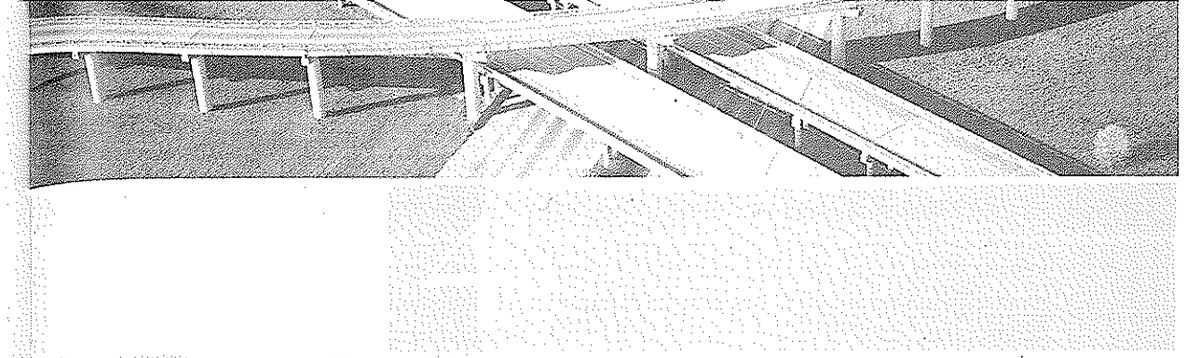


Mrežni sloj

U prethodnom poglavlju naučili smo da transportni sloj obezbeđuje različite vidove komunikacije između procesa, oslanjajući se na uslugu komunikacije između računara, koju obezbeđuje mrežni sloj. Naučili smo takođe da transportni sloj to radi bez ikakvog saznanja o tome kako mrežni sloj u stvari ostvaruje tu uslugu. Možda se sada pitate šta omogućava ovu komunikaciju između računara, šta je to pokreće?

U ovom poglavlju naučićemo kako mrežni sloj ostvaruje uslugu komunikacije od računara do računara. Videćemo da, za razliku od transportnog sloja i aplikativnog sloja, određeni deo mrežnog sloja postoji u svakom računaru i ruteru na mreži. Zato su protokoli mrežnog sloja među najizazovnijim (stoga i najzanimljivijim!) u skupu protokola.

Mrežni sloj je ujedno jedan od najsloženijih slojeva u skupu protokola i zato nas u ovom poglavlju čeka puno tema. Proučavanje počinjemo kraćim prikazom mrežnog sloja i usluga koje on pruža. Zatim ćemo ispitati dva opšta rešenja koja se koriste za organizovanje isporuke paketâ namrežnom sloju – model datagrama i model virtuelnih kola – i videćemo suštinsku ulogu adresiranja u isporuci paketâ do odredišnog računara.

U ovom poglavlju uočićemo značajnu razliku između funkcionalnosti **prosleđivanja** i **rutiranja** koji se obavljaju na mrežnom sloju. Prosleđivanje obuhvata prenos paketa od ulaznog do izlaznog linka unutar *istog* rutera. Rutiranje obuhvata

sve mrežne rutere, čijom se saradnjom pomoću protokola rutiranja određuju putanje kojima paketi putuju od izvornog do odredišnog čvora. Ovo će biti značajna razlika koju bi trebalo da imate na umu dok budete napredovali u ovom poglavlju.

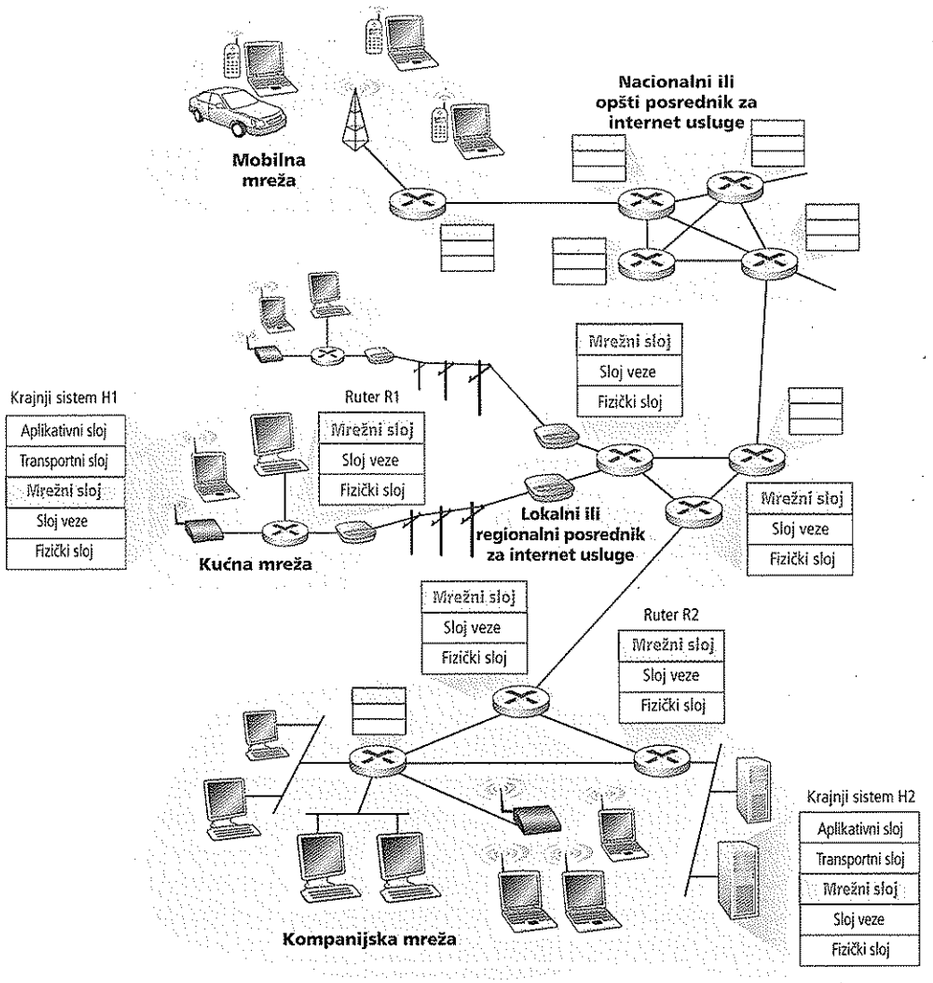
Da bismo bolje razumeli prosleđivanje paketa, zavirićemo „unutar" rutera – u njegovu hardversku arhitekturu i organizaciju. Zatim razmatramo prosleđivanje paketa na internetu, uključujući i slavni IP (Internet Protocol). Istražićemo adresiranje na mrežnom sloju i format datagrama IPv4. Zatim istražujemo prevođenje mrežnih adresa (engl. *network address translation*, NAT), fragmentiranje datagrama, internet protokol za kontrolne poruke (Internet Control Message Protocol, ICMP) i IPv6.

Nakon toga svoju pažnju usmeravamo na rutiranje na mrežnom sloju. Videćemo da je zadatak algoritma rutiranja pronalaženje dobrih putanja (odnosno ruta) od pošiljalaca do primalaca. Prvo teoretski proučavamo algoritame rutiranja, a posebnu pažnju posvećujemo dvema preovlađujućim klasama algoritama: algoritmu stanja linkova i algoritmu vektora rastojanja. S obyirom da složenost algoritama rutiranja značajno raste sa porastom broja rutera u mreži, obradićemo i hijerarhijsko rutiranje. Zatim ćemo, kada pređemo na protokole rutiranja unutar autonomnog sistema (RIP, OSPF i IS-IS) i protokol rutiranja između autonomnih sistema, BGP, videti kako se ova teorija sprovodi u praksi. Poglavlje zaključujemo opisom difuznog i višeznačnog rutiranja.

Sve u svemu, ovo poglavlje ima tri dela. Prvi deo – odeljci 4.1 i 4.2 – obuhvata opise funkcija i usluga mrežnog sloja. Drugi deo – u odeljcima 4.3 i 4.4 – obrađuje se prosleđivanje. Na kraju, treći deo – u odeljcima od 4.5 do 4.7 – obrađuje se rutiranje.

4.1 Uvod

Na slici 4.1 je prikazana jednostavna mreža sa dva računara, H1 i H2, i nekoliko rutera na putanji između računara H1 i H2. Pretpostavimo da H1 šalje informacije ka H2 i posmatramo koju ulogu ima mrežni sloj u ovim računarima i u ruterima koji ih povezuju. Mrežni sloj računara H1 preuzima segmente od transportnog sloja računara H1, enkapsulira svaki segment u datagram (odnosno, paket mrežnog sloja) i zatim šalje te datagrame na najbliži ruter R1. U prijemnom računaru, H2, mrežni sloj prima datagrame od najbližeg rutera R2, izdvaja segmente transportnog sloja i isporučuje ih naviše – transportnom sloju računara H2. Primarna uloga rutera je prosleđivanje datagrama od ulaznih do izlaznih linkova. Obratite pažnju na to da su ruteri na slici 4.1 prikazani bez svih protokola iz skupa protokola, tj. bez slojeva iznad mrežnog sloja, jer (osim zbog kontrole) ruteri ne izvršavaju protokole aplikativnog i transportnog sloja kakve smo razmatrali u poglavljima 2 i 3.



Slika 4.1 ◆ Mrežni sloj

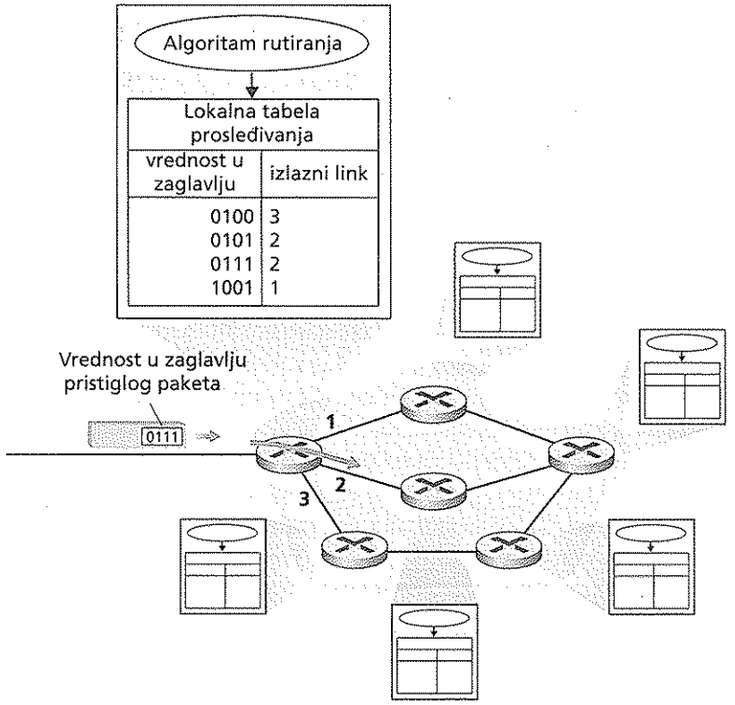
4.1.1 Prosleđivanje i rutiranje

Uloga mrežnog sloja je naizgled jednostavna – prenošenje paketa od predajnog do prijemnog računara. Da bi to obavio, mrežni sloj mora da obavi dva važna zadatka:

- *Prosleđivanje.* Kada paket stigne na ulazni link rutera, on mora da ga premesti na odgovarajući izlazni link. Na primer, paket koji od računara H1 stigne u ruter R1 mora da se prosledi sledećem ruteru na putanji prema računaru H2. U odeljku 4.3 zavirićemo unutar rutera i ispitati kako se u suštini paket prosleđuje od ulaznog linka do izlaznog linka rutera.
- *Rutiranje.* Mrežni sloj mora da utvrdi rutu ili putanju kojom paketi protiču od pošiljaoca do primaoca. Algoritme koji izračunavaju te putanje nazivamo **algoritmima rutiranja**. Algoritam rutiranja trebalo bi da, na primer, utvrdi putanju kojom paketi teku od računara H1 do H2.

Prilikom opisivanja mrežnog sloja autori često mešaju izraze *prosleđivanje* i *rutiranje*. Mi ćemo ih u ovoj knjizi koristiti mnogo preciznije. *Prosleđivanje* se odnosi na delatnost unutar rutera prilikom prenosa paketa sa interfejsa ulaznog linka na odgovarajući interfejs izlaznog linka. *Rutiranje* se odnosi na proces određivanja putanje sa kraja na kraj, kroz celu mrežu kojom paketi prolaze od izvora do odredišta. Koristeći poređenje sa automobilima, vratimo se primeru putovanja od Pensilvanije do Floride iz odeljka 1.3.1. Tokom ovog putovanja naš vozač prolazi više raskrsnica na putu do Floride. Prosleđivanje možemo posmatrati kao prolazak kroz pojedinačnu raskrsnicu: auto ulazi u raskrsnicu sa jednog uključenja i zatim određuje kojim bi isključenjem trebalo da ide da bi napustio raskrsnicu. Rutiranje možemo posmatrati kao planiranje putovanja od Pensilvanije do Floride: pre nego što krene na put, vozač proučava kartu i bira jednu od više mogućih putanja, gde se svaka putanja sastoji od više delova puta povezanih raskrsnicama.

Svaki ruter ima **tabelu prosleđivanja**. Ruter prosleđuje paket tako što ispituje vrednost određenog polja u zaglavlju pristiglog paketa, a zatim, koristeći tu vrednost pretražuje tabelu prosleđivanja rutera. Rezultat dobijen iz tabele prosleđivanja ukazuje na koji bi izlazni interfejs linka rutera trebalo proslediti taj paket. U zavisnosti od protokola mrežnog sloja, ova vrednost iz zaglavlja paketa može da bude odredišna adresa tog paketa ili naznaka veze kojoj taj paket pripada. Na slici 4.2 dat je jedan primer. Na slici 4.2, paket koji u polju zaglavlja ima vrednost 0111 stiže u ruter. Ruter pretražuje svoju tabelu prosleđivanja za tu vrednost i utvrđuje da je interfejs izlaznog linka za ovaj paket interfejs 2. Posle toga, ruter interno prosleđuje paket interfejsu 2. U odeljku 4.3 zavirićemo u ruter i prosleđivanje proučiti mnogo podrobnije.



Slika 4.2 ◆ Algoritmi rutiranja određuju vrednosti u tabeli prosleđivanja

Možda se pitate kako se konfigurišu tabele prosleđivanja u ruterima. Ovo je presudno pitanje, ono iz koga se vidi značajno međudejstvo rutiranja i prosleđivanja. Kao što se vidi na slici 4.2, algoritam rutiranja određuje vrednosti koje su ubačene u tabele prosleđivanja rutera. Algoritam rutiranja može da bude centralizovan (algoritam se izvršava na jednom mestu, a pojedini ruteri preuzimaju informacije za rutiranje) ili decentralizovan (na svakom ruteru izvršava se deo distribuiranog algoritma rutiranja). U oba slučaja, ruter prima poruke protokola rutiranja, koje se koriste za konfigurisanje njegove tabele prosleđivanja. Razlika između prosleđivanja i rutiranja i njihova drugačija svrha mogu se bolje prikazati još jednim hipotetičkim (nerealnim, ali tehnički izvodljivim) primerom mreže u kojoj sve tabele prosleđivanja neposredno konfigurišu mrežni operateri, fizički prisutni kod rutera. U tom slučaju *ne* bi bio potreban nikakav protokol rutiranja! Naravno, operateri bi morali da se dogovaraju oko konfigurisanja tabela 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 u topologiji mreže. Sreća je naša da sve mreže samostalno obavljaju prosleđivanje *i* rutiranje!

Dok se bavimo terminologijom, trebalo bi pomenuti još dva izraza koja se često mešaju, a koja ćemo mi koristiti pažljivije. Izraz *komutator paketa* koristićemo sa značenjem opšti uređaj za komutiranje paketa, koji prenosi paket iz interfejsa ulaznog linka u interfejs izlaznog linka, u zavisnosti od vrednosti u polju zaglavlja paketa. Neki komutatori paketa, koje zovemo **komutatori sloja veze** (koje istražujemo u poglavlju 5), svoju odluku o prosleđivanju zasnivaju na vrednosti u polju sloja veze; komutatori se stoga mogu posmatrati kao uređaji sloja veze (sloj 2). Drugi komutatori paketa, koje nazivamo **ruterima**, svoju odluku o prosleđivanju zasnivaju na vrednosti u polju zaglavlja mrežnog sloja. Ruteri su stoga uređaji mrežnog sloja (sloj 3), ali moraju takođe da realizuju i protokole sloja 2, pošto uređaji sloja 3 zahtevaju usluge sloja 2 da bi ostvarili svoju funkcionalnost (sloj 3). (Da biste sasvim shvatili ovu važnu razliku, bilo bi dobro da ponovo pročitate odeljak 1.5.2 u kome razmatramo datagrame mrežnog sloja i okvire sloja veze kao i njihove odnose.) Da zakomplikujemo stvari, literatura iz oblasti marketinga često „komutatore sloja 3" naziva ruterima sa Eternet interfejsima, međutim to su u stvari uređaji sloja 3. Pošto se u ovom poglavlju bavimo mrežnim slojem, koristićemo izraz *ruter,* umesto izraza *komutator paketa*. Izraz *ruter* koristićemo čak i dok budemo govorili o komutatorima paketa u mrežama sa virtuelnim kolima (o kojima će uskoro biti reči).

Uspostavljanje veze

Upravo smo rekli da mrežni sloj obavlja dva važna zadatka, prosleđivanje i rutiranje. Ali, uskoro ćemo videti da u nekim računarskim mrežama postoji i treća značajna funkcija mrežnog sloja, a to je uspostavljanje veze. Verovatno se sećate iz izlaganja o protokolu TCP da je potrebno trostruko usaglašavanje pre nego što podaci počnu da teku od pošiljaoca ka primaocu. Na takav način pošiljalac i primalac postavljaju potrebne informacije o stanju (na primer, redni broj i početnu veličinu prozora za kontrolu toka). Slično tome, neke arhitekture mrežnog sloja – na primer, ATM i frame-relay i MPLS (engl. Multi Protocol Label Switching odnosno višeprotokolsko komutiranje paketa na osnovu oznaka; o čemu ćemo govoriti u poglavlju 5.8) – od rutera duž izabrane putanje, od izvora do odredišta, zahtevaju usaglašavanje da bi se postavilo stanje pre nego što paketi podataka mrežnog sloja unutar date veze od izvora do odredišta počnu da teku. Na mrežnom sloju ovaj postupak naziva se uspostavljanje veze. Uspostavljanje veze istražujemo u odeljku 4.2.

4.1.2 Modeli usluga mreže

Pre nego što dublje uronimo u mrežni sloj, pogledajmo opštiju sliku i razmotrimo kakve sve vrste usluga može da ponudi mrežni sloj. Kada transportni sloj na predajnom računaru prenese paket u mrežu (odnosno, preda ga naniže, mrežnom sloju na tom računaru), može li transportni sloj da računa na to da će mrežni sloj isporučiti taj paket na odredište? Kada se šalje više paketa, da li će oni biti isporučeni transportnom sloju prijemnog računara istim redom kojim su poslati? Da li će vreme koje protekne između slanja dva uzastopna paketa biti jednako vremenu koje protekne

između njihovih prijema? Da li će mreža obezbediti ikakvu povratnu informaciju o zagušenju na mreži? Kakav je apstraktan prikaz (tj. koja su svojstva) kanala koji povezuje transportne slojeve na predajnom i prijemnom računaru? Odgovori na ova pitanja i na mnoga druga zavise od modela usluge koju pruža mrežni sloj. **Model usluge mreže** definiše karakteristike prenosa podataka od kraja do kraja, to jest, između predajnog i prijemnog krajnjeg sistema.

Razmotrimo sada neke usluge koje bi mrežni sloj mogao da pruža. Na predajnoj strani, kada transportni sloj preda paket mrežnom sloju, specifične usluge koje bi mrežni sloj mogao da ponudi obuhvataju:

- *Garantovana isporuka.* Ova usluga garantuje da će paket pre ili kasnije stići na svoje odredište.
- *Garantovana isporuka sa ograničenim kašnjenjem.* Ova usluga, ne samo da garantuje isporuku paketa, već garantuje isporuku od računara do računara u određenom roku (na primer, u roku od 100 ms).

