju da obave obimniji posao po paketu od komutatora, zato što moraju da obraduju i pakete u sloju-3.



**Slika 5.33** ♦ Obrada paketa u komutatorima, ruterima i računarima

Imajući u vidu da i komutatori i ruteri imaju svoje razloge za i protiv, kada bi jedna institucionalna mreža (na primer, mreža univerzitetskog naselja ili korporativna mreža) trebalo da koristi komutatore, a kada ruter? Po pravilu, male mreže koje se sastoje od nekoliko stotina računara, imaju nekoliko segmenata LAN-a. Komutatori su dovoljni za ove male mreže, zato što oni lokalizuju saobraćaj i povećavaju združenu propusnu moć bez zahtevanja bilo kakvog konfigurisanja IP adresa. Ali uobičajeno je da veće mreže, koje se sastoje od hiljada računara, uključuju ruteru unutar mreže (pored komutatora). Ruteri obezbeđuju robustniju izolaciju saobraćaja, kontrolišu difuzne oluje i koriste „inteligentnije“ putanje između računara u mreži.

U ovom odeljku naučili smo da habovi, komutatori i ruteri mogu da se koriste kao uređaji za međusobno povezivanje računara i segmenata LAN-a. U tabeli 5.1 zbirno su date karakteristike svakog od ovih uređaja za međusobno povezivanje. Veb lokacija kompanije Cisco daje veliki broj poređenja različitih tehnologija za međusobno povezivanje [Cisco Switches 2004],

|                      | Habovi | Ruteri | Ethernet komutatori |
|----------------------|--------|--------|---------------------|
| Izolacija saobraćaja | ne     | da     | da                  |
| Plug and play        | da     | ne     | da                  |
| Optimalno rutiranje  | ne     | da     | ne                  |
| Cut-through          | da     | ne     | da                  |

**Tabela 5.1** ♦ Poređenje tipičnih svojstava popularnih uređaja za međusobno povezivanje

## 5.7 PPP: protokol od tačke to tačke

Najveći deo naše diskusije o protokolima veze podataka do sada je bio usredosreden na protokole za difuzne kanale. U ovom odeljku objasnićemo protokol veze podataka za linkove od tačke do tačke - PPP, protokol od tačke do tačke (*point-to-point protocol*). S obzirom na to da je PPP uobičajen protokol koji se koristi za povezivanje kućnih računara i ISP-ja, on je van svake sumnje jedan od danas najšire prime-njivanih protokola veze podataka. Drugi značajan protokol u današnjoj upotrebi je HDLC (*high-level data link control*); radi objašnjenja HDLC-a, pročitajte [Spragins 1991]. Naše objašnjenje jednostavnijeg protokola PPP će nam omogućiti da istražimo mnoge značajne karakteristike protokola veze podataka od tačke do tačke.

Kao što mu i samo ime govori, protokol od tačke do tačke (PPP) [RFC 1661; RFC 2153] je protokol sloja veze podataka koji radi nad linkom **od tačke do tačke** - linkom koji direktno povezuje dva čvora, po jednog na svakom svom kraju. Link od tačke do tačke na kome radi PPP bi mogao da bude serijska telefonska linija (na primer, veze preko modema 56K), link SONET/SDH, veza X.25, ili ISDN vodova. Kao što je napomenuto, PPP je postao protokol koji se bira za povezivanje kućnih korisnika sa njihovim posrednicima Internet usluga preko pozivne konekcije.

Pre nego što zaronimo u detalje PPP-a, poučno je da ispitamo originalne zahteve koje je organizacija IETF postavila za konstrukciju protokola PPP [RFC 1547]:

- ◆ *Stavljanje paketa u okvire.* Pošiljalac protokola PPP sloja veze podataka mora da bude sposoban da preuzme paket mrežnog nivoa i enkapsulira ga u okvir PPP sloja veze podataka, takav da će primalac moći da identificuje početak i kraj i okvira veze podataka i paketa mrežnog sloja unutar okvira.
- ◆ *Transparenost.* Protokol PPP ne srne da postavi nikakva ograničenja na podatke koji se pojavljuju na paketu mrežnog sloja (zaglavlja ili podatke). Tako, na primer, protokol PPP ne može da zabrani upotrebu izvesnih kombinacija bitova u paketu mrežnog sloja. Na ovo pitanje vratićemo se uskoro u našem objašnjenju popunjavanja bajtova.
- ◆ *Višestruki protokoli mrežnog sloja.* Protokol PPP mora da bude u stanju da istovremeno podrži više protokola mrežnog sloja (na primer, IP i DECnet) koji rade na istom fizičkom linku u isto vreme. Baš kao što se od IP protokola zahteva da multipleksira različite protokole transportnog nivoa (na primer, TCP i UDP) na istoj vezi od jednog do drugog kraja, tako i PPP mora da bude u stanju da multipleksira različite protokole mrežnog sloja po jednoj vezi od tačke do tačke. Ovaj zahtev znači da će, kao minimum, PPP verovatno zahtevati polje „tipa protokola“ ili neki sličan mehanizam, tako da prijemna strana PPP može da demultipleksira primljeni okvir naviše do odgovarajućeg protokola mrežnog sloja.
- ◆ *Više vrsta linkova.* Pored sposobnosti istovremenog prenosa višeg protokola višeg nivoa, PPP mora takođe da bude u stanju da radi na širokom skupu vrsta linkova, uključujući tu linkove koji su serijski (prenose jedan bit istovremeno u datom

pravcu) ili paralelni (prenose bitove paralelno), sinhroni (prenose signal generatora takta uz bitove podataka) ili asinhroni, male ili velike brzine, električni ili optički.

- ◆ **Otkrivanje grešaka.** PPP prijemnik mora da bude sposoban da otkrije greške u primljenom okviru.
- ◆ **Životnost konekcije.** PPP mora da bude u stanju da otkrije отказ na nivou linka (na primer, nesposobnost da se prenose podaci od predajne do prijemne strane lirika) i da signalizira tu pojavu mrežnom sloju.
- ◆ **Usaglašavanje adrese mrežnog sloja.** PPP mora da obezbedi mehanizam za komuniciranje mrežnih slojeva (na primer, IP) da bi saznali ili konfigurisali jedni drugima adrese mrežnog sloja.
- ◆ **Jednostavnost.** Od PPP-a se zahtevalo da ispunji mnoštvo dodatnih zahteva pored onih koje smo gore naveli. Prvi i najvažniji od svih bio je onaj za "Jednostavnosću". U RFC-u 1547 stoji da bi „ključna reč za protokol od tačke do tačke trebalo da bude jednostavnost“. Zaista težak zahtev, imajući u vidu sve druge zahteve koji su postavljeni za projektovanje PPP-a! Više od 50 RFC-ova (zahteva za komentarima) definišu različite aspekte ovog Jednostavnog protokola!

Iako može da izgleda kako je mnogo zahteva postavljeno za projektovanje PPP-a, situacija bi stvarno mogla da bude još mnogo teža! Projektni zahtevi za PPP takođe eksplicitno beleže i funkcionalnosti koje *nisu* zahtevi za implementaciju u PPP-u:

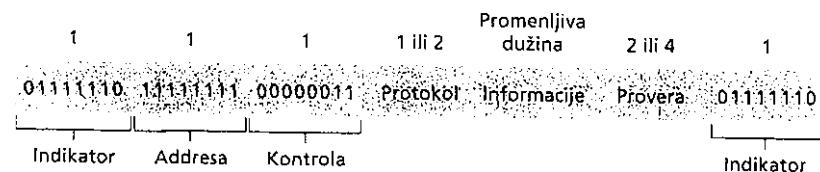
- ◆ **Ispravljanje grešaka.** Zahteva se da PPP otkrije, ali *ne i* da ispravi greške.
- ◆ **Kontrola toka.** Od prijemnika PPP očekuje se da može da prima okvire punom brzinom fizičkog sloja. Ako viši sloj ne može da preuzima pakete tom punom brzinom, onda je na fizičkom sloju da odbacuje pakete ili da priguši predajnik na višem nivou. Odnosno, umesto da PPP predajnik smanjuje sopstvenu brzinu prenosa, odgovornost protokola višeg nivoa je da usporava brzinu kojom se paketi isporučuju PPP-u za slanje.
- ◆ **Sekvenciranje.** Od PPP se *ne* zahteva da isporučuje okvire prijemniku linka u istom redosledu u kome ih je poslao predajnik linka. Zanimljivo je zapaziti da, dok je ta fleksibilnost u saglasnosti sa modelom IP usluga (koji dozvoljava IP paketima da se isporučuju od tačke do tačke u bilo kom redosledu), drugi protokoli mrežnog sloja koji rade nad PPP-om **zaista** zahtevaju isporuku paketa *u* redosledu.
- ◆ **linkovi za više stаницa-** PPP treba da radi samo na linkovima koji imaju jednog pošiljaoca i jednog primaoca. Drugi protokoli sljiva veze podataka (na primer, HDLC) mogu da prihvate više primalaca na linku (scenarij sličan Ethemetu).

Pošlo smo razmotrili ciljeve (i one koji to nisu) projekta PPP, hajde da pogledamo kako on ispunjava te ciljeve.

### 5.7.1 Pravljenje podataka u PPP okvire

Na slici 5.34 prikazanje PPP okvir podataka koji je napravljen slično HDLC-u [RFC 1662]. Okvir PPP sadrži sledeća polja:

- ◆ **Polje indikatora.** Svaki PPP okvir počinje i završava se jednobajtnim poljem indikatora sa vrednošću 01111110.
- ◆ **Polje adrese.** Jedina moguća vrednost za ovo polje je 11111111.
- ◆ **Kontrolno polje.** Jedina moguća vrednost za ovo polje je 00000011. S obzirom na to da i adresno i kontrolno polje trenutno mogu imati samo fiksne vrednosti, mogli bismo se zapitati zašto su polja uopšte i definisana. U specifikaciji PPP [RFC 1662] kaže se da druge vrednosti „mogu da se definisu kasnije“, mada ih do danas niko nije definisao. Imajući u vidu da ta polja imaju fiksne vrednosti, PPP dozvoljava predajniku da prosto ne šalje adresne i kontrolne bajtove, štedeći tako dva bajta dodatnog opterećenja u PPP okviru.
- ◆ **Protokol.** Polje za protokol saopštava PPP prijemniku protokol gornjeg sloja kome pripadaju primljeni enkapsulirani podaci (odnosno, sadržaj informacionog polja PPP okvira). Po prijemu PPP okvira, PPP prijemnik će proveriti njegovu ispravnost i zatim proslediti enkapsulirane podatke odgovarajućem protokolu. RFC 1700 i RFC 3232 definišu 16-bitne kodove protokola koje koristi PPP. Nas zanima IP protokol (odnosno, podaci enkapsulirani u PPP okvir u vidu IP datagrama), koji ima heksadecimalnu vrednost 21; drugi protokoli mrežnog sloja su npr. AppleTalk (29) i DECNet (27); PPP protokol kontrole linka (heksadecimalno C021) o kome detaljno govorimo u sledećem odeljku; i IP kontrolni protokol, IPCP (8021). Ovaj poslednji protokol PPP poziva kada se prvi put aktivira link, da bi konfigurisao vezu između uređaja sposobnih za IP na svakom kraju linka (pogledajte dalje u tekstu).
- ◆ **Informacije.** Ovo polje sadrži enkapsulirani paket (za PPP to su podaci) koji je poslao protokol gornjeg sloja (na primer, IP). Maksimalna podrazumevana dužina polja za informacije je 1500 bajtova, mada to može da se promeni kada se link prvi put konfiguriše, kao sto je objašnjeno dalje u tekstu.
- ◆ **Kontrolni zbir.** Polje za kontrolni zbir koristi se za otkrivanje grešaka u prenesenom okviru. Ono koristi ili dvobajtni ili četvorobajtni standardni HDLC ciklični redundantni kod.



**Slika 5.34** ◆ Format PPP okvira za podatke

### Popunjavanje bajtova

Pre nego što zaključimo naše izlaganje o pravljenju PPP okvira, razmotrimo problem koji se javlja kada neki protokol koristi određeni šablon bitova u polju indikatora kako bi opisao početak ili kraj okvira. Sta se dešava ako se šablon indikatora i sam pojavljuje na nekom drugom mestu u paketu? Na primer, šta se događa ako se vrednost polja indikatora 01111110 pojavljuje u polju za informacije? Da li će prijemnik neispravno otkriti kraj PPP okvira?

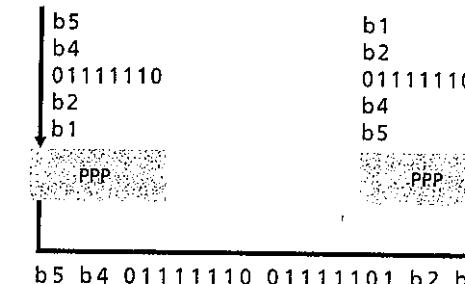
Jedan od načina da se reši ovaj problem bio bi da PPP zabrani protokolu gornjeg sloja da šalje podatke koji sadrže šablon bitova polja indikatora. Zahtev za transparenčnu PPP-a, o kome smo ranije diskutovali, odstranjuje ovu mogućnost. Alternativno rešenje, ono koje je preuzeto u PPP-u i mnogim drugim protokolima je da se koristi tehnika poznata kao popunjavanje bajtova.

PPP definiše specijalan kontrolni bajt, 01111101. Ako se sekvenca indikatora, 01111110 pojavljuje bilo gde u okviru, izuzev u polju indikatora, PPP stavlja ispred tog slučaja šablonu indikatora kontrolni bajt. To znači da „popunjava“ (dodaje) kontrolni bajt u niz podataka koji se prenosi, pre 0111110, da bi ukazao da bajt 01111-110 koji sledi nije vrednost indikatora nego, u stvari, stvarni podatak. Prijemnik koji vidi 01111110 kome prethodi 01111101 će, naravno, ukloniti popunjeni kontrolni bajt da bi rekonstruisao originalne podatke. Slično tome, ako se sam šablon kontrolnog iskočnog bajta pojavi u stvarnim podacima, njemu takođe mora da prethodi popunjeni kontrolni bajt. Na taj način, kada prijemnik ugleda jedan kontrolni bajt u nizu podataka, on zna daje bajt bio popunjeno u tom nizu. Par kontrolnih bajtova jedan do drugog Znači da se jedan od njih pojavljuje u originalnim podacima koji se šalju. Na slici 5.35 ilustrovano je PPP popunjavanje bajtova. (U stvari, PPP takođe vrši operaciju ekskluzivno ILI na bajtu podataka sa heksadecimalnom vrednošću 20, a kome je prethodio kontrolni bajt, stope detalj koji ćemo radi jednostavnosti ovde preskočiti.)

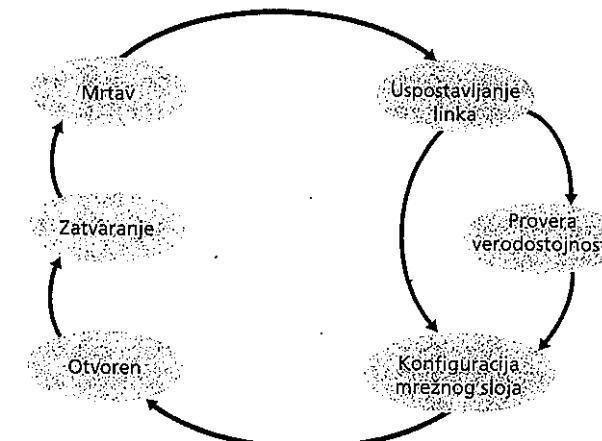
### 5.7.2 PPP protokol kontrole linka (LCP) i protokoli kontrole mreže

Do sada smo videli kako se PPP okviri šalju preko linka od tačke do tačke. Ali, kako se inicijalizuje link kada se računar ili ruter na jednom kraju PPP linka uključe prvi put? Inicijalizacija, održavanje, izveštavanje o greškama i zatvaranje PPP linka obavlja se upotrebom PPP protokola za kontrolu linka (LCP) i familije PPP protokola za kontrolu mreže.

Pre nego što se bilo kakav podatak razmeni preko PPP linka, dva ravnopravna uređaja (po jedan na svakom kraju PPP linka) moraju prvo da obave prilično mnogo rada da bi konfigurisati link, na način veoma sličan onom na koji TCP pošiljalac i primalac moraju da izvedu trostruku sinhronizaciju (procitajte odeljak 3.5) da bi podesili parametre TCP konekcije pre nego što počne prenos TCP segmenta podataka. Na slici 5.36 Hrustovan je dijagram stanja za protokol LCP za konfigurisanje, održavanje i zatvaranje PPP linka.



Slika 5.35 ♦ Popunjavanje bajtova



Slika 5.36 ♦ Stanja PPP protokola kontrole linka

PPP link uvek počinje i završava se mrtvim stanjem. Kada događaj, kao što je prepoznavanje nosioca ili intervencija administratora mreže, ukaže na to da je prisutan fizički sloj i da je spreman za upotrebu, PPP ulazi u stanje uspostavljanja linka. U tom stanju, jedan kraj linka šalje svoju željenu konfiguraciju linka koristeći LCP okvir – `configure-request` (PPP okvir sa poljem protokola podešenim na LCP i PPP poljem za informacije koje sadrži određen zahtev za konfiguraciju). Druga strana zatim odgovara okvirom `configure-ack` (sve opcije prihvatljive), okvirom `configure-nak` (sve opcije su shvaćene, ali nisu prihvatljive), ili okvirom za odbijanje konfiguracije (opcije nisu prepoznatljive, ili nisu prihvatljive za

posredovanje). LCP opcije konfiguracije uključuju maksimalnu veličinu okvira za link, specifikaciju protokola za proveru autentičnosti (ako postoji) koji treba da se koristi i opciju da se preskoči upotreba adresnih i kontrolnih polja u PPP okvirima.

Jednom kadaje link uspostavljen, opcije linka dogovorene i provera autentičnosti (ako je ima) izvršena, onda dve strane PPP linka međusobno razmenjuju pakete za kontrolu mreže specifične za mrežni sloj. Ako IP radi preko PPP linka, koristi se IP kontrolni protokol [RFC 1332] da bi se konfigurisali moduli IP protokola na svakom kraju PPP linka. Podaci IPCP se prenose unutar PPP okvira (čija je vrednost polja za protokol 8021), baš kao što se podaci LCP prenose u PPP okviru. IPCP dopušta da dva IP modula razmenjuju ili konfigurišu svoje IP adrese i pregovaraju da li će njihovi IP datagrami biti, ili neće biti poslati u komprimovanom obliku. Slični protokoli za kontrolu mreže definisani su i za druge protokole mrežnog sloja, kao što su DECnet [RFC 1762] i AppleTalk [RFC 1378]. Jednom kada se konfiguriše mrežni sloj, PPP može da počne sa slanjem datagrama mrežnog sloja - link je u otvorenom stanju i podaci počinju da teku preko PPP linka. LCP okvir sa zahtevom za echo i okvir sa echo odzivom mogu da se razmene između dve krajnje tačke PPP-a da bi se proverilo stanje linka.

PPP link ostaje konfigurisan za komunikaciju sve dok se ne pošalje LCP paket sa zahtevom za završetak. Ako se LCP okvir sa zahtevom za završetak pošalje sa jednog kraja PPP linka i na njega odgovori sa LCP okvirom za potvrdu završetka, link ulazi u mrtvo stanje.

Sve u svemu, PPP je protokol veze podataka pomoću koga dve stанице nivoa linka razmenjuju PPP okvire koji sadrže datagrame mrežnog sloja. Osnovne komponente PPP-a su:

- ◆ *Pravljenje okvira.* Metoda za enkapsuliranje podataka u PPP okvir, koja identificuje početak i kraj okvira i otkriva greške u okviru.
- ◆ *Protokol kontrole linka.* Protokol za inicijalizovanje, održavanje i ukidanje PPP linka.
- ◆ *Protokoli za kontrolu mreže.* Skup protokola, po jedan za svaki protokol gornjeg sloja mreže, koja dopušta modulima mrežnog sloja da se sami konfigurišu pre nego što datagrami mrežnog nivoa počnu da teku preko PPP linka.

## 5.8 Vizuelizacija linka: mreža kao sloj veze

Imajući u vidu da se u ovom poglavlju bavimo protokolima sloja veze podataka i da se sada primičemo njegovom kraju, hajde da razmislimo koliko se razvilo naše razumevanje termina *link*. Počeli smo ovo poglavlje sagledavanjem linka kao fizičkog provodnika koji povezuje dva računara u komunikaciji, kao što je ilustrovano na slici 5.2. U proučavanju protokola sa višestrukim pristupom, videli smo da više računara može da se poveže pomoću deljenog provodnika i da bi „žica“ koja ih povezuje mogla da bude radio spektar ili drugi medijumi. To nas je navelo da posmatramo

link apstraktnije, kao kanal, a ne kao provodnik. U našoj studiji Etherneta LAN-ova (slike 5.26 - 5.28), videli smo da bi medijumi za međusobno povezivanje mogli da budu dosta složene komutacione strukture. Međutim, kroz ceo taj razvoj, sami računari su zadržavali predstavu daje medijum za međusobno povezivanje prostog kanala sloja veze podataka koji povezuje dva ili više računara. Videli smo, na primer, da Ethernetski računar može da bude blaženo nesvestan da li je sa drugim računarima u LAN-u povezan pomoću kratkog segmenta LAN-a (slika 5.9), ili pomoću geografski raširenog komutiranog LAN-a (slika 5.28).

U odeljku 5.7 smo videli da se protokol PPP često koristi za savremene komunikacije između dva računara. Ovde je link koji povezuje dva računara u stvari telefonska mreža - logički odvojena, globalna telekomunikaciona mreža sa svojim sopstvenim komutatorima, linkovima i skupovima protokola za prenos podataka i signalizaciju. Međutim, sa tačke gledišta Internetskog sloja veze podataka, na pozivnu konekciju preko telefonske mreže se gleda kao na običnu „žicu“. U tom smislu, Internet vizualizuje telefonsku mrežu, videvši je kao tehnologiju sloja veze podataka koja obezbeđuje povezivost sloja veze podataka između dva računara na Internetu. Setite se iz naše diskusije o prekrivajućim mrežama u poglavlju 2, da jedna takva mreža slično vidi Internet kao sredstvo za obezbeđivanje povezivosti između čvorova, tražeći prekrivanje na Internetu na isti način kao što Internet prekriva telefonsku mrežu.

U ovom odeljku, razmotrićemo asinhroni režim prenosa (*asynchronous transfer mode, ATM*) i mreže sa višeprotokolnim komutiranjem na osnovu oznaka (*Multi-protocol Label Switching, MPLS*). Za razliku od telefonskih mreža sa komutacijom kola, i ATM i MPLS su sa komutacijom paketa, mreže sa virtuelnim kolima na svoj način. One imaju svoje sopstvene formate paketa i ponašanje prilikom prosleđivanja. Dakle, sa pedagoške tačke gledišta, diskusija o ATM-u i MPLS-u se dobro uklapa u sloj veze podataka. Međutim, sa tačke gledišta Interneta, možemo da smatramo ATM i MPLS, kao i telefonsku mrežu i komutirani Ethernet, kao tehnologije sloja veze podataka koje služe da povezuju IP uređaje. Iz tog razloga ćemo razmatrati i MPLS i ATM u našoj diskusiji o sloju veze podataka. Mreže Frame Relay takođe mogu da se upotrebile za međusobno povezivanje IP uređaja, ali one predstavljaju nešto stariju (mada i dalje primenjiviju) tehnologiju, pa ih ovde nećemo objašnjavati; za detalje pogledajte vrlo čitljivu knjigu [Goralski 1999]. Naša obrada ATM-a i MPLS-a će namemo biti kratka, jer bi o tim mrežama mogla da se napiše (i napisana je) čitava knjiga. Ovde ćemo se uglavnom usredosrediti na to kako ove mreže služe da međusobno povezuju IP uređaje, iako ćemo se dublje upustiti i u tehnologije koje čine njihovu osnovu.

### 5.8.1 Asinhroni režim prenosa (ATM)

Standardi za asinhroni režim prenosa (ATM) prvi put su razvijeni sredinom 1980-ih godina, sa ciljem projektovanja jedne mrežne tehnologije koja bi se upotrebljavala za prenos govora i videa u realnom vremenu, kao i teksta, slika i elek-

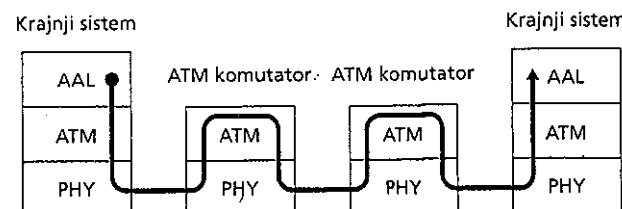
tronske pošte. Dva komiteta za standarde, ATM Forum [ATM 2002] i International Telecommunications Union [ITU 2002], razvili su standarde za ATM.

### Glavne karakteristike ATM-a

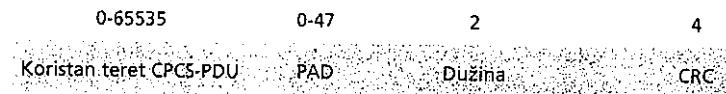
Kao stoje rečeno u odeljku 4.1, ATM podržava više modela usluga, uključujući usluge konstantne, promenljive, raspoložive i nespecifirane bitske brzine. ATM je mrežna arhitektura za komutaciju paketa sa virtualnim kolima (VC). Setite se da smo u izvesnoj meri razmatrali virtualna kola u odeljku 4.2.1, Opšta arhitektura ATM-a je organizovana u tri sloja, kao što je prikazano na slici 5.37.

**ATM adaptacioni sloj (AAL)** približno je analogan Internetovom transportnom sloju i prisutan je samo na ATM uređajima na granici ATM mreže. Na predaj-noj strani, AAL prosleduje podatke iz aplikacije višeg nivoa ili protokola (kao što je IP, ako se ATM koristi za povezane IP uređaje). Na prijemnoj strani, on prosledjuje podatke naviše ka protokolu višeg nivoa ili aplikaciji. AAL-ovi su definisani za usluge konstantne bitske brzine i emulaciju kola (AAL1), za usluge promenljive bitske brzine kao stoje video promenljive bitske brzine (AAL2) i za usluge podataka kao stoje transport IP datagrama (AAL5). Među uslugama koje izvršava AAL su otkrivanje greške i segmentacija/ponovo sastavljanje. Jedinicu podataka kojom rukuje AAL generički se zove AAL protokolna jedinica podataka (*protocol data unit, PDU*), i približno je ekvivalentna UDP ili TCP segmentu.

AAL PDU je prikazana na slici 5.38. PDU polja su relativno jasna. PAD obezbeđuje daje PDU celobrojni umnožak od 48 bajtova, zato što će se PDU izdeliti na segmente tako da stane u 48-bajtne korisne terete ATM paketa (poznatih kao ATM *ćelije*). Dužina polja identificuje veličinu korisnog tereta PDU, tako da PAD može da se ukloni kod prijemnika. Polje CRC obezbeđuje otkrivanje greške, koristeći istu cikličnu proveru redundantnosti kao Ethernet. Polje za koristan teret može da bude dužine do 65535 bajtova.



Slika 5.37 ♦ Tri ATM sloja. Sloj AAL je prisutan samo na krajevima ATM mreže.

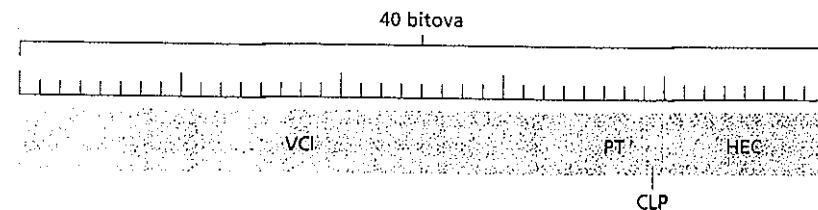


Slika 5.38 ♦ AAL5 PDU

Hajde da se spustimo sloj niže i da razmatramo **ATM sloj**, koji leži u srcu ATM arhitekture. ATM sloj definiše strukturu ATM ćelije i značenja polja unutar ćelije. ATM ćelija je važna za ATM mrežu kao stoje to IP datagram za IP mrežu. Prvih 5 bajtova ćelije čini ATM zaglavljje; preostalih 48 bajtova čine ATM koristan teret. Na slici 5.39 prikazana je struktura zaglavljiva ATM ćelije.

Polja u ATM ćeliji imaju sledeće funkcije:

- ♦ **Identifikator virtuelnog kanala (VCI).** Pokazuje virtuelni kanal kome pripada ćelija. Kao i u većini mrežnih tehnologija koje koriste virtualna kola, ćelijin VCI se prenosi od linka do linka (pročitajte odeljak 4.2.1).
- ♦ **Vrsta korisnog tereta (PT).** Pokazuje vrstu korisnog tereta koji sadrži ćeliju. Imat će više vrsta korisnog tereta podataka, više vrsta korisnog tereta za održavanje i vrsta korisnog tereta slobodne ćelije. Polje PT takođe obuhvata bit koji služi da pokaže na poslednju ćeliju u fragmentovanoj AAL PDU.
- ♦ **Bit prioriteta gubitka ćelije (CLP).** Tvor može da ga postavi da bi se napravila razlika između saobraćaja visokog i niskog prioriteta. Ako dođe do zagušenja i ATM komutator mora da odbaci ćelije, on može da upotrebi taj bit kako bi prvo odbacio saobraćaj niskog prioriteta.
- ♦ **Bajt za kontrolu greške zaglavljiva (HEC).** Bitovi za otkrivanje grešaka koji štite zaglavljiva ćelije.



Slika 5.39 ♦ Format zaglavljiva ATM ćelije

Pre nego što izvor može da počne sa slanjem célija odredištu, ATM mreža mora da uspostavi virtuelni kanal (VC) od izvora do odredišta. Virtuelni kanal nije ništa više od virtuelnog kola, opisanog u odeljku 4.2.1. Svaki VC je putanja koja se sastoji od niza linkova između izvora i odredišta. Identifikator virtuelnog kanala (VCI) se pridružuje svakom linku na VC-u. Kad god se VC uspostavi ili ukine, moraju da se ažuriraju VC tabele (procitajte odeljak 4.3.1). Ako se koriste trajni VC-ovi, nema potrebe za njihovim dinamičkim uspostavljanjem i ukidanjem. Kada se traži dinamičko uspostavljanje i ukidanje VC-ova, protokol Q.2931 [Black 1997, ITU-T Q.2-931] obezbeđuje potrebnu signalizaciju između ATM komutatora i krajnjih sistema.

ATM fizički sloj je na samom dnu familije ATM protokola i bavi se naponima, tajmirigom bitova i pravljenje okvira na fizičkom medijumu. Dobar deo fizičkog sloja zavisi od fizičkih karakteristika linka. Postoje dve široke klase fizičkih slojeva: oni koji imaju strukturu okvira prenosa (na primer, T1, T3, SONET ili SDH) i oni koji je nemaju. Ako fizički sloj ima strukturu okvira, on je onda odgovoran za generisanje i izdvajanje okvira. Termin *okviri* u ovom odeljku ne treba da se pomeša sa okvirima sloja veze podataka (odnosno, Etherneta), koji se koriste u prethodnim odeljcima ovog poglavlja. Okvir prenosa je mehanizam fizičkog sloja sličan TDM-u, za organizovanje bitova koji se šalju preko linka. Neki mogući fizički slojevi uključujući:

- ◆ SONET/SDH (sinhrona optička mreža/ sinhrona digitalna hijerarhija) po mono-modnom optičkom vlaknu. Kao T1 i T3, SONET i SDH imaju strukturu okvira kpja uspostavlja sinkronizaciju bitova između predajnika i prijemnika na dva kraja linka. Imma više standardizovanih brzina, uključujući:

OC-3: 51,84 Mb/s OC-3:

155,52 Mb/s OC-12:

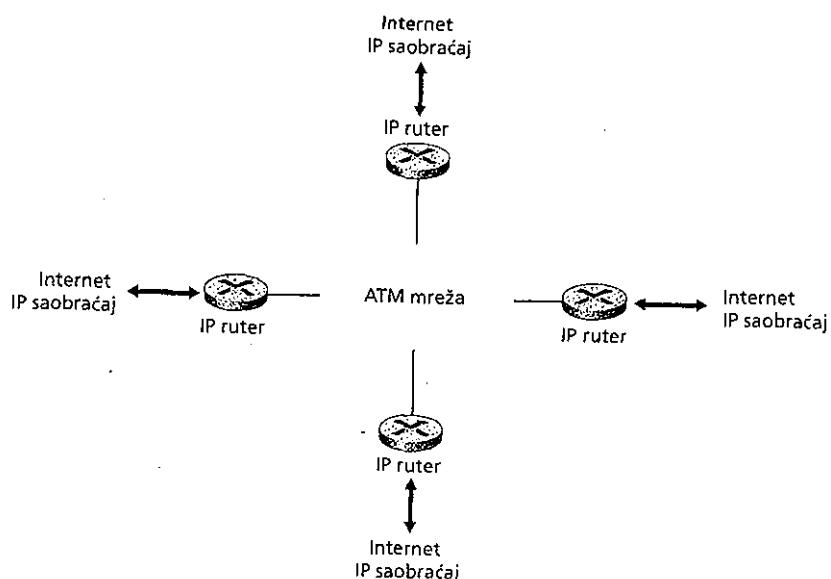
622,08 Mb/s OC-48: 2,5

Gb/s

- ◆ T1/T3 okviri preko optičkog vlakna, mikrotalasa i bakarnih provodnika.
- ◆ Zasnovani na célijama, bez okvira. U ovom slučaju, generator takta prijemnika se izvodi iz predajnog signala.

#### IP preko ATM-a

Hajde sada da razmotrimo kako ATM mreža može da se upotrei da bi obezbedila veza između IP uređaja. Na slici 5.40 prikazana je ATM okosnica sa četiri ulazne/ izlazne tačke za Internet IP saobraćaj. Zapazite da je svaka ulazna/izlazna tačka ruter. ATM okosnica može da se prostire preko celog kontinenta i može da ima desetine ili čak stotine ATM komutatora. Većina ATM okosnica ima postojani virtuelni kanal (VC) između svakog para ulaznih/izlaznih tačaka. Upotreboom postojanih VC-ova, ATM célige se rutiraju od ulazne do izlazne tačke a da pri tom VC-ovi ne moraju da se dinamički uspostavljaju i ukidaju. Međutim, postojani VC-ovi su izvodljivi samo



**Slika 5.40** ◆ ATM mreža u jezgru Internet okosnice

kada je broj ulaznih/izlaznih tačaka relativno mali. Za  $n$  ulaznih tačaka, potrebno je  $n(n - 1)$  postojanih VC-ova da direktno povezuju ulaznih/izlaznih tačaka.

Interfejsu svakog ruteru koji se povezuje sa ATM mrežom biće potrebne dve adrese, na dosta sličan način kao što su IP računari potrebne dve adrese za Ethernet interfejs: IP adresa i MAC adresa. Slično tome, ATM interfejs će imati IP adresu i MAC adresu. Razmotrite sada IP datagram koji treba da se prenese preko ATM mreže na slici 5.40. U najprostijem slučaju, ATM mreža izgleda kao jedan logički link - ATM međusobno povezuje četiri ruteru. Hajde da ruter u kome datagram ulazi u ATM mrežu zovemo „ulazni ruter“, a ruter sa koga datagram napušta mrežu zovemo „izlazni ruter“. Ulagani ruter radi sledeće:

1. Ispituje adresu odredišta datograma.
2. Indeksira svoju tabelu rutiranja i određuje IP adresu izlaznog ruteru (odnosno, sledećeg ruteru na putanji **datagTama**).
3. Da bi doveo datagram na izlazni ruter, ulazni ruter vidi ATM samo kao jedan protokol flloja veze podataka. Da bi se datagram preneo na sledeći ruter, mora da se odredi fizička adresa ruteru u sledećem skoku. Setite se iz naše rasprave u odeljku 5.4.2, da se to radi upotreboom ARP-a. U slučaju ATM interfejsa, ulazni

ruter indeksira ATM ARP tabelu sa IP adresom izlaznog rutera i određuje ATM adresu izlaznog rutera. Protokol ATMARP je opisan u [RFC 2225]. 4. IP u ulaznom ruteru onda prosledjuje naniže datagram sloju veze podataka (odnosno ATM-u) zajedno sa ATM adresom izlaznog rutera.

Kada se završe ova četiri postupka, posao prenosa datagrama u izlazni ruter je van kontrole IP-a i prelazi u ruke ATM-a. ATM sada mora da prenese datagram na adresu ATM odredišta, dobijenu u gore pomenutom postupku 3. Ovaj zadatak ima dva podzadataka:

1. Određivanje VCI-ja za VC koji vodi do adrese ATM odredišta.
2. Deljenje datagrama na celije na predajnoj strani VC-a (odnosno, u ulaznom ruteru) i ponovno sastavljanje celija u prvobitni datagram na prijemnoj strani VC-a (odnosno, u izlaznom ruteru).

Prvi od navedenih podzadataka je jasan. Interfejs na predajnoj strani održava tabelu koja preslikava ATM adrese u VCI-ove. S obzirom na to da smo pretpostavili da su VC-ovi postojani, ova tabela je ažurna i statična. (Ako VC-ovi ne bi bili postojani, onda bi bio potreban ATM stognalni protokol Q.2931 da dinamički uspostavlja i ukida VC-ove.) Drugi podzadatak zaslužuje pažljivije razmatranje. Jedan pristup je da se koristi IP fragmentacija, kao što je objašnjeno u odeljku 4.4.- Kod IP fragmentacije, predajni ruter bi prvo izdelio prvobitni datagram na fragmente, gde nijedan fragment ne bi bio veći od 48 bajtova, tako da bi mogao da stane u koristan teret ATM celije. Ali, taj pristup fragmentacije ima veliki problem - svaki IP fragment po pravilu ima 20 bajtova zaglavljiva, tako da bi ATM celija koja ga prenosi imala 25 bajtova „jalovog opterećenja“ i samo 28 bajtova korisnih informacija. ATM zato koristi AAL5 da bi obezbedio efikasniji način za deljenje i ponovno sastavljanje datagrama.

ATM mreža zatim prenosi svaku celiju kroz mrežu do adrese ATM odredišta. U svakom ATM komutatoru između ATM izvora i odredišta, ATM celiju obrađuju ATM fizički sloj i ATM sloj, ali ne i AAL sloj. U svakom komutatoru VCI se obično prevodi (pročitajte odeljak 4.2.1) i ponovo se izračunava HEC. Kada celije stignu na adresu ATM odredišta, one se usmeravaju u privremenu memoriju AAL koja je rezervisana za određeni VC. AAL5 PDU se rekonstruiše i IP datagram se izdvaja i prosledjuje naviše protokolu IP sloja.

## 5.8.2 Višeprotokolna komutacija na osnovu oznaka (MPLS)

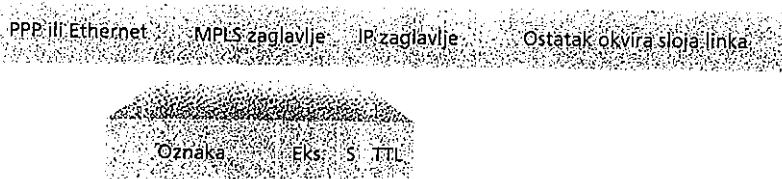
Višeprotokolna komutacija na osnovu oznaka (MPLS) se razvila iz brojnih industrijskih napora, od sredine da kraja 1990-ih godina, da se poboljša brzina prosleđivanja IP rutera usvajanjem ključnog koncepta iz sveta mreža sa virtuelnim kolima: oznake fiksne dužine. Cilj nije bio da se napusti infrastruktura prosleđivanja IP datagrama na osnovu odredišta radi one koja je zasnovana na oznakama fiksne dužine i virtu-

elnim kolima, nego da se ona pojača selektivno označavajući datagrame i dozvoljavajući ruterima da, kadaje to moguće, prosleđuju datagrame na osnovu oznaka fiksne dužine (a ne na osnovu IP adresa odredišta). Važno je da ove tehnike idu ruku po ruku sa IP-om, koristeći IP adresiranje i rutiranje. IETF je ujedinio te napore u protokolu MPLS [RFC 3031, RFC 3032], efikasno spajajući VC tehnike u mreži rutiranih datagrama.

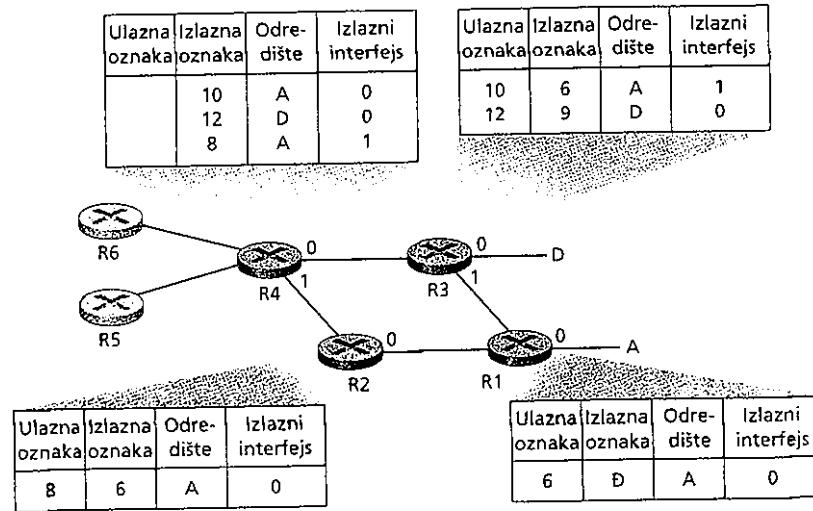
Hajde da počnemo naše proučavanje MPLS-a razmatranjem formata okvira sloja veze podataka sa kojim radi ruter sposobljen za MPLS. Na slici 5.41 pokazuje se da okvir sloja veze podataka koji se prenosi na PPP linku ili LAN-u (kao što je Ethernet) ima malo MPLS zaglavljive, dodato između zaglavja sloja-2 (na primer, PPP ili Ethernet) i zaglavja sloja-3 (na primer, IP). RFC 3032 definiše format MPLS zaglavja za takve linkove; zaglavja se definišu za ATM i Frame Relay mreže, kao i u drugim RFC-ovima. Među poljima MPLS zaglavja su oznaka (koja služi kao identifikator virtualnog kola, koga smo sreli ranije u odeljku 4.2.1), 3 bita rezervisana za eksperimentalnu upotrebu, jedan S bit, koji se koristi da ukaže na kraj serije „naslaganih“ MPLS zaglavja (naprednija tema, koju ovde nećemo objašnjavati) i polje sa životnim vremenom.

Sa slike 5.41 se odmah vidi da okvir koji je ojačan pomoću MPLS-a može da se šalje samo između ruta koji su oba sposobljeni za MPLS (zato što bi ruter koji nije sposobljen za MPLS bio sasvim zburnjen kada bi pronašao MPLS zaglavje tamo gde je očekivao da nade IP zaglavje!). Ruter sposobljen za MPLS se često zove ruter komutiran pomoću oznake, zato što prosleđuje MPLS okvir tražeći MPLS oznaku u svojoj tabeli prosleđivanja i odmah prosleđujući datagram na odgovarajući izlazni interfejs. Dakle, ruter sposobljen za MPLS ne izdvaja adresu odredišta i traži podudarnost najdužeg prefiksa u tabeli prosleđivanja. Ali kako ruter zna da li je njegov sused zaista sposobljen za MPLS i koju oznaku da pridruži datom IP odredištu? Da bismo odgovorili na ova pitanja, biće potrebno da bacimo pogled na interakciju unutar grupe ruta sposobljenih za MPLS.

U primeru na slici 5.42, ruteri R1 do R4 su sposobljeni za MPLS. R5 i R6 su standardni IP ruteri. R1 je najavio R2 i R3 da on (R1) može da rutira ka odredištu A i da će primljeni okvir sa MPLS oznakom 6 biti prosleđen odredištu A. Ruter R3 je



**Slika 5.41** ♦ MPLS zaglavje: između zaglavja sloja veze podataka i zaglavja mrežnog sloja



**Slika 5.42** ♦ Prosleđivanje ojačano pomoću MPLS-a

najavio ruteru R4 da on može da rutira ka odredištima A i D i da će dolazeći okviri sa MPLS oznakama 10 i 12 respektivno biti komutirani ka tim odredištima. Ruter R2 je takođe najavio ruteru R4 da on (R2) može da dostigne odredište A i da primljeni okvir sa MPLS oznakom 8 može da se komutira ka A. Zapazite daje ruter R4 sada u zanimljivom položaju, zato što ima *dve* MPLS putanje da stigne do odredišta A preko interfejsa 0 sa izlaznom MPLS oznakom 10 i preko interfejsa 1 sa MPLS oznakom 8. Na slici 5.42 je prikazano da su IP R4, R5, A i D međusobno povezani preko MPLS infrastrukture (ruteri sposobljeni za MPLS R1, R2, R3 i R4) na dosta sličan način kako komutirani LAN ili ATM mreža mogu da povezu IP uređaje. I kao komutirani LAN ili ATM mreža, ruteri sposobljeni za MPLS R1 do R4 to rade *a da nikada i ne dotaknu IP zaglavje paketa*.

U našoj diskusiji nismo navodili specifičan protokol koji se koristi za raspodelu oznaka između ruta osposobljenih za MPLS, zato što su detalji te signalizacije izvan domena ove knjige. Međutim, napominjemo da je radna grupa IETF za MPLS odredila u RFC 3468 da će proširenje protokola RSVP (koji ćemo proučiti u poglavljvu 7), poznato kao RSVP-TE [RFC 3209] biti žiža njenih napora za MPLS signalizaciju. Zato zainteresovanpm Čitaocu prepričujemo da pročita RFC 3209.

Do sada je naglasak u našoj raspravi o MPLS bio na Činjenici da MPLS *izvodi* komutaciju zasnovanu na oznakama, bez potrebe da razmatra IP adresu paketa. Prava prednost MPLS i razlog trenutnog interesa za MPLS nije, međutim, u potencijalnim povećanjima brzina komutacije, nego pre u novim sposobnostima za upravljanje saobraćajem koje daje MPLS. Kao Stoji gore napomenuto, R4 ima *dve* MPLS

putanje do A. Ako bi prosleđivanje bilo izvršeno do IP sloja na osnovu IP adrese, IP protokoli za rutiranje koje smo proučili u poglavljju 4 bi odredili samo jednu, najjeftiniju putanju do A. Dakle, MPLS obezbeđuje sposobnost da se paketi prosleđuju putanjama koje ne bi bile moguće korišćenjem standardnih IP protokola za rutiranje. To je jednostavan oblik inženjerstva saobraćaja upotreboom MPLS [RFC 3346, Xiao 2000], u kome operator mreže može da stavi van snage normalno IP rutiranje i da prisili deo saobraćaja usmeren ka datom odredištu da ide jednom putanjom, a drugi deo saobraćaja usmeren ka istom odredištu da ide drugom putanjom (zbog politike, performanse ili iz nekog drugog razloga).

Moguće je, takođe, upotrebiti MMPLS za mnoge druge namene. Na primer, da se izvede brzo obnavljanje MPLS putanja za prosleđivanje, odnosno da se ponovo rutira saobraćaj preko preračunate putanje kao odziv na otkaz linka [Kar 2000, Huang 2002, RFC 3469]. MPLS takođe može da se upotrebí za implementaciju okvira za diferencirano uslugu, koju ćemo proučiti u poglavljvu 7. Najzad, napominjemo da MPLS može da se upotrebí, i već je bio upotrebljen, za implementaciju takozvanih *virtuelnih privatnih mreža* (*virtualprivate networks*, VPN). U implementaciji VPN za kupca, ISP koristi svoju mrežu sposobljenu za MPLS, da bi međusobno povezao kupčeve različite mreže. MPLS može da se upotrebí da bi izolovao i resurse i adresiranje koje koristi kupčeva VPN od drugih korisnika koji prolaze kroz ISP-ovu mrežu; za detalje pročitajte [DeClercq 2002].

Naša rasprava o MPLS je namerno bila kratka i mi vam prepričujemo da pogledate reference koje smo spomenuli. Napominjemo da uz toliko mnogo mogućih upotreba MPLS, izgleda da ona brzo postaje „Švajcarski perorez“ inženjerstva saobraćaja na Internetu!

## 5.9 Rezime

U ovom poglavljvu, ispitali smo sloj veze podataka - njegove usluge, principne na kojima se zasniva rad i izvestan broj značajnih protokola koji te principne koriste u implementiranju usluga veze podataka.

Videli smo daje osnovna usluga sloja veze podataka da prenosi datagram mrežnog sloja od jednog Čvora (rutera ili računara) do susednog Čvora. Takođe, videli smo da svi protokoli veze podataka rade pomoću enkapsuliranja datagrama mrežnog sloja u okvir sloja veze podataka pre prenošenja okvira preko „linka“ do susednog čvora. Međutim, izvan ove zajedničke funkcije pravljjenja okvira, naučili smo da različiti protokoli sloja veze podataka obezbeđuju veoma različite usluge pristupa linku, isporuke (pouzdanost, otkrivanje/ispravljanje grešaka), kontrole toka i prenosa (na primer, puni dupleks ili polidupleks). Te razlike su delom posledica različitih vrsta linkova preko kojih protokoli sloja veze podataka moraju da rade. Jednostavan link od tačke do tačke ima jednog pošiljaoca i jednog primaoca koji komuniciraju preko jedne „žice“. Link sa višestrukim pristupom se deli između više pošiljalaca i

primalaca; zbog toga, protokol veze podataka za kanal sa višestrukim pristupom ima protokol za usklajivanje pristupa linku. U slučajevima ATM, X.25 i Frame Relay, videli smo da „link“ koji povezuje dva susedna čvora (na primer, dva IP rutera koji su susedni u smislu IP-a - da su IP ruteri sledećeg skoka ka nekom odredištu) mogu u stvari da budu **mreža** u i po samoj sebi. U izvesnom smislu, ideja da se mreža posmatra kao „link“ ne **bi** trebalo da izgleda čudno. Na primer, telefonski „link“ koji povezuje kućni moden Vračunar sa udaljenim modemom/ruterom, u stvari je putanja kroz usavršenu i složenu telefonsku **mrežu**.

Između principa na kojima se zasniva komunikacija veze podataka, ispitali smo tehnike otkrivanja i ispravljanja grešaka, protokole sa višestrukim pristupom, adresiranje veze podataka i konstrukciju proširenih lokalnih računarskih mreža preko habova i komutatora. U slučaju otkrivanja/ispravljanja grešaka, ispitali smo kako je moguće dodati bitove zaglavju okvira da **bi** se otkrile **i**, u nekim slučajevima, ispravile greške koje **bi** mogle da se dogode kada se okvir prenosi preko linka. Objasnili smo jednostavne Šeme parnosti i kontrolnog zbira, kao i robustniju cikličnu proveru redundantnosti. Onda smo prešli na temu protokola sa višestrukim pristupom. Identifikovali smo i prostidirali tri široka načina za usklajivanje pristupa difuznom kanalu: pristupe deljenja kanala (TDM, FDM, CDMA), metode slučajnog pristupa (protokoli ALOHA i CSMA) i metode sa pristupom po redu (prozivanje i prosleđivanje žetona). Videli smo da je posledica toga što više čvorova deli jedan difuzni kanal potreba da se obezbede adrese Čvorova na nivou veze podataka. Naučili smo da su fizičke adrese sasvim drugačije od adresa mrežnog sloja **i** da se, u slučaju Interneta, koristi specijalni protokol (ARP - protokol za prevodenje adresa) da bi izvršio prevodenje između ta dva oblika adresiranja. Onda smo ispitali kako Čvorovi koji dele difuzni kanal formiraju lokalnu mrežu računara (LAN) i kako više LAN-ova mogu da se povezu da bi formirali veće LAN-ove, sve to **bez** primene tehnika rutiranja radi međusobnog povezivanja tih lokalnih čvorova.

Takođe smo detaljno objasnili izvestan broj specifičnih protokola sloja veze podataka - Etherneta i PPP. Završili smo naše proučavanje sloja veze podataka usred-sređujući se na to kako ATM i MPLS mreže obezbeđuju usluge sloja veze podataka kada povezuju IP rutere. Pošto smo objasnili sloj veze podataka, **naše putovanje niz familiju protokola je sada završeno!** Svakako, fizički sloj leži ispod sloja veze podataka, ali detalje o fizičkom sloju možda je najbolje ostaviti za neki drugi kurs (na primer, u teoriji komunikacija, pre nego u računarskom umrežavanju). Međutim, ipak smo dotakli nekoliko aspekata fizičkog sloja u ovom poglavlju (na primer, u našoj kratkoj raspravi o Manchester kodovanju u odeljku 5.5) i U poglavlju 1 (naša rasprava o fizičkim medijumima u odeljku 1.4). Razmatraćemo opet fizički sloj kada se budemo baviti karakteristikama bežičnih veza u sledećem poglavlju.

Iako je putovanje kroz protokole završeno, naša studija o računarskom umrežavanju ipak nije došla do kraja. U sledeća Četiri poglavlja objasnićemo bežično umrežavanje, multimedijalno umrežavanje, bezbednost mreže i upravljanje mrežom. Te četiri teme se ne uklapaju pogodno ni u jedan sloj; zaista, svaka tema preseca više

## **p| Domaći zadatak: problemi i pitanja Poglavlje 5**

### Kontrolna pitanja

#### ODEUCIS5.1-5.2

1. Ako bi svi linkovi u Internetu obezbedivali uslugu pouzdane isporuke, da li bi TCP usluga pouzdane isporuke bila kompletno redundantna? Zašto da, ili zašto ne?
2. Koje su neke od mogućih usluga koje protokol sloja veze podataka može da ponudi mrežnom sloju? Koje od ovih usluga sloja veze podataka imaju odgovarajuće usluge u IP-u? U TCP-u?

#### ODEUAK 5.3

3. Pretpostavite da dva čvora počinju da šalju paket dužine  $L$  u isto vreme preko difuznog kanala brzine  $R$ . Označite kašnjenje usled propagacije između dva čvora kao  $t_{prop}$ . Da li će biti kolizija ako je  $t_{prop} < URI$ ? Zašto hoće, ili zašto neće?
4. U odeljku 5.3, naveli smo četiri poželjne karakteristike difuznog kanala. Koje od tih karakteristika ima ALOHA sa odsecima? Koje od tih karakteristika ima prosleđivanje žetona?
5. Opишite protokole sa prozivanjem i sa prosleđivanje žetona, koristeći analogiju interakcija na koktelu.
6. Zašto bi Token Ring protokol bio neefikasan ako bi LAN imao veoma velik obim?

#### ODEUAK 5.A

7. Koliko je veliki MAC adresni prostor? Adresni prostor IPv4? Adresni prostor IPv6?
8. Pretpostavite daje svaki od čvorova A, B i C priključen na isti difuzni LAN (preko njihovih adaptera). Ako A šalje hiljadu IP datagrama ka B sa svakim od enkapsuliranih okvira adresiranih na MAC adresu B, da li će adapter čvora C obradivati te okvire? Ako hoće, da li će adapter C prosleđivati IP datagrame u tim okvirima Čvoru C (odnosno, Čvoru kome adapter pripada)? Kako će se vas "odgovor izmeniti ako A šalje okvire sa difaznom MAC adresom?
9. Zašto se upit ARP šalje unutar difuznog okvira? Zašto se odziv ARP šalje unutar okvira sa određenom MAC adresom odredišta?

10. Za mrežu na slici 5.19, ruter ima dva ARP modula, od kojih je svaki sa svojom sopstvenom ARP tabelom. Da li je moguće da se ista MAC adresa pojavi u obe tabeli?

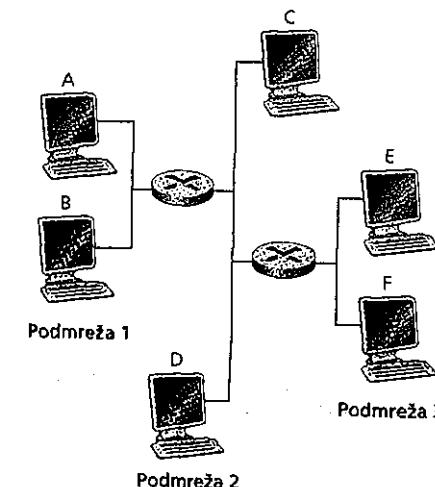
#### ODEUAK 5.5

11. Poredite strukture okvira za IOBaseT, IOOBaseT, 802.11b i Gigabit Ethernet. U čemu se one razlikuju?
12. Prepostavite da adapter od 10 Mb/s šalje u kanal beskrajan niz jedinica, koristeći Manchester kodovanje. Koliko će promena u sekundi imati signal koji se pojavljuje iz adaptora?
13. U CSMA/CD, posle pete kolizije, kolika je verovatnoća da vrednost  $K$  koju čvor bira bude 4? Kolikom kašnjenju u sekundama na Ethernetu od 10 Mb/s odgovara rezultat  $K = 4$ ?

## Problemi

- Prepostavite da je informacioni sadržaj paketa Šablon bitova 1010101010101011 i da se koristi parna šema pamosti. Koja bi bila vrednost kontrolnog zbiru u slučaju dvodimenzionalne parne řeme? Vaš odgovor bi trebalo da bude takav da se koristi polje kontrolnog zbiru minimalne dužine.
- Pokažite (dajte primer drugačiji od onog sa slike 5.6) da dvodimenzionalna provera pamosti može da ispravi i otkrije jednu grešku bita. Pokažite (dajte primer) da pogrešna dva bita mogu da se otkriju, ali ne mogu da se isprave.
- Prepostavite da informacioni deo paketa (D na slici 5.4) sadrži deset bajtova koji se sastoje od 8-bitnih binarnih prezentacija (bez znaka) celih brojeva od 0 do 9. Izračunajte Internet kontrolni zbir za ove podatke.
- Razmotrite 4-bitni generator, G, prikazan na slici 5.8, i prepostavite da D ima vrednost 10101010. Koja je vrednosti?
- U odeljku 5.3, dali smo kratko izvođenje efikasnosti ALOHE sa odsečcima. U ovom problemu upotpunimo to izvođenje.
  - Setite se da kada postoji //aktivnih čvorova, efikasnost ALOHE sa odsečcima je data izrazom  $Np(l-p)N-l$ . Pronadite vrednost  $p$  koja daje maksimalnu vrednost tog izraza.
  - Koristeći vrednost  $p$  pronađenu u delu a, pronađite efikasnost ALOHE sa odsečcima ako A'teži beskonačnosti. (Savet:  $(1-l/A)$  teži ka  $1/e$  kada  $N$  teži beskonačnosti.)
- Pokažite daje efikasnost čiste ALOHE jednaka  $1/(2e)$ . Zapazite: ovaj problem je jednostavan, ako ste resili prethodni problem!

- Nacrtajte grafik efikasnosti ALOHE sa odsečcima i čiste ALOHE u funkciji od  $p$ ;  $V=100$ .
- Razmotrite difuzni kanal sa JYčvorova i brzinom prenosa od  $R$  b/s. Prepostavite da difuzni kanal za višestruki pristup koristi prozivanje (sa dodatnim čvorom koji proziva). Prepostavite da je dužina vremena od kada čvor završi prenos do kada se sledećem čvoru dopusti da prenosi (odносно, kašnjenje prozivanja)  $t_{pmi}$ . Prepostavite da se u jednoj rundi prozivanja datom čvoru dopušta da prenese najviše  $Q$  bitova. Koja je maksimalna propusna moć difuznog kanala?
- Razmotrite tri LAN-a međusobno povezana pomoću dva ruteru, kao što je prikazano na sledećoj slici.
  - Precrtajte sliku, tako da uključi adaptere.
  - Dodelite IP adrese svim interfejsima. Za LAN 1 koristite adrese u obliku 111.111.111.xxx; za LAN 2 koristite adrese u obliku 122.222.222.xxx; a za LAN 3 koristite adrese u obliku 133.133.133.xxx.
  - Dodelite MAC adrese svim adapterima.
  - Razmotrite slanje IP datagrama od računara A do računara R. Prepostavite da su sve ARP tabele ažurne. Nabrojte sve postupke kao sto je to uradeno za primer sa jednim ruterom u odeljku 5.4.2.
  - Ponovite (d), sada prepostavljajući daje ARP tabela u računaru pošiljaocu prazna (a da su druge tabele ažurne).



10. Setite se da kod protokola CSMA/CD, adapter čeka  $\kappa = 512$  vremena trajanja jednog bita posle kolizije, gde je  $\alpha$  slučajan broj. Za  $\kappa = 100$ , koliko dugo adapter čeka do vraćanja na postupak 2 za Ethernet od 10 Mb/s? Za Ethernet od 100 Mb/s?
11. Prepostavite da su Čvorovi A i B na istom segmentu Etherneta od 10 Mb/s i daje kašnjenje usled propagacije između dva čvora 225 vremena trajanja jednog bita. Prepostavite da čvor A počinje prenošenje okvira i da, pre nego što on završi, čvor B počinje prenošenje okvira. Može li A da završi prenošenje pre nego što otkrije da je B prenosi? Zašto može, ili zašto ne može? Ako je odgovor da, onda A pogrešno veruje da je njegov okvir uspešno prenesen bez kolizija. (Savet: Prepostavite da u trenutku  $t = 0$  vremena trajanja jednog bita, A počinje prenošenje okvira. U najgorem slučaju, A prenosi okvir minimalne veličine od  $512 + 64$  vremena trajanja jednog bita. Tako bi A završio sa prenošenjem okvira u  $t = 512 + 64$  vremena trajanja jednog bita. Prema tome, odgovor je ne, ako signal čvora B stigne do A pre  $t = 512 + 64$  vremena trajanja jednog bita. U najgorem slučaju, kada signal Čvora B stiže do A?)
12. Prepostavite da su čvorovi A i B na istom segmentu Etherneta od 10 Mb/s i da je kašnjenje usled propagacije između dva čvora 225 vremena trajanja jednog bita. Prepostavite da A i B šalju okvire u isto vreme, da okviri dolaze u koliziju i A i B biraju različite vrednosti  $\kappa$  u algoritmu CSMA/CD. Prepostavljajući da nema drugih aktivnih čvorova, može li doći do kolizije ponovnih prenosa iz čvorova A i B? Za naše potrebe, dovoljno je da se obradi sledeći primer. Prepostavite da A i B počinju prenos u  $t = 0$  vremena trajanja jednog bita. Oba čvora otkrivaju koliziju u  $t = 225$  vremena trajanja jednog bita. Oni završavaju prenošenje signala zastoja u  $t = 225 + 48 = 273$  vremena trajanja jednog bita. Prepostavite da su KA = 0 i KB = 1. U koje vreme B planira svoj ponovni prenos? U koje vreme A počinje prenos? (Obratite pažnju: Čvorovi moraju da čekaju na slobodan kanal posle vraćanja na postupak 2 - pogledajte protokol.) U koje vreme signal čvora A stiže do B? Da li B odustaje od prenošenja u svoje planirano vreme? .
13. Razmotrite Ethernet IEEE 802.3BaseT od 100 Mb/s. Koliko bi trebalo da bude maksimalno rastojanje između Čvora i haba da bi se postigla efikasnost od 0,50? Prepostavite daje dužina okvira 64 bajta i da nema habova. Da li maksimalno rastojanje takođe obezbeđuje da će predajni čvor A biti sposoban da otkrije da li je bilo koji drugi čvor prenosi za vreme prenosa čvora A? Zašto da, ili zašto ne? Kako se važe maksimalno rastojanje poredi sa aktuelnim standardom za 100 Mb/s?
14. U ovom problemu izveštete efikasnost protokola sa višestrukim pristupom sličnom CSMA/CD. U ovom protokolu, vreme je izdeljeno na odsečke i svi adapteri su sinhronizovani te odsečke. Međutim, za razliku od ALOHA sa odsečcima, dužina odsečka (u sekundama) je mnogo manja od vremena okvira (vremena za prenos okvira). Neka je  $S$  dužina odsečka. Prepostavite da su svi okviri konstantne dužine  $L = kRS$ , gde je  $R$  brzina prenosa kanala, a  $k$  veliki ceo

broj. Prepostavite da postoji A' čvorova, svaki sa beskonačnim brojem okvira za slanje. Takođe prepostavljamo daje  $t_{pmsf} < S$ , tako da svi čvorovi mogu da otkriju koliziju pre završetka vremenskog odsečka. Protokol je sledeći:

- ♦ Ako, u datom odsečku, nijedan čvor ne poseduje kanal, svi čvorovi se bore za kanal; posebno, svaki čvor šalje u vremenskom odsečku sa verovatnoćom<sup>a</sup>. Ako tačno jedan čvor šalje u odsečku, on preuzima vlasništvo nad kanalom za sledećih  $k - 1$  odsečaka i šalje svoj okvir u celosti.
- ♦ Ako neki čvor poseduje kanal, svi drugi čvorovi odustaju od prenošenja dok čvor koji poseduje kanal ne završi prenošenje svog okvira. Jednom kadaje taj čvor preneo svoj okvir, svi čvorovi se bore za kanal.