Osim toga, sledeće usluge bi se mogle ponuditi *toku paketa* između datog izvora i odredišta:

- *Isporuka u ispravnom redosledu.* Ova usluga garantuje da paketi stižu na odredište onim redom po kome se šalju.
- *Garantovani minimalni propusni opseg.* Ova usluga mrežnog sloja oponaša ponašanje prenosnog linka određene brzine (npr. 1 Mb/s) između predajnog i prijemnog računara. Sve dok predajni računar otprema bitove (kao delove paketa) brzinom manjom od utvrđene, nema gubitaka paketa i svaki paket stiže od računara do računara u okviru unapred određenog kašnjenja (na primer, u roku od 40 ms).
- *Garantovana maksimalna promenljivost kašnjenja.* Ova usluga garantuje da će vreme koje protekne između slanja dva uzastopna paketa kod pošiljaoca biti jednako vremenu koje protekne između njihovog prijema na odredištu (ili da se razlika između ova dva vremena neće menjati za više od neke utvrđene vrednosti).
- *Bezbednosne usluge.* Korišćenjem tajnog ključa koji znaju samo izvorni i odredišni računar, mrežni sloj na izvoru može da šifrira korisne podatke u svim datagramima koji se šalju odredišnom računaru. Mrežni sloj na odredišnom računaru odgovoran je za dešifrovanje ovih podataka. Takvom uslugom mogla bi se obezbediti poverljivost svih (TCP i UDP) segmenata transportnog sloja između izvornog i odredišnog računara. Pored poverljivosti, mrežni sloj bi mogao da ponudi usluge očuvanja integriteta podataka i proveru autentičnosti izvora podataka.

Ovo je samo delimičan spisak usluga koje bi mrežni sloj mogao da ponudi – moguće su bezbrojne varijacije.

Mrežni sloj interneta obezbeđuje jedinstvenu uslugu, poznatu kao **usluga najboljeg pokušaja**. Iz tabele 4.1 mogao bi se steći utisak da je *usluga najboljeg po-*

kušaja eufemizam za *bez ikakve usluge*. U usluzi najboljeg pokušaja ne garantuje se pravovremena isporuka paketa, ne garantuje se da će paketi biti primljeni redosledom kojim su poslati, niti se uopšte garantuje isporuka poslatih paketa. Prema datoj definiciji, mreža koja ne isporučuje *nijedan* paket na odredište zadovoljava definiciju usluge najboljeg pokušaja. Međutim, kao što ćemo uskoro objasniti, za tako minimalistički model usluge mrežnog sloja postoje valjani razlozi.

Mrežna arhitektura	Model usluge	Garantovanje propusnog opsega	Garantovanje da nema gubitaka	Redosled	Ispunjenje vremenskih zahteva	Ukazivanje na zagušenje
Internet	najbolji pokušaj	nema	nema	po bilo kom mogućem redosledu	ne održava se	nema
ATM	CBR	garantovanostalna brzina	da	po redosledu	održava se	ne dolazi do zagušenja
ATM	ABR	Garantovana minimalna brzina	nema	po redosledu	ne odražava se	obezbeđeno ukazivanje na zagušenje

Tabela 4.1 ◈ Modeli usluga internet, ATM CBR i ATM ABR mreža

U drugim mrežnim arhitekturama su definisani i ostvaruju se modeli usluga koji prevazilaze uslugu najboljeg pokušaja koju nudi internet. Na primer, arhitektura ATM mreže [MFA Forum 2012, Black 1995] predviđa višestruke modele usluga, što znači da različite veze dobijaju različite klase usluga unutar iste mreže. Objašnjenje načina na koji ATM mreža obezbeđuje takve usluge prevazilazi okvire ove knjige; ovde nam je cilj samo da napomenemo da postoje alternative za uslugu najboljeg pokušaja koju nudi internet. Dva najvažnija modela usluge ATM mreža su usluga stalne bitske brzine i usluga raspoložive bitske brzine:

- ***Usluga stalne bitske brzine (constant bit rate, CBR) ATM mreže.*** Ovo je bio prvi standardizovan model usluga ATM mreža u kome se odražava interes telefonskih kompanija za ATM mreže i pogodnost CBR usluge za prenos audio i video saobraćaja u realnom vremenu stalnom bitskom brzinom. Cilj CBR usluge je po zamisli jednostavan – obezbediti protok paketâ (poznatih kao ćelije u terminologiji ATM mreža) virtuelnim cevovodom koji se ponaša kao da između predajnog i prijemnog računara postoji link fiksnog propusnog opsega. Sa CBR uslugom, protok ATM ćelija preko mreže obavlja, garantujući da će kašnjenje ćelija sa kraja na kraj, promenljivost kašnjenja ćelija sa kraja na kraj (odnosno, džiter) i udeo ćelija koje se izgube, ili se prekasno isporuče, biti manji od utvrđene vrednosti. Predajni računar i ATM mreža dogovaraju se o tim vrednostima prilikom prvog uspostavljanja CBR veze.

- ***Usluga raspoložive bitske brzine (available bit rate, ABR) ATM mreže.*** Ako se za uslugu koju nudi internet može reći da je usluga najboljeg pokušaja, onda bi najbolji opis za ABR uslugu ATM mreža bio da je to usluga neznatno bolja od najboljeg pokušaja. Kao i sa modelom usluge interneta, ćelije mogu da se izgube korišćenjem ABR usluge. Međutim, za razliku od interneta, ćelijene mogu promeniti redosled (mada mogu da se izgube), a vezi koja koristi ABR uslugu garantuje se minimalna brzina prenosa ćelija (minimum cell transmission rate, MCR). Ako u određenom trenutku mreža ima dovoljno slobodnih resursa, pošiljalac može uspešno da šalje ćelije brzinom većom od MCR. Osim toga, kao što smo videli u odeljku 3.6, ABR usluga ATM mreže može pošiljaocu da obezbedi povratne informacije (u vidu bita za upozorenje o zagušenju ili obaveštenja o brzini kojom bi trebalo da šalje) koje regulišu kako pošiljalac prilagođava svoju brzinu između MCR brzine i dozvoljene maksimalne brzine ćelija.

4.2 Mreže sa virtuelnim kolima i mreže sa datagramima

Sećate se iz trećeg poglavlja da transportni sloj može da ponudi aplikacijama uslugu sa uspostavljanjem veze ili uslugu bez uspostavljanja veze. Na primer, transportni sloj interneta nudi aplikacijama izbor između dve usluge: UDP usluge, bez uspostavljanja veze ili TCP usluge, sa uspostavljanjem veze. Slično tome, i mrežni sloj može da ponudi uslugu bez uspostavljanja veze i uslugu sa uspostavljanjem veze između dva računara. Usluge mrežnog sloja sa uspostavljanjem i bez uspostavljanja veze imaju mnoge sličnosti sa uslugama transportnog sloja sa uspostavljanjem veze i bez uspostavljanja veze. Na primer, usluga mrežnog sloja sa uspostavljanjem veze počinje usaglašavanjem između izvornog i odredišnog računara, a usluga mrežnog sloja bez uspostavljanja veze obavlja se bez prethodnih priprema.

Iako usluge mrežnog sloja sa uspostavljanjem i bez uspostavljanja veze imaju neke sličnosti sa uslugama transportnog sloja sa uspostavljanjem veze i bez uspostavljanja veze, postoje neke bitne razlike:

- Na mrežnom sloju su to usluge od računara do računara koje mrežni sloj pruža transportnom sloju. Na transportnom sloju su to usluge od procesa do procesa koje transportni sloj pruža aplikativnom sloju.
- U svim najvažnijim arhitekturama računarskih mreža do danas (internet, ATM, frame-relay itd.) mrežni sloj nudi bilo uslugu bez uspostavljanja veze od računara do računara ili uslugu sa uspostavljanjem veze od računara do računara, ali ne i obe. Računarske mreže koje na mrežnom sloju nude samo uslugu sa uspostavljanjem veze nazivamo **mreže sa virtuelnim kolima (virtual circuit, VC)**; računarske mreže koje na mrežnom sloju nude samo uslugu bez uspostavljanja veze nazivamo **mreže sa datagramima**.
- Usluga sa uspostavljanjem veze transportnog sloja i usluga sa uspostavljanjem veze mrežnog sloja ostvaruju se bitno drugačije. U prethodnom poglavlju videli smo da se usluga sa uspostavljanjem veze transportnog sloja ostvaruje na obodu

mreže u krajnjim sistemima; uskoro ćemo videti da se usluga sa uspostavljanjem veze mrežnog sloja ostvaruje u ruterima u jezgru mreže, a isto tako i u krajnjim sistemima.

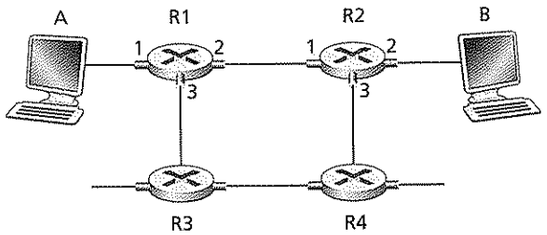
Mreže sa virtuelnim kolima i mreže sa datagramima su dve osnovne klase računarskih mreža. Za svoje odluke o prosleđivanju koriste veoma različite informacije. Pogledajmo detaljnije kako se ostvaruju.

4.2.1 Mreže sa virtuelnim kolima

Naučili smo da je internet mreža sa datagramima. Međutim, većina alternativnih mrežnih arhitektura – uključujući ATM mreže i frame relay mreže – predstavlja mreže sa virtuelnim kolima i, prema tome, koriste veze mrežnog sloja. Ove veze mrežnog sloja nazivaju se **virtuelna kola (virtual circuit, VC)**. Razmotrimo sada kako se u računarskoj mreži ostvaruje VC usluga.

Virtuelno kolo (VC) sastoji se od: (1) putanje (to jest, niza linkova i rutera) između izvornog i odredišnog računara, (2) brojeva virtuelnog kola (VC broj), po jedan broj za svaki link duž putanje i (3) stavki u tabeli prosleđivanja u svakom ruteru duž putanje. Paket koji pripada virtuelnom kolu u svom zaglavlju nosi VC broj. Pošto virtuelno kolo može da ima drugačiji VC broj na pojedinim linkovima, svaki posredni ruter mora da zameni stari VC broj u paketima koji kroz njega prolaze novim VC brojem. Novi VC broj dobija se iz tabele prosleđivanja.

Da bismo pojasnili o čemu pričamo, posmatrajmo mrežu prikazanu na slici 4.3. Brojevi pored linkova rutera R1 na slici 4.3 su brojevi interfejsa linkova. Pretpostavimo sada da računar A zatraži da mreža uspostavi VC između njega i računara B. Pretpostavimo takođe da mreža izabere putanju A-R1-R2-B i dodeli VC brojeve 12, 22 i 32 na linkovima ove putanje za ovo virtuelno kolo. U ovom slučaju, kada paket u ovom virtuelnom kolu napušta računar A, vrednost u polja VC broja u njegovom zaglavlju je 12; kada napušta ruter R1 ta vrednost je 22; a kada napušta R2 ta vrednost je 32.



Slika 4.3 ◆ Jednostavna mreža sa virtuelnim kolima

Kako ruter utvrđuje kojom bi vrednošću trebalo zameniti VC broj za paket koji prolazi kroz taj ruter? U VC mreži tabela prosleđivanja svih rutera obuhvata prevođenje VC brojeva; na primer, tabela prosleđivanja u R1 mogla bi da izgleda ovako:

Dolazni 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 prosleđivanja. Slično tome, kad god se neko virtuelno kolo prekine, uklanjaju se odgovarajuće stavke iz svih tabela duž njegove putanje.

Možda se pitate zašto paket jednostavno ne zadrži isti VC broj na svim linkovima duž svoje putanje. Postoje dva razloga. Prvo, zamenom broja od linka do linka smanjuje se potrebna dužina VC polja u zaglavlju paketa. Drugo, još važnije, uspostavljanje virtuelnog kola značajno se pojednostavljuje time što su dozvoljeni različiti VC brojevi na linkovima duž putanje virtuelnog kola. Tačnije, 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 zajednički VC broj za sve linkove duž putanje, ruteri bi morali da razmene i obrade priličan broj poruka da bi se usaglasili oko zajedničkog VC broja (odnosno, broja koji na tim ruterima trenutno ne koristi nijedno drugo virtuelno kolo) koji bi se koristio za tu vezu.

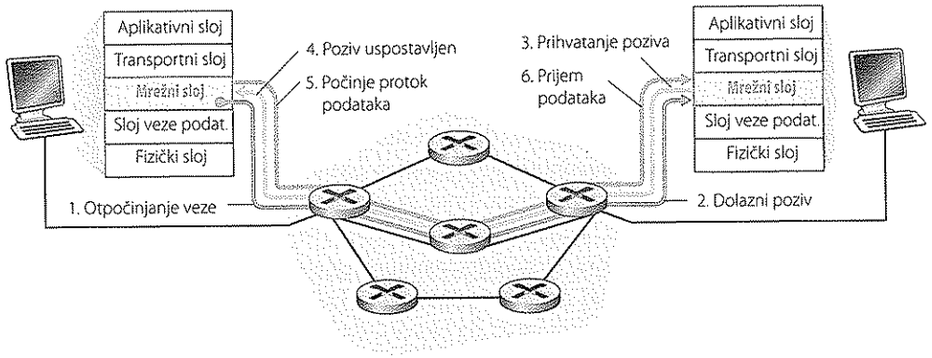
U mreži sa virtuelnim kolima, mrežni ruteri moraju da održavaju **informacije o stanju veze** za uspostavljene veze. Tačnije, kad god se uspostavi nova veza preko nekog rutera, u tabelu prosleđivanja tog rutera mora da se doda nova stavka za tu vezu; a svaki put kada se neka veza prekine iz te tabele trebalo bi izbaciti stavku za nju. Obratite pažnju na to da i kada ne bi bilo prevođenja VC brojeva, ipak je neophodno održavati informacije o stanju veze i kojima se VC brojevima pridružuju brojevi izlaznih interfejsa. Pitanje da li ruteri održavaju ili ne održavaju informacije o stanju veze za sve uspostavljene veze od presudnog je značaja – često ćemo mu se vraćati u ovoj knjizi.

U virtuelnom kolu prepoznaju se tri faze:

- *Uspostavljanje virtuelnog kola.* Tokom faze uspostavljanja, transportni sloj pošiljaoca stupa u vezu sa mrežnim slojem, navodi adresu primaoca i čeka da mreža uspostavi virtuelno kolo. Mrežni sloj utvrđuje putanju između pošiljaoca i primaoca, to jest, niz linkova i rutera kroz koje će putovati svi paketi virtuelnog kola. Mrežni sloj takođe određuje VC brojeve za sve linkove duž putanje. Na kraju, mrežni sloj dodaje odgovarajuću stavku u tabelu prosleđivanja svih rutera duž putanje. Tokom uspostavljanja virtuelnog kola, mrežni sloj takođe

može da rezerviše resurse (na primer, propusni opseg) duž putanje tog virtuelnog kola.

- *Prenos podataka.* Kao što je prikazano na slici 4.4, kada se virtuelno kolo uspostavi, kroz njega može da počne proticanje podataka.
- *Raskidanje virtuelnog kola.* Ovo otpočinje 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 prosleđivanja u svim ruterima paketa na putanji, čime se označava da to virtuelno kolo više ne postoji.



Slika 4.4 ◈ Uspostavljanje virtuelnog kola

Postoji jedna jedva primetna, ali važna razlika između uspostavljanja virtuelnog kola na mrežnom sloju i uspostavljanja veze na transportnom sloju (na primer, TCP trostruko usaglašavanje koje smo proučavali u poglavlju 3). Uspostavljanje veze na transportnom sloju tiče se samo dva krajnja sistema. Tokom uspostavljanja veze na transportnom sloju, dva krajnja sistema sami utvrđuju sve parametre (na primer, početni redni broj i veličinu prozora za kontrolu toka) veze na transportnom sloju. Iako su krajnji sistemi svesni veze transportnog sloja, ruteri unutar mreže uopšte nemaju pojma o njoj. S druge strane, na mrežnom sloju virtuelnog kola, *ruteri duž putanje između dva krajnja sistema uključeni su u uspostavljanje virtuelnog kola i svaki ruter je sasvim svestan svih virtuelnih kola koja prolaze kroz njega.*

Poruke koje krajnji sistemi šalju na mrežu za otpočinjanje ili raskidanje virtuelnog kola i poruke koje razmenjuju ruteri za uspostavljanje virtuelnog kola (odnosno, da bi promenili stanja veze u tabelama rutera) poznate su kao **signalizacione poruke**, a protokoli koji se koriste za razmenu tih poruka često se nazivaju **protokoli signalizacije**. Uspostavljanje virtuelnog kola prikazano je na slici 4.4. U ovoj knjizi

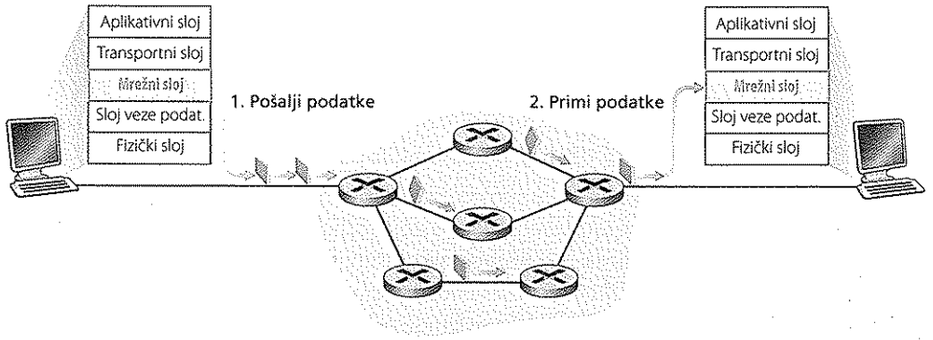
ne obrađujemo protokole signalizacije; opšti opis razmene signalizacije u mrežama sa uspostavljanjem veze naći ćete u [Black 1997], a specifikaciju protokola signalizacije ATM mreža Q.2931 u [ITU-T Q.2931 1994].

4.2.2 Mreže sa datagramima

U **mrežama sa datagramima**, kad god krajnji sistem hoće da pošalje neki paket, on taj paket obeleži adresom krajnjeg odredišnog sistema i zatim taj paket ubaci u mrežu. Kao što je prikazano na slici 4.5, to se obavlja bez uspostavljanja virtuelnog kola. Ruteri u mreži sa datagramima ne održavaju nikakve informacije o stanju virtuelnih kola (jer ovde i nema virtuelnih kola!).

Tokom prenosa od izvora do odredišta paket prolazi kroz niz rutera. Svi ovi ruteri koriste adresu odredišta u paketu za njegovo prosleđivanje. Tačnije, svaki ruter ima tabelu prosleđivanja u kojoj se adrese odredišta preslikavaju u interfejse linkova; kada paket stigne u određeni ruter, taj ruter koristi adresu odredišta paketa, da bi u tabeli prosleđivanja pronašao odgovarajući interfejs izlaznog linka. Ruter zatim prosleđuje paket na taj interfejs izlaznog linka.

Da bismo stekli bolji uvid u pretraživanje tabele, pogledajmo određen primer. Pretpostavimo da su sve adrese odredišta 32-bitne (koliko baš i jeste dužina adrese odredišta u IP datagramu). Najjednostavnije bi bilo da tabela prosleđivanja ima po jednu stavku za svaku moguću adresu odredišta. Pošto postoji više od 4 milijarde mogućih adresa, takvo rešenje nipošto ne dolazi u obzir –bila bi nam potrebna ogromna tabela prosleđivanja.



Slika 4.5 ◈ Mreža sa datagramima

Pretpostavimo zatim da naš ruter ima četiri linka, obeležena brojevima od 0 do 3, a da bi pakete trebalo prosleđivati na interfejse linkova na sledeći način:

Opseg adresa odredišta	Interfejs linka
11001000 00010111 00010000 00000000 do 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 do 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 do 11001000 00010111 00011111 11111111	2
ostalo	3

Sasvim je jasno da u ovom slučaju nije potrebno imati 4 milijarde stavki u tabeli prosleđivanja rutera. Mogli bismo, na primer, da imamo sledeću tabelu prosleđivanja sa samo četiri stavke:

Prefiks	Interfejs linka
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
ostalo	3

Sa ovako napravljenom tabelom prosleđivanja, ruter upoređuje **prefiks** adrese odredišta paketa sa stavkama u tabeli; ako postoji jednakost, ruter prosleđuje paket na link pridružen toj vrednosti. Na primer, pretpostavimo da je adresa odredišta 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 nijednoj od prve tri stavke, ruter prosleđuje paket na interfejs 3. Iako ovo zvuči prilično jednostavno, postoji nešto o čemu bi i te kako trebalo voditi računa. Možda ste primetili da je moguće da se adresa odredišta podudara sa više stavki. Na primer, prvih 24 bita adrese 11001000 00010111 00011000 10101010 podudara se sa drugom stavkom u tabeli, a prvih 21 bitova podudara se sa trećom stavkom. Kada postoji više podudarnosti, ruter primenjuje **pravilo podudaranja najdužeg prefiksa**; tj. pronalazi najdužu podudarnu stavku u tabeli i prosleđuje paket na interfejs linka koji je pridružen najdužem podudarnom prefiksu. Videćemo zgog čega se tačno ovo pravilo podudaranja najdužeg prefiksa koristi, kada budemo u odeljku 4.4 detaljnije izučavali internet adresiranje.