Zapazite da kanal naizmenično menja dva stanja: „proizvodno stanje”, koje traje tačno  $k$  odsečaka i „neproizvodno stanje”, koje traje slučajan broj odsečaka. Jasno, efikasnost kanala je odnos  $k/(k + x)$ , gde je  $x$  očekivani broj susednih neproizvodnih odsečaka,

- a. Za fiksno A' podredite efikasnost ovog protokola.
- b. Za fiksno A', odredite  $p$  koje daje maksimalnu vrednost efikasnosti.
- c. Koristeći (koje je u funkciji od  $A_0$  pronadeno u delu pod b, odredite efikasnost kada  $N$  teži beskonačnosti.
- d. Pokažite da se ta efikasnost približava 1 kada dužina okvira postaje velika.

15. Prepostavite da su dva čvora, A i B, priključena na suprotnim krajevima kabla od 900 m i da svaki od njih ima okvir od 1000 bitova (uključujući sva zaglavila i preamble) da pošalje onom drugom. Oba čvora pokušavaju da prenose u trenutku  $t = 0$ . Prepostavite da postoje četiri haba između A i B, od kojih svaki unosi 20-bitno kašnjenje. Prepostavite daje brzina prenosa 10 Mb/s i da se koristi CSMA/CD sa intervalima odstupanja u vidu umnožaka od 512 bitova. Posle prve kolizije, A uzima  $\kappa = 0$  a B uzima  $\kappa = 1$  u protokolu eksponencijalnog odstupanja. Zanemarite signal zastoja i kašnjenje u trajanju 96 bita.
  - a. Koliko je kašnjenje jednosmerni propagacije (uključujući kašnjenja habova) između A i B u sekundama? Prepostavite daje brzina propagacije signala  $2 \times 10^8$  m/s.
  - b. U koje vreme (u sekundama) je ceo paket čvora A isporučen čvoru B?
  - c. Sada prepostavite da samo A ima paket za slanje i da su habovi zamenjeni komutatorima. Prepostavite da svaki komutator ima 20-bitno kašnjenje obrade pored kašnjenja smeštanja i prosleđivanja. U koje vreme u sekundama je paket Čvora A isporučen čvoru B?
16. Setite se da ATM koristi 53-bajtne pakete koji se sastoje od pet bajtova zaglavila i 48 bajtova tovara. Pedeset tri bajta je neuobičajeno malo za pakete fiksne dužine; većina protokola umrežavanja (IP, Ethernet, Frame Relay itd.) koriste pakete koji su, u prošeku, značajno veći. Jedan od nedostataka male veličine paketa je da veliki deo propusnog opsegna linka troše dodatni bajtovi zaglavljaja;

u slučaju ATM-a, skoro 10 procenata propusnog opsega je „uzaludno potrošeno“ na ATM zaglavlj. U ovom problemu ispitujemo zastoje odabrana mala veličina paketa. Za sada, prepostavite da se ATM ćelija sastoji od  $P$  bajtova (možda različito od 48) i 5 bajtova zaglavlj.

- Razmotrite slanje digitalno kodovanog govornog signala direktno preko ATM-a. Prepostavite daje izvor kodovan konstantnom brzinom od 64 Kb/s. Prepostavite da je svaka ćelija potpuno popunjena pre nego što je izvor pošalje u mrežu. Vreme koje je potrebno za popunjavanje ćelije je kašnjenje **pripreme** paketa (*packetization delay*). Izraženo pomoću  $L$ , odredite kašnjenje pripreme paketa u milisekundama.
  - Kašnjenja pripreme paketa veća od 20 ms mogu da prouzrokuju primetan i neugodan eho. Odredite kašnjenje pripreme paketa za  $L \sim 1500$  bajtova (što približno odgovara Ethernet paketu maksimalne veličine) i za  $L = 48$  (što odgovara ATM ćeliji).
  - Proračunajte kašnjenje memorisanja i prosleđivanja na jednom ATM komutatoru za brzinu linka od  $R = 155$  Mb/s (popularna brzina linka za ATM) za  $i = 1500$  bajtova i za  $L = 48$  bajtova.
  - Komentarišite prednosti upotrebe male veličine ćelije.
17. Razmotrite MPLS mrežu prikazanu na slici 5.42 i prepostavite da su ruteri R5 i R6 sada sposobljeni za MPLS. Prepostavite da želimo da uradimo inženjeringu saobraćaja, tako da paketi iz R5 namenjeni za A budu komutirani ka A preko R6-R4-R3-R1, a paketi iz R5 namenjeni za B budu komutirani ka B preko R5-R4-R2-R1. Prikažite MPLS tabele u R5 i R6, kao i promenjenu tabelu u R4, koje bi to omogućile.

- Iskoristite Web da biste pronašli brojeve protokola korišćenih u Ethernet okviru za IP datagram i za ARP paket.
- Pročitajte reference [Xiao 2000, Huang 2002 i RFC 3346] o inženjeringu saobraćaja korišćenjem MPLS. Napravite listu ciljeva za inženjeringu saobraćaja. Koji od ovih ciljeva mogu da se ispune samo sa MPLS, a koji mogu da se ispune korišćenjem postojećih protokola (bez MPLS)? U poslednjem slučaju, koje prednosti nudi MPLS?

## Ethereal Lab

U pretećoj veb lokaciji za ovu <http://www.awl.com/kurose-ross>, pronaći ćete **NETFLIX** web lokaciju za ovu <http://www.netflix.com>. Pokušajte da pronađete IBBN 802.3 i format Ethernet okvira. Drugi istražuje upotrebu protokola DHCP koji smo proučili u odeljku 5.4.3.

## Teze za diskusiju

Preporučujemo vam da prokrstarite Webom pre nego što odgovorite na sledeća pitanja.

- Koliko približno iznosi trenutni opseg cena adaptera 10/100 Mb/s? Adaptera Gigabit Ethernet? Kakve su te cene u poređenju sa modemom 56 kb/s ili ADSL modemom?
- Cene habova i komutatora se često izražavaju brojem interfejsa (koji se u LAN žargonu takođe zovu i portovi). Koja je približna trenutna cena po interfejsu haba od 10 Mb/s? Haba od 100 Mb/s? Komutatora koji se sastoji samo od interfejsa za 10 Mb/s? Komutatora koji se sastoji samo od interfejsa za 100 Mb/s?
- Mnoge od funkcija adaptera mogu da se izvedu u softveru koji radi na centralnom procesoru čvora. Koje su prednosti i nedostaci premeštanja te funkcionalnosti iz adaptera u čvor?

## Sajmon S. Lem

Sajmon S. Lem je profesor i Šef katedre za računarske nauke na Teksaškom univerzitetu u Ostinu. Od 1971. do 1974. godine, bio je u ARF Centru za mrežna merenja (NeKvork Measurement Center) na UCLA, gde je radio na satelitskom i radio komutiranju paketa. Vodio je istraživačku grupu koja je pronašla bezbedne sokete i 1993. godine napravila prototip prvog bezbednog sloja soketa

(nazvanog Secure Network Programming). Njegova istraživačka interesovanja su u oblasti projektovanja i analize mrežnih protokola i bezbednosnih usluga. Diplomirao je elektrotehniku na Državnom univerzitetu u državi Vašington, o akademske stepene magistra i doktora nauka ostvario je na UCLA.



### Zašto ste odlučili da se specijalizujete u oblasti umrežavanja?

Kada sam stigao na UCLA kao novi diplomirani student u jesen 1969. godine, imao sam nameru da studiram teoriju upravljanja. Onda sam pohađao časove iz teorije čekanja u redovima koje je držao Leonard Klajnrok i on me je veoma impresionirao. Posle kratkog vremena radio sam na adaptivnom upravljanju sistema čekanja u redovima, kao mogućoj temi za tezu. Početkom 1972. godine, Lari Roberts je otpeo projekat ARPANET Satellite Svstem (koji je kasnije nazvan Packet Satellite). Profesor Klajnrok me je zamolio da se pridružim projektu. Prva stvar koju smo uradili bila je da uvedemo jednostavan, a ipak realističan algoritam odstupanja u protokol Aloha sa vremenskim odsećcima. Ubrzo posle toga, pronašao sam mnogo zanimljivih problema za istraživanje, kao sto je problem Alohine nestabilnosti i potreba za adaptivnim odstupanjem, koji bi formirali srce moje teze.

Bili ste aktivni u prvim donima Interneta u 1970-im godinama, počinjući svoje studentske dane na UCLA. Kako je tada bilo? Da li su ljudi imali ikakav nagoveštaj o tome što će Internet postati?

Atmosfera zaista nije bila različita od drugih projekata izgradnje sistema koje sam imao priliku da vidim u industriji i u akademskim krugovima. Prvobitno postavljeni cilj ARPANET-a bio je prilično skroman, odnosno, da se obezbedi pristup skupim računarima sa udaljenih lokacija tako da mnogo više naučnika može da ih koristi. Međutim, sa početkom projekta Packet Satellite 1972. godine i projekta Packet Radio 1973. godine, cilj ARPA-e se značajno proširio. Do 1973. godine, ARPA je gradila tri različite paketske mreže u isto vreme i postalo je potrebno da Vint Cerf i Bob Kahn razviju strategiju međusobnog povezivanja.

U to doba, svi ti napredni razvoji u umrežavanju bili su viđeni (verujem) kao logični, a ne kao magični. Niko nije mogao da predviđa obim Interneta i snagu današnjih personalnih računara. To je bilo deset godina pre pojave prvih PC-ja. Da bi sagledali stvari, većina studenata su podnosiли svoje programe u vidu hrpe bušenih kartica za paketnu obradu. Samo neki studenti su imali direktni pristup računarima, koji su se obično nalazili u ograničenom području. Modemi su bili spori i još uvek retkost. Kao diplomirani student, imao sam samo telefon na stolu i koristio sam olovku i papir da uradim veći deo svog posla.

### Gde ste videli da se u budućnosti usmerava oblast umrežavanja i Interneta?

U prošlosti, jednostavnost Intemetovog IP protokola bila je njegova najveća snaga u slamanju konkurenčije i postajanju de facto standarda za međusobno umrežavanje. Za razliku od konkurenčije, kao sto je bio X.25 u 1980-im i ATM u 1990-im godinama, IP može da radi iznad bilo koje tehnologije umrežavanja sloja veze podataka, zato što on nudi samo najbolju moguću uslugu datagrama. Zato svaka paketska mreža može da se poveže sa Internetom.

Na nesreću, IP-ova najveća snaga sada je njegov nedostatak. IP liči na tesan kaput koji ograničava razvoj Interneta samo na specifične pravce. IP sloj je ekonomski suviše važan da bi se popravljao podrškom novim funkcionalnostima, kao što su višestruko usmeravanje i kvalitet usluge. Poslednjih godina, mnogi istraživači su preusmerili svoje napore na aplikacijski i transportni sloj za višestruko usmeravanje i kvalitet usluge. Većina drugih trenutnih tema za istraživanje Interneta, kao što su bezbednost i P2P sistemi, obuhvataju samo aplikacijski sloj. Postoji takođe dosta istraživanja bežičnih i ad hoc mreža, senzorskih i satelitskih mreža. Na te mreže može da se gleda kao na samostalne sisteme ili sisteme sloja veze podataka, koji mogu da cvetaju zato što su izvan tesnog IP kapula.

Mnogi ljudi su uzbudeni zbog mogućnosti P2P sistema kao platforme za nove primene Interneta. Međutim, P2P sistemi su veoma neefikasni u svojoj upotrebi Internet resursa. Brinem se da li će kapacitet prenosa i komutiranja jezgra Interneta nastaviti da se povećava brže od saobraćajnih zahteva na Internetu, kako on raste da bi međusobno povezao sve vrste uređaja i podržao buduće aplikacije osposobljene za P2P. Bez značajno većeg obezbeđenja kapaciteta, osiguranje stabilnosti mreže u prisustvu zlonamernih napada i zagušenja, bio bi glavni zadatak.

### Koji je najizazovniji deo vašeg posla?

Najizazovniji deo mog posla kao profesora je podučavanje i motivisanje *svakog* studenta u mojoj klasi i *svakog* doktoranta pod mojim nadzorom, a ne samo onih koji postižu visoke rezultate. Veoma inteligentni i motivisani mogu tražiti malo vođenja, ali ne mnogo više od toga. Ja često učim više od tih studenata nego što oni uče od mene. Obrazovanje i motivisanje onih koji postižu slabije rezultate predstavlja glavni izazov.

### Koje uticaje predviđate da će tehnologija imati na učenje u budućnosti?

Na kraju, gotovo celokupno ljudsko znanje će biti dostupno preko Interneta, koji će biti najmoćnija alatka za učenje. Ta ogromna baza znanja će imati potencijal ravнопрavnog terena za studente širom sveta. Na primer, motivisani studenti u bilo kojoj zemlji će biti u stanju da pristupaju najboljim web lokacijama za Časove, multimedijskim predavanjima i materijalima za podučavanje. Već sada je rečeno da su IEEE i ACM digitalne biblioteke ubrzale razvoj istraživača u oblasti računarskih nauka u Kini. Vremenom, Internet će prevazići sve geografske prepreke za učenje.

## Bežične i mobilne mreže

U svetu telefonije, neki smatraju proteklih 10 godina decenijom mobilne telefonije. Broj preplatnika mobilnih telefona u svetu porastao je sa 34 miliona u 1993. na preko milijarde u 2003. godini, tako daje broj mobilnih preplatnika sada veći od broja osnovnih telefonskih linija [ITU Statistics 2004]. Glavna prednost mobilnog telefona je svakome očigledna - neograničen pristup globalnoj telefonskoj mreži bilo gde i bilo kada putem lako prenosivog i laganog uređaja. Da li se pojavori laptopova, palmtop računara i PDA-ova i njihovog obećanog neograničenog pristupa globalnom Internetu sa bilo kog mesta u bilo kom trenutku spremu slična eksplozija u upotrebi bežičnih uređaja za Internet?

Bez obzira na buduće širenje bežičnih uređaja za Internet, već je jasno da bežične mreže i usluge vezane za mobilnost nisu prolazna pojava. Iz aspekta umrežavanja, izazovi ovih mreža, pogotovo u sloju veze podataka i mrežnom sloju, toliko se razlikuju od uobičajenih ožičenih računarskih mreža daje potrebno zasebno poglavlje posvećeno bežičnim i mobilnim mrežama (*ovo poglavlje*).

Poglavlje počinjemo opisom mobilnih korisnika, bežičnih linkova i mreža, kao i njihovog odnosa prema većim (obično ožičenim) mrežama sa kojima se povezuju. Moramo da razlikujemo izazove do kojih dolazi zbog *bežične* prirode komunikacionih linkova u takvim mrežama, i onih koji potiču od *mobilnosti* koju ti bežični linkovi omogućavaju. Povlačenje ove značajne granice - između bežičnog i mobilnog - omogućiće nam da bolje izolujemo, prepoznamo i savladamo ključne koncepte iz

svakog od tih područja. Obratite pažnju na to da postoje mnoga mrežna okruženja u kojima su mrežni čvorovi bežični ali nisu mobilni (na primer, bežične kućne ili kancelarijske mreže sa nepokretnim radnim stanicama i velikim monitorima) i ograničene vrste mobilnosti za koje nisu potrebni bežični linkovi (na primer, radnik koji kod kuće koristi laptop povezan žicom, isključi ga, ode na posao i priključi laptop na žičanu mrežu kompanije). Naravno, najzanimljivija su ona mrežna okruženja gde su korisnici i bežični i mobilni - na primer, scenario u kome mobilni korisnik (na primer, na zadnjem sedištu automobila) održava poziv tipa „govor preko IP-a“ i nekoliko aktivnih TCP konekcija dok juri autoputem brzinom od 160 km/h. Najzanimljivije tehničke izazove nalazimo baš tu, na preseku bežičnog i mobilnog!

Počećemo prvo sa prikazom konfiguracije u kojoj ćemo razmatrati bežične komunikacije i mobilnost - mrežu u kojoj se bežični (i eventualno mobilni) korisnici povezuju sa većom mrežnom infrastrukturom putem bežičnog linka na rubu mreže. Zatim ćemo, u odeljku 6.2, razmotriti karakteristike tog bežičnog linka. U isti odeljak smo dodali kratak uvod u CDMA (*Code Division Multiple Access*), pristupni protokol deljenih medijuma koji se često koristi u bežičnim mrežama. U odeljku 6.3 malo detaljnije ćemo razmotriti aspekte sloja veze u standardu za bežični LAN, IEEE 802.11 (Wi-Fi); ukratko ćemo pomenuti i Bluetooth. U odeljku 6.4 dajemo pregled mobilnog pristupa Internetu, uključujući nove mobilne tehnologije 3G koje omogućavaju i govor i pristup Interneta velike brzine. U odeljku 6.5 obraćemo pažnju na mobilnost usredsređujući se na probleme pronalaženja mobilnog korisnika, rutiranja prema mobilnom korisniku i „predavanju“ mobilnog korisnika koji se dinamički kreće od jedne priključne tačke na mreži do druge. U odeljcima 6.6 i 6.7 ispitaćemo kako se ove usluge za mobilnost implementiraju u standardu za mobilni IP, odnosno u GSM-u. Na kraju, u odeljku 6.8, razmotrićemo uticaj bežičnih linkova i mobilnosti na protokole transportnog sloja i na aplikacije umrežavanja.

## 6.1 Uvod

Na slici 6.1 prikazana je konfiguracija u kojoj ćemo razmatrati teme bežične komunikacije podataka i mobilnosti. Počećemo uopštenim opisom koji pokriva široku lepezu mreža uključujući i bežične LAN-ove, kao što je IEEE 802.11 i mobilne mreže, kao Stope mreža 3G; detaljnije opise konkretnih bežičnih arhitektura ostavićemo za kasnije odeljke. U bežičnoj mreži postoje sledeći elementi:

- ◆ **Bežični računari.** Kao i kod ožičenih mreža, računari su krajnji uređaji na kojima se izvršavaju aplikacije. **Bežični računar** može da bude laptop, palmtop, PDA, telefon ili stoni računar. Sami računari mogu, ali ne moraju da budu mobilni.

## KRATAK OSVRT

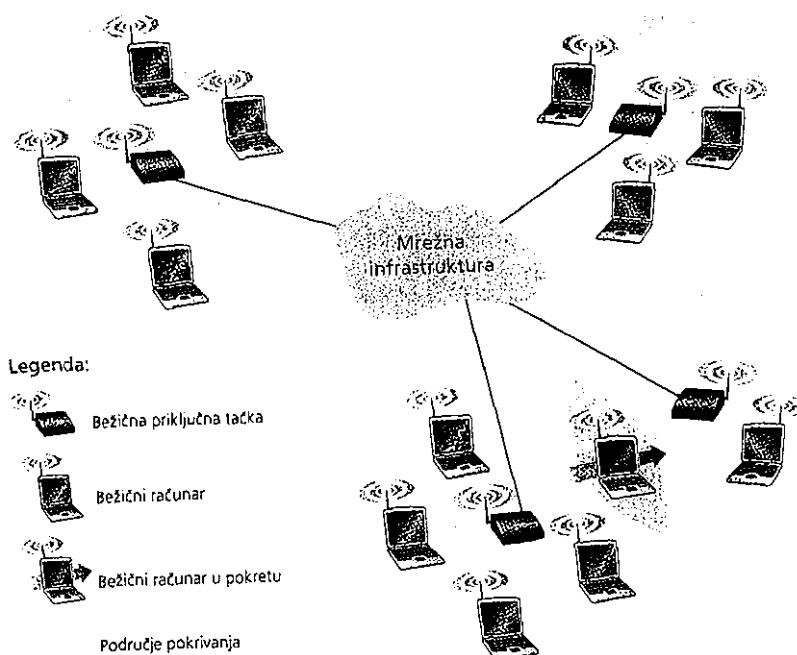
### JAVNI VVI-FI PRISTUP: USKORO STIŽE NA VAŠ ČOŠAK?

Pre samo 5 godina bežične računarske mreže bile su retkost. Mada je mnogo investirano u licence radio spektra za sisteme 3G (potražite kratak osrvt u kojem se poredi 3G mobilnost sa bežičnim LAN-ovima), 3G sistemi bili su (i još uvek su) tek u ranim fazama razvoja. U ono vreme, nekoliko prvih korisnika je počelo da isprobava upravo standardizovanu tehnologiju za bežični LAN, IEEE 802.11. Kakva pramaena za ovih 5 godina! Danas mnoge korporacije, univerziteti i kuće imaju vlastite bežične LAN-ove tipa IEEE 802.11. Sto je još značajnije naglo raste broj bežičnih priključnih tačaka - javnih lokacija na kojima korisnici mogu da dobiju bežični pristup 802.11. Grupa Gartner procenjuje da je 2003. godine postojala 71.000 javnih priključnih tačaka, sto je 50 puta više od broja koji je postojao 2001. U Sjedinjenim Američkim Državama restorani, kao što su Starbucks i McDonald, na mnogim lokacijama nude Wi-Fi pristup. U Nju Jorku, Verizon Communications je postavio tačke za Wi-Fi pristup u preko 1.000 javnih telefona i povezao javne telefone sa Internetom [Verizon 2004], i tako prolaznicima i okolnim kancelarijama omogućio Wi-Fi pristup. Početkom 2004. godine, T-Mobile [T-Mobile 2004] obezbedio je preko 4.000 javnih Wi-Fi priključnih tačaka na lokacijama kao što su aerodromi, restorani i knjižare. Novoosnovana kompanija Cometa najavila je 2003. godine da planira da postavi 20.000 komercijalnih Wi-Fi priključnih tačaka na 50 gradskih područja do 2005. godine. Uz ovakve tendencije, može se desiti da će se snovti o nesmetanom pristupu globalnom Internetu skoro na svakom mestu i u svakom trenutku ostvariti brže nego što smo mislili!

**Bežični linkovi.** Računar se povezuje sa baznom stanicom (koju ćemo definisati malo kasnije) ili sa drugim bežičnim računaram pomoću **bežičnog komunikacionog linka**. Različite tehnologije bežičnih linkova imaju različite brzine prenosa i mogu da prenose na različite razdaljine. Na slici 6.2 prikazano je nekoliko ključnih karakteristika poznatijih standarda za bežične linkove. Te ćemo standarde opisati u prvoj polovini ovog poglavlja; u odeljku 6.2 razmotrićemo takođe i druge karakteristike bežičnog linka (kao što je učestalost grešaka u bitovima i njihovi uzroci).

Na slici 6.1 bežični linkovi povezuju računare koji se nalaze na periferiji mreže u jednu veću mrežnu infrastrukturu. Naglašavamo da se bežični linkovi ponekad koriste i *unutar* mreže i za povezivanje rutera, komutatora i druge mrežne opreme. Međutim, u ovom poglavlju se usredsređujemo na upotrebu bežične komunikacije na rubovima mreže pošto se tu javlja većina najuzbudljivijih tehničkih izazova i veći deo proširenja.

**Bazna stanica.** **Bazna stanica** je ključni deo bežične mrežne infrastrukture. Za razliku od bežičnog računara i bežičnog linka, za baznu stanicu ne postoji



**Slika 6.1** ♦ Elementi bežične mreže

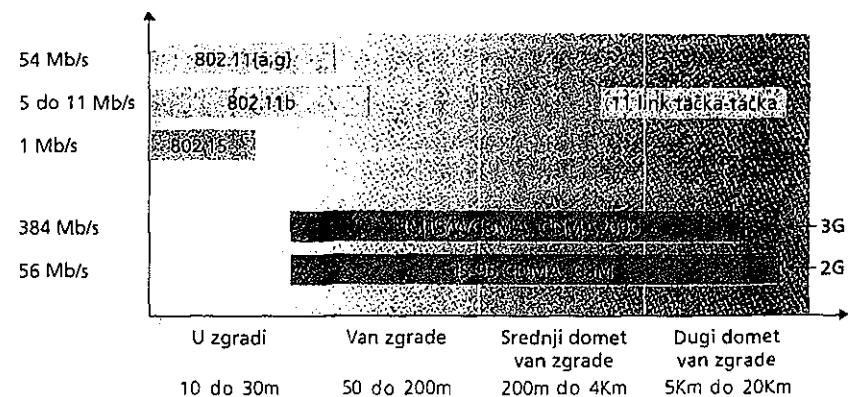
odgovarajući ekvivalent u ožičenoj mreži. Bazna stanica je zadužena za slanje i prihvatanje podataka (npr. paketa) prema bežičnom računaru pridruženom toj baznoj stanici. Bazna stanica se često zadužuje za koordinisanje prenosa za vise bežičnih računara kojima je pridružena. Kada kažemo da je bežični računar „pridružen“ jednoj baznoj stanici podrazumevamo da (1) računar se nalazi u granicama bežične komunikacije sa baznom stanicom i (2) da računar koristi tu baznu stanicu za prenošenje podataka prema većoj mreži i od nje. **Celijski tornjevi** u mobilnoj telefoniji i pristupne **tačke** u bežičnim LAN-ovima tipa 802.11 su pri-meri baznih stanica.

Na slici 6.1 bazna stanica je povezana sa većom mrežom (tj. sa Internetom, korporacijskom ili kućnom mrežom, odnosno sa telefonskom mrežom), pa tako funkcioniše kao retev sloja veze između bežičnog računara i ostatka sveta sa kojim računar komunicira.

Za računare koji su pridruženi baznoj stanici često kažemo da rade u **infrastrukturnom režimu**, pošto sve tradicionalne mrežne usluge (npr. dodeljivanje adresa i rutiranje) obezbeđuje mreža na koju je računar priključen preko bazne stanice. U *ad hoc mrežama* bežični računari nemaju takvu infrastrukturu sa kojom bi se povezali. U odsustvu te infrastrukture sami računari moraju da obezbede usluge kao sto je rutiranje, dodeljivanje adresa, prevodenje DNS imena i tome slično. U ovoj knjizi, pre svega, bavimo se mrežama u infrastrukturnom režimu.

Kada mobilni računar izade iz dometa jedne bazne stanice i uđe u domet druge, on će promeniti tačku povezivanja sa većom mrežom (tj. promeniti baznu stanicu kojoj je pridružen) - taj postupak nazivamo **predavanje**. Ovakva mobilnost otvara mnoga izazovna pitanja. Ako računar može da se kreće, kako da se pronađe njegova trenutna lokacija u mreži da bi mu se prosledili podaci? Kako se vrši adresiranje kada računar može da se nalazi na jednoj od više mogućih lokacija? Ako se računar kreće *za vreme* TCP konekcije ili telefonskog poziva, kako se podaci rutiraju, a da se konekcija ne prekine? Zbog ovih i mnogih (mnogih!) drugih pitanja bežično i mobilno umrežavanje predstavlja područje uzbudljivih istraživanja.

♦ **Mrežna infrastruktura.** Ovo je veća mreža sa kojom bežični računar može da komunicira. Sada ćemo detaljnije razmotrili tehnička pitanja koja se javljaju u bežičnim i mobilnim mrežama. Počinjemo prvo razmatranjem pojedinačnog bežičnog linka, a opisivanje mobilnosti ostavljamo za kasnije u ovom poglavljju.



**Slika 6.2** ♦ Karakteristike linkova odabranih standarda za bežične mreže

## 6.2 Bežični linkovi i mrežne karakteristike

Počećemo razmatranjem jedne jednostavne ožičene mreže, recimo kućne mreže, sa ožičenim Ethernet komutatorom (procitajte odeljak 5.6) između računara. Ako ožičeni Ethernet zamenimo bežičnom mrežom tipa 802.11 u računare bismo umesto kartica za ožičeni Ethernet stavili kartice za bežični NIC, umesto Ethernet komutatora stavili bismo pristupnu tačku, dok na mrežnom sloju kao i u višim slojevima ne bi bile potrebne praktično nikakve promene. Ovde vidimo da pažnju treba usmeriti na sloj veze ako nas zanimaju važne razlike između ožičene i bežične mreže. Zaista, između ožičenog i bežičnog linka postoji niz značajnih razlika:

- ◆ *Smanjena jačina signala.* Elektromagnetno zračenje slab je pri prolasku kroz različite materijale (npr. radio signal koji prolazi kroz zid). Čak i u slobodnom prostoru se signal raspršuje i to dovodi do smanjene jačine signala (što se ponekad naziva **gubitak usled putovanja**) sa povećanjem razdaljine između pošiljaoca i primaoca.
- ◆ *Smetnje od drugih izvora.* Radio izvori koji emituju u istom frekventnom opsegu utiču jedni na druge. Na primer, bežični telefoni sa 2,4 GHz i bežični LAN-ovi tipa 802.11b emituju u istom frekventnom opsegu. Prema tome, korisnik bežičnog LAN-a tipa 802.11b koji razgovara bežičnim telefonom sa 2,4 GHz ne može se nadati dobrim performansama ni na mreži ni na telefonu. Osim smetnji od različitih izvora emitovanja, elektromagnetni šum u okruženju (npr. motor ili mikro.talasna pećnica u blizini) takođe može dovesti do smetnji.
- ◆ *Propagiranje po višestrukim putanjama.* Propagiranje po **višestrukim putanjama** se javlja kada se delovi elektromagnetskog talasa odbiju od objekata ili od zemlje, pa stižu od pošiljaoca do primaoca putanjama različite dužine. To dovodi do zamućenja primljenog signala kod primaoca. Pokretni objekti između pošiljaoca i primaoca mogu da dovedu do toga da se propagiranje po višestrukim putanjama menja u vremenu.

Iz gornjeg opisa se naslućuje da će greške u bitovima biti češće u bežičnim linkovima nego u ožičenim. Zbog toga možda ne iznenadjuće da protokoli za bežične linkove (kao stoje protokol 802.11 koji razmatramo u sledećem odeljku) koriste ne samo moćne CRC kodove za otkrivanje grešaka već i ARQ protokole u sloju veze koji ponovo šalju oštećene okvire.

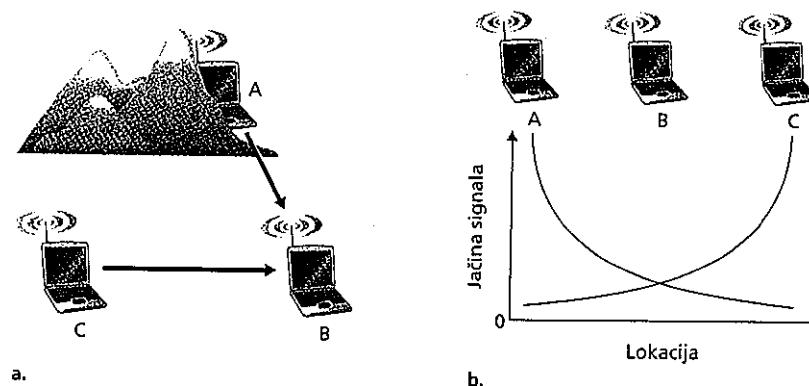
Veća i promenljiva učestalost grešaka u bitovima nije jedina razlika između ožičenog i bežičnog linka. Setite se da su u slučaju ožičenih linkova za difuzno emitovanje svi čvorovi primali prenos od svih ostalih čvorova. Kao što se vidi na slici 6.3, kada je reč o bežičnim linkovima situacija nije tako jednostavna. Prepostavimo da stanica A šalje stanici B. Prepostavimo takođe da stanica C šalje stanici B. Takozvani **problem sakrivenog terminala**, javlja se kada fizičke prepreke u okruženju (na primer, planina ili zgrada) onemoguće komunikaciju između A i C iako se na odredištu

B njihova emitovanja međusobno ometaju. Ovo je prikazano na slici 6.3(a). Drugi scenario dovodi do kolizija koje se kod primaoca teško otkrivaju a potiču od **opadanja** jačine signala prilikom kretanja kroz bežični medijum. Na slici 6.3(b) prikazan je slučaj gde su A i C tako postavljeni da njihovi signali nisu dovoljno jaki da bi se međusobno otkrili, ali jesu dovoljno jaki da jedan drugog ometaju u stanici B. Kao što ćemo videti u odeljku 6.3, višestruki pristup je u bežičnoj mreži daleko složeniji nego u ožičenoj zbog problema sakrivenog terminala i zbog opadanja.

### 6.2.1 CDMA

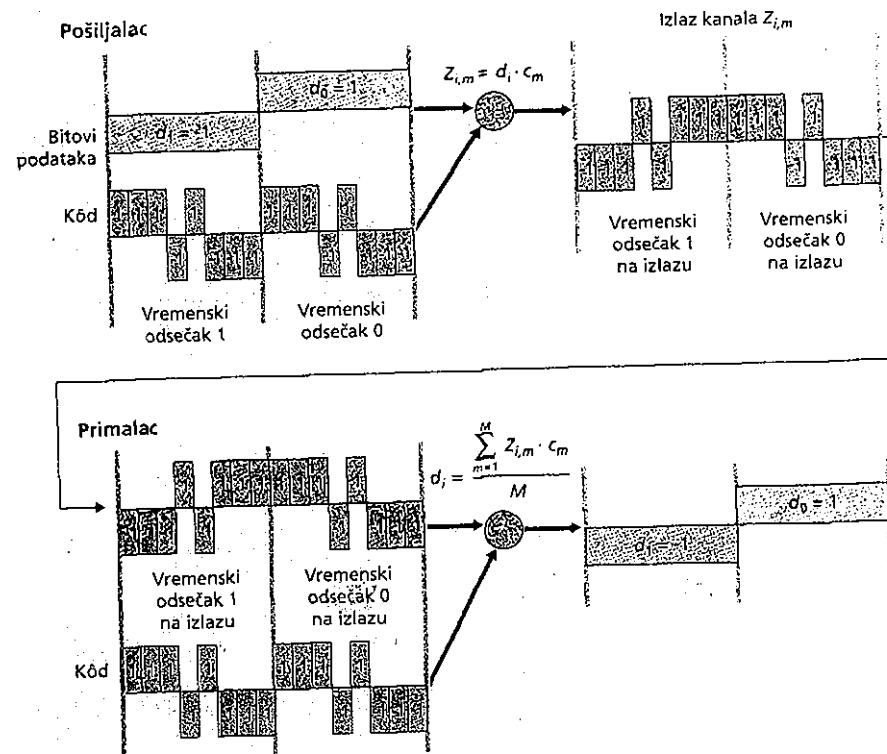
Verovalno se sećate iz poglavlja 5 daje za komunikaciju računara preko zajedničkog medijuma potreban protokol da se signali od različitih pošiljalaca ne bi međusobno ometali kod primaoca. U poglavlju 5 opisali smo tri klase protokola za pristup medi-jumima: podela kanala, slučajni pristup i redno korišćenje. CDMA (*Code Division Multiple Access*) je četvrta vrsta protokola za pristup deljenom medijumu koja preo-vlađuje u tehnologijama bežičnog LAN-a i mobilne telefonije. Pošto je CDMA tako važan u bežičnom svetu sada ćemo ga ukratko razmotriti pre nego što se u sledećim odeljcima udubimo u konkretnе tehnologije bežičnog pristupa.

U protokolu CDMA, svaki bit koji se šalje kodira se tako što se pomnoži jednim signalom (kodom) koji se menja daleko brže (**brzina seckanja**) od prvobitnog niza bitova podataka. Na slici 6.4 prikazan je jednostavan idealizovani scenario CDMA kodiranja/dekodiranja. Prepostavimo da jedinicu vremena definisi brzina kojom originalni bitovi podataka stižu do CDMA nkodera; tj. da se u jednoj jedinici vremena prenosi jedan originalni bit podataka. Neka  $d_i$  bude vrednost bita podataka



**Slika 6.3** ♦ Problem sakrivenog terminala (a) i opadanje (b)

u (-tom vremenskom odsečku. Da bi matematički bilo pogodnije, bit podataka sa vrednošću 0 prikazujemo kao -1. Svaki odsečak bitova deli se na  $M$  mini odsečaka; na slici 6.4 M je jednako 8, mada je u praksi  $M$  daleko veće. CDMA kod koju koristi pošiljalac je niz od  $M$  vrednosti,  $c_m$ ,  $m = 1, \dots, M$  od kojih je svaka +1 ili -1. U primeni na slici 6.4 pošiljalac koristi CDMA kod od  $M$  bitova (1, 1, 1, -1, 1, -1, -1, -1). Da bismo ilustrovali kako CDMA funkcioniše, razmotrićemo i-ti bit podataka,  $d_i$ . Za  $m$ -ti mini vremenski odsečak za prenos bita  $d_i$ , CDMA kod daje  $Z_{i,m}$ , a to je vrednost  $d_i$  pomnožena sa  $m$ -tim bitom dodeljenog CDMA koda,  $c_m$ :



**Slika 6.4** ♦ Primer jednostavnog protokola CDMA: pošiljalac koduje primalac dekoduje.

U jednostavnijem svem, kada ne bi bilo međusobnog ometanja pošiljalaca, primalac bi dobio kodovane bitove,  $Z_{jm}$  i ponovo napravio originalni bit podataka  $d_p$  tako što bi izračunao:

$$i^u$$

$$(6.2)$$

Čitalac bi mogao detaljno da proradi primer sa slike 6.4 i uveri se da se pomoću jednačine 6.2 kod primaoca zaista ispravno dobijaju originalni bitovi podataka.

Svet je međutim, daleko od idealnog, pa kako smo već napomenuli, CDMA mora da funkcioniše u prisustvu pošiljalaca koji se međusobno ometaju, koji koduju i prenose podatke pomoću drugačije dodeljenog koda. Ali, kako će CDMA primalac ponovo da dobije originalne bitove podataka pošiljaoca ako su ti bitovi podataka pomešani sa bitovima koje prenose drugi pošiljaoci? CDMA funkcioniše pod pretpostavkom da su pomešani preneseni signali aditivni. To znači, na primer, da ako 3 pošiljaoca šalju vrednost 1, a četvrti pošiljalac šalje vrednost -1 tokom istog mini-odsečka, tada primljeni signal kod svih primalaca tog mihi-odsečka iznosi 2 (postoje 1 + 1 + J-1 = 2). Kada imamo više pošiljalaca, pošiljalac? izračunava svoj kodovani izlaz, tačno na isti način kao u jednačini 6.1. Vrednost koju primalac dobije tokom  $m$ -toga mini-odesčka u (-tom bitu, međutim sada će biti *zbir* prenetih bitova od svih  $N$  pošiljalaca u tom mini-odsečku:

$$N$$

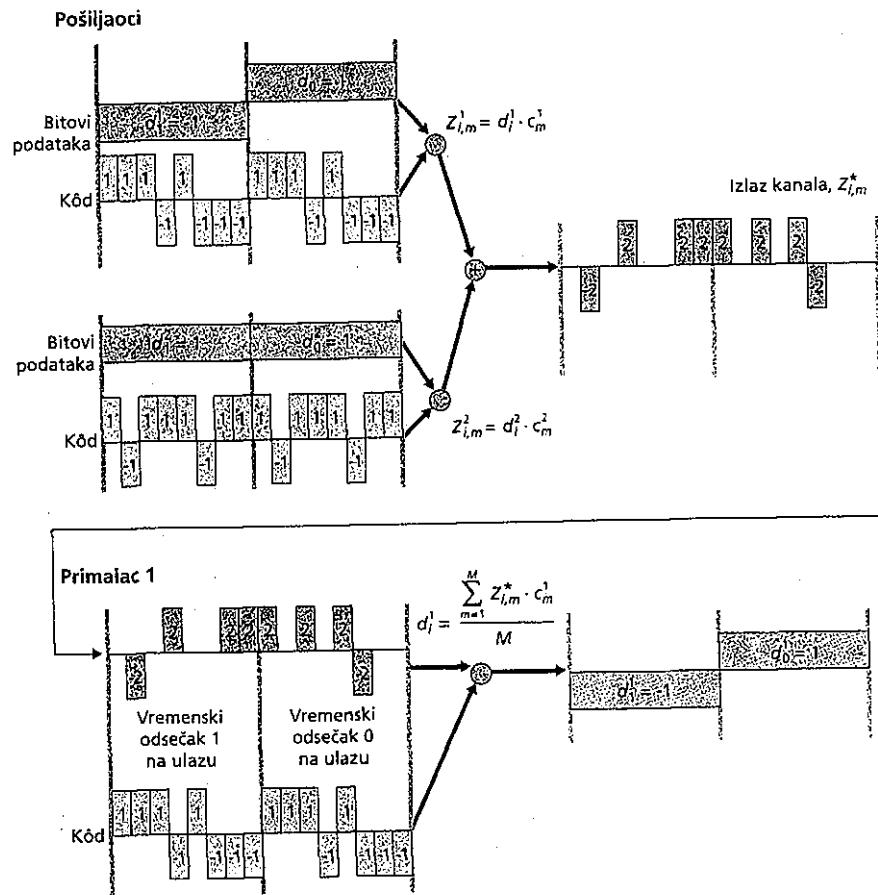
Ukoliko se pažljivo biraju kodove pošiljalaca, svaki primalac može da rekonstruiše poslate podatke datog pošiljaoca iz zbirnog signala ako jednostavno primeni kod pošiljaoca tačno na isti način kao u jednačini 6.2:

$$4 = Tjfx-$$

$$(6.3)$$

Na slici 6.5 prikazan je primer sa dva CDMA pošiljaoca. M-bitni CDMA kod gornjeg pošiljaoca je (1, 1, 1, -1, 1, -1, -1, -1) a kod donjeg (1, -1, 1, 1, 1, -1, 1, 1). Na slici 6.5 prikazan je primalac koji ponovo dobija originalne bitove podataka od gornjeg pošiljaoca. Obratite pažnju na to da primalac uspeva da izdvoji podatke pošiljaoca 1 i pored ometanja prenosom od pošiljaoca 2.

Verovatno se sećate analogije sa koktelom iz poglavlja 5. CDMA protokol bio bi sličan govoru u različitim jezicima; u takvom slučaju ljudi dosta uspešno prate razgovor na jeziku koji razumeju i filtriraju sve ostalo. Ovde vidimo daje CDMA protokol koji deli kodni prostor (za razliku od protokola sa podelom vremena ili frekvencija) i dodeljuje svakom Čvoru određeni deo kodnog prostora.



**Slika 6.5** ♦ CDMA primer sa dva pošiljaoca

Ovaj opis protokola CDMA morao je da bude kratak; u praksi je potrebno uzeti u obzir niz složenih pitanja. Prvo, da bi CDMA primaoci mogli da izdvoje signal konkretnog pošiljaoca, potrebno je pažljivo birati CDMA kodove. Drugo, u opisu smo pretpostavili da su jačine signala različitih pošiljalaca iste, a to se u stvarnosti teško postiže. Veliki broj knjiga bavi se ovim i drugim pitanjima vezanim za CDMA; detalje možete naći u knjigama [Pickholtz 1982; Viterbi 1995].

### 6.3 Wi-Fi: Bežični LAN-ovi tipa 802.11

S obzirom na to da se koriste na poslu, kod kuće, u obrazovnim ustanovama, kafe-ima, aerodromima i u uličnim govornicama, bežični LAN-ovi danas predstavljaju jednu od najvažnijih tehnologija za pristup mrežama na Internetu. Mada je tokom '90-tih godina razvijeno više tehnologija i standarda, neosporno pobeđuje jedna određena klasa standarda; bežični LAN IEEE 802.11, poznat takođe kao Wi-Fi. U ovom odeljku detaljno ćemo razmotriti bežični LAN 802.11 tako što ćemo proučiti strukturu okvira 802.11, protokol za pristup medijima 802.11 i međusobno povezivanje mreža LAN 802.11 sa ožičenim Ethernet LAN-ovima.

Za tehnologiju bežičnog LAN-a postoji nekoliko standarda 802.11. Tu između ostalog spadaju 802.11b, 802.11a i 802.11g. Tj tabeli 6.1 ukratko su prikazane glavne karakteristike ovih standarda. Dok ovo pišemo (proleće 2004) bežični LAN-ovi 802.11b daleko su najzastupljeniji. Međutim, postoji i široka ponuda proizvoda 802.11a i 802.11g, pa se u narednim godinama može očekivati značajno uvođenje ovih bežičnih LAN-ova sa većim brzinama.

Tri standarda 802.11 imaju mnoge zajedničke karakteristike. Sva tri koriste isti protokol za pristup medijumima CSMA/CA koji ćemo ubrzo opisati. Svi za okvire sloja veze koriste istu strukturu okvira. Sva tri standarda su u stanju da smanje brzinu prenosa da bi se povećala dostupna udaljenost. Svi omogućavaju „režim infrastrukture“ i „ad hoc režim“, što ćemo takođe ubrzano opisati. Međutim, kao što se vidi iz tabele 6.1 ova tri standarda sadrže neke bitne razlike u fizičkom sloju.

| Standard | Raspot frekvencija | Brzina prenosa podataka |
|----------|--------------------|-------------------------|
| 802.11b  | 2.4-2.485 GHz      | do to 11 Mbps           |
| 802.11a  | 5.1-5.8 GHz        | do to 54 Mbps           |
| 802.11g  | 2.4-2.485 GHz      | do to 54 Mbps           |

**Tabela 6.1** ♦ Pregled standarda IEEE 802.11

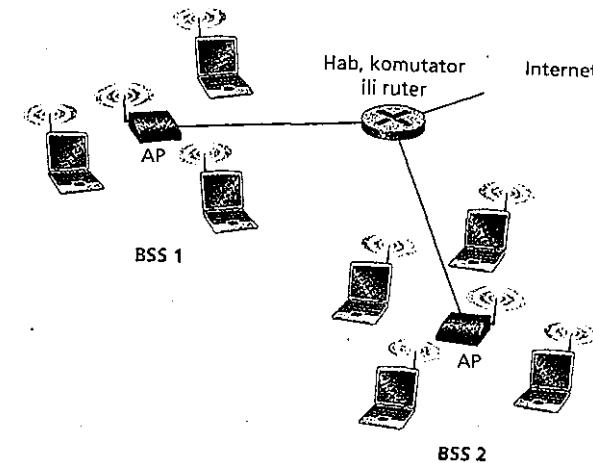
Bežični LAN 802.11b ima brzinu prenosa od  $11 \text{ Mb/s}$ . Stoji više nego dovoljno za većinu kućnih mreža sa pristupom Internetu preko širokopojasnog kabla ili DSL-a. LAN-ovi 802.11b funkcionišu u opsegu frekvencija bez licenci (2,4 do 2,485 GHz), u kojem mora da se bori sa telefonima od 2,4 GHz i mikrotalasnim pećnicama. Bežični LAN-ovi 802.11a mogu da rade na značajno većim brzinama, ali na većim frekvencijama. Međutim, pošto rade na većoj frekvenciji, LAN-ovi 802.11a imaju kraći domet prenosa za dati nivo snage i podložniji su smetnjama od propagiranja po višestrukim putanjama. Pošto LAN-ovi 802.11a funkcionišu na istom nižem frekventnom opsegu kao 802.11 b, ali pri većim brzinama prenosa kao 802.11a, trebalo bi korisnicima da omoguće „i jare i pare“.

### 63.1 Arhitektura 802.11

Na slici 6.6 prikazane su glavne komponente arhitekture bežičnog LAN-a 802.11. Osnovni element arhitekture 802.11 je BSS (*Basic Service Set*, skup osnovne usluge). BSS sadrži jednu ili više bežičnih stanica i centralnu baznu stanicu koja se u žargonu 802.31 naziva AP (*Access Point*, pristupna tačka). Na slici 6.6 prikazano je kako se pristupne tačke (AP) u svakom od dva BSS-a povezuju sa uređajem za povezivanje (hab, komutator ili ruter) koji zatim vodi do Interneta. U prosečnoj kućnoj mreži postoji jedan AP i jedan ruter (često se u istoj kutiji nalazi i kablovski ili ADSL modem) koji povezuje BSS sa Internetom.

Kao i kod Ethernet uređaja, svaka bežična stanica 802.11 ima 6-bajtnu MAC adresu koja se čuva u firmveru kartice (tj. kartice mrežnog interfejsa 802.11). Svaki AP takođe ima MAC adresu za svoj bežični interfejs. Kao i kod Interneta, ovim MAC adresama administrira IEEE i one su (teoretski) globalno jedinstvene.

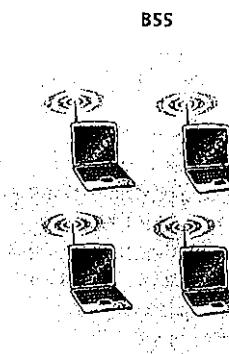
Kao stoje napomenuto u odeljku 6.1, bežični LAN-ovi sa pristupnim tačkama često se nazivaju infrastrukturni bežični LAN-ovi, gde se „infrastruktura“ sastoji od pristupnih tačaka zajedno sa ozičenom Ethernetsom infrastrukturom koja povezuje pristupne tačke i ruter. Na slici 6.7 prikazano je da stanice IEEE 802.11 mogu takođe da se grupišu u ad hoc mrežu - mrežu bez centralne kontrole i bez povezivanja sa „spoljnjim svetom“. Ovdje se mreža formira „usput“ od mobilnih uređaja koji su trenutno u blizini, a imaju potrebu za međusobnom komunikacijom, iako u blizini nema postojeće mrežne infrastrukture. Ad hoc mreža može da se napravi kada se sastanu ljudi sa laptopovima (na primer, u sali za konferencije, u vozu ili automobilu) koji žele da razmene podatke, a u blizini ne postoji centralizovana pristupna tačka. Kako se širi upotreba prenosivih uređaja za komunikaciju, javlja se izuzetno zanimanje za ad hoc umrežavanje. U ovom odeljku ćemo se ipak usredosrediti na infrastrukturne bežične LAN-ove.



**Slika 6.6** ♦ Arhitektura LAN-a IEEE 802.11

### Kanali i pridruživanje

Pod protokolom 802.11, da bi bežična stanica mogla da šalje ili prima okvire 802.11 koji sadrže podatke mrežnog sloja, ona se mora pridružiti nekoj pristupnoj tački. Mađa svi standardi 802.11 koriste pridruživanje, ovu temu ćemo konkretno opisati u kontekstu protokola IEEE 802.11 b



**Slika 6.7** ♦ Ad hoc mreža IEEE 802.11

Kada mrežni administrator instalira pristupnu tačku, on joj dodeljuje identifikacioni broj SSID (*Service Set Identifier*) od jedne ili dve reči. (Kada, na primer, u Microsoft Windowsu XP „prikaže dostupne mreže“ prikazuje se lista SSID-ova svih dostupnih pristupnih tačaka.) Administrator mora pristupnoj tački takođe da dodeli i broj kanala. Da biste shvatili brojke kanala, imajte na umu da protokol 802.11 b funkcioniše u frekventnom opsegu 2,4 GHz do 2,485 GHz. U ovom rasponu od 85 MHz, protokol 802.11b definiše 11 kanala koji se delimično preklapaju. Dva kanala se ne preklapaju ako i samo ako ih razdvaja 4 ili više kanala. Konkretno, skup kanala 1, 6 i 11 je jedini skup od tri kanala koji se ne preklapaju. Ovo znači da bi administrator mogao da napravi bežični LAN sa ukupnom maksimalnom brzinom prenosa od 33 Mb/s tako što bi na istoj fizičkoj lokaciji instalirao 3 pristupne tačke 802.11b, dodelio im kanale 1, 6 i 11 i povezao svaku pristupnu tačku sa jednim komutatorom.

Uz ovo osnovno poznавanje kanala 802.11, opisáćemo jednu zanimljivu (ne baš tako retku) situaciju - Wi-Fi džunglu. Wi-Fi džungla je svaka fizička lokacija u kojoj bežična stanica prima dovoljno jak signal od dve ili više pristupnih tačaka. Na primer, u mnogim kafeima u New Yorku bežična stanica može da uhvati signal od mnogih obližnjih pristupnih tačaka. Jedna od pristupnih tačaka bi mogla da pripada kafeu dok ostale mogu da budu u stanovima u blizini kafea. Svaka od ovih pristupnih tačaka se verovatno nalazi u drugoj podmreži sa nezavisno dodeljenim kanalom.

Pretpostavimo sada da u takvu Wi-Fi džunglu uđete sa svojim prenosivim raču-narom i naručite bežični pristup Internetu i kolač sa borovnicama. Uzmimo da u džungli ima 5 pristupnih tačaka. Da bi dobila pristup Internetu, vaša bežična stanica mora da pristupi tačno jednoj podmreži i mora da se pridruži tačno jednoj pristupnoj tački. Pridruživanje znači da bežična stanica pravi virtualnu žicu između sebe i pristupne tačke. Konkretno, samo pridružena pristupna tačka će da šalje okvire podataka (tj. okvire koji sadrže podatke, kao što su datagrami) prema vašoj bežičnoj stanicici, a vaša bežična stanica će slati okvire podataka na Internet samo preko pridružene pristupne tačke. Ali, kako će se vaša bežična stanica pridružiti konkretnoj pristupnoj tački? Još važnije, kako će vaša bežična stanica sazнати da li u džungli ima pristupnih tačaka i koje su one?

Standard 802.11 zahteva da pristupna tačka povremeno šalje okvire za navođenje koji sadrže SSID i MAC adresu pristupne tačke. Pošto zna da pristupne tačke šalju okvire za navođenje vaša bežična stanica osmatra svih 11 kanala i čeka okvir za navođenje bilo koje pristupne tačke u blizini (od kojih neke emituju na istim kanalima - prava džungla!). Kada iz okvira za navođenje utvrdi koje pristupne tačke su dostupne, vaš bežični računar bira jednu od pristupnih tačaka za pridruživanje. Kada se izabere pristupna tačka, vaš bežični računar i izabrana pristupna tačka pregovaraju koristeći protokol za pridruživanje 802.11. Ako se u tom pregovaranju sve dobro završi, vaša bežična stanica postaje pridružena izabranoj pristupnoj tački. Implicitno, tokom faze pridruživanja, vaša bežična stanica pristupa podmreži kojoj pripada izabrana pristupna tačka. Odmah nakon faze pridruživanja, bežična stanica

će preko pridružene pristupne tačke obično poslati u podmrežu poruku DHCP otkrivanja (pročitajte odeljak 5.4.3) kako bi dobila IP adresu u podmreži pristupne tačke. Od tog trenutka, vaš računar vidi ostatak Interneta jednostavno kao računar u podmreži pristupne tačke.

Da bi uspostavila pridruživanje sa konkretnom pristupnom tačkom, bežična stanica ponekad mora pristupnoj tački da dokaže svoju autentičnost. Bežični LAN-ovi 802.11 predviđaju niz alternativa za dokazivanje autentičnosti i za pristupanje. Jedan pristup, koji koriste mnoge kompanije, jeste da se dozvoli pristup bežičnoj mreži na osnovu MAC adrese određene stanice. Drugi pristup, koji se koristi u mnogim Internet kafeima upotrebljava korisnička imena i lozinke. U ova slučaja, pristupna tačka obično komunicira sa serverom za proveru autentičnosti tako što prenosi podatke između krajnje bežične stanice i servera za proveru autentičnosti koristeći protokole kao što su RADIUS [RFC 2138] ili DIAMETER [RFC 3588]. Razdvajanjem servera za proveru autentičnosti od pristupne tačke omogućava se da jedan server opsluži mnoge pristupne tačke i centralizuju se (često osetljive) odluke o autentičnosti i pristupu, pa tako troškovi i složenost pristupnih tačaka ostaju niski. U odeljku 8.8.4 videćemo da novi protokol IEEE 802.11 i koji definiše bezbednosne aspekte familije protokola 802.11 prihvata upravo ovaj pristup.

### 6.3.2 MAC protokol 802.11

Pošto se bežična stanica pridruži pristupnoj tački, ona može da počne da šalje okvire podataka pristupnoj tački i da ih prima od nje. Ali, pošto se dešava da više stanica u isto vreme pokušava da šalje okvire podataka preko istog kanala, potreban je protokol za višestruki pristup koji će koordinisati te prenose. Ovde, stanica može biti bežična stanica ili pristupna tačka. Kao što je opisano u poglavlju 5 i odeljku 6.2.1, u opštem smislu postoje Četiri klase protokola višestrukog pristupa: podela kanala, slučajni pristup, redna upotreba i CDMA. Pod uticajem velikog uspeha Etherneta i njegovog protokola direktnog pristupa, projektanti protokola 802.11 su za bežični LAN 802.11 izabrali protokol sa direktnim pristupom. Ovaj protokol se naziva CSMA sa izbegavanjem kolizija ili ukratko CSMA/CA. Kao i kod Ethernetovog CSMA/CD, „CSMA“ znači Carrier Sense Multiple Access, što znači da svaka stanica pronalazi kanal pre nego što počne da emituje i da se uzdržava od emitovanja kada oseti da je kanal zauzet. Mada i Ethernet i 802.11 koriste direktni pristup sa ispitivanjem nosioca, između ova dva MAC protokola postoje značajne razlike. Prvo, umesto otkrivanja kolizija 802.11 koristi tehnike za izbegavanje kolizija. Drugo, zbog relativno visoke učestalosti grešaka u bitovima u bežičnim kanalima, 802.11 (za razliku od Etherneta) koristi ARQ šemu u sloju veze za potvrđivanje i ponovno slanje. Sada ćemo opisati izbegavanje kolizija i šeme za potvrđivanje u sloju veze u protokolu 802.11.

Verovatno se sećate iz odeljaka 5.3 i 5.5 da u Ethernetu algoritmu za otkrivanje kolizija Ethernet stanica osluškuje kanal tokom emitovanja. Ako tokom prenosa otkrije da neka druga stanica takođe emituje, ova prekida prenos i pokušava da

šalje kasnije po isteku malog, slučajno izabranog vremenskog intervala. Za razliku od Ethernet protokola 802.3, MAC protokol 802.11 ne primenjuje otkrivanje kolizija. Za to postoje dva važna razloga:

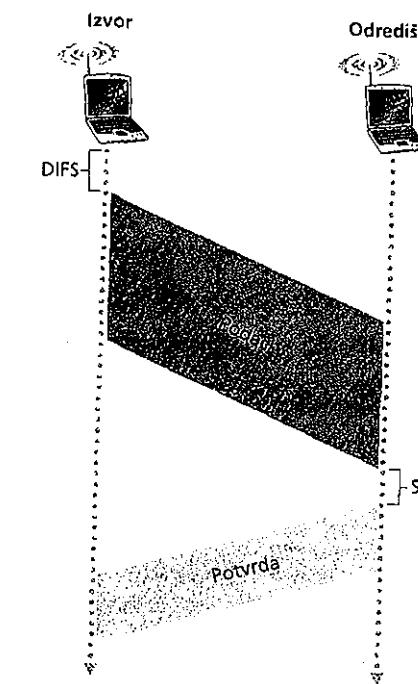
- ◆ Sposobnost da se otkriju kolizije zahteva sposobnost istovremenog slanja (vlastitog signala stанице) i primanja (da bi se utvrdilo da li još neka stаница takođe šalje). Pošto je jačina primljenog signala obično veoma mala u poređenju sa jačinom signala koji se šalje sa kartice 802.11, skupo je praviti hardver koji je u stanju da otkrije koliziju.
- ◆ Još važnije, Čak i kada bi kartica mogla istovremeno da emituje i da sluša (i eventualno prekine slanje kada otkrije opterećenost kanala), adapter ipak ne bi mogao da otkrije sve kolizije zbog problema sakrivenog terminala i opadanja kao što je opisano u odeljku 6.2.

Pošto bežični LAN-ovi 802.11 ne koriste otkrivanje kolizije, stаница koja počne sa slanjem okvira šalje okvir u potpunosti; tj. kad se stаница pokrene, nema više povratka. Logično je da slanje celih okvira (pogotovo dugачkih okvira) u situaciji sa mnogo kolizija može značajno da utiče na performanse protokola višestrukog pristupa. Da bi se smanjila verovatnoća kolizija, 802.11 koristi nekoliko tehnika za izbegavanje kolizija koje ćemo uskoro opisati.

Pre nego što predemo na razmatranje izbegavanja kolizija, moramo prvo da razmotrimo šemu potvrda u sloju veze protokola 802.11. U odeljku 6.2 je već rečeno da kada stаница na bežičnom LAN-u pošalje okvir, on može iz različitih razloga da ne stigne do odredišne stанице. Zbog ove nezanemarljive verovatnoće neuspeha MAC 802.11 koristi potvrde u sloju veze. Kao stoje prikazano na slici 6.8, kada odredišna stаница primi okvir koji uspešno prođe CRC proveru, ona Čeka jedan kratak vremenski interval poznat kao SIFS (Short Inter-frame Spacing), a zatim vraća okvir potvrde. Ako otpremna stаница u zadatom intervalu ne primi potvrdu, ona pretpostavlja da došlo do greške i ponovo šalje isti okvir tako što za pristup kanalu ponovo koristi protokol CSMA/CA. Ako potvrda ne stigne nakon određenog broja ponovnih slanja, otpremna stаница odustaje i odbacuje okvir.

Pošto smo opisali kako 802.11 koristi potvrde sloja veze, sada možemo da opišemo CSMA/CA protokol 802.11. Prepostavimo da stаница (bežična stаница ili pristupna tačka) ima okvir za slanje.

1. Ako stаница na početku vidi da je kanal slobodan, ona šalje svoj okvir nakon kratkog vremenskog intervala poznatog kao DIFS (Disistributed Interframe Space); slika 6.8.
2. Inače, stаница bira slučajnu vrednost za odstupanje i odbrojava od te vrednosti sve dok je kanal slobodan. Kada se otkrije zauzeće kanala, vrednost brojača se zamrzava.



**Slika 6.8** ♦ 802.11 koristi potvrde sloja veze

3. Kada brojač dode do 0 (obratite pažnju na to da loga jedino može doći kada je kanal slobodan), stаница šalje ceo okvir i zatim Čeka na potvrdu.
4. Ako se primi potvrda, otpremna stаница zna daje njen okvir pravilno primljen u odredišnoj stаници. Ako stаница ima još neki okvir za slanje, ona pokreće protokol CSMA/CA od tačke 2.
5. Ako se potvrda ne primi, otpremna stаница ponovo prelazi na fazu oporavka iz koraka 2 tako što slučajnu vrednost bira iz većeg intervala.

Proničljivi čitalac je možda primetio da u koraku 2 stаница bira slučajnu vrednost za odstupanje i počinje odbrojavanje, pa tako odlaže prenos čak i kada vidi da je kanal slobodan. U Ethernetovom CSMA/CD protokolu za višestruki pristup (odeljak 5.5.2)'medutim, stаница počinje sa slanjem čim otkrije da je kanal slobodan. Zbog čega se u ovom pogledu CSMA/CD i CSMA/CA tako različito ponašaju?

Da bismo odgovorili na ovo pitanje, razmotrimo scenario u kojem dve stанице imaju okvire podataka za slanje, ali nijedna od njih ne šalje odmah zbog toga što otkriva da neka treća stаница već emituje. U Ethernetovom CSMA/CD-u obe stанице

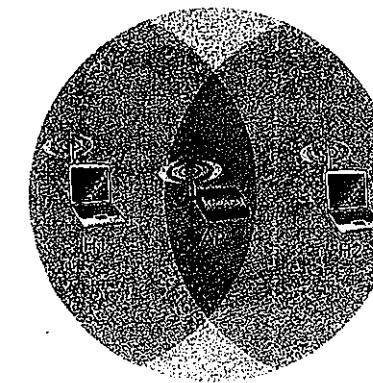
bi počele da šalju. Čim bi otkrile daje treća stanica prestala sa slanjem. To bi dovelo do kolizije što u protokolu CSMA/CD nije ozbiljan problem pošto bi obe stanice prekinule sa slanjem i tako tzbegle nepotrebno emitovanje ostatka okvira koji je pretr-peo koliziju. Međutim, u protokolu 802.11 situacija je sasvim različita. Pošto 802.11 ne prekida prenos kada otkrije koliziju, okvir koji je pretrpeo koliziju bi se preneo u celosti. U protokolu 802.11 cilj je da se kolizije izbegnu kad god je to moguće. U ovom protokolu, ako dve stanice vide daje kanal zauzet, one obe odmak pokreću odstupanje sa slučajnim vrednostima za koje se prepostavlja da će biti različite. Ako su te vrednosti zaista različite, kada se kanal oslobođi jedna od stanica će početi da šalje pre druge stanice, pa će „gubilnička stanica“ (pod prepostavkom da stanice nisu sakrivene jedna od druge) uočiti signal „pobedničke stanice“, zamrznuti svoj brojač i sačekati sa slanjem sve dok pobednička stanica ne završi svoje slanje. Na ovaj način izbegnutaje skupa kolizija. Naravno, i sa protokolom 802.11 su kolizije moguće: ako su stanice sakrivene jedna od druge ili ako su izabrale identične slučajne vrednosti za odstupanje.

#### Tretiranje sakrivenih terminala: RTS i CTS

MAC protokol 802.11 takođe sadrži izvanrednu (opcionu) šemu rezervacija koja pomaže da se izbegnu kolizije. Čak i u slučaju sakrivenih terminala. Ispitajmo ovu šemu u kontekstu slike 6.9 gde su prikazane dve bežične stanice i jedna pristupna tačka. Obe bežične stanice su u dometu pristupne tačke (čiji domet je prikazan kao osenčeni krug) i obe su se pridružile pristupnoj tački. Međutim, zbog opadanja, domet signala bežičnih stanica ograničenje na unutrašnji deo osenčenih krugova prikazanih na slici 6.9. Zbog toga, svaka bežična stanica je sakrivena od druge iako nijedna nije sakrivena od pristupne tačke.

Razmotrimo sada zašto sakriveni terminali mogu da predstavljaju problem. Prepostavimo da stanica H1 šalje okvir i da na pola tog slanja mrežni sloj u stanici H2 preda MAC protokolu 802.11 jedan okvir (koji ćemo ovde nazvati okvir DATA). Pošto ne primiče slanje iz stanice H1, H2 će prvo sačekati jedno kratko slučajno izabrano vreme, a zatim će početi da šalje okvir DATA i tako dovesti do kolizije. Kanal će prema tome biti protračen tokom celog intervala emitovanja iz H1 kao i iz H2.

Da bi se ovaj problem izbegao, protokol IEEE 802.11 omogućava stanici da jednim kratkim kontrolnim okvirom RTS (Request To Send) i jednim kratkim kontrolnim okvirom CTS (Clear To Send) rezerviše pristup kanalu. Kada pošiljalac hoće da pošalje okvir DATA, on može pristupnoj stanici prvo da pošalje okvir RTS, i naznači ukupno vreme potrebno za prenos okvira DATA i okvira potvrde (ACK). Kada pristupna stanica dobije okvir RTS, ona odgovara difuznim emitovanjem okvira CTS. Okvir CTS ima dve svrhe: on daje pošiljaocu eksplisitnu dozvolu da šalje i takođe obaveštava ostale stanice da ne šalju tokom rezervisanog perioda.



**Slika 6.9** ♦ Primer sakrivenog terminala: H1 je sakriven od H2 i obratno.

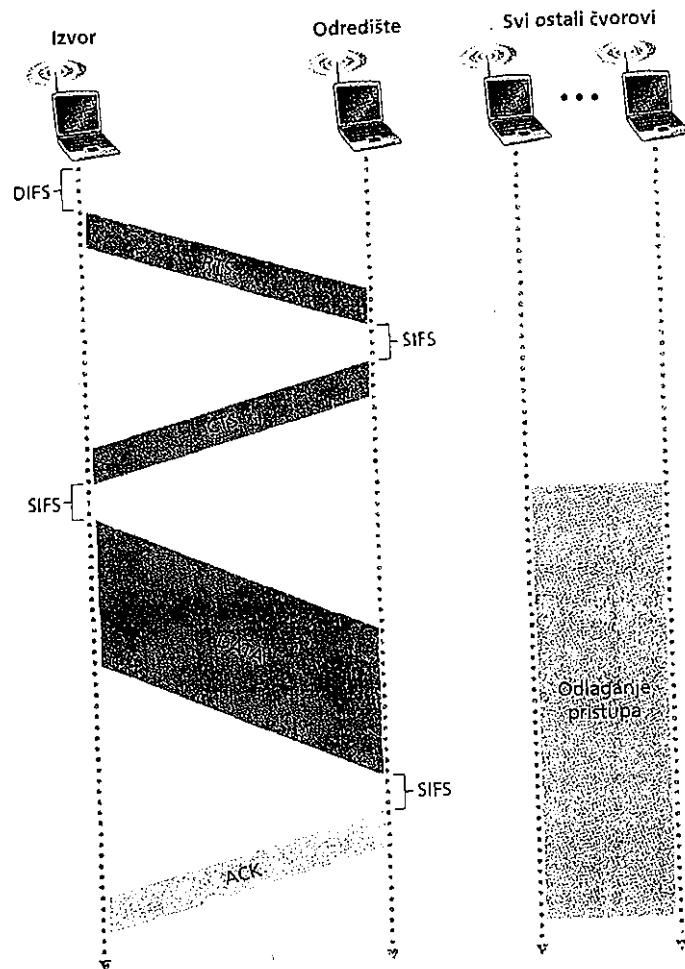
Na taj način, kako vidimo na slici 6.10, prvo nego što pošalje okvir DATA, H1 prvo difuzno emituje okvir RTS koji Čuju sve stanice u njegovom krugu uključujući pristupnu tačku AP. Pristupna tačka zatim odgovara svojim okvirom CTS koji Čuju sve stanice unutar dometa, uključujući H1 i H2. Stanica H2, postoje Čula CTS, uzdržava se od slanja u intervalu navedenom u okviru CTS. Na slici 6.10 prikazani su okviri RTS, CTS, DATA i ACK.

Korišćenje okvira RTS i CTS poboljšava performanse na dva značajna načina:

- ◆ Problem sakrivenih stanica se ublažava pošto se dugački okvir DATA šalje tek postoje kana! rezervisan.
- ◆ Posto su okviri RTS i CTS kratki, kolizija u kojoj učestvuje RTS ili CTS okvir trajaće samo koliko traje takav kratak okvir. Pošto se okviri RTS i CTS ispravno prenesu, naredni okviri DATA i ACK trebalo bi da prođu bez kolizije.

Pogledajte aplet 802.11 na veb lokaciji ovog udžbenika. Ovaj interaktivni aplet ilustruje protokol CSMA/CA uključujući i razmenu RTS/CTS.

Mada razmena RTS/CTS može da pomogne u smanjenju broja kolizija, ona takođe dovodi do kašnjenja i troši resurse kanala. Iz tog razloga se razmena RTS/CTS koristi jedino da bi se rezervisao kanal za prenos dugačkog okvira DATA. TJ praksi, svaka bežična stanica može da uspostavi RTS prag, pa da se paketi RTS/



Slika 6.10 ♦ Izbegavanje kolizije pomoću okvira RTS i CTS



### Korištenje protokola 802.11 u linku od tačke do tačke

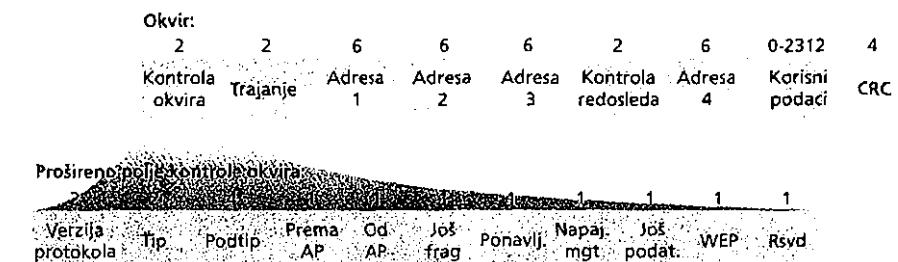
Do sada smo opisivali 802.11 u situaciji višestrukog pristupa. Trebalo bi da pomenemo da ako dva čvora imaju usmerive antene, oni mogu da usmere te antene jednu prema drugoj i da izvršavaju protokol 802.11 preko linka koji je u suštini od tačke do tačke. Zbog jeftinog hardvera za 802.11, korišćenja usmerivih antena i sve veće jačine prenosa, 802.11 može da se koristi kao jeftin način za bežične konekcije od tačke do tačke na razdaljinama od nekoliko desetina kilometara. U knjizi [Bhagvatt 2003] opisana je takva bežična mreža sa više skokova koja funkcioniše u seoskim područjima oko reke Gang u Indiji sa linkovima 802.11 od tačke do tačke.

### Okvir IEEE 802.11

Mada okvir 802.11 ima mnogo sličnosti sa Ethernetovim okvirovima, on takođe sadrži niz polja specifičnih za njegove bežične linkove. Okvir 802.11 prikazan je na slici 6.11. Brojevi iznad polja u okviru predstavljaju dužinu tih polja u bajtovima; brojevi iznad podpolja u kontrolnom polju okvira predstavljaju dužine tih podpolja u bitovima. Ispitajmo sada polja u okviru, kao i neke od značajnijih podpolja u kontrolnom polju okvira.

### Polja korisnog terera i CRC

Glavno polje u okviru je polje korisnog tereta koje obično sadrži jedan IP datagram ili ARP paket. Mada je dozvoljeno da polje bude dugačko 2312 bajtova, ono je obično manje od 1500 bajtova i sadrži jedan IP datagram ili ARP paket. Kao i Ethernet okviri, okviri 802.11 sadrže CRC (Cyclic Redundancy Check) tako da primalač može da otkrije greške u bitovima primljenog okvira. Kao što smo videli, greške u bitovima su mnogo češće u bežičnim LAN-ovima nego u ozičenim, pa je CRC ovde još korisniji.



Slika 6.11 ♦ Okvir 802.11

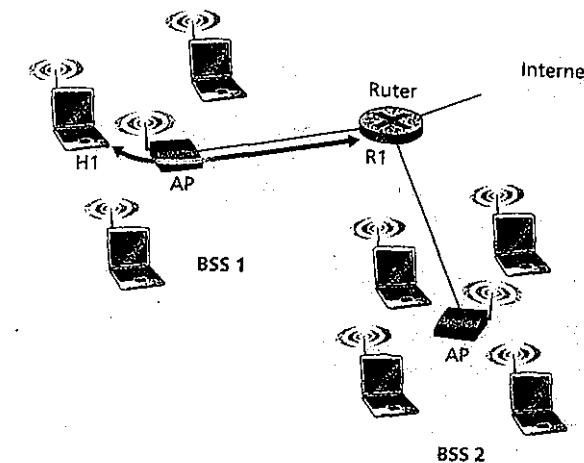
### Polja adresa

Možda je najupadljivija novost u okviru 802.11 to što postoje *četiri* adresna polja od kojih svako može da primi 6-bajtnu MAC adresu. Ali zašto četiri adresna polja? Zar nije dovoljno MAC polje izvora i MAC polje odredišta kao kod Etherneta? Ispada da su za međusobno povezivanje mreža potrebna tri adresna polja - konkretno, za prenos datagrama mrežnog sloja iz bežične stанице kroz pristupnu tačku u interfejs ruteru. Četvrto adresno polje koristi se u ad hoc mrežama, ali ne i u infrastrukturnim mrežama. Pošto ovde razmatramo samo infrastrukturne mreže, obratićemo pažnju na prva tri adresna polja. Standard 802.11 definiše ova polja na sledeći način:

- ◆ Adresa 2 je MAC adresa stанице koja šalje okvir. Prema tome, ako bežična stаница šalje okvir, u adresno polje 2 stavlja se MAC adresa te stанице. Slično tome, ako pristupna tačka šalje okvir, u adresno polje 2 stavlja se MAC adresa pristupne tačke.
- ◆ Adresa 1 je MAC adresa bežične stанице koja treba da primi okvir. Prema tome, ako okvir šalje mobilna bežična stаницa, polje adrese 1 sadrži MAC adresu odredišne pristupne tačke. Slično tome, ako okvir šalje pristupna tačka, polje adrese 1 sadrži MAC adresu odredišne bežične stанице.
- ◆ Da biste shvatili adresu 3, setite se da BSS (koji se sastoji od pristupne tačke i bežičnih stаницa) čini deo podmreže i da se ova podmreža povezuje sa drugim podmrežama preko nekog ruterskog interfejsa, Adresa 3 sadrži MAC adresu ovog interfejsa na ruteru.

Da bismo bolje shvatili svrhu adrese 3, razmotrimo primer međumrežnog povezivanja u kontekstu slike 6.12. Na ovoj slici postoje dve pristupne tačke od kojih je svaka zadužena za niz bežičnih stаницa. Svaka od pristupnih tačaka ima direktnu vezu sa ruterom koji se zatim povezuje sa globalnim Internetom. Trebalo bi imati na umu da je pristupna tačka uredaj sloja veze, pa ni jedna od njih „ne govori“ IP niti razume IP adresu. Razmotrite sada prenos datagrama od ruterskog interfejsa R1 do bežične stанице H1. Ruter nije svestan da između njega i stanicu H1 postoji pristupna tačka AP; iz perspektive ruteru, H1 je prosti računar u jednoj od podmreža sa kojima je on (ruter) spojen.

- ◆ Ruter, koji zna IP adresu stанице H1 (na osnovu adrese odredišta u datagramu), koristi ARP da bi utvrdio MAC adresu H1, kao u običnom Ethernet LAN-u. Kada pribavi MAC adresu stанице H1, ruterski interfejs R1 enkapsulira datagram u Ethernet okvir. Polje izvorne adrese u ovom okviru sadrži MAC adresu interfejsa R1, a polje adrese odredišta sadrži MAC adresu stанице H1.



**Slika 6.12** ◆ Korишћење adresnih polja u okvirima 802.11: prenos okvira između H1 i R1

- ◆ Kada Ethernet okvir stigne u pristupnu tačku AP, AP konvertuje Ethernet okvir 802.3 u okvir 802.11 pre nego što ga pošalje u bežični kanal. AP popunjava prvu adresu MAC adresnog računara H1, a adresu 2 vlastitom MAC adresom, kao sto je gore opisano. U polje za treću adresu AP ubacuje MAC adresu ruteru R1. Na taj način, H1 će (iz adrese 3) saznati MAC adresu ruterskog interfejsa koji je poslao datagram u podmrežu.

Razmotrimo sada šta se događa kada bežična stаница H1 odgovori slanjem datagrama iz H1 u R1.

- ◆ H1 pravi okvir 802.11 i popunjava adresu 1 MAC adresom pristupne tačke, a polje adrese 2 MAC adresom stанице H1, kao sto je gore opisano. U polje adrese 3 H1 unosi MAC adresu ruteru R1.
- ◆ Kada AP primi okvir, 802.11 konvertuje ga u Ethernet okvir. Polje adrese izvora u tom okviru biće MAC adresa stанице H1, a polje adrese odredišta biće MAC adresa ruteru R1. Prema tome, polje adrese 3 omogućava pristupnoj tački da odredi odgovarajuću MAC adresu odredišta da bi napravila Ethernet okvir.

Da rezimiramo, adresa 3 igra ključnu ulogu za međusobno umrežavanje bazne stanice sa ozičenim LAN-om.

#### Polja rednog broja, trajanja i kontrole okvira

Verovatno se sećate da u standardu 802.11 stanica uvek vraća potvrdu kad god pravilno primi okvir od druge stанице. Pošto potvrde mogu da se izgube, otpremna stаница bi mogla da posalje više primeraka datog okvira. Kao što smo videli u opisu protokola rdt2.1 (odeljak 3.4.1) redni brojevi omogućavaju primaocu da razlikuje novi okvir od nekog ponovo poslatog starog okvira. Polje rednog broja u okviru 802.11 prema tome ima istu svrhu ovde u sloju veze kao što je imalo u transportnom sloju u poglavljiju 3.

Imajte na umu da protokol 802.11 dozvoljava otpremnoj stанице da rezerviše kanal za određen vremenski period dovoljan da se pošalje okvir podataka i da se primi potvrda. Ta vrednost se stavlja u polje trajanja (kako u okvir podataka tako i u okvire RTS i CTS).

Kao što je prikazano na slici 6.11 kontrolno polje okvira ima više delova. Reći ćemo samo nekoliko reči o nekim najvažnijim delovima; potpun opis možete naći u specifikaciji 802.11 [Heid 2001; Crow 1997; IEEE 802.11 1999]. *Polja tip i podtip* služe da bi se okviri podataka razlikovali od pridruženih okvira RTS, CTS i ACK. *Polja prema i od* se koriste da bi se odredilo značenje različitih polja za adrese (Ova značenja se menjaju zavisno od toga da li se koristi ad hoc režim ili režim infrastrukture, a u slučaju da se koristi režim infrastrukture, značenje zavisi od toga da li okvir šalje bežična stаница ili pristupna tačka.) Na kraju, polje WEP označava da li se koristi šifrovanje. (WEP se opisuje u poglavljju 8.)

#### 6.3.4 Mobilnost unutar iste IP podmreže

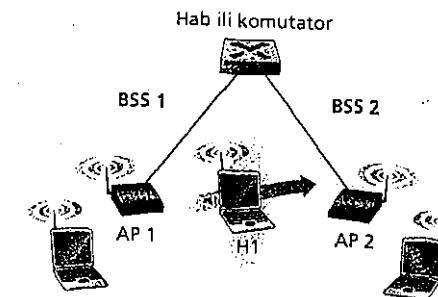
Da bi se povećao fizički domet bežičnog LAN-a, kompanije i univerziteti postavljaju više baznih stаница unutar iste IP podmreže. Ovo, naravno, otvara pitanje mobilnosti između baznih stаница — kako da se bežične stанице neprimetno prebacuju sa jedne na drugu baznu stanicu ne prekidajući postojeće TCP konekcije? Kao što ćemo videti u ovom odeljku, kada bazne stанице pripadaju istoj podmreži mobilnost se relativno jednostavno omogućava. Kada se stанице kreću po različitim podmrežama, biće potrebni složeniji protokoli za upravljanje koje ćemo proučiti u odeljcima 6.5 i 6.6.

Pogledajmo sada konkretni primer mobilnosti između baznih stаница u istoj podmreži. Na slici 6.13 prikazane su dve međusobno povezane bazne stанице i jedan računar, H1 koji se kreće između baznih stаница BSS1 i BSS2. Pošto u ovom primeru uređaj koji međusobno povezuje bazne stанице nije ruter, sve stанице iz ovih baznih stаница, uključujući i pristupne tačke AP pripadaju istoj IP podmreži. Prema tome, kada H1 pređe sa BSS1 na BSS2 on može da zadrži svoju IP adresu i sve TCP

konekcije koje su u toku. Da je uređaj za međusobno povezivanje bio ruter, tada bi H1 morao ili da promeni IP adresu, pa da prekine TCP konekcije koje su u toku ili da upotrebi protokol za mobilnost iz mrežnog sloja kakav je mobilni IP kao što je opisano u odeljku 6.6.

Ali, što se konkretno dešava kada H1 pređe sa BSS1 na BSS2? Kako se H1 udaljava od pristupne tačke AP 1, on otkriva da signal pristupne tačke slab je počinje da traži neki jači signal. H1 prima okvire za navođenje od pristupne tačke AP2 (koja će u mnogim korporacijskim i univerzitetskim okruženjima imati isti SSID kao AP1). H1 zatim raskida pridruživanje sa pristupnom tačkom AP 1 i pridružuje se tački AP2 zadržavajući istu IP adresu i ne prekidajući TCP konekcije koje su bile u toku.

Ovo sve funkcioniše kako treba ako je uređaj za međusobno povezivanje hab. Ali, ako je taj uređaj komutator - što je često slučaj - potrebno je obratiti posebnu pažnju. Kao što se možda sećate iz poglavljaja 5, komutatori „sami uče“ i automatski grade svoje tabele za prosleđivanje. Ta njihova sposobnost je korisna prilikom povremenih premeštanja (na primer, kad nekog zaposlenog premeste iz jednog odeljenja u drugo); međutim, komutatori nisu projektovani za podršku veoma pokretnih korisnika koji žele da održe TCP konekcije dok se premeštaju između baznih stаница. Da biste shvatili problem koji st. ovde javlja, imajte na umu da pre premeštanja komutator u tabeli prosleđivanja ima stavku u kojoj se MAC adresa stанице H1 uparuje sa izlaznim interfejsom komutatora preko kojeg se može pronaći H1. Ako se H1 na početku nalazi u baznoj stanci BSS1, tada će se datagram čije je odrediste H1 uputiti preko pristupne tačke AP1. Međutim, kada se H1 pridruži BSS2, njegove okvire treba uputiti prema pristupnoj tački AP2. Jedno rešenje (u suštini, doskočica) je da AP2 odmah nakon novog pridruživanja pošalje komutatoru difuzno emitovani



Slika 6.13 ♦ Mobilnost u istoj podmreži

Ethernet okvir sa adresom izvora H1. Kada komutator primi taj okvir, on ažurira svoju tabelu prosledivanja i tako dozvoljava da se do H1 stigne preko AP2. U grupi standarda 802.1 lf razvija se protokol za komunikaciju među pristupnim tačkama koji se bavi ovim i sličnim pitanjima.

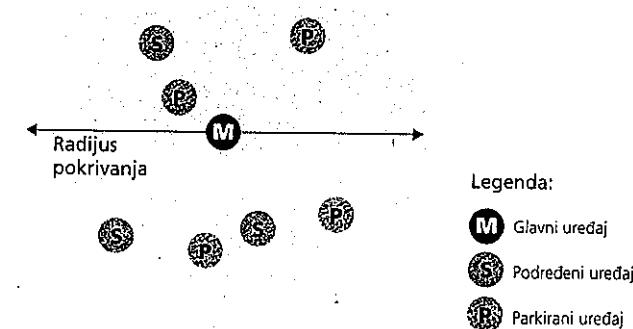
### 6.3.5 802.15 i Bluetooth

Kao što se vidi na slici 6.2, Wi-Fi standard IEEE 802.11 namenjen je komunikaciji među uređajima udaljenim do 100 metara dok pristup mreži preko mobilne telefonije ima domet na desetine kilometara. Pre nego što se upustimo u detalje ovih bežičnih mreža sa većim udaljenostima, pogledajmo ukratko standard 802.15 WPAN (*Wireless Personal Area Network*) koji je u familiji IEEE 802 srođan sa 802.11, a namenjen je povezivanju među ličnim uređajima na rastojanju od približno 10 metara. 802.15 je u suštini tehnologija male snage, kratkog dometa i malih brzina, „zamena za kabl“ za međusobno povezivanje notbuka, perifernih uređaja, mobilnih telefona i PDA-a dok je 802.11 „pristupna“ tehnologija veće snage, srednjeg dometa i veće brzine.

Mreža IEEE 802.15 funkcioniše na kratkom rastojanju, manjom snagom i manje košta. Slojevi veze i fizički sloj se kod 802.15 zasnivaju na prethodnoj specifikaciji Bluetooth za lične mreže [Held 2001, Bisdikian 2001]. Mreže 802.15 funkcionišu u radio opsegu 2,4 GHz za koji nisu potrebne licence, u vremenskom multipleksu i sa vremenskim odsećcima od 624 mikro sekunde. U svakom vremenskom odsečku pošiljalac emituje najednom od 79 kanala tako što se kanal menja na poznat, ali pseudo slučajan način od odsečka do odsečka. Taj oblik preskakanja kanala, poznat kao FHSS (*Frequency Hopping Spread Spectrum*) emituje sa podelom vremena po frekventnom spektru. 802.15 omogućuje brzine podataka do 721 Kb/s.

Mreže 802.15 su ad hoc mreže: za povezivanje uređaja 802.15 nije potrebna nikakva mrežna infrastruktura (npr. pristupna tačka). Prema tome, uređaji 802.15 moraju sami da se organizuju. Uredaji 802.15 prvo se organizuju u piko mrežu od najviše osam aktivnih uređaja kao stoje prikazano na slici 6.14. Jedan od ovih uređaja postaje glavni, a ostali uređaji su mu podređeni. Glavni čvor zaista upravlja piko mrežom - njegov sat određuje vreme u piko mreži, on može da emituje u svakom neparnom odsečku dok podređeni može da emituje samo pošto je glavni čvor sa njim komunicirao u prethodnom odsečku, pa čak i tada on može da emituje samo prema glavnom čvoru. Pored podređenih uređaja u mreži može biti i maksimalno 255 parkiranih uređaja. Ovi uređaji ne mogu da komuniciraju dok im glavni čvor ne promeni status iz parkiranog u aktivni.

Citaoci koje zanima WAPN i njegov 802.15 naći će više informacija u referencama za Bluetooth [Held 2001, Bisdikian 2001] ili na zvaničnoj veb lokaciji IEEE 802.15 [IEEE 802.15 2004].



**Slika 6.14 ♦ Piko mreža 802.15**

## 6.4 Celularni pristup Internetu

U prethodnom odeljku ispitali smo kako računar može da pristupi Internetu kada se nađe unutar Wi-Fi lokacije za aktiviranje tj. kada se nalazi u blizini pristupne tačke 802.11. Ali, većina Wi-Fi lokacija za aktiviranje ima mali domet, između 10 i 100 metara u prečniku. Šta da radimo kada imamo očajnicu potrebu za bežičnim pristupom Internetu, a nije nam dostupna Wi-Fi lokacija za aktiviranje? S obzirom na to daje mobilna telefonija sada sveprisutna na mnogim područjima u celom svetu, prirodna strategija bi bila da se celularne mreže prošire tako da podrže ne samo govornu telefoniju već i bežični pristup Internetu. Idealno bi bilo da taj pristup Internetu bude relativno brz i da obezbedi nesmetanu mobilnost tako da korisnici ne prekidaju TCP konekcije za vreme putovanja, na primer, autobusom ili vozom. Pod uslovom da uzvodne i nizvodne brzine budu dovoljne, korisnik bi čak mogao da održi video konferencijsku sesiju dok se kreće. Taj scenario nije posebno nerealan. Dok ovo pišem (proleće 2004) mnogi operateri mobilne telefonije nude pretplatnicima celularni pristup Internetu za manje od 100 dolara mesečno sa uobičajenim uzvodnim i nizvodnim brzinama od nekoliko stotina kilobita u sekundi.

U ovom odeljku dajemo kratak pregled tehnologija za pristup Internetu koje trenutno postoje ili se upravo pojavljuju. Fokus će nam biti na bežičnom prvom skoku između mobilnog telefona i ožičene infrastrukture telefonske mreže; u odeljku 6.7 razmotrićemo kako se pozivi rutiraju korisniku koji se kreće između baznih stanica. Ovako kratak opis može sadržati samo uproščeni okvirni opis celularnih tehnologija. Moderne celularne komunikacije, naravno, mnogo su složenije, pa mnogi uni-

## KRATAK OSVRT

### POREĐENJE BEŽIČNIH LAN-OVA SA CELULARnim MOBILnim TELEFONIMA 3G

Mnogi operateri mobilne telefonije uvode celularne mobilne 3G sisteme sa unutrašnjim brzinama od 2 Mb/s, a spoljnir od 384 kb/s. 3G sistemi se uvode u opsezima radio frekvencija gde su potrebne licence tako da neki operateri plaćaju vladama čak 2 hiljade dolara po preplatniku za licence. 3G sistemi će korisnicima omogućiti pristup Internetu sa udaljenih spoljnih lokacija dok su u pokrelu na način sličan današnjem pristupu mobilnih telefona. Na primer, tehnologija 3G dozvoliće korisniku da pristupi informacijama o putnim kartama dok vozi kola ili informacijama o repertoaru bioskopa dok se sunča na plazi. Ipak, mnogi ekspertri se danas pitaju da li će tehnologija 3G biti uspešna s obzirom na troškove i na konkureniju tehnologije bežičnog LAN-a [VWeinstein 2002]. Posebno, ovi ekspertri tvrde:

- ◆ Nova infrastruktura bežičnih LAN-ova poslaje skoro sveprisutna. Kao što je napomenuto u prethodnom kratkom osrvtu, sve više ima bežičnih LAN-ova IEEE 802.11 koji funkcionišu brzinom od 1-1 Mb/s, pa i višom (javni Wi-Fi pristup). Uskoro će skoro svi prenosivi računari i PDA-ovi imati fabrički ugrađene LAN kartice 802.11. Štaviše, novi uređaji za Internet - kao što su bežične kamere i elektronski bežični ramovi za slike - takođe će koristiti male kartice za bežični LAN slabe snage.
- ◆ Većina bežičnog saobraćaja podataka polaziće ili će se završavati u lokalnim okruženjima. Pod pretpostavkom da se bežični saobraćaj koji polazi iz tržnih centara, poslovnih zgrada itd. ili se *završava u njima* prenosi jeftinim bežičnim LAN-ovima, za skupe sisteme 3G ostaće relativno mala količina saobraćaja.
- ◆ Bazne stanice bežičnog LAN-a moguće bi da opsluže i uređaje mobilnih telefona. To bi omogućilo jeftin pristup podataka za cellularne mobilne uređaje kao i za uređaje za bežični LAN. Prema tome, mobilni telefon koji nema karticu za bežični LAN, ali se nalazi u lokalnom okruženju moći će da zaobiđe 3G sisteme za pristup Internetu.

Naravno, mnogi drugi ekspertri veruju da će tehnologija 3G biti ne samo veoma uspešna, već da će takođe dramatično promeniti način našeg rada i života. Naravno, i Wi-Fi i 3G bi mogli da postanu preovladjujuće bežične tehnologije, pa bi bežični uređaji u romingu automatski birali pristupnu tehnologiju koja na trenutnoj fizičkoj lokaciji pruža najbolju uslugu (pročitajte u ovom odeljku opis bežičnog pristupa 4G).

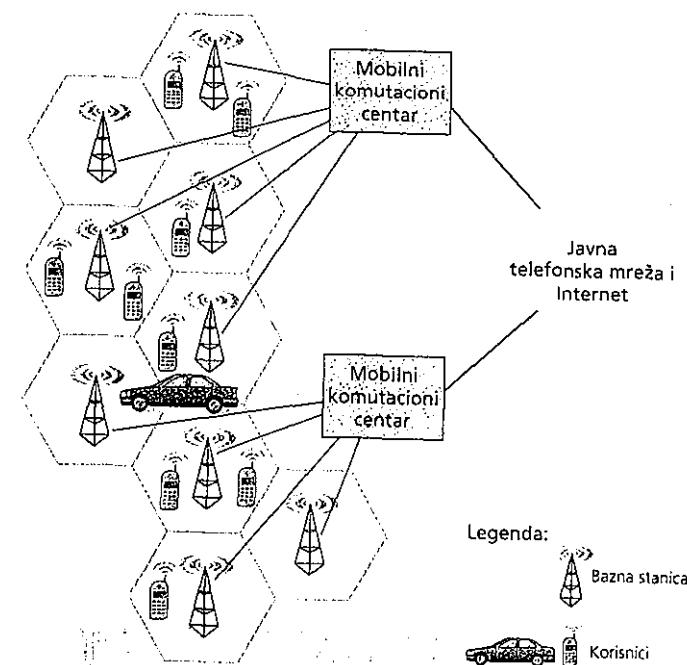
verziteti nude više predmeta na tu temu. Čitaoci koji žele dublje poznavanje upućuju se na [Goodman 1997; Scourias 1997; Korhonen 2003; Kaaranen 2001; Lin 2001] kao t na posebno dobru i iscrpnju referencu [Mouly 1992].

### 6-4.1 Opšti pregled cellularne arhitekture

Izraz *celularni* odnosi se na činjenicu da se geografsko područje deli na određen broj geografskih područja pokrivanja, koje nazivamo celijama, kao što je prikazano na levoj strani slike 6.1S. Svaka celija sadrži jednu baznu stanicu koja emituje signale i prima signale od mobilnih stanica u svojoj celiji. Područje pokrivanja jedne celije zavisi od mnogih faktora uključujući emisionu snagu bazne stanice, emisionu snagu mobilne stanice, od građevina koje zaklanjavaju prostor u celiji i visinu antene bazne stanice. Mada na slici 6.15 svaka celija sadrži jednu baznu stanicu koja se nalazi na sredini celije, mnogi današnji sistemi postavljaju bazne stanice na uglove gde se sekutri tri celije tako da jedna bazna stanica sa usmerenim antenama može da opsluži tri celije.

#### Osnovna mrežna arhitektura

Kao što se vidi na slici 6.15, svaka bazna stanica povezana je sa regionalnom mrežom - kao što je javna telefonska mreža - ili direktno sa Internetom preko ožičene



**šilka 6.15** ♦ Komponente arhitekture cellularne mreže

infrastrukture. Konkretno, na slici 6.15 vidi se daje svaka bazna stanica povezana sa mobilnim komutacionim centrom MSC (*Mobile Switching Center*) koji upravlja uspostavljanjem i raskidanjem poziva mobilnih korisnika. MSC sadrži dobar deo funkcionalnosti koji postoji u običnom telefonskom komutacionom centru (kakav je PBX ili centrala), ali ima dodatnu funkcionalnost potrebnu za rad sa mobilnim korisnicima.

#### Pristupne tehnike vazdušnog interfejsa

Uobičajena je situacija da se u datoj celiji istovremeno obavlja više poziva. Ovi pozivi moraju da dele raspon radio spektra koji je dodeljen posredniku za celularne usluge. Većina celularnih sistema danas koristi jedan od dva opšta pristupa za delje-nje radio spektra:

- ◆ *Kombinaciju frekventnog (FDM) i vremenskog (TDM) muiitiplexiranja.* Verovatno se sećate iz poglavlja 1 da se kod frekventnog muiitiplexiranja kanal deli na određen broj frekventnih opsega tako da se svaki opseg namenjuje jednom pozivu. Takođe smo u poglavlju 1 videli da se kod čistog vremenskog muiitiplexiranja vreme deli na okvire i da se svaki okvir dalje deli na odsečke, pa svaki poziv dobija na korišćenje određeni odsečak u pokretnom okviru. U sistemima sa kombinacijom FDM/TDM kanal se deli na niz frekventnih podopsegova; u svakom podopsegu se vreme deli na okvire i odsečke. Prema tome, u sistemu sa kombinacijom FDM/TDM, ako je kanal podeљen na  $F$  podokvira, a vreme na  $T$  odsečaka, tada će kanal biti u stanju da podrži  $F \times T$  istovremenih poziva.
- ◆ *Višestruki poziv sa kodnom podealom (CDMA).* Sigurno se sećate iz odeljka 6.2.1 da CDMA ne deli ni frekvencije niti vreme. Umesto toga, svi korisnici dele istu radio frekvenciju u isto vreme. Svaki korisnik u celiji dobija zaseban niz bitova koji nazivamo niz za seckanje. Kao što smo videli u odeljku 6.2.1, kada pošiljalac i primalac koriste isti niz za seckanje, primalac može iz istovremenih slanja svih pošiljalaca da rekonstruiše ono što je poslao njegov pošiljalac. Glavna prednost sistema CDMA je u tome što eliminiše potrebu da se dodeljuju frekvencije. Kada se koristi FDM/TDM, na primaocu utiče interferencija od drugih signala u istom frekventnom opsegu. Prema tome, u sistemu FDM/TDM se dala frekvencija može ponovo koristiti samo u celijama koje su prostorno dovoljno udaljene tako da ne bude interferencije. Prilikom projektovanja CDMA sistema takvo ponovno korišćenje frekvencija ne predstavlja značajno pitanje.

#### 6.4.2 Celularni standardi i tehnologije: kratak pregled

Kada ljudi govore o celijamim tehnologijama, obično ih klasificuju na nekoliko „generacija“. Starije generacije su bile projektovane pre svega za govorni saobraćaj; noviji celularni sistemi osim govora podržavaju i pristup Internetu. Postoje ovo

knjiga o umrežavanju računara, a ne o govornoj telefoniji, nas, naravno, više zanimaju novije generacije cellularnih sistema. Ali, pošto su novije generacije direktno nastale od starijih, ipak počinjemo pregled kratkim opisom bežičnih sistema prve i druge generacije.

Pri opisivanju ovih generacija naići ćete na niz izraza koji se koriste u različitim tehnologijama. Kao čitalac, nemate obavezu da potpuno prihvati i zapamtite sve ove izraze i skraćenice. Svrha ovog pregleda jeste da vam pruži uvid u postojanje različitih generacija bežičnih sistema i da posluži kao brzi podsetnik za različite izraze i skraćenice.

Sistemi prve generacije (IG) su bili analogni FDMA sistemi projektovani samo za govorne komunikacije. Ovi sistemi su sada skoro nestali, a zamenili su ih digitalni sistemi 2G.

#### Druga generacija (2G)

Sistemi druge generacije, iako su digitalni takođe suprojektovani za govornu komunikaciju. Ali, pošto su današnji sistemi 2,5G i 3G, projektovani da podrži i prenos podataka, nastali od sistema 2G, važno je reći nekoliko reči o sistemima 2G. Mobilni telefon 2G konvertuje analogni govorni signal u digitalni format pre nego što ga modulira i zatim pošalje u etar. Digitalna tehnologija 2G ima mnoge prednosti nad analognom tehnologijom IG a to su povećani kapacitet usluga unutar celije, imapredena bezbednost od zloupotreba i naprednije usluge kao što su identifikacija pozivaoca i poruke. Većina današnjih mobilnih operatera koristi tehnologiju 2G. U širokoj upotrebi su različiti 2G standardi i tehnologije:

- ◆ *IS-136 (Interim Standard 135) TDMA.* Ovo je kombinovani FDM/TDM sistem koji potiče iz FDMA tehnologije 1G. On je u širokoj upotrebi u Sevemoj Americi.
- ◆ *GSM (Global System for Mobile Communications).* Tokom '80-tih godina Evropljani su uvideli potrebu za digitalnim sistemom koji će važiti svuda u Evropi i zameiti njihove nekompatibilne sisteme IG i omogućiti neprimetnu mobilnost medu zemljama kao i mogućnosti i usluge koje nisu moguće u analognim sistemima. Ova potreba dovela je do standarda GSM za celularne komunikacije. Evropljani su veoma uspešno uveli GSM tehnologiju početkom '90-tih. GSM se zatim proširio na Aziju i Sevemu Ameriku, i sada predstavlja standard za celu-lamu komunikaciju sa najširom primenom. GSM standard za celularne sisteme 2G za vazdušni interfejs koristi kombinovani FDM/TDM. GSM sistemi se sastoje od opsega frekvencija od po 200 kHz gde svaki opseg podržava 8 TDM poziva. GSM kodira govor na 13 kb/s i 12,2 kb/s.
- ◆ *IS-95 CDMA.* Za razliku od sistema IS-136 i GSM, koji koriste FDM/TDM, sistem IS-95 CDMA koristi višestruki pristup sa kodnom podealom (odeljak 6.2.1). Kompanija Qualcomm demonstrirala je izvodljivost sistema CDMA za

mobilnu telefoniju krajem '90-tih godina; od tad je uvedeno više sistema IS-95, pogotovo u Sjevernoj Americi i Koreji.

#### Prelaz sa druge generacije na treću (2,5G)

2G sistemi kao što su IS-95, GSM i IS-136 optimizirani su za govorne usluge i nisu pogodni za prenos podataka. Tokom '90-tih organizacije za standarde utvrdile su da postoji potreba za celularnu tehnologiju 3G koja bi bila pogodna i za govor i za prenos podataka (uključujući pristup Internetu). Međutim, pošto uvođenje tehnologije 3G traje godinama, kompanije su razvile privremene protokole i standarde koji omogućavaju prenos podataka preko postojeće infrastrukture 2G. Takvi sistemi nose zajedničko ime „celularni sistemi 2,5G“. Navodimo neke od njih:

- ◆ **GPRS** (*General Packet Radio Service*) je razvijen iz GSM-a. Za rad sa podacima, GSM igra ulogu modema između korisničkog uređaja i odredišne mreže podataka-tj, GSM koristi komutiranje vodova kako za podatke tako i za govorni saobraćaj. Kao što smo naučili u poglavljiju 1 komutiranje vodova je krajnje neefikasno za povremeni prenos podataka. Štaviše, standard GSM podržava brzine prenosa do 9,6 kb/s što je nedopustivo sporo za bilo šta osim za čisti tekst. GPRS predstavlja privremeno rešenje koje omogućava efikasnije usluge za podatke sa paketima pri većim brzinama (obično u rasponu od 40 do 60 kb/s). GPRS usluga pruža se na osnovu postojeće GSM mreže. Međutim, za razliku od čistog GSM-a, mobilna GPRS stanica može na zahtev da koristi više vremenskih odsečaka u datom kanalu. Kada se koristi GPRS, određen broj odsečaka se rezerviše za prenos podataka i dinamički se dodeljuje mobilnim stanicama zavisno od njihove trenutne potražnje.
- ◆ **EDGE** (*Enhanced Data Rates for Global Evolution*). Glavni cilj sistema EDGE jeste da se povećaju moguće brzine u mreži GSM/GPRS, tj. da se bolje iskoristi GSM kanal od 200 kHz sa okvirima od po 8 TDMA odsečaka. To se postiže pre svega zamenom modulacione Šeme GSM-a jednom snažnjom Šemom. Teoretski, EDGE može korisnicima da obezbedi 384 kb/s za prenos podataka. Izvrstan pregled sistema EDGE možete naći u knjizi [Ericsson 2004].
- ◆ **CDMA2000, faza 1.** Ova tehnologija 2,5G nastala je od IS-95. Ona može da obezbedi usluge za pakete podataka do 144,4 kb/s i priprema teren za uvođenje 3G sistema CDMA2000, faza 2.

#### Treća generacija (3G)<sup>1</sup>

Celularni sistemi treće generacije potrebni su za pružanje telefonske usluge, kao i za prenos podataka pri osetno većim brzinama od odgovarajućih sistema druge generacije. Od sistema treće generacije se pre svega očekuje da omoguće:

- ◆ 144 kb/s za korisnike u vožnji;
- ◆ 384 kb/s za upotrebu van zgrade u mirovanju ili pri brzini hoda;
- ◆ 2 Mb/s za upotrebu u zgradama.

U području treće generacije postoje dva glavna standarda koja se trenutno bore za prevlast:

- ◆ **UMTS** (*Universal Mobile Telecommunications Service*) je proširenji GSM sa podrškom za mogućnosti treće generacije. Arhitektura UMTS mreže većim delom nasleđuje već uspostavljenu arhitekturu GSM mreže. Međutim, radio pristup UMTS-a značajno se razlikuje od šeme FDMA/TDMA koja se koristi za GSM. Konkretno, UMTS koristi CDMA tehniku poznatu pod imenom DS-WCDMA (Direct Sequence Wideband CDMA). UMTS se upravo uvodi u velikom delu Evrope što ne iznenaduje s obzirom na to da UMTS potiče od GSM-a.
- ◆ **CDMA-2000** je razvijen od 2G sistema IS-95 i kompatibilan je unazad sa sistemom IS-95. Kako bi se i očekivalo na osnovu imena, on u svom vazdušnom interfejsu takođe koristi CDMA. CDMA-2000 se uvodi u Sjevernoj Americi i u delovima Azije.

#### Četvrta generacija (4G)

Pošto smo razmotrili i tehnologiju bežičnog LAN-a i niz tehnologija za celularni prisnisp, vratimo se malo i razmotrimo Šta bismo mi kao korisnici u idealnoj situaciji očekivali od bežičnog pristupa Internetu. Ovde je lista želja:

- ◆ Želeli bismo sveprisutni bežični pristup Internetu. Hteli bismo da se omogući pristup Internetu bez obzira na to da li smo kod kuće, u kancelariji, u kolima, u kafiću ili na plaži.
- ◆ Zavisno od naše fizičke lokacije i brzine kojom se krećemo, želeli bismo najbrži mogući pristup Internetu. Na primer, ako se nalazimo na u/ici gde je dostupan 802.11b od 11 Mb/s i 3G pristup od 384 kb/s želeli bismo da naš sistem automatski izabere 802.11b, tj. onaj sistem koji u to vreme i na tom mestu nudi najveću brzinu.
- ◆ Dok se krećemo kroz ovo heterogeno okruženje automatski i transparentno se prebacujemo iz jedne tehnologije pristupa u drugu (na primer, iz 802.11 u 3G) zavisno od dostupnosti, bez ikakve intervencije korisnika.
- ◆ Naravno, želimo tokom našeg kretanja da zadržimo aktivne TCP konekcije. Štaviše, hteli bismo da sistem zna gde se mi nalazimo da bismo primili nove pozive i kad smo u pokrebu.

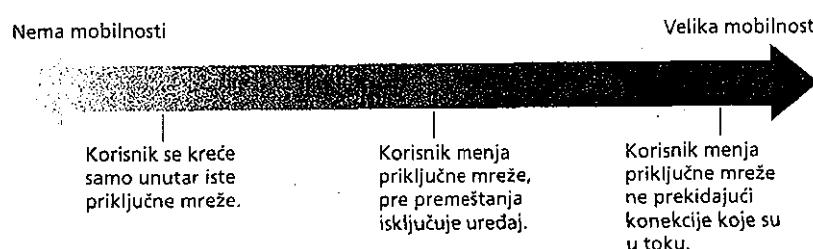
- ♦ Želeli bismo da sistem podržava govor i video u realnom vremenu preko IP-a, da bismo svi mogli da nosimo špijunske satove i vodimo video konferencije sa svojim priateljima i kolegama bez obzira na to gde se nalazimo.

A naravno, želeli bismo da sve to bude besplatno (ili barem da ne bude skupo). Mada ovo može da zvuči kao bežična nirvana, dobra vest glasi da većina potrebnih tehnoloških komponenti već postoji. Sada je jedino pitanje kako da se naprave protokoli i tehnologije koje će ih povezati da bi se ova lista želja ostvarila. U te komponente spadaju tehnologije pristupa kao što su 802.11 i 3G kao i protokoli za upravljanje mobilnošću (odeljci 6.5, 6.6 i 6.7), protokoli za kodovanje i proveru autentičnosti (poglavlje 8) i multimedijiški mrežni protokoli (kao stoje SIP za prenos govora preko IP-a, koji se opisuje u poglavlju 7).

## 6.5 Upravljanje mobilnošću: principi

Pošto smo sada opisali *bežičnu* prirodu komunikacionih linkova u bežičnoj mreži, vreme je da obratimo pažnju na *mobilnost* koju ti bežični linkovi omogućavaju. U najširem smislu, mobilni čvor je onaj koji u vremenu menja svoju tačku priključenja na mrežu. Pošto izraz *mobilnost* ima različita značenja u svetu računara i u svetu telefonije, prvo ćemo donekle detaljno razmotriti nekoliko dimenzija mobilnosti.

- ♦ *Koliko je korisnik mobilan iz aspekta mrežnog sloja?* Fizički mobilni korisnik postavlja pred mrežni sloj veoma različite probleme zavisno od načina na koji menja priključnu tačku sa mrežom. Najednom kraju spektra sa slike 6.16 korisnik može da nosi laptop sa karticom bežičnog mrežnog interfejsa po nekoj zgradi. Kao što smo videli u odeljku 6.3.4 ovaj korisnik iz perspektive mrežnog sloja *nije* mobilan. Štaviše, ako se korisnik pridružuje istoj pristupnoj tački bez obzira na to gde se nalazi on čak nije mobilan ni iz perspektive sloja veze.



Slika 6.16 ♦ Različiti stepeni mobilnosti iz aspekta mrežnog sloja

Na drugom kraju spektra zamislite korisnika koji juri auto putem u BMW-u brzinom od 150 km/h, koji prolazi kroz mnoge bežične mreže i želi da zadrži TCP konekciju sa udaljenom aplikacijom tokom celog putovanja. Ovaj korisnik *svakako jeste* mobilan! Između ta dva ekstrema imamo korisnika koji uzme svoj laptop sa jednog mesta (npr. kancelarije ili spavaonice) i prenese ga na drugo (npr. u kafić ili učionicu) i hoće da se poveže sa mrežom na tom novom mestu. Ovaj korisnik je takođe mobilan (mada manje od vozača BMW-a), ali mu nije potrebno da zadrži postojeću konekciju tokom premeštanja između mrežnih pristupnih tačaka. Na slici 6.16 prikazan je ovaj spektar korisničke mobilnosti iz aspekta mrežnog sloja.

- ♦ *Koliko je važno da adresa mobilnog Čvora ostane uvek ista?* U mobilnoj tele-foniji, vaš broj telefona - u suštini to je adresa vašeg telefona u mrežnom sloju - ostaje uvek isti kada se krećete iz mreže jednog operatera mobilne telefonije u mrežu drugog operatera. Da li laptop mora na sličan način da zadrži istu IP adresu kada se kreće među IP mrežama?

Odgovor na ovo pitanje umnogome zavisi od aplikacija koje se izvršavaju. Za vozača BMW-a koji želi da zadrži TCP konekciju sa udaljenom aplikacijom dok juri auto putem bilo bi zgodno da zadrži istu IP adresu. Verovatno se sećate iz poglavlja 3 da internetska aplikacija mora da zna IP adresu i broj porta na udaljenom uređaju sa kojim komunicira. Ako je mobilni uređaj u stanju da zadrži istu IP adresu dok se kreće, njegova mobilnost iz aspekta aplikacije ostaje neprimetna. Ova transparentnost ima veliki značaj - aplikacija ne mora da brine o eventualno promenljivoj IP adresi, pa isti aplikacijski kod služi za mobilne konekcije i za one koje to nisu. U sledećem odeljku videćemo da mobilni IP obezbeduje tu transparentnost, dozvoljavajući mobilnom čvoru da zadrži svoju trajnu IP adresu dok se kreće po mrežama.

S druge strane, skromniji mobilni korisnik će možda samo hteti da isključi laptop u kancelariji, doneše ga kući, uključi ga i nastavi da radi od kuće. Ako laptop od kuće funkcioniše pre svega kao klijent u klijent-server aplikacijama (npr. slanje i primanje e-pošte, pretraživanje Weba, Telnet prema udaljenom računaru), konkretna IP adresa koju koristi nije toliko važna. Pogotovo, dobro će poslužiti i adresa koju laptopu privremeno dodeljuje posrednik za Internet usluge čiji nalog koristite kod kuće, U odeljku 5.4.3 videli smo da DHCP već omogućava ovu funkciju.

- ♦ *Koja je očišćena infrastruktura dostupna za podršku?* U svim gore opisanim scenarijima, implicitno smo prepostavili da postoji fiksna infrastruktura sa kojom mobilni korisnik može da se poveže, na primer, kućna mreža posrednika Internet usluga, bežična priključna mreža u kancelariji ili bežična priključna mreža duž celog autoputa. A šta ako takve infrastrukture nema? Ako se dva korisnika nalaze dovoljno blizu, da li mogu da ustpostave mrežnu vezu i bez ikakve druge infrastrukture mrežnog sloja? Ad hoc umrežavanje pruža upravo tu mogućnost. Ovo područje koje se ubrzano razvija predstavlja vrhunac istraživanja mobil-

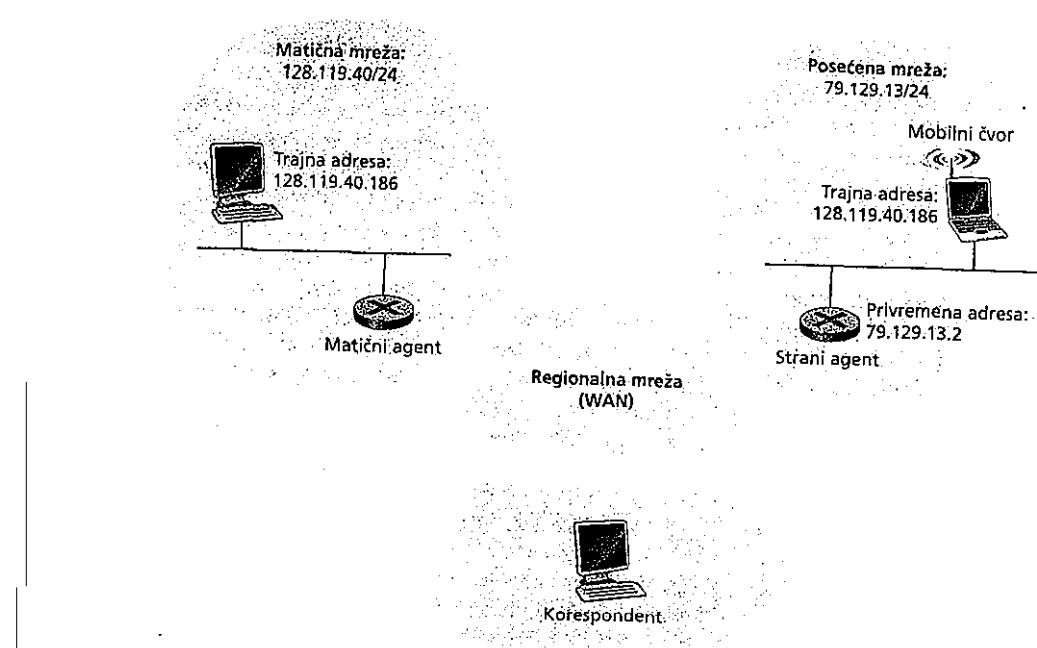
nog umrežavanja i prelazi okvire ove knjige. [Perkins 2000] i veb stranice IETF radne grupe Mobile Ad Hoc Network [manet 2004] sadrže sveobuhvatnu obradu ove teme.

Da bismo ilustrovali probleme koje postavlja zahtev da mobilni korisnik zadrži postojeće koneksijske dok se kreće među mrežama, razmotrimo jednu analogiju iz svakodnevног života. Mlada osoba koja napusti roditeljsku kuću postaje mobilna, stanuje u raznim spavaonicama ili stanovima i često menja adrese. Ako stari prijatelj hoće da ga pronađe, kako će da utvrdi adresu svog mobilnog prijatelja? Uobičajeni načinje da se potraži porodica, pošto mobilni mladić često prijavljuje svoju trenutnu adresu (ako ni zbog čega drugog onda zato da bi roditelji mogli da mu pošalju novac ako mu nedostaje za stanarinu!). Porodična kuća sa svojom trajnom adresom postaje mesto koje će drugi ljudi prvo potražiti kako bi stupili u kontakt sa mobilnim mladićem. Nakon toga, komunikacija sa prijateljem može biti indirektna (na primer, da se pošta šalje roditeljima koji je zatim prosleđuju mobilnom mladiću) ili direktna (na primer, kada prijatelj upotrebi dobijenu adresu i šalje poštu direktno mobilnom prijatelju).

U mrežnom okruženju, trajna „kuća“ mobilnog čvora (kao stoje laptop ili PDA) naziva se matičnom mrežom, a entitet u matičnoj mreži koji upravlja funkcijama mobilnosti u ime mobilnog čvora naziva se matični agent. Mreža u kojoj se mobilni čvor trenutno nalazi naziva se strana (ili posećena) mreža, a entitet u stranoj mreži koji pomaže mobilnom čvoru u funkcijama upravljanja mobiinošću naziva se strani agent. Za mobilne profesionalce, matična mreža je verovatno mreža njihove kompanije, a posećena mreža može biti mreža nekog kolege kod kojeg se nalaze. Kore-spondent je entitet koji želi da komunicira sa mobilnim čvrom. Slika 6.17 ilustruje ovaj koncept, kao i koncept adresiranja koji će biti izložen u nastavku. Na slici 6.17 može se primetiti da se agenti nalaze na istom mestu gde i ruteri (na primer, kao procesi koji se izvršavaju u ruterima), ali se oni mogu izvršavati i na drugim računarima ili serverima u mreži.

### 6.5.1 Adresiranje.

Već smo napomenuli daje poželjno da mobilni čvor zadrži istu adresu prilikom premeštanja iz mreže u mrežu zato da bi njegova mobilnost bila transparentna za mrežne aplikacije. Kada se mobilni čvor nalazi u stranoj mreži, sav saobraćaj adresiran na njegovu trajnu adresu mora da se rutira u stranu mrežu. Kako se to postiže? Jedna mogućnost je da strana mreža objavi svim ostalim mrežama da se mobilni čvor nalazi kod nje. To bi moglo da se postigne putem uobičajene razmene informacija o rutiranju unutar domena i među domenima, i zahtevalo bi samo male izmene u postojećoj infrastrukturi rutiranja. Strana mreža bi jednostavno objavila svojim susedima da poseduje veoma specifičnu rutu prema trajnoj adresi mobilnog čvora (tj. u suštini bi obavestila ostale mreže da poseduje tačnu putanju za rutiranje



**Slika 6.17** ♦ Početni elementi mobilne mrežne arhitekture

datagrama prema trajnoj adresi mobilnog čvora; pročitajte odeljak 4.4). Ti susedi bi zatim propagirali ovu informaciju po celoj mreži u sklopu normalne procedure za ažuriranje informacija o rutiranju i tabela prosleđivanja. Kada mobilni čvor napusti jednu stranu mrežu i pređe u drugu, nova strana mreža će objaviti novu, sasvim konkretnu rutu prema mobilnom čvoru, a staru stranu mreža će povući informacije o rutiranju koje se odnose na mobilni čvor.

Na ovaj način se istovremeno rešavaju dva problema i to bez značajnih izmena infrastrukture mrežnog sloja. Druge mreže znaju za lokaciju mobilnog čvora i datagrami se lako rutiraju prema njemu pošto će tabele prosleđivanja usmeriti datagram u stranu mrežu. Međutim, tu postoji značajan nedostatak skalabilnosti. Kada bi upravljanje mobiinošću postalo zadatak mrežnih ruteru, oni bi možda morali u tabeli prosleđivanja da održavaju stavke za milione mobilnih čvorova i da ih ažuriraju u skladu sa kretanjem čvorova. U problemima na kraju ovog poglavlja istražuju se još neki nedostaci.

Drugi pristup (pristup koji je prihvaćen u praksi) jeste da se funkcije mobilnosti prenesu iz jezgra mreže na njenu periferiju - tema na koju stalno nailazimo u proučavanju arhitekture Interneta. Prirodno je da se to uradi preko matične mreže mobilnog čvora. Na sličan način kao što roditelji mobilnog tinejdžera prate njegovu lokaciju, matični agent u matičnoj mreži mobilnog čvora može da prati stranu mrežu u kojoj se mobilni Čvor nalazi. Da bi se ažurirala prava lokacija mobilnog Čvora svakako će biti potreban i protokol između mobilnog čvora (ili stranog agenta koji ga predstavlja) i njegovog matičnog agenta.

Razmotrimo sada malo detaljnije stranog agenta. Prema najjednostavnijoj zamisli prikazanoj na slici 6.17, strani agenti se stavlaju u periferne rutere strane mreže. Jedna od uloga stranog agenta je da napravi takozvanu **privremenu adresu** (*care-of address, COA*) za mobilni čvor u kojoj mrežni deo adrese odgovara stranoj mreži. Na taj način su mobilnom čvoru pridružene dve adrese, njegova **trajna adresa** (analogna roditeljskoj adresi mobilnog tinejdžera) i njegova COA koju ponekad nazivamo **stranom adresom** (analogno adresi kuće u kojoj mobilni tinejdžer trenutno stanuje). U primeru sa slike 6.17, trajna adresa mobilnog Čvora je 128.119.40.186. Dok se nalazi u poseti mreži 79.129.13/24, mobilni čvor ima adresu COA 79.129.13.2. Druga uloga stranog agenta je da obavesti matičnog agenta da se mobilni čvor nalazi u njegovoj (stranoj) mreži i daje dobio adresu COA. Uskoro ćemo videti da se COA koristi za „prerutiranje“ datograma mobilnom čvoru preko njegovog stranog agenta.

Mada smo razdvojili funkcije mobilnog čvora i stranog agenta, treba napomenuti da zadatke stranog agenta može da preuzme i sam mobilni čvor. Na primer, mobilni čvor bi mogao da pribavi adresu COA u stranoj mreži (na primer, pomoću protokola kakav je DHCP) i sam obavesti matičnog agenta o toj privremenoj adresi.