Iako ruteri u mrežama sa datagramima ne održavaju nikakve informacije o stanju veze, oni ipak u svojim tabelama prosleđivanja održavaju informacije o stanju prosleđivanja. Međutim, ove informacije o stanju prosleđivanja menjaju se relativno sporo. U suštini, algoritmi rutiranja ažuriraju tabele prosleđivanja u mreži sa datagramima, koji obično ažuriraju tabele prosleđivanja svakih od minut do pet minuta, otprilike. U VC mreži, tabela prosleđivanja u ruteru menja se kad god se uspostavi nova veza kroz taj ruter, ili kad god se raskine neka postojeća veza kroz njega. To se u ruteru okosnice nivoa 1 lako može dogoditi u intervalima koji se mere mikrosekundama.

Pošto se tabele prosleđivanja u mrežama sa datagramima mogu menjati u bilo kom trenutku, niz paketâ koji se pošalje od jednog krajnjeg sistema u drugi može da prođe različitim putanjama kroz mrežu i može da stigne bez reda. [Paxson 1997] i [Jaiswal 2003] predstavljaju zanimljivu studiju sa pokazateljima dobijenim merenjem o izmenjenim redosledima pristizanja paketâ i drugih fenomena u javnom internetu.

4.2.3 Poreklo VC mreža i mreža sa datagramima

Razvoj mreža sa datagramima i VC mreža odražava njihove početke. Pojam virtuelnog kola na kome se zasniva organizacija mreže potiče iz sveta telefonije u kojoj se koriste stvarna električna kola. Zbog uspostavljanja poziva i održavanja stanja svih poziva u ruterima, mreža sa virtuelnim kolima nesumnjivo je složenija od mreže sa datagramima (mada u [Molinero 2002] možete naći zanimljivo poređenje složenosti mreža sa komutacijom kanala i mreža sa komutacijom paketa). I to je nasleđe iz telefonije. U telefonskim mrežama, sama mreža je nužno morala da bude složena, jer je kao krajnje sisteme povezivala proste uređaje, kao što su telefoni sa rotacionim brojčanikom (za one koji su premladi da bi to znali, telefon sa rotacionim brojčanikom je analogni telefon bez dugmadi – ima samo kružni brojčanik).

S druge strane, internet kao mreža sa datagramima je proistekao iz potrebe za povezivanjem računara. Pošto su kao krajnje sisteme imali naprednije uređaje, arhitekte interneta su odlučile da model usluge mrežnog sloja bude što jednostavniji. Kao što smo već videli u poglavljima 2 i 3, dodatna funkcionalnost (na primer, isporuka po redosledu, pouzdan prenos podataka, kontrola zagušenja i DNS prevođenje naziva) se stoga ostvaruje na višem sloju, u krajnjim sistemima. Ovaj model predstavlja suprotnost telefonskoj mreži, pa imamo i neke zanimljive posledice.

- Dobijen je model usluge mrežnog sloja interneta koji nudi minimalne (nikakve!) garancije usluga, i stoga mrežnom sloju nameće minimalne zahteve. Ali olakšava međusobno povezivanje mreža koje koriste veoma različite tehnologije na sloju veze (na primer, satelitske linkove, Eternet, optička vlakna ili radio

linkove) i koje imaju veoma različite brzine prenosa i karakteristike gubitaka. Međusobno povezivanje IP mreža podrobno obrađujemo u odeljku 4. 4.

- Kao što smo videli u poglavlju 2, aplikacije kao što su e-pošta, veb, pa čak i usluga koja se zasniva na mrežnom sloju kao što je DNS, ostvaruju se u računarima (serverima) po obodu mreže. Mogućnost da se nova usluga doda jednostavnim priključivanjem računara na mrežu i definisanjem novog protokola aplikativnog sloja (kao što je HTTP) omogućila je da se nove aplikacije, kao što je veb, prihvate na internetu za veoma kratko vreme.

4.3 Šta se nalazi unutar rutera?

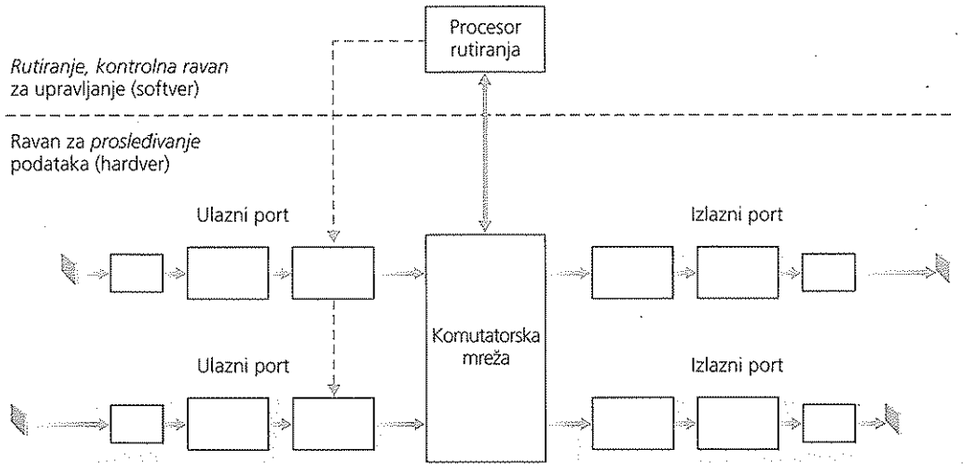
Pošto smo videli opšti pregled zadataka i usluga mrežnog sloja, obratimo pažnju na **funkciju prosleđivanja** mrežnog sloja – koju predstavlja prenošenje paketa od ulaznog linka rutera do odgovarajućeg izlaznog linka. U odeljku 4.2 već smo spomenuli nekoliko pitanja koja se tiču prosleđivanja, tačnije, adresiranje i podudaranje najdužeg prefiksa. Ovde napominjemo da istraživači i praktičari umrežavanja računara često naizmenično koriste izraze *prosleđivanje* i *komutiranje*. U ovoj knjizi koristićemo oba izraza.

Najopštiji prikaz arhitekture rutera dat je na slici 4.6. Uočavaju se četiri komponente rutera.

- *Ulazni portovi*. Ulazni port obavlja nekoliko zadataka. Obavlja funkcije fizičkog sloja povezivanjem ulaznog fizičkog linka sa ruterom. Ovo je prikazano u sasvim levom okviru ulaznog porta i sasvim desnom okviru izlaznog porta na slici 4.6. Ulazni port obavlja funkcije sloja veze podataka koje su potrebne za saradnju sa funkcijama sloja veze podataka na udaljenoj strani dolaznog linka. Ovo je prikazano srednjim okvirima u ulaznom i izlaznom portu. Možda je najznačijnija funkcija pretraživanja, koja se pojavljuje u sasvim desnom okviru ulaznog porta. Ovde se koristi tabela prosleđivanja, kako bi se utvrdio izlazni port rutera na koji će paket koji dolazi biti prosleđen putem komutatorkse mreže. Kontrolni paketi (na primer, paketi koji prenose informacije protokola rutiranja) prosleđuju se od ulaznog porta prema procesoru rutiranja. Obratite pažnju da se termin *port* koji se ovde odnosi na ulazne i izlazne fizičke interfejse rutera, značajno razlikuje od softverskih portova koji su povezani sa mrežnim aplikacijama i soketima o kojima smo pričali u poglavljima 2 i 3.
- *Komutatorska mreža*. Komutatorska mreža povezuje ulazne portove rutera sa njegovim izlaznim portovima. Čitava komutatorska mreža nalazi se unutar rutera – mreža unutar mrežnog rutera!
- *Izlazni portovi*. Izlazni port skladišti pakete koji su mu prosleđeni kroz komutatorsku mrežu, a zatim ih predaje na izlazni link izvršavanjem neophodnih funkcija sloja veze i fizičkog sloja. Ako je link dvosmeran (tj. prenosi saobraćaj u

oba smera), izlazni port tog linka obično se uparuje sa ulaznim portom tog linka na istoj linijskoj kartici (štampana ploča koja sadrži jedan ili više ulaznih portova koji su povezani na komutatorsku mrežu).

- *Procesor rutiranja*. Procesor rutiranja izvršava protokole rutiranja (na primer, protokole koje proučavamo u odeljku 4.6), održava tabele rutiranja i informacije o stanju pridruženih linkova i izračunava tabele prosleđivanja za ruter. On takođe izvršava neke funkcionalnosti upravljanja mrežom koje proučavamo u poglavlju 9.



Slika 4.6 ◆ Arhitektura rutera

Setite se da smo u poglavlju 4.1.1 pravili razliku između funkcija prosleđivanja i rutiranja. Ulazni, izlazni portovi i komutatorska mreža rutera zajedno ostvaruju funkciju prosleđivanja i skoro uvek su implementirani u hardveru, kao što je prikazano na slici 4.6. Funkcije prosleđivanja se zajedno nekada nazivaju **ravan za prosleđivanje rutera**. Da bi shvatili zašto je potrebna hardverska implementacija, uzmimo u obzir ulazni link brzine 10Gb/s i 64-bajtni IP datagram, da ulazni port ima samo 51.2 ns da obradi datagram pre nego što dođe novi datagram. Ako se N portova kombinuje na linijskoj kartici (kao što se često radi u praksi), cevovodna obrada podataka mora da radi N puta brže – da bude brza skoro kao softverska implementacija. Hardver ravni prosleđivanja može da se implementira pomoću hardverskog dizajna samog proizvođača rutera, ili može da se konstruiše pomoću kupljenih serijskih silokonskih čipova (npr. koji prodaju kompanije poput „Intel-a" i „Broadcom-a").

Dok ravan prosleđivanje funkiconiše u skali vremena koja se meri nanosekundama, kontrolne funkcije rutera, koje izvršavaju protokole rutiranja, odgovaraju na uključenja i isključenja pridruženih linkova, i izvršavaju upravljačke funkcije poput

onih o kojima govorimo u poglavlju 9, funkcionišu u vremenskom okviru milisekundi ili sekundi. Ove funkcije **kontrolne ravni rutera** su obično implementirane softverski i izvršavaju se na procesoru za rutiranje (obično tradicionalni procesor).

Pre nego što zađemo u detalje kontrole rutera i ravni sa podacima, vratimo se na analogiju iz odeljka 4.1.1, gde su paketi koji se prosleđuju bili poređeni sa automobilima koji ulaze i izlaze iz raskrsnice. Pretpostavimo da je raskrsnica kružni tok i da je potrebno malo obrade pre nego što auto uđe u taj kružni tok – auto se zaustavlja na ulaznoj stanici i pokazuje na konačno odredište (ne na lokalni kružni tok, već konačno odredište putovanja). Radnik na ulaznoj stanici traži konačno odredište, utvrđuje izlaz iz kružnog toka koji vodi do konačnog odredišta i govori vozaču gde da se isključi iz kružnog toka. Automobil ulazi u kružni tok (koji može biti prepunjen ostalim automobilima koji se uključuju sa drugih ulaznih puteva i kreću se ka drugim izlazima iz kružnog toka) i najzad, napušta kružni tok na izlaznoj rampi, gde može da se susretne sa drugim automobilima koji se isključuju na istoj rampi.

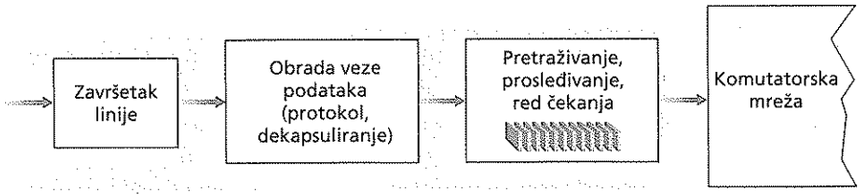
Osnovne komponente rutera iz ove analogije možemo da prepoznamo na slici 4.6 – ulazni put i ulazna stanica odgovaraju ulaznom portu (pomoću funkcije pretraživanja utvrđuje lokalni izlazni port); kružni tok se podudara sa komutatorskom mrežom, a isključenje iz kružnog toka sa izlaznim portom. Pomoću ove analogije, poučno je obratiti pažnju gde bi se moglo pojaviti usko grlo. Šta se dešava ukoliko je automobil neverovatno brz (na primer, izlazni put je u Nemačkoj ili Italiji), ali radnik na stanici spor? Koliko brz radnik mora da bude da ne dođe do gužve na ulaznom putu? Čak kada je radnik neverovatno brz, šte će se desiti ako automobili ulaze u kružni tok sporo – da li će se gužva opet pojaviti? I šta će se desiti ukoliko bi svi automobili koji ulaze u kružni tok želeli da ga napuste na istoj izlaznoj rampi – da li će se gužva pojaviti na izlaznoj rampi ili negde drugde? Kako bi kružni tok trebalo da se ponaša, ukoliko želimo da dodelimo prioritete različitim automobilima, ili blokiramo određene automobile, da ne bi uopšte ni ušli u kružni tok? Ovo su sve analogije koje se bave kritčnim pitanjima u vezi sa ruterima i programerima komutatora.

U narednim pododeljcima, detaljnije ćemo sagledati funkcije rutera. [Iyer 2008, Chao 2001; Chuang 2005; Turner 1988; McKeown 1997a; Partridge 1998] opisuju arhitekture određenih rutera. Da bi opis koji sledi bio jasniji, pretpostavimo da se mreža datagrama zasniva na odredišnoj adresi (a ne na VC broju u mreži sa virtuelnim kolima). Međutim, slična pravila i tehnike važe i za mrežu sa virtuelnim kolima.

4.3.1 Način rada ulaznog porta

Podrobniji prikaz načina na koji radi ulazni port dat je na slici 4.7. Kao što je već rečeno, funkcija završetka linije ulaznog porta i obrada sloja veze primenjuju fizičke i slojeve veze za taj pojedinačni ulazni link. Modul pretraživanja u ulaznom portu ključan je za način rada rutera – ovde ruter koristi tabelu prosleđivanja, kako bi potražio izlazni port na koji će paket koji stiže biti prosleđen putem komutatorske mreže. Tabelu prosleđivanja izračunava i ažurira procesor rutiranja, a njene kopije

obično se čuvaju u svakom ulaznom portu. Tabela prosleđivanja se kopira iz procesora rutiranja na linijske kartice preko izdvojene magistrale (npr. PCI magistrale) na koju ukazuje isprekidana linija od procesora rutiranja do ulazne linijske kartice na slici 4.6. Pomoću kopije, odluka vezana za prosleđivanje može da se donese lokalno, na svakom ulaznom portu, bez pozivanja centralizovanog procesora rutiranja za svaki paket, pa se time izbegava usko grlo centralizovane obrade.



Slika 4.7 ◆ Zadaci koje obavlja ulazni port

Za datu tabelu prosleđivanja, njeno pretraživanje je u suštini jednostavno – pregledamo tabelu prosleđivanja, tražeći stavku sa najdužim podudarnim prefiksom, kao što je opisano u odeljku 4.2.2. Ali, pri brzini prenosa u gigabitima, ovo pretraživanje mora da se izvrši u nanosekundama (setite se našeg primera od ranije i linka

ISTORIJSKA ČITANKA

CISKO SYSTEMS: DOMINACIJA JEZGROM MREŽE

Dok ovo pišemo 2012. godine „Cisco" zapošljava više od 65 000 ljudi. Kako je nastala ova ogromna kompanija koja se bavi umrežavanjem? Sve je počelo 1984. godine u dnevnoj sobi jednog stana u Silicijumskoj dolini.

Len Bosak i njegova žena Sandi Lerner radili su na Univerzitetu Stanford kada im je palo na pamet da prave rutere za internet i prodaju ih istraživačkim i akademskim ustanovama, koji su prvi u to vreme usvojili internet. Sandi Lerner je predložila ime Cisco (skraćenica za San Francisco), a takođe je napravila 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 zarada kompanije „Cisco" dostigla je 250 000 dolara mesečno. Krajem 1987. godine Cisco je konačno uspeo da privuče ulaganje kapitala – 2 miliona dolara od kompanije „Sequoia Capital" za jednu trećinu vlasništva kompanije. U sledećih nekoliko godina „Cisco" je nastavio da raste i osvajao je sve veći deo tržišta. Istovremeno, odnosi između bračnog para Bosak/Lerner i uprave kompanije „Cisco" su se kvarili. „Cisco" se pojavio na berzi 1990. godine, te iste godine Lerner i Bosak napustili su kompaniju.

Tokom godina, „Cisco" se proširio izvan tržišta rutera, prodajući proizvode i usluge koji se koriste u obezbeđenju, za bežični prenos, Eternet komutatore, infrastrukturu centara podataka, video konferencije i proizvode i usluge za prenos govora preko interneta. Međutim, „Cisco" se suočava sa sve većom međunarodnom konkurencijom, posebno kompanije „Huawei", kineske kompanije u naglom rastu koja se bavi proizvodnjom opreme za umrežavanje. Ostali konkurenti koji se nadmeću sa kompanijom „Cisco" za udeo u tržištu rutera i Eternet komutatora su „Alcatel-Lucent" i „Juniper".

brzine 10 Gb/s i 64-bajtnog IP datagrama). Prema tome, ne samo da bi pretraga trebalo da se izvrši hardverski, već su potrebne i tehnike koje prevazilaze jednostavnu linearnu pretragu kroz veliku tabelu; pregled brzih algoritama za pretraživanje može da se nađe u [Gupta 2001, Ruiz-Sanchez 2001]. Posebna pažnja bi trebalo da se obrati na vreme pristupa memorije, što dovodi do dizajna sa ugrađenim čipom DRAM i bržom SRAM (koristi se kao DRAM keš memorija) memorijom. Adresne memorije ternarnog sadržaja (TCAM, engl. Ternary Content Address Memmories) se često koriste za pretraživanje. Pomoću TCAM, 32-bitna IP adresa se predstavlja memoriji, koja vraća sadržaj tabele prosleđivanja za tu adresu za konstantno vreme. *Cisko* 8 500 ima 64K CAM za svaki ulazni port.

Kada se pomoću pretraživanja utvrdi izlazni port paketa, paket može da se pošalje u komutatorsku mrežu. Kod nekih tipova dizajna, ukoliko paketi iz drugih ulaznih portova trenutno koriste komutatorsku mrežu, može trenutno da bude blokiran ulazak paketa u komutatorsku mrežu. Blokirani paket čekaće u redu na ulaznom portu i biće raspoređen za prelazak u komutatorsku mrežu malo kasnije. U odeljku 4.3.4 ćemo se detaljnije baviti blokiranjem, čekanjem u redu i raspoređivanjem paketa (i na ulaznim i na izlaznim portovima). Iako je „pretraživanje" možda najvažnija aktivnost u obradi ulaznog porta, moraju se preduzeti i mnoge druge aktivnosti: (1) mora se obaviti obrada fizičkog i sloja veze, na prethodno opisan način; (2) broj verzije paketa, kontrolni zbir i polje za vreme života, o čemu ćemo govoriti u odeljku 4.4.1, moraju da se provere, a poslednja dva polja moraju ponovo da se upišu; i (3) brojači koji se koriste za upravljanje mrežom (poput broja primljenih IP datagrama) moraju da se ažuriraju.

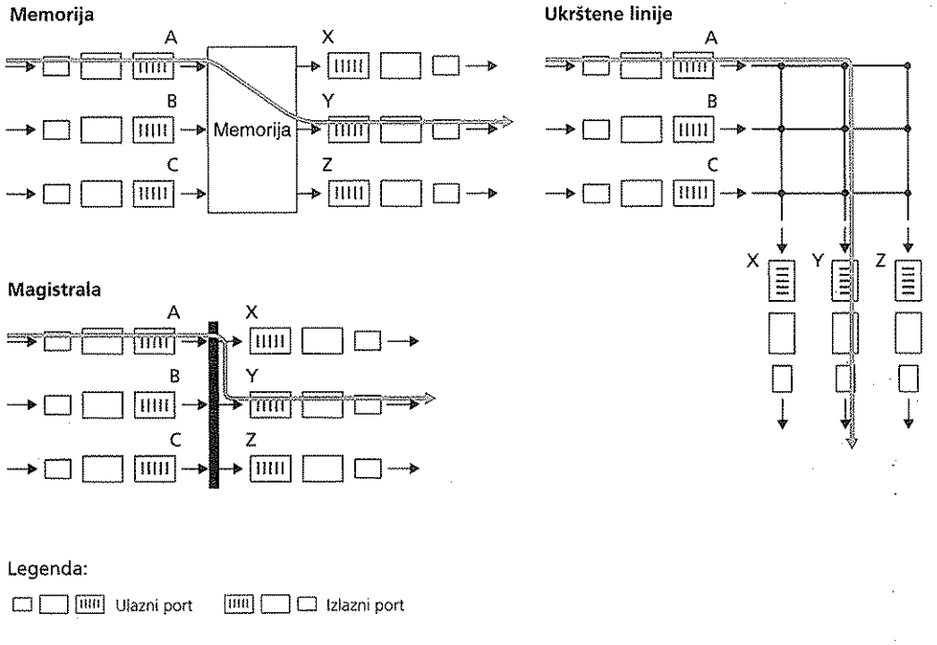
Našu diskusiju o načinu obrade ulaznog porta završićemo napomenom da ulazni port koji počinje sa pretraživanjem IP adrese („podudarnost") a zatim šalje paket u komutatorsku mrežu („aktivnost") predstavlja poseban slučaj opštijeg pojma „podudaranje plus aktivnost" koje se izvršava u mnogim mrežnim uređajima, ne samo ruterima. U komutatorima sloja veze (koji su obrađeni u poglavlju 5, odredišne adrese sloja veze su pretražene i može se preduzeti nekoliko aktivnosti pored slanja okvira u komutatorsku mrežu ka izlaznom portu. U mrežnim barijerama (o kojima se govori u poglavlju 8), uređajima koji filtriraju izabrane dolazeće pakete, može biti sprečeno prosleđivanje (aktivnost) dolazećih paketâ čije zaglavlje odgovara zadatim kriterijumima (npr. kombinacija IP adresa izvor/odredište i brojevi porta transportnog sloja). Kod prevođenja mrežnih adresa (NAT, koji se obrađuje u poglavlju 4.4) broj porta dolazećeg paketa čiji broj porta transportnog sloja odgovara zadatoj vrednosti biće pre prosleđivanja (aktivnosti) ponovo upisan. Prema tome, pojam „podudaranje plus aktivnost" je moćan i nadmoćan u mrežnim uređajima.