### 6.5.2 Rutiranje prema mobilnom čvoru

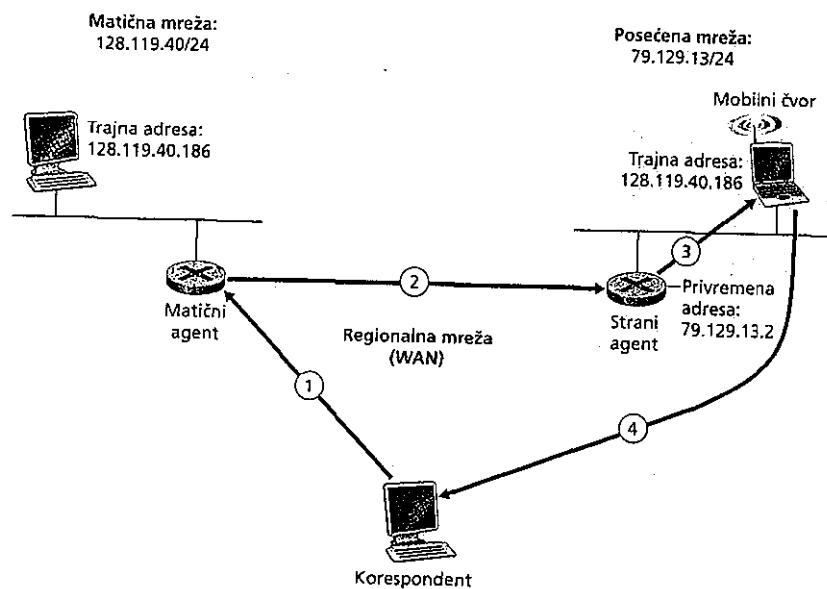
Sada smo videli kako mobilni Čvor dobija adresu COA i kako se matični agent obaveštava o toj adresi. Ali, to što matični agent zna za adresu COA rešava samo deo problema. Kako adresirati datagrame i kako ih proslediti mobilnom čvoru? Pošto za lokaciju mobilnog čvora zna samo matični agent (a ne i ruteri svuda u mreži) neće više biti dovoljno da se datagram jednostavno adresira na trajnu adresu mobilnog Čvora i pošalje u infrastrukturu mrežnog sloja. Potrebno je uraditi nešto više. Moguća su dva pristupa koja ćemo nazvati indirektnim i direktnim rutiranjem.

#### Indirektno rutiranje prema mobilnom čvoru

Razmotrimo prvo korespondenta koji želi da pošalje datagram mobilnom čvoru. Ako se koristi **indirektno rutiranje**, korespondent jednostavno adresira datagram na trajnu adresu mobilnog Čvora i šalje datagram u mrežu, u blaženom neznanju da

li se mobilni Čvor nalazi u svojoj matičnoj mreži ili u poseti stranoj mreži; mobilnost ostaje za korespondenta potpuno transparentna. Takvi datagrami se prvo rutiraju kao i obično u matičnu mrežu mobilnog čvora. Ovo je prikazano kao korak 1 na slici 6.18.

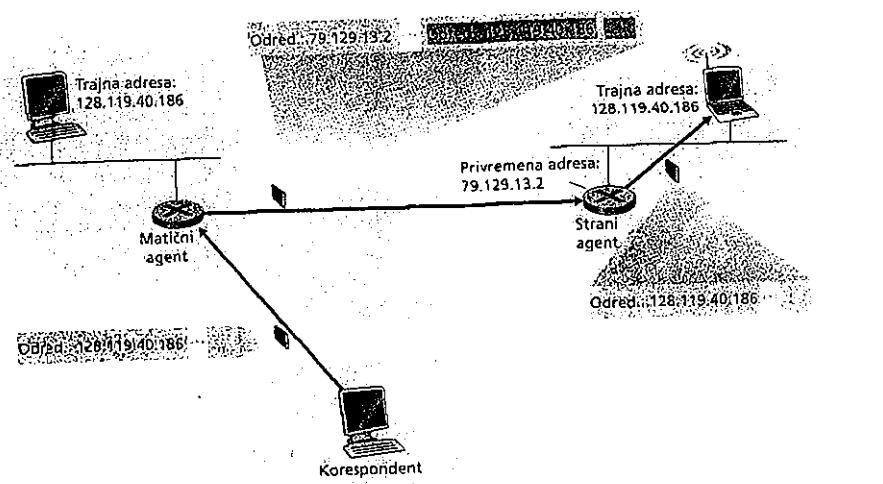
Obratimo sada pažnju na matičnog agenta. Osim što je zadužen za interakciju sa stranim agentom kako bi pratio adresu COA mobilnog čvora, matični agent ima još jednu veoma važnu funkciju. Njegov drugi zadatak je da čeka datagrame adresirane na čvorove čija matična mreža jeste mreža matičnog agenta, ali se trenutno nalaze u stranoj mreži. Matični agent presreće te datagrame i zatim ih prosledjuje mobilnom čvoru u postupku od dva koraka. Datagram se prvo prosledjuje stranom agentu pomoću adrese COA mobilnog čvora (korak 2 na slici 6.18), a zatim se prosledjuje od stranog agenta ka mobilnom Čvoru (korak 3 na slici 6.18).



**Slika 6.18** ♦ Indirektno prosleđivanje prema mobilnom čvoru

Proučimo detaljnije ovo prerutiranje. Da bi mrežni sloj rutirao datagram u stranu mrežu, matični agent mora da adresira datagram pomoću adrese COA mobilnog čvora. S druge strane, poželjno je da datagram korespondenta ostane netaknut pošto aplikacija koja prima datagram ne srne da primeti daje datagram posleden preko matičnog agenta. Oba ova zahteva mogu da se zadovolje ako matični agent enkapsulira kompletan prvobitni datagram dobijen od korespondenta u jedan novi (veći) datagram. Ovaj veći datagram se adresira i isporučuje na adresu COA mobilnog čvora.

Strani agent, „vlasnik“ adrese COA, će primiti i dekapsulirati datagram, tj. izdvojiti prvobitni datagram korespondenta iz većeg enkapsulirajućeg datagrama i zatim proslediti (korak 3 na slici 6.18) prvobitni datagram mobilnom čvoru. Na slici 6.19 prikazanje prvobitnog datagrama korespondenta koji se šalje u matičnu mrežu, enkapsulirani datagram koji se šalje stranom agentu i prvobitni datagram koji se isporučuje mobilnom čvoru. Pronicljivi čitalac će primetiti da je enkapsuliranje i dekapsuliranje koje ovde opisujemo identično pojmu tunelovanja koji smo u poglavlju 4 opisivali u kontekstima višežnačnog IP upućivanja i protokola IPv6.



**Slika 6.19** ♦ Enkapsuliranje i dekapsuliranje

Pogledajmo sada kako mobilni čvor šalje datagrame korespondentu. To je sasvim jednostavno pošto mobilni čvor može svoje datagrame da adresira *direktно* korespondentu (koristeći vlastitu trajnu adresu kao adresu izvora, a korespondentovu adresu kao adresu odredišta). Pošto mobilni čvor zna adresu korespondenta, nije potrebno da se datagram rutira preko matičnog agenta. Ovo je prikazano kao korak 4 na slici 6.18.

U ovoj rekapitulaciji indirektnog rutiranja dajemo spisak novih funkcija mrežnog sloja potrebnih za podršku mobilnosti.

- ♦ *Protokol između mobilnog čvora i stranog agenta.* Mobilni čvor će se prijaviti kod stranog agenta prilikom povezivanja u stranu mrežu. Slično tome, mobilni čvor će se odjaviti kod stranog agenta kada bude napuštao stranu mrežu.
- ♦ *Protokol kojim strani agent vrši registraciju kod matičnog agenta.* Strani agent će prijaviti adresu COA mobilnog čvora matičnom agentu. Sirani agent ne mora eksplicitno da odjavи adresu COA kada mobilni čvor napusti njegovu mrežu pošto će to resiti naknadno prijavljivanje nove adrese COA kada mobilni čvor pređe u novu mrežu.
- ♦ *Protokol kojim matični agent enkapsulira datagram.* Za enkapsuliranje i prosleđivanje prvobitnog datagrama korespondenta u datagramu adresiranom na adresu COA.
- ♦ *Protokol kojim strani agent dekapsulira datagram.* Za izdvajanje korespondentovog prvobitnog datograma iz enkapsulirajućeg datagrama i za prosleđivanje prvobitnog datagrama mobilnom čvoru.

Prethodni opis sadrži sve delove - strane agente, matičnog agenta i indirektno prosleđivanje - potrebne da bi mobilni čvor zadržao neprekinutu konekciju dok se kreće po raznim mrežama. Kao primer ove saradnje pretpostavimo daje mobilni čvor vezan za stranu mrežu A, da je prijavio adresu COA iz mreže A svom matičnom agentu i da prima datagrame indirektno rutirane preko njegovog matičnog agenta. Mobilni čvor sada prelazi u stranu mrežu B i registruje se kod stranog agenta u mreži B koji obaveštava matičnog agenta o novoj adresi COA mobilnog čvora. Počevši od tog trenutka, matični agent će prerutirati datagrame u stranu mrežu B. Što se tiče korespondenta, mobilnost je transparentna - i pre i posle premeštanja datagrami se rutiraju preko istog matičnog agenta. Što se tiče matičnog agenta, nema prekida u toku datagrama - pristigli datagrami se prvo prosleđuju u stranu mrežu A; nakon promene adrese COA datagrami se prosleđuju u stranu mrežu B. Ali, da li će mobilni čvor primetiti prekid u toku datagrama kada promeni mrežu? Ako je vreme od trenutka kada mobilni čvor prekine konekciju sa mrežom A (kada više ne može da prima datagrame preko mreže A) do trenutka kada se priključi na mrežu B (kada bude prijavio novu adresu COA svom matičnom agentu) dovoljno kratko, izgubiće se mali broj datagrama. Verovatno se sećate iz poglavlja 3 da u konekcijama sa kraja na kraj može doći do gubitaka datagrama zbog zagubljenja mreže. Prema tome,

povremenim gubitak datagrama u konekciji kada čvor prelazi iz mreže u mrežu nikako ne predstavlja katastrofalan problem. Ako je potrebna komunikacija bez gubitaka, mehanizmi u višim slojevima će se pobrinuti za oporavak od gubitaka, bez obzira na to da li je do gubitka došlo zbog zagušenja mreže ili zbog mobilnosti korisnika.

U standardu za mobilni IP [RFC 3220] prihvaćen je pristup sa indirektnim ruti-ranjem kao sto je opisano u odeljku 6.6.

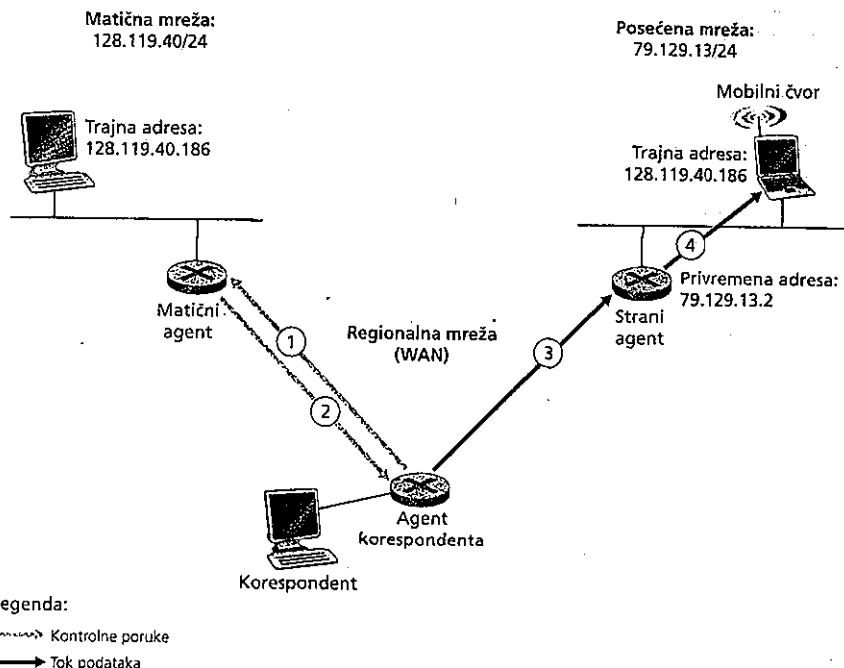
#### Direktno rutiranje prema mobilnom čvoru

Pristup sa indirektnim rutiranjem prikazan na slici 6.18 pati od problema neefikasnosti poznatog pod imenom **trougaon rutiranje** - datagrami adresirani mobilnom čvoru moraju se prvo rutirati matičnom agentu i zatim stranoj mreži. Čak i kada između korespondenta i mobilnog čvora postoji mnogo efikasnija ruta. U najgorem slučaju, zamislite mobilnog korisnika u poseti stranoj mreži nekog kolege. Njih dvojica sede jedan pored drugog i razmenjuju podatke preko mreže. Datagrami od korespondenta (u ovom slučaju kolege našeg posetioca) rutiraju se matičnom agentu mobilnog korisnika i ponovo nazad u stranu mrežu!

**Direktno rutiranje** etiminiše problem trougla rutiranja, ali zato povećava složenost. U pristupu sa direktnim rutiranjem, za adresu COA mobilnog čvora prvo saznaće **agent korespondenta** u mreži korespondenta. To se može postići tako da agent korespondenta posalje upit matičnom agentu pod pretpostavkom (kao u slučaju indirektnog rutiranja) daje mobilni čvor prijavio ažurnu vrednost adrese COA svom matičnom agentu. Takođe je moguće da funkciju agenta korespondenta vrši sam korespondent, kao što je mobilni čvor mogao da vrši funkciju stranog agenta. Ovo je prikazano kao koraci 1 i 2 na slici 6.20. Agent korespondenta zatim direktno tuneluje datagrame na adresu COA mobilnog čvora slično tunelovanju koje je obavljao matični agent (koraci 3 i 4 na slici 6.20).

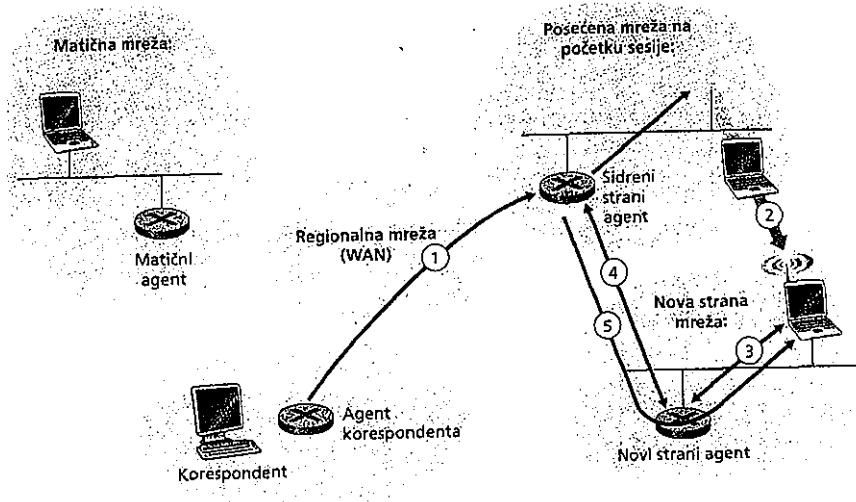
Iako direktno rutiranje rešava problem trougla rutiranja, ono dovodi do dodatne složenosti:

- ◆ Potreban je poseban **protokol za pronalaženje mobilnog korisnika** da bi agent korespondenta od matičnog agenta saznao adresu COA mobilnog čvora (koraci 1 i 2 na slici 6.20).
- ◆ Kada mobilni čvor prelazi iz jedne strane mreže u drugu, kako će se podaci proslediti u novu stranu mrežu? U slučaju indirektnog rutiranja, ovaj problem se jednostavno rešavao ažuriranjem adrese COA pri matičnom agentu. Međutim, kod direktnog rutiranja agent korespondenta samo jedanput na početku sesije traži adresu COA od matičnog agenta. Prema tome, ažuriranje COA kod matičnog agenta (mada neophodno) neće biti dovoljno da resi problem rutiranja podataka prema novoj stranoj mreži mobilnog čvora.



Slika 6.20 ♦ Direktno rutiranje prema mobilnom korisniku

Ovde bijedno rešenje bilo da se napravi novi protokol za obaveštavanje korespondenta o promjenjenoj adresi COA. Drugo rešenje koje je primenjeno u praksi u GSM mrežama, funkcioniše na sledeći način. Prepostavimo da se podaci trenutno prosleđuju mobilnom čvoru u stranoj mreži u kojoj se mobilni čvor nalazio na početku sesije (korak 1 na slici 6.21). Strani agent u stranoj mreži u kojoj se mobilni čvor nalazi na početku nazvaćemo **sidreni strani agent**. Kada mobilni čvor pređe u novu stranu mrežu (korak 2 na slici 6.21) mobilni čvor se prijavljuje novom stranom agentu (korak 3), a novi strani agent predaje sidrenom stranom agentu novu adresu COA mobilnog čvora (korak 4). Sidreni strani agent koji primi enkapsulirani datagram za mobilni čvor koji je otiašao ponovo će enkapsulirati datagram i proslediti ga na novu COA (korak 5). Ako mobilni čvor kasnije ponovo ode u novu stranu mrežu, strani agent nove posećene mreže će tada obavestiti sidrenog stranog agenta da se pripremi prosleđivanje u tu novu stranu mrežu.



**Slika 6.21** ♦ Krećanje mobilnog korisnika među mrežama uz direktno rutiranje

## 6.6 Mobilni IP

Arhitektura Interneta i protokoli za podršku mobilnosti koji se zajedno zovu mobilni IP prvo su definisani u standardu RFC 3220. Mobilni IP je fleksibilan standard, koji podržava mnoge različite režime rada, na primer, rad sa stranim agentom ili bez njega, više načina za međusobno otkrivanje agenata i mobilnih čvorova, korišćenje jedne ili više adresa COA i više oblika enkapsuliranja. Kao takav, mobilni IP je složeni standard i bila bi potrebna cela knjiga da se detaljno opiše; jedna od tih knjiga je [Perkins 1998b]. Naš skromni cilj je da ovde opišemo najvažnije aspekte mobilnog IP-a i da prikažemo njegovo korišćenje u nekoliko uobičajenih situacija.

Arhitektura mobilnog IP-a sadrži mnoge od elemenata koje smo upravo razmatrali uključujući i koncepciju matičnih agenata, stranih agenata, privremenih adresa, kao i enkapsulaciju i dekapsulaciju. Važeći standard [RFC 3220] zahteva indirektno rutiranje prema mobilnom čvoru.

Mobilni standard ima tri glavna dela:

- ♦ *Otkrivanje agenta.* Mobilni IP definiše protokole kojima matični i strani agent objavljuju svoje usluge mobilnim čvorovima i protokole kojima mobilni čvorovi zahtevaju usluge stranog ili matičnog agenta.

- ♦ *Prijavljanje matičnom agentu.* Mobilni IP definiše protokole pomoću kojih mobilni Čvor i/ili strani agent prijavljuje i odjavljuje adresu COA mobilnog čvora matičnom agentu.

- ♦ *Indirektno rutiranje datagrama.* Standard takođe definiše način na koji matični agent prosleđuje datagrame mobilnim Čvorovima, uključujući pravila za prosleđivanje datagrama, pravila za obradu grešaka i nekoliko različitih oblika enkapsuliranja [RFC 2003, RFC 2004].

Pitanja bezbednosti imaju veliki značaj u celom standardu za mobilni IP. Na primer, svakako je potrebna provera autentičnosti mobilnog čvora da bi se zlona-memni korisnik spremio da prijavi matičnom agentu lažnu privremenu adresu, pa bi svi datagrami adresirani na neku IP adresu bili preusmereni zlonamernom korisniku. Mobilni IP postiže bezbednost pomoću mnogih mehanizama koje ćemo obraditi u poglavljiju 8, pa se u našem sledećem tekstu nećemo baviti pitanjima bezbednosti.

### Otkrivanje agenta

Mobilni IP čvor koji stiže u novu mrežu, bilo da se priključuje stranoj mreži ili se vraća u svoju matičnu mrežu, mora da sazna identitet odgovarajućeg stranog ili matičnog agenta. Zaista, mrežni sloj u mobilnom Čvoru saznaće daje premešten u novu stranu mrežu kada otkrije novog stranog agenta sa novom mrežnom adresom. Ovaj postupak se naziva otkrivanje agenta. Otkrivanje agenta može se postići na jedan od sledeća dva načina: putem objave agenta ili putem zahteva agentu.

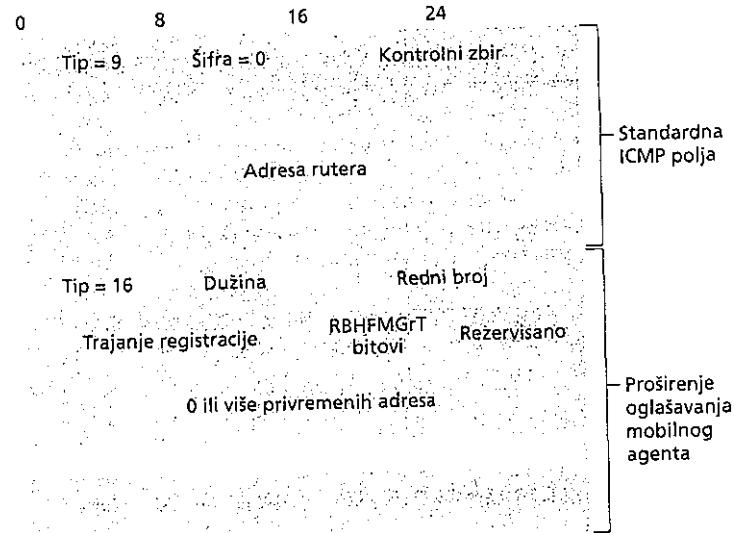
Ako se koristi oglašavanje agenta, strani ili matični agent oglašava svoje usluge pomoću proširenja postojećeg protokola za otkrivanje rutera [RFC 1256]. Agent svim linkovima na koje je povezan povremeno difuzno emituje ICMP poruku sa vrednošću 9 (otkrivanje rutera) u polju za tip poruke. Poruka za otkrivanje rutera sadrži IP adresu rutera (tj. agenta) i tako omogućava mobilnom čvoru da sazna IP adresu agenta. Poruka za otkrivanje rutera takođe sadrži proširenje sa objavom mobilnog agenta koje sadrži dodatne informacije potrebne mobilnom čvoru. Značajnija polja u tom proširenju su:

- ♦ *Bit matičnog agenta (H).* Označava da je reč o matičnom agentu za mrežu u kojoj se nalazi.
- ♦ *Bit stranog agenta (F).* Označava da je reč o stranom agentu za mrežu u kojoj se nalazi.
- ♦ *Bit obaveznog registrovanja (R).* Označava da mobilni korisnik u ovoj mreži *mora* da se prijavi stranom agentu. Konkretno, mobilni korisnik ne može sam da pribavi privremenu adresu u stranoj mreži (na primer, pomoću DHCP-a) i da sam preuzme funkcije stranog agenta bez prijavljivanja stranom agenom.

- ♦ *M, G bitovi enkapsuliranja.* Označavaju da li će se koristiti neki vid enkapsulira-nja osim enkapsuliranj a IP-u-IP.
- ♦ *Poљa COA (Care-of addresses).* Spisak od jedne ili nekoliko privremenih adresa koje nudi strani agent. U našem sledećem primeru, COA će biti pridružena stranom agentu koji je zadužen za primanje datagrama koji se šalju na adresu COA i za njihovo prosleđivanje odgovarajućem mobilnom čvoru. Mobilni korisnik će prilikom prijavljivanja svom matičnom agentu izabrati jednu od ovih adresa za svoju adresu COA.

Na slici 6.22 prikazana su neka od ključnih polja u agentovoj poruci objave.

Ako se koriste zahtevi agentu, mobilni čvor koji želi da sazna za agente, a ne da čeka dok primi oglašavanje agenta, može difuzno da emituje poruku zahteva agentu - to je jednostavno ICMP poruka sa vrednošću 10 u polju tipa. Agent koji primi takav zahtev jednoznačno će uputiti oglašavanje agenta direktno mobilnom čvoru koji će zatim postupiti kao daje primio oglašavanje koje nije sam tražio.



**Slika 6.22** ♦ ICMP poruka za otkrivanje rutera sa proširenjem oglašavanja agenta mobilnosti

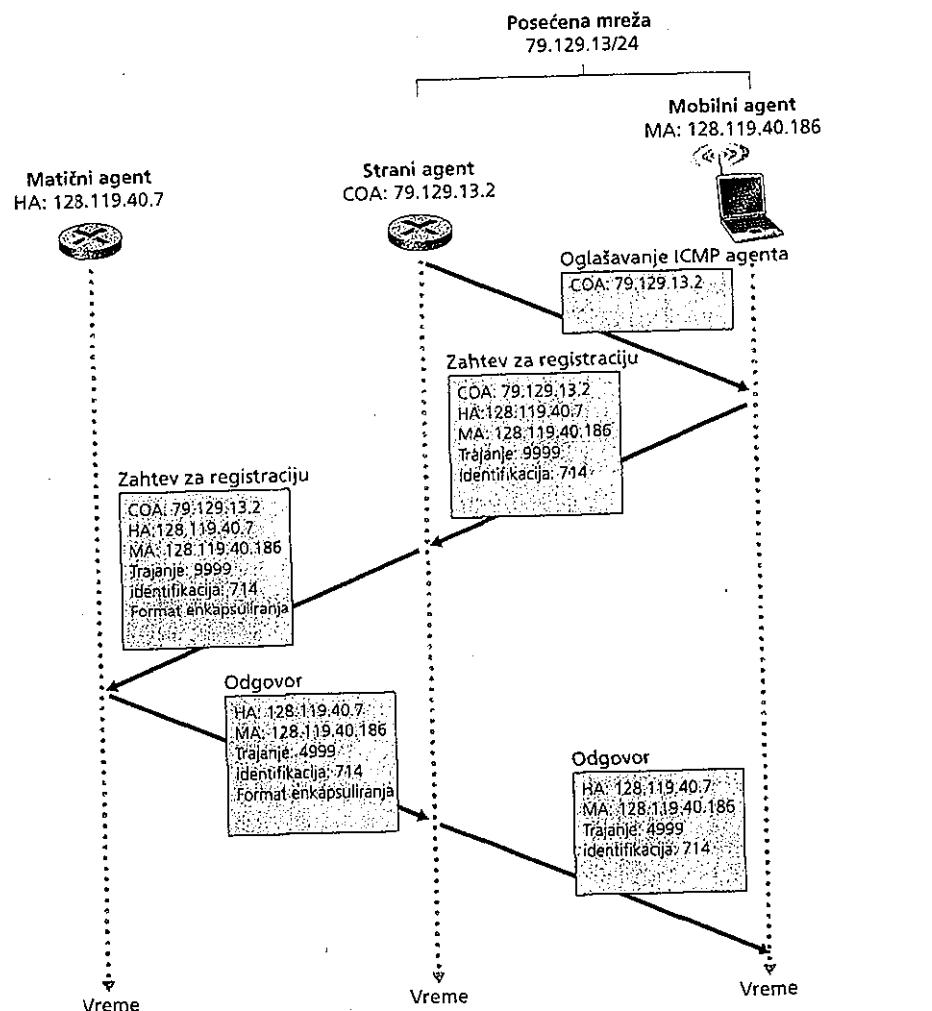
#### Prijavljinje matičnom agentu

Kada mobilni IP čvor pribavi adresu COA, ta adresa mora da se prijavi matičnom agentu. To može da uradi strani agent (koji zatim prijavljuje adresu COA matičnom agentu) ili direktno sam mobilni IP čvor. Razmotrićemo prvi slučaj koji se sastoji iz četiri koraka.

1. Pošto primi oglašavanje stranog agenta, mobilni čvor šalje stranom agentu poruku za prijavljivanje mobilnog IP-a. Poruka prijavljivanja stavlja se u UDP datagram i šalje na port 434. Poruka prijavljivanja sadrži adresu COA iz oglašavanja stranog agenta, adresu matičnog agenta (*home agent*, HA), trajnu adresu mobilnog Čvora (MA), zahtevano trajanje prijave i identifikaciju prijave od 64 bita. Zahtevano trajanje prijave je broj sekundi koliko prijava treba da važi. Ako se u navedenom intervalu prijave ne obnovi kod matičnog agenta, ona prestaje da važi. Identifikator prijave služi kao redni broj i potreban je za uparivanje odgovora o primljenoj rezervaciji sa zahtevom za rezervaciju, što ćemo kasnije opisati.
2. Strani agent prima poruku prijavljivanja i evidentira trajnu IP adresu mobilnog čvora. Strani agent sada zna da treba očekivati datagrame koji sadrže enkapsulirane datagrame čija je odredišna adresa jednaka trajnoj adresi mobilnog čvora. Strani agent tada šalje poruku sa mobilnom IP prijavom (opet u UDP datagramu) na port 434 matičnog agenta. Poruka sadrži COA, HA, MA, zahtevani format enkapsuliran, a zahtevano trajanje prijave i identifikaciju prijave.
3. Matični agent prima zahtev za prijavu i proverava autentičnost i ispravnost. Matični agent povezuje trajnu IP adresu mobilnog čvora sa adresom COA; ubuduće, datagrami koji budu stizali matičnom agentu, a namenjeni su mobilnom čvoru biće enkapsulirani i tunelovani ka adresi COA. Matični agent šalje odgovor na mobilnu IP prijavu, a on sadrži HA, MA, odobreno trajanje prijave identifikaciju prijave iz zahteva na koji se ovaj odgovor odnosi.
4. Strani agent prima odgovor na prijavu i prosleđuje ga mobilnom čvoru.

U ovom trenutku prijava je potpuna i mobilni čvor može da prima datagrame upućene na njegovu trajnu adresu. Ovi su koraci prikazani na slici 6.23. Obratite pažnju na to da matični agent odobrava manje trajanje nego što je mobilni čvor zahtevao.

Strani agent ne mora eksplicitno da odjavi adresu COA kada mobilni čvor napusti njegovu mrežu. To će se postići automatski kada mobilni Čvor stigne u novu mrežu (bilo drugu stranu mrežu ili svoju matičnu mrežu) i prijavi novu adresu COA.



**Slika 6.23** ♦ Oglašavanje agenata i mobilno IP prijavljivanje

Osim ovde opisanih, standard za mobilni IP dozvoljava i mnoge druge scenarije. Čitaoci koje to zanima trebalo bi da pročitaju knjige [Perkms 1998b; RFC 32201-]

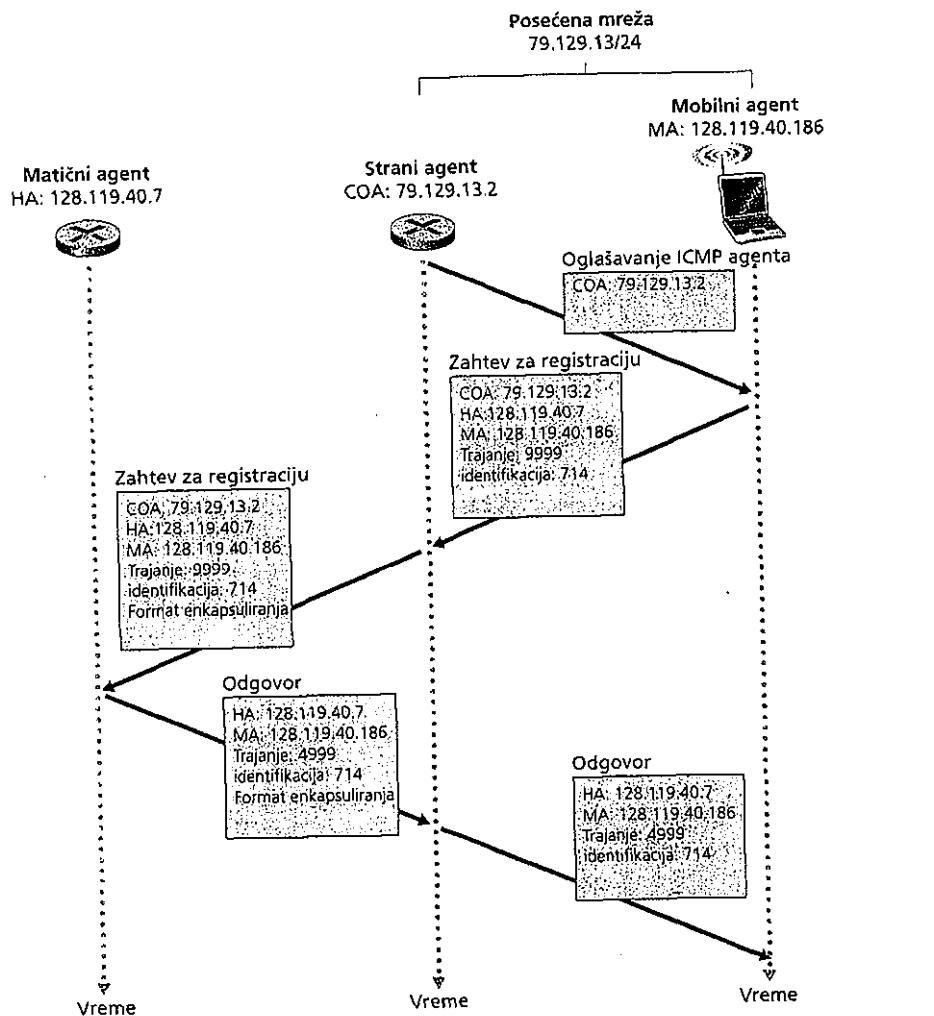
## 6.7 Upravljanje mobiinošću u celularnim mrežama

Pošto smo razmotrili način na koji se mobiinošć upravlja u IP mrežama, обратимо сада пажњу на мреже са још већим искуством за подршку мобилности - мреже мобилне телефоније. Док smo у одељку 6.4 analizirali беžични link првог скока у celularним мрежама, овде se usredosređujemo на мобилност у архитектури GSM мреже [Goodman 1997; Mouly 1992; Scourias 1997; Kaaranen 2001; Korhonen 2002] постоје то зрећа и широко примењена технологија. Као и у slučaju мобилног IP-a, видећемо дате у архитектуру GSM мреже уgrađen niz осnovних принципа које smo utvrdili u одељку 6.5.

Isto као и мобилни IP, GSM приhvata принцип индиректног rutiranja (одељак 6.5.2), тако што poziv korespondenta прво rutira у матичну мрежу мобилног корисника, а оданде у posećenu mrežu. У terminologiji GSM-a матична мрежа мобилног корисника назива се **матична јавна земаљска mobilna mreža** (home Public Land Mobile Network, home PLMN). Пошто se akronim PLMN teško izgovara i s obzirom na нашу жељу да izbegnemo prevelik broj akronima, матични PLMN u GSM mreži назваћемо једnostavno **матичном мрежом**. Матична мрежа је мобилни оператор код којег је мобилни корисник pretplaćen (тј. оператор који кориснику шале месечне фактуре за услуге). Posećeni PLMN, којег ћемо једноставно звати **posećena mreža**, јесте мрежа у којој се мобилни корисник trenutno налази.

Kao i u slučaju мобилног IP-a, задужења матичне и posećene мреже се veoma razlikuju.

- ◆ Матична мрежа održava bazu podataka poznatu kao матични регистар локација (**Home Location Register**, HLR) који за svakog preplatnika sadrži trajni broj телефона и информације о профилу preplatnika. Veoma je važno što HLR takođe sadrži информацију о trenutnoj lokaciji preplatnika. То јест, ако је мобилни корисник trenutno у romingu у мрежи другог оператора, HLR sadrži dovoljno информација за utvrđivanje (postupkom који ћемо uskoro opisati) адресе у posećenoj мрежи на коју треба rutirati pozive мобилном кориснику. Као што ћемо видети, када poziva мобилног корисника korespondent stupa u vezu sa posebnim komutatorom u матичној мрежи који se назива **GMSC (Gateway Mobile Services Svitching Center)**. I ovde, da bismo smanjili broj upotrebljenih akronima GMSC ћемо nazivati **матични MSC**.
- ◆ Posećena mreža održava bazu podataka poznatu под именом регистар локација посјетилаца (**Visitor Location Register**, VLR). VLR sadrži по једну stavku за svakog мобилног корисника који se trenutno налази у делу мреже који опслужује VLR. Stavke VLR-a se zato dodaju i izbacuju како мобилни корисници улaze у мрежу или је napustaju. VLR se obično налази на истом mesni где је MSC (Mobile Switching Center) који координира uspostavljanje poziva prema posećenoj mreži и из ње.



**Slika 6.23** ♦ Oglasavanje agenata i mobilno IP prijavljivanje

Osim ovde opisanih, standard za mobilni IP dozvoljava i mnoge drugačije scenarije. Čitaoci koje to zanima trebalo bi da pročitaju knjige [Perkins 1998b; RFC 3220].

## 6.7 Upravljanje mobilnošću u celularnim mrežama

Pošto smo razmotrili način na koji se mobilnošću upravlja u IP mrežama, обратимо сада пажњу на мреже са још већим искуством за подршку mobilnosti - мреже мобилне телефоније. Док smo у оделјку 6.4 analizirali беžићни link првог скока у celularnim мрежама, овде se usredstavljemo на mobilnost u arhitekturi GSM мреже [Goodman 1997; Mouly 1992; Scourias 1997; Kaaranen 2001; Korhonen 2002] постоје то зреља и широко применета технологија. Као и у случају mobilnog IP-a, видећемо даје у arhitekturu GSM mreže ugrađen niz osnovnih principa које smo utvrdili u odeljku 6.5.

Isto као и у mobilni IP, GSM приhvata принцип indirektnog rutiranja (odeljak 6.5.2), тако што poziv korespondenta прво rutira у матичну мрежу mobilnog korisnika, а оданде у posećenu mrežu. У терминологији GSM-a матична мрежа mobilnog korisnika назива се **матична јавна земаљска mobilna mreža** (*home Public Land Mobile Network*, *home PLMN*). Постоје akronim PLMN teško izgovara и с обзиром на нашу жељу да izbegnemo prevelik broj akronima, matični PLMN у GSM мрежи называјмо једноставно **матичном' mrežom**. Матична мрежа је mobilni operater код кога је mobilni korisnik pretplaćen (тј. оператор који кориснику шаље месечне фактуре за услуге). Posećeni PLMN, којег ћемо једноставно звати **posećena mreža**, јесте мрежа у којој се mobilni korisnik trenutno налази. Као и у случају mobilnog IP-a, zaduženja матичне и posećene mreže се veoma razlikuju.

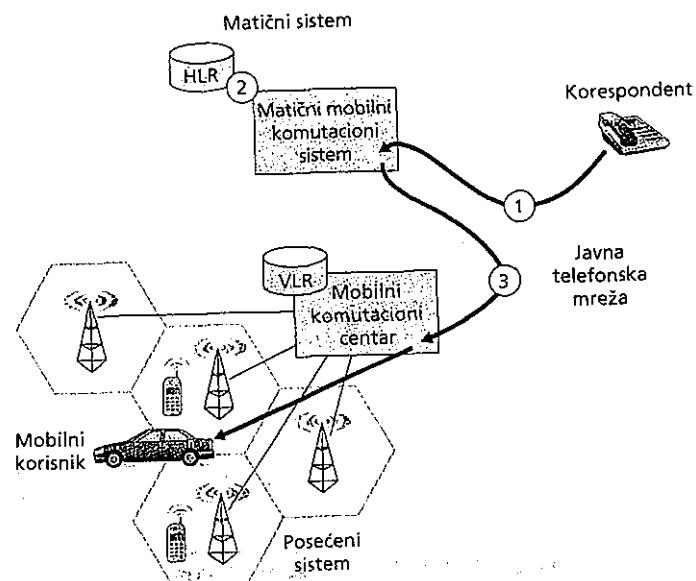
- ◆ Matična mreža održava базу података познату као матични регистар локација (*Home Location Register, HLR*) који за сваког претплатника садржи трајни број телефона и информације о профилу претплатника. Veoma je važno što HLR takođe садржи информацију о trenutnoj lokaciji претплатника. To jest, ако је mobilni korisnik trenутно у ромуингу у мрежи другог оператора, HLR садржи довољно информација за utvrđivanje (postupkom који ћемо uskoro opisati) adrese у posećenoj mreži на коју треба rutirati pozive mobilnom korisniku. Као што ћемо видети, када poziva mobilnog korisnika korespondent stupa у везу са posebnim komutatorom у матичној мрежи који се назива **GMSC (Gateway Mobile Services Switching Center)**. I ovde, da bismo smanjili broj upotrebljenih akronima GMSC ћемо
  - називати **матични MSC**.
- ◆ Posećena mreža održava базу података познату под именом регистар локација посетилача (*Visitor Location Register, VLR*). VLR садржи по једну ставку за сваког mobilnog korisnika који се *trenutno* налази у делу мреже који опслужује VLR. Stavke VLR-a се зато dodaju i izbacuju како mobilni korisnici ulaze u mrežu ili je napuštaju. VLR se obično налази на истом месту где и MSC (Mobile Switching Center) који координира uspostavljanje poziva prema posećenoj mreži i из ње.

U praksi, celularna mreža jednog posrednika igra ulogu matične mreže za svoje preplatnike, a posećene mreže za mobilne korisnike preplaćene kod drugog mobilnog operatera.

### 6.7.1 Rutiranje poziva prema mobilnom korisniku

Sada smo možemo da opišemo kako se poziva mobilni GSM korisnik u posećenoj mreži. Razmotrićemo sledeći jednostavan slučaj; složeniji scenariji opisani su u knjizi [Mouly 1992]. Kao stoje prikazano na slici 6.24, koraci su sledeći:

1. Korespondent bira telefonski broj mobilnog korisnika. Sam broj se ne odnosi ni na jednu konkretnu telefonsku liniju ni lokaciju (broj telefona je stalan, a korisnik je mobilan!). Početne cifre u broju su dovoljne za globalnu identifikaciju matične mreže mobilnog korisnika. Poziv se rutira od korespondenta kroz javnu telefonsku mrežu do matičnog MSC-a u matičnoj mreži mobilnog korisnika. Ovo je prvi krak poziva.
2. Matični mobilni komutacioni centar prima poziv i ispituje HLR kako bi utvrdio lokaciju mobilnog korisnika. U najprostijem slučaju HLR vraća roming broj



**Slika 6.24** ♦ Pozivanje mobilnog korisnika: indirektno rutiranje

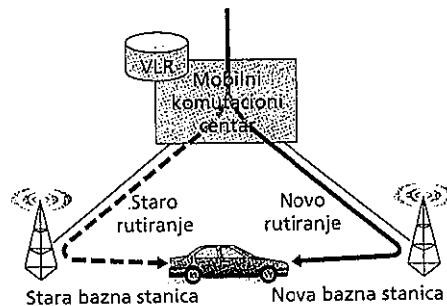
mobilne stanice MSRN (*Mobile Station Roaming Number*), koji ćemo zvati roming broj. Obratite pažnju na to da se ovaj broj razlikuje od trajnog broja telefona koji je povezan sa matičnom mrežom mobilnog korisnika. Roming broj je nepostojan: on se privremeno dodeljuje mobilnom korisniku prilikom ulaska u posećenu mrežu. Roming broj igra ulogu sličnu privremenoj adresi u mobilnom IP-u i isto kao COA nevidljiv je za korespondenta i za mobilnog korisnika. Ako HLR ne raspolaže roming brojem, on vraća adresu VLR-a u posećenoj mreži. U tom slučaju (koji nije prikazan na slici 6.24) matični MSC moraće da pošalje upit kako bi od VLR-a pribavio roming broj mobilnog čvora. Ali kako HLR uopšte pribavlja roming broj ili VLR adresu? Šta se događa sa ovim vrednostima kada mobilni korisnik pređe u neku drugu posećenu mrežu? Uskoro ćemo razmotriti ova značajna pitanja. 3. Kada-pribavi roming broj, matični MSC uspostavlja drugi krak poziva kroz mrežu do MSC-a u posećenoj mreži. Poziv je kompletiran pošto je rutiran od korespondenta do matičnog MSC-a odatle do posećenog MSC-a, a od njega do bazne stanice koja opslužuje mobilnog korisnika.

U koraku 2 ostaje nerazrešeno pitanje kako HLR pribavlja informaciju o lokaciji mobilnog korisnika. Kada se mobilni telefon uključi ili kada uđe u deo posećene mreže koji pokriva novi VLR, on mora da se prijavi posećenoj mreži. To se obavlja razmenom signalnih poruka između mobilnog telefona i VLR-a. Zatim posećeni VLR šalje poruku sa zahtevom za ažuriranje lokacije matičnom HLR-u mobilnog telefona. Ova poruka obaveštava HLR ili o roming broju na kojem se može stupiti u vezu sa mobilnim telefonom ili o adresi VLR-a (od kojeg se kasnije može dobiti broj mobilnog telefona). U sklopu ove razmene VLR takođe pribavlja od HLR-a preplatničke informacije o mobilnom korisniku i utvrđuje koje usluge mu treba omogućiti u posećenoj mreži.

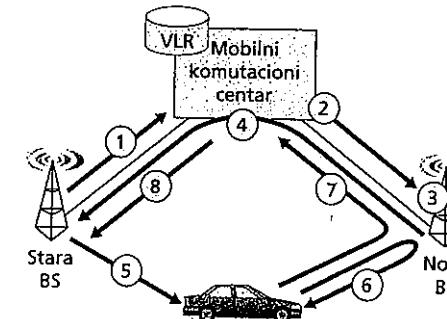
### 6.7.2 Predavanje u GSM-u

Kada mobilna stanica tokom poziva promeni pridruživanje od jedne bazne stanice na drugu kažemo da je došlo do predavanja. Kao što je prikazano na slici 6.25, mobilni poziv se na početku (pre predavanja) rutira mobilnom korisniku preko jedne bazne stanice (koju ćemo sada nazivati stara bazna stanica), a nakon predavanja se rutira preko druge bazne stanice (koju ćemo nazivati nova bazna stanica). Obratite pažnju na to da prilikom predaje među baznim stanicama ne dolazi samo do promene u komunikaciji između mobilne stanice i nove bazne stanice, već takođe i u preusmeravanju poziva između centralne u mreži i nove bazne stanice. Podimo od pretpostavke da stara i nova bazna stanica pripadaju istom MSC-u i da se preusmeravanje obavlja u tom MSC-u.

Do predavanja može doći iz više razloga, uključujući (1) ako signal između trenutne bazne stanice i mobilne stanice do te mere izgubi na kvalitetu da postoji opasnost od prekida poziva i (2) ako je celija postala preopterećena zbog velikog



**Slika 6.25** ♦ Scenario predavanja između baznih stanica sa zajedničkim MSC-om



**Slika 6.26** ♦ Koraci potrebeni za predavanje među baznim stanicama koje imaju zajednički MSC.

broja poziva. Ovo zagušenje može da se ublaži predavanjem mobilnih stanica manje zagušenim obližnjim celijama.

Dok je pridružena baznoj stanici, mobilna stanica povremeno meri jačinu signala za navodenje iz trenutne bazne stanice, kao i jačine signala za navodenje iz obližnjih baznih stanica koje može da „čeče“. Ova merenja se jednom ili dva puta u sekundi prenose trenutnoj baznoj stanici. U GSM-u predavanje pokreće stara bazna stanica na osnovu ovih merenja, trenutnog opterećenja mobilnim korisnicima u obližnjim celijama i drugih faktora [Mouly 1992]. Standard GSM ne određuje konkretni algoritam koji bazna stanica treba da primeni kako bi utvrdila da li da obavi predavanje ili ne.

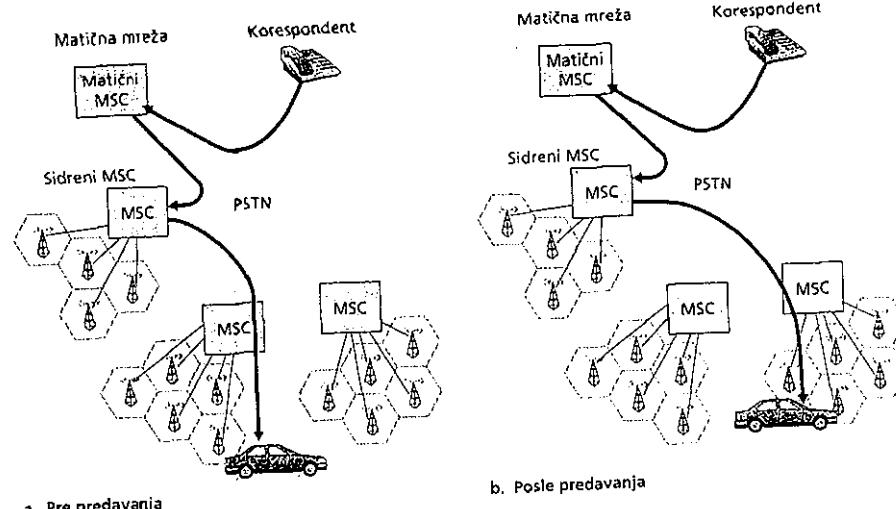
Na slici 6.26 prikazani su koraci po kojima bazna stanica odlučuje da preda mobilnog korisnika:

1. Stara bazna stanica (BS) obaveštava posećeni MSC da će se obaviti predavanje i navodi BS (ili skup BS-ova) kojima bi se mobilna stanica mogla predati.
2. Posećeni MSC inicira uspostavljanje putanje prema novoj BS, dodeljuje resurse potrebne za prenos preusmerenog poziva i obaveštava novu BS da će doći do predavanja.
3. Nova BS dodeljuje i aktivira radio kanal koji će koristiti mobilna stanica.
4. Nova BS odgovara posećenom MSC-u i staroj BS daje uspostavljena putanja od posećenog MSC-a do nove bazne stanice i daje potrebitno obavestiti mobilnu stanicu da će nastupiti predavanje. Nova BS obezbeđuje sve informacije potrebne mobilnoj stanicu za pridruživanje sa novom BS.
5. Mobilna stanica se obaveštava da treba da izvrši predavanje. Obratite pažnju na to da je sve do ovog trenutka mobilna stanica bila u blaženom neznanju da je mreža pripremala teren za predavanje (tj. dodeljivala kanal novoj baznoj stanicu i dodeljivala putanju od posećenog MSC-a do nove BS).

6. Mobilna i nova BS razmenjuju jednu ili više poruka da bi se potpuno aktivirao novi kanal u novoj BS.
7. Mobilna i nova BS razmenjuju jednu ili više poruka da bi se potpuno aktivirao novi kanal u novoj BS.
8. Mobilna stanica šalje novoj baznoj stanicu poruku da je predavanje dovršeno, a nova BS prosledjuje tu poruku posećenom MSC-u. Posećeni MSC zatim preu-smerava poziv koji je u toku mobilnoj stanicu preko nove BS.
9. Zatim se oslobadaju resursi na putanji prema staroj baznoj stanicu.

Zaključimo ovaj opis predavanja razmatranjem događaja do kojih dolazi kada se mobilna stanica premesti na baznu stanicu koja pripada *različitom* MSC-u od stare bazne stanice i šta se događa kada više puta dolazi do takvog predavanja među MSC-ovima. Kako je prikazano na slici 6.27, u GSM-u je defmisan pojam **sidrenog** MSC-a. Sidreni MSC je onaj MSC u kome se mobilna stanica nalazila na početku poziva; sidreni MSC se dakle ne menja tokom celog poziva. Za svo vreme dok poziv traje i bez obzira na broj prelazaka među različitim MSC-ovima poziv se rutira od matičnog MSC-a u sidreni MSC, a zatim od sidrenog MSC-a u posećeni MSC gde se mobilna stanica trenutno nalazi. Kada mobilna stanica prelazi iz područja koje pokriva jedan MSC u drugo područje, poziv koji je u toku prerutira se iz sidrenog MSC-a u novi posećeni MSC koji sadrži novu baznu stanicu. Prema tome, između korespondenta i mobilne stanice nalazi se najviše 3 MSC-a (matični MSC, sidreni MSC i posećeni MSC). Na slici 6.27 prikazano je rutiranje poziva među MSC-ovima koje mobilni korisnik posećuje.

Alternativa ovakvom povezivanju gde se zadržava samo jedan skok od sidrenog MSC-a do trenutnog MSC-a moglo bi biti zadržavanje lanca posećenih MSC-ova tako što bi stari MSC prosledio poziv koji je u toku novom MSC-u svaki put kada se mobilna stanica premesti. Do takvog ulančavanja MSC-a zaista i dolazi u celu-



**Slika 6.27** ♦ Prerutiranje preko sidrenog MSC-a

ljam mrežama 1S-4! uz opcionalno minimiziranje koraka putanje kojim se uklanjaju MSC-ovi između sidrenog i trenutno posećenog MSC-a [L-in 2001].

Zaključimo sada opis upravljanja GSM mobilnošću jednim poređenjem upravljanja mobilnošću u GSM-u i mobilnom IP-u. Poređenje u tabeli 6.2 ukazuje na to da i pored fundamentalnih razlika između IP-a i celularnih mreža oni imaju iznenadujuće mnogo sličnosti i zajedničkih funkcionalnih elemenata za podršku mobilnosti.

## 6.8 Bežične mreže i mobilnost: posledice po protokole viših nivoa

U ovom poglavljiju ćemo videti da se bežične mreže značajno razlikuju od ožičenih kako u sloju veze (zbog karakteristika bežičnih kanala kao što su opadanje, višestruke putanje i sakriveni terminali) tako i u mrežnom sloju (zbog mobilnih korisnika koji menjaju tačke priključka sa mrežom). Da li postoje značajne razlike i u transportnom i aplikacijskom sloju? Moglo bi se pomisliti da će te razlike biti minimalne pošto mrežni sloj pruža višim slojevima isli model usluge najboljeg mogućeg prenosa u ožičenim i u neožičenim mrežama.

Slično tome, ako se i u ožičenim i u bežičnim mrežama za pružanje usluga aplikacijama u transportnom sloju koriste protokoli kao što su TCP ili UDP, tada ni

| Element GSM-a                                                | Komentar o elementu GSM-a                                                                                                                                                                                                                        | Element mobilnog IP-a |
|--------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| Matični sistem                                               | Mreža kojoj pripada trojni broj telefona mobilnog korisnika                                                                                                                                                                                      | matična mreža         |
| GMSC ili prosti matični MSC, registar matične lokacije (HLR) | Matični MSC: kontakt za pribavljanje rukovitne adrese mobilnog korisnika. HLR: baza podataka u matičnom sistemu koja sadrži trojni telefonski broj, informacije o profilu, trenutnu lokaciju mobilnog korisnika, preplatičke informacije.        | matični agent         |
| Posećeni sistem                                              | Mreža različita od matičnog sistema u kom se mobilni korisnik trenutno nalazi                                                                                                                                                                    | posećena mreža        |
| Posećeni mobilna centrala, VLR – registar lokacije posećenca | Posećeni MSC: zadužen za uspostavljanje poziva od mobilnih čvorova i prema njima u celijama koje pripadaju MSC-u. VLR: privremeni zapis u bazi podataka posećenog sistema, sadrži preplatičke informacije za svakog mobilnog korisnika u poseti. | strani agent          |
| MSRN (roming broj mobilne stanice) ili prosti roming broj    | Rukovitna adresa za segment telefonskog poziva između matičnog MSC-a i posećenog MSC-a, nije vidljiva ni mobilnoj stanici ni korespondentu                                                                                                       | COA                   |

**Tabela 6.2** ♦ Zajednički elementi mobilnog IP-a i mobilnosti u GSM-u

aplikacijski sloj ne mora da se menja. Na neki način intuitivna logika je tačna - TCP i UDP mogu da funkcionišu i u mrežama sa bežičnim linkovima. S druge strane, transportni protokoli uopšte, a posebno TCP, ponekad imaju veoma različite performanse u ožičenim i bežičnim mrežama. Pogledajmo zašto.

Setite se da TCP ponovo šalje segment koji se na putanji od pošiljaoca do primaoca izgubio ili oštetio. U slučaju mobilnih korisnika gubici mogu da nastanu bilo zbog zagušenja mreže (prekoračenja privremenih memorija u ruterima) ili zbog predavanja (npr. zbog odlaganja u preusmeravanju segmenata prema novoj pristupnoj tački mobilne stanice sa mrežom). U svim tim slučajevima, ACK od primaoca ka pošiljaocu u TCP-u ukazuje samo na to da segment nije stigao neoštećen; pošiljalac ne zna da li je segment izgubljen zbog zagušenja, tokom predavanja ili zbog toga što su otkrivene greške u bitovima. U svim slučajevima pošiljalac reaguje jednak - ponovo šalje segment. TCP-ova kontrola zagušenja takođe reaguje isto u svim slučajevima - TCP smanjuje prozor zagušenja kao što je opisano u odeljku 3.7. Bezuslovnim smanjivanjem prozora zagušenja TCP implicitno prepostavlja da do gubitka segmenata dolazi zbog zagušenja, a ne zbog oštećenja ili predavanja. U odeljku 6.2 videli smo da su greške u bitovima daleko češće u bežičnim mrežama nego u ožičenim. Kada dođe do takvih grešaka ili do gubitka zbog predavanja, zaista nema razloga da TCP pošiljalac smanji prozor zagušenja (i tako smanji brzinu slanja). Zaista, sasvim je moguće da su pomoćne memorije u ruterima prazne i da zagušenje ne omela pakete sa kraja na kraj putanje.

Istraživači su početkom '90-tih utvrdili da zbog visoke učestalosti grešaka u bitovima na bežičnim linkovima i verovatnoće gubitaka prilikom predavanja, kontrola zagušenja u TCP-u može u bežičnom okruženju da postane problematična. U rešavanju ovog problema postoje dve široke klase pristupa [Balakrishnan 1995]:

- ◆ *Lokalni oporavak.* Kod lokalnog oporavka cilj je da se oporavak od grešaka u bitovima obavi na licu mesta i u trenutku gde do njih dolazi (npr. u bežičnom linku). To znači da su potrebni (1) protokoli koji oporavljaju od gubitaka i oštećenja u sloju veze (npr. ARQ protokol 802.11 koji smo proučili u odeljku 6.3 ili složeniji pristupi koji koriste i ARQ i FEC [Avanoglu 1995]); (2) protokoli transportnog sloja koji dele TCP konekciju na dva segmenta, jedan od izvora do bežičnog linka, a drugi od bežičnog linka do odredišta [Bakre 1995; Brovnn 1997]; i (3) protokoli sloja veze svesni TCP-a [Balakrishnan 1995; Liu 2002],
- ◆ *TCP pošiljalac svestan bežičnih linkova.* U pristupu sa lokalnim oporavkom, TCP pošiljalac živi u blaženom neznanju da neki njegovi segmenti koriste bežični link. Alternativni pristup jeste da TCP pošiljalac i primalac budu svesni postojanja bežičnog linka, da razlikuju gubitke uzrokovane zagušenjem u ožiče-noj mreži od oštećenja i gubitaka do kojih dolazi u bežičnom linku, pa da pozivaju kontrolu zagušenja samo u slučaju gubitaka na zagušenoj ožičenoj mreži. U knjizi [Balakrishnan 1995] istražuju se različiti tipovi TCP-a pod pretpostavkom da krajnji sistemi budu u stanju da otkriju tu razliku. U knjizi [Wei 2004] istražuju se tehnike za otkrivanje razlika između gubitaka na ožičenim od gubitaka na bežičnim delovima putanje sa kraja na kraj.

Ovde je naš opis TCP-a preko bežičnih linkova morao da bude kratak. Konsultujte reference gde ćete naći detalje o ovom području koje se još uvek istražuje.

Pošto smo razmotrili protokole transportnog sloja, razmotrimo sada na koji način bežične veze i mobilnost utiču na protokole aplikacijskog sloja. Ovde je značajno naglasiti da bežični linkovi često imaju relativno male propusne opsege kao što smo videli na slici 6.2. Zbog toga aplikacije koje rade preko bežičnih linkova, pogotovo preko celularnih bežičnih linkova, moraju tretirati propusni opseg kao deficitaran resurs. Na primer, veb server koji šalje sadržaj čitaču koji se izvršava na telefonu 3G verovatno neće moći da obezbedi isti bogat sadržaj koji šalje čitaču preko ožičene konstrukcije. Mada bežični linkovi postavljaju izazove pred aplikacijskim slojem, mobilnost koju omogućavaju takođe omogućava široku paletu aplikacija zavisnih od konteksta i od lokacije [Chen 2000]. Uopšteno govoreći, bežične i mobilne mreže imaće ključnu ulogu u realizaciji sveprisutnih računarskih okruženja budućnosti [Weiser 1991]. Može se reći da smo videli samo vrh ledenog brega kadaje reč o uticaju bežičnih i mobilnih mreža na mrežne aplikacije i njihove protokole!

## 6.9 Rezime

Bežične i mobilne mreže predstavljaju revoluciju u telefonijima i imaju sve veći uticaj u svem računarskim mrežama. Svojim pristupom globalnoj mrežnoj infrastrukturi u svakom trenutku na svakom mestu bez ikakvog zadržavanja, ne samo što omogućavaju sveprisutni pristup mreži, već omogućavaju jedan nov i uzbudljiv skup usluga zavisno od lokacije. S obzirom na sve veći značaj bežičnih komunikacija i mreža ovo poglavje govori o principima, tehnologijama zajedničkih linkova i mrežnim arhitekturama za podršku bežične i mobilne komunikacije.

Poglavlje smo započeli jednim uvodom u bežične i mobilne mreže, povlačeći značajnu granicu između zadataka koje nameće *bežična* priroda komunikacionih linkova u tim mrežama i onih koji su posledica *mobilnosti* koje ti bežični linkovi omogućavaju. Na taj način smo bolje razdvojili, prepoznali i savladali ključne koncepte svakog od ovih područja. Prvo smo se usredosredili na bežičnu komunikaciju, razmotrili karakteristike bežičnog linka u odeljku 6.2. U odeljcima 6.3 i 6.4 razmotrili smo aspekte u sloju veze IEEE standarda bežičnog LAN-a 802.11 (Wi-Fi) i celularni pristup Internetu. Zatim smo obratili pažnju na pitanje mobilnosti. U odeljku 6.5 ukazali smo na nekoliko oblika mobilnosti od kojih svaki postavlja različite izazove i dozvoljava različita rešenja. Ispitali smo probleme lociranja i rutiranja prema mobilnom korisniku, kao i različite pristupe predavanja mobilnog korisnika koji se dinamički kreće od jedne pristupne tačke sa mrežom do druge. Ispitali smo kako su ta pitanja rešena u standardu za mobilni IP u odeljku 6.6 odnosno u GSM-u u odeljku 6.7. Na kraju, u odeljku 6.8 razmatrali smo uticaj bežičnih linkova i mobilnosti na protokole transportnog sloja i na mrežne aplikacije.

Mada smo posvetili celo poglavje studiranju bežičnih i mobilnih mreža, za potpun pregled ovog uzbudljivog područja koje se BTZo širi bila bi potrebna cela knjiga (ili više njih). Da biste detaljnije istražili ovo polje, poslužite se brojnim referencama pomenutim u ovom poglavljju.

## Domaći zadatak: problemi i pitanja

### Poglavlje 6 • Kontrolna pitanja

1. Opišite ulogu okvira za navođenje u protokolu 802.11.
2. Opišite metode dostupne za proveru autentičnosti korisnika u mrežama 802.11.
3. Tačno ili netačno: Pre nego što stаницa 802.11 pošalje okvir podataka, ona mora prvo da pošalje RTS okvir i primi odgovarajući CTS okvir.
4. Zašto se potvrde koriste u protokolu 802.11, a ne i u ožičenom Internetu?
5. Tačno ili netačno: Ethernemet i 802.11 koriste istu strukturu okvira.