4.3.2 Komutiranje

Komutatorska mreža predstavlja srce rutera. Komutatorska mreža je baš ono mesto gde se paketi komutiraju (odnosno, prosleđuju) od ulaznog porta do izlaznog porta. Komutiranje se može obaviti na više načina, kao što je prikazano na slici 4.8.

◦ *Komutiranje preko memorije.* Najjednostavniji, prvobitni ruteri često su bili tradicionalni računari u kojima se komutiranje između ulaznih i izlaznih portova odvijalo pod neposrednom kontrolom procesora (procesora rutiranja). Ulazni i izlazni portovi ponašali su se kao uobičajeni U/I uređaji u običnom operativnom sistemu. Ulazni port na koji pristigne paket prvo bi o tome obavestio procesor rutiranja pomoću prekida. Paket se zatim kopira iz ulaznog porta u memoriju procesora. Posle toga, procesor rutiranja izdvaja adresu odredišta iz zaglavlja, traži odgovarajući izlazni port u tabeli prosleđivanja i kopira paket u bafer izlaznog porta. U ovom scenariju, ako je propusni opseg memorije toliki da u memoriju može da se upiše ili iz nje pročita B paketa u sekundi, tada ukupna propusna moć komutatora (ukupna brzina kojom se paketi prenose od ulaznih do izlaznih portova) mora da bude manja od $B/2$. Primetite takođe da dva paketa ne mogu da se proslede u isto vreme, čak i kada imaju različite odredišne portove, jer samo jedno upisivanje/čitanje u/iz memorije može da se izvrši preko deljene sistemske magistrale.

Mnogi savremeni ruteri takođe komutiraju preko memorije. Međutim, glavna razlika u odnosu na prvobitne rutere je u tome što traženje adrese odredišta i skladištenje paketa na odgovarajuće mesto u memoriji izvršavaju procesori na ulaznim linijskim karticama. Na neki način, ruteri koji komutiraju putem memorije veoma liče na multiprocesore sa deljenom memorijom, gde procesiranje na linijskoj kartici komutira (upisuje) pakete u memoriju odgovarajućeg izlaznog porta. *Cisco* komutatori serije *Catalyst* 8 500 [Cisco 8500 2012] prosleđuju pakete putem deljene memorije.



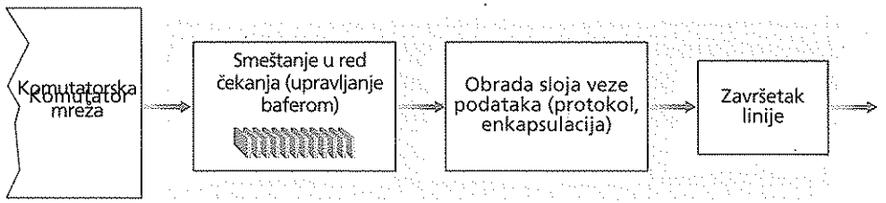
Slika 4.8 ◈ Tri tehnike komutiranja

- *Komutiranje preko magistrale.* Kod ovog rešenja, ulazni portovi prenose paket neposredno u izlazni port preko zajedničke magistrale, bez posredovanja procesora rutiranja. Ovo se najčešće vrši tako što ulazni port doda internu oznaku komutatora (zaglavlje) paketu koja ukazuje na lokalni izlazni port na koji je ovaj paket prosleđen i prosleđuje paket na magistralu. Paket primaju svi izlazni portovi, ali samo jedan od njih koji se podudara sa oznakom će zadržati paket. Oznaka se tada uklanja na izlaznom portu, jer se samo koristi unutar komutatora kako bi se prešla magistrala. Ukoliko više paketa stigne u ruter u isto vreme, svaki na različiti ulazni port, svi sem jednog moraju da čekaju, jer samo po jedan paket u jednom trenutku može da pređe magistralu. Pošto svaki paket mora da pređe jednu istu magistralu, brzina komutiranja rutera je ograničena brzinom magistrale; ako primenimo analogiju kružnog toka, to bi bilo kao kada bi u kružni tok ulazio po jedan automobil. Ipak, komutiranje putem magistrale je često dovoljno za rutere koji funkcionišu u malim lokalnim oblastima i kompanijskim mrežama. *Cisko* 5 600 [Cisco Switches 2012] komutira pakete putem magistrale od 32 Gb/s.
- *Komutiranje preko višestruko povezane mreže.* Jedan od načina da se prevaziđe ograničenje propusnog opsega samo jedne, zajedničke magistrale je korišćenje složenije, višestruko povezane mreže kakve su se ranije koristile za međusobno povezivanje procesora u višeprocesorskoj arhitekturi računara. Komutator sa unakrsnim linijama je višestruko povezana mreža koja se sastoji od $2N$ magistrala koje povezuje N ulaznih portova sa N izlaznih portova, kao što je prikazano na slici 4.8. Svaka vertikalna magistrala seče se sa svakom horizontalnom magistralom u tački preseka, koju može da otvori ili zatvori u bilo koje vreme kontroler komutatorske mreže (čija je logika deo komutatorske mreže). Kada paket stigne od porta A i kada bi trebalo da se prosledi do porta Y, kontroler komutatora zatvara tačku preseka na mestu gde se seku magistrale A i Y, a port A šalje paket na magistralu, koji može da pokupi (jedino) magistrala Y. Primetite da paket iz porta B može da se prosledi do porta X u isto vreme, jer paketi A do Y i B do X koriste različite ulazne i izlazne magistrale. Prema tome, za razliku od prethodna dva pristupa komutiranja, mreže sa unakrsnim linijama su sposobne za prosleđivanje više paketa u isto vreme. Međutim, ako su dva paketa iz dva različita ulazna porta upućena na isti izlazni port, tada će jedan od njih morati da čeka na ulaz, jer preko magistrale u dato vreme može da se pošalje samo jedan paket.

  Mnogo komplikovanije, višestruko povezane mreže, koriste komutirane elemenata u više etapa kako bi dozvolile paketima iz različitih ulaznih portova da se kreću ka istom izlaznom portu u isto vreme preko komutatorske mreže. U [Tobagi 1990] naći ćete pregled komutatorskih arhitektura. Komutatori familije *Cisco* 12 000 [Cisco 12000 2012] koriste višestruko povezanu mrežu.

4.3.3 Procesiranje izlaznog porta

Procesiranje na izlaznom portu, prikazano na slici 4.9, obuhvataju uzimanje paketa koji se čuvaju u memoriji izlaznog porta i prenošenje preko odlaznog linka. Ovo uključuje biranje i izlazak paketa iz redova za prenos, kao i izvršavanje potrebnih funkcija prenosa sloja veze i fizičkog sloja.



Slika 4.9 ◆ Zadaci koje obavlja izlazni port

4.3.4 Gde dolazi do čekanja u redu?

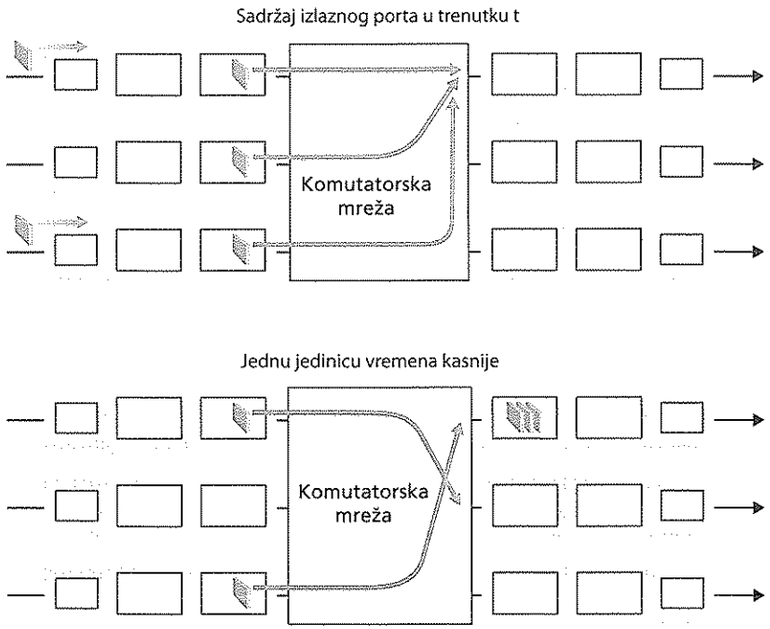
Ako pogledamo funkcionalnosti koje obavljaju ulazni i izlazni portovi i konfiguracije prikazane na slici 4.8, očigledno je da se redovi paketa mogu stvoriti kako na ulaznim tako *i* na izlaznim portovima, što možemo povezati sa slučajevima u kojima automobili mogu da čekaju na ulazak u raskrsnicu i izlazak iz raskrsnice u primeru sa kružnim tokom. Lokacija i dužina čekanja u redu (bilo na ulaznom ili izlaznom portu) zavise od opterećenosti saobraćaja, prosečne brzine komutatorske mreže i brzine linije. Posmatrajmo malo detaljnije ove redove za čekanje, jer kako ovi redovi rastu, tako se eventualno može iscrpeti memorija i pojaviće se **gubitak paketa**, kada ne bude bilo dovoljno memorije za skladištenje pristiglih paketa. Setite se da smo u ranijim objašnjenjima rekli da su paketi bili „izgubljeni unutar mreže" ili „ispušteni na ruteru". Na ovom mestu, u ovim redovima za čekanje unutar rutera će takvi paketi biti ispušteni ili izgubljeni.

Pretpostavićemo da ulazne i izlazne brzine linije (brzine prenosa) imaju identičnu brzinu prenosa R_{linija} paketa po sekundi, i da ima N ulaznih i N izlaznih portova. Da bismo pojednostavili objašnjenje, pretpostavićemo da su svi paketi iste fiksne dužine, i da stižu do ulaznih portova na sinhron način. To znači da je vreme potrebno da se pošalje paket na bilo koji link jednako vremenu potrebnom da se paket primi na bilo koji link, a tokom tog vremenskog intervala, na ulazni link stići će nula ili jedan paket. Brzinu prenosa komutatorske mreže $R_{komutator}$ definisaćemo kao brzinu kojom paketi mogu da se pomeraju od ulaznog do izlaznog porta. Ako je $R_{komutator}$ N puta brže od R_{linija}, onda će se pojaviti zanemarljivo čekanje u redu na ulaznim portovima. To je zato što će i u najgorem slučaju, kada su na svih N ulaznih linija paketi koji se primaju, a svi paketi bi trebalo da se proslede na isti izlazni port, svaka grupa od N paketa (jedan paket pod ulaznom portu) će biti propuštena kroz komutatorsku mrežu pre nego što pristigne sledeća grupa.

Ali, šta može da se desi na izlaznim portovima? Pretpostavimo da je $R_{komutator}$ i dalje N puta brže od R_{linija}. Da ponovimo, paketi koji stižu na svaki od N ulaznih portova se upućuju na isti izlazni port. U ovom slučaju, za vreme koje je potrebno da se pošalje pojedinačan paket preko izlaznog linka, N novih paketa će stići na ovaj izlazni port. S obzirom da izlazni port može da prenese samo pojedinačan paket u jedinici vremena (vreme prenosa paketa), N pristiglih paketa će morati da čekaju u redu na prenos preko izlaznog linka. Tada će možda pristići još N paketa za vreme

koje je potrebno da se prenese samo jedan od N paketa koji su prethodno bili u redu za čekanje. I tako dalje. Na kraju, broj paketa na čekanju može da poraste toliko da iscrpi raspoloživu memoriju na izlaznom portu, i u tom slučaju se paketi ispuštaju.

Čekanje u redu izlaznog porta prikazano je na slici 4.10. U trenutku t na svaki od ulaznih portova stigao je po jedan paket i svi su namenjeni za gornji izlazni port. Ako pretpostavimo da su brzine linija iste, a da komutator radi tri puta brže od linija, nakon jedne jedinice vremena (odnosno, za vreme potrebno da se primi ili pošalje jedan paket) sva tri prvobitna paketa preneta su na odlazni port i čekaju u redu na prenošenje. Tokom sledeće jedinice vremena jedan od ta tri paketa biće prenet preko odlaznog linka. U našem primeru, dva *nova* paketa su pristigla na ulaznu stranu komutatora; a jedan od njih je namenjen za gornji izlazni port.



Slika 4.10 ◈ Čekanje u redu na izlaznom portu

Pošto je bafer rutera neophodan da bi se ublažila neravnomernost opterećenosti saobraćaja, prirodno se nameće pitanje *koliko* je privremene memorije potrebno. Godinama je važilo pravilo [RFC 3439] da bi veličina bafera (B) trebalo da bude jednaka prosečnoj vrednosti vremena povratnog puta (RTT, recimo 250 ms) pomnoženoj sa kapacitetom linka (C). Do ove vrednosti došlo se analizom ponašanja paketa u redovima za čekanje sa relativno malim brojem TCP tokova [Villamizar 1994]. Stoga bi za link propusnog opsega 10 Gb/s sa vrednošću RTT od 250 msec bio potreban bafer od $B = RTT \cdot C = 2{,}5$ Gb. Skorašnja teoretska i eksperimentalna istraživanja [Appenzeller 2004], međutim, pokazuju da je u slučaju kada

kroz neki link prolazi veliki broj TCP tokova (N) neophodna količina bafera od $B = RTT \cdot C/\sqrt{N}$. U slučajevima kada veliki broj tokova prolazi kroz linkove rutera velikih okosnica interneta (pogledajte npr. [Fraleigh 2003]), vrednost N je velika zbog čega je potrebna veličina bafera značajna. [Appenzeller 2004; Wischik 2005; Beheshti 2008] nude izuzetno čitku raspravu o potrebnoj veličini bafera sa teoretskog stanovišta, sa osvrtom na praktičnu primenljivost i ostvarljivost.

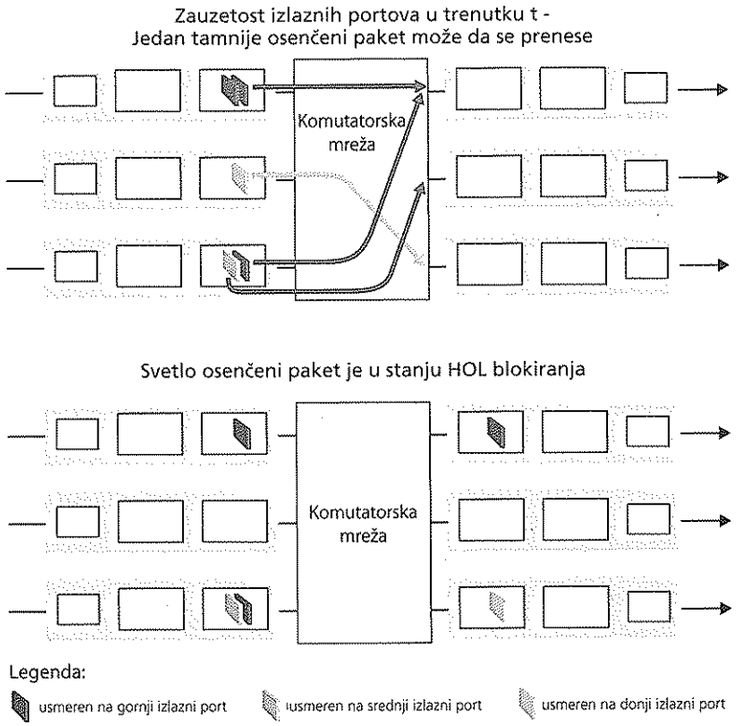
Posledica čekanja u redu izlaznog porta je da **raspoređivač paketa** (engl. *packet scheduler*) na izlaznom portu mora da izabere jedan paket između onih koji čekaju u redu za prenos. Odabir može da se obavlja krajnje jednostavno, kao što je po redosledu pristizanja (FCFS, od first-come-first-served – prvopristigli se prvo uslužuje), ili po mnogo složenijem raspoređivanju, kao što je težinsko ravnopravno čekanje (weighted fair queuing, WFQ) u kome se izlazni link ravnopravno deli između različitih veza sa kraja na kraj čiji paketi čekaju na prenos. Raspoređivanje paketâ veoma je važno da bi se obezbedio **garantovan kvalitet usluge**. Ova tema biće opširno obrađena u poglavlju 7. Razmatranje tehnika koje se koriste za raspoređivanje paketa na izlaznom portu možete naći u [Cisco Queue 2012].

Slično tome, ako nema dovoljno memorije da se ulazni paket smesti u bafer, potrebno je doneti odluku da li da se odbaci paket koji upravo pristiže (postupak poznat kao **odbacivanje začelja**), ili da se ukloni jedan paket, ili više njih iz reda čekanja, i napravi mesto za novopristigli paket. U nekim slučajevima korisnije je odbaciti paket (ili staviti oznaku u njegovo zaglavlje) *pre* nego što se bafer napuni, kako bi se pošiljaocu signaliziralo zagušenje. U [Labrador 1999, Hollot 2002] predlaže se i analizira čitav niz različitih postupaka za odbacivanje i označavanje paketa (koji su postali poznati pod zajedničkim nazivom algoritmi za **aktivno upravljanje redovima čekanja** (engl. *active queue management*, **AQM**). Jedan od najviše proučavanih i najčešće primenjivanih AQM algoritama je algoritam **RED** (engl. *Random Early Detection* – **nasumično rano otkrivanje**). RED algoritam se zasniva na izračunavanju težinskog proseka dužine izlaznog reda čekanja. Ako je u trenutku pristizanja nekog paketa prosečna dužina reda za čekanje manja od minimalne vrednostipraga, min_{th}, taj paket se prima u red za čekanje. Obrnuto, ako je u trenutku pristizanja paketa red za čekanje pun ili je prosečna dužina reda veća od maksimalne vrednosti praga, max_{th}, taj paket se označava ili odbacuje. I konačno, ako paket pristigne i zatekne prosečnu dužinu reda u opsegu $[min_{th}, max_{th}]$ označava se ili odbacuje sa verovatnoćom koja se određuje u zavisnosti od prosečne dužine reda i vrednosti min_{th} i max_{th}. Predloženo je više različitih funkcija kojima se izračunava verovatnoća za označavanje ili odbacivanje paketa, a postoji više analitičkih modela, simulacija i verzija algoritma RED. U [Christiansen 2001] i [Floyd 2012] dati su prikazi o tome kao i predlozi za dodatnu literaturu.

Ako komutatorska mreža nije dovoljno brza (u odnosu na brzinu ulazne linije) za prenošenje *svih* pristiglih paketâ bez kašnjenja kroz komutatorsku mrežu tada nastaju redovi za čekanje i na ulaznim portovima, pošto paketi moraju da sačekaju da budu preneti kroz komutatorsku mrežu do izlaznog porta. Da bismo bolje prikazali vrlo važnu posledicu ovog čekanja u redu, posmatrajmo komutatorsku, višestruko

povezanu mrežu i pretpostavimo: (1) da su brzine svih linkova iste, (2) da se jedan paket može preneti od bilo kog ulaznog porta do datog izlaznog porta za isto vreme koje je potrebno da se paket primi na ulaznom linku i (3) da se paketi premeštaju iz ulaznog reda za čekanje u željeni izlazni red za čekanje po redosledu pristizanja (FCFS). Moguće je istovremeno prenositi više paketa, samo ako se njihovi izlazni portovi razlikuju. Međutim, ukoliko su dva paketa koji se nalaze na čelu dva ulazna reda za čekanje namenjena za isti izlazni red za čekanje, jedan od njih se zaustavlja i mora da sačeka u ulaznom redu za čekanje – komutatorska mreža istovremeno može da prenosi samo jedan paket do datog izlaznog porta.