6. Opišite kako funkcioniše RTS prag.
7. Kada bi RTS i CTS okviri protokola IEEE 802.11 bili dugački kao i standardni DATA i ACK okviri, da li bi CTS i RTS okviri bili od ikakve koristi? Zašto?
8. U odeljku 6.3.4 opisana je mobilnost u protokolu 802.11 gde se bežična stanica premešta sa jedne bazne stanice na drugu unutar iste podmreže. Kada su pristupne tačke međusobno povezane komutatorom, pristupna tačka ponekad mora da pošalje okvir sa lažnom MAC adresom kako bi komutator pravilno prosledio okvire. Zašto?
9. Naučili smo u odeljku 6.3.2 da postoje dva glavna 3G standarda: UMTS i CDMA-2000. Od kojih standarda druge i 2,5 generacije potiču ta dva standarda?

## Problemi

1. Razmotrite primer sa jednim CDMA pošiljaocem sa slike 6.4. Šta bi bio izlaz pošiljaoca (za prikazana 2 bita podataka) ako bi CDMA kod pošiljaoca bio  $(1, -1, 1, -1, 1, -1 = 1, -1)$ ?
2. Razmotrite pošiljaoca 2 na slici 6.5. Staje izlaz pošiljaoca prema kanalu (pre nego što se doda na signal od pošiljaoca 1),  $Z_{...}$ ?
3. Prepostavite da primalac sa slike 6.5 želi da primi podatke koje šalje pošiljalac 2. Dokažite (računski) da primalac zaista može da izdvoji podatke pošiljaoca 2 iz signala agregatnog kanala tako što će upotrebiti kod pošiljaoca 2.
4. Prepostavite da postoje 2 posrednika za Internet usluge koja nude Wi-Fi pristup u određenom kafeu tako što svaki posrednik ima vlastitu pristupnu tačku i vlastiti blok IP adresa.
  - a. Prepostavite dalje da su oba posrednika slučajno konfigurisala pristupne tačke tako da rade na kanalu 11. Da li će protokol 802.11 u ovoj situaciji potpuno da otkaže? Opišite šta bi se desilo kada bi dve stanice od kojih je svaka pridružena drugom posredniku istovremeno pokušale da emituju.
  - b. Sada prepostavite da jedna pristupna tačka radi na kanalu 1, a druga na kanalu 11.
5. U koraku 4 protokola CSMA/CA stanica koja uspešno prenese jedan okvir započinje protokol CSMA/CA za sledeći okvir od koraka 2 umesto od koraka 1. Koji su razlog imali projektanti protokola CSMA/CA da ne dozvole takvoj stanici da odmah počne, sa slanjem drugog okvira (ako otkrije daje kanal sloboden)?
6. Prepostavite daje stanica 802.11b konfigurisana tako da uvek rezerviše kanal nizom poruka RTS/CTS. Prepostavite da ta stanica iznenada želi da pošalje 1 000 bajtova podataka i da su u tom trenutku sve druge stanice neaktivne.

- Zanemarujući kašnjenje zbog propagiranja i pod pretpostavkom da nema grešaka bitova izračunajte vreme potrebno da se prenese okvir i primi potvrda (kao funkciju SIFS i DIFS).
7. U odeljku 6.5 jedno od predloženih rešenja koje je omogućavalo mobilnim korisnicima da zadrže IP adresu prilikom kretanja po stranim mrežama bilo je da strana mreža objavljuje sasvim konkretnu mrežu do mobilnog korisnika i iskoristi postojeću infrastrukturu rutiranja za propagiranje ove informacije po celoj mreži. Uočili smo da bijedan od problema mogao biti skalabilnost. Prepostavite da kad mobilni korisnik prede iz jedne mreže u drugu nova strana mreža objavi konkretnu rutu do mobilnog korisnika, a da stara strana mreža povuče svoju rutu. Razmotrite kako se informacija za rutiranje propagira u algoritmu vektora rastojanja (pogotovo za slučaj međudomenskog rutiranja među mrežama koje pokrivaju celu zemlju).
    - a. Da li će drugi ruteri moći da rutiraju datagrame u novu stranu mrežu čim strana mreža počne da objavljuje tu rutu?
    - b. Da li je moguće da razni ruteri smatraju da mobilni korisnik pripada različitim stranim mrežama?
    - c. Opišite vremenski raspon u kojem će drugi ruteri u mreži konačno saznati putanje prema mobilnim korisnicima.
  8. Prepostavite daje korespondent sa slike 6.17 mobilan. Skicirajte dodatnu infrastrukturu u mrežnom sloju koja bi bila potrebna da se datagram rutira od prvobitno mobilnog korisnika do korespondenta (koji je sada mobilan). Prikažite strukturu datagrama između prvobitno mobilnog korisnika i sada mobilnog korespondenta kao na slici 6.18.
  9. U mobilnom IP-u, kakav bi efekat imala mobilnost na kašnjenje datagrama sa kraja na kraj između izvora i odredišta?
  10. Razmotrite primer sa ulančavanjem opisan na kraju odeljka 6.7.2. Prepostavite da mobilni korisnik posećuje strane mreže A, B i C, a da korespondent započne konekciju sa mobilnim korisnikom kada se on nalazi u stranoj mreži A. Navedite redosled poruka među stranim agentima i između stranih agenata i matičnog agenta kako se mobilni korisnik kreće iz mreže A u mrežu B, pa u mrežu C. Zatim, prepostavite da nema ulančavanja, a da korespondent (kao i matični agent) mora eksplicitno da se obavesti o promenama adrese COA mobilnog korisnika. Navedite redosled poruka koje bi morale da se razmene u ovom drugom scenaruju.
  11. Razmotrite dva mobilna čvora u stranoj mreži sa stranim agentom. Da li je u mobilnom IP-u moguće da ta dva mobilna čvora koriste istu adresu COA? Objasnite.pdgovor. .
  12. U našem opisu kako je VLR ažurirao HLR informacijom o trenutnoj lokaciji mobilne stanice, koje su prednosti i mane da se u HLR evidentira MSRN, a ne VLR?

### Teze za diskusiju

1. Navedite 5 proizvoda na današnjem tržištu koji sadrže interfejs Bluetooth ili 802.15.
2. Da li su u vašoj okolini dostupne bežične usluge 3G? Kakve su cene? Koje se aplikacije podržavaju?
3. Koje ste probleme uočili kao korisnik protokola IEEE 802.11? Kako bi dizajn standarda 802.11 mogao da se razvije da bi se ti problemi prevazišli?

### Laboratorija Ethereal

Na veb lokaciji koja pripada ovom udžbeniku: <http://www.awl.com/kurose-ross> naći ćete Ethereal vežbu za ovo poglavlje koja hvata i ispituje okvire 802.11 koje razmenjuju bežični laptop i jedna pristupna tačka.

v

## Carli Perkins

Caris E. Perkins je istraživač najvišeg strukovnog ranga u Laboratoriji za komunikacione sisteme Nokijrnog Istraživačkog centra, gde se bavi mobilnim bežičnim umrežavanjem i dinamičkim konfiguracionim protokolima. On je izdavač nekoliko ACM i IEEE časopisa za područja vezana za bežično umrežavanje. Obavlja funkciju redaktora dokumenata za radnu grupu Mobile-IR u IETF-u. On je pisac i saradnik na izradi standarda u radnim grupama mobileip, manet, IPv6 i seamoby (Seamless Mobility). Takođe je među izdavačima časopisa *Mobile Communications and Computing*, zvanične publikacije interesne grupe ACM SIGMOBIE, □ radio je i u uredništvu časopisa *IEEE Internet Computing*, Carli je napisao i redigovao knjige o mobilnom iP-u i od *hoc* umrežavanju, a objavio je niz dokumenata i članaka koji su dobili nagrade u područjima mobilnog umrežavanja, *ad hoc* umrežavanja, opfimizacije ruta za mobilno umrežavanje, otkrivanje resursa i automatskog konfigurisanja za mobilne računare.

Zašto ste odlučili da se specijalizujete za bežično umrežavanje i mobilnost?

### j

Moja povezanost sa bežičnim umrežavanjem i mobilnošću prirodno proističe iz rada na projektima u IBM-ovom istraživačkom centru krajem '80-tih. Imali smo radio-linkove i pokušavali da napravimo uređaj u stilu „ThinkPad“ (kao neki Palm Pilot) sa bežičnim povezivanjem i prepoznavanjem rukopisa.

Napravili smo jednostavno reSenje (koje je kasnije dobilo ime „Mobile IP“) i primetili da funkcioniše. To je, naravno, neobično u poređenju sa većinom istraživačkih rešenja. Na osnovu iskustva sa mobilnim IP-om napravili smo brzu i efikasnu modifikaciju za RIP koja je mogla da se izvršava *ad hoc*. I to je prilično dobro funkcionalo. Pod „funkcionisanjem“ podrazumevam da su aplikacije pravilno radile bez ikakvih izmena i da mreža nije posrfala pod našim novim dizajnom. To se naziva svojstvima „transparentnost u odnosu na aplikacije“ i „skalabilnost“.

### Koji vam je bio prvi posao u računarskoj industriji?

Radio sam u firmi TRW Controls u Hjoustonu (Teksas). To je bila drastična promena u odnosu na univerzitetске studije.

Na tom poslu sam utvrdio koliko je loš softver za podršku Čak i najvažnijih pomoćnih kontrolnih sistema. Ti su sistemi namenjeni kontroli električnog toka u velikim snažnim mrežama, a softver koji ih podržava bio je napravljen tako da vam se kosa digne na glavi. Osim toga, rokovi su uvek bili kratki, a programeri su cinično komentarisali namere uprave i svoje ustove rada. Ceo sistem je trebalo preuređiti iz osnova. Nisam ubeden da su se stvari promenile za ovih 30 godina, pogotovo kad se setimo raspada električnog sistema 2003. U

## Kakva je vaša vizija budućnosti multimedijskog umrežavanja?

Mi smo sada u prelaznoj fazi, na samo nekoliko godina od trenutka kada će IP postati univerzalna platforma za multimedijске usluge. Očekujemo da će radio, telefon i TV raditi čak i za vreme snežnih oluja i zemljotresa, tako da će korisnici, kada Internet preuzme ulogu ovih namenskih mreža, imati isti nivo pouzdanosti.

Istraživanja umrežavanja se u izvesnoj meri suočavaju sa problemom koji su istraživanja operativnih sistema imala godinama, ili čak i više. Dok je još uvek moguće sprovoditi „butik“ istraživanja operativnih sistema u malim zajednicama, pokretanje odvojene mreže u velikoj meri nema svrhe. Moraćemo da naučimo da projektujemo mrežne tehnologije za međusobno povezivanje konkurentnih nosilaca, sa mnogo krajnjih korisnika koji su neznalice ili zlonamemi.

Vidljiva napredovanja u multimedijskim mrežama pokreća činoci koji se nalaze izvan same ove oblasti, kao što su dostignuća u brzinama pristupa i okosnice, kao i jeftina računarska snaga.

## Zašto SIP ima budućnost koja obećava?

Kako se danas nastavlja nadgrađivanje bežičnih mreža na 3G mreže, postoji nuda za proširivanje jednog multimedijskog signalnog mehanizma na sve vrste mreža, od kablovskog modela do korporativnih telefonskih i javnih bežičnih mreža. Zajedno sa softverskim radi-om, to će u budućnosti omogućiti da jedan uređaj može da se koristi u kućnoj mreži, kao bežični Bluetooth telefon, u korporativnoj mreži preko 802.11 i u globalnom području preko 3G mreža. Čak pre nego što budemo imali takav univerzalni bežični uređaj, mehanizmi ličnih mobilnih aplikacija omogućavaju da se sakriju razlike između mreža. Jedan identifikator postaje univerzalno sredstvo da se dođe do neke osobe, i zamjenjuje pamćenje ili pro-laženje preko pola tuceta tehnologija ili telefonskih brojeva koji zavise od lokacije.

SIP takođe razdvaja prenos glasa od govornih usluga. Sada postaje tehnički moguće da se odvoji od monopolna lokalne telefonije, gde jedna kompanija obezbeđuje neutralni bitski prenos, dok druge obezbeđuju IP „tonske biranje“ i klasične telefonske usluge kao što su mrežni prolazi, posleđivanje poziva i identifikacija poziva.

Pored multimedijskih signalizacija, SIP nudi novu uslugu koja je nedostajala na Internetu: beleženje događaja. Mi smo aproksimirali takve usluge pomoću HTTP improvizacija i elektronske pošte, ali to nikada nije bilo baš zadovoljavajuće. Kako su događaji uobičajena apstrakcija za distribuirane sisteme, ovo može da uprosti konstrukciju novih usluga.

## Imate li neki savet za studente koji ulaze u oblast umrežavanja?

Umrežavanje je gotovo klasična disciplina premoćavanja. Ono ima korene u elektrotehnici, računarskim naukama, operacionim istraživanjima i drugim disciplinama. Dakle, istraživači umrežavanja treba da poznaju predmete koji su daleko izvan njihove osnovne oblasti.

Rad na mrežama može da pruži ogromno zadovoljstvo, zato što dopusta ljudima da komuniciraju i razmenjuju ideje, što je jedna od osnovnih crta ljudskosti. Mnoga područja inženjerstva već su dosegla najviši nivo svog razvoja; automobili, vozovi i avioni još uvek uglavnom rade ono što su radili pre 20 ili više godina i verovatno neće ići mnogo brže u siedecih 20. Glavna promena biće u sposobnosti da ti entiteti komuniciraju i tako poboljšaju svoju performansu, postavši sigurniji, izbegavajući saobraćajna zagušenja.



# Bezbednost u računarskim mrežama

Predstavljamo vam Alisu i Boba, dvoje ljudi koji žele da komuniciraju i da to rade „bezbedno“. Imajući u vidu da je ovo knjiga o umrežavanju, trebalo bi da primetimo kako bi Alisa i Bob mogli da budu dva rutera koji žele da bezbedno razmene tabele rutiranja, klijent i server koji žele da uspostave bezbednu transportnu konekciju, ili dve aplikacije elektronske pošte koje žele da razmenjuju bezbednu elektronsku poštu - što su sve slučajevi koje ćemo razmatrati kasnije u ovom poglavlju. Alisa i Bob su dobro poznati u zajednici koja se bavi bezbednošću, možda zato što su njihova imena veselija od generičkog entiteta zvanog „A“ koji želi da bezbedno komunicira sa generičkim entitetom zvanim „B“ - Zabranjene ljubavne veze, komunikacija u toku rata i poslovne transakcije obično se navode kao ljudske potrebe za bezbednim komuniciranjem; pošto više volimo prvu situaciju u odnosu na druge dve, srećni smo da koristimo Alisu i Boba kao našeg pošiljaoca i primaoca, i da ih zamislimo u ovom prvom scenaru.

Rekli smo da Alisa i Bob žele da komuniciraju i da to rade „bezbedno“, ali Šta to tačno znači? Kao Što ćemo videti, bezbednost je (kao i ljubav) šarolika stvar; odnosno, ima mnogo aspekata bezbednosti. Naravno, Alisa i Bob bi voledi da sadržaj njihove komunikacije ostane tajna za prisluškivače (recimo, za ljubomornu suprugu). Oni bi takođe voledi da osiguraju da, kada komuniciraju, oni zaista to čine jedno sa drugim i da, ako se u njihovu komunikaciju mesa neki prisluškivač, to mešanje bude otkriveno. U prvom delu ovog poglavlja, objasnićemo tehnikе koje dozvoljavaju

šifrovanje/dešifrovanje komunikacije, autentifikovanje strane sa kojom se komunicira i osiguranje integriteta poruke.

U drugom delu ovog poglavlja, razmotrićemo činjenicu da krajnje tačke u komunikaciji rade u čvorovima unutar preduzeća ili kućne mreže, na sličan način kao što bi Alisa i Bob mogli da žive u kućama iz kojih bi komunicirali slanjem pošte. Videćemo da mreža (kao kuća) može da bude ranjiva na napade, ali da može da preduzme protivmere kako bi se zaštitiла od "loših momaka" koji nesumnjivo postoje. Zato ćemo objasniti i mrežne barijere, koje obezbeđuju izvestan stepen izolacije i zaštite od onih koji se nalaze izvan nečije mreže. Razmotrićemo i razne oblike napada na mrežnu infrastrukturu, kao i protivmere koje se preduzimaju da bi se uklonile, ili bar ublažile posledice tih napada. Poglavlje zaključujemo studijama primera bezbednosti u aplikaciji, transportu, mreži i slojevima linka.

## 8.1 Šta je bezbednost mreže

Počinimo proučavanje bezbednosti mreže vrativši se našim ljubavnicima, Alisi i Bobu, koji žele da komuniciraju „bezbedno“. Šta to tačno znači? Svakako, Alisa želi da samo Bob može da razume poruku koju je ona poslala, čak i ako oni komuniciraju preko „nebezbednog“ medijuma na kojem uljez (Trudi) može da presretne i čita sve što se prenosi od Alise do Boba, kao i da to računarski obradi. Bob takođe želi da bude siguran da je poruku koju prima od Alise ona zaista i poslala, a Alisa želi da bude sigurna kako sadržaj njihovih poruka nije promenjen u prenosu. Imajući u vidu navedene zahteve, možemo da identifikujemo sledeće poželjne osobine bezbe-dne komunikacije:

- ♦ **Tajnost.** Trebalо bi da samo pošiljalac i nameravani primalac razumeju sadržaj prenesene poruke. S obzirom na to da prislушkivači mogu da presretnu poruku, to obavezno zahteva da ona bude nekako šifrovana (njeni podaci maskirani), tako da presretnutu poruku prislushkivač ne može da dešifruje (razume). Ovaj aspekt tajnosti je verovatno najčešće podrazumevano značenje termina „bezbe-dna komunikacija“. Zapazite, međutim, da to ne samo daje ograničena definicija bezbedne komunikacije (kasnije ćemo navesti dodatne aspekte bezbedne komunikacije), nego i ograničena definicija same tajnosti. Na primer, Alisa bi takođe mogla da poželi da sama činjenica da ona komunicira sa Bobom (ili vreme ili učestalost njene komunikacije) bude tajna! Kriptografske tehnike za šifrovanje i dešifrovanje podataka proučićemo u odeljku 8.2. Videćemo da se tajna komunikacija Često oslanja na jedan ključ ili na više ključeva koji se koriste za šifrovanje/dešifrovanje. Temu distribucije ovih ključeva obradićemo u odeljku 8.5.

- ♦ **Autentifikacija.** I pošiljalac i primalac trebalo bi da mogu da potvrde identitet druge strane u komunikaciji - odnosno, da potvrde kako je druga strana zaista onaj ili ono što tvrdi da jeste. Ljudska komunikacija licem u lice taj problem lako rešava vizuelnim prepoznavanjem. Kada entiteti koji komuniciraju razmenjuju poruke preko medijuma gde ne mogu da „vide“ drugu stranu, autentifikacija nije tako jednostavna. Zašto biste, na primer, verovali daje poruka elektronske pošte, koja sadrži deo teksta u kome se kaže daje ona došla od vašeg prijatelja, zaista došla od vašeg prijatelja? Ako vas neko pozove preko telefona tvrdeći daje iz vaše banke i traži broj vašeg računa, tajni lični identifikacioni broj (PIN) i bilanse računa radi verifikacija, da li biste mu dali te informacije preko telefona? Nadajmo se da ne biste! Ispitaćemo tehnike autentifikacije u odeljku 8.3, uključujući više njih koje se, možda iznenadjuće, takođe oslanjaju na kriptografske tehnike iz poglavlja 8.2.

- ♦ **Integritet i neporecivost poruke.** Čak i kada pošiljalac i primalac mogu da autentikuju jedan drugog, oni takođe žele da obezbede da se sadržaj njihove komunikacije ne menja u prenosu, bilo zlonamerno ili slučajno. Proširenja tehnika kontrolnog zbirka koje smo sreli u pouzdanom transportu i protokolima linka podataka mogu da se iskoriste da bi se obezbedio takav integritet poruka. Što je tema koju ćemo proučavati u odeljku 8.4. Videćemo takođe kako primalac može da „dokaže“ daje poruka morala da stigne od najavljenog pošiljaoca. Videćemo da se i integritet poruke i neporecivost takođe oslanjaju na kriptografske koncepte iz odeljka 8.2.

- ♦ **Raspoloživost i kontrola pristupa.** Potrebu za mrežnom bezbednošću su na bolan način u poslednjih nekoliko godina potencirali brojni napadi odbijanja usluga (*denial-of-service*, DoS), koji su za legitimne korisnike izbacili iz upotrebe mrežu, računar ili neki drugi deo mrežne infrastrukture; možda najčuveniji od tih DoS napada bili su oni protiv veb lokacija velikih kompanija. Dakle, ključni zahtev za bezbednu komunikaciju je da ona pre svega može da se odvija - da „loši momci“ ne mogu da spreče da legitimni korisnici koriste infrastrukturu. Činjenica da neki korisnici mogu da budu legitimni dok drugi to nisu, prirodno dovodi do pojma kontrole pristupa - što obezbeduje da se entitetima koji traže da dobiju pristup resursima to dozvoljava samo ako imaju odgovarajuća pristupna prava i ako pristupaju na dobro definisan način.

Tajnost, autentifikacija, integritet poruka i neporecivost se već izvesno vreme smatraju ključnim komponentama bezbedne komunikacije [McCumber 1991]. Raspoloživost i kontrola pristupa su novija proširenja pojma bezbedne komunikacije [Maconachy 2001], bez sumnje motivisana stvarnim poslovima na obezbeđivanju mrežne infrastrukture od potencijalnih napada „loših momaka“. Jedan od najsigurnijih načina da se obezbedi da „loši momci“ ne mogu da naprave nikakvu štetu je da se njihovi paketi ne puste ni da uđu u mrežu. Mrežna barijera (*firewall*) je uredaj između mreže koja treba da se štiti („nas“) i ostatka sveta („loših momaka“) -

,njih'). Ona kontroliše pristup ka mreži i slanje od nje, regulišući koji paketi mogu da produ unutar nečije mreže i izadu van nje. Mrežne barijere su broj postale uobičajena komponenta u mrežama, u opsegu od malih kućnih mreža do onih koje pripadaju najvećim korporacijama na svem. Ispitaćemo kako mrežne barijere obezbeđuju kontrolu pristupa (i po paketima i po usluzi) u odeljku 8.6.

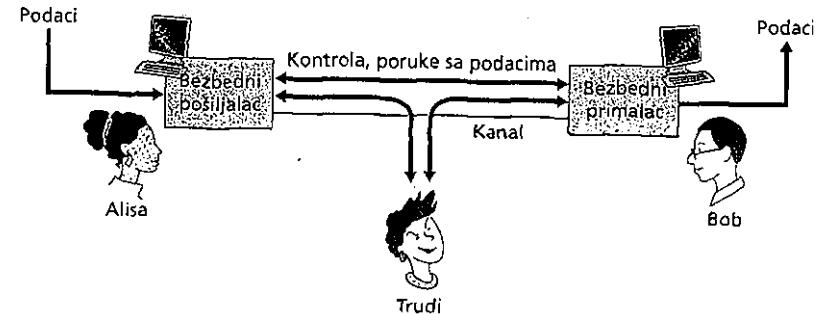
Navedena definicija bezbedne komunikacije usredstvena je prvenstveno na *zaštitu* komunikacije i mrežnih resursa. U praksi, mrežna bezbednost obuhvata ne samo zaštitu, nego i *detektovanje* kršenja bezbedne komunikacije i napada na infrastrukturu, a zatim i *odgovor* na te napade. U mnogo slučajeva, u odgovoru na napade, administrator mreže može da primeni dodatne zaštitne mehanizme. U tom smislu, mrežna bezbednost se postiže kroz neprestani ciklus zaštite, detekcije i odgovora. (Sa malo podataka, ali veoma čitljiva, knjiga u kojoj se ilustruju aspekti ovog ciklusa je [Stoll 1995]).

Nastavljemo ovo poglavlje objašnjenjem (u odeljku 8.7) jednog broja napada koji si; se do sada koristili, kao ; prolivmera koje prema njima mogu da se preduzmu. U odeljku 8.8 zaključićemo ovo poglavlje studijama primera u kojima se koriste tehnike iz odeljaka od 8.2 do 8.5 u aplikacionom sloju, transportnom sloju, mrežnom sloju i sloju veze podataka.

Pošto smo definisali bezbednost mreže, sada ćemo videti kojim informacijama uljez može da ima pristup i koja aktivnosti može da preduzme. Scenario je ilustrovan na slici 8.1. Alisa, pošiljalac, želi da pošalje podatke Bobu, primaocu. Da bi se podaci bezbedno razmenjivali, uz ispunjavanje zahteva tajnosti, autentifikacije i integriteta poruke, Alisa i Bob će razmeniti kontrolne poruke i poruke sa podacima (na isti način kao što TCP pošiljaoci i primaoci razmenjuju kontrolne segmente i segmente sa podacima). Sve, ili neke od ovih poruka će biti šifrovane. Pasivni uljez može da potencija/no izvede

- ♦ *prisluškivanje* - slušanje i registrovanje kontrolne poruke i poruke sa podacima na kanalu;
- ♦ *modifikaciju, umetanje ili brisanje* poruka ili njihovog sadržaja.

Kao što ćemo videti, izuzev ako su preduzete odgovarajuće prottvmere, ove sposobnosti dopuštaju uljezu da izvede širok skup različitih napada na bezbednost u opsegu od prisluškivanja komunikacije (moguća kрадa lozinke i podataka), do predstavljanja drugog entiteta, „kidnapovanja“ sesije u toku, odbijanja usluge legitimnim korisnicima mreže preopterećivanjem sistemskih resursa. Ove i druge napade objasnićemo u odeljku 8.7. Slične bezbednosne pretnje objašnjene su u zbirici eseja [Deruiing 1997] i u veoma Čitljivoj Rubinovoj knjizi [Rubin 2001]. Sumarni pregled napada ažurira se u organizaciji CERT Coordination Center [CERT 2002]. Pročitajte takođe [Cisco Security 1997; Vovdock 1983; Bhimani 1996].



**Slika 8.1** ♦ Pošiljalac, primač i uljez (Alisa, Bob i Trudi)

Pošto smo ustanovili da zaista postoje prave pretnje (poznate i kao „Trudi“) prisutne na Internetu, šta su Internet ekvivalenti Alise i Boba, naših dveju ličnosti koje treba bezbedno da komuniciraju? Svakako, „Bob“ i „Alisa“ **bi** mogli da budu ljudi, korisnici dva krajnja sistema, na primer, stvarna Alisa i stvarni Bob koji zaista žele da razmenjuju bezbednu elektronsku poštu. Oni **bi**, takođe, mogli da budu učesnici u elektronskoj komercijalnoj transakciji. Na primer, stvarna Alisa **hi** mogla željeti da bezbedno prenese broj svoje kreditne kartice veb serveru da bi onlajn kupila neki predmet **ili** uslugu. Slično tome, stvarna Alisa **bi** mogla željeti da ima onlajn interakciju sa svojom bankom. Međutim, kao što piše u [RFC 1636], strane kojima je potrebna bezbedna komunikacija mogle **bi** i same da budu deo mrežne infrastrukture. Selite se da sistem za imenovanje domena (*Domain Name System*, DNS, pročitajte odeljak 2.5) ili programi rutiranja koji menjaju tabele rutiranja (pročitajte odeljak 4.6) zahtevaju bezbednu komunikaciju između dve strane. Isto važi i za aplikacije upravljanja mrežom, stoje teme o kojih govorimo u poglavljju 9. Uljez koji **bi** mogao aktivno da utiče, kontroliše ili oštećuje DNS pretraživanja **i** ažuriranja [RFC 2535, IETF dnsext 2004], proračune rutiranja [Murphy 2003], ili funkcije upravljanja mrežom [RFC 2574], mogao bi da napravi pravu propast na Internetu.

Pošto smo ustanovili okvir, nekoliko najvažnijih definicija i potrebu za mrežnom bezbednošću, hajde sada da zaronimo u kriptografiju. Dok je upotreba kriptografije u obezbeđivanju tajnosti jasna sama po sebi, uskoro ćemo videti da ona takođe ima centralnu ulogu i u obezbeđivanju autentifikacije, integritetu poruke, neosporavanju i kontroli pristupa - stoje čini kamenom temeljcem mrežne bezbednosti.

## 8.2 Principi kriptografije

Iako kriptografija ima dugu istoriju koja datira još od Julija Cezara (uskoro ćemo pogledati takozvanu Cezarovu šifru), savremene kriptografske tehnike, uključujući i mnoge od onih koje se koriste na današnjem Internetu, zasnovane su na dostignućima iz poslednjih 30 godina. Kanova knjiga, *The Codebreakers* [Kahn 1967] i Sin-

gova knjiga, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* [Singh 1999], daju fascinantni pogled na dugu istoriju kriptografije. Detaljna (ali zabavna i Čitljiva) tehnička diskusija o kriptografiji, posebno sa tačke gledišta mreže, nalazi se u Kaufmanovoj knjizi [Kaufman 1995]. Difi u svojoj knjizi [Diffie 1998] ispituje političke i socijalne aspekte (na primer, privatnost) koji su sada nerazmrsivo isprepletani sa kriptografijom. Celovita rasprava o kriptografiji sama po sebi zahteva čitavu knjigu [Kaufmari 1995; Schneier 1995] pa ćemo zasada samo dodirnuti njene suštinske aspekte, posebno kada se primenjuju na današnjem Internetu. Odlična onlajn lokacija je FAQ stranica RSA Labs [RSA FAQ 2002]. Takode napominjemo da dok će se naše izlaganje u ovom odeljku baviti pre svega upotrebom kriptografije radi tajnosti, uskoro ćemo videti da su kriptografske tehnike utkane u autentifikaciju, integritet poruke, neporecivost i još mnogo toga.

Kriptografske tehnike dozvoljavaju pošiljaocu da maskira podatke tako da uljez ne može da dobije nikakvu informaciju iz presremutih podataka. Primalac, naravno, mora da bude u stanju da obnovi originalne podatke iz maskiranih podataka. Na slici 8.2 ilustrovani su neki od značajnih kriptografskih termina.

Pretpostavite sada da Alisa želi da pošalje poruku Bobu. Njena poruka u svom originalnom obliku (na primer, „Bob, I love you . Alice“) je poznata kao **otvoren tekst** (*plaintext*), ili **jasan tekst** (*cleartext*). Alisa šifruje svoju poruku u otvorenom (čitljivom) tekstu koristeći **algoritam šifrovanja** tako da Šifrovana poruka, poznata kao **šifrovan tekst** (*ciphertext*), svakom uljezu izgleda nečitljivo. Zanimljivo je da, u mnogim savremenim kriptografskim sistemima, uključujući one koji se koriste na Internetu, sama tehnika šifrovanja *poznata* - objavljena, standardizovana i svima raspoloživa (na primer, [RFC 1321; RFC 2437; RFC 2420]), pa čak i potencijalnom uljezu! Jasno, ako svako zna metod za šifrovanje podataka, onda mora da postoji neka tajna informacija koja uljeza sprečava da dešifruje podatke koji se prenose. Ovde nastupaju ključevi.

Na slici 8.2, Alisa obezbeđuje **ključ**,  $K_A$ , tj. niz brojeva ili znakova, kao ulaz u algoritam za Šifrovanje. Algoritam za šifrovanje uzima ključ i poruku u otvorenom tekstu,  $m$ , kao ulaz i pravi Šifrovani tekst kao izlaz. Notacija  $KJm$  odnosi se na Šifrovani tekst (šifrovan upotrebom ključa  $K_A$ ) poruke otvorenog teksta,  $m$ . Stvarni algoritam za Šifrovanje koji koristi ključ  $K_A$  shvatite ćete iz konteksta. Slično tome, Bob će obezbediti ključ,  $K_B$ , za algoritam za dešifrovanje, koji uzima Šifrovani tekst i Bobov ključ kao ulaz i pravi originalan otvoren tekst kao izlaz. Odnosno, ako Bob prima Šifrovani poruku  $K_B(K_A(m)) = m$ , on je dešifruje izračunavajući  $K_B(K_A(m)) = m$ . U **simetričnim sistemima ključeva**, Alisin i Bobov ključ su istovetni i tajni. U **javnim sistemima ključeva**, koristi se par ključeva. Jedan od ključeva znaju i Bob i Alisa (u stvari, zna ga ceo svet). Drugi ključ znaju ili Bob ili Alisa (ali ne i oboje). U sledeća dva pododeljka, detaljnije ćemo razmotriti simetrične i javne sisteme ključeva.

### 8.2.1 Kriptografija simetričnog ključa

Svi kriptografski algoritmi obuhvataju zamenu jedne stvari drugom, na primer, uzimanje dela otvorenog teksta a zatim izračunavanje i zamenu odgovarajućeg teksta šifrovanim tekstrom, sve u cilju pravljenja Šifrovane poruke. Pre nego što proučimo

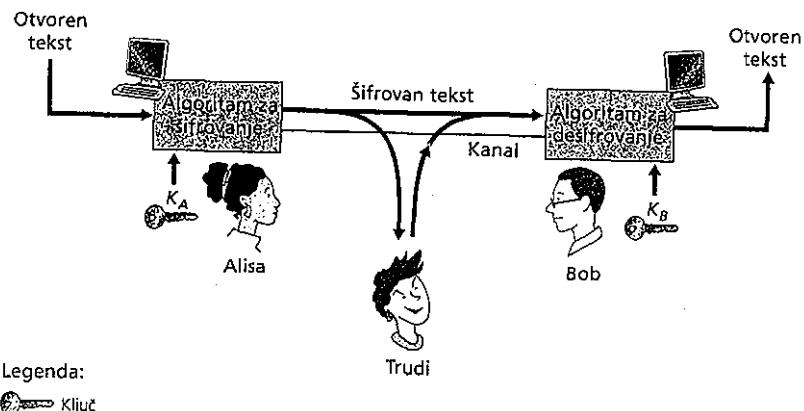
## KRATAK OSVRT

### TAKMIČENJA U RAZBIJANJU KODOVA

Usvojen prvo od strane Vlade SAD 1977. godine, 56-bitni algoritam standarda za šifrovanje podataka (*Data Encryption Standard, DES*) još uvek široko koriste finansijske službe i druge industrije širom sveta do bi zaštitile osetljive informacije. Da bi istakla potrebu za jačim šifrovanjem od postojećeg 56-bitnog standarda, kompanija RSA je bila sponzor niza takmičenja u razbijanju DES-a koji se široko koristi kako u SAD tako i u međunarodnoj trgovini. Svaki izazov sastojao se u tome da se dešifruje poruka šifrovana pomoću 56-bitnog DES-a u okviru odvedenog vremena. Razbijajući koda su prihvatali izazov iscrpljeno pretražujući prostor svih mogućih tajnih ključeva. Kao deo takmičenja, RSA dodeljuje pobednicima nagradu od 10.000 USD.

Kompanija RSA je pokrenula izazov u januaru 1997. godine, sa ciljem da pokaze kako standard Vlade SAD za šifrovanje podataka, sa svojim 56-bitnim ključem, nudi samo ograničenu zaštitu protiv upornog protivnika. Na prvom takmičenju pobedio je tim iz Kolorada koji je otkrio tajni ključ za manje od četiri meseca. Od tada, poboljšana tehnologija je omogućila mnogo bržu iscrpljeno pretraživanja. U februaru 1998. godine, organizacija Distributed.Net pobedila je na takmičenju DES Challenge II-i kompanije RSA u roku od 41 dana, a u julu je firma Electronic Frontier Foundation (EFF) pobedila na takmičenju DES Challenge 11-2 razbijivši DES poruku za 56 sati.

U januaru 1999. godine, Distributed.Net, svelska koalicija računarskih entuzijasta, radila je na „Deep Crack“, posebno konstruisanim superračunaram firme EFF i mrežom širom sveta od skoro 100000 PC-ja na Internetu, da bi pobedila na takmičenju DES Challenge III u rekordnom roku od 22 sata i 15 minuta. EFF-ov „Deep Crack“ i računari Distributed.Net su ispitivali brzinom od 245 milijardi ključeva u sekundi kada je ključ bio pronađeni



**Slika 8.2** ♦ Kriptografske komponente

savremen kriptografski sistem zasnovan na ključu, hajde prvo da proučimo veoma star jednostavan algoritam simetričnog ključa koji se pripisuje Juliju Cezaru, poznat kao *Cezarova Šifra*.

Za engleski tekst, Cezarova Šifra funkcioniše uzimanjem svakog slova u poruci otvorenog teksta i zamenom tog slova onim koje je  $k$  slova dalje u abecedi (dozvoljavajući povratak na početak, to jest da posle slova „z“ sledi slovo „a“). Na primer, ako je  $k = 3$ , onda slovo „a“ u otvorenom tekstu postaje „d“ u Šifrovanom tekstu; „b“ u otvorenom tekstu postaje „e“ u Šifrovanom tekstu itd. Ovde vrednost  $k$  služi kao ključ. Kao primer, poruka otvorenog teksta „bob, i love you . alice“ u Šifrovanom tekstu postaje „ere, 1 oryhbx. dolfh“. Mada Šifrovan tekst zaista izgleda besmisleno, ne bi bilo potrebno mnogo vremena da se razbije kod ako biste znali da je upotrebljena Cezarova šifra, zato što postoji samo 25 mogućih vrednosti ključa.

Poboljšanje Cezarove šifre je jednoazbučna Šifra, gde se takođe jedno slovo abecede zamjenjuje drugim. Međutim, umesto da se zamena vrši u skladu sa nekim pravilnim uzorkom (na primer, zamena sa pomerajem  $k$  za sva slova), svako slovo može biti zamjenjeno bilo kojim drugim slovom, sve dok svako slovo ima jedinstveno slovo koje ga zamjenjuje i obrnuto. Pravilo zamene na slici 8.3 prikazuje jedno od mogućih pravila za Šifrovanje otvorenog teksta.

Poruka otvorenog teksta „bob, i love you. alice“ postaje „nkn, s gktc wky. mgsbc“. Dakle, kao i u slučaju Cezarove šifre, ovo izgleda nerazumljivo. Jednoazbučna Šifra bi izgledala bolje od Cezarove Šifre utoliko što ima 261 (reda veličine  $IO^{26}$ ) a ne samo 25 mogućih uparivanja. Pristup grube sile, tj. pokušaja sa svih  $IO^{26}$  mogućih uparivanja zahtevao bi suviše mnogo rada da **bi** bio izvodljiv način za razbijanje algoritma Šifrovanja i dešifrovanje poruke. Međutim, statistička analiza otvorenog teksta, na primer, znanje da su slova „e“, „i“, „t“ ona koja se najčešće pojavljuju u uobičajenom engleskom tekstu (13 i 9 procenata pojavljivanja), i znanje da su neke pojave po dva ili tri slova zajedno posebno česte (na primer, „in“, „it“, „the“, „ion“, „ing“, itd.) Čine razbijanje ovog koda relativno lakim. Ako uljezima neko saznanje o mogućem sadržaju teksta, onda je još lakše razbiti kod. Na primer, ako je uljez Trudi, Bobova supruga, i ako ona sumnja da je Bob var u Alisu, onda bi ona mogla da posumnja da se imena „bob“ i „alice“ pojavljuju u tekstu. Ako **bi** Trudi znala da se ta imena sigurno pojavljuju u Šifrovanom tekstu i imala kopiju gornjeg primera Šifrovanog teksta, ona **bi** onda odmah mogla da odredi sedam od 26 uparivanja slova, zahtevajući  $IO^9$  manje mogućnosti za proveravanje metodom grube sile. Zaista, ako **bi** Trudi sumnjala da je Bob var, ona bi mogla da očekuje da nade i druge odabrane reči u poruci.

Slovov otvorenog teksta: a b c d e f g h i j k l m n o p q r s t u v w x y z  
Slovov Šifrovanog teksta: m a b v c x z a s d f g h i k l p o i u y t r e w q

**Slika 8.3** ♦ Jednoazbučna Šifra

Prilikom razmatranja koliko bi Trudi bilo lako da razbijje Bobovu i Alisinu šemu Šifrovanja, možemo da razlikujemo tri različita scenarija, zavisno od toga koliko informacija uljez poseduje.

- ♦ *Napad samo na Šifrovan tekst.* U nekim slučajevima, uljez može da ima pristup samo na preseđeni Šifrovan tekst, bez ikakve sigurne informacije o sadržaju poruke otvorenog teksta. Videli smo kako statistička analiza može da pomogne u ovakvom napadu samo na Šifrovan tekst.
- ♦ *Napad poznatim otvorenim tekstom.* Videli smo da ako bi Trudi na neki način sigurno znala da su se „bob“ i „al ice“ pojavili u poruci Šifrovanog teksta, ona bi onda mogla da odredi uparivanja (otvoren tekst, Šifrovan tekst) za slova a, 1, i, c, e, b and o. Trudi bi takođe mogla da ima dovoljno sreće da je zapisala sve prenose Šifrovanog teksta i da je pronašla Bobovu sopstvenu dešifrovanu verziju jednog od prenosa, zapisanu na parčetu kartice. Kada uljez zna neka od uparivanja (otvoren lekst, Šifrovan tekst), mi to zovemo napadom na šemu Šifrovanja poznatim otvorenim tekstom.
- ♦ *Napad izabranim otvorenim tekstom.* U napadu izabranim otvorenim tekstom, uljez može da biro poruku otvorenog teksta i dobije njen odgovarajući oblik u Šifrovanom tekstu. Za jednostavne algoritme Šifrovanja koje smo do sada videli, ako bi Trudi mogla da navede Alisu da pošalje poruku, „The quick brown fox jumps over the- lazy dog“, ona bi mogla u potpunosti da razbijje šemu Šifrovanja. Uskoro ćemo videti da za složenije tehnike Šifrovanja, napad izabranim otvorenim tekstom ne znači obavezno da tehnika Šifrovanja može da se razbije.

Pre pet stotina godina, pronađene su tehnike poboljšavanja monoazbučnog Šifrovanja, poznate kao višeazbučno Šifrovanje. Ideja na kojoj se zasniva višeazbučno Šifrovanje je da se koristi više jednoazbučnih Šifri, sa posebnom jednoazbučnom Šifrom za kodovanje slova na posebnoj poziciji u poruci otvorenog teksta. Dakle, isto slovo, koje se pojavljuje na različitim pozicijama u poruci otvorenog teksta moglo bi da bude različito kodovano. Primer višeazbučne šeme Šifrovanja prikazan je na slici 8.4. Ono ima dve različite Cezarove Šifre (sa  $k = 5$  i  $k = 19$ ), prikazane kao redovi na slici 8.4. Možemo izabrati da koristimo te dve Cezarove Šifre, C<sub>1</sub> i C<sub>2</sub>, po ponavljanjem uzorku C<sub>1</sub>, C<sub>2</sub>, C<sub>1</sub>, C<sub>2</sub>. Odnosno, prvo slovo otvorenog teksta će biti kodovano koristeći C<sub>1</sub>, drugo i treće koristeći C<sub>2</sub>. Četvrto koristeći C<sub>p</sub> i peto koristeći C<sub>r</sub>. Uzorak se onda ponavlja, pa se šesto slovo koduje koristeći C<sub>1</sub>, sedmo pomoću C<sub>2</sub> itd. Poruka otvorenog teksta „bob, i love you.“ je tako Šifrovana u „gnu, n etox dhz“. Zapazile da je prvo „b“ u poruci otvorenog teksta Šifrovan u koristeći C<sub>1</sub>, dok je drugo „b“ Šifrovan u koristeći C<sub>2</sub>. U ovom primeru, „ključ“ ■ za Šifrovanje i dešifrovanje je poznavanje dva Cezarova ključa ( $k = 5$ ,  $k = 19$ ) i uzorka C<sub>1</sub>, C<sub>2</sub>, C<sub>p</sub>, C<sub>r</sub>

Standard za šifrovanje podataka (DES) i napredni standard za šifrovanje (AES)

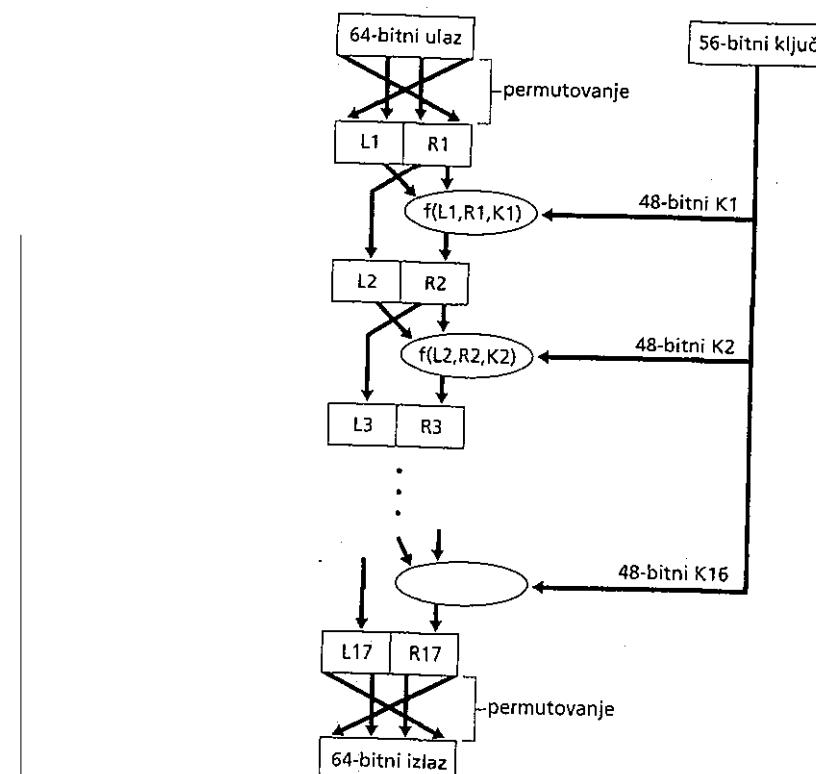
Hajde da se sada brzo prebacimo u moderna vremena i ispitamo Standard za Šifrovanje podataka (DES) [NIST 1993], standard šifrovanja sa simetričnim ključem, objavljen 1977. godine i najskorije ažuriran 1993. godine od strane Nacionalnog biroa SAD za standarde, za komercijalnu i javnu upotrebu Vlade SAD. DES šifruje otvoreni tekst u 64-bitnim „kriškama“ koristeći 64-bitni ključ. U stvari, osam od tih 64 bita ključa su bitovi neparne pamosti (postoji po jedan bit pamosti za svaki od osam bajtova), pa je ključ DES stvarne dužine od 56 bitova. Nacionalni institut za standarde i tehnologiju (naslednik Nacionalnog biroa za standarde) ovako postavlja cilj DES-a: „Cilj je da se potpuno šifruju podaci i ključ, tako da svaki bit šifrovanog teksta zavisi od svakog bita podataka i svakog bita ključa. . . . Sa dobrim algoritmom, ne bi trebalo da bude nikakve korelacije između šifrovanog teksta i bilo originalnih podataka, bilo ključa“. [NIST 1999].

Osnovni rad DES-a ilustrovan je na slici 8.5. U našoj diskusiji, napravićemo pregled rada DES-a, ostavljujući osnovne detalje i detalje na nivou bita (ima ih mnogo) drugim izvorima [Kaufman 1995; Schneier 1995]. Šnajer u svojoj knjizi [Schneier 1995] obuhvata i C implementaciju. DES se sastoji od dva permutaciona koraka (prvi i poslednji korak algoritma), u kojima se svih 64 bita permutoju, sa 16 istovetnih „rundi“ rada između. Rad svake runde je identičan, tako što se uzme izlaz prethodne runde kao ulaz. Za vreme svake runde, 32 bita ulaza krajne desno se pomeraju na 32 bita izlaza levo. Ceo 64-bitni ulaz u -tu rundu i 48-bitni ključ za  $i$ -tu rundu (izveden iz većeg 56-bitnog DES ključa) uzimaju se kao ulaz funkcije koja obuhvata širenje četvorobitnih ulaznih kriški na Sestobitne kriške, izvršavanje operacije isključivo ILI sa prošireniom šestobitnim kriškama 48-bitnog ključa  $K_i$ , operaciju supstitucije i još jednu operaciju isključivo ILI sa 32 bita ulaza krajne levo; više detalja o tome pročitajte u knjizi [Kaufman 1995; Schneier 1995]. Rezultujući 32-bitni izlaz funkcije se onda koristi kao 32 bita krajne desno 64-bitnog izlaza runde, kao što je prikazano na slici 8.5. Dešifrovanje radi obrnutim izvršavanjem operacija algoritma.

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| Slovo otvorenog teksta: | a b c d e f g h i j k l m n o p q r s t u v w x y z<br>f g h i j k l m n o p q r s t u v w x y z a b c d e |
| $C_1(k=5)$ :            | b c d e f g h i j k l m n o p q r s t u v w x y z a                                                        |
| $C_2(k=19)$ :           | c d e f g h i j k l m n o p q r s t u v w x y z a b                                                        |

Slika 8.4 ♦ Višeazbučno šifrovanje korišćenjem dve Cezarove šifre

Koliko dobro radi DES? Koliko je bezbedan? To ne može niko da kaže sa sigurnošću [Kaufman 1995]. U 1997. godini, kompanija za mrežnu bezbednost, RSA Data Security Inc., pokrenula je takmičenje DES Challenge za „razbijanje“ (deko-dovanje) kratke rečenice koju je šifrovala koristeći 56-bitni DES. Nešifrovana rečenica („Strong cryptography makes the world a safer place.“) bila je utvrđena za manje od četiri meseca od strane tima koji je koristio dobrovoljce širom Interneta da sistematski istražuju prostor ključeva. Tim je došao po nagradu od 10000 USD posle pretraživanja samo Četvrtine prostora ključeva - oko 18 kva-driliona ključeva [RSA Challenge 2002]. Najnoviji DES Challenge III u 1999. godini bio je osvojen u rekordnom vremenu od nešto malo više od 22 sata, sa mrežom dobrovoljaca i računaram specijalne namene koji je bio izgrađen za manje od 250000 USD (sa nadimkom „Deep Crack“) i dokumentovan onlajn [EFF 1999].



Slika 8.5 ♦ Osnovni rad DES-a

Trebalo bi takođe da napomenemo da se naš prethodni opis bavio samo šifrova-njem 64-bitnih količina podataka. Kada se šifruju duže poruke, što je obično slučaj, DES se često koristi sa tehnikom koja se zove **ulančavanje šifrovanih blokova**, u kojoj se šifrovana verzija  $-i$  te 64-bitne količine podataka podvrgava operaciji isključivo ILI sa  $(i+1)$ -vom jedinicom podataka pre nego što se šifruje  $(i+1)$ -va jedinica podataka.

Ako se 56-bitni DES smatra suviše nesigurnim, možemo jednostavno da pokrenemo 56-bitni algoritam više puta, uzimajući 64-bitni izlaz iz jedne iteracije DES-a kao ulaz u sledeću iteraciju DES-a, koristeći svaki put različit ključ. Na primer, **tro-struki-DES** (3DES) je standard Vlade SAD [NIST 1999b] koji zamjenjuje upotrebu DES-a, koji iščezava i dozvoljen je samo u naslednjim sistemima. U jednoj konfiguraciji, 3DES prvo pokreće DES algoritam za šifrovanje podataka koristeći prvi 56-bitni ključ, zatim pokreće DES algoritam za dešifrovanje izlaza prve runde šifrovanja koristeći drugi ključ i najzad i konačno pokreće DES algoritam za šifrovanje izlaza druge runde koristeći treći ključ. 3DES je bio predložen kao standard za šifrovanje protokola od tačke do tačke (PPP) [RFC 2420] za sloj veze podataka (pročitajte odeljak 5.8). Detaljna diskusija dužina ključeva i procenjenog vremena i budžeta koji je potreban da bi se razbio DES može da se pronaude u knjizi [Blaže 1996].

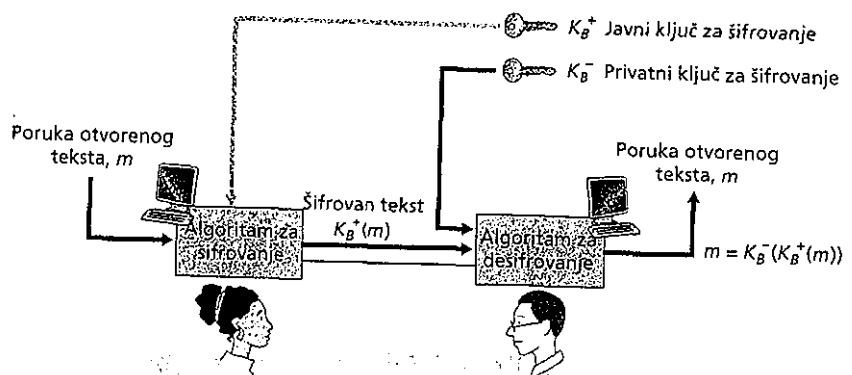
U novembru 2001. godine, NIST ne najavio naslednika DES-a: napredni standard za šifrovanje (*Advanced Encryption Standard*, AES) [NIST 2001, Danielvan 2001], takođe poznatog i kao Rijndelov algoritam [Daemen 2000]. AES je algoritam simetričnog ključa koji obrađuje podatke u 128-bitnim blokovima i može da radi sa ključevima koji su dužine od 128, 192 i 256 bitova. NIST procenjuje da bi mašini koja bi mogla da razbije 56-bitni DES u jednoj sekundi (odnosno; da pokuša brzinu od  $2^{55}$  ključeva u sekundi) trebalo približno 149 triliona godina da razbije 128-bitni AES ključ.

## 8.2.2 Šifrovanje javnim ključem

Za više od 2000 godina (od vremena Cezarove šifre do 1970-ih godina), šifrovana komunikacija zahtevala je da dve strane koje komuniciraju dele zajedničku tajnu - simetrični ključ koji koriste za šifrovanje i dešifrovanje. Jedna od teškoća ovakvog pristupa je da dve strane nekako moraju da se sporazumeju o deljenom ključu; ali da bi se to uradilo, potrebna je (po pretpostavci sigurna) komunikacija! Možda bi strane mogle da se prvo sretnu i lično sporazumeju o ključu (na primer, dva Cezarova centuriona bi mogla da se sretnu u rimskim kupatilima) i posle toga šifrovano komuniciraju. Međutim, u umreženom svetu, može da se desi da strane koje komuniciraju nikada da ne razgovaraju, izuzev preko mreže. Da li je moguće da dve strane šifrovano komuniciraju bez posedovanja deljenog tajnog ključa koji je unapred poznat? U 1976. godini, Difi i Helman [Diffie 1976] su pokazali algoritam (poznat kao Dif-fie-Hellmanova razmena ključa) koji radi baš to - staje radikalno drugačiji i neodoljivo elegantan pristup sigurnoj komunikaciji koji je doveo do razvoja današnjih kriptografskih sistema sa javnim ključevima. Uskoro ćemo videti da kriptografski

sistemi sa javnim ključevima takođe imaju više divnih osobina koje ih čine korisnim ne samo za šifrovanje, nego i za autentifikaciju i digitalne potpise. Zanimljivo je da je nedavno došlo do ideja sličnih onima u knjigama [Diffie 1976] i [RSA 1978] koje su nezavisno razvijene u ranim 1970-im godinama u nizu tajnih izveštaja istraživača u grupi Communications-Electromes Securitv Group u Ujedinjenom kraljevstvu [Ellis 1987]. Kao što je obično slučaj, velike ideje mogu da se nezavisno pojave na više mesta; na svu sreću, napredak javnih ključeva dogodio se ne samo u privatnom, nego i u javnom domenu.

Upotreba kriptografije javnim ključem svodi se na sasvim jednostavnu zamisao. Prepostavite da Alisa želi da komunicira sa Bobom. Kao što je prikazano na slici 8.6, umesto da Bob i Alisa dele jedan tajni ključ (kao u slučaju sistema simetričnih ključeva), Bob (primalac Alisinih poruka) ima dva ključa - **javni ključ** koji je raspoloživ svakome na svetu (uključujući i Trudi, uljeza) i **privatni ključ** koji je poznat samo njemu. Koristićemo notaciju  $K_B^*$  i  $K_B'$  da bismo se pozivali na Bobov javni i privatni ključ, tim redom. Da bi komunicirala sa Bobom, Alisa prvo nabavlja njegov javni ključ. Alisa zatim šifruje svoju poniku za Boba,  $m$ , koristeći Bobov javni ključ i poznat (na primer, standardizovani) algoritam za šifrovanje; odnosno, Alisa izračunava  $K_B(m)$ . Bob prima Alisinu šifrovaniu poruku i koristi svoj privatni ključ i poznat (na primer, standardizovani) algoritam za dešifrovanje da bi dešifrovaо Alisinu šifrovaniu poruku. Odnosno, Bob izračunava  $K_B(K_B^*(m))$ . Videćemo dalje u tekstu da postoje algoritmi za šifrovanje/dešifrovanje i tehnike za biranje javnih i privatnih ključeva takvi da je  $K_B(K_B'(m)) = m$ ; odnosno, primenjujući Bobov javni ključ,  $K_B$ , na poruku,  $m$  (da bi se dobilo  $K_B^*(m)$ ), a zatim primenjujući Bobov privatni ključ,  $K_B'$ , na šifrovaniu verziju  $m$  (odnosno, izračunavanjem  $K_B(K_B^*(m))$ ) dobijamo opet  $m$ . To je izvanredan rezultat! Na taj način, Alisa može da koristi Bobov javno raspoloživ ključ da bi poslala tajnu poruku Bobu, a da nijedna od njih ne mora



Slika 8.6 ♦ Kriptografija javnim ključem

da distribuira bilo kakve tajne ključeve! Uskoro ćemo videti da možemo da zame-nimo Šifrovanje javnim i privatnim ključem i da dobijemo isti izvanredan rezultat, odnosno,  $K_g(K_g^*(m)) = K * (K_B'(m)) = m$ .

Dakle, upotreba kriptografije javnim ključem je jednostavna zamisao. Međutim, tu se mogu javiti dva problema. Prvi problem je, da mada uljez koji presreće Alisinu šifrovanu poruku vidi samo nešto nerazumljivo, on zna i ključ (Bobov javni ključ, koji može da vidi ceo svet) i algoritam koji je Alisa upotrebila za šifrovanje. Trudi na taj način može da preduzme napad izabranim otvorenim tekstom, koristeći poznat standardizovan algoritam za šifrovanje i Bobov javno raspoloživ ključ za šifrovanje, da bi dešifrovala svaku poruku koju izabere! Trudi bi sasvim lepo mogla da pokuša da, na primer, šifruje poruke ili delove poruka za koje sumnja da bi Alisa mogla da ih šalje. Jasno, ako se radi kriptografija javnim ključem, izbor ključa i Šifrovanje/dešifrovanje mora da se uradi na takav način daje uljezu nemoguće (ili, u najmanju ruku, toliko teško daje skoro nemoguće) da ili odredi Bobov privatni ključ ili da nekako drugačije dešifruje ili pogodi Alisinu poruku Bobu. Drugi potencijalni problem je to što, s obzirom na to da je Bobov ključ za Šifrovanje javan, svako može da pošalje šifrovanu poruku Bobu, uključujući Alisu ili nekog ko tvrdi da je Alisa. U slučaju jednog deljenog tajnog ključa, činjenica da pošiljalac zna tajni ključ implicitno identificuje pošiljaoca primaocu. Međutim, u slučaju kriptografije javnim ključem, to više nije tačno, zato što svako može da pošalje, šifrovanu poruku Bobu koristeći Bobov javno raspoloživ ključ. Da bi se pošiljalac povezao jedinstveno sa porukom koju šalje potreban je digitalni potpis, što je tema koju ćemo proučiti u odeljku 8.4.

Iako ima mnogo algoritama i ključeva koji rešavaju ova pitanja, algoritam RSA (nazvan po svojim pronalazačima, Ronu Rivestu, Adiju Shamiru i Leonardu Adlemanu) postao je skoro sinonim za kriptografiju javnim ključem. Hajde najpre da vidimo kako RSA radi, a zatim ispitajmo zašto to radi. Prepostavite da Bob želi da prima šifrovane poruke, kao što je prikazano na slici 8.6. Postoje dve međusobno povezane komponente algoritma RSA;

- ◆ Izbor javnog i privatnog ključa
- ◆ Algoritam za šifrovanje i dešifrovanje

Da bi odabrao javni i privatni ključ, Bob mora da izvede sledeće postupke:

1. Odabrat dva velika prosta broja,  $p$  i  $q$ . Koliko bi  $p$  i  $q$  trebalo da budu veliki? Što su vrednosti veće, to je teže razbiti RSA, ali je potrebno više vremena da bi se izvršilo šifrovanje i dešifrovanje. RSA Laboratorije preporučuju da proizvod  $p \cdot q$  bude reda 1024 bita za profesionalnu upotrebu (npr. u kompanijama) i 768 bitova za rad sa „manje vrednim informacijama“ [RSA Key 2004] (što dovodi do pitanja zastoje kompanijska upotreba toliko značajnija od ostale upotrebe!). O tome kako se pronađaju veliki prosti brojevi, pročitajte knjigu [Caldwell 2004].

2. Izračunati  $n = pq \mid z = \{p \sim I\}(g - 1)$ .
3. Izabrat broj,  $e$ , manji od  $n$ , koji nema zajedničkih činilaca (izuzev 1) sa brojem  $z$ . (U ovom slučaju, za  $e$  i  $z$  se kaže da su relativno prosti.) Slovo  $e$  se koristi zato što će ta vrednost biti korišćena u šifrovanju.
4. Pronaći broj,  $d$ , takav daje  $ed - 1$  tačno deljivo (odnosno, bez ostatka) sa  $z$ . Slovo  $f$  je upotrebljeno zato što će se ta vrednost koristiti u dešifrovanju. Drugačije rečeno, za dato  $e$ , biramo  $d$  tako da celobrojni ostatak kada se  $ed$  podeli sa  $z$  iznosi 1. (Celobrojni ostatak kada se ceo broj  $x$  podeli celim brojem  $n$ , označava se sa  $x \bmod n$ ).
5. Javni ključ koji Bob stavlja na raspolaganje celom svetu,  $K_s^+$ , je par brojeva  $(n, e)$  njegov privatni ključ,  $K_B^-$ , je par brojeva  $(n, d)$ .

Alisino šifrovanje i Bobovo dešifrovanje se obavljaju na sledeći način:

- ♦ Prepostavite da Alisa želi da pošalje Bobu uzorak bitova ili broj,  $m$ , takav daje  $m < n$ . Da bi šifrovala, Alisa obavlja operaciju eksponenciranja,  $m^e$ , a zatim izračunava celobrojni ostatak kada se  $m^e$  podeli sa  $n$ . Dakle, šifrovana vrednost,  $c$ , poruke otvorenog teksta,  $m$ , koji Alisa šalje je

$$c = nf \bmod n$$

- ♦ Da bi dešifrovala poruku šifrovanog teksta,  $c$ , Bob izračunava

$$m = c^d \bmod n$$

Što zahteva upotrebu njegovog privatnog ključa  $(n, d)$ .

Kao jednostavan primer algoritma RSA, prepostavite da Bob bira  $p = 5$  i  $q = 7$ . (Iskreno govoreći, te vrednosti su suviše male da bi bile sigurne.) Onda je  $n = 35$  i  $z = 24$ . Bob bira  $e = 5$ , zato što 5 i 24 nemaju zajedničkih činilaca. Najzad, Bob bira  $d = 29$ , zato stoje  $5 \cdot 29 - 1$  (odnosno,  $ed - 1$ ) tačno deljivo sa 24. Bob čini dve vrednosti javnim,  $n = 35$  i  $e = 5$ , a zadržava vrednost  $d = 29$  kao tajnu. Posmatrajući ove dve javne vrednosti, prepostavite da Alisa sada želi da pošalje Bobu slova „I“, „o“, „v“ i „e“. Interpretirajući svako slovo kao broj između 1 i 26 (gde je „a“ 1, a „z“ 26), Alisa i Bob izvršavaju šifrovanje i dešifrovanje prikazano u tabelama 8.1 and 8.2, tim redom.

Imajući u vidu daje pojednostavljeni primeru tabelama 8.1 i 8.2 već proizveo neke izuzetno velike brojeve, i s obzirom na to da znamo da smo ranije videli kako bi  $p$  i  $q$  trebalo da budu dužine više stotina bitova, dolazimo do nekih praktičnih pitanja u pogledu algoritma RSA. Kako se biraju veliki prosti brojevi? Kako se biraju  $e$  i  $d$ ? Kako se vrši eksponenciranje sa velikim brojevima? Diskusija o tim značajnim pitanjima je izvan domena ove knjige; pročitajte knjigu [Kaufman 1995] i reference koje se tamo nalaze da biste saznali više pojedinosti o tome.

Ovde samo napominjemo da je RSA eksponenciranje proces koji troši dosta vremena. S druge strane, DES je najmanje 100 puta brži u softveru i između 1000

| Slovo otvorenog teksta | $m$ : numerička predstava | $m^e$   | šifrovan tekot $c = m^e \text{ mod } n$ |
|------------------------|---------------------------|---------|-----------------------------------------|
| I                      | 12                        | 248832  | 17                                      |
| O                      | 15                        | 759375  | 15                                      |
| V                      | 22                        | 5153632 | 22                                      |
| E                      | 5                         | 3125    | 10                                      |

**Tabela 8.1** ♦ Alisino RSA šifrovanje,  $e = 5$ ,  $n = 35$

| šifrovan tekst c | $c^d$                                      | $m = c^d \bmod n$ | Slovo otvorenog teksta |
|------------------|--------------------------------------------|-------------------|------------------------|
| 17               | 481968572106750915091411825223071697       | 12                | i                      |
| 15               | 12783403948858939111232757568359375        | 15                | o                      |
| 22               | 851643319086537701956194499721106030592    | 22                | v                      |
| 10               | 100000000000000000000000000000000000000000 | 5                 | e                      |

**Tabela 8.2** ♦ Bobovo RSA dešifrovanje,  $d = 29$ ,  $n = 35$

i 10000 puta brži u hardveru [RSA Fast 200.4]. Kao rezultat, RSA se u praksi često koristi u kombinaciji sa DES-om ili AES-om. Na primer, ako Alisa želi da pošalje Bobu veliku količinu šifrovanih podataka velikom brzinom, ona bi trebalo da uradi sledeće. Alisa prvo bira DES ključ koji će se upotrebiti za šifrovanje samih podataka; taj ključ se ponekad zove ključ sesije,  $K_s$ . Alisa mora da informiše Boba o ključu sesije, zato Stoe to deljeni simetrični ključ koji će oni koristiti za DES. Alisa dakle šifruje vrednost ključa sesije koristeći Bobov javni RSA ključ, odnosno, izračunava  $c = (K^e \text{ mod } n)$ . Bob prima RSA-šifrovan ključ sesije,  $c$ , i dešifruje ga da bi dobio ključ sesije,  $K_s$ . Bob sada zna ključ sesije koji će Alisa koristiti za prenos DES-šifrovanih podataka.

Kako RSA radi?

Gore opisano RSA šifrovanje/dešifrovanje izgleda kao prava magija. Zašto bismo, kada primenimo najpre algoritam za Šifrovanje a zatim algoritam za dešifrovanje, rekonstruisali originalnu poruku? Da bismo razumeli kako RSA radi, biće potrebno da izvedemo aritmetičke operacije koristeći aritmetiku modulo-«. U modularnoj

aritmetici, izvršavamo uobičajene operacije sabiranja, množenja i eksponenciranja. Međutim, rezultat svake operacije se zamenjuje celobrojnim ostatkom koji preostaje kada se rezultat podeli sa  $n$ . Uzećemo da je  $n = pg$ , gde su  $p$  i  $q$  veliki prosti brojevi koji se koriste u RSA algoritmu.

Setite se da kod RSA šifrovanja, poruka (predstavljena celim brojem),  $m$ , prvo eksponencira na stepen  $e$  koristeći aritmetiku modulo- $\varphi$ . Dešifrovanje se vrši podizanjem točne vrednosti na stepen  $d$ , koristeći opet aritmetiku modulo- $\varphi$ . Rezultat postupka šifrovanja praćenog postupkom dešifrovanja je zato  $(m^e)^d$ . Hajde da sada vidimo šta može da se kaže o ovoj veličini.

Imamo:

(V/mod  $\ll m^{\text{mod}}$

Iako pokušavamo da uklonimo nešto od „magije“ u radu algoritma RSA, ovde će biti potrebno da iskoristimo jedan prilično magičan rezultat iz teorije brojeva. Naročito će nam trebati rezultat koji kaže da ako  $\supin$  prosti brojevi, i ako je  $\ll = pq$ , onda je  $\# > \text{mod}$   $\ll$  isto što je  $\chi^{<\text{ymod}}(\rho - V)$   $i$ -tih W mod n [Kaufman 1995]. Primenjujući ovaj rezultat, imamo

$$(m^c)^d \bmod n = m^{<^{ud} " >_{OD}(P" M^{(X)})} \bmod n$$

Ali, setite se da biramo  $e$  i  $d$  tako da je  $ed - 1$  tačno deljivo (bez ostatka) sa  $(p - 1)(q - 1)$ , ili ekvivalentno tome, daje  $ed$  deljivo sa  $(p - 1)(q - 1)$  sa ostatkom 1, pa je na taj način  $ed$  mod  $(p - 1)(q - 1) = 1$ . To nam daje

$$(m^c)^d \bmod n = m^l \bmod n$$

odnosno,

$$(m^e)^d \bmod n = r$$

To je rezultat kome smo se nadali! Prvo eksponenciranjem na stepen od  $e$  (odnosno šifrovanjem) a zatim eksponenciranjem na stepen od  $d$  (odnosno, dešifrova-njem), dobijamo originalnu vrednost,  $m$ . Još bolja od toga jeste činjenica da ako prvo eksponenciramo na stepen od  $d$ , a zatim eksponenciramo na vrednost od  $e$  - odnosno, obmemu redosled šifrovanja i dešifrovanja, izvodeći prvo operaciju dešifrovanja, a zatim primenjujući operaciju šifrovanja takođe dobijamo originalnu vrednost,  $m$ . (Dokaz ovog rezultata sledi iz potpuno istog rezonovanja kao u prethodnom delu teksta.) Uskoro ćemo videti da će nam ova divna osobina algoritma RSA,

$$(m^e)^d \bmod n = m \equiv (m^d)^e \bmod n$$

biti veoma korisna.

Bezbednost RSA oslanja se na činjenicu da ne postoje poznati algoritmi za brzo faktorisanje broja, u ovom slučaju javne vrednosti  $n$ , na proste brojeve  $p$  i  $q$ . Ako bi neko znao  $p$  i  $g$ , onda bi uz datu javnu vrednost  $e$ , mogao lako da izračuna tajni ključ,  $d$ . S druge strane, ne zna se da li postoje ili ne postoje brzi algoritmi za faktorisanje broja, i u tom smislu bezbednost RSA nije „garantovana“.

## 8.3 Autentifikacija

**Autentifikacija** je proces dokazivanja identiteta jedne osobe drugoj osobi. Kao ljudi, autentifikujemo jedan drugoga na mnogo načina: prepoznajemo jedni drugima lica kada se sretnemo, prepoznajemo jedni drugima glasove preko telefona, autenti-fikuje nas carinik koji nas proverava u odnosu na sliku u pasošu.

U ovom odeljku razmatramo kako jedna sirana može da autentificuje drugu, kada one komuniciraju preko mreže. Usredsredićemo se na autentifikovanje „žive“ strane, u trenutku kada se komunikacija stvarno događa. Videćemo daje to malo drugačiji problem od dokazivanja da je poruka, primljena u nekom trenutku u prošlosti (na primer, daje mogla da se arhivira), zaista došla od onoga koje tvrdio daje pošiljalac. Ovaj poslednji problem zove se problem **digitalnog potpisa**, kojeg ćemo objasniti u odeljku 8.4.

Kada izvršavaju autentifikaciju preko mreže, strane u komunikaciji ne mogu da se oslone na biometrijske informacije, kao što je vizuelna pojava ili uzorak glasa. Zaista, videćemo u kasnijim studijama primera da mrežni elementi, kao što su ruteri i procesi klijent/server, Često moraju da autentifikuju jedan drugoga. Ovde autentifikacija mora da se uradi samo na osnovu poruka i podataka razmenjenih u okviru **protokola autentifikacije**. Uobičajeno je da se, protokol autentifikacije izvrši pre nego što dve strane koje komuniciraju pokrenu neki drugi protokol (na primer, protokol pouzdanog prenosa, protokol za razmenu tabela rutiranja ili protokol za elektronsku poštu). Protokol autentifikacije prvo ustanovljava identitet strana na obostrano zadovoljstvo; samo posle autentifikacije strane mogu da predu na posao koji ih očekuje.

Kao u slučaju razvoja našeg protokola za pouzdan prenos podataka (*reliable data transfer*, rdt) u poglavlju 3, otkrićemo daje ovde poučno razviti različite verzije protokola za autentifikaciju, koji ćemo zvati **ap** („autentifikacioni protokol“) i razmatrati svaku verziju kako napredujemo dalje u izlaganju. (Ako volite ovakav postepen razvoj dizajna, mogli biste takođe pročitati knjigu [Bryant 1988], u kojoj se prepričava izmišljen razgovor između dizajnera i sistema za autentifikaciju otvorene mreže, i njihovo otkriće mnogih pitanja u vezi sa tim.)

Prepostavimo da Alisa treba da se autentificuje Bobu.

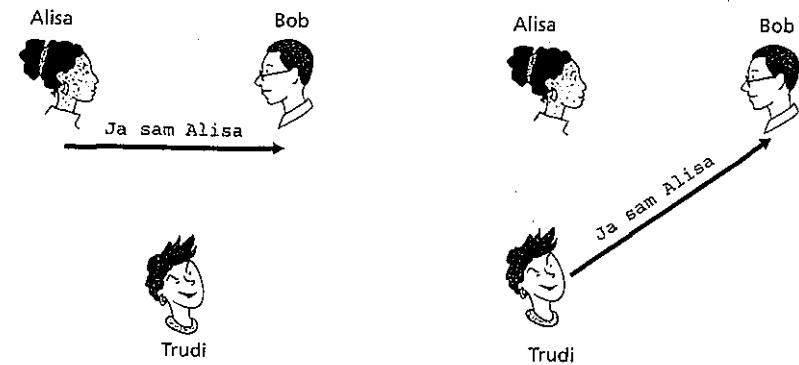
### 8.3.1 Autentifikacioni protokol apl.O

Možda je najjednostavniji autentifikacioni protokol koji možemo da zamislimo onaj u kom Alisa prosto šalje poruku Bobu u kojoj kaže daje ona Alisa. Taj protokol je prikazan na slici 8.7. Ovde je greška očigledna - nema načina da Bob stvarno zna da je osoba koja je poslala poruku „Ja sam Alisa“ zaista Alisa. Na primer, Trudi (uljez) bi mogla takođe da pošalje takvu poruku.

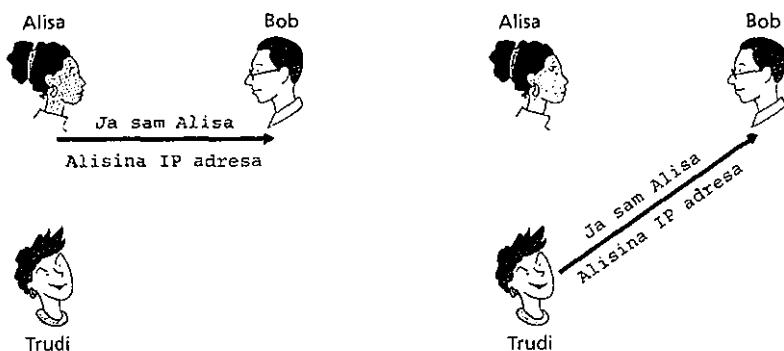
### 8.3.2 Autentifikacioni protokol ap2.0

U slučaju da Alisa ima dobro poznatu mrežnu adresu (na primer, IP adresu) sa koje ona uvek komunicira, Bob bi mogao da pokuša da autentificuje Alisu proveravajući da izvorna adresa na IP datagramu koji nosi poruku o autentifikaciji odgovara Ali-sinoj dobro poznatoj adresi. Ako je tako, onda bi Alisa bila autentifikovana. To bi moglo da zaustavi veoma naivnog uljeza da se predstavlja kao Alisa. Ali, to ne bi zaustavilo odlučnog studenta koji čita ovu knjigu ili još mnogo drugih ljudi!

S obzirom na to da smo do sada proučili i mrežni sloj i sloj linka podataka, znamo da nije tako teško napraviti IP datagram (na primer, ako smo imali pristup kodu operativnog sistema i mogli da izgradimo sopstveno jezgro operativnog sistema, kao u slučaju Linuxa i nekoliko drugih besplatno raspoloživih operativnih sistema), staviti u IP datagram bilo kakvu IP adresu izvora (na primer, Alisinu dobro poznatu IP adresu) i poslati datagram preko protokola sloja linka na ruter prvog skoka. Odатле pa nadalje, netačno izvorno adresiran datagram bi revnosno bio pro-sledjen Bobu. Ovaj pristup, prikazan na slici 8.8, predstavlja oblik IP obmane, veoma dobro poznat bezbednosni napad koji ćemo detaljnije razmatrati u odeljku 8.7. IP obmana može da se izbegne ako je Trudin ruter prvog skoka konfigurisan da prosleđuje samo datagrame koji sadrže Trudinu IP izvornu adresu [RFC 2827]. Međutim, ova sposobnost nije uvek prisutna niti se obavezno primenjuje. Bob bi zato bio glup ako bi prepostavio daje Trudin upravnik mreže (koji bi mogla da bude i sama Trudi) konfigurisao njen ruter prvog skoka da prosleđuje samo one datagrame koji su adresirani na odgovarajući način.



**Slika 8.7** ♦ Protokol *ap1.0* i scenario neuspeha



**Slika 8.8** ♦ Protokol *ap2.0* i scenario neuspeha

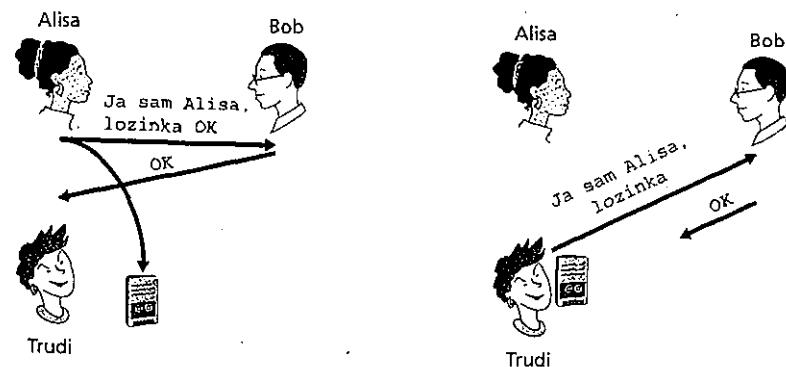
### 8.3.3 Autentifikacioni protokol *ap3.0*

Jedan od klasičnih pristupa autentifikaciji je da se koristi tajna lozinka. Svi koristimo PTN-ove da bismo se identifikovali bankomatima i lozinke prijavljivanja za operativne sisteme. Lozinka je deljena **rajna** između onog ko autentificuje i osobe koja se autentificuje. U odeljku 2.2. smo videli da HTTP koristi autentifikacionu šemu zasnovanu na lozincima. Telnet i FTP takođe koriste autentifikaciju pomoću lozinke. U protokolu *ap3.0*, Alisa zato šalje svoju tajnu lozinku Bobu, kao Stope prikazano na slici 8.9.

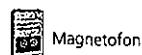
S obzirom na to da se lozinke tako široko koriste, mogli bismo da pomislimo kako je protokol *ap3.0* prilično bezbedan. Međutim, ne bismo bili u pravu! Ovde je jasna bezbednosna greška. Ako Trudi prisluškuje Alisinu komunikaciju, ona bi onda mogla da sazna njenu lozinku. Da ne biste pomisili kako je to neverovatno, razmotrite činjenicu da kada se neko obraća preko Telneta drugoj mašini i prijavljuje se, lozinka za prijavljivanje se nešifrovano šalje Telnet serveru. Neko povezan sa Telnet klijentom ili na serverovom LAN-u mogao bi da „onjuši“ (procita i zapise) sve pakete koji se prenose preko LAN-a i tako ukrade lozinku za prijavljivanje. U stvari, to je dobro poznat pristup za kradu lozinke (procitajte, na primer, knjigu [Jimenez 1997]). Takva pretnja je očigledno veoma realna, pa je jasno da *ap3.0* ne bi obavio posao.

### 8.3.4 Autentifikacioni protokol *ap3.1*

Kada smo proučili prethodni odeljak o kriptografiji, naša sledeća ideja za popravljanje protokola *ap3.0* je prirodno da upotrebimo Šifrovanje kako bismo šifrovali lozinku. Šifrovanjem lozinke, možemo da sprečimo Trudi da sazna Alisinu lozinku. Ako prepostavimo da Alisa i Bob dele simetrični tajni ključ,  $K_{A2g}$  onda Alisa može da šifruje lozinku i pošalje Bobu svoju identifikacionu poruku, „I am Alice“ i svoju šifrovani lozinku. Bob onda dešifruje lozinku i, pod pretpostavkom da



Legenda:



**Slika 8.9** ♦ Protokol *ap3.0* i scenario neuspeha

lozinka tačna, autentificuje Alisu. Bob se prilikom autentifikacije Alise oseća komotno, zato što Alisa ne samo da zna lozinku, nego poznaje i deljeni tajni ključ koji je potreban da bi se šifrovala lozinka. Nazovimo ovaj protokol *ap3.1*. Dok je tačno da *ap3.1* sprečava Trudi da sazna Alisinu lozinku, upotreba kriptografije ovde ne rešava problem autentifikacije. Na Boba je izvršen takozvani **napad reprodukcije**: Trudi treba samo da prisluškuje Alisinu komunikaciju, zapiše Šifrovani verziju lozinke i zatim kasnije reprodukuje Šifrovani verziju lozinke Bobu, kako bi se pretvarala daje ona Alisa. Upotreba Šifrovane lozinke u protokolu *ap3.1* situaciju ne Čini bitno drugačijom od one za protokol *ap3.0* na slici 8.9.

### 8.3.5 Autentifikacioni protokol *ap4.0*

Problem sa protokolom *ap3.1* je u tome što se stalno koristi ista lozinka. Jedan način da se taj problem reši bio bi da se svaki put koristi drukčija lozinka. Alisa i Bob bi mogli **da** se dogovore o sekvenci lozinke (ili o algoritmu za generisanje lozinke) i da koriste svaku lozinku samo jednom, u sekvenci. Ta ideja se koristi u sistemu S/K£Y [RFC 1760], koji usvaja Lamportov pristup za generisanje sekvence lozinki (Lamport 1983).

Ali, umesto da se ovde zadovoljimo tim rešenjem, hajde da razmotrimo opštiji pristup za borbu protiv napada reprodukcije. Scenario neuspeha sa slike 8.9 posle-

dica je činjenice da Bob ne bi mogao da razlikuje originalnu Alisino autentifikaciju od kasnije reprodukcije Alisine originalne autentifikacije. Odnosno, Bob ne bi mogao da kaže da li je Alisa bila „uživo“ (to jest, da li je stvarno bila na drugom kraju konekcije) ili da li su poruke koje je on primao bile reprodukcija snimljene prethodne Alisine autentifikacije. Čitalac sa velikom (veoma velikom) sposobnošću opažanja setiće se da je TCP protokol sa trostrukom sinhronizacijom trebalo da rešava isti problem - serverska strana TCP konekcije nije htela da prihvati konekciju ako je primljeni SYN segment bio stara kopija (retransmisija) SYN segmenta iz neke ranije konekcije. Kako je serverska strana resila problem određivanja da li je klijent stvarno išao „uživo“? Ona je izabrala početni broj sekvence koji nije bio korišćen dugo vremena, poslala taj broj klijentu i zatim čekala da klijent odgovori ACK segmentom koji sadrži taj broj. Možemo ovde da usvojimo istu ideju u svrhu autentifikacije.

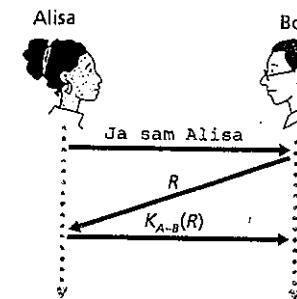
Jednokratni broj (*nonce*) je broj koji će protokol upotrebiti samo jednom zauvek. Odnosno, jednom kada protokol upotrebi jednokratni broj, on taj broj više nikada neće koristiti. Naš protokol *ap4.0* koristi jednokratni broj na sledeći način:

1. Alisa šalje Bobu poruku „I am Alice“.
2. Bob bira jednokratni broj,  $R$ , i šalje ga Alisi.
3. Alisa šifruje jednokratni broj koristeći svoj i Bobov simetrični tajni ključ,  $K_{A-B}$ , i šalje natrag Bobu šifrovani jednokratni broj,  $K_{A-B}(R)$ . Kao u protokolu *apS.i*, činjenica je da Alisa zna  $K_{A-B}$  i koristi ga za šifrovanje vrednosti koja dopušta Bobu da zna kako je poruku koju on prima napravila Alisa. Jednokratni broj se koristi da bi obezbedio da Alisa ide „uživo“.
4. Bob dešifruje primljenu poruku. Ako je dešifrovan jednokratni broj jednak jednokratnom broju koji je on poslao Alisi, onda je Alisa autentifikovana.

Protokol *ap4.0* ilustrovan je na slici 8.10. Koristeći vrednost za  $R$  samo jednom zauvek i zatim proveravajući vraćenu vrednost,  $K_{A-B}^B(R)$ , Bob može da bude siguran daje Alisa ona za koju se izdaje (zato što poznaje vrednost tajnog ključa potrebnu za šifrovanje  $R$ ) i da ide „uživo“ (zato stoje šifrovala jednokratni broj,  $R$ , koji je Bob upravo napravio).

### 8.3.6 Autentifikacioni protokol *ap5.0*

Upotreba jednokratnog broja i kriptografije simetričnog ključa stvorila je osnovu za naš uspešni protokol autentifikacije, *ap4.0*. Prirodno pitanje je da li možemo da koristimo jednokratan broj i kriptografiju javnog ključa (a ne kriptografiju simetričnog ključa) da bismo resili problem autentifikacije. Upotreba javnog ključa izbegla bi teškoću koja se javlja u svakom sistemu sa deljenim ključem - brigu o tome kako bi obe strane saznale vrednost tajnog deljenog ključa. Protokol koji koristi kriptografiju javnog ključa na sličan način kao kriptografiju deljenog ključa u protokolu *ap4.0* je protokol pod imenom *ap5.0*.



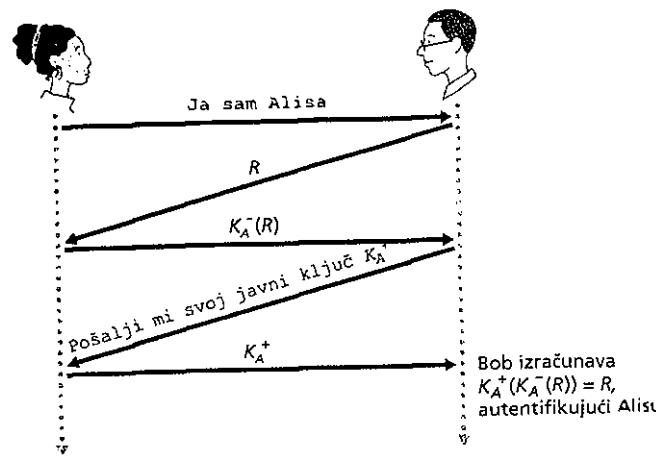
**Slika 8.10** ♦ Protokol *ap4.0*: nema scenarija neuspeha.

1. Alisa šalje Bobu poruku „I am Alice“.
2. Bob bira jednokratni broj,  $R$ , i šalje ga Alisi. Još jednom, jednokratni broj će biti upotrebljen da bi se osiguralo da Alisa ide „uživo“.
3. Alisa koristi svoj privatni ključ,  $K_A$ , da bi šifrovala jednokratni broj i šalje Bobu rezultujuću vrednost  $K_A(R)$ . Kako samo Alisa zna svoj privatni ključ, niko sem nije ne može da generiše  $K_A(R)$ .
4. Bob primenjuje Alisin javni ključ,  $K$ , na primljenu poruku; odnosno, Bob izračunava  $K_A^B(K(R))$ . Setite se iz naše diskusije o RSA kriptografiji javnim ključem u odeljku 8.2 da je  $K^B(K(R)) = R$ . Dakle, Bob izračunava  $R$  i autentifikuje Alisu.

Rad protokola *ap5.0* prikazan je na slici 8.11. Da lije protokol *ap5.0* tako bezbedan kao protokol *ap4.0*? Oba protokola koriste jednokratne brojeve. Kako *ap5.0* koristi tehnike javnih ključeva, on zahteva da Bob dobije Alisin javni ključ. To dovodi do zanimljivog scenarija, prikazanog na slici 8.12, u kome Trudi može da Bobu glumi Alisu.

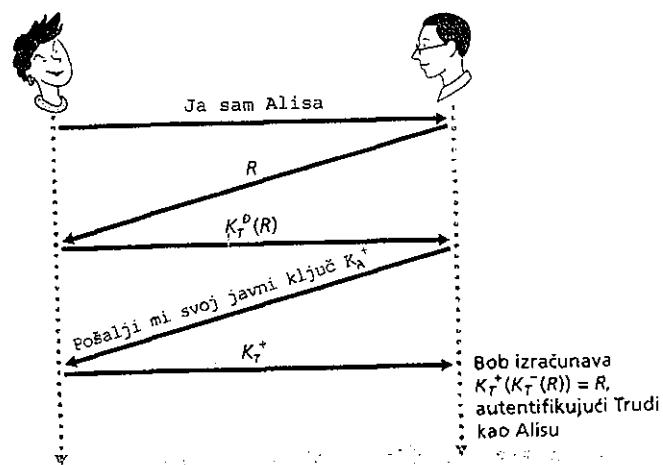
1. Trudi šalje Bobu poruku „I am Alice“.
2. Bob bira jednokratni broj,  $R$ , i šalje ga Alisi, ali Trudi presreće tu poruku.
3. Trudi koristi svoj privatni ključ,  $K_T$ , da bi šifrovala jednokratni broj i šalje Bobu rezultujuću vrednost,  $K_T(R)$ . Za Boba,  $K_T(R)^p$  samo hrpa bitova i on ne zna da li bitovi predstavljaju  $K_T(R)$  ili  $K(R)$ .
4. Bob sada mora da dobije Alisin javni ključ da bi primenio  $K_A^B$  na vrednost koju je baš primio. On šalje Alisi poruku u kojoj od nje traži  $K$  (Bob bi takođe mogao da dobije Alisin javni ključ sa njene veb lokacije). Trudi presreće i tu poruku i odgovara Bobu sa  $K_T^B(K(R))$ , odnosno, svojim javnim ključem. Bob proračunava  $K_T^B(K(R)) = R$  i tako autentifikuje Trudi kao Alisu!

Iz gornjeg scenarija, jasno je daje protokol *ap5.0* „siguran“ samo koliko je i distribucija javnih ključeva sigurna. Na svu sreću, postoje bezbedni načini distribuiranja javnih ključeva, kao što ćemo uskoro videti u odeljku 8.5.



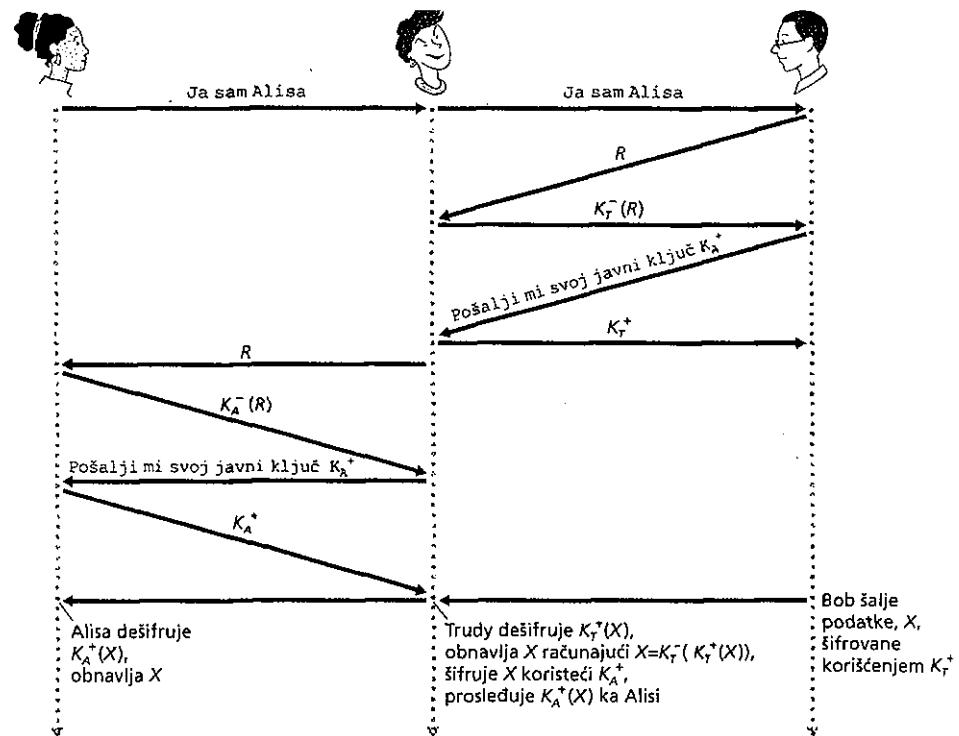
Slika 8.11 ♦ Protokol ap5.0 radi ispravno.

U scenariju na slici 8.12, Bob i Alisa bi na kraju mogli zajedno da otkriju kako nešto nije u redu, zato što će Bob tvrditi daje imao interakciju sa Ahsom, ah Alisa zna da nikada nije imala interakciju sa Bobom. Postoji čak još lukaviji napad koji bi



Slika 8.12 ♦ Bezbednosna pukotina u protokolu ap5.0

izbegao to otkrivanje. U scenariju na slici 8.13, i Alisa i Bob pričaju jedno sa drugim ali, koristeći istu pukotinu u protokolu autentifikacije, Trudi može da se *transpa-rentno* ubaci između Alise i Boba. Posebno, ako Bob počne da šalje Alisi šifrovane podatke koristeći ključ za šifrovanje koji prima od Trudi, Trudi može da rekonstru-še otvoreni tekst u komunikaciji od Boba do Alise. U isto vreme, Trudi može da prosledi Bobove podatke ka Alisi (posle ponovnog Šifrovanja podataka korišćenjem Alisinog stvarnog javnog ključa).



Slika 8.13 ♦ Napad „čoveka u sredini“

Bob je zadovoljan što šalje šifrovane podatke, a Alisa je zadovoljna što prima šifro-vane podatke koristeći sopstveni javni ključ; oboje su nesvesni Trudinog prisustva. Ako bi se Bob i Alisa kasnije sreli i porazgovarali o svojoj interakciji, Alisa bi rekla daje primila tačno ono stoje Bob posalo, pa se ne bi otkrilo ništa što nije u redu. Ovo je jedan od primera takozvanog napada Čoveka u sredini (ovde bi više odgovaralo, napada „žene u sredini“). To se ponekad naziva napad brigade sa kofama, zato što Trudino prosleđivanje podataka između Alise i Boba liči na prosleđivanje kofa vode duž lanca ljudi („brigade sa kofama“) koji gase vatru koristeći udaljeni izvor vode.

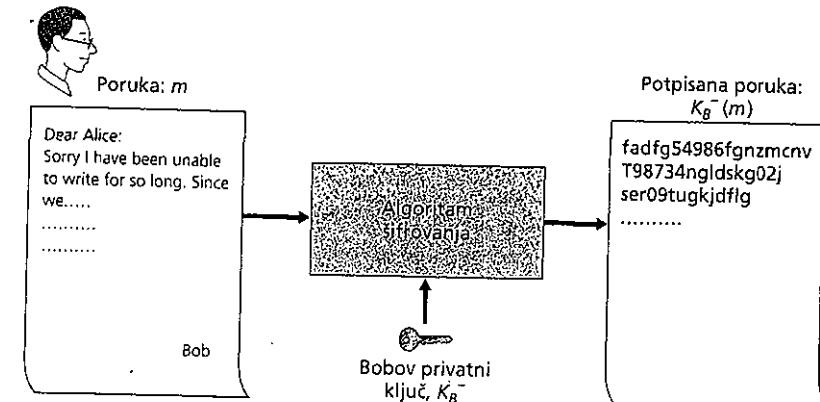
## 8.4 Integritet

Pomislite koliko ste se puta potpisali na parčetu hartije u toku protekle nedelje. Potpisujete Čekove, račune kreditnih kartica, pravna dokumenta i pisma. Vaš potpis potvrđuje da ste vi (nasuprot nekom drugom) potvrdili i/ili se saglasili sa sadržajem dokumenta. U digitalnom svetu, često želimo da ukažemo na vlasnika ili stvaraoca dokumenta, ili da označimo nečiju saglasnost sa sadržajem dokumenta. Digitalni potpis je kriptografska tehnika za postizanje ovih ciljeva u digitalnom svetu. Baš kao i potpsi ljudi, digitalno potpisivanje bi trebalo da se izvrši na takav način da digitalni potpsi budu proverljivi, da se ne mogu falsifikovati ili ne priznavati. Odnosno, mora biti moguće da se „dokaže“ kako je dokument koji je neko potpisao zaista potpisala ta osoba (potpis mora da bude proverljiv) i daje samo ta osoba mogla da potpiše dokument (potpis ne može da se falsificuje, a potpisnik ne može kasnije da porekne ili da ne prizna daje potpisao dokument). To se lako postiže korišćenjem tehnika kriptografije javnim ključem.

### 8.4.1 Stvaranje digitalnih potpisa

Prepostavite da Bob želi da digitalno potpiše „dokument“,  $m$ . O dokumentu možemo da razmišljamo kao o datoteci ili o poruci koju će Bob da potpiše i pošalje. Kao što je prikazano na slici 8.14, da bi potpisao taj dokument, Bob jednostavno koristi svoj privatni ključ,  $K_B$ , da bi izračunao  $K_B^{-1}(m)$ . Na prvi pogled, moglo bi izgledati čudno što Bob koristi svoj privatni ključ (koji se, kao što smo videli u odeljku 8.2, koristi za dešifrovanje poruke koja je bila šifrovana njegovim javnim ključem) da bi potpisao dokument. Ali, setite se da se šifrovanje i dešifrovanje svode na matematičku operaciju (eksponenciranje na stepen e ili *du* algoritmu RSA; pročitajte odeljak 8.2) i da Bobov cilj nije da šifruje ili učini nejasnim sadržaj dokumenta, nego da potpiše dokument na način koji je proverljiv, bez mogućnosti falsifikovanja i neopoziv. Bob ima dokument,  $m$ , a njegov digitalni potpis dokumenta je  $K_B(m)$ .

Da li digitalni potpis,  $K_B(m)$ , ispunjava naše zahteve daje proverljiv, bez mogućnosti falsifikovanja i neopoziv? Prepostavite da Alisa ima  $m$  i  $K_B^{-1}(m)$ . Ona želi da dokaže na sudu (u parnicu) da je Bob zaista potpisao dokument i da je bio jedina osoba koja je uopšte mogla da ga potpiše. Alisa uzima Bobov javni ključ,  $K_B^*$ , i pri-menuje ga na digitalni potpis,  $K_B^{-1}(m)$ , pridružen dokumentu,  $m$ . Odnosno, ona izra-



**Slika 8.14** ♦ Stvaranje digitalnog potpisa za dokument

čunava  $K_B^*(K_B^{-1}(m))$  i dobija  $m$ , koji tačno odgovara originalnom dokumentu! Alisa zatim objašnjava daje samo Bob mogao da potpiše dokument, iz sledećih razloga;

- ♦ Bilo ko daje potpisao poruku morao je da koristi privatni ključ,  $K_B$ , u izračunavanju potpisa  $K_B^*(m)$ , tako daje  $K_B^+(K_B(m)) = m$ .
- ♦ Bob je jedina osoba koja je mogla da zna privatni ključ,  $K_B$ . Setite se iz naše diskusije o algoritmu RSA u odeljku 8.2 da poznавanje javnog ključa,  $K_B^*$ , ne pomaže u saznavanju privatnog ključa,  $K_B$ . Dakle, jedina osoba koja bi znala  $K_B$  jeste osoba koja je stvorila par ključeva,  $(K_B^+, K_B)$ , a to je Bob. (Zapazite da to, međutim, pretpostavlja da Bob nikome nije dao  $K_B$  niti je bilo ko „ukrao“  $K_B$  od Boba.)

Važno je takođe zapaziti da ako je originalni dokument,  $m$ , ikada promenjen u neki drugi oblik,  $m'$ , potpis koji je Bob napravio za  $m$  neće važiti za  $m'$ , zato što  $K_B(K_B(m))$  nije jednako  $m'$ .

Dakle, vidimo da kriptografija javnim ključem obezbeđuje jednostavan i elegantan način da se digitalno potpisuju dokumenti, način koji je proverljiv, ne može se falsifikovati, neopoziv je i štititi od kasnijih promena dokumenta.

### 8.4.2 Izvodi poruka

Videli smo da tehnologija javnog ključa može da se upotrebi da bi se napravio digitalni potpis. Međutim, jedna od nevolja sa potpisivanjem podataka je da su šifrovanje i dešifrovanje računarski skupe operacije. Kada se digitalno potpisuje dokument koji je zaista važan, recimo ugovor o spajanju dve velike multinacionalne korporacije ili ugovor sa detetom da ono jednom nedeljno počisti svoju sobu, računarska cena može da ne bude važna. Međutim, mnogi mrežni uređaji i procesi (na primer,

ruteri koji razmenjuju informacije tabela rutiranja i korisnički agenti za elektronsku poštu koji razmenjuju elektronsku poštu) rutinski razmenjuju podatke koji ne moraju da se šifruju. Svejedno, oni ipak žele da obezbede da:

- ◆ pošiljalac podataka zaista jeste onaj za koga se izdaje, odnosno, daje pošiljalac potpisao podatke i da taj potpis može da se proveri.
- ◆ preneti podaci nisu promenjeni od kada ih je pošiljalac stvorio i potpisao.

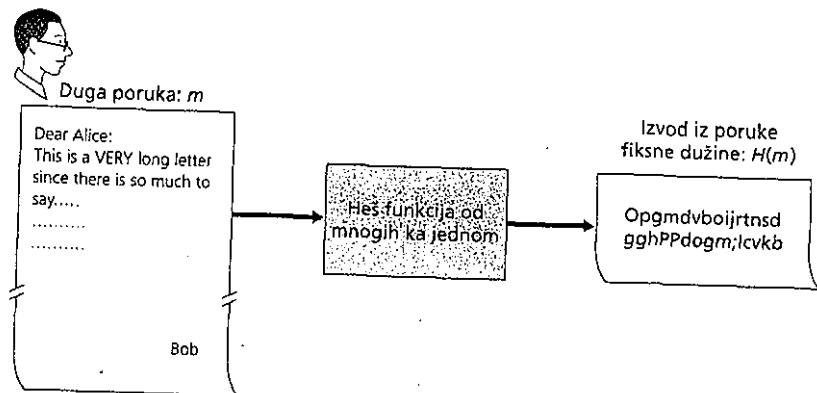
S obzirom na to da operacije šifrovanja i dešifrovanja predstavljaju računaru zaista veliko opterećenje, potpisivanje celog skupa podataka (kompletног dokumenta) može da bude krajnje nepraktično. Efikasniji pristup, upotrebom tzv. izvoda poruka, može da ispunи ova dva cilja bez šifrovanja cele poruke.

Izvod iz poruke u mnogome liči na kontrolni zbir. Algoritam za izvod iz poruke uzima poruku,  $m$ , proizvoljne dužine i proračunava „otisak prsta“ fiksne dužine koji se zove izvod iz poruke,  $H(m)$ . Izvod iz poruke štiti podatke u smislu da ako se  $m$  promeni u  $m'$  (zlonamerno ili slučajno) onda  $H(m)$ , izračunat za originalne podatke (i prenesen sa podacima), neće odgovarati  $H(m')$  izračunatom za promenjene podatke,  $m'$ . Dok izvod iz poruke obezbeđuje integritet podataka, kako on pomaže u potpisivanju poruke  $m$ ? Ovde je cilj da Bob, umesto da potpiše celu poruku izračunavajući  $K_B(m)$ , potpisuje samo izvod iz poruke izračunavajući  $K_B(H(m))$ . Odnosno, posedovanje  $m$  i  $K_B(H(m))$  zajedno (zapazite da  $m$  nije šifrovano) trebalo bi da bude „isto tako dobro“ kao posedovanje potpisane celokupne poruke,  $K_B(m)$ . To znači da se  $m$  i  $K_B(H(m))$  zajedno ne mogu falsifikovati, da su proverljivi i neopoziv. Nemogućnost falsifikovanja zahteva da izvod iz poruke ima neke specijalne osobine, kao što ćemo videti dalje u tekstu.

Naša definicija izvoda poruke može da izgleda sasvim slično kao definicija kontrolnog zbirja (na primer, kontrolnog zbirja za Internet, pročitajte odeljak 3.3.2) ili moćnijeg koda za ispravljanje grešaka, kao što je ciklična provera redundantnosti (pročitajte odeljak 5.2). Da lije ona zaista različita? Kontrolni zbirovi, ciklične provere redundantnosti i izvodi iz poruka - sve su to primeri takozvanih heš funkcija. Kao što je prikazano na slici 8.15, heš funkcija uzima ulaz,  $m$ , i proračunava niz fiksne dužine koji je poznat kao heš. Kontrolni zbirovi za Internet, CRC-ovi i izvodi iz poruka svi zadovoljavaju ovu definiciju. Ako potpisivanje izvoda poruke treba da bude „isto tako dobro“ kao potpisivanje cele poruke, posebno ako će ono da zadovolji zahtev nemogućnosti falsifikovanja, onda algoritam za izvod poruke mora da ima sledeće dodatno svojstvo:

- ◆ Računarski je neizvodljivo pronaći bilo koje dve različite poruke\* i y takve daje  $H(x)=H(y)$ .

Nezvanično, ovo svojstvo znači da je uljezu računarski neizvodljivo da zameni jednu poruku drugom, koja je zaštićena izvodom poruke. Odnosno, ako su  $(m, H(m))$  poruka i njen pripadajući izvod koji je napravio pošiljalac, onda uljez ne može da falsifikuje sadržaj druge poruke,  $y$ , koja ima istu vrednost izvoda kao originalna poruka. Kada Bob potpiše  $m$  izračunavajući  $K_B(H(m))$ , mi znamo da nijedna druga



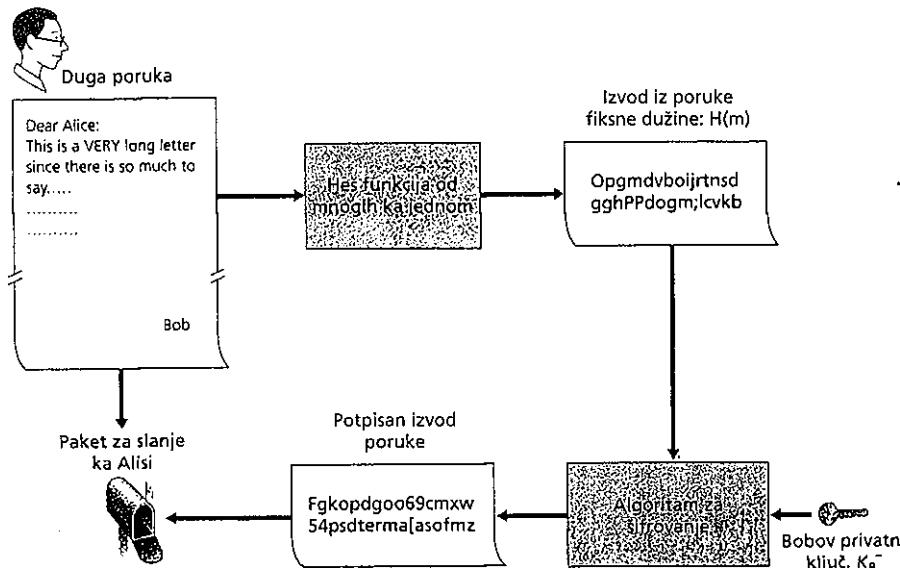
**Slika 8.15** ◆ Heš funkcije se koriste za pravljenje izvoda iz poruka.

poruka ne može da zameni  $m$ . Pored toga, Bobov digitalni potpis  $H(m)$  ga jedinstveno identificuje kao proverljivog, neopozivog potpisnika  $H(m)$  (i, kao posledica, i samu  $m$ ), kao što je razmatrano ranije u odeljku 8.4.1.

U kontekstu Bobovog slanja poruke Alisi, na slici 8.16 dat je zbirni pregled procedure stvaranja digitalnog potpisa. Bob propušta svoju originalnu dugačku poruku kroz heš funkciju da bi napravio izvod poruke. On zatim digitalno potpisuje izvod poruke svojim privatnim ključem. Originalna poruka (u obliku otvorenog teksta) zajedno sa digitalno potpisanim izvodom poruke (na koji ćemo se od sada pozivati kao na digitalni potpis) šalje se Alisi. Na slici 8.17 dat je zbirni pregled procedure proveravanja integriteta poruke. Alisa primenjuje pošiljačev javni ključ na poruku da bi rekonstruisala poslati izvod poruke. Alisa takođe primenjuje heš funkciju na poruku otvorenog teksta da bi dobila izračunati izvod poruke. Ako su ova dva izvoda poruke identična, onda Alisa može da bude sigurna u pogledu integriteta i autora poruke.

#### 8.4.3 Algoritmi heš funkcija

Hajde da se uverimo kako bi jednostavan kontrolni zbir, kao što je kontrolni zbir za Internet, bio slab algoritam za izvod poruke. Umesto da koristimo aritmetiku prvog komplementa (kao u kontrolnom zbiru za Internet), hajde da izračunamo kontrolni zbir obradujući svaki znak kao bajt i dodajući bajtovе jedan drugom koristeći „kri-Ške“ od po četiri bajta istovremeno. Prepostavimo da Bob duguje Alisi 100,99 USD i da joj šalje poruku IOU (poruku u kojoj kaže da joj duguje [I Owe You], prim. prev.) koja se sastoji od tekstualnog niza „IOU100 . 9980B“. ASCII prezentacija

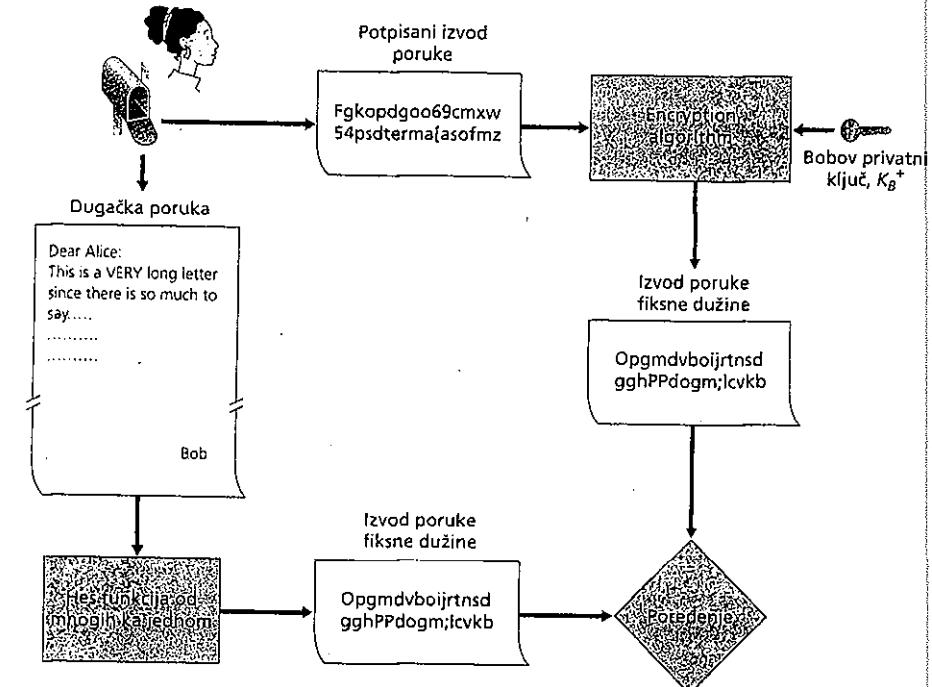


Slika 8.16 ♦ Slanje digitalno potpisane poruke

tih slova (u heksadecimainoj notaciji) je 4 9, 4F, 55, 31, 30, 30, 2E, 3 9, 39, 42, 4F, 42.

Na gornjem delu slike 8.18 pokazano je daje četvorobajtni kontrolni zbir za tu poruku B2 C1 D2 AC. Neznatno različita poruka (ali sa mnogo većim troškom za Boba) prikazanije u donjoj polovini slike 8.18. Poruke „IOU100 . 99BOB“ i „IOU900 . 19BOB“ imaju *isti* kontrolni zbir. Dakle, ovaj jednostavan algoritam kontrolnog zbira krši dva gorna zahteva. Uz date originalne podatke, jednostavno je pronaći drugi skup podataka sa istim kontrolnim zbirom. Jasno je da će nam, iz bezbednosnih razloga, biti potrebna moćnija heš funkcija od kontrolnog zbira.

Algoritam za izvod poruke MD5 čiji je autor Ron Rivest [RFC 1321] danas je u širokoj upotrebi. On proračunava 128-bitni izvod poruke u procesu od četiri koraka, koji se sastoje od koraka dopunjavanja (dodavanja jedinice praćene sa dovoljno nula tako da poruka zadovoljava izvesne uslove), koraka dodavanja (dodavanja 64-bitne prezentacije dužine poruke pre dopunjavanja), inicijalizacije akumulatora i konačnog ulaska u petlju u kojoj se obraduju blokovi poruke od po 16 reči u Četiri runde obrade. Ne zna se da li MD5 stvarno zadovoljava gore navedene zahteve. Autor MD5 tvrdi, „Teškoća dolaženja do bilo koje poruke koja ima isti izvod je reda veli-



Slika 8.17 ♦ Provera integriteta potpisane poruke

ćine  $2^M$  operacija, a teškoća dolaženja do bilo kakve poruke koja ima dati izvod poruke je veličine reda veličine  $2^{128}$  operacija“. Niko nije osporio ovu tvrdnju. Radi opisa poruke MD5 (uključujući implementaciju C izvornog koda) pročitajte [RFC 1321].

Drugi glavni algoritam za izvod poruke danas u upotrebi je SHA-1, Secure Hash Algorithm [FIPS 1995]. Ovaj algoritam se zasniva na principima sličnim onima koji su upotrebljeni u projektovanju poruke MD4 [RFC 1320], prethodnika poruke MD5. Upotreba SHA-1, saveznog standarda u SAD, zahteva se kad god se traži bezbedni algoritam za izvode poruka koje se primenjuju u federalnim institucijama. On daje 160-bitni izvod poruke. Veća izlazna dužina čini SHA-1 sigurnijim.

|         |             | ASCII          |    |    |    |
|---------|-------------|----------------|----|----|----|
| Poruka  | Predstava   | B2             | C1 | D2 | AC |
| I O U 1 | 49 4F 55 31 |                |    |    |    |
| 0 0 . 9 | 30 30 2E 39 |                |    |    |    |
| 9 B O B | 39 42 4F 42 |                |    |    |    |
|         |             | Kontrolni zbir |    |    |    |
|         |             |                |    |    |    |
|         |             | ASCII          |    |    |    |
| Poruka  | Predstava   | B2             | C1 | D2 | AC |
| I O U 9 | 49 4F 55 39 |                |    |    |    |
| 0 0 . 1 | 30 30 2E 31 |                |    |    |    |
| 9 B O B | 39 42 4F 42 |                |    |    |    |
|         |             | Kontrolni zbir |    |    |    |

**Slika 8.18** ♦ Početna poruka i prevarantska poruka imaju isti kontrolni zbir!

## 8.5 Distribucija i overavanje ključeva

U odeljku 8.2 videli smo daje nedostatak kriptografije simetričnim ključem bio potreban da bi se dve strane koje komuniciraju unapred dogovorile o svom tajnom ključu. Kod kriptografije javnim ključem, taj *a priori* sporazum o tajnom ključu nije potreban. Međutim, kao što smo objasnili u odeljku 8.2, kriptografija javnim ključem ima sopstvene teškoće, posebno problem dobijanja nečijeg tačnog javnog ključa. Oba ova problema - određivanje deljenog ključa za kriptografiju simetričnim ključem i bezbedno dobijanje javnog ključa za kriptografiju javnim ključem - mogu da se reše korišćenjem **poverljivog posrednika**. Poverljivi posrednik za kriptografiju simetričnim ključem zove se **centar za distribuciju ključeva** (*key distribution center*, KDC), koji je poverljivi mrežni entitet sa kojim neko može da ima ustanovljen tajni ključ. Videćemo da taj neko može da koristi KDC kako bi dobio deljene ključeve potrebne da bi bezbedno komunicirao sa svim drugim mrežnim entitetima, izbegavajući neke od zamki koje smo otkrili u odeljku 8.3. Za kriptografiju javnim ključem, poverljivi posrednik zove se **sertifikaciono telo** (*certification authority*, CA). Sertifikaciono telo overavlja da javni ključ pripada posebnom entitetu (osobi ili mrežnom entitetu). Za overen *j^vni* ključ, ako neko može bezbedno da veruje CA-u koji je overio taj ključ, onda taj neko može da bude siguran kome javni ključ pripada. Jednom kada je javni ključ overen, on može da se distribuira skoro svuda, uključujući tu server javnih ključeva, ličnu veb stranicu ili disketu.



## PRINCIPI U PRAKSI

### KERBEROS

Kerberos [RFC 1510; Neuman 1994] predstavlja uslužnu autentifikaciju razvijenu na MIT-u, koja koristi tehnike šifrovanja simetričnim ključem i centar za distribuciju ključeva. Mada je konceptualno isti kao generički KDC koji smo opisali u odeljku 8.5.1, njegov rečnik je neznatno drugačiji. Kerberos takođe sadrži nekoliko 2-godinskih varijacija i proširenja osnovnih KDC mehanizama. Kerberos je projektovan da autentikuje korisnike koji pristupaju mrežnim serverima i u početku je bio namenjen za upotrebu unutar jednog administrativnog domena, kao što su univerzitetsko naselje ili kompanija. Dakle, Kerberos je namenjen korisnicima koji žele da pristupaju mrežnim uslugama (serverima) koristeći mrežne programe aplikacionog nivoa, kao što je Telnet (za daljinsko prijavljivanje) i NFS (za pristup udaljenim datotekama), a ne za učesnike u razgovorima između ljudi koji žele da se autentifikuju jedan drugom, kaa u našim prethodnim primerima. Svejedno, *ključne* (igra reči!) osnovne tehnike ostaju iste.

Kerberosov server za autentifikaciju (AS) igra ulogu KDC-a. AS je skladište ne samo tajnih ključeva svih korisnika (tako da svaki korisnik može bezbedno da komunicira sa AS-om) nego i informacija o tome koji korisnici imaju privilegije pristupa kojim uslugama na kojim mrežnim serverima. Kada Alisa želi da pristupi Bobovoj usluzi (o Bobu sada mislimo kao o serveru), protokol tačno sledi naš primer na slici 8.19.

- 1 Alisa uspostavlja vezu sa Kerberos AS-om, pokazujući da želi da koristi Boba. Celakupna komunikacija između Alise i AS-a šifrovana je upotrebom tajnog ključa koji dele Alisa i AS. U Kerberosu, Alisa prvo daje svoje ime i lozinku svom lokalnom računaru. Tada Alisina lokalna računara i AS utvrđuju jednokratni tajni ključ za sesiju za šifrovanje komunikacije između Alise i AS-a.
2. AS autentificuje Alisu, proverava da li ona ima privilegiju pristupa Bobu i generiše jednokratni tajni simetrični ključ sesije, RJ, za komunikaciju između Alise i Boba. Autentifikacioni server (koji se u žargonu Kerberosa zove Ticlef *Grant/mg Server* - server za odobravanje ulaznica), šalje Alisi vrednost *RJ*, kao i ulaznicu za Bobove usluge. Ulaznica sadrži Alisino ime, ključ za sesiju Alisa-Bob i vreme isteka važnosti, sve šifrovano upotrebom Bobovog tajnog ključa (koji znaju samo Bob i AS), kao na slici 8.19. Alisina ulaznica važi samo do trenutka kada joj ističe važnost i Bob će je odbaciti oko mu se pokaže posle 1og Irenutka. Za Kerberos V4, maksimalni životni vek ulaznice je oko 21 sat. U Kerberosu V5, životni vek mora da istekne pre kraja 9999. godine, što je definitivno problem 10000. godine!
3. Alisa tada šalje Bobu svoju ulaznicu. Ona takođe uz to šalje vremensku oznaku šifrovani pomoću *R/I*, kaja se koristi kao jednokratni broj. Bob dešifruje ulaznicu koristeći svoj tajni ključ, dobija ključ sesije i dešifruje vremensku oznaku koristeći ključ sesije koji je upravo saznao. Bob šalje Alisi natrag jednokratni broj, šifrovani pomoću *R/I*, pokazujući na laj način da Bob zna *R/I* i do ide „uživo”.

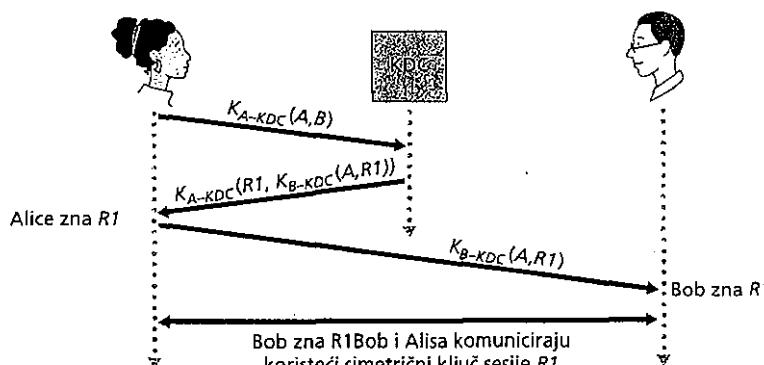
Najnovija verzija Kerberosa (V5) obezbeđuje podršku za više autentifikacionih servera, proveravanje prava pristupa i obnovljive ulaznice. Obilje detalja o svemu ovome nalazi se u knjigama [Kaufmann 1995] i [RFC 1510].

### 8.5.1 Centar za distribuciju ključeva

Prepostavite još jednom da Bob i Alisa žele da komuniciraju koristeći kriptografiju simetričnim ključem. Prepostavite da se oni nisu nikada sreli i zato nemaju una-pred ustanovljen deljeni tajni ključ. Kako oni mogu da se sporazumeju o tajnom ključu, ako mogu da komuniciraju jedno sa drugim samo preko mreže? Rešenje koje se često usvaja u praksi je da se koristi poverljivi centar za distribuciju ključeva (KDC).

KDC je server koji deli različite tajne simetrične ključeve sa svakim registro-vanim korisnikom. Taj ključ može da se instalira ručno na serveru kada se korisnik prvi put registruje. KJDC zna tajni ključ svakog korisnika, a svaki korisnik može da komunicira sa KDC-om koristeći taj ključ. Hajde sada da vidimo kako poznavanje tog ključa dozvoljava korisniku da bezbedno dobije ključ za komuniciranje sa bilo kojim drugim registrovanim korisnikom. Prepostavite da su Alisa i Bob korisnici KDC-a; oni znaju samo svoje individualne ključeve,  $K_{A\text{-}KDC}$ ,  $K_{B\text{-}KDC}$  prema redosledu kojim smo ih ovde naveli, za bezbedno komuniciranje sa KDC-om. Alisa preduzima prvi korak i oni nastavljaju kao stoje prikazano na slici 8.19.

1. Koristeći  $K_{A\text{-}KDC}$  bi šifrovala svoju komunikaciju sa KDC-om, Alisa šalje poruku KDC-u koja kaže da ona (A) želi da komunicira sa Bobom (5). Označavamo ovu poruku kao  $K_{AKDC}(A, B)$ .
2. KDC, poznавajući  $K_{A\text{-}KDC}$ , dešifruje  $K_{A\text{-}mc}(A, B)$ . KDC tada generiše slučajni broj,  $R1$ . To je vrednost deljenog ključa koji će Alisa i Bob koristiti da bi izveli simetrično šifrovanje kada komuniciraju jedno sa drugim. Taj ključ se označava kao ključ za jednokratnu sesiju, zato što će ga Alisa i Bob upotrebiti samo



**Slika 8.19** ♦ Uspostavljanje jednokratnog ključa sesije korišćenjem centra za distribuciju ključeva

za tu jednu sesiju koju trenutno uspostavljaju. KDC sada treba da informiše Alisu i Boba o vrednosti brojač/. KDC šalje natrag Alisi poruku, šifrovani upotreboom ključa  $K_{A\text{-}KDO}$  koja sadrži sledeće:

- ♦  $R1$ , jednokratni ključ za sesiju koji će Alisa i Bob koristiti da bi komunicirali.
- ♦ Par vrednosti: A i  $R1$ , koji je šifrova KDC koristeći Bobov ključ,  $K_{B\text{-}KDC}$ . To označavamo sa  $K_{B\text{-}KDC}(A, R1)$ . Važno je zapaziti da KDC ne šalje Alisi samo vrednost  $R1$  za njenu sopstvenu upotrebu, nego i Šifrovani verziju vrednosti  $R1$  i Alisinog imena, Šifrovani korišćenjem Bobovog ključa. Alisa ne može da dešifruje ovaj par vrednosti u poruci (ona ne zna Bobov ključ za šifrovanje), ali ona to stvarno ne mora ni da radi. Uskoro ćemo videti da će Alisa jednostavno proslediti taj šifrovani par vrednosti Bobu, koji će moći da ga dešifruje.
- ♦ KDC stavlja te stavke u poruku, šifruje ih koristeći Alisin deljeni ključ i šalje ih ka Alisi. Poruka od KDC-a do Alise je  $K_{AmKDC}(R1, K_{B\text{-}qC}(A, R1))$ .
- 3. Alisa prima poruku od KDC-a, dešifruje je i izvlači  $i?$  iz poruke. Alisa sada zna jednokratni ključ sesije,  $R1$ . Alisa takođe izvlači  $K_{B\text{-}mc}(A, R1)$  i to prosledjuje Bobu.
- 4. Bob dešifruje primljenu poruku,  $/C_{B\text{-}CDC}(A, R1)$  koristeći  $K_{B\text{-}KDC}$  i izvlači A i  $R1$ . Bob sada zna jednokratni ključ sesije,  $R1$  i osobu sa kojom deli taj ključ, A. Naravno, on preduzima mere da bi autentifikovao Alisu koristeći  $R1$  pre nego što nastavi bilo šta drugo.