Na slici 4.11 dat je primer u kojem su dva paketa (tamnije osenčena) na čelu svojih ulaznih redova za čekanje namenjena za isti, gornji desni izlazni port. Pretpostavimo da komutatorska mreža za prenošenje izabere paket sa čela gornjeg levog reda za čekanje. U tom slučaju tamnije osenčeni paket u donjem levom redu mora da sačeka. Ali, ne samo što ovaj tamnije osenčeni paket mora da čeka, već mora da čeka i svetlije osenčeni paket koji se u donjem levom redu za čekanje nalazi iza njega, iako *nema* drugih paketâ koji su upućeni na srednji desni izlazni port (odredište svetlije osenčenog paketa). Ova pojava poznata je kao **blokiranje spreda** (engl. *head-of-the-line (HOL) blocking*) u ulaznom redu za čekanje komutatora – paket koji čeka u ulaznom redu mora da sačeka prenošenje kroz komutatorsku mrežu (mada je njegov izlazni port slobodan) jer je sprečen drugim paketom koji se nalazi ispred njega. [Karol 1987] pokazuje da zbog HOL blokiranja ulazni red za čekanje neograničeno raste (što je isto kao da se kaže da dolazi do značajnog gubitka paketa) pod određenim uslovima, čim brzina pristizanja paketa na ulazne linkove dostigne samo 58 procenata njihovog kapaciteta. U [McKeown 1997b] opisuje se više rešenja za HOL blokiranje.



Slika 4.11 ◆ HOL blokiranje u ulaznom redu komutatora.

4.3.5 Ravan kontrole rutiranja

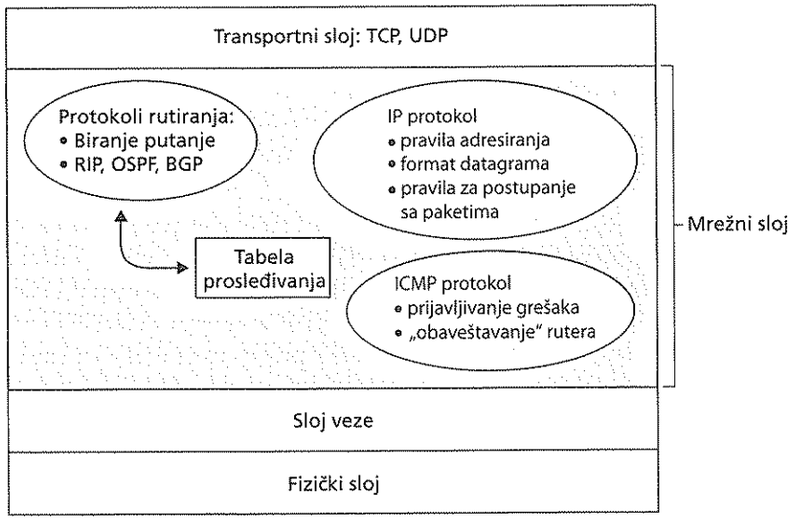
Do sada smo u našim objašnjenjima, kao i na slici 4.6, indirektno pretpostavljali da se ravan kontrole rutiranja u celosti nalazi i izvršava u procesoru rutiranja unutar rutera. Ravan kontrole rutiranja je širom mreže pa je, prema tome, decentralizovana – sa različitim delovima (npr. algoritma rutiranja) koji se izvršavaju na različitim ruterima i međusobno komuniciraju slanjem kontrolnih poruka jedni drugima. Zaista, današnji ruteri interneta i algoritmi rutiranja koje ćemo opisati u odeljku 4.6 funkcionišu upravo na ovaj način. Pri tome, proizvođači rutera i komutatora kompletiraju svoje hardverske ravni podataka sa softverskom kontrolnom ravni u zatvorene (ali koje mogu međusobno da sarađuju) platforme u vertikalno integrisan proizvod.

Nedavno su brojni istraživači [Caesar 2005a, Casado 2009, McKeown 2008] istraživali nove arhitekture kontrolne ravni rutera u kojoj se deo kontrolne ravni implementira u ruterima duž ravni podataka (npr. lokalna merenja/izveštavanje o stanju veze, instalacija tabele prosleđivanja i održavanje), a deo kontrolne ravni može da se implementira eksterno na ruter (npr. na centralizovani server, koji može da izvrši izračunavanje putanje). Dobro definisani interfejs za programiranje aplikacija (API) propisuje kako će ovi delovi međusobno delovati i komunicirati međusobno. Ovi istraživači diskutuju da razdvajanje softverske kontrolne ravni od hardverske ravni sa podacima (sa minimalnom kontrolnom ravni koja je na ruteru, to jestruter-rezident) može pojednostaviti rutiranje zamenom distribuiranog izračunavanja rutiranja centralizovanim izračunavanjem rutiranja, a omogućava inovacije mreže tako što dozvoljava različite prilagođene kontrolne ravni, koje će se izvršavati na bržim hardverskim ravnima podataka.

4.4 Internet protokol (IP): prosleđivanje i adresiranje na internetu

Do sada smo prosleđivanje i adresiranje mrežnog sloja opisivali posebno, ne pominjući nijednu određenu računarsku mrežu. U ovom odeljku usredsredićemo se na to kako se prosleđivanje i adresiranje vrše na internetu. Videćemo da su prosleđivanje i adresiranje značajne komponente protokola internet protokola IP (Internet

Protocol). Trenutno se koriste dve verzije protokola IP. Prvo ćemo ispitati opšteprihvaćenu verziju 4 internet protokola, poznatu kao IPv4 [RFC 791]. Na kraju odeljka ispitaćemo verziju 6 protokola IP [RFC 2460; RFC 4291] koja je predložena da zameni IPv4.



Slika 4.11 ◈ Pogled unutar mrežnog sloja interneta

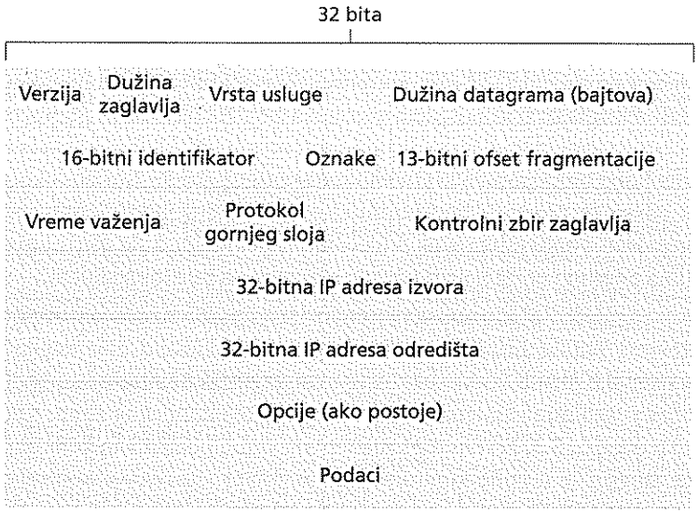
Ali, pre nego što krenemo u pohod na IP, vratimo se za jedan korak i razmotrimo komponente od kojih se sastoji mrežni sloj interneta. Kao što je prikazano na slici 4.12, mrežni sloj interneta ima tri glavne komponente. Prva komponenta je protokol IP, tema ovog odeljka. Druga važna komponenta mrežnog sloja je komponenta za rutiranje koja utvrđuje putanju kojom datagram putuje od izvora do odredišta. Spomenuli smo ranije da protokoli rutiranja izračunavaju tabele prosleđivanja koje se koriste za prosleđivanje paketa kroz mrežu. U odeljku 4.6 proučavamo protokole rutiranja na internetu. Poslednja komponenta mrežnog sloja omogućava slanje obaveštenja o greškama u datagramima i pružanje izvesnih informacija o mrežnom sloju. U odeljku 4.4.3 obrađujemo internet protokol ICMP (Internet Control Message Protocol – internet protokol za kontrolne poruke) koji se koristi za obavešetavanje o greškama i davanje informacija o mrežnom sloju interneta.

4.4.1 Format datagrama

Sećate se da smo paket mrežnog sloja nazvali *datagram*. Proučavanje protokola IP počinjemo pregledom sintakse i semantike datagrama protokola IPv4. Možda mislite da nema ničeg suvoparnijeg od sintakse i semantike bitova nekog paketa. Ipak, datagram igra centralnu ulogu na internetu – neophodno je da ga svaki student ili profesionalac u umrežavanju pažljivo sagleda, dobro se upozna i ovlada njime.

Format datagrama protokola IPv4 prikazan je na slici 4.13. Ključna polja datagrama protokola IPv4 su sledeća:

- *Broj verzije*. Ova četiri bita navode verziju protokola IP određenog datagrama. Prema broju verzije ruter određuje kako bi trebalo da protumači ostatak IP datagrama. Različite verzije protokola IP koriste različite formate datagrama. Format datagrama za trenutnu verziju protokola IP, IPv4, prikazan je na slici 4.13. Format datagrama za novu verziju protokola IP (IPv6) opisan je pri kraju ovog odeljka.
- *Dužina zaglavlja*. Pošto datagram IPv4 može da sadrži promenljivi broj opcija (koje se nalaze u zaglavlju datagrama IPv4) ova četiri bita su potrebna da bi se odredilo mesto gde u IP datagramu počinju sami podaci. Većina IP datagrama ne sadrži opcije, tako da uobičajeni IP datagram ima zaglavlje od 20 bajtova.



Slika 4.13 ◈ Format datagrama protokola IPv4

- *Vrsta usluge*. Bitovi za vrstu usluge (engl. *type of service*, TOS) uključeni su u IPv4 zaglavlje da bi se omogućilo raspoznavanje između različitih vrsta IP datagrama (na primer, datagrami koji posebno zahtevaju manje kašnjenje, veću propusnu moć ili pouzdanost). Na primer, moglo bi da bude korisno razlikovati datagrame za komunikaciju u realnom vremenu (na primer, one koji se koriste za neke aplikacije za IP telefoniranje) od saobraćaja koji se ne odvija u realnom vremenu (na primer, FTP). Kakav se tačan nivo usluga nudi, zavisi od pravila koja postavlja administrator rutera. Temu koja se tiče diferenciranja usluga podrobno obrađujemo u poglavlju 7.

- *Dužina datagrama.* Ovo je ukupna dužina IP datagrama (zaglavlje plus podaci), merena u bajtovima. S obzirom da je ovo polje dužine 16 bitova, teoretski maksimalna veličina IP datagrama je 65.535 bajtova. Međutim, retko kad su datagrami duži od 1.500 bajtova.
- *Identifikator, oznake i pomeraj fragmentacije.* Ova tri polja tiču se takozvane IP fragmentacije, teme koju uskoro podrobnije obrađujemo. Zanimljivo je da nova verzija protokola IP, IPv6, ne dozvoljava fragmentaciju na ruterima.
- *Vreme života.* Polje za vreme života (eng. time-to-live, TTL) služi da bi se sprečilo večno kruženje datagrama u mreži (na primer, zbog dugotrajne petlje rutiranja). Vrednost ovog polja smanjuje se za jedan, nasvakom na ruteru gde se datagram obrađuje. Ako polje TTL dostigne vrednost 0, odgovarajući datagram mora da se odbaci.
- *Protokol.* Ovo polje se koristi samo kada IP datagram stigne do svog konačnog odredišta. Vrednost ovog polja ukazuje na to kom protokolu transportnog sloja bi trebalo predati podatke iz IP datagrama. Na primer, vrednost 6 znači da se podaci predaju protokolu TCP, dok vrednost 17 znači da se podaci predaju protokolu UDP. Za spisak svih mogućih vrednosti pogledajte [IANA Protocol Numbers 2012]. Obratite pažnju na to da broj protokola u IP datagramu ima istu ulogu koju ima polje broja porta u segmentu transportnog sloja. Broj protokola je lepak koji povezuje mrežni i transportni sloj, dok je broj porta lepak koji povezuje transportni i aplikativni sloj. Videćemo u poglavlju 5 da okvir sloja veze takođe ima posebno polje koje povezuje sloj veze sa mrežnim slojem.
- *Kontrolni zbir zaglavlja.* Kontrolni zbir zaglavlja pomaže ruteru da otkrije greške u bitovima unutar primljenog IP datagrama. Kontrolni zbir zaglavlja izračunava se tako što se svaka dva bajta u zaglavlju posmatraju kao broj, potom se ti brojevi sabiraju i izračunava prvi komplement dobijenog zbira. Kao što je opisano u odeljku 3.3, prvi komplement ovog zbira, poznat kao kontrolni zbir interneta, smešta se u polje kontrolnog zbira. Ruter izračunava kontrolni zbir zaglavlja za sve primljene IP datagrame i otkriva grešku, ako kontrolni zbir u zaglavlju datagrama nije jednak izračunatom kontrolnom zbiru. Ruteri obično odbacuju datagrame u kojima otkriju grešku. Obratite pažnju na to da se na svakom ruteru kontrolni zbir mora ponovo izračunati i sačuvati ponovo jer se polje TTL, kao i polje mogućih opcija, može promeniti. Zanimljivo razmatranje brzih algoritama za izračunavanje kontrolnog zbira interneta nalazi se u [RFC 1071]. Ovde se često postavlja pitanje zbog čega TCP/IP vrši proveravanje grešaka i na transportnom i na mrežnom sloju? Puno je razloga za to ponavljanje. Prvo, vodite računa o tome da se na IP sloju kontrolni zbir računa samo za zaglavlje protokola IP, dok se kontrolni zbir protokola TCP/UDP izračunava za čitav TCP ili UDP segment. Drugo, protokoli TCP i UDP i protokol IP ne moraju da pripadaju istom skupu protokola. TCP može, u opštem slučaju, da se izvršava i preko drugih protokola (na primer, protokola ATM mreže), a IP može da prenosi podatke koji neće pripadati protokolima TCP/UDP.

- *IP adrese izvora i odredišta.* Prilikom izrade datagrama izvor umeće svoju IP adresu u polje IP adrese izvora, a adresu krajnjeg odredišta u polje IP adrese odredišta. Često, izvorni računar pretraživanjem utvrđuje adresu odredišta DNS, kao što je opisano u poglavlju 2. IP adresiranje podrobno obrađujemo u odeljku 4.4.2.
- *Opcije.* Polja opcija omogućavaju proširivanje IP zaglavlja. Zamišljeno je da se opcije zaglavlja koriste retko – otud i odluka da se smanji opterećenje, tako što se informacije u poljima opcija ne uključuju u zaglavlje svih datagrama. Međutim, već samo postojanje opcija stvara teškoće – s obzirom da zaglavlja datagrama mogu da budu promenljive dužine, ne može se unapred utvrditi gde počinje polje sa podacima. Osim toga, pošto neki datagrami zahtevaju obradu opcija a drugi ne, vreme potrebno za obradu IP datagrama u ruteru može biti veoma različito. Sve ovo postaje posebno značajno 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 (korisno opterećenje).* Konačno, stižemo do poslednjeg i najvažnijeg polja – *razloga* zbog čega datagram uopšte i postoji! U većini slučajeva, polje podataka u IP datagramu sadrži segment transportnog sloja (TCP ili UDP) koji bi trebalo isporučiti na odredište. Međutim, polje podataka može da sadrži i druge vrste podataka, kao što su ICMP poruke (koje razmatramo u odeljku 4.4.3).

Obratite pažnju na to da zaglavlje IP datagrama ima ukupno 20 bajtova (pod pretpostavkom da nema opcija). Ako datagram prenosi TCP segment, tada svaki (nefragmentirani) datagram u svom zaglavlju prenosi ukupno 40 bajtova (20 bajtova IP zaglavlja i 20 bajtova TCP zaglavlja) pored poruke aplikativnog sloja.

Fragmentacija IP datagrama

Videćemo u poglavlju 5 da svi protokoli sloja veze ne mogu da prenose pakete iste veličine. Neki protokoli mogu da prenose ogromne pakete, dok drugi protokoli mogu da prenose samo manje pakete. Na primer, Eternet okviri ne mogu da prenose više od 1 500 bajtova podataka, dok okviri linkova nekih široko pojasnih mreža ne mogu da prenose više od 576 bajtova. Maksimalna količina podataka koju može da prenese okvir sloja veze naziva se maksimalna jedinica prenosa (maximum transfer unit, MTU). S obzirom da se za prenos od jednog rutera do sledećeg rutera svaki IP datagram enkapsulira unutar okvira sloja veze, MTU protokola sloja veze predstavlja strogo ograničenje za dužinu IP datagrama. Stroga, granica za veličinu IP datagrama nije veliki problem. Problem je što linkovi duž putanje između pošiljaoca i odredišta mogu da koriste različite protokole sloja veze, a pri tome svaki od tih protokola može da ima drugačiji MTU.

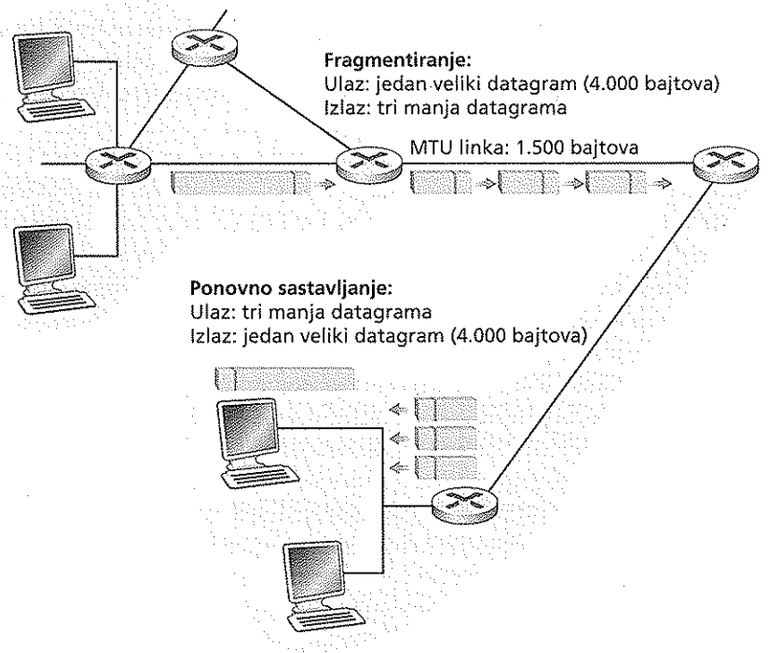
Da biste bolje razumeli ovaj problem, zamislite da ste *vi* jedan ruter koji povezuje nekoliko linkova od kojih svaki izvršava drugačiji protokol sloja veze sa različitim MTU vrednostima. Pretpostavimo da primite IP datagram sa jednog linka. Proveravate svoju tabelu prosleđivanja da biste odredili izlazni link i ispostavlja se da taj izlazni link ima MTU vrednost manju od dužine tog IP datagrama. Nastupa

panika – kako da ugurate veliki IP paket u polje podataka unutar okvira sloja veze? Rešenje je da podatke IP datagrama podelite na dva ili više manjih IP datagrama, enkapsulirate sve te manje IP datagrame u zasebne okvire sloja veze i da te okvire pošaljete preko izlaznog linka. Ovi manji datagrami nazivaju se **fragmenti**.

Fragmenti moraju ponovo da se sastave pre nego što se predaju transportnom sloju na odredištu. Zaista, i TCP i UDP od mrežnog sloja očekuju da prime čitave, nefragmentirane segmente. Projektanti protokola IPv4 su smatrali da bi ponovno sastavljanje datagrama u ruterima dovelo do toga da protokol bude složeniji i da bi to umanjilo performanse rutera. (Da ste ruter, da li biste pored svog ostalog posla koji imate voleli još i da sastavljate fragmente?) Držeći se pravila da bi jezgro mreže trebalo da ostane što jednostavnije, projektanti protokola IPv4 odlučili su da zadatak ponovnog sastavljanja datagrama prepuste krajnjim sistemima, a ne mrežnim ruterima.

Kada odredišni računar primi niz datagrama iz 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 da primljene fragmente ponovo sastavi, da bi se dobio prvobitni datagram. Da bi odredišni računar mogao da obavi te zadatke ponovnog sastavljanja, projektanti protokola IP (verzije 4) su u zaglavlje IP datagrama umetnuli polja za *identifikaciju*, *oznake* i *pomeraj fragmentacije*. Prilikom kreiranja datagrama predajni računar obeležava datagram identifikacionim brojem, kao i adresama izvora i odredišta. Obično, predajni računar povećava identifikacioni broj za svaki datagram koji pošalje. Kada ruter mora da vrši fragmentaciju datagrama, svi tako nastali datagrami (odnosno, fragmenti) označavaju se adresom izvora, adresom odredišta i identifikacionim brojem prvobitnog datagrama. Kada odredišni računar primi niz datagrama od istog predajnog računara, on ispituje identifikacione brojeve datagrama i utvrđuje koji su datagrami u stvari fragmenti jednog, istog većeg datagrama. Pošto je IP nepouzdana usluga, jedan ili više fragmenata možda nikada ne stignu na odredište. Zbog toga, da bi odredišni računar bio potpuno siguran da je primio poslednji fragment prvobitnog datagrama, bit oznake u poslednjem fragmentu postavljen je na 0, dok su bitovi oznake u svim ostalim fragmentima postavljeni na 1. Osim toga, da bi odredišni računar mogao da utvrdi da li neki fragment nedostaje (i da može ponovo da sastavi fragmente po pravilnom redosledu), koristi se polje pomeraja, kako bi se odredilo gde se odgovarajući fragment nalazi unutar prvobitnog IP datagrama.

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 sa MTU vrednošću od 1 500 bajtova. To znači da bi 3 980 bajtova podataka iz prvobitnog datagrama trebalo raspodeliti u tri zasebna fragmenta (svi oni su takođe IP datagrami). Pretpostavimo da je prvobitni datagram obeležen identifikacionim brojem 777. Karakteristike ova tri fragmenta prikazane su u tabeli 4.2. Vrednosti u tabeli 4.2 odražavaju zahtev da deo prvobitnih korisnih podataka u svim fragmentima osim poslednjeg bude deljiv sa 8, a da vrednost pomeraja bude data u odsečcima od po 8 bajtova.



Slika 4.14 ◆ Fragmentacija i ponovno sastavljanje IP datagrama

Korisni podaci iz datagrama predaju se transportnom sloju na odredištu, tek pošto IP sloj ponovo potpuno sastavi prvobitni IP datagram. Ako jedan ili više fragmenata ne stignu do odredišta, nepotpuni datagram se odbacuje i ne predaje transportnom sloju. Ali, kao što smo naučili u prethodnom poglavlju, ako se na transportnom sloju koristi TCP, tada će se TCP oporaviti od ovog gubitka, tako što će izvor primorati da ponovo pošalje podatke iz prvobitnog datagrama.

Fragment	Bajtovi	Identifikacija	Pomeraj	Oznaka
Prvi fragment	1, 480 bajtova u polju podataka IP datagrama	Identifikacija = 777	Pomeraj = 0 (znači da bi podatke trebalo umetnuti od bajta 0)	Oznaka = 1 (što znači da ima još fragmenata)
Drugi fragment	1, 480 bajtova podataka	Identifikacija = 777	Pomeraj = 185(znači da bi podaci trebalo da budu umetnuti, počevši od bajta 1. 480. Obratite pažnju da je 185 · 8 = 1. 480)	Oznaka = 1 (što znači da ima još fragmenata)
Treći fragment	1, 020 bajtova (= 3, 980 - 1, 480 -1,480) podataka	Identifikacija = 777	Pomeraj = 370 (znači da bi podaci trebalo da budu umetnuti, počevši od bajta 2, 960. Obratite pažnju da je 370· 8 = 2, 960)	Oznaka = 0 (što znači značithis da je ovo posledniji fragmnet)

Tabela 4.2 ◆ IP fragmenti

Upravo smo saznali da IP fragmentacija igra važnu ulogu u povezivanju više raznovrsnih tehnologija sloja veze. Ali, fragmentacija ima i svoju cenu. Prvo, ruteri i krajnji sistemi su mnogo složeniji, s obzirom da ih je potrebno projektovati tako da mogu da obavljaju fragmentaciju i ponovno sastavljanje datagrama. Drugo, fragmentacija se može iskoristiti za izuzetno opasan DoS napad u kome napadač šalje čitav niz besmislenih i neočekivanih fragmenata. Klasičan primer je tzv. Jolt2 napad u kome napadač šalje veliki broj malih fragmenata na ciljani računar, pri kome nijedan od njih nema pomeraj fragmentiranja postavljen na nulu. Ciljani računar može da padne, pokušavajući ponovo da sastavi datagram od neispravnih paketa. Za drugu vrstu napada šalju se preklopljeni IP fragmenti, odnosno, fragmenti čija je vrednost pomeraja postavljena tako da se fragmenti ne mogu valjano spakovati. Napadnuti operativni sistemi, ne znajući šta da rade sa preklopljenim fragmentima, mogu da se sruše [Skoudis 2006]. Kao što ćemo videti pri kraju ovog odeljka, nova verzija protokola IP, IPv6, uopšte ne koristi fragmentaciju, na taj način pojednostavljujući obradu IP paketa i čineći IP manje podložan napadima.

Na veb lokaciji ove knjige pripremili smo *Java* aplet koji pravi fragmente. Zadajete veličinu pristiglog datagrama, MTU i identifikaciju pristiglog datagrama. Aplet automatski generiše fragmente. Potražite adresu http://www.awl.com/kurose-ross.

4.4.2 IPv4 adresiranje

Sada svoju pažnju usmeravamo na adresiranje protokola IPv4. Mada adresiranje može da izgleda kao jednostavna tema, nadamo se da ćete do kraja ovog poglavlja uvideti da adresiranje na internetu, ne samo da je sočna, tanana i zanimljiva tema, već i tema koja je suštinski važna za internet. IPv4 adresiranje je odlično obrađeno u [3Com Addressing 2012] i prvom poglavlju knjige [Stewart 1999].

Međutim, pre nego što pređemo na IP adresiranje, moramo nešto reći o tome kako su računari i ruteri povezani u mrežu. Računar obično ima samo jedan link prema mreži; kada IP u tom računaru želi da pošalje datagram, on to radi preko tog linka. Granica između računara i fizičkog linka naziva se **interfejs**. Posmatrajmo sada ruter i njegove interfejse. Pošto je zadatak rutera da primi datagram sa jednog linka i prosledi ga na neki drugi link, ruter obavezno mora da bude povezan sa najmanje dva linka. Granica između rutera i bilo kojeg od njegovih linkova takođe se naziva interfejs. Stoga ruter ima više interfejsa, po jedan za svaki link. Pošto svi računari i ruteri mogu da šalju i primaju IP datagrame, IP zahteva da svi interfejsi računara i rutera imaju sopstvenu IP adresu. Prema tome, IP adresa je tehnički pridružena interfejsu, a ne računaru ili ruteru na kome se nalazi taj interfejs.

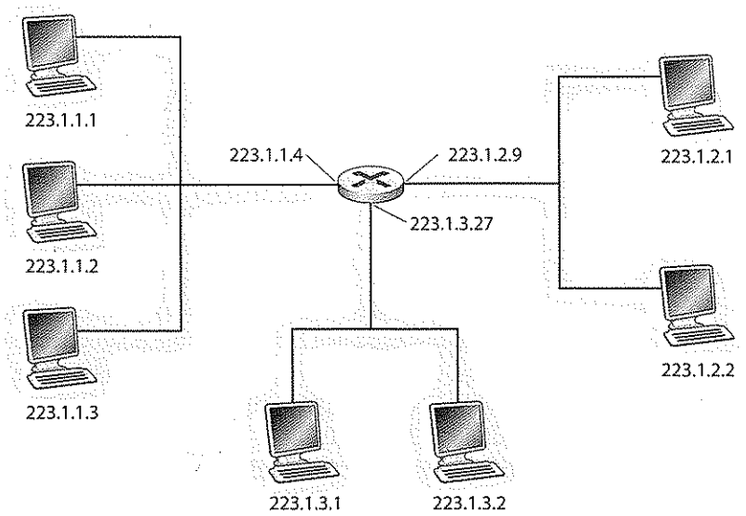
Svaka IP adresa dugačka je 32 bita (što odgovara četiri bajta), pa tako postoji ukupno 2^{32} mogućih IP adresa. Aproksimacijom 2^{10} sa 10^3, jasno se vidi da postoji oko 4 milijarde mogućih IP adresa. Te adrese se obično pišu u takozvanoj **decimalnoj notaciji sa tačkama**, u kojoj se svaki bajt adrese zapisuje u decimalnom obliku, a od ostalih bajtova u adresi razdvaja se tačkama. Uzmimo za 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 je

11000001 00100000 11011000 00001001

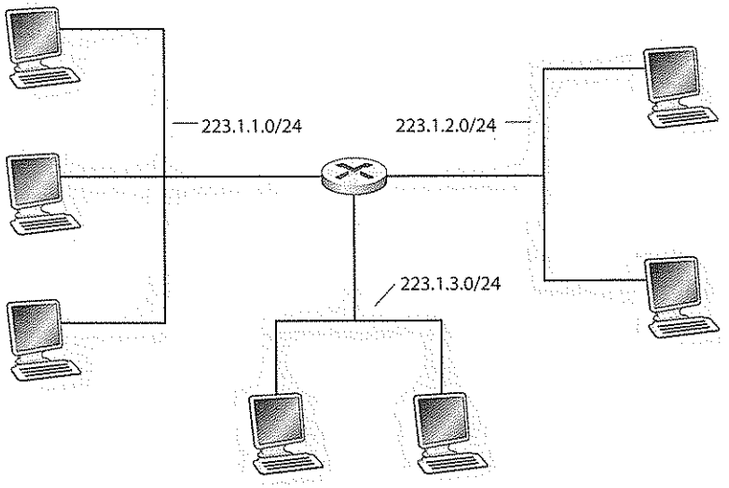
Svaki interfejs na svakom računaru i na ruteru na globalnom internetu mora da ima IP adresu koja je globalno jedinstvena (osim interfejsa ispred kojih se koristi NAT prevođenje adresa koje opisujemo na kraju ovog odeljka). Ove adrese, međutim, ne mogu da se biraju proizvoljno, kako kome padne na pamet. Deo IP adrese interfejsa određuje podmreža sa kojom je povezan.

Na slici 4.15 imamo primer IP adresiranja i interfejsa. Na ovoj slici, jedan ruter (sa tri interfejsa) koristi se za povezivanje sedam računara. Pogledajte bolje IP adrese pridružene interfejsima računara i rutera; potrebno je da obratite pažnju na nekoliko stvari. Tri računara u gornjem levom delu slike 4.15 i interfejs rutera sa kojim su povezani imaju IP adrese u obliku 223.1.1.xxx. To jest, svi oni imaju ista krajnje leva 24 bita u svojim IP adresama. Ova četiri interfejsa takođe su međusobno povezana mrežom *bez rutera*. Ta mreža bi mogla da bude, na primer, Eternet LAN, u kom slučaju bi interfejsi bili povezani Eternet komutatorom (što ćemo objasniti u poglavlju 5.) ili bežičnom pristupnom tačkom (što razmatramo u pogavlju 6). Ovu mrežu bez rutera koja za sada povezuje ove računare predstavljamo kao oblak, a u unutrašnjost ovih mreža zadiremo u poglavljima 5 i 6.



Slika 4.15 ◆ Adrese interfejsa i podmreže

U IP žargonu, ova mreža koja povezuje interfejse tri računara i interfejs rutera predstavlja **podmrežu** [RFC 950]. (Podmreža se u internet literaturi naziva *IP mreža* ili prosto *mreža*.) IP adresiranje pridružuje adresu ovoj podmreži: 223.1.1.0/24, gde oznaka /24, koja se ponekad naziva **maska podmreže**, znači da 24 krajnje leva bita 32-bitne vrednosti 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.



Slika 4.16 ◆ Adrese podmreža

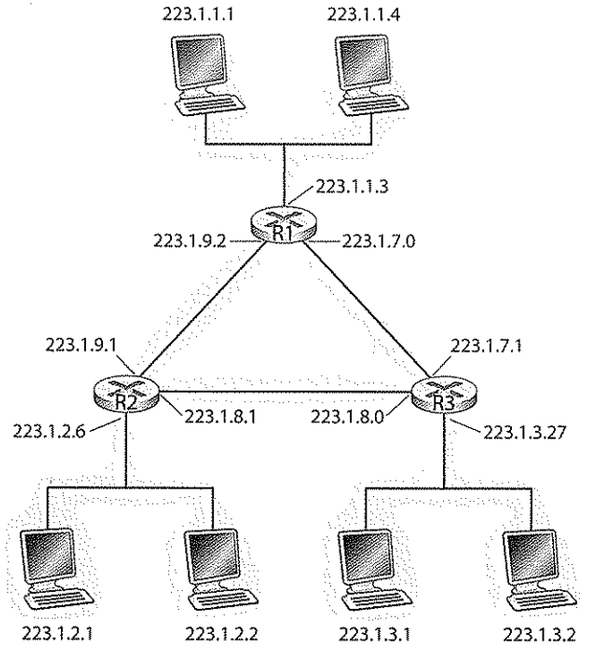
IP definicija podmreže nije ograničena na delove Eternet mreže koji povezuju više računara sa interfejsom rutera. Da biste to bolje shvatili, pogledajte sliku 4.17 na kojoj su prikazana tri rutera međusobno povezana namenskim tačka-tačka linkovima. Svaki ruter ima tri interfejsa, po jedan za svaki namenski link i jedan za difuzni link koji povezuje ruter neposredno sa parom računara. Koje podmreže postoje 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 interfejse koji povezuju rutere R1 i R2; još jedna podmreža 223.1.8.0/24 za interfejse koji povezuju rutere R2 i R3 i treća podmreža 223.1.7.0/24 za interfejse koji povezuju rutere R3 i R1.

Za opšti sistem međusobno povezanih rutera i računara možemo upotrebiti sledeće pravilo kojim se definišu podmreže u tom sistemu:

> *Da biste odredili podmreže, razdvojite sve interfejse od njihovih računara ili rutera, praveći ostrva zasebnih mreža, sa interfejsima koji se nalaze na krajevima tih zasebnih mreža. Sve te zasebne mreže nazivaju se* ***podmreže****.*

Ako primenimo ovaj postupak na međusobno povezani sistem sa slike 4.17, dobićemo šest ostrva ili podmreža.

Iz prethodnog razmatranja, jasno je da neka organizacija (preduzeće ili akademska ustanova) sa više Eternet segmenata i namenskih linkova ima više podmreža, pri čemu svi uređaji u datoj podmreži imaju istu adresu podmreže. U opštem slučaju, različite podmreže mogu da imaju sasvim različite adrese podmreže. U praksi, međutim, njihove adrese podmreža imaju dosta zajedničkog. Da biste shvatili zašto, razmotrimo kako se adresiranje sprovodi na globalnom internetu.



Slika 4.17 ◆ Tri rutera međusobno povezana u šest podmreža

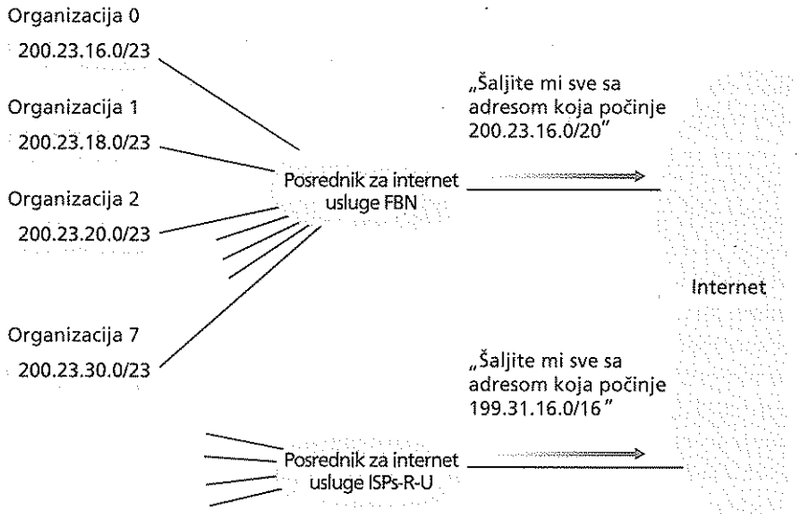
Strategija dodeljivanja adresa na internetu poznata je kao **besklasno rutiranje između domena** (engl. *Classless Interdomain Routing*, CIDR – izgovara se *saider*) [RFC 4632]. Korišćenje CIDR šeme uopštava adresiranje podmreža. Kao i kod adresiranja podmreža, 32-bitna IP adresa deli se na dva dela, i ovog puta 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*, sačinjava mrežni deo IP adrese i obično se naziva **prefiks** (ili *mrežni prefiks*) adrese. Organizaciji se obično dodeljuje grupa susednih adresa, tj. raspon adresa sa zajedničkim prefiksom (pročitajte izdvojeni deo: Principi u praksi). U ovom slučaju, IP adrese uređaja u organizaciji imaju zajednički prefiks. Kada u odeljku 4.6 budemo obrađivali protokol rutiranja na internetu, BGP, videćemo da ruteri izvan mreže ove organizacije uzimaju u obzir samo tih vodećih *x* bitova prefiksa. Drugim rečima, kada ruter izvan organizacije prosleđuje datagram čija je odredišna adresa u organizaciji, dovoljno je da u obzir uzme samo tih vodećih *x* bitova adrese. Ovo značajno smanjuje veličinu tabela prosleđivanja u ruterima, pošto je samo *jedan* unos u obliku *a.b.c.d/x* dovoljan za prosleđivanje paketa na *bilo koje* odredište unutar organizacije.

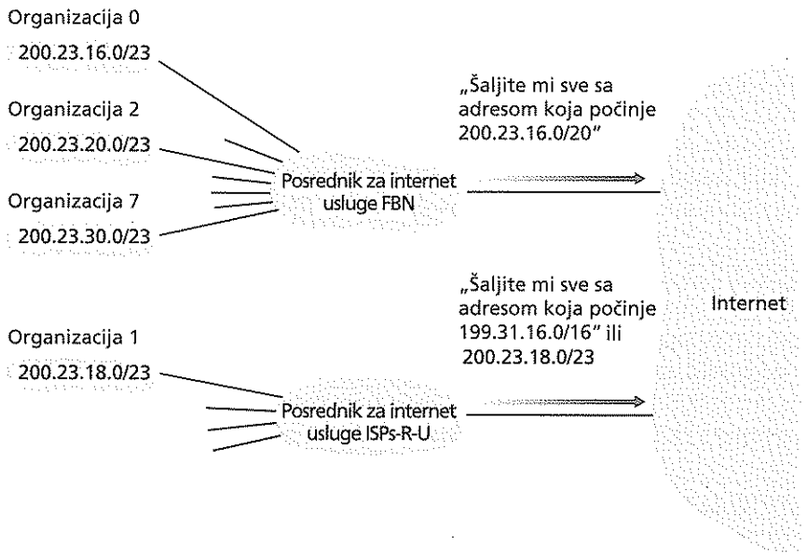
PRINCIPI U PRAKSI

Ovaj primer sa posrednikom za internet usluge koji povezuje osam organizacija sa internetom lepo pokazuje kako pažljivo dodeljene adrese, korišćenjem CIDR šeme, olakšavaju rutiranje. Pretpostavimo, kao što je prikazano na slici 4.18, da taj posrednik (koga ćemo nazvati FBN od Fly-By-Night) objavljuje spoljnom svetu da bi mu trebalo slati sve datagrame čijih se prvih 20 bitova u adresi poklapa sa 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, svaka sa svojom vlastitom podmrežom. Ova mogućnost da se koristi jedan prefiks za predstavljanje više mreža često se naziva **agregacija adresa** (a takođe i **agregacija ruta** ili **sažimanje ruta**).

Agregacija adresa radi izuzetno dobro, kada se adrese dodeljuju posrednicima za internet usluge u blokovima, a zatim ih posrednik u blokovima dodeljuje organizacijama klijentima. Ali, šta se događa kada adrese nisu dodeljene na takav hijerarhijski način? Šta bi se, na primer, desilo ako posrednik FBN kupi mrežu posrednika ISPs-R-U, potom „Organizaciju 1" poveže sa internetom preko tog novog posrednika? Kako je prikazano na slici 4.18, dodatno kupljeni posrednik ISPs-R-U ima sopstveni blok adresa 199.31.16.0/16, ali IP adrese „Organizacije 1" su nažalost izvan tog bloka adresa. Šta sada da se radi? Naravno, „Organizacija 1" bi mogla da promeni brojeve svih svojih rutera i računara, tako da imaju adrese unutar bloka adresa posrednika ISPs-R-U. Ali, to je skupo rešenje, a „Organizacija 1" bi u budućnosti mogla da bude pridružena nekom drugom posredniku. Rešenje koje se obično koristi jeste da „Organizacija 1" zadrži svoje IP adrese u opsegu 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 ISPs-R-U objavljuje blok adresa 199.31.16.0/16. Međutim, posrednik ISPs-R-U sada *takođe* objavljuje i blok adresa za „Organizaciju 1", 200.23.18.0/23. Kada ostali ruteri u većem internetu naiđu na blok adresa 200.23.16.0/20 (posrednika FBN) i 200.23.18.0/23 (posrednika ISPs-R-U), a žele da pakete usmere prema nekoj adresi iz bloka adresa 200.23.18.0/23, oni će upotrebiti pravilo najdužeg podudaranja prefiksa (videti odeljak 4.2.2) i usmeriti pakete prema mreži ISPs-R-U, pošto on objavljuje najduži (najodređeniji) prefiks adrese koji se preklapa sa adresom odredišta.