### 8.5.2 Overa javnog ključa

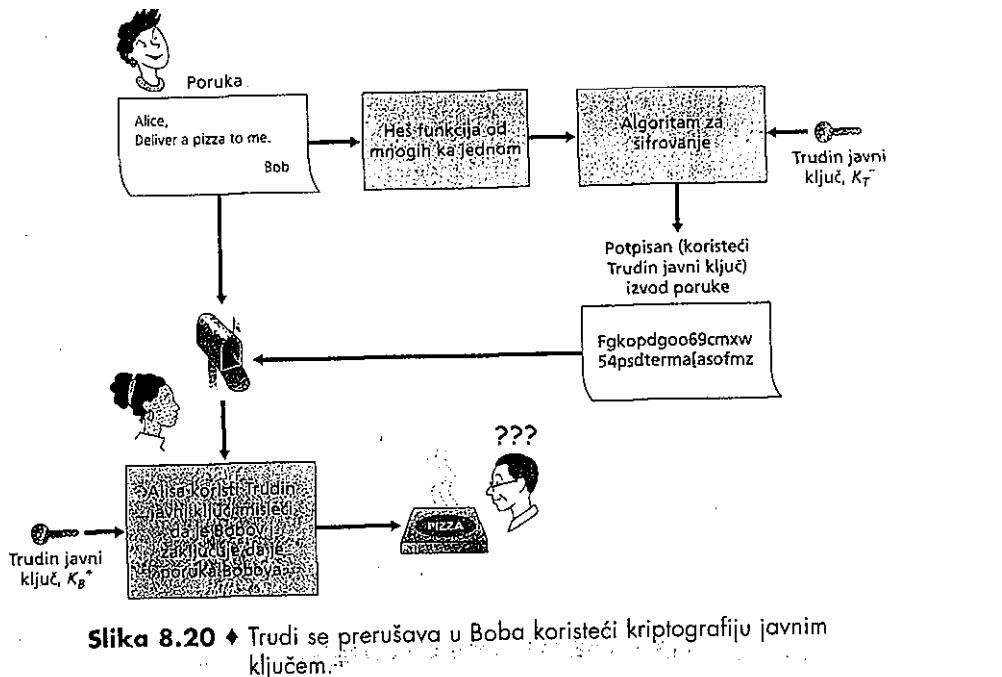
Jedno od glavnih svojstava kriptografije javnim ključem je to što je moguće da dva entiteta razmenjuju tajne poruke bez potrebe da razmene tajne ključeve. Na primer, kada Alisa želi da pošalje Bobu tajnu poruku, ona jednostavno šifruje poruku Bobovim javnim ključem i šalje mu šifrovani poruku; ona ne mora da zna Bobov privatni ključ, niti Bob treba da zna njen privatni ključ. Na taj način, kriptografija javnim ključem izbegava potrebu za infrastrukturom KDC.

Naravno, kod kriptografije javnim ključem, entiteti koji komuniciraju moraju da razmene javne ključeve. Korisnik može da učini svoj javni ključ javno raspoloživim na mnogo načina, na primer, postavljanjem javnog ključa na svoju veb stranicu, postavljanjem ključa na server javnih ključeva, ili slanjem ključa korespondentu preko elektronske pošte. Komercijalna veb lokacija može da postavi svoj javni ključ na svoj server na takav način da pretraživači automatski preuzmu javni ključ kada se povezu sa lokacijom. Ruteri mogu da postave svoje javne ključeve na servere javnih ključeva, dozvoljavajući tako drugim mrežnim entitetima da ih pročitaju.

Međutim, sa kriptografijom javnim ključem postoji suptilan, ali ipak kritičan problem. Da bismo stekli uvid u taj problem, hajde da razmotrimo primer Internet trgovine. Prepostavite da Alisa radi na isporuci pica i da prihvata porudžbine preko Interneta. Bob, ljubitelj pica, šalje Alisi poruku otvorenog teksta koja obuhvata nje-

govu kućnu adresu i vrstu pice koju želi. U toj poruci, Bob takođe uključuje digitalni potpis (odnosno, potpisani izvod poruke ili originalni otvoreni tekst poruke). Kao što smo diskutovali u odeljku 8.4, Alisa može da dobije Bobov javni ključ (sa njegove lične veb stranice, servera javnih ključeva, ili iz poruke elektronske pošte) i potvrdi digitalni potpis. Na taj način, ona je sigurna da je porudžbinu napravio Bob, a ne neki mlađi nestaska.

Sve je to lepo dok se ne pojavi pametna Trudi. Kao Stojko prikazano na slici 8.20, Trudi odlučuje da im podvali. Trudi šalje Alisi poruku u kojoj kaže daje ona Bob, daje Bobovu kućnu adresu i naručuje picu. Ona takođe priklučuje digitalni potpis, ali to radi potpisujući izvod poruke svojim (odnosno, Trudinim) privatnim ključem. Trudi se takođe prerađava u Boba šaljući Alisi Trudin javni ključ, ali rekavši da on pripada Bobu. U ovom primeru, Alisa će primeniti Trudin javni ključ (misleći daje Bobov) na digitalni potpis i zaključiće da je poruka otvorenog teksta zaista napravio Bob. Bob će se veoma iznenaditi kada isporučilac donese u njegovu kuću picu sa svim i svačim na njoj!



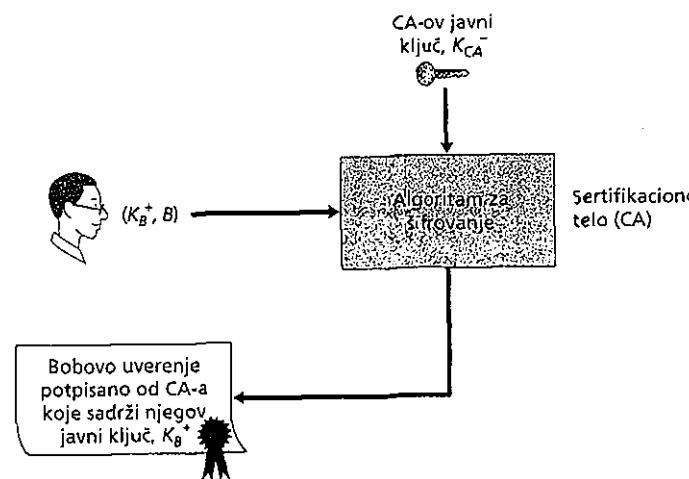
Iz ovog primera vidimo kako, da bi kriptografija javnim ključem bila korisna, entiteti (korisnici, pretraživači, ruteri itd.) treba sigurno da znaju da imaju javni ključ entiteta sa kojim komuniciraju. Na primer, kada Alisa komunicira sa Bobom koristeći kriptografiju javnim ključem, ona treba sigurno da zna da javni ključ za koji se prepostavlja da je Bobov, njemu zaista i pripada. Slične probleme imali smo i u našim protokolima za autentifikaciju na slikama 8.12 i 8.13.

Povezivanje javnog ključa sa nekim posebnim entitetom po pravilu vrši sertifikaciono telo (*Certificate Authority, CA*), čiji posao je da proveri identitete i izda uverenja. CA ima sledeće uloge:

1. CA verifikuje da je entitet (ličnost, ruter itd.) to što kaže da jeste. Nema propisanih procedura kakose vrši overavanje. Kada radimo sa CA-om, moramo da verujemo daje CA izvršio strogu proveru identiteta. Na primer, ako bi Trudi bila u stanju da ušeta u sertifikaciono telo Fly-by-Night, jednostavno objavi „Ja sam Alisa“ i primi uverenja pridružena identitetu „Alisa“, onda ne bismo mogli da imamo mnogo poverenja u javne ključeve koje overava sertifikaciono telo Fly-by-Night. S druge strane, mogli bismo (ili ne bismo mogli!) da imamo više poverenja u CA koji je deo saveznog programa ili programa koji podržava država (na primer, država Utalji ovlašćuje CA-ove u toj državi [Utah 2002]). Možemo da imamo poverenja u „identitet“ pridružen javnom ključu samo onoliko koliko možemo da verujemo CA-u i njegovim tehnikama za proveru identiteta. Kakvu zamršenu mrežu poverenja smo pokrenuli!
2. Jednom kada CA proveri identitet entiteta, CA pravi uverenje koje povezuje javni ključ entiteta sa identitetom. Uverenje sadrži javni ključ i globalno jedinstvenu identifikujuću informaciju o vlasniku javnog ključa (na primer, ime osobe ili IP adresu). Uverenje digitalno potpisuje CA. Ovi postupci prikazani su na slici 8.21.

Hajde da sada vidimo kako uverenja mogu da se iskoriste za borbu protiv pica-šaljivdžija, kao sto je Trudi, i drugih nepoželjnih osoba. Kada Alisa prima Bobovu porudžbinu, ona dobija Bobovo uverenje, koje može da se nalazi na njegovoj veb stranici, u poruci elektronske pošte, ili na serveru za uverenja. Alisa koristi CA-ov javni ključ da bi proverila verodostojnost Bobovog uverenja koje je potpisao CA. Ako prepostavimo da je sam javni ključ CA svima poznat (na primer, on bi mogao da se objavi na poverljivom, javnom i dobro poznatom mestu, kao što je *The New York Times*, tako da je poznat svima i ne može da bude lažan), onda Alisa može da bude sigurna da ona zaista ima posla sa Bobom. Na slici 8.21 ilustrovani su postupci koji su obuhvaćeni šifrovanjem javnog ključa uz posredstvo CA-a. Možete videti CA uverenja uskladištena u vašem pretraživaču Netscape ako izaberete Communi-cator, Tools, Security Info, Certificates... u Internet Exploreru, birajte Tools, Internet Options, Content, Certificates.

I International Telecommunication Union (ITU) i IETF razvili su standarde za sertifikaciona tela. ITU X.509 [ITU 1993] određuje uslugu autentifikacije kao specifi-



**Slika 8.21** ♦ Bob dobija uverenje od sertifikacionog tela.

fičnu sintaksu za uverenja. [RFC 1422] opisuje upravljanje ključevima zasnovano na CA-u za upotrebu sa bezbednom elektronskom poštom na Internetu. On je kompatibilan sa X.509 ali ide dalje od X.509 uspostavljanjem procedura i konvencija za arhitekturu upravljanja ključevima. U tabeli 8.3 opisana su neka značajna polja u uverenju.

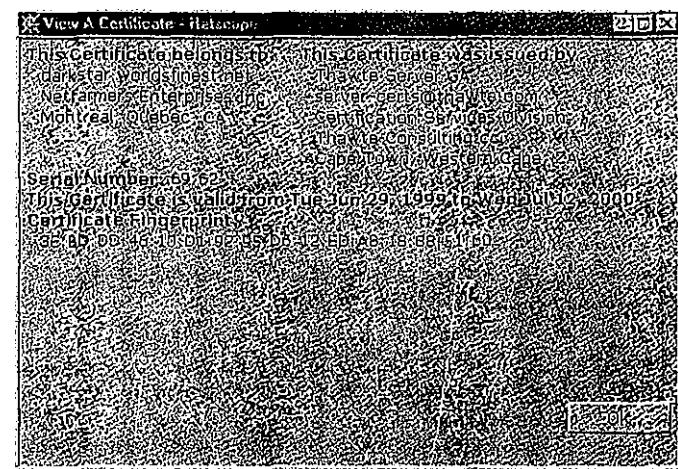
| Naziv polja          | Opis                                                                                                                            |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Verzija              | Broj verzije specifikacije X.509                                                                                                |
| Serijski broj        | Jedinstveni identifikator uverenja koji je izdao CA                                                                             |
| Potpis               | Određuje algoritam koji CA koristi da bi „potpisao“ ovo uverenje                                                                |
| Naziv izdavača       | Identitet CA koji izdaje ovo uverenje, u takočvanom DN formatu (distinguished name, prepoznajljiva ime) [RFC 2253]              |
| Period važenja       | Početak i kraj perioda važenja uverenja                                                                                         |
| Ime subjekta         | Identitet entiteta čiji je javni ključ pridružen ovom uverenju, u DN formatu                                                    |
| Javni ključ subjekta | Javni ključ subjekta kao i indikacija algoritma javnog ključa (i parametri algoritma) koji treba da se upotrebti sa tim ključem |

**Tabela 8.3** ♦ Odabrana polja u specifikaciji X.509 i RFC 1422 uverenju za javne ključeve

Sa nedavnim naglim razvojem elektronske trgovine i široko rasprostranjenom potrebotom za bezbednim transakcijama, povećalo se zanimanje za sertifikaciona tela. Između ostalih kompanija koje obezbeđuju usluge CA-a, tu su Digital Signature Trust Companv [Digital Signature 2004] i Verisign [Verisign 2004]. Uverenje koje je kompanija Thawte Consulting izdala firmi Netfarmers Enterprisers, Inc., onako kao što se vidi pomoću pretraživača Netscape, prikazano je na slici 8.22.

## 8.6 Kontrola pristupa: mrežne barijere

U ovom poglavlju videli smo da Internet nije baš sigurno mesto - tamo su „loši momci“, koji prave sve same propasti. Sa tačke gledišta mrežnog administratora, svet se sasvim jasno deli na dva tabora - dobre momke (koji pripadaju organizaciji koja upravlja mrežom i koji bi trebalo da mogu da pristupaju resursima unutar administratorove mreže na relativno neograničen način) i loše momke (svi ostali, čiji pristup mreži treba pažljivo razmatrati). U mnogim organizacijama, od srednje-vekovnih zamkova do savremenih zgrada korporativnih kancelarija, postoji jedna tačka za ulaz/izlaz gde se i dobri i loši momci koji ulaze i napuštaju organizaciju bezbednosno proveravaju. U zamku, to se radilo na kapiji najednom kraju pokretnog mosta; u korporativnoj zgradi, to se radi na Šalteru za bezbednost/prijem. U računarskoj mreži, kada se saobraćaj koji ulazi/napušta mrežu bezbednosno proverava, prijavljuje, odbacuje i/ili prosleduje, to se radi na uređaju koji se zove mrežna barijera.



**Slika 8.22** ♦ Uverenje koje je kompanija Thawte Consulting izdala firmi Netfarmers Enterprisers, Inc.

**Mrežna barijera (firewall)** jeste kombinacija hardvera i softvera koja izoluje unutrašnju mrežu organizacije od Interneta, dozvoljavajući nekim paketima da prođu i blokirajući ostale. Mrežna barijera dopušta administratoru mreže da kontroliše pristup između spoljašnjeg sveta i resursa unutar administrirane mreže upravljanjem tokom saobraćaja ka resursima i od tih resursa. Na slici 8.23 prikazana je mrežna barijera, koja se nalazi na granici između administrirane mreže i ostatka Interneta. Dok velike organizacije mogu da koriste višestruke nivoje mrežnih barijera ili distribuirane mrežne barijere [Ioannidis 2000], postavljanje mrežne barijere na jednu tačku pristupa mreži, kao sto je prikazano na slici 8.23, olakšava upravljanje i sprovođenje politike bezbednosnog pristupa.

Postoje dve vrste mrežnih barijera: **mrežne barijere sa filtriranjem paketa** (koje rade u mrežnom sloju) i **mrežni prolazi aplikacionog nivoa** (koji rade u aplikacionom sloju). Objasnićemo svaku od njih u sledeća dva pododeljka.

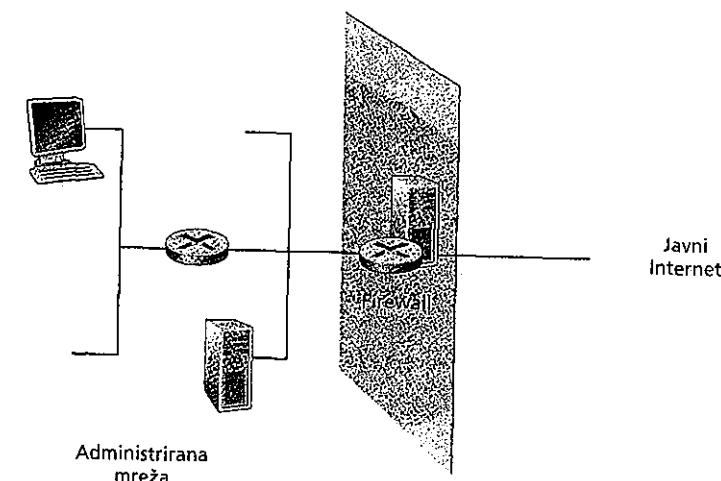
### 8.6.1 Filtriranje paketa

Kao sto je prikazano na slici 8.23, organizacija obično ima ruter mrežnog prolaza koji povezuje njenu internu mrežu sa ISP-om (i na taj način sa javnim Internetom). Ceo saobraćaj koji ulazi u internu mrežu i izlazi sa nje prolazi kroz ovaj ruter i tu dolazi do **filtriranja paketa**. Filtri paketa rade tako što prvo sintaksno analiziraju zaglavljena datagrama, a zatim primenjuju pravila filtriranja iz skupa pravila koji je defmisao administrator, da bi odredili da li da odbace datagram ili da ga puste da prođe. Odluke o filtriranju po pravilu se zasnivaju na:

- ◆ IP adresi izvora ili odredišta;
- ◆ TCP ili UDP portu izvora ili odredišta;
- ◆ vrsti poruke ICMP (Internet protokol za kontrolu poruka);
- ◆ datagramima za inicijalizaciju konekcije korišćenjem bitova TCP SYN ili ACK.

Kao jednostavan primer, filter može da se podeši da blokira sve UDP segmente i sve Telnet konekcije. Takva konfiguracija sprečava one koji nisu članovi da se prijavljuju na interne računare koristeći Telnet i članove da se prijavljuju na spoljašnje računare koristeći Telnet konekcije, blokirajući sve TCP segmente (od kojih je svaki enkapsuliran u datagram) čiji je broj porta izvora ili odredišta 23 (što odgovara Tel-netu). Filtriranje UDP saobraćaja je popularna politika u korporacijama - na veliku žalost vodećih prodavača živog zvuka i videa, čiji proizvodi teku preko UDP-a. Filtriranje Telnet konekcija je takođe popularno, zato što sprečava uljeze spolja da se prijavljuju na internim mašinama. [CERT Filtering 2002] daje listu preporučenih paketnih filtriranja porta/protokola da bi se izbegao izvestan broj dobro poznatih pukotina u bezbednosti postojećih mrežnih aplikacija.<sup>6</sup>

Politika filtriranja može takođe da se zasniva na kombinaciji brojeva adrese i porta. Na primer, ruter za filtriranje može da prosleduje sve Telnet datagrame (one sa brojem porta 23) izuzev onih koji odlaze sa liste posebnih IP adresa i dolaze na nju. Takva politika dozvoljava Telnet konekcije ka računaru sa dozvoljene liste i od



Slika 8.23 ♦ Postavljanje mrežne barijere između administrirane mreže i spoljašnjeg sveta

njega. Na nesreću, zasnivanje politike na spoljasnjim adresama ne obezbeđuje nikavu zaštitu od datagrama koji imaju adresu izvora koja pripada računaru sa dozvoljene liste, ali su u stvari poslati sa drugog računara (najverovatnije vlasništva „lošeg momka“) koji je falsifikovao adresu izvora datagrama. Takvu IP prevaru ispitaćemo u sledećem odeljku.

Filtriranje takođe može da se zasniva na tome da li je bit TCP ACK postavljen ili nije. Ovaj trik je sasvim koristan ako organizacija želi da dopusti svojim internim klijentima da se povezuju sa spoljasnjim serverima, ali želi da spreči spoljašnje klijente da se povezuju sa internim serverima. Setite se iz odeljka 3.5 da prvi segment u svakoj TCP konekciji ima bit ACK podešen na 0, dok svi drugi segmenti u konekciji imaju bit ACK podešen na 1. Dakle, ako organizacija želi da spreči spoljašnje klijente da započnu konekcije sa internim serverima, ona jednostavno filtrira sve dolazne segmente sa bitom ACK postavljenim na 0. Ovakva politika obustavlja sve TCP konekcije koje potiču iz spoljašnjeg sveta, ali dozvoljava konekcije koje su stvorene iznutra.

Iako ovi primjeri mogu da daju utisak kako je dosta lako odrediti pravila filtriranja koja implementiraju datu politiku bezbednosti, u toj oblasti ima mnogo suptilnosti i mogućih zamki. Da bismo ilustrovali neka od tih pitanja, hajde da razmotrimo primer, prilagođen izknjige [Chapman 1992]. Filtriranje paketa radi tako što sekven-cijalno proverava pravila filtriranja na datagramu koji se ispituje; prvo pravilo koje

odgovara datagramu određuje akciju koja će da se preduzme. Evo našeg scenarija. Prepostavite da Alisa administriše mrežom kompanije 222.22.0.0/16 i da, uopšte, želi da zabrani pristup svojoj mreži sa javnog Interneta (pravilo R3 u tabeli 8.4). Međutim, Alisa saraduje sa Bobom i njegovim kolegama, koji su na univerzitetu, pa Alisa želi da dozvoli korisnicima sa Bobovog univerziteta (Čija je mrežna adresa 111.11.16) pristup posebnoj podmreži, 222.22.22/24 unutar mreže njene kompanije (pravilo R1). Činilac koji sve ovo otežava je to što Alisa zna daje Trudi, dobro poznali haker, na Bobovom univerzitetu i da je Trudina podmreža, 111.11.11/24, pravi raj za nepouzdane hakere. Zato Alisa želi da nikakav saobraćaj iz podmreže 111.11.11/24 ne uđe nigde u njenu mrežu (pravilo R2). Alisina pravila filtriranja paketa data su u tabeli 8.4.

U tabeli 8.5 prikazano je kako Alisina korporativna mrežna barijera radi sa odabranim datagramima kada se pravila primenjuju u jednom od dva različita redosleda. Kada se pravila primenjuju u redosledu prvo najspecifičnije adrese: R2, R1, R3 (u pretposlednjoj koloni tabele 8.5), dobijamo željene rezultate: datagrami kao što je P1 koji imaju izvor u hakerskoj podmreži i odredište u korporativnoj mreži, ali van specijalne podmreže, ne propuštaju se kroz mrežnu barijeru (po pravilu R2). Slično tome, datagrami kao Stope P2 iz hakerske podmreže koji za odredište imaju specijalnu podmrežu ne propuštaju se kroz mrežnu barijeru (po pravilu R2). Pod redosledom pravila R2, R1, R3, datagrami iz opšte univerzitetske mreže propuštaju se ka specijalnoj podmreži (datagram P3) ali ne i u druge delove korporativne mreže (datagram P4).

Prepostavite, međutim, da se pravila primenjuju u redosledu R1, R2, R3 (poslednja kolona u tabeli 8.5). U tom slučaju, datagramu P2 se pogrešno dozvoljava pristup specijalnoj podmreži pod pravilom R1, žalo što se adrese izvora i odredišta P2 podudaraju sa specifikacijom izvora i odredišta za pravilo R1, a pravilo R1 se primenjuje kao prvo. Lekcija koju smo naučili iz ovog jednostavnog primera sa tri pravila je jasna - važan je redosled procene pravila mrežne barijere. Naravno, zahteva se pažljivo razmišljanje - zamislite teškoće kada se određuju mrežne barijere sa hiljadama pravila! Moglo bi se pomisliti da primenjivanje prvo specifičnijih pravila uvek izbegava nepredviđeno ili neželjeno ponašanje koje nastaje usled pitanja redosleda. Međutim, videćemo u problemima koji su dati u domaćem zadatku na kraju poglavlja da to ne mora uvek tako da bude.

| Pravilo | Adresa izvora | Adresa odredišta | Akcija  | Komentari                                                             |
|---------|---------------|------------------|---------|-----------------------------------------------------------------------|
| R1      | 111.11/16     | 222.22.22/24     | dopušta | Pušta datagrame iz Bobove univerzitetske mreže u ograničenu podmrežu. |
| R2      | 111.11.11/24  | 222.22.16        | odbija  | Ne pušta saobraćaj iz Trudine podmreže nigde u Alisinoj mreži.        |
| R3      | 0.0.0.0/0     | 0.0.0.0/0        | odbija  | Ne pušta saobraćaj u Alisinoj mreži.                                  |

Tabela 8.4 ♦ Pravila za filtriranje paketa

| Broj datograma | IP adresa izvora                                             | IP adresa odredišta                   | Poželjna akcija | Akcija pod R2, R1, R3 | Akcija pod R1, R2, R3 |
|----------------|--------------------------------------------------------------|---------------------------------------|-----------------|-----------------------|-----------------------|
| P1             | 111.11.11.1<br>(hakerska podmreža)                           | 222.22.6.6<br>(korporativna podmreža) | odbija          | odbija (R2)           | odbija (R2)           |
| P2             | 111.11.11.1<br>(hakerska podmreža)                           | 222.22.22.2<br>(specijalna podmreža)  | odbija          | odbija (R2)           | dopušta (R1)          |
| P3             | 111.11.6.6<br>(univerzitetska mreža,<br>nehakerska podmreža) | 222.22.22.2<br>(specijalna podmreža)  | dopušta         | dopušta (R1)          | dopušta (R1)          |
| P4             | 111.11.6.6<br>(univerzitetska mreža,<br>nehakerska podmreža) | 222.22.6.6<br>(korporativna podmreža) | odbija          | odbija (R3)           | odbija (R3)           |

Tabela 8.5 ♦ Rezultati filtriranja paketa, prema redosledu pravila

## 8.6.2 Aplikacioni mrežni prolaz

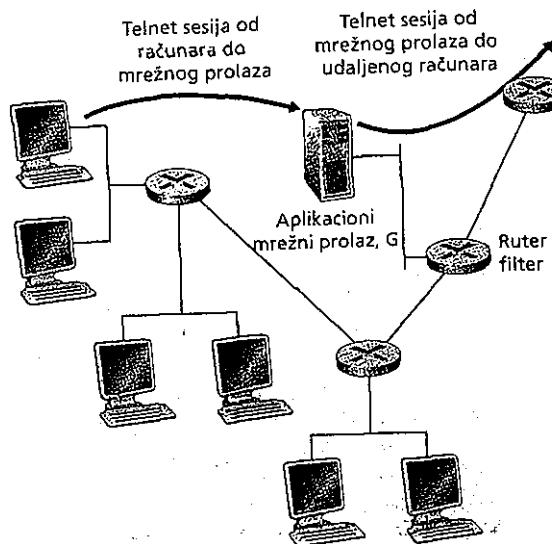
U prethodnim primerima, videli smo da filtriranje na nivou paketa dozvoljava organizaciji da grubo filtrira IP i TCP/UDP zaglavla, uključujući IP adrese, brojne portove i bitove potvrde. Na primer, videli smo da filtriranje zasnovano na kombinaciji IP adresa i brojeva portova može da dopusti internim klijentima da ostvare Telnet konekcije spolja, dok istovremeno sprečava spoljašnje klijente da obavljaju Telnet sesije unutar interne mreže. Ali, šta ako organizacija želi da obezbedi uslugu Telnet ograničenom skupu internih korisnika (prema njihovim IP adresama)? I, šta ako organizacija želi da se tako privilegovani korisnici autentikuju pre nego što im se dozvoli da stvore Telnet sesije prema spoljašnjem svetu? Takvi zadaci su izvan mogućnosti filtra. Zaista, informacija o identitetu internih korisnika nije uključena u IP/TCP/UDP zaglavla, nego se, umesto toga, nalazi u podacima aplikacionog sloja.

Da bismo imali bezbednost finijeg nivoa, mrežne barijere moraju da kombinuju filtre paketa sa aplikacionim mrežnim prolazima. Aplikacioni mrežni prolazi gledaju dalje od IP/TCP/UDP zaglavla i prave odluke upravljanja zasnovane na aplikacionim podacima. Aplikacioni mrežni prolaz je aplikaciono specifičan server kroz koji moraju da prođu svi podaci aplikacije (kako ulazni tako i izlazni). Najednom računaru može da radi više aplikacionih mrežnih prolaza, ali svaki od njih je poseban server sa sopstvenim procesima.

Da bismo dobili neki uvid u aplikacione mrežne prolaze, hajde da konstruišemo mrežnu barijeru koja dozvoljava samo ograničenom skupu internih korisnika da radi Telnet spolja i sprečava sve spoljašnje klijente da rade Telnet unutra. Takva politika može da se ostvari implementiranjem kombinacije filtra paketa (u ruteru) i Telnet mrežnog prolaza, kao što je prikazano na slici 8.24. Ruterov filter je konfigurisan da

blokira sve Telnet konekcije izuzev onih koje potiču sa IP adresu aplikacionog mrežnog prolaza. Takva konfiguracija filtra prisiljava sve odlazne Telnet konekcije da prodru kroz aplikacioni mrežni prolaz. Zamislite sada internog korisnika koji želi da radi Telnet sa spoljasnjim svetom. Korisnik mora prvo da uspostavi Telnet sesiju sa aplikacionim mrežnim prolazom. Aplikacija radi kroz mrežni prolaz, koji osluškuje dolazne Telnet sesije, traži od korisnika njegov ID i lozinku. Kada korisnik obezbedi te informacije, aplikacioni mrežni prolaz proverava kako bi video da li korisnik ima dozvolu da radi Telnet sa spoljasnjim svetom. Ako nema, mrežni prolaz obustavlja Telnet konekciju internog korisnika sa mrežnim prolazom. Ako korisnik ima dozvolu, onda mrežni prolaz (1) traži od korisnika ime spoljašnjeg računara sa kojim želi konekciju, (2) uspostavlja Telnet sesiju između mrežnog prolaza i spoljašnjeg računara i (3) prenosi spoljašnjem računaru sve podatke koji stižu od korisnika, a korisniku sve podatke koji stižu od spoljašnjeg računara. Na taj način Telnet aplikacioni mrežni prolaz ne samo da izvršava ovlašćenje korisnika nego takođe radi kao Telnet server i Telnet klijent, prenoseći informacije između korisnika i udaljenog Telnet servera. Zapazite da će filter dozvoliti korak 2 zato što mrežni prolaz započinje Telnet konekciju prema spoljašnjem svetu.

Interne mreže Često imaju više aplikacionih mrežnih prolaza, na primer, mrežne prolaze za Telnet, HTTP, FTP i elektronsku poštu. U stvari, server za poštu organizacije (pročitajte odeljak 2.4) i keš memorija za Web su aplikacioni mrežni prolazi.



**Slika 8.24** ♦ Mrežna barijera koja se sastoji od aplikacionog mrežnog prolaza i filtra.

Ni aplikacioni mrežni prolazi nisu bez nedostataka. Prvo, potreban je različit aplikacioni mrežni prolaz za svaku aplikaciju. Drugo, postoji cena u performansi koja mora da se plati, zato što će se svi podaci prenositi preko mrežnog prolaza. To postaje nevolja, posebno kada više korisnika ili aplikacija koriste istu mašinu za mrežni prolaz. Najzad, mora da se preduzme i dodatni napor prilikom konfigurisanja:

- ◆ softver klijenta mora da zna kako da uspostavi kontakt sa mrežnim softverom umesto spoljašnjeg servera kada korisnik uputi zahtev, i mora da zna kako da kaže aplikacionom mrežnom prolazu sa kojim spoljašnjim serverom treba da se poveže, ili
- ◆ korisnik mora izričito da se poveže sa spoljašnjim serverom preko aplikacionog mrežnog prolaza.

Zaključujemo ovaj odeljak napomenom da mrežne barijere ni u kom slučaju nisu rešenje za sve probleme bezbednosti. One uvode kompromis između stepena komunikacije sa spoljašnjim svetom i nivoa bezbednosti. S obzirom na to da filtri ne mogu da spreče prevare u vezi sa IP adresama i brojevima portova, oni često koriste politiku „sve ili ništa“ (na primer, zabranu celokupnog UDP saobraćaja). Mrežni prolazi takođe mogu da imaju softverske greške, dozvoljavajući napadačima da prodru u njih. Najzad, mrežne barijere su čak i manje efikasne ako interno generisane komunikacije mogu da dodu do spoljašnjeg sveta bez prolaza kroz mrežnu barijeru. Dva takva primera su bežične komunikacije i modemii.

## 8.7 Napadi i kontramere

Pošto smo ispitali šifrovanje/dešifrovanje, autentifikaciju, integritet poruka, distribuciju ključeva i mrežne barijere, hajde sada da vidimo kako ove tehnike mogu da se upotrebe u borbi protiv različitih napada na mreže. Neki od najsenzacionalnijih bezbednosnih napada, kao što su CodeRed [CERT 2001-19], virus Melissa [CERT 1999-04] i crv Slammer [CERT 2003-04, Moore 2003], koriste Internet da bi se širili, ali napadali samo operativne sisteme (prepunj avaj uči privremene memorije u Microsoftovom IIS serveru u slučaju CodeReda) ili aplikacioni softver (Microsoft Word u slučaju virusa Melissa), a ne samu mrežu. S obzirom na to daje ovo knjiga o mrežama, ovde ćemo se usredosrediti na napade koji iskorišćavaju mrežu, napadaju je, ili se bave njom na neki poseban način.

### 8.7.1 Preslikavanje

U „stvarnom svetu“ (suprotno virtualnom svetu) napadu obično prethodi prikupljanje infonnacija. Gangsteri iž filmova „ostvaruju kontakt“; vojnici „izvidaju oblast“. Svrha je jasna - Što se više zna o meti pre napada, manje je verovatno da ćemo biti uhvaćeni i veći su izgledi na uspeh. To je istina i u virtualnoj stvarnosti. Pre napada na mrežu, napadači bi voleli da znaju IP adrese mašina u mreži, operativne sisteme

koje one koriste i usluge koje one nude. Sa tim informacijama, njihovi napadi će biti usredstveniji i manje je verovatno da će izazvati uzbunu. Proces prikupljanja tih informacija poznat je kao **preslikavanje**.

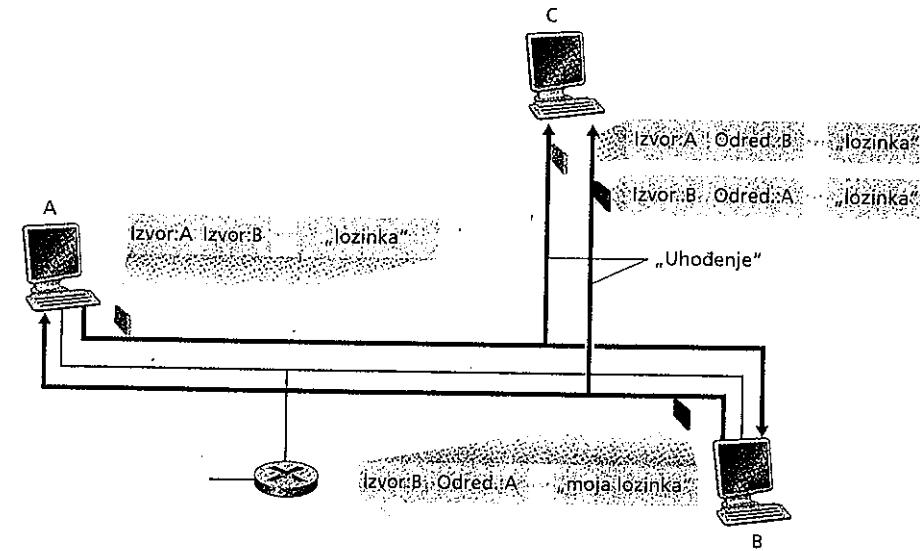
Program kao sto je ping može da se upotrebni da bi se utvrdile IP adrese mašina u mreži jednostavnim posmatranjem koje adrese odgovaraju na poruku ping. **Ske-niranje portova** odnosi se na tehniku sekvencijalnog kontaktiranja (preko zahteva za TCP konekciju, ili preko jednostavnog UDP datagrama) brojeva portova na mašini i gledanja što se dešava kao odgovor. Ti odgovori, sa svoje strane, mogu da se upotrebe da bi se odredile usluge (na primer, HTTP ili FTP) koje mašina nudi. Nmap [Nmap 2004] je široko korišćen pomoći program otvorenog koda za „istraživanje mreža i reviziju bezbednosti“ koji skenira portove. Mnoge mrežne barijere, kao što su one koje prodaje Checkpoint [Checkpoint 2004], otkrivaju preslikavanje i aktivnost posle skeniranja, kao i druge takve „zlonamerne aktivnosti“ i prijavljaju ih upravljaču mreže.

### 8.7.2 Uhodenje paketa

**Uhoda paketa** (packet sniffer) je program na uređaju priključenom na mrežu, koji pasivno prima sve okvire sloja podataka koji prolaze kroz mrežni adapter uređaja. U difuznom okruženju, kao što je Ethernet LAN, to znači da uhoda paketa prima sve okvire koji se prenose ka svim računarima u LAN-u ili iz njih. Svaki računar sa Ethernet karticom može lako da posluži kao uhoda paketa, zato što Ethernet adapter treba samo da se podesi na zajednički **režim** (*promiscuous mode*) da bi primao sve Ethernet okvire koji prolaze. Ti okviri, sa svoje strane, mogu da se proslede aplikacionim programima koji izvlače podatke aplikacionog nivoa. Na primer, u Telnet scenaru prikazanom na slici 8.25, lozinka za prijavljivanje poslata od A do B, kao i lozinka unesena u B „uhode se“ u računaru C. Pošto su dobili lozinke za korisničke naloge, napadači mogu kasnije da upotrebe te naloge da bi započeli napade odbijanja usluga, o čemu ćemo govoriti dalje u tekstu. Uhodenje paketa je mač sa dve oštice - ono može biti dragoceno mrežnom administratoru za nadgledanje i upravljanje mrežom (procitajte poglavljje 9), ali takođe može da ga upotrebi i neki haker.

Softver za uhodenje paketa je besplatno dostupan na različitim veb lokacijama, ali i u vidu komercijalnih proizvoda. Poznati su slučajevi profesora koji su na kursevima iz mreža davali laboratorijske vežbe koje su obuhvatale i pisanje programa za uhodenje paketa i rekonstrukciju podataka aplikacionog nivoa. Zaista, Ethereal laboratorijske vežbe [Ethereal 2004], pridružene ovom tekstu (procitajte uvodnu Ethereal laboratorijsku vežbu na kraju poglavљa 1) koriste tačno takav program za uhodenje paketa!

Ključ za otkrivanje uhodenja paketa je da se detektuju mrežni interfejsi koji rade u zajedničkom režimu. Unutar preduzeća, mrežni administratori mogu da instaliraju softver u svim računarima preduzeća koji će ubuntiti administratore kada se neki interfejs konfiguriše za rad u zajedničkom režimu. Da bi se otkrili interfejsi u zajedničkom režimu, daljnji mogu da se izvedu različiti trikovi. Na primer,



**Slika 8.25** ♦ Uhodenje paket.

računar koji odgovara na datagram ICMP echo (odnosno, koji izaziva ICMP echo odgovor; pogledajte sliku 4.21), koji je ispravno adresiran u IP datagramu ali sadrži netačnu adresu MAC (nivoa okvira), verovatno ima svoj interfejs konfiguriran da radi u zajedničkom režimu. Ključna stvar da se preživi uz uhodenje paketa je da se šifruju svi podaci (posebno lozinke) koji prolaze preko mrežnog linka.

### 8.7.3 Varanje

Svaki uređaj povezan sa Internetom obavezno šalje IP datagrame u mrežu. Setite se iz poglavљa 4 da ti datagmani nose pošiljačevu IP adresu, kao i podatke iz višeg sloja. Korisnik sa potpunom kontrolom nad softverom uređaja (posebno njegovim operativnim sistemom) može lako da promeni protokole uređaja tako da postave proizvoljnu IP adresu u datagramovo polje za adresu izvora. To se zove IP varanje (*IP spoofing*). Korisnik na taj način može da napravi IP paket koji po želji sadrži bilo kakve podatke (iz višeg sloja) u korisnom teretu i izgleda kao da su podaci poslati iz proizvoljnog IP računara. IP varanje se često koristi u napadima odbijanja usluge da bi se sakrio organizator napada, o čemu ćemo govoriti kasnije. Sa lažnom IP adresom izvora u datagramu, teško je pronaći računar koji je stvarno poslao datagram.

Sa tehničke tačke gledišta, varanje može lako da se spreči. Ruteri koji izvršavaju **filtriranje pristupa** proveravaju IP adresu dolazećih datagrama i utvrđuju da li je adresa izvora u opsegu mrežnih adresa za koje se zna da su dostižne preko tog

interfejsa. Ta provera može lako da se izvede na ivici mreže, na primer, na korporativnom mrežnom prolazu ili mrežnoj barijeri, gde se zna da su svi računari u korporaciji unutar datog opsega adresa. Filtriranje pristupa se smatra „trenutno najboljom praksom“ [RFC 2827] na Internetu. Mada tehnički jednostavno, filtriranje pristupa ne može da se naredi, pa je zato teško da se postavi sa socijalne tačke gledišta. Kao rezultat, filtriranje pristupa se ne primenjuje univerzalno.

#### 8.7.4 Napadi odbijanja usluge i distribuiranog odbijanja usluge

Široku klasu bezbednosnih pretnji čine napadi **odbijanja usluge** (*denial-of-service*, DoS). Kao što im ime nagovještava, DoS napadi Čine mrežu, računar, ili neki deo mrežne infrastrukture neupotrebljivim za legitimne korisnike. DoS napad obično deluje tako što stvara toliko mnogo posla za infrastrukturom koja mu je izložena, da legitimni rad ne može da se obavi. U takozvanom napadu SYN **poplave** [CERT SYN 1996], napadač potapa server paketima TCP SYN, od kojih svaki ima lažnu IP adresu izvora. Server, nemoćan da napravi razliku između legitimnog i lažnog SYN-a, završava drugi korak TCP potvrđivanja spremnosti za prenos (pročitajte odeljak 3.5.6) za lažni SYN, dodeljujući strukturu i stanje podataka. Napadač nikad ne izvrši treći korak trostrukre sinhronizacije, ostavljajući za sobom delimično ostvarene konekcije čiji se broj stalno povećava. Opterećenje SYN paketima koji treba da se obrade i iscrpljivanje slobodne memorije na kraju bacu server na kolena. Srođan oblik napada šalje IP fragmente računaru, ali ih ne šalje nikada dovoljno da bi se upotpunio datagram. Napadnuti računar nastavlja da akumulira fragmente, uzalud Čekajući one koji bi upotpunili datagram, trošeći vremenom sve veću količinu memorije. Napad „**štrumfova**“ [CERT Smurf 1998] radi tako što veliki broj nevinih računara odgovara na pakete zahteva za ICMP echo (pročitajte odeljak 4.4.3) koji sadrže lažnu IP adresu izvora. To ima za rezultat veliki broj paketa odgovora na ICMP echo koji se šalju računaru čija je IP adresa bila lažna.

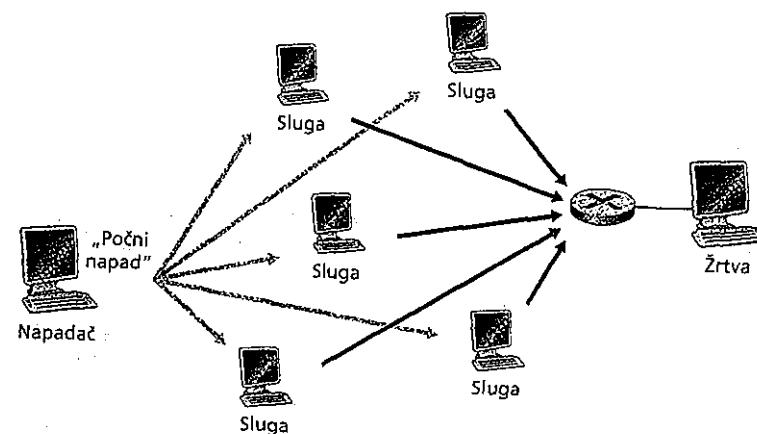
U napadu **distribuiranog odbijanja usluge** (*distributeri denial-ofservice*, DDoS), napadač najpre dobija pristup korisničkim nalozima na brojnim računarima na Internetu (na primer, uhodenjem lozinki ili drugim vrstama prodora u korisnikov nalog). Kao što je prikazano na slici 8.26, napadač onda instalira i pušta u rad program-slugu na svakoj od lokacija, koji tiho čeka komandu od programa-gospodara. Sa velikim brojem takvih programa-slugu koji se izvršavaju, program-gospodar onda uspostavlja kontakt i nalaže svakom od njih da započne napad odbijanja usluge usmeren na isti ciljni računar. Rezultujući koordinirani napad je posebno razoran, zato što dolazi sa tako mnogo napadačkih računara u isto vreme.

Napadi odbijanja usluge su punili novinske naslove u februaru 2000. godine, kada su bile napadnute eBay, Yahoo, CNN i druge velike veb lokacije [Cnet 2000]. Teško je zaštititi se od DoS i (čak i više) od DDoS napada. Filtriranje paketa nije lako, zato što je teško razlikovati dobre od loših datagrama. Na primer, kako mrežna barijera za filtriranje paketa može da zna da li je dolazeći SYN za legitimnu kone-

kciiju (na primer, kupac koji želi da obavi kupovinu na kompanijskoj veb lokaciji) ili zlonamemi SYN, koji će vezati resurse servera a da kasnije ne obavi trostruku sinhronizaciju. Kod korišćenja IP varanja, teško je locirati pravi izvor napada. Izvestan broj novijih istraživačkih radova [Savage 2000; Snoeren 2001; Adler 2002] bavio se tehnikama za označavanje IP zaglavljaka dok prolaze kroz ruter da bi se tok DoS datagrama pratio unazad do njihovog izvora. Jednom kada je kompromitovan izvorni računar identifikovan, on može da se stavi u karantin, iako je to obično spor proces koji zahteva intervenciju čoveka. Napad DDoS je Čak i teži i zahteva više vremena za rešenje.

#### 8.7.5 Pljačkanje

Prepostavite da su Alisa i Bob u konekciji daje Trudi u položaju da nadgleda pakete koji teku između Boga i Alise. Trudi može da preuzme, ili opljačka, konekciju između Alise i Boga. Posebno, Trudi može da prevari Boba da poveruje kako on i dalje komunicira sa Alisom, čak i kada komunicira sa Trudi, Trudi prvo izbacuje Alisu pokretanjem napada odbijanja usluge protiv nje. Pošto je prisluškivala Alisinu i Bobovu komunikaciju, Trudi zna celokupno stanje (na primer, broj sekvence, broj ACK, najavljeni okvir primaoca) Alisine TCP konekcije sa Bobom. Trudi na taj način može da napravi lažne IP datagrame ka Bobu (koristeći Alisinu adresu kao adresu izvora) koji sadrže važeće TCP segmente i proizvoljan korisnički teret. Zamislite katastrofu koju Trudi može da izazove u odnosu između Boga i Alise!



Slika 8.26 ♦ Distribuirani napad odbijanja usluge

Razni mrežni napadi i bezbednosne pretnje prodiskutovani su u zbirci eseja [Denning 1997] i u vrlo čitljivoj Rubinovoj knjizi [Rubin 2001 j.]. Pregled napada o kojima postoje izveštaji održava se u CERT-ovom koordinacionom centru [CERT 2004]. Pročitajte i [Cisco Security 2004; Vovdock 1983; Bhimani 1996].

## 8.8 Bezbednost u mnogo slojeva: studije primera

U prethodnim odeljcima, ispitali smo osnovna pitanja mrežne sigurnosti, uključujući kriptografiju pomoću simetričnog i javnog ključa, autentifikaciju, raspodelu ključeva, integritet poruke i digitalne potpisne. Sada ćemo ispitati kako se ti alati koriste da bi se obezbedila sigurnost na Internetu. Zanimljivo je da se mogu pružiti usluge bezbednosti na svakom od četiri gornja sloja Internetskih protokola. Kada se bezbednost osigurava za specifični protokol aplikacionog sloja, onda će aplikacija koja koristi protokol dobijati jednu ili više usluga bezbednosti, kao sto je tajnost, autentifikacija ili integritet. Kada se bezbednost osigurava za protokol transportnog sloja, onda sve aplikacije koje koriste taj protokol dobijaju usluge bezbednosti transportnog protokola. Kada se bezbednost pruža za mrežni sloj od računara do računara, onda svi segmenti transportnog sloja (i zato svi podaci aplikacionog sloja) dobijaju usluge bezbednosti mrežnog sloja. Kada se bezbednost pruža na nivou veze podataka, onda podaci u svim okvirima koji putuju preko mreže dobijaju ove bezbednosne usluge.

U ovom odeljku ispitujemo kako se bezbednosni alati koriste u aplikacionom, transportnom, mrežnom sloju i sloju veze podataka. Dosledno opštoj strukturi ove knjige, počinjemo na vrhu protokola i diskutujemo o bezbednosti u aplikacionom sloju. Naš pristup će biti da iskoristimo specifičnu aplikaciju, elektronsku poštu, kao studiju primera za bezbednost aplikacionog sloja. Zatim se spuštamо niz protokole. Ispitaćemo SSL protokol (koji pruža bezbednost na transportnom sloju), IPsec (koji pruža bezbednost na mrežnom sloju) i bezbednost IEEE 802.11 protokola za bežični LAN.

Mogli biste se zapitati zašto se bezbednosna funkcionalnost pruža na više od jednog sloja na Internetu. Zar ne bi bilo dovoljno da se funkcionalnost bezbednosti jednostavno pruži na mrežnom sloju i završi sa tim? Na ovo pitanje postoje dva odgovora. Prvo, mada bezbednost na mrežnom sloju može da ponudi „prekrivač“ šifrovanjem svih podataka u datagramima (odnosno, svih segmenata transportnog sloja) i autentifikovanjem svih IP adresa izvora, ona ne može da pruži bezbednost na nivou korisnika. Na primer, trgovinska lokacija ne može da se pouzda u bezbednost IP sloja da bi autentikovala kupca koji kupuje robu na komercijalnoj lokaciji. Dakle, postoji potreba za bezbednosnom funkcionalnošću na višim slojevima kao i za prekrivačem na nižim slojevima. Drugo, obično je lakše primeniti nove usluge Interneta, uključujući i usluge bezbednosti, na višim nivoima protokola. Dok Čekaju da se bezbednost široko primeni na mrežnom sloju, što će se verovatno dogoditi kroz mnoga godina u budućnosti, mnogi programeri aplikacija „to samo urade“ i uvedu funkcionalnost bezbednosti u svoje omiljene aplikacije. Klasični primer je Pretty Good Privacy (PGP), koji obezbeđuje sigurnu elektronsku poštu (o čemu

ćemo govoriti kasnije u ovom odeljku). Zahtevajući samo klijentov i serverov aplikacioni kod, PGP je bila jedna od prvih bezbednosnih tehnologija koje se široko koriste na Internetu.

### 8.8.1 Bezbedna elektronska pošta

U ovom odeljku koristimo mnoge od tehnika uvedenih u prethodnom odeljku, da bismo napravili projekat bezbednog sistema elektronske pošte. Taj projekat visokog nivoa stvaramo na postupan način, tako što u svakoj fazi uvodimo nove usluge bezbednosti. Kod projektovanja bezbednog sistema elektronske pošte, imajmo na umu duhovit primer uveden u odeljku 8.1 - ljubavnu aferu između Alise i Boba. Zamislite da Alisa želi da pošalje Bobu poruku elektronske pošte i da Trudi želi da bude uljez:

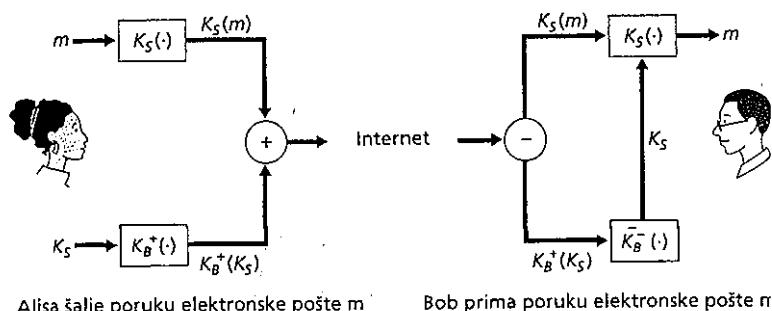
Pre nego što počnemo da projektujemo bezbedan sistem elektronske pošte za Alisu i Boba, trebalo bi prvo da razmotrimo koje bi bezbednosne karakteristike za njih bile najpoželjnije. To je, pre svega, *tajnost*. Kao što je rečeno u odeljku 8.1, ni Alisa ni Bob ne žele da Trudi čita Alisinu poruku elektronske pošte. Drugo svojstvo koje bi Alisa i Bob najverovatnije želeli da vide u bezbednom sistemu elektronske pošte je *autentifikacija pošiljaoca*. Posebno, kada Bob primi od Alise poruku „I don't love you anymore. I never want to see you again. Formerly yours, Alice“, on bi prirodno želeo da bude siguran daje poruka stigla od Alise, a ne od Trudi. Drugo svojstvo koje bi dvoje ljubavnika centri je *integritet poruke*, odnosno, osiguranje da poruka koju Alisa šalje nije promenjena u toku puta do Boba. Najzad, sistem elektronske pošte bi trebalo da obezbedi *autentifikaciju primaoca*; odnosno, Alisa želi da bude sigurna da ona stvarno šalje pismo Bobu, a ne nekom drugom (na primer, Trudi) ko se predstavlja kao Bob.

Pa, hajde da počnemo sa rešavanjem najznačajnije brige Alise i Boba, odnosno, tajnosti. Najdirektniji način da se obezbedi tajnost za Alisu je da šifruje poruku tehnologijom simetričnog ključa (kao što je DES ili AES), a za Boba da dešifruje poruku po njenom prijemu. Kao stoje prodi sku to vano u odeljku 8.2, ako je simetrični ključ dovoljno dugačak i ako samo Alisa i Bob imaju ključ, onda je izuzetno teško da bilo ko drugi (uključujući Trudi) čita poruku. Mada je ovakav pristup direkstan, on ima osnovnu teškoću o kojoj smo diskutovali u odeljku 8.2 - teško je distribuirati simetričan ključ tako da samo Alisa i Bob imaju njegove kopije. To je razlog zbog kojeg mi razmatramo alternativni pristup, odnosno, kriptografiju javnim ključem (koristeći, na primer, RSA). U pristupu javnog ključa, Bob čini svoj javni ključ svima raspoloživim (na primer, u serveru javnih ključeva ili na svojoj ličnoj veb stranici), Alisa šifruje svoju poruku Bobovim javnim ključem i šalje je na Bobovu adresu elektronske pošte. (Šifrovana poruka je enkapsulirana pomoću MIME zaglavla i poslala preko običnog SMTP-a, kao stoje objašnjeno u odeljku 2.4.) Kada Bob primi poruku, on je jednostavno dešifruje pomoću svog privatnog ključa. Pod pretpostavkom da Alisa sigurno zna da je javni ključ Bobov (i da je ključ dovoljno dugačak), ovaj pristup je odlično sredstvo da se obezbedi željena tajnost. Međutim,

problem je stope šifrovanje javnim ključem relativno neefikasno, posebno za duge poruke, (Duge poruke elektronske pošte su sada opšteprisutne na Internetu, zahvaljujući sve većoj upotrebi priloga, slike, zvuka i videa.)

Da bismo prevazišli problem efikasnosti, hajde da koristimo ključ sesije (disku-tovan u odeljku 8.5). Alisa (1) slučajno bira simetrični ključ sesije,  $K_s$ , (2) šifruje svoju poruku,  $m$ , simetričnim ključem,  $K_s$ , (3) šifruje simetrični ključ Bobovim javnim ključem,  $K_B^*$ , (4) spaja šifrovani poruku i šifrovani simetrični ključ da bi uboliočila „pakovanje“ i (5) šalje pakovanje na Bobovu adresu elektronske pošte. Postupci su ilustrovani na slici 8.27. (Na ovoj slici i na sledećim slikama, „+“ predstavlja spajanje, a „-“ razdvajanje.) Kada Bob primi pakovanje, on (1) koristi svoj privatni ključ,  $K_B$ ; da bi dobio simetrični ključ,  $K_s$ , i (2) koristi simetrični ključ  $K_s$  da bi dešifrova poruku  $m$ .

Pošto smo projektovali bezbedni sistem elektronske pošte koji obezbeđuje tajnost, hajde sada da projektujemo drugi sistem, koji obezbeđuje i autentifikaciju pošiljaoca i integritet poruka. Pretpostavimo, za trenutak, da se Alisa i Bob više ne brinu o tajnosti (oni žele da podeli svoja osećanja sa svima!), i zabrinuti su samo za autentifikaciju pošiljaoca i integritet poruka. Da bismo izvršili taj zadatok, koristimo digitalne potpise i izvode poruka, kao sto je opisano u odeljku 8.4. Alisa (1) primenjuje heš funkciju,  $H$  (na primer, MD5), na svoju poruku,  $m$ , da bi dobila izvod iz poruke, (2) potpisuje rezultat heš funkcije pomoću svog privatnog ključa,  $K_A$ ; da bi napravila digitalni potpis, (3) spaja originalnu (nešifrovani) poruku sa potpisom da bi napravila pakovanje i (4) šalje pakovanje na Bobovu adresu elektronske pošte. Kada Bob primi pakovanje, on (1) primenjuje Alisin javni ključ,  $K_A^*$ , na potpisani izvod poruke i (2) poređi rezultat te operacije sa sopstvenim hešom poruke,  $H$ . Postupci su ilustrovani na slici 8.28. Kao sto je diskutovano u odeljku 8.5, ako su dva rezultata ista, Bob može da bude prilično siguran u to da je poruka došla od Alise i daje neizmenjena.



**Slika 8.27** ♦ Alisa koristi simetrični ključ sesije,  $K_s$ , da bi poslala Bobu tajnu poruku elektronske pošte.

## KRATAK OSVRT

### FIL CIMERMAN I PGP

Filip R. Cimerman (Philip R. Zimmermann) je tvorac softvera Prelv Good Privacy (PGP). Bio je mela trogodišnje kriminalne istrage, zato što je vlada tvrdila da su izvozna ograničenja SAD prekršena kada se PGP, posle njegovog objavljinja kao softvera za probu pre korišćenje 1991. godine, raširio po celom svetu. Posle objavljinjanja PGP-a kao probnog softvera, neko drugi ga je stavio na Internet i preuzimali su ga strani građani. Programi za kriptografiju su u SAD saveznim zakonom klasifikovani kao municija i ne mogu da se izvoze.

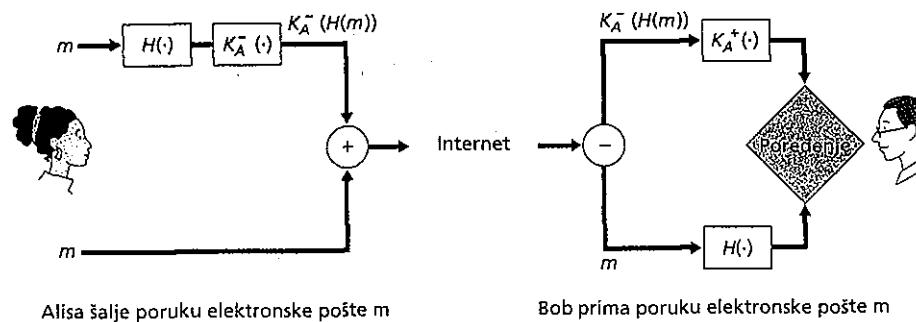
Uprkos nedostatku novca, bilo kakvog plaćenog osoblja i kompanije koja bi stala iz njega, kao i intervencijama vlade, PGP je ipak postao najsiće korišćen softver za šifrovanje elektronske pošte na svetu. Začudujuće je da je vlada SAD možda nehotice doprinela širenju PGP-a, čineći ga popularnijim upravo zbog Cimermanovog slučaja.

Vlada SAD je digla ruke od slučaja početkom 1996. godine. Vest je proslavljenia među aktivistima Interneta. Cimermanov slučaj je prerastao u priču o nevinoj ličnosti koja se bori za svoja prava protiv maltretiranja od strane moćne vlade. Vladino odustajanje bilo je dobrodošla vest, delimično zbog kampanje za cenzuru na Internetu u Kongresu i pritiska FBI-a da se dozvoli povećano piislulkivanje od strane državnih organa.

Pošto je vlada odustala od njegovog slučaja, Cimerman je osnovao firmu PGP Inc., koju je u decembru 1997. godine kupila kompanija Network Associates. Cimerman je sada nezavisni sovetnik u oblasti kriptografije.

Hajde sada da razmotrimo projektovanje sistema elektronske pošte koji obezbeđuje tajnost, autentifikaciju pošiljaoca i integritet poruke. To može da se uradi kom-binovanjem postupaka na slikama 8.27 i 8.28. Alisa prvo pravi prethodno pakovanje, tačno kao na slici 8.28, koje se sastoji od njenе originalne poruke uz digitalno potpisani izvod poruke. Ona zatim obrađuje to prethodno pakovanje kao novu poruku i obrađuje je kroz postupke pošiljaoca na slici 8.27, praveći novo pakovanje koje se šalje Bobu. Postupci koje Alisa preduzima prikazani su na slici 8.29. Kada Bob primi pakovanje, on prvo primeni svoju stranu slike 8.27, a zatim svoju stranu slike 8.28. Jasno je da ova konstrukcija obezbeđuje tajnost, autentifikaciju pošiljaoca i integritet poruke. Zapazite da, u ovoj šemi, Alisa koristi kriptografiju javnim ključem dva puta: jednom sa sopstvenim privatnim ključem i jednom sa Bobovim javnim ključem! Slično tome, Bob takođe koristi kriptografiju javnim ključem dva puta - jednom sa svojim privatnim ključem i jednom sa Atisnim javnim ključem.

Projekat bezbedne elektronske pošte skiciran na slici 8.29 verovatno obezbeđuje zadovoljavajuću sigurnost većini korisnika elektronske pošte i to u većini slučajeva.



**Slika 8.28** ♦ Upotreba heš funkcija i digitalnih potpisa da bi se obezbedili autentifikacija pošiljaoca i integritet poruke.

Ali, postoji ipak još jedno važno pitanje koje treba da se resi. Konstrukcija na slici 8.29 zahteva da Alisa dobije Bobov javni ključ, a da Bob dobije Alisin javni ključ. Distribucija tih javnih ključeva nije trivijalan problem. Na primer, Trudi bi mogla da se maskira u Boba i da da Alisi svoj javni ključ, tvrdeći da je to Bobov javni ključ, što bi joj omogućilo da primi poruku namenjenu Bobu. Kao što smo naučili u odeljku 8.5, popularan način distribucije javnih ključeva je da se javni ključevi *overavaju* koristeći usluge sertifikacionog tela.