Slika 4.18 ◆ Hijerahijsko adresiranje i agregacija ruta



Slika 4.19 ◆ Posrednik ISPs-R-U ima određeniju rutu do Organizacije 1

Namena preostala 32 - x bita adrese jeste da se napravi razlika između uređaja *unutar* organizacije, pri čemu svi oni imaju isti mrežni prefiks. To su bitovi koji će biti uzeti u obzir onda kada se paketi prosleđuju na ruterima *unutar* organizacije. Ovi bitovi nižeg reda mogu (ali ne moraju) da dodatno budu uređeni u obliku podmreže, kao što smo već ranije govorili o tome. Na primer, pretpostavimo da prvih 21 bitova adrese CIDR šeme a.b.c.d/21 određuju mrežni prefiks neke organizacije i da su zajednički za IP adrese svih uređaja u toj organizaciji. Preostalih 11 bitova predstavljaju određene računare u organizaciji. Unutrašnja struktura organizacije može da bude takva da se tih 11 krajnje desnih bitova koristi za podmreže unutar organizacije, kao što je prethodno opisano. Na primer, a.b.c.d/24 može da se odnosi na određenu podmrežu unutar te organizacije.

Dok nije prihvaćena CIDR šema, mrežni deo IP adrese morao je da bude dužine 8, 16 ili 24 bitova, u šemi adresiranja poznatoj kao **klasno adresiranje**, jer su podmreže sa adresama od 8, 16 ili 24 bitova bile poznate kao mreže klase A, B, odnosno C. Zahtev da deo IP adrese koji se odnosi na podmrežu bude dužine tačno 1, 2 ili 3 bajta se pokazao kao problematičan, jer je ovaj sistem adresiranja bio nedovoljan da bi se podržao sve veći broj organizacija sa malim i srednjim podmrežama. Podmreža klase C (/24) mogla bi da obuhvati samo $2^8 - 2 = 254$ računara (dve od $2^8 = 256$ adresa ostavljene su za posebnu namenu) – premalo za većinu organizacija. Međutim, podmreža klase B (/16), koja podržava do 65 634 računara bila je prevelika. U klasnom adresiranju, organizacija sa, recimo, 2 000 računara obično je dobijala adresu podmreže klase B (/16). To je dovelo do naglog trošenja adresnog prostora klase B i loše iskorišćenosti dodeljenog adresnog prostora. Na primer, organizaciji koja bi koristila adresu klase B za svojih 2 000 računara bio je dodeljen adresni prostor dovoljan za 65 534 interfejsa – ostavljajući neiskorišćenih 63 000 adresa, koje druge organizacije nisu mogle da koriste.

Pogrešili bismo da ne pomenemo još jednu vrstu IP adrese, IP adresu za difuzno emitovanje 255.255.255.255. Kada računar pošalje datagram sa adresom odredišta 255.255.255.255, poruka se isporučuje svim računarima u istoj podmreži. Ruteri mogu tu poruku da proslede i susednim IP podmrežama (mada to obično ne rade).

Pošto smo podrobno proučili IP adresiranje, potrebno je da saznamo kako računari ili podmreže uopšte dobijaju svoje adrese. Prvo ćemo razotkriti kako neka organizacija dobija blok adresa za svoje uređaje, a zatim ćemo videti kako se nekom uređaju (na primer, računaru) dodeljuje jedna od adresa iz bloka adresa te organizacije.

Dobijanje bloka adresa

Da bi dobio blok IP adresa koje će se koristiti u podmreži organizacije, administrator mreže može prvo da pozove svog posrednika za internet usluge, koji će mu obezbediti adrese iz većeg bloka adresa koje su ranije dodeljene tom posredniku za internet usluge. Na primer, neka je posredniku za internet usluge dodeljen blok adresa 200.23.16.0/20. Posrednik je svoj blok adresa podelio na osam manjih jednakih neprekidnih blokova adresa i po jedan od tih blokova dodelio svakoj od najviše osam organizacija koje podržava, kao što je prikazano niže. (Podvukli smo mrežni deo ovih adresa, da bismo vam olakšali.)

Blok posrednika	200.23.16.0/20	11001000 00010111 0001**0000 00000000**
Organizacija 0	200.23.16.0/23	11001000 00010111 0001000**0 00000000**
Organizacija 1	200.23.18.0/23	11001000 00010111 0001001**0 00000000**
Organizacija 2	200.23.20.0/23	11001000 00010111 0001010**0 00000000**
...		...
Organizacija 7	200.23.30.0/23	11001000 00010111 0001111**0 00000000**

Dobijanje skupa adresa od posrednika za internet usluge jedan je od načina za dobijanje bloka adresa, ali nije i jedini. Jasno je, mora da postoji neki način na koji i sam posrednik za internet usluge dobija blok adresa. Da li postoji globalno ovlašćeno telo sa najvišom odgovornošću za upravljanje prostorom IP adresa i dodeljivanje blokova adresa posrednicima za internet usluge i ostalim organizacijama? Zaista postoji! IP adresama upravlja Internet korporacija za dodeljena imena i brojeve (Internet Corporation for Assigned Names and Numbers, ICANN) [ICANN 2012] na osnovu smernica postavljenih u [RFC 2050]. Uloga neprofitne organizacije ICANN [NTIA 1998] nije samo dodeljivanje IP adresa, već takođe i upravljanje korenskim 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 adresiranju ICANN [ASO-ICANN 2012]) koji zatim dodeljuju adrese i upravljaju njima svaki unutar svojih regiona.

Dobijanje adrese računara: protokol za dinamičko konfigurisanje računara

Nakon što organizacija pribavi blok adresa, može da dodeli pojedinačne IP adrese interfejsima računara i rutera u svojoj organizaciji. Administrator sistema najčešće ručno konfiguriše IP adrese unutar rutera (često to obavlja daljinski, alatkama za upravljanje mrežom). Adrese računara takođe mogu da se konfigurišu ručno, ali se mnogo češće to obavlja korišćenjem **protokola za dinamičko konfigurisanje računara (Dynamic Host Configuration Protokol, DHCP)** [RFC 2131]. DHCP

omogućava da računar automatski dobije (da mu se dodeli) IP adresa. Mrežni administrator može da konfiguriše DHCP tako da dati računar dobija uvek istu IP adresu, uvek kada se povezuje sa mrežom ili tako da se računaru dodeljuje **privremena IP adresa** koja će biti svaki put drugačija kada se određeni računar poveže sa mrežom. Pored dodeljivanja IP adresa računarima, DHCP omogućava računaru da sazna dodatne informacije, kao što su maska podmreže, adresa rutera koji mu je prvi na putu (koji se obično naziva podrazumevani mrežni prolaz) i adresa njegovog lokalnog DNS servera.

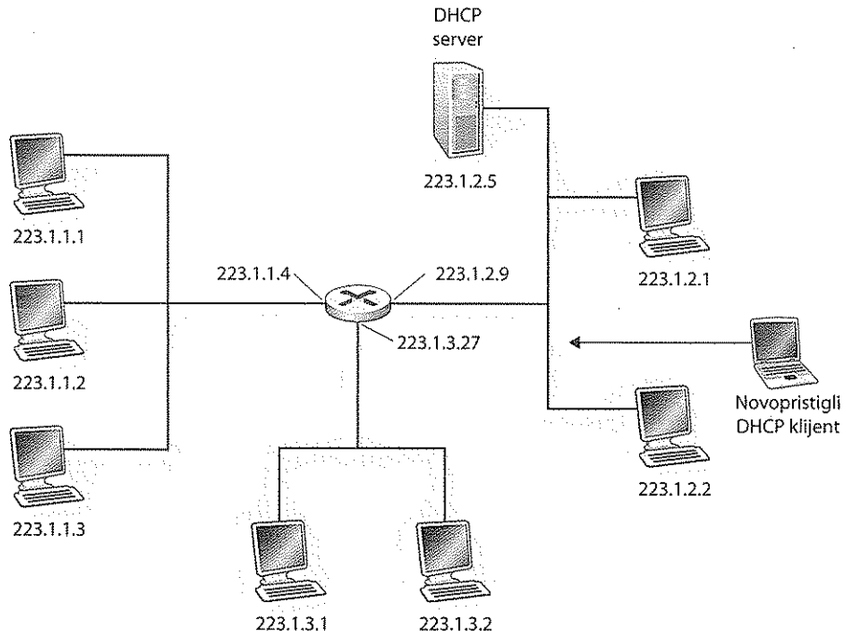
Pošto DHCP može da automatizuje postupke koji se tiču povezivanja računara u mrežu, često se naziva **protokol uključi i radi** (eng. plug-and-play). Zbog tog svojstva, *veoma* je privlačan mrežnim administratorima koji bi inače morali te zadatke da obave ručno! DHCP takođe ima široku primenu u kućnim mrežama za pristup internetu i u bežičnim LAN mrežama u kojima računari često pristupaju mreži i napuštaju je. Uzmimo na primer studenta koji nosi laptop od svoje spavaonice, zatim do biblioteke i na kraju do učionice. Na svim tim mestima, laptop tog studenta bio bi povezan sa novom podmrežom, pa bi mu stoga bila potrebna nova IP adresa na svim tim mestima. DHCP je savršeno prilagođen za takve okolnosti, gde mnogi korisnici dolaze i odlaze, a adrese su im potrebne samo ograničeno vreme. DHCP je izuzetno koristan i pristupnim mrežama posrednika za internet usluge za stanovništvo. Uzmite, na primer, da regionalni ISP posrednik ima 2 000 korisnika, ali da nikad nije istovremeno priključeno više od 400. U tom slučaju, umesto da zauzme blok od 2 048 adresa, DHCP serveru koji adrese raspodeljuje dinamički neophodan je samo blok od 512 adresa (na primer, blok u obliku a.b.c.d/23). Kako računari dolaze i odlaze, DHCP server mora da ažurira listu svojih dostupnih IP adresa. Kad god se neki računar priključi, DHCP server mu dodeljuje proizvoljnu adresu iz trenutno raspoloživog skupa adresa; kada se neki računar isključi, njegova adresa vraća se u listu dostupnih adresa.

DHCP je klijentsko serverski protokol. Klijent je obično novopristigli računar koji čeka da dobije informacije o konfiguraciji mreže, među njima i IP adresu koju će koristiti. U najjednostavnijem slučaju, svaka podmreža (u smislu adresiranja sa slike 4.17) ima DHCP server. Ukoliko na podmreži ne postoji takav server, neophodan je posrednik za DHCP usluge (obično neki ruter) koji zna adresu DHCP servera za tu mrežu. Na slici 4.20 prikazan je DHCP server povezan s podmrežom 223.1.2/24, sa ruterom koji služi kao posrednik za pristigle klijente, povezanim u podmreže 223.1.1/24 i 223.1.3/24. U razmatranju koje sledi, podrazumevamo da je DHCP server dostupan na podmreži.

Za novopristigli računar, protokol DHCP se odvija u četiri koraka, kao što je prikazano na slici 4.21 za mrežnu postavku prikazanu na slici 4.20. Na ovoj slici, `yiaddr` (što znači „vaša internet adresa") označava adresu koja se dodeljuje novopristiglom klijentu. Četiri koraka su:

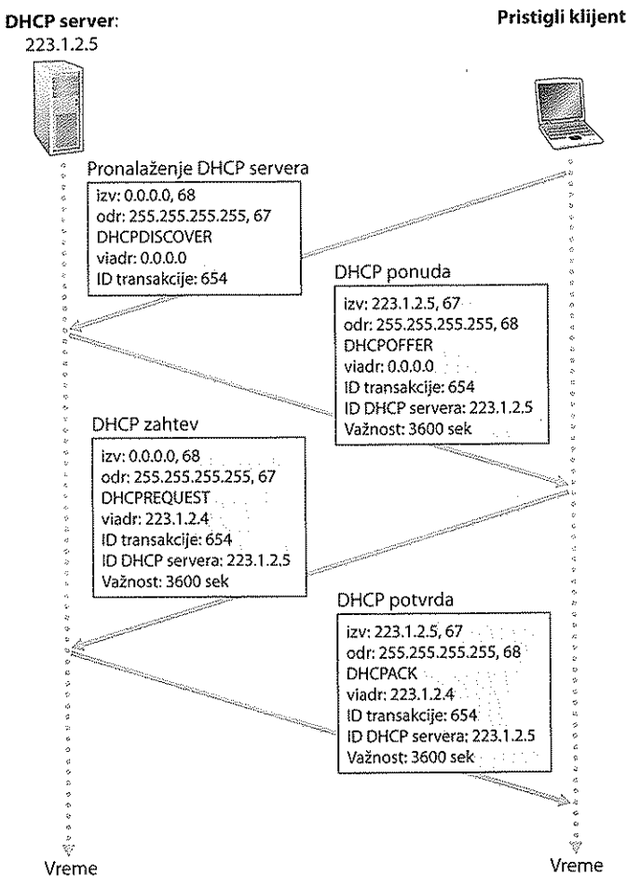
- *Pronalaženje DHCP servera.* Prvi zadatak novopristiglog računara je da pronađe DHCP server sa kojim će da sarađuje. To se obavlja **DHCP porukom za pronalaženje**, koju klijent šalje unutar UDP paketa na port 67. UDP paket

se enkapsulira u IP datagram. Ali, kome bi trebalo poslati taj datagram? Računar, ne samo da ne zna adresu DHCP servera za tu mrežu, već ne zna ni IP adresu mreže sa kojom se povezao. Zbog toga, DHCP klijent pravi IP datagram koji sadrži njegovu DHCP poruku za pronalaženje sa IP adresom odredišta 255.255.255.255 i izvornom IP adresom „ovog računara" 0.0.0.0. DHCP klijent prenosi ovaj IP datagram do sloja veze, koji potom taj okvir difuzno dostavlja svim čvorovima povezanim u podmrežu (difuzno slanje sloja veze podrobnije razmatramo u odeljku 5.4).



Slika 4.20 ◆ Slučaj DHCP klijenata i servera

- *Ponuda (ili ponude) DHCP servera.* DHCP server koji primi DHCP poruku za pronalaženje odgovara klijentu **DHCP porukom sa ponudom** koju difuzno dostavlja svim čvorovima na mreži, pri čemu i on koristi IP adresu za difuzno slanje 255.255.255.255. (Razmislite o tome zašto odgovor servera takođe mora da se difuzno šalje.) Pošto na podmreži može da bude više DHCP servera, klijent može da se nađe u zavidnom položaju – može da bira između više ponuda. Poruke sa ponudom svih servera sadrže transakcioni ID primljene poruke za pronalaženje, predloženu IP adresu za klijenta, mrežnu masku i **vreme iznajmljivanja IP adrese** – vreme tokom koga će ta IP adresa biti važeća. Uobičajeno je da serveri trajanje iznajmljivanja postave na nekoliko sati ili nekoliko dana [Droms 2012].



Slika 4.21 ◈ Interakcija DHCP servera i klijenata

- *DHCP zahtev*. Novopristigli klijent bira ponudu jednog ili više servera i odgovara na izabranu ponudu **DHCP porukom sa zahtevom**, vraćajući serveru neizmenjene parametre konfiguracije.
- *DHCP potvrda prijema (DHCP ACK)*. Server na DHCP poruku sa zahtevom odgovara **DHCP ACK potvrdom**, potvrđujući zahtevane parametre.

Kada klijent primi DHCP potvrdu, postupak je gotov i klijent može da koristi IP adresu koju mu je DHCP server dodelio tokom vremena za koje je iznajmljena. Pošto klijent možda želi da koristi ovu adresu i posle isteka ovog vremena, DHCP takođe nudi mehanizme koji klijentu omogućavaju da obnovi iznajmljenu IP adresu.

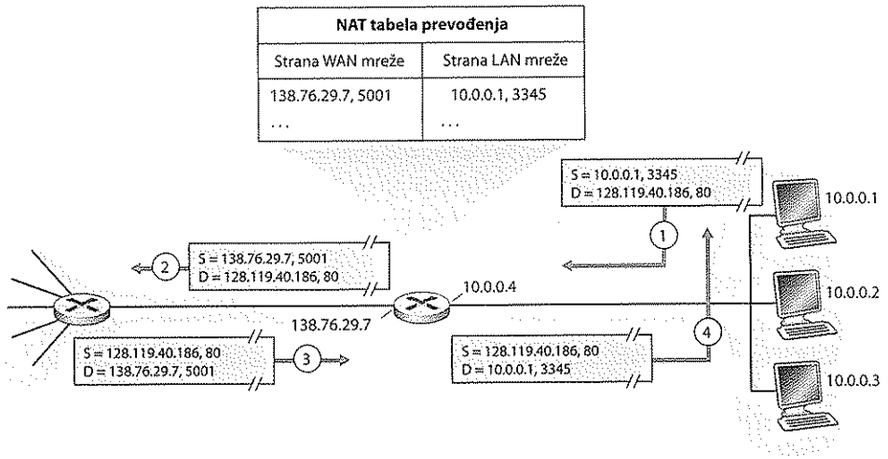
Značaj koju ima mogućnost protokola DHCP da se računar priključi i odmah radi je očigledna, s obzirom na to da je druga mogućnost da se IP adrese računara ručno konfigurišu. Posmatrajmo šta bi bilo sa studentom koji svoj laptop nosi od spavaonice, preko biblioteke do učionice, pridružujući se novoj podmreži svaki put i dobijajući novu IP adresu na svim tim mestima. Nemoguće je i zamisliti kako bi to administrator sistema mogao ispočetka da konfiguriše laptop na svim tim mestima, a tek nekolicina studenata (ne računajući one koje prisustvuju predavanjima o umrežavanju) bi umela da samostalno konfiguriše svoje laptopove. Sa stanovišta mobilnostii, međutim, DHCP ima i svoje nedostatke. Pošto se nova IP adresa dobija od DHCP servera, uvek kada se neki računar poveže sa novom podmrežom, TCP veza sa udaljenom aplikacijom ne može da se održi, jer se mobilni čvor premešta između podmreža. U poglavlju 6 istražujemo mobilni IP – najnovije proširenje IP infrastrukture koje omogućava mobilnom čvoru da koristi samo jednu trajnu adresu, dok se premešta između podmreža. Dodatne pojedinosti o protokolu DHCP mogu se naći u [Droms 2002] i [dhc 2012]. Javno dostupno pojašnjenje primene protokola DHCP moguće je dobiti od udruženja *Internet Systems Consortium* [ISC 2012].

Prevođenje mrežnih adresa

S obzirom na ono što smo dosad rekli o internet adresiranju i formatu IPv4 datagrama, nesumnjivo je da svaki uređaj koji može da se poveže na mrežu mora da ima IP adresu. Sa naglim širenjem takozvanih kućnih SOHO (small office, home office) podmreža, to bi naizgled značilo, da kad god bi takav neki korisnik želeo da instalira LAN mrežu za međusobno povezivanje više mašina, posrednik za internet usluge morao bi da mu dodeli čitav raspon adresa, kako bi se obuhvatile sve te mašine. Ako bi podmreža kasnije porasla (na primer, kada deca kod kuće imaju ne samo svoje računare, već i pametne telefone i umrežene konzole za igranje), morao bi im se dodeliti još veći broj adresa. A, šta ako je posrednik u međuvremenu već drugima dodelio neprekidne delove iz trenutnog adresnog prostora te SOHO mreže? I šta uopšte prosečan kućni korisnik hoće (ili bi trebalo) da zna o upravljanju IP adresama? Srećom, postoji jednostavniji način za dodeljivanje adresa koji se najviše koristi u takvom slučaju: **prevođenje mrežnih adresa** (engl. *Network Address Translation*, **NAT**) [RFC 2663; RFC 3022; Zhang 2007].

Na slici 4.22 prikazan je način rada NAT rutera. Kućni NAT ruter ima interfejs koji je deo kućne mreže na desnoj strani slike 4.22. Adresiranje unutar kućne mreže istovetno je onome što smo videli ranije – sva četiri interfejsa kućne mreže imaju istu adresu podmreže, 10.0.0/24. Adresni prostor 10.0.0/8 je jedan od tri dela prostora IP adresa koji su dokumetnom [RFC 1918] predviđeni za privatne mreže ili **prostor** privatnih adresa, kao što je kućna mreža na slici 4.22. *Prostor privatnih adresa* označava mrežu čije adrese samo uređajima unutar te mreže nešto znače. Da biste videli zašto je to važno, uzmite činjenicu da postoji na stotine hiljada kućnih mreža od kojih mnoge koriste isti adresni prostor, 10.0.0/24. Uređaji unutar

date kućne mreže mogu jedan drugome da šalju pakete, koristeći šemu adresiranja 10.0.0/24. Međutim, jasno je da paketi koji se šalju *preko granice* kućne mreže, na veći globalni internet, ne mogu da koriste te adrese (bilo kao adresu izvora ili kao adresu odredišta), jer ima 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 naglobalni internet, ili se primaju iz interneta gde adrese moraju da budu jedinstvene? Odgovor se nalazi u NAT prevođenju.