#### PGP

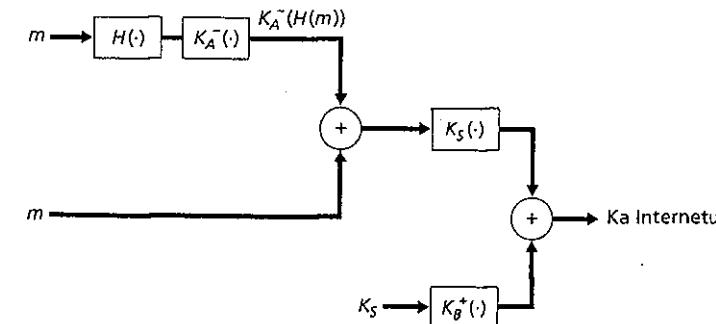
Pretty Good Privacy (PGP), kojeg je prvobitno napisao Phil Zimmermann 1991. godine, postao je *de facto* standardna šema za šifrovanje elektronske pošte. Njegova veb lokacija pruža više od milion stranica mesečno korisnicima u 166 različitim zemaljama [PGPI 2002]. Verzije PGP-a raspoložive su u javnom domenu; na primer, možete da pronadete softver PGP za vašu omiljenu platformu kao i mnogo zanimljivih tekstova za čitanje na lokaciji International PGP Home Page [PGPI 2002]. (Posebno zanimljiv je esej autora PGP-a [Zimmermann 1999].) PGP se takođe može komercijalno nabaviti, a na raspolaganju je i kao dodatak za mnoge agente korisnika elektronske pošte, uključujući Microsoftove Exchange i Outlook, kao i Eudoru firmе Qualcomm. Konstrukcija PGP-a je u suštini ista kao ona koja je prikazana na slici 8.29. Zavisno od verzije, softver PGP koristi MD5 ili SHA za izračunavanje izvoda poruke; CAST, trostruki DES, ili IDEA za šifrovanje simetričnog ključa; i RSA za šifrovanje javnog ključa. Pored toga, PGP obezbeđuje i kompresiju podataka.

Kada se PGP instalira, softver stvara par javnih ključeva za korisnika. Javni ključ može da se postavi u serveru javnih ključeva na korisnikovoj veb lokaciji. Privatni ključ se štiti upotrebom lozinke. Lozinka treba da se unese svaki put kada kori-

snik pristupa privatnom ključu. PGP pruža korisniku opcije digitalnog potpisivanja poruke, šifrovanja poruke, ili i digitalnog potpisivanja i šifrovanja. Na slici 8.30 prikazana je potpisana PGP poruka. Ta poruka se pojavljuje posle zaglavila MIME (Multipurpose Internet Mail Extensions, višenamenska proširenja Internet pošte). Šifrovani podaci u poruci su  $K/(H(m))$ , odnosno, digitalno potpisani izvod poruke. Kao što smo ranije diskutovali, da bi Bob proverio integritet poruke, on treba da ima pristup Alisinom javnom ključu,

Na slici 8.31 prikazana je tajna PGP poruka. Ta poruka se takođe pojavljuje posle zaglavila MIME. Naravno, poruka otvorenog teksta nije uključena unutar tajne poruke elektronske pošte. Kada pošiljalac (kao Alisa) želi i tajnost i integritet, PGP sadrži poruku kao što je ona na slici 8.31 unutar poruke sa slike 8.30.

PGP takođe obezbeđuje mehanizam za overu javnih ključeva, ali je taj mehanizam sasvim drugačiji od konvencionalnijih metoda sertifikacionog tela. Javni ključevi PGP overavaju se pomoću Weba od poverenja. Alisa može sama da overi svaki par ključ/korisničko ime kada veruje da članovi para zaista pripadaju jedan drugom. Pored toga, PGP dozvoljava Alisi da kaže da ona veruje drugom korisniku i da jamči za autentičnost više ključeva. Neki korisnici PGP-a potpisuju jedni drugima ključeve. Oni se fizički okupljaju, razmenjuju diskete koje sadrže javne ključeve i overavaju jedni drugima ključeve potpisujući ih pomoću svojih privatnih ključeva. Javni ključevi PGP se takođe distribuiraju na PGP serverima javnih ključeva na Internetu. Kada korisnik podnese javni ključ takvom serveru, on memorije kopiju ključa, šalje kopiju ključa svim drugim serverima javnih ključeva i stavlja na raspolaganje ključ svakome ko ga traži. Mada strane za potpisivanje ključeva i serveri PGP javnih ključeva stvarno postoje, daleko uobičajeniji način da korisnici distribuiraju svoje javne ključeve je da ih izlažu na svojim ličnim veb stranicama i oglašavaju ih u svojim porukama elektronske pošte.



**Slika 8.29** ♦ Alisa koristi kriptografiju simetričnog ključa, heš funkciju i digitalni potpis da bi obezbedila tajnost, autentifikaciju pošiljaoca i integritet poruke.

---- BEGIN PGP SIGNED MESSAGE ----

**Hash:**

**SHA1 Bob:**

**Can I see you tonight?**

**Passionately yours, Alice**

---- BEGIN PGP SIGNATURE ----

**Version:** PGP for Personal Privacy 5.0

**Charset:** noconv

**yhHJRhhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2**

---- END PGP SIGNATURE ----

Slika 8.30 ♦ Potpisana PGP poruka

----BEGIN PGP MESSAGE ---

**Version:** PGP for Personal Privacy 5.0

**u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX681iKmSF6Gc4sDfcXyt**

**Rfds10juHgbcfDssWe7/K=1KhnmikLoO + 1/BvcX^t==Ujk9PbcD'5**

**Thdf2awQfgHbnmKlok8iy6gThlp**

----END PGP MESSAGE

Slika 8.31 ♦ Tajna PGP poruka

## 8.8.2 Sloj bezbednih soketa (SSL) i bezbednost transportnog sloja (TLS)

U prethodnom odeljku, razmatrali smo kako aplikacioni sloj može da koristi (u bezbednosti elektronske pošte) različite bezbednosne mehanizme koje smo poučili ranije u ovom poglavlju: šifrovanje, autentifikaciju, distribuciju ključeva, integritet poruka i digitalne potpise. U ovom odeljku, nastavicećemo naše studije primera različitih bezbednosnih mehanizama spuštajući se sloj niže u protokolima i objašnjavajući bezbedne sokete i bezbednost transportnog sloja. Kao primer poslužiće nam aplikacija Internet trgovine, zato što su poslovne i finansijske transakcije značajna pokretačka sila za bezbednost na Internetu.

Hajde da prošetamo kroz uobičajeni scenario Internet trgovine. Bob krstari Webom i stiže na lokaciju Alice Incorporated, gde se prodaje neka trajna roba. Na lokaciji Alice Incorporated prikazuje se formular u koji se od Boba očekuje da unese željenu količinu, svoju adresu i broj svoje kartice za plaćanje, Bob unosi te informacije, mišem bira „submit“ i zatim očekuje da primi robu (recimo, preko obične pošte); on takođe očekuje da primi račun za robu u svom sledećem izveštaju platne kartice. To sve lepo zvuči, ali ako nisu preduzete nikakve mere bezbednosti - kao

Što su šifrovanje ili autentifikacija - Bob bi mogao da doživi nekoliko iznenadenja.

- ♦ Uljez bi mogao da presrete porudžbinu i dobije informacije o Bobovoj platnoj kartici. U tom slučaju uljez bi mogao da kupuje na Bobov račun.

- ♦ Lokacija bi mogla da prikazuje poznati logo Alice Incorporated, ali da u stvari bude lokacija koju održava Trudi, koja se maskira u Alice Incorporated. Trudi bi mogla da uzme Bobov novac i pobegne. Ili, Trudi bi mogla da pravi sopstvene kupovine i opterećuje tim troškovima Bobov račun.

Moguća su i mnoga druga iznenadenja, a mi ćemo o nekim od njih raspravljati u sledećem pododeljku. Ali dva problema koja smo gore istakli su najozbiljniji. Trgovina na Internetu koja koristi protokol SSL rešava oba ova problema.

Sloj bezbednih soketa (*secure sockets layer*, SSL), koji je prvo bitno razvio Netscape, jeste protokol projektovan da obezbedi šifrovanje podataka i autentifikaciju između veb klijenta i veb servera. Protokol počinje fazom provere spremnosti za prenos kojom se dogovaraju algoritam šifrovanja (na primer, DES ili IDEA) i ključevi, i klijentu autentificuje server. Postoji i druga mogućnost, a to je da se klijent autentificuje serveru. Kada se klijent i server sinhronizuju i počne prenos podataka aplikacije, svi podaci se šifruju koristeći ključeve sesije dogovorene za vreme faze sinhronizacije. SSL se široko koristi u trgovini na Internetu, budući daje implementiran u skoro svim popularnim pretraživačima i veb serverima. Pored toga, on je i osnova protokola Bezbednost transportnog sloja (*Transport Layer Security*, TLS) [RFC 2246].

SSL i TLS nisu ograničeni na primenu na VVebu; na primer, oni na sličan način mogu da se upotrebe za autentifikaciju i šifrovanje podataka za IMAP (Internet Mail Access Protocol) pristup pošti. SSL može da se posmatra kao sloj koji se nalazi između aplikacionog i transportnog sloja. Na predajnoj strani, SSL prima podatke (kao što su HTTP ili IMAP poruka iz aplikacije), šifruje podatke i usmerava šifro-vane podatke na TCP soket. Na prijemnoj strani, SSL čita iz TCP soketa, dešifruje podatke i usmerava ih u aplikaciju. Iako SSL može da se upotrebi sa mnogim Internet aplikacijama, mi ćemo raspravljati o njegovim mogućnostima u kontekstu Weba, gde se danas koristi uglavnom za Internet trgovinu.

SSL obezbeđuje sledeća svojstva:

- ♦ *SSL autentifikaciju servera*, koja korisniku dozvoljava da potvrdi identitet servera. Pretraživač sa omogućenim SSL-om održava listu serifikacionog tela (CA), zajedno sa javnim ključevima CA-ova. Kada pretraživač želi da posluje sa veb serverom na kome je uključen SSL, on od servera dobija uverenje koje sadrži serverov javni ključ. Uverenje izdaje (odnosno, digitalno potpisuje) CA koji se nalazi u klijentovoj listi CA-ova od poverenja. To svojstvo dozvoljava pretraživaču da autentificuje server pre nego što korisnik podnese broj kartice za plaćanje. U kontekstu prethodnog primera, ta autentifikacija servera omogućava Bobu da proveri da li zaista šalje broj svoje kartice za plaćanje lokaciji Alice Incorporated, a ne nekom drugom ko bi mogao da se maskira kao Alice Incorporated.
- ♦ *SSL autentifikaciju klijenta*, koja serveru dozvoljava da potvrdi identitet korisnika. Analogno autentifikaciji servera, autentifikacija klijenta koristi klijentova uverenja, koja su takođe izdali CA-ovi. Ta autentifikacija je značajna ako je ser-

ver, na primer, banka koja korisniku šalje poverljivu finansijsku informaciju i želi da proveri identitet primaoca. Iako podržana od SSL-a, autentifikacija klijenta je opcionala. Da bismo usredsredili našu diskusiju, mi ćemo je u daljem izlaganju zanemarivati.

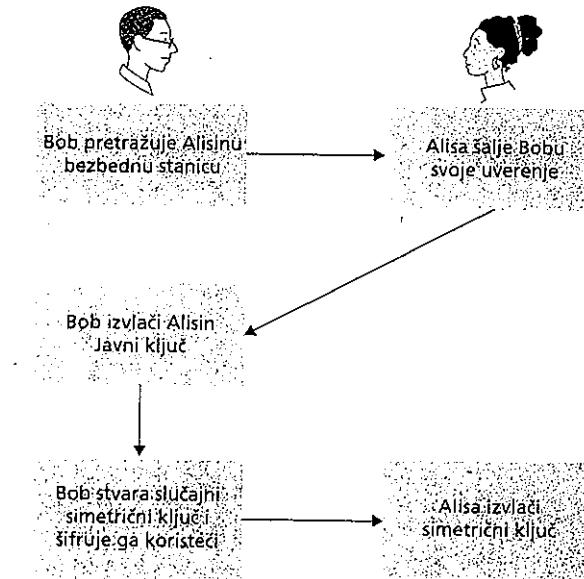
- ♦ **Šifrovani SSL sesiju**, u kojoj su sve informacije poslate između pretraživača i servera šifrovane pomoću predajnog softvera (pretraživača ili veb servera) i dešifrovane pomoću prijemnog softvera (pretraživača ili veb servera). Ova tajnost može da bude važna i kupcu i prodavcu. Takođe, SSL obezbeđuje mehanizam za otkrivanje falsifikovanja informacija od strane nekog uljeza.

### Kako radi SSL

Korisnik, recimo Bob, krstari Webom i bira link koji ga dovodi do bezbedne stranice na kojoj je smešten Alisin server osposobljen za SSL. Protokolski deo URL-a za tu stranicu je „https”, a ne uobičajeno „http”. Pretraživač i server tada izvršavaju SSL protokol za proveru spremnosti za prenos, koji (1) autentificira server i (2) stvara deljeni simetrični ključ. U oba ova zadatka koristi se RSA tehnologija javnih ključeva. Glavni tok događaja u fazi provere spremnosti za prenos prikazan je na slici 8.32. Za vreme te faze, Alisa šalje Bobu svoje uverenje, iz koga Bob dobija njen javni ključ. Tada Bob stvara slučajni simetrični ključ, šifruje ga pomoću Alisinog javnog ključa i šalje šifrovani ključ Alisi. Bob i Alisa sada dele simetrični ključ sesije. Kada se sinhronizacija sa serverom završi, svi podaci koji se šalju između pretraživača i servera (preko TCP konekcija) šifrovani su pomoću simetričnog ključa sesije.

Pošto smo dali globalni pregled SSL-a, hajde da bliže pogledamo neke od najvažnijih detalja. U SSL sinhronizaciji preduzimaju se sledeći postupci:

1. Pretraživač šalje serveru broj svoje SSL verzije i spisak najpoželjnijih kripto-grafskih metoda koje predlaže za korišćenje. On to radi da bi se sa serverom dogovorio koji algoritam simetričnog ključa i koja heš funkcija će se koristiti.
2. Server šalje pretraživaču broj svoje SSL verzije, svoju odluku o kriptografskoj metodi i svoje uverenje. Setite se da uverenje obuhvata serverov RSA javni ključ i daje potvrđeno od nekog CA-a; odnosno, uverenje je potpisano CA-ovim privatnim ključem.
3. Pretraživač ima listu CA-ova od poverenja i javni ključ za svaki CA-sa liste. Kada pretraživač primi uverenje od servera, on proverava da li je CA na listi. Ako nije, korisnik dobija upozorenje i obaveštenje da ne može da se automatski uspostavi šifrovana i autentifikovana konekcija. Ako je CA na listi, pretraživač koristi CA-ov javni ključ da bi proverio uverenje i dobio serverov javni ključ.



**Slika 8.32** ♦ Globalni prikaz faze SSL sinhronizacije

4. Pretraživač stvara simetrični ključ sesije, šifruje ga pomoću serverovog javnog ključa, i šalje serveru šifrovani ključ sesije.
5. Pretraživač šalje serveru poruku obaveštavajući ga da će buduće poruke od klijenta biti šifrovane pomoću ključa sesije. On zatim šalje posebnu (šifrovani) poruku, pokazujući da je pretraživačev deo sinhronizacije završen.
6. Server šalje pretraživaču poruku obaveštavajući ga da će buduće poruke od servera biti šifrovane pomoću ključa sesije. On zatim šalje posebnu (šifrovani) poruku, pokazujući da je serverov deo sinhronizacije završen.
7. Sada je SSL sinhronizacija završena, a SSL sesija je počela. Pretraživač i server koriste ključeve sesije da bi šifrovali i dešifrovali podatke koje šalju jedan drugom i da bi proveravali njihov integritet.

SSL sinhronizacija u stvari ima mnogo više postupaka od onih koji su gore navedeni. Više detalja o SSL-u možete da pronađete na [Netscape SSL 1998]. Pored kupovine putem kartica za plaćanje, ovde ističemo da SSL može da se koristi (i koristi se) za druge finansijske transakcije, uključujući onlajn bankarstvo i robnu trgovinu.

### Ograničenja SSL-a u Internet trgovini

Zbog svoje jednostavnosti i ranog razvoja, SSL je široko implementiran u pretraživačima, serverima i komercijalnim proizvodima za Internet. Ti serveri i pretraživači osposobljeni za SSL obezbeđuju popularnu platformu za transakcije pomoću platnih kartica. I pored toga, trebalo bi imati na umu da SSL nije posebno pravljen za transakcije sa platnim karticama, nego za svojevrsnu bezbednu komunikaciju između klijenta i servera. Zbog toga SSL-u nedostaju mnoga svojstva koja bi industrija platnih kartica volela da ima u protokolu za trgovinu na Internetu.

Razmotrite još jednom što se dešava kada Bob kupuje od firme Alice Incorporated preko SSL-a. Potpisano uverenje koje Bob prima od Alise ubeduje Boba da on stvarno ima posla sa firmom Alice Incorporated i da je to dobronamema kompanija. Međutim, generičko uverenje ne pokazuje da lije Alice Incorporated ovlašćena da prihvata kupovine putem platnih kartica, niti da li je kompanija pouzdan trgovac. To otvara vrata trgovinskoj prevari. Sličan problem postoji i za ovlašćenje klijenta. Čak ako je iskorišćena SSL autentifikacija klijenta, uverenje klijenta ne povezuje Boba sa određenom ovlašćenom platnom karticom; dakle, Alice Incorporated nema nikakvo osiguranje daje Bob ovlašćen da kupuje pomoću platne kartice. To otvara vrata svim vrstama prevara, uključujući kupovinu pomoću ukradene platne kartice i kupčeve nepriznavanje kupljene robe.

### 8.8.3 Bezbednost mrežnog sloja: IPsec

Protokol IP security, poznatiji kao IPsec, jeste niz protokola koji pružaju bezbednost na mrežnom sloju. IPsec je vrlo složen, a njegovi različiti delovi opisani su u više od dvanaestak RFC-ova. U ovom odeljku, govorimo o IPsec-u u specifičnom kontekstu, odnosno, u kontekstu gde svi računari na Internetu podržavaju IPsec. Iako će do toga proći još mnogo godina, to će za sada pojednostaviti priču i pomoći nam da razumemo glavna svojstva IPsec-a. Dva ključna RFC-a su RFC 2401, u kome se opisuje ukupna arhitektura IP bezbednosti i RFC 2411, u kome se daje pregled niza protokola IPsec i dokumenata koji ga opisuju.

Pre nego što uđemo u specifičnosti IPsec-a, hajde da se vratimo unazad i razmotrimo što to znači pružiti bezbednost na mrežnom sloju. Pogledajmo, pre toga, što znači obezbediti tajnost mrežnog sloja. Mrežni sloj bi obezbedio tajnost ako bi svi podaci koje nose svi IP datagrami bili šifrovani. To znači da kad god računar želi da pošalje datagram, on šifruje polje za podatke pre nego što ga isporuči mreži. U principu, šifrovanje bi moglo da se izvede pomoću simetričnog ključa, javnog ključa ili pomoću ključeva sesije koji su dogovorenim korišćenjem šifrovanja javnim ključem. Polje za podatke moglo bi da bude TCP segment, UDP segment, poruka ICMP itd. Ako bi takva usluga mrežnog sloja bila prisutna, svi podaci koje šalju računari - uključujući tu elektronsku poštu, veb stranice, kontrolne poruke i poruke upravljanja (kao što su ICMP i SNMP) - bili bi sakriveni za bilo koju treću stranu koja „prisluškuje“ mrežu. Dakle, takva usluga bi obezbedila pokrivač za celokupan saobraćaj na Internetu, pružajući nam na taj način izvestan osećaj bezbednosti.

Pored tajnosti, moglo bi se takođe poželeti da mrežni sloj obezbedi i autentifikaciju izvora. Kada računar odredišta primi IP datagram sa određenom IP adresom izvora, on autentificuje izvor uveravajući se da je IP datagram zaista stvoren u računaru sa tom IP adresom. Takva usluga sprečava da se autentikuju datagrami sa lažnim IP adresama.

U nizu protokola IPsec postoje dva glavna protokola: protokol za autentifikaciju zaglavljaja (*Authentication Header, AH*) i protokol za enkapsulaciju bezbednog tereta (*Encapsulation Security Payload Protocol, ESP*). Kada izvorni računar šalje bezbedne datagrame odredišnom računaru, on to radi pomoću AH protokola ili pomoću ESP protokola. Protokol AH obezbeđuje autentifikaciju izvora i integritet podataka, ali ne obezbeđuje tajnost. Protokol ESP obezbeđuje autentifikaciju, integritet podataka\*! tajnost. Objedinjujući više usluga, protokol ESP je prirodno složeniji i zahteva više obrade od AH protokola.

I u AH i u ESP protokolu, pre slanja bezbednih datagrama od izvornog do odredišnog računara, izvor i mrežni računari vrše sinbranizaciju i stvaraju logičku konekciju mrežnog sloja. Taj logički kanal zove se asocijacija bezbednosti (*Security Association, SA*). Dakle, IPsec transformiše uobičajeni mrežni sloj Interneta bez konekcija u sloj sa logičkim konekcijama! Logička konekcija koju definiše SA je simpleksna konekcija; to znači, da je ona jednosmerna. Ako oba računara žele da šalju jedan drugom bezbedne datagrame, onda je potrebno da se uspostave dve SA-e (odnosno, logičke konekcije), po jedna u svakom smeni. SA je jedinstveno identificovana trojkom koja se sastoji od:

- ◆ identifikatora bezbednosnog protokola (AH ili ESP);
- ◆ IP adrese izvora za simpleksnu konekciju;
- ◆ 32-bitnog identifikatora konekcije koji se zove Security Parameter Index (SPI).

Za dan SA (odnosno, dan logičku konekciju od izvornog do odredišnog računara), svaki IPsec datagram će imati specijalno polje za SPI. Svi datagrami u konekciji SA koristiće istu vrednost SPI u tom polju.

#### Protokol za autentifikaciju zaglavljaja (AH)

Kao što je gore napomenuto, protokol AH obezbeđuje autentifikaciju izvornog računara i integritet podataka, ali ne i njihovu tajnost. Kada izvorni računar želi da pošalje jedan datagram ili više datagrama na neko odredište, on prvo uspostavlja SA sa odredištem. Pošto je uspostavio SA, izvor može da šalje bezbedne datagrame odredišnom računaru. Bezbedni datagrami uključuju AH zaglavljje, koje je umetnuto između originalnih podataka IP datagrama (na primer, TCP ili UDP segment) i IP zaglavljja, kao sto je prikazano na slici 8.33. Na taj način AH zaglavljje povećava polje originalnih podataka i to povećano polje za podatke se enkapsulira kao standardni IP datagram. Za polje protokola u IP zaglavljiju, upotrebljena je vrednost 51 da bi se pokazalo da datagram uključuje AH zaglavljje. Kada odredišni računar primi

IP datagram, on zapisuje 51 u polju protokola i obraduje datagram koristeći AH protokol. (Setite se da se polje za protokol u IP datagramu koristi za određivanje protokola višeg sloja - na primer, UDP, TCP ili ICMP - kome će se proslediti deo IP datagrama sa podacima.) Ruteri na putu između izvora i odredišta obrađuju datagrame kao i uvek - oni ispituju IP adresu odredišta i u skladu sa njom prosledjuju datagrame.

Zaglavje AH obuhvata više polja, odnosno:

- ◆ Polje *Next Header*, koje ima ulogu polja za protokol u običnom datagramu. Ono pokazuje da li je podatak koji sledi iza AH zaglavija TCP segment, UDP segment, ICMP segment itd. (Setite se da se polje za protokol u IP datagramu sada koristi da bi ukazalo na AH protokol, pa ne može više da se upotrebi da bi ukazalo na protokol transportnog sloja.)
- ◆ Polje *Security Parameter Index (SPI)*, proizvoljna 32-bitna vrednost koja, u kombinaciji sa IP adresom odredišta i bezbednosnim protokolom, jedinstveno identificuje SA za datagram.
- ◆ Polje *Sequence Number*, 32-bitno polje koje sadrži broj sekvence za svaki datagram. Ono je u početku podešeno na 0 prilikom uspostavljanja konekcije SA. Protokol AH koristi brojeve sekvenci da bi sprečio napade reprodukcijom i Čoveka u sredini (procitajte odeljak 8.3).
- ◆ Polje *Auhtentication Data*, polje promenljive dužine koje sadrži potpis izvod iz poruke (odnosno, digitalni potpis) za taj datagram. Izvod iz poruke izračunava se preko originalnog IP datagrama, obezbeđujući tako autentifikaciju izvornog računara i integritet IP datagrama. Digitalni potpis se izračunava korišćenjem algoritma koji određuje SA, kao sto je MD5 ili SHA.

Kada odredišni računar primi IP datagram sa AH zaglavljem, on određuje SA za datagram i zatim autentikuje integritet datagrama obradujući polje sa podacima autentifikacije. IPsec za autentifikaciju (i za AH i za ESP protokol) koristi šemu koja se zove HMAC, koja je šifrovan izvod poruke opisan u knjizi [RFC 2104], HMAC za autentifikaciju poruke koristi deljeni tajni ključ između dve strane, a ne metode javnog ključa. Više detalja o AH protokolu može se pronaći u knjizi [RFC 2402].



Slika 8.33 ♦ Položaj AH zaglavija u IP datagramu

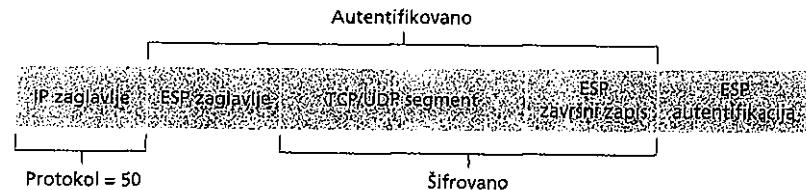
### Protokol ESP

Protokol ESP obezbeđuje tajnost mrežnog sloja, kao i autentifikaciju izvornog računara i integritet podataka. Još jednom, sve počinje sa izvornim računaram koji uspostavlja SA sa odredišnim računaram. Tada izvorni računar može da pošalje bezbedne datagrame odredišnom računaru. Kao sto je prikazano na slici 8.34, bezbedni datagram se pravi uokvirivanjem podataka originalnog IP datagrama pomoću zaglavja i završnog zapisa, a zatim umetanjem tih enkapsuliranih podataka u polje za podatke IP datagrama. Vrednost 50 se uzima za polje protokola u zaglavju IP datagrama, da bi pokazala da datagram uključuje ESP zaglavlje i završni zapis. Kada odredišni računar primi IP datagram, on beleži 50 u polju za protokol i obraduje datagram koristeći protokol ESP. Kao što je prikazano na slici 8.34, podaci originalnog IP datagrama sa ESP poljem završnog zapisa su šifrovani. Tajnost se obezbeđuje pomoću šifrovanja DES-CBC [RFC 2405]. Zaglavje ESP sastoji se od 32-bitnog polja za SPI i 32-bitnog polja za broj sekvence, koji imaju istu ulogu kao u protokolu AH. Završni zapis uključuje polje *Next Header*, koje takođe ima istu ulogu kao u AH. Zapazite da zato što je polje *Next Header* šifrovano zajedno sa originalnim podacima, uljez neće moći da odredi upotrebljeni transportni protokol. Iza završnog zapisa postoji polje *Authentication Data*, koje takođe ima istu ulogu kao u protokolu AH. Više detalja o ESP protokolu može se pronaći u knjizi [RFC 2406].

### SA i upravljanje ključevima

Za uspešnu primenu IPseca, potrebna je skalabilna i automatizovana SA i šema za upravljanje ključevima. Za ove zadatke definisano je više protokola.

- ◆ Algoritam za razmenu ključeva (*Internet Key Exchange, IKE*) [RFC 2409] je podrazumevan alat za upravljanje ključevima za IPsec.
- ◆ Protokol za bezbednosnu asocijaciju i upravljanje ključevima (*Internet Security Association and Key Management Protocol, ISKMP*) definiše procedure za uspostavljanje i ukidanje konekcije SA [RFC 2407; RFC 2408]. ISKMP-ova bezbednosna asocijacija je sasvim razdvojena od IKE razmene ključeva.



Slika 8.34 ♦ Polja ESP u IP datagramu

Ovim se završava naša diskusija o IPsecu. Govorili smo o IPsecu u kontekstu IPv4 i „transportnog režima“. IPsec takođe definiše „mnelski režim“, u kome funkcionalnost bezbednosti uvode ruteri, a ne računari. Najzad, IPsec opisuje procedure šifrovanja za IPv6 kao i za IPv4.

#### 8.8.4 Bezbednost u IEEE 802.11

Bezbednost je posebno značajna u bežičnim mrežama, gde radio-talasi koji nose okvire mogu da se prostiru daleko van zgrada u kojima se nalaze osnovna bežična stanica (e) i računari. Dobar primer je 18-mesečni „ratni“ napor koji je preduzeo Piter Šipli, vožeći se unaokolo po zalivu San Franciska sa prenosnim računarom i karticom za 802.11, tražeći bežične mreže koje su bile „vidljive“ van zgrade, o čemu možete čitati u knjizi [Shipley 2001]. On je registrovao više od 9000 takvih mreža; jedan ugao u San Francisku ponudio mu je šest različitih raspoloživih mreža! Stope možda još vise za uzbunu, Šipli je zapazio da 85 procenata od tih 9000 mreža čak i ne koristi (kao što ćemo videti) dosta slabe mehanizme u originalnom standardu 802.11.

Pitanje bezbednosti u 802.11 privuklo je veliku pažnju, kako u tehničkim krugovima, tako i u medijima. Mada je diskusija bila široka, bilo je malo debate u pravom smislu reči - izgleda da postoji opšta saglasnost da originalna specifikacija 802.11 sadrži brojne ozbiljne nedostatke. Zaista, sada može da se preuzme softver u javnom domenu koji eksploatiše te pukotine, čineći korisnike običnih sigurnosnih mehanizama 802.11 toliko otvorenim za bezbednosne napade koliko i onih 85 % korisnika bežičnih mreža u oblasti zaliva San Franciska koji uopšte ne koriste bezbednosna svojstva.

U daljem tekstu, govorićemo o bezbednosnim mehanizmima koji su u početku standardizovani u specifikaciji 802.11, grupno poznatim kao Wired Equivalent Privacy (WEP). Kao što mu i samo ime govorи, WEP je namenjen da ostvari nivo bezbednosti u mrežama 802.11 sbičan onom koji se može naći u ožičenim mrežama. Zatim ćemo nešto reći o pukotinama u WEP-u i o standardu 802.11 i, fundamentalno bezbednjom verzijom 802.11 koja je usvojena 2004. godine.

##### Wired Equivalent Privacy (WEP)

Protokol IEEE 802.11 WEP [IEEE 802.11 1999] obezbeđuje autentifikaciju i šifrovanje podataka između računara i bežične pristupne tačke (odnosno, osnovne stanice), koristeći pristup simetričnog deljenog ključu. WEP ne određuje algoritam za upravljanje ključevima, pa se podrazumeva da su se računar i bežična pristupna tačka nekako sporazumeli o ključu preko metoda van opsega. Autentifikacija je izvršena kao u protokolu \$p4 . £7 koji smo razvili u odeljku 8.3. Time su obuhvaćena Četiri postupka:

1. Bežični računar prvo zahteva autentifikaciju od strane pristupne tačke.
2. Pristupna tačka odgovara na zahtev za autentifikaciju 128-bajtnom jednokratnom vrednošću.
3. Bežični računar šifruje jednokratnu vrednost koristeći simetrični ključ koji deli sa pristupnom tačkom.
4. Pristupna tačka dešifruje jednokratnu vrednost.

Ako dešifrovana jednokratna vrednost odgovara onoj koja je prvobitno poslata računari, onda je pristupna tačka autentifikovala računar.

Algoritam WEP-za šifrovanje podataka ilustrovan je na slici 8.35. Tajni 40-bitni simetrični ključ,  $K_s$ , po pretpostavci poznaju i računar i pristupna tačka. Pored toga, 24-bitni inicijalizacioni vektor (IV) dodaje se na 40-bitni ključ da bi se stvorio 64-bitni ključ koji će se koristiti za šifrovanje jednog okvira. IV će se menjati od jednog do drugog okvira, pa će zato svaki okvir biti šifrovan različitim 64-bitnim ključem. Šifrovanje se izvršava na sledeći način. Prvo se izračunava četvorobajtna CRC vrednost (pročitajte odeljak 5.2) za polje podataka. Ovo polje i četiri CRC bajta se zatim šifruju koristeći šifru RC4. Ovde nećemo objašnjavati detalje u vezi sa šifrom RC4 (njih možete naći u knjizi [Schneier 1995]). Za našu namenu, dovoljno je znati da, kada mu se predstavi vrednost ključa (u ovom slučaju, 64-bitnog ključa ( $K^IV$ )), algoritam RC4 proizvodi niz vrednosti ključeva,  $k_1, k_2, k_3, \dots$ , koji se koriste za šifrovanje podataka i CRC vrednosti unutar postojećeg okvira. Iz praktičnih razloga, možemo da zamislimo da se te operacije izvršavaju odjednom nad jednim bajtom. Šifrovanje se izvodi izvršavanjem operacije ekskluzivno ILI (-toga podataka,  $d_i$ ) sa i-tim ključem,  $f_t^{k_i}$ , u nizu vrednosti ključeva koje je stvorio par ( $K^IV$ ) da bi proizveo i-ti bajt šifrovanog teksta,  $c_i$ :

$$c_i = d_i \oplus f_t^{k_i}$$

Vrednost IV menja se od jednog okvira do sledećeg i uključena je u otvorenom tekstu u zaglavlju svakog WEP-šifrovanog okvira 802.11, kao što je prikazano na slici 8.35. Primalac uzima tajni 40-bitni simetrični ključ koji deli sa pošiljaocem, dodaje IV i koristi rezultujući 64-bitni ključ (koji je identičan ključu koji je pošiljalac upotrebo za šifrovanje) da bi dešifroval okvir.

$$d_i = c_i \oplus f_t^{k_i}$$

Pravilna upotreba algoritma RC4 zahteva da se ista 64-bitna vrednost ključa nikada ne koristi više od jednog puta. Setite se da se WEP ključ menja od okvira do okvira. Za dati  $K_s$  (koji se retko menja, ako uopšte čini), ovo znači da postoji samo  $2^{2a}$  jedinstvenih ključeva. Ako se ti ključevi retko biraju, može se pokazati [Walker 2000] da je verovatnoća da se izabere ista vrednost IV (i zato koristi isti 64-bitni ključ) veća od 99 procenata posle samo 12000 okvira. Sa okvirima veličine

1 K i brzinom prenosa podataka od 11 Mb/s, potrebno je samo nekoliko sekundi da se prenese 12000 okvira. Štaviše, kako se IV u okviru prenosi u otvorenom tekstu, prislушкиvač će znati kada se koristi duplikat vrednosti IV.

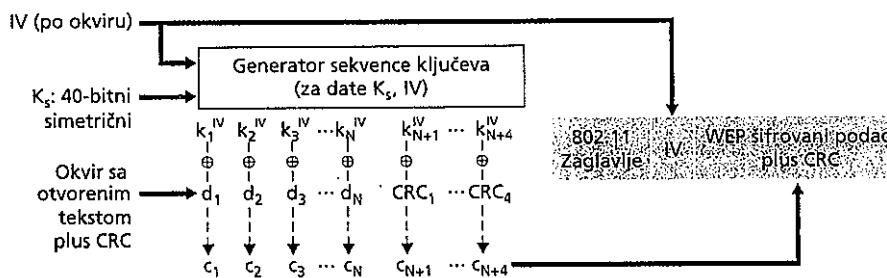
Da biste shvatili jedan od više problema koji se pojavljuju kada se koristi duplikat ključa, razmotrite sledeći napad izabranim otvorenim tekstom koji Trudi preduzima protiv Alise. Pretpostavite da Trudi (možda koristeći IP varanje) šalje zahtev (na primer, HTTP ili FTP) ka Alisi da prenese datoteku sa poznatim sadržajem,  $d$ ,

$d_2 > d_3, d_4, \dots$  Trudi takođe posmatra šifrovane podatke  $c_1, c_2, c_3, c_4, \dots$  Kako je  $d_i = c_i \text{ XOR } k^i$ , ako izvršimo ekskluzivno ILI  $c_i$  sa svake strane ove jednačine, imamo:

$$4 \text{ XOR } c_i = k^i$$

Pomoću ove relacije, Trudi može da upotrebi poznate vrednosti  $d_i$  i  $c_i$  da bi izračunala  $k^i$ . Sledеći put kada Trudi vidi da se koristi ista vrednost IV, ona će znati sekvencu ključeva  $k^1, k^2, k^3, \dots$  i na taj način će moći da dešifruje šifrovani poruku.

Postoji i više dodatnih briga sa WEP-om. U knjizi [Fluhler 2001], opisan je napad koji koristi poznatu slabost RC4 kada se odaberu izvesni slabi ključevi. U radu [Stubblefield 2002] govori se o efikasnim načinima da se implementira i eksploratiše taj napad. Druga nevolja sa WEP-om obuhvata CRC bitove, prikazane na slici 8.35 i prenesene u 802.11 okviru da bi otkrili promenjene bitove u korisnom teretu. Međutim, napadač koji promeni šifrovani sadržaj (odnosno, zameni originalne podatke nečim besmislenim), izračuna CRC preko zamenjenog besmislenog sadržaja i smesti CRC u WEP okvir, može da napravi 802.11 okvir koji će primalac da prihvati. Ono što je ovde potrebno su tehnike za integritet poruka, kao one koje smo proučili u odeljku 8.4, da bi se otkrilo falsifikovanje ili zamena sadržaja. Za više detalja o bezbednosti WEP-a, pročitajte [Walker 2000; Weatherspoon 2000, 802.11 Security 2004] i reference koje se tamo navode.

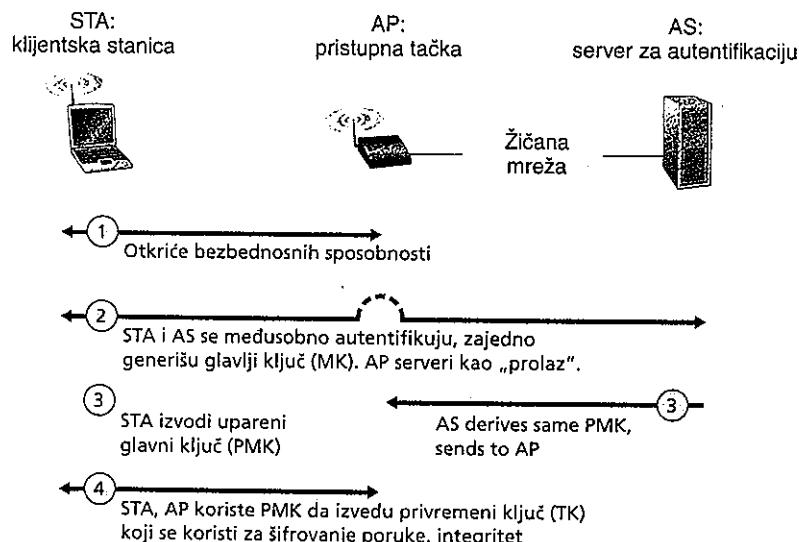


Slika 8.35 ♦ 802.11 WEP protokol

### IEEE802.11i

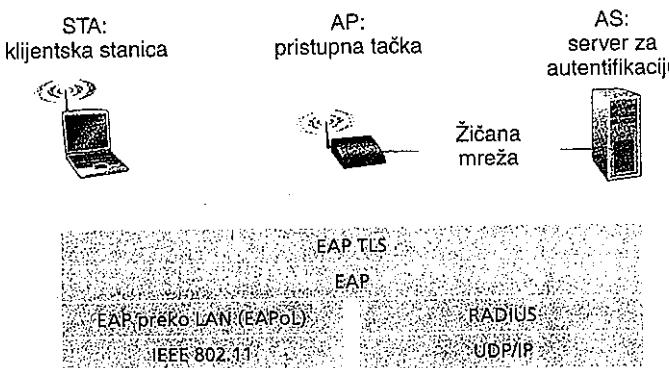
Uskoro posle izdanja IEEE 802.11 1999. godine, počeo je rad na razvoju nove i njegove poboljšane verzije sa jačim bezbednosnim mehanizmima. Novi standard, poznat kao 802.11i prolazi kroz konačnu ratifikaciju i trebalo bi da se odobri početkom 2004. godine. Kao što ćemo videti, dok je WEP obezbeđio relativno slabo šifrovanje, samo jedan način da se izvrši autentifikacija i nijedan mehanizam za distribuciju ključeva, IEEE 802.11i ima mnogo, jače oblike šifrovanja, prošir-ljiv skup mehanizama za autentifikaciju i mehanizam za distribuciju ključeva. U nastavku ćemo dati pregled 802.11i; odličan tehnički pregled (živi zvuk) 802.11i je [TechOnline 2004].

Na slici 8.36 dat je pregled okvira 802.11i. Pored bežičnog klijenta i pristupne tačke (AP), 802.11i definiše i server za autentifikaciju sa kojim AP može da komunicira. Razdvajanje servera za autentifikaciju od AP dozvoljava serveru za autentifikaciju da opslužuje mnogo AP-ova, centralizujući (često osetljive) odluke u pogledu autentifikacije i pristupa unutar jednog servera i držeći cenu i složenost AP na niskom nivou. 802.11i radi u četiri faze:



Slika 8.36 ♦ 802.11i: četiri faze rada

1. *Otkriće.* U fazi otkrića, AP oglašava svoje prisustvo i oblike autentifikacije i šifrovanja koje može da pruži bežičnom klijentskom čvoru. Klijent onda zahteva određene oblike autentifikacije i šifrovanja po želji. Iako klijent i AP već razmenjuju poruke, klijent još uvek nije autentifikovan, niti ima ključ za Šifrovanje, pa će tako biti zahtevano više postupaka pre nego što klijent bude mogao da komunicira sa proizvoljnim udaljenim računaram preko bežičnog kanala.
2. *Međusobna autentifikacija i generisanje glavnog ključa (MK).* Autentifikacija se dešava između bežičnog klijenta i servera za autentifikaciju. U ovoj fazi, pristupna tačka u suštini radi kao relaj, prosleđujući poruke između klijenta i servera za autentifikaciju. Protokol za proširljivu autentifikaciju (*Extensible Authentication Protocol*, EAP) [RFC 2284] definiše formate poruka od kraja do kraja koje se koriste u jednostavnom režimu interakcije zahtev/odziv između klijenta i servera za autentifikaciju. Kao stoje prikazano na slici 8.37, poruke EAP se enkapsuliraju korišćenjem EAPoL (EAP preko LAN-a, [IEEE 802.1X]) i šalju preko bežičnog linka 802.11. Te poruke EAP se onda dekapsuliraju u pristupnoj tački<sup>^</sup> a zatim ponovo enkapsuliraju korišćenjem protokola RADIUS za prenos preko UDP/IP do servera za autentifikaciju. Dok se RADIUS server i protokol [RFC 2138] ne zahtevaju u protokolu 802.1 li, oni su *de facto* njegove standardne komponente. Nedavno standard izovan i protokol DIAMETER [RFC 3588] će u bliskoj budućnosti verovatno zameniti RADIUS.



**Slika 8.37** ♦ EAP je protokol od kraja do kraja. Poruke EAP se enkapsuliraju koristeći EAPoL preko bežičnog linka između klijenta i pristupne tačke i RADIUS preko UDP/IP između pristupne tačke i servera za autentifikaciju.

3. *Generisanje uparenog glavnog ključa (Pairwise MasterKey, PMK).* MK je deljena tajna koju znaju samo klijent i server za autentifikaciju, a koju svaki od njih koristi da bi generisao drugi ključ, upareni glavni ključ (PMK). To smo hteli! Server za autentifikaciju onda šalje PMK u AP. Klijent i AP sada imaju deljeni ključ (setite se da u WEP-u, uopšte nije bio rešavan problem distribucije ključeva) i međusobno su se autentifikovali. Sada su oni spremni da predu na posao.
4. *Generisanje privremenog ključa (TK).* Sa PMK, bežični klijent i AP sada mogu da generišu dodatne ključeve koji će biti upotrebljeni za komunikaciju. Od posebnog značaja je privremeni ključ (TK), koji će se upotrebiti da se izvede šifrovanje na nivou linka podataka koji se šalju preko bežičnog linka ka proizvoljnom udaljenom računaru.

802.11i obezbeđuje više oblika šifrovanja, uključujući šemu za šifrovanje zasnovanu na AES-u i pojačanu verziju WEP šifrovanja.

## 8.9 Rezime

U ovom poglavlju, ispitali smo različite mehanizme koje naši tajni ljubavnici, Bob i Alisa, mogu da upotrebe kako bi „bezbedno“ komunicirali. Videli smo da su Bob i Alisa zainteresovani za tajnost (tako da samo oni mogu da razumeju sadržaj prenesene poruke), autentifikaciju (tako da su sigurni da razgovaraju jedno sa drugim) i integritet poruke (tako da su sigurni da njihove poruke nisu usput promenjene). Naravno, potreba za bezbednom komunikacijom nije ograničena samo na tajne ljubavnike. Zaista, videli smo u odeljcima 8.6 do 8.8 daje bezbednost potrebu u različitim slojevima u mrežnoj arhitekturi da bismo se zaštitili od „loših momaka“ koji imaju na raspolaganju čitav arsenal mogućnosti za napad.

U prvom delu ovog poglavlja, predstavljeni su različiti principi na kojima se zasniva bezbedna komunikacija. U odeljku 8.2 objasnili smo kriptografske tehnike za kodovanje i dekodovanje podataka, uključujući i kriptografiju simetričnim ključem i kriptografiju javnim ključem. DES i RSA su ispitani kao posebne studije primera ove dve glavne klase kriptografskih tehnika koje se koriste u današnjim mrežama. U odeljku 8.3 obratili smo pažnju na autentifikaciju i razvili smo niz sve složenijih protokola za autentifikaciju, koji obezbeđuju daje sagovornik zaista onaj za koga se on/ona izdaje i da ide „uživo“. Videli smo da i kriptografija simetričnim ključem i kriptografija javnim ključem mogu da igraju važnu ulogu u skrivanju podataka (Šifrovanju/dešifrovanju), ali takođe i u obavljanju autentifikacije. Tehnike za „potpisivanje“ digitalnog dokumenta na način koji je proverljiv, nekrivotovorljiv i neopoziv, objašnjene su u odeljku 8.4. Još jednom, primena kriptografskih tehnika dokazala se kao suštinska. Ispitali smo i digitalne potpise i izvode iz poruka, od kojih su ovi poslednji skraćen način za potpisivanje digitalnog dokumenta. U odeljku 8.5

ispitali smo protokole za raspodelu ključeva. Videli smo da za Šifrovanje simetričnim ključem, centar za raspodelu ključeva - mrežni entitet od povejena - može da se koristi za raspodelu deljenog simetričnog ključa između strana u komunikaciji. Za Šifrovanje javnim ključem, sertifikaciono telo distribuira uverenja da bi potvrdilo javne ključeve.

Naoružani tehnikama objašnjenim u odeljcima od 8.2 do 8.5, Bob i Alisa mogu bezbedno da komuniciraju. (Možemo se samo nadati da su njih dvoje studenti umrežavanja koji su proučili ovaj materijal i zato mogu da izbegnu da Trudi otkrije njihov sastanak!) Ali, sve više, tajnost je samo mali deo slike o mrežnoj bezbednosti. Mrežna bezbednost je sve više usmerena na mrežnu infrastrukturu, a protiv mogućeg juriša „loših momaka“. Zato smo u poslednjem delu ovog poglavlja objasnili mrežne barijere (kojima se reguliše pristup u zaštićene mreže i izlazak iz njih) i niz napada koje lica spolja mogu da pokrenu protiv mreže, kao i protivmere koje protiv njih mogu da se prednazmu. Zaključili smo poglavljje studijama primera u kojima se primenjuju tehnike iz odeljaka od 8.2 do 8.5 u aplikacionom sloju, transportnom sloju, mrežnom sloju i sloju veze podataka.

## Domaći zadatak: problemi i pitanja Poglavlje 7

### Kontrolna pitanja

1. Koje su razlike između tajnosti i integriteta poruke? Da li jedna može bez druge? Opravdajte svoj odgovor.
2. Koja je razlika između aktivnog i pasivnog uljeza?
3. Koja je razlika između sistema simetričnog ključa i sistema javnog ključa?
4. Prepostavite da uljez ima šifrovani poruku, kao i dešifrovani verziju te poruke, Da li uljez može da preduzme napad samo šifrovanim tekstrom, napad poznatim otvorenim tekstrom ili napad izabranim otvorenim tekstrom?
5. Prepostavite da  $N$  ljudi želi da komunicira sa svakim od  $N-1$  drugih ljudi, koristeći Šifrovanje simetričnim ključem. Celokupna komunikacija između dvoje ljudi,  $i$  i  $j$ , vidljiva je svim drugim ljudima u toj grupi od  $N$ , a nijedna druga osoba u toj grupi ne bi trebalo da može da dešifruje njihovu komunikaciju. Koliko ključeva se zahteva u sistemu u celini? Sada prepostavite da se koristi šifrovanje javnim ključem. Koliko ključeva se zahteva u tom slučaju?
6. Koja je namena jednokratne vrednosti u protokolu autentifikacije?
7. Šta znači kada se kaže daje jednokratna vrednost ona koja se upotrebi jednom u životu? U Čijem životu?
8. Šta je to napad Čoveka u sredini? Može li taj napad da se dogodi kada se koriste simetrični ključevi?

9. Šta znači za potpisani dokumenc da bude proverljiv, nekrivotvorljiv i neopoziv?
10. Na koji način izvod iz poruke obezbeđuje bolju proveru integriteta poruke od kontrolnog zbirka kao sto je Internet kontrolni zbir?
11. Na koji način izvod iz poruke šifrovani javnim ključem obezbeđuje „holji“ digitalni potpis od korišćenja poruke šifrovane javnim ključem?
12. Da li je poruka pridružena izvodu iz poruke šifrovana? Objasnite svoj odgovor.
13. Staje li centar za raspodelu ključeva? Staje li sertifikaciono telo?
14. Sumirajte ključne razlike u uslugama koje pružaju protokol Authentication Header(AH) i protokol Encapsulation Security Payload (ESP) u IPsecu.

### Problemi

1. Koristeći jednoazbučnu Šifru na slici 8.3, šifrujte poruku „This is an easy problem“. Dešifrujte poruku „rmij ' u uamu xyj“.
2. Pokažite da Trudin napad poznatim otvorenim tekstrom, u kome ona zna parove prevoda (šifrovani tekst, otvoreni tekst) za sedam slova, smanjuje broj mogućih zamena koji treba da se proveri u primeru 8.2.1 za približno  $10^9$ .
3. Razmotrite višeazbučni sistem prikazan na slici 8.4. Da li će napad izabranim otvorenim tekstrom koji je u stanju da dobije šifrovanje otvorenog teksta poruke, „The quick brown fox jumps over the lazy dog“ biti dovoljan da dešifruje sve poruke? Zašto, ili zašto ne?
4. Korišćenjem RSA, izaberite  $p = 3$  i  $q = 11$  i Šifrujte reč „hello“. Primenite algoritam za dešifrovanje na šifrovani verziju da biste rekonstruisali prvobitnu poruku otvorenog teksta.
5. Razmotrite naš protokol autentifikacije 4.0, u kome se Alisa autentificuje Bobu, za koji smo videli da dobro radi (to jest, u njemu nismo pronašli greške). Sada prepostavite da u isto vreme kada se Alisa autentificuje Bobu, Bob i sam mora da se autentificuje Alisi. Dajte scenario u kome Trudi, praveći se daje AHsa, sada može da se autentificuje Bobu kao Alisa. (Savet: zamislite da sekvencija operacija  $ap4,0$ , ona koju započinje Trudi i ona koju započinje Bob, mogu da se proizvoljno isprepletu. Posebno obratite pažnju na činjenicu da će i Bob i Alisa koristiti jednokratnu vrednost i da, ako se ne pazi, može da se zlo-namerno upotrebii ista jednokratna vrednost).
6. U napadu čoveka u sredini na slici 8.13, Alisa nije autentifikovala Boba. Daje Alisa zahteva da se Bob autentificuje koristeći  $ap5,0$ , da li bi napad čoveka u sredini mogao da se izbegne? Objasnite svoje razmišljanje.

7. Internet protokol za rutiranje BGP-a koristi MD5 izvod iz poruke a ne šifrovanje javnim ključem za potpisivanje BGP poruka. Šta mislite zašto je izabran MD5 a ne šifrovanje javnim ključem?
8. Izračunajte treću poruku, različitu od dve poruke na slici 8.18, koja ima isti kontrolni zbir kao poruke na slici 8.18.
9. Zašto Alisa ne mora eksplisitno da autentikuje Boba u protokolu i diskusiji slike 8.19?
10. Zašto nema eksplisitne autentifikacije u protokolu na slici 8.19? Da li je autentifikacija potrebna? Zašto?
11. Razmotrite KDC i CA servere. Pretpostavite da KDC otkaze. Kako to utiče na sposobnost strana da bezbedno komuniciraju; odnosno, ko može, a ko ne može da komunicira? Obrazložite svoj odgovor. Sada pretpostavite da otkaze CA. Kakav je uticaj tog otkaza?
12. Razmotrite sledeću varijaciju mrežne barijere za filtriranje paketa u odeljku 8.6. Pretpostavite da Alisa želi da zabrani pristup svojoj mreži 222.22.0.0/16 sa javnog Interneta (pravilo R3 u prvoj tabeli dalje u tekstu). Alisa opet sarađuje sa Bobom i njegovim kolegama, koji su na univerzitetu, pa Alisa zato želi da dozvoli korisnicima sa Bobovog univerziteta (čija je mrežna adresa 111.11.11/16) pristup određenoj podmreži, 222.22.22/24, unutar mreže njene kompanije (pravilo R1 dole). Alisa zna da je Trudi, dobro poznati haker, na Bobovom univerzitetu i da je Trudina podmreža, 111.11.11/24, pravi raj za nepouzdane hakere. Zato Alisa želi da nikakav saobraćaj iz 111.11.11/24 ne uđe bilo gde u njenu mrežu (pravilo R2), izuzev u specijalnu podmrežu 222.22.22/24 (dozvoljeno pravilom R1; ovaj izuzetak je značajna razlika od našeg primera u odeljku 8.6). Alisina pravila za filtriranje paketa sumirana su u sledećoj tabeli.

| Pravilo | Adresa izvora | Adresa odredišta | Akcija  | Komentari                                                                                                              |
|---------|---------------|------------------|---------|------------------------------------------------------------------------------------------------------------------------|
| R1      | 111.11/16     | 222.22.22/24     | dopušta | Pušta datagrome iz Bobove univerzitetske mreže (uključujući i one iz Trudine hakerske podmreže) u ograničenu podmrežu. |
| R2      | 111.11.11/24  | 222.22/16        | odbija  | Ne pušta saobraćaj iz Trudine podmreže u Alisinu mrežu; ali pogledajte R1.                                             |
| R3      | 0.0.0.0/0     | 0.0.0.0/0        | odbija  | Ne pušta saobraćaj u Alisinu mrežu.                                                                                    |

- ◆ Popunite sledeću tabelu akcijama koje se preduzimaju u ovom scenaruju prema redosledu R1, R2, R3 i prema redosledu R2, R1, R3.
- ◆ Koji bi bio rezultat uklanjanja pravila R2 za pakete P1, P2, P3 i P4?

| Broj datagrama | IP adresa izvora                                | IP adresa odredišta                  | Poželjna akcija | Akcija prema R2, R1, R3 | Akcija prema R1, R2, R3 |
|----------------|-------------------------------------------------|--------------------------------------|-----------------|-------------------------|-------------------------|
| P1             | 111.11.11.1<br>(hakerska podmreža)              | 222.22.6.6<br>(korp.mreža)           | odbija          |                         |                         |
| P2             | 111.11.11.1<br>(hakerska podmreža)              | 222.22.22.2<br>(specijalna podmreža) | dopušta         |                         |                         |
| P3             | 111.11.6.6<br>(univ.mreža, nehakersko podmreža) | 222.22.22.2<br>(specijalna podmreža) | dopušta         |                         |                         |
| P4             | 111.11.6.6<br>(univ.mreža, nehakersko podmreža) | 222.22.6.6<br>(korp.mreža)           | odbija          |                         |                         |

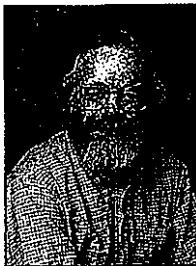
- V3. Na slici 8.29 prikazane operacije koje Alisa mora da izvede da bi obezbedila tajnost, autentifikaciju, integritet. Napravite dijagram odgovarajućih operacija tcoje Bob mora da izvede na pakovanju primljenom od Alise.

### Teze za diskusiju

1. Pretpostavite da uljez može i da umetne i da ukloni DNS poruke u mreži. Dajte tri scenarija koji rešavaju probleme koje bi takav uljez mogao da prouzrokuje.
2. Niko nije formalno dokazao da su 3DES ili RSABezbedni. Imajući to u vidu, kakav dokaz imamo da su oni zaista bezbedni?
3. Ako IPsec ostvaruje bezbednost u mrežnom sloju, zašto su onda bezbednosni mehanizmi i dalje potrebi u slojevima iznad IP-a?
4. Idite na medunarodnu veb lokaciju PGP (<http://www.pgpi.org/>). Koja verzija PGP-a vam je legalno dozvoljena za preuzimanje, imajući u vidu zemlju u kojoj se nalazite?

## Stiven M. Belovin

Stiven M. Belovin je AT&T Fellow u NeNvork Seivices Research Lab u AT&T Labs Research u Froham Parku, New Jersey. Oblasti njegovog interesovanja su mreže, bezbednost i piianje zašto su te dve stvari nekomaptibilne. 1995. godine, dobio je nagradu Usenix Lifetime Achievement Award za svoj rad na stvaranju Useneta, prve mreže za grupnu razmenu vesti koja je povezala dva ili više računara i dozvolila korisnicima da dele informacije i učestvuju u diskusijama. Sliv je takođe izabrani član National Academy of Engineering. Diplomirao je na Kolumbijskom univerzitetu, a doktorirao na Univerzitetu Severne Kroline u Capel Hilu.



### Šta Vas je navelo da se specijalizujete u oblasti bezbednosti umrežavanja?

Ovo će da zvuči čudno, ali odgovor je prost: to je bilo zabavno. Moji korenii su bili u programiranju sistema i sistemskoj administraciji, što dosta prirodno vodi ka bezbednosti. I uvek sam bio zainteresovan za komunikacije, još od poslova sistemskog programiranja sa delimičnim radnim vremenom, kada sam bio na koledžu.

Moj rad na bezbednosti nastavlja da bude motivisan dvema stvarima - željom da raču-nari budu i dalje korisni, Sto znači da njihovu funkciju ne ugroze napadači i željom da se zaštiti privatnost.

### Kakva je Vaša vizija Useneta iz vremena kada ste ga razvijali? A sada?

Prvobitno smo ga videli kao način da pričamo o računarskoj nauci i programiranju računara Sirom zemlje, sa mnogo lokalne upotrebe za administrativne poslove, za oglase o rasprodajama itd. U stvari, moje prvobitno predviđanje bilo je jedna do dve poruke na dan, na možda 50 do 100 lokacija. Ali, stvarni rast bio je u temama koje su u vezi sa ljudima, uključujući - ali ne ograničavajući se na - ljudske interakcije sa računarima. Moje omiljene elektronske grupe, tokom godina, bile su one kao *rec.woodworking*, kao i *sci.crypt*.

U izvesnoj meri, Web je istisnuo mrežne vesti. Kada bili danas počeo da ih projektu-jem, to bi izgledalo mnogo drugačije. Ali, one su i dalje način da se postigne veoma velik auditorijum koji je zainteresovan za određenu temu, bez obaveze da se oslanja na posebne veb lokacije.

### Da li Vas je neko profesionalno inspirisao? Na koji način?

Profesor Fred Bruks - osnivač i prvi šef katedre odeljenja za računarske nauke na Univerzitetu Severne Kroline u Čapel Hilu, rukovodilac tima koji je razvio IBM S/360 i OS/360 i autor knjige *The Mythical Man-Month* - imao je ogroman uticaj na moju karijeru. Više od bilo Čega drugog, on je predavao o gledištima i kompromisima - kako da se

sagledavaju problemi u kontekstu stvarnog sveta (i koliko je stvarni svet liaočiniji od onoga što bi teoretičar voleo) i kako da se uravnoteže suprotni interesi u projektovanju rešenja. Najveći deo rada sa računarima je inženjersrvo - umetnost pravljenja dobrih kompromisa da bi se zadovoljili mnogi kontradiktorni ciljevi.

### Kakva je Vaša vizija budućnosti umrežavanja i bezbednosti?

Do sada, najviše bezbednosti koju imamo dolazi od izolacije. Na primer, mrežna barijera radi tako što preseca pristup izvesnim mašinama i uslugama. Ali, mi smo u dobu povećanog povezivanja - pa je sve teže **da** se stvari odvoje. Što je još gore, naši proizvodni sistemi zahtevaju daleko više razdvojene celine, medusobno povezane pomoću mreža. Obezbedenje svega toga jedan je od naših najvećih izazova.

### Da li biste mogli da nam kažete nešto o specijalnim projektima na kojima baš sada radite?

Mnogo vremena posvećujem operacionainim pitanjima. Za sistem nije dovoljno da bude bezbedan; on treba **da** bude i upotrebljiv. To sa svoje strane znači da mrežni operatori moraju da budu sposobni za nadgledanje stvari. Ali, nadgledanje može da se sukobi sa bezbednošću - bezbednost teži da sakrije i zaštititi računare i informacije, ali operatori treba da vide izvesne stvari. Pored toga, sistemi treba da nastave sa radom, čak i kada najvažniji delovi otkazu. Kako da delimo informacije pravim stranama, a da ne pustimo loše momke unutra? To je pitanje čijim se odgovorom baš sada bavim.

### Za Šta biste rekli da su najveća dostignuća u bezbednosti? Koliko daleko možemo da idemo?

Sa naučne tačke gledišta, znamo da radimo kriptografiju. To je bilo od velike pomoći. Ali najveći problemi bezbednosti proističu iz programa sa greškama, a to je mnogo teži problem. U stvari, to je najstariji nerešeni problem u računarskoj nauci i ja mislim da će on to i ostati. Izazov je da se smisli kako da se obezbede sistemi kada treba da ih izgradimo od nebezbednih sastavnih delova. To već možemo da uradimo u pogledu hardverskih otkaza; da li možemo da uradimo isto i za bezbednost?

### Da li imate neki savet o Internetu i bezbednosti umrežavanja koji biste uputili studentima?

Učenje mehanizama je lakši deo. Učenje kako da se „paranoidno razmišlja“ je teže. Treba **da** zapamrite da se raspodele verovatnoće ne primenjuju - napadači mogu da nađu, i naći će neverovatne uslove. A detalji su veoma važni.