Slika 4.22 ◆ Prevođenje mrežnih adresa

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 4.22 sav saobraćaj koji polazi od kućnog rutera prema većem internetu ima izvornu IP adresu 138.76.29.7 i sav saobraćaj koji dolazi do kućnog rutera mora imati adresu odredišta 138.76.29.7. U suštini, NAT ruter krije pojedinosti kućne mreže od spoljnog sveta. (Uzgred, možda se pitate gde računari u kućnoj mreži dobijaju adrese i gde ruter dobija tu jednu IP adresu. Često je odgovor isti – od DHCP servera! Ruter dobija svoju adresu od DHCP servera posrednika za internet usluge, a isti kućni ruter obavlja poslove DHCP servera koji obezbeđuje adrese za računare unutar adresnog prostora kućne mreže, kojim upravlja NAT DHCP ruter).

Ako svi datagrami koji do NAT rutera stižu iz širokopojasne mreže imaju istu IP adresu odredišta (tačnije, adresu interfejsa NAT rutera prema širokopojasnoj mreži), kako ruter može da zna kom računaru unutar mreže bi trebalo da prosledi dati datagram? Caka je u korišćenju **NAT tabele prevođenja** u NAT ruteru i u tome da se u stavke tabele osim IP adresa dodaju i brojevi portova.

Pogledajmo primer prikazan na slici 4.22. Pretpostavimo da korisnik koji sedi za računarom 10.0.0.1 u kućnoj mreži zatraži veb stranicu na nekom veb serveru (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 138.76.29.7 koju koristi za WAN i zamenjuje prvobitni broj izvornog porta 3345 novim brojem izvornog porta 5001. Prilikom izbora novog broja izvornog porta, NAT ruter može da izabere bilo koji broj izvornog porta koji se trenutno ne nalazi u NAT tabeli prevođenja. (Obratite pažnju na to da pošto polje za broj porta ima 16 bitova, protokol NAT može da podrži preko 60 000 istovremenih veza sa samo jednom IP adresom tog rutera sa WAN strane!). NAT u ruteru takođe dodaje stavku u svoju NAT tabelu prevođenja. Veb server, u blaženom neznanju da je pristigli datagram koji sadrži HTTP zahtev prepravljen u NAT ruteru, odgovara datagramom čija je adresa odredišta IP adresa NAT rutera, a broj odredišnog porta je 5001. Kada taj datagram stigne do NAT rutera, on pomoću odredišne IP adrese i broja porta u svojoj NAT tabeli prevođenja pronalazi odgovarajuću IP adresu (10.0.0.1) i odredišni broj porta (3345) za veb pretraživač u kućnoj mreži. Ruter tada u datagramu prepravlja adresu odredišta i broj odredišnog porta i prosleđuje datagram u matičnu mrežu.

NAT se poslednjih godina sve više primenjuje. Ipak, trebalo bi pomenuti da se mnogi čistunci u IETF zajednici žučno protive tome. Prvo, prigovaraju da su brojevi portova namenjeni za adresiranje procesa, a ne za adresiranje računara. (Kršenje ovog pravila zaista može da stvori probleme serverima koji se izvršavaju u matičnoj mreži pošto, kako smo videli u poglavlju 2, serverski procesi čekaju dolazne zahteve na dobro poznatim brojevima portova). Drugo, prigovaraju da bi ruteri trebalo da obrađuju pakete samo do sloja 3. Treće, prigovaraju da protokol NAT krši takozvani argument od kraja do kraja; odnosno, da bi računari trebalo neposredno da komuniciraju, a ne da usputni čvorovi menjaju IP adrese i brojeve portova. I četvrto, prigovaraju da bi za rešenje nedostatka IP adresa trebalo upotrebiti IPv6 (pogledajte odeljak 4.4.4), a ne lakomisleno rešavati 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 protokolom NAT jeste u tome da on ometa P2P aplikacije, uključujući P2P aplikacije za deljenje datoteka i VoIP aplikacije. Verovatno se sećate iz poglavlja 2 da je kod P2P aplikacija moguće da bilo koji ravnopravni učesnik A uspostavi TCP vezu sa bilo kojim drugim ravnopravnim učesnikom B. Suština problema je u tome da, ukoliko se učesnik B nalazi iza NAT rutera, on ne može da se ponaša kao server i da prihvata TCP veze. Kao što ćemo videti u domaćim zadacima, ovaj problem sa protokolom NAT može da se izbegne ukoliko učesnik A nije iza NAT rutera. U ovom slučaju, ravnopravni računar A prvo se povezuje sa ravnopravnim računarom B preko posrednog ravnopravnog računara C, koji nije iza NAT rutera i sa kojim je B već uspostavio TCP vezu. Ravnopravni računar A

može tada da zatraži od ravnopravnog računara B, preko ravnopravnog računara C, da uspostavi TCP vezu neposredno sa njim. Pošto se uspostavi neposredna P2P TCP veza između ravnopravnih računara A i B, oni mogu da razmenjuju poruke ili datoteke. Ovaj trik, nazvan **preokretanje veze** (eng. connection reversal), koristi se u mnogim P2P aplikacijama za tzv. **zaobilaženje NAT** rutera. Međutim, kada su oba ravnopravana računara A i B iza svojih NAT rutera, stvari su nešto složenije, ali se mogu prevazići korišćenjem tzv. releja u aplikacijama, kao što smo videli u poglavlju 2 sa relejima aplikacije *Skype*.

UpnP

Zaobilaženje NAT rutera uglavnom se sprovodi protokolom UPnP (Universal Plug and Play), koji predstavlja protokol koji računaru omogućava da pronađe i konfiguriše susedni NAT ruter [UPnP Forum 2012]. UPnP zahteva da i računar i NAT ruter podržavaju UPnP. UPnP se koristi tako što aplikacija koja se izvršava na nekom računaru zahteva NAT preslikavanje njenih vrednosti (*privatna IP adresa, privatni broj porta*) u (*javna IP adresa, javni broj porta*) za neki zahtevani javni broj porta. Ukoliko NAT prihvati ovaj zahtev i obavi preslikavanje, onda čvorovi spolja mogu da uspostave TCP veze, koristeći vrednosti (*javna IP adresa, javni broj porta*). Pored toga, UPnP obaveštava aplikaciju o vrednostima (*javna IP adresa, javni broj porta*), tako da ta aplikacija može o njima da obavesti spoljni svet.

Na primer, pretpostavimo da vaš računar, iza NAT rutera koji podržava UPnP, ima privatnu adresu 10.0.0.1 i da je na njemu pokrenut BitTorrent na portu 3345. Takođe, pretpostavimo da je javna IP adresa NAT rutera 138.76.29.7. Prirodno je da vaša aplikacija, koja koristi BitTorrent, želi da joj se omogući da prihvata veze od drugih računara, tako da može da razmenjuje odsečke datoteka sa njima. Da bi to ostvarila, BitTorrent aplikacija na vašem računaru traži od NAT rutera da napravi „otvor" kojim preslikava (10.0.0.1, 3345) u (138.76.29.7, 5001). (Broj javnog porta 5001 izabrala je aplikacija). BitTorrent aplikacija na Vašem računaru takođe može da objavi svom pratiocu da je dostupna na (138.76.29.7, 5001). Na taj način, računar izvan mreže na kome se izvršava BitTorrent može da stupi u vezu sa tim pratiocem i da sazna da se Vaša BitTorrent aplikacija izvršava na (138.76.29.7, 5001). Taj spoljni računar može da pošalje TCP SYN paket na (138.76.29.7, 5001). Kada NAT ruter primi taj SYN paket, on menja IP adresu odredišta i broj porta u paketu na (10.0.0.1, 3345) i paket prosleđuje kroz NAT ruter.

Ukratko, UPnP omogućava spoljnim računarima da uspostave vezu sa računarima iza NAT rutera, korišćenjem protokola TCP ili UDP. NAT ruteri dugo su bili pogibeljni za P2P aplikacije; protokol UPnP koji nudi efikasno i pouzdano rešenje za zaobilaženje NAT rutera može da bude njihov spasitelj. Ovde smo objasnili NAT rutere i protokol UPnP samo u najkraćim crtama. Za podrobnije razmatranje NAT rutera pogledajte [Huston 2004, Cisco NAT 2012].

4.4.3 Protokol ICMP

Sećate se da mrežni sloj interneta ima tri glavne komponente: IP protokol, razmatran u prethodnom odeljku; protokole rutiranja interneta (uključujući RIP, OSPF i BGP) o kojima se govori u odeljku 4.6 i protokol ICMP koji je tema ovog odeljka.

ICMP (engl. *Internet Control Message Protokol* – Internet protokol za kontrolne poruke), definisan u dokumentu [RFC 792], koriste računari i ruteri za međusobnu razmenu informacija mrežnog sloja. Protokol ICMP se najčešće koristi za izveštavanje o greškama. Na primer, prilikom izvršavanja Telnet, FTP ili HTTP sesija, možda ste naišli na poruku o greški kao što je: „Destination network unreachable" (odredišna mreža nedostupna). Poreklo te poruke je u protokolu ICMP. 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.

Za ICMP se obično smatra da je deo protokola IP, mada se u arhitekturi interneta nalazi neposredno iznad protokola IP, pošto se ICMP poruke prenose unutar IP datagrama. To jest, ICMP poruke se prenose kao korisni podaci protokola IP, baš kao što se i TCP ili UDP segmenti prenose kao korisni podaci protokola IP. Slično, kada računar primi IP datagram u kome je kao protokol gornjeg sloja naveden ICMP, on demultipleksira sadržaj tog datagrama za ICMP, baš kao što bi demultipleksirao sadržaj nekog datagrama za protokole TCP ili UDP.

ICMP poruke imaju polja za tip i za kôd poruke, a sadrže zaglavlje i prvih 8 bajtova IP datagrama zbog kojih je ICMP poruka uopšte i nastala (tako da pošiljalac može da utvrdi koji je datagram doveo do greške). Nekoliko ICMP poruka prikazano je na slici 4.23. Obratite pažnju na to da se ICMP poruke ne koriste samo za obaveštavanje o grešakama.

Dobro poznati program *ping* šalje ICMP poruku tipa 8, sa kodom 0 određenom računaru. Odredišni računar, videvši zahtev da vrati eho, šalje nazad ICMP eho odgovor tipa 0 sa kodom 0. U većini realizacija protokola TCP/IP `ping` server je implementiran u samom operativnom sistemu; tj. ovaj server nije proces. U poglavlju 11 knjige [Stevens 1990] nalazi se izvorni kôd klijentskog programa ping. Obratite pažnju na to da je neophodno da klijentski program naredi operativnom sistemu da napravi ICMP poruku tipa 8, sa kodom 0.

Još jedna zanimljiva ICMP poruka je poruka za suzbijanje izvora. Ta poruka se retko koristi u praksi. Njena prvobitna namena bila je da se tako kontroliše zagušenje – omogućiti zagušenom ruteru da računaru pošalje ICMP poruku za suzbijanje izvora i primora taj računar da smanji brzinu slanja. U poglavlju 3 smo videli da TCP ima sopstveni mehanizam za kontrolu zagušenja koji svoj posao obavlja na transportnom sloju, bez korišćenja povratnih informacija mrežnog sloja kao što je ICMP poruka za suzbijanje izvora.

U poglavlju 1 predstavili smo program *Traceroute* koji nam omogućava da sledimo putanju od nekog računara do bilo kog računara na svetu. Zanimljivo, program *Traceroute* ostvaren je pomoću ICMP poruka. Da bi utvrdio imena i adrese rutera između izvora i odredišta, *Traceroute* na izvoru šalje ka odredištu niz običnih IP

datagrama. Svaki od ovih datagrama nosi jedan UDP segment sa malo verovatnim brojem UDP porta. Prvi od ovih datagrama ima TTL vrednost 1, drugi 2, treći 3 itd. Izvor takođe pokreće tajmere za sve te datagrame. Kada *n*-ti datagram stigne do *n*-tog rutera, *n*-ti ruter primećuje da je TTL vreme datagrama upravo isteklo. Prema pravilima protokola IP, ruter odbacuje taj datagram i izvoru šalje ICMP poruku upozorenja (tip 11 kôd 0). Ta poruka upozorenja sadrži ime rutera i njegovu IP adresu. Kada se ta ICMP poruka vrati do izvora, izvor vreme povratnog puta dobija od tajmera, a ime i IP adresu *n*-tog rutera dobija iz ICMP poruke.

ICMP tip	Kôd	Opis
0	0	eho odgovor (za ping)
3	0	odredišna mreža nedostupna
3	1	odredišni računar nedostupan
3	2	odredišni protokol nedostupan
3	3	odredišni port nedosutapan
3	6	odredišna mreža nepoznata
3	7	odredišni računar nepoznat
4	0	suzbijanje izvora (kontrola zagušenja)
8	0	eho zahtev
9	0	objavljivanje rutera
10	0	otkrivanje rutera
11	0	TTL vreme isteklo
12	0	IP zaglavlje loše

Slika 4.23 ◆ Vrste ICMP poruka

Kako *Traceroute* na izvoru zna kada da prestane sa slanjem UDP segmenata? Sećate se da izvor povećava vrednost polja TTL za svaki datagram koji pošalje. Jedan od datagrama će pre ili kasnije preći čitav put do odredišnog računara. Pošto datagram sadrži UDP segment sa verovatno nepostojećim brojem porta, odredišni računar vraća ICMP poruku da je odredišni port nedostupan (tip 3 kôd 3). Kada izvorni računar primi takvu ICMP poruku, on zna da ne bi trebalo više da šalje probne pakete. (Standardni program *Traceroute* u stvari šalje grupe od po tri paketa sa istim TTL vremenom; tako da njegov *Traceroute* rezultat sadrži po tri rezultata za svako TTL vreme).

BEZBEDNOST PRE SVEGA

ISPITIVANJE DATAGRAMA: ZAŠTITNE BARIJERE I SISTEMI ZA OTKRIVANJE ULJEZA

Pretpostavimo da dobijete zadatak da administrirate mrežu: u kući, na fakultetu ili u čitavom preduzeću. Napadači, znajući opseg IP adresa Vaše mreže, lako mogu da šalju IP datagrame do adresa u tom opsegu. Ovi datagrami mogu da vrše razne vrste podmuklih stvari, među njima otkrivanje topologije Vaše mreže, uz pomoć skeniranja mreže signalom ping i skeniranjem portova, obaranje osetljivih računara zlonamerno napravljenim paketima, plavljenje servera mnoštvom ICMP paketâ i zarazu računara umetanjem zlonamernog softvera unutar paketa. Šta, kao administrator mreže, možete da uradite protiv tih silnih zloća koji šalju zlonamerne pakete u Vašu mrežu? Dva omiljena mehanizma za odbranu od napada zlonamernim paketima su zaštitne barijere (eng. firewalls) i sistemi za otkrivanje uljeza (intrusion detection systems, IDS).

Prvo što biste uradili, kao administrator mreže, jeste da instalirate zaštitnu barijeru između svoje mreže i interneta. (Većina današnjih pristupnih rutera ima funkcionalnosti zaštitne barijere.) Zaštitne barijere proveravaju polja zaglavlja datagrama i segmenta, sprečavajući ulaženje sumnjivih paketâ unutar mreže. Na primer, zaštitna barijera može da se konfiguriše tako da zaustavlja sve pakete sa ICMP eho zahtevima, na taj način sprečavajući napadača da pregleda otvorene portove unutar Vašeg opsega IP adresa. Zaštitna barijera takođe može da zaustavlja pakete na osnovu IP adresa i brojeva portova izvora i odredišta. Pored toga, zaštitne barijere mogu da budu postavljene tako da prate TCP veze, dozvoljavajući ulaz samo onim datagramima koji pripadaju odobrenim vezama.

Dodatnu zaštitu nudi IDS sistem. IDS, smešten obično na granicama mreže, obavlja „dublju proveru paketa", ne ispitujući samo polja zaglavlja, već i korisne podatke u datagramu (među njima i podatke aplikativnog sloja). IDS ima bazu podataka sa potpisima paketa za koje se zna da su deo napada. Ova baza podataka se automatski ažurira čim se otkrije nova vrsta napada. Prilikom prolaska paketa kroz IDS, on pokušava da utvrdi preklapanje polja zaglavlja i korisnih podataka u tom paketu sa potpisima iz svoje baze podataka. Ukoliko pronađe podudarnost, upozorava o tome. Sistem za sprečavanje napada (Intrusion Prevention System, IPS) sličan je IDS sistemu, osim po tome što on, pored toga što upozorava na sumnjive pakete, i sprečava njihov prolazak. U poglavlju 8 podrobnije obrađujemo zaštitne barijere i IDS sisteme.

Mogu li zaštitne barijere i IDS sistemi sasvim da zaštite Vašu mrežu od napada? Jasno je da je odgovor da ne mogu, pošto napadači stalno pronalaze nove načine za napad za koje potpisi još ne postoje. Ipak, zaštitne barijere i tradicionalni IDS sistemi koji se baziraju na potpisu su i te kako korisni za zaštitu Vaše mreže od poznatih napada.

Na ovaj način, izvorni računar saznaje koliko i kojih rutera se nalazi između njega i odredišnog računara kao i vreme povratnog puta između ova dva računara. 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 mora da bude u stanju da primi obaveštenje operativnog sistema o prispeću ICMP poruka. Pošto sada znate kako *Traceroute* radi, možda ćete poželeti da se još malo igrate sa njim.

4.4.4 Ipv6

Početkom devedesetih godina, udruženje IETF (Internet Engineering Task Force – tehnička radna grupa za internet) pokrenulo je razvijanje naslednika protokola IPv4. Osnovni podstrek bio je u tome što se uvidelo da je 32-bitni prostor IP adresa skoro potrošen, a da se brzinom koja oduzima dah povećava broj novih podmreža i IP čvorova koji se povezuju sa internetom (i dobija jedinstvene IP adrese). Da bi se zadovoljila potreba za velikim prostorom IP adresa, razvijen je novi IP protokol, protokol IPv6. Projektanti protokola IPv6 iskoristili su tu priliku da dodatno prilagode i prošire i druge elemente protokola IPv4, na osnovu prikupljenog iskustva pri radu sa protokolom IPv4.

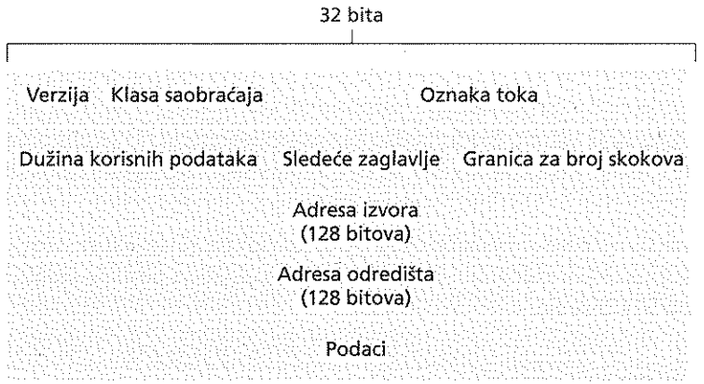
Vodile su se duge rasprave o tome kada će sve IPv4 adrese biti potrošene (i kada nijedna nova podmreža neće moći da se poveže sa internetom). Previđanja dvojice vođa radne grupe IETF za procenu očekivanog trajanja adresa (Address Lifetime Expectations) bila su da će se adrese istrošiti 2008. odnosno 2018. godine [Solensky 1996]. Februara 2011. godine, IANA (engl. Internet Assigned Numbers Authority – Organizacija za dodeljene brojeve nainternetu), raspodelila je poslednji preostali blok nedodeljenih Ipv4 adresa u regionalni registar. Iako ovi registri još uvek imaju raspoložive Ipv4 adrese unutar svog bloka, jednom kada se ove adrese istroše, više neće biti raspoloživih blokova adresa koji bi se mogli alocirati iz centralnog skupa [Huston 2011a]. Iako je sredinom devedesetih godina procena o iscrpljivanju Ipv4 adresa pretpostavljala da se ceo IPv4 adresni prostor 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 – IP protokol sledeće generacije) [Bradner 1996; RFC 1752]. Rezultat tog projekta bila je specifikacija protokola IP, verzija 6 (IPv6) [RFC 2460] o kojoj ćemo dole pričati. (Često se postavlja pitanje gde je nestao IPv5. Prvobitno je bilo predviđeno da protokol ST-2 postane IPv5, ali se od protokola ST-2 kasnije odustalo). Odlični izvori informacija o IPv6 su [Huitema 1998, IPv6 2012].

Format datagrama IPv6

Format datagrama IPv6 prikazan je na slici 4.24. Najvažnije promene koje donosi IPv6 očigledne su u formatu datagrama:

- *Proširene mogućnosti adresiranja.* IPv6 povećava veličinu IP adrese sa 32 bita na 128 bitova. Time je obezbeđeno da se nikada ne potroše sve IP adrese. Sada, svako zrnce peska na svetu može da dobije svoju IP adresu. Osim jednoznačnih i višeznačnih IP adresa, IPv6 je uveo i novu vrstu adrese, nazvanu **proizvoljna adresa** (eng. anycast), koja omogućava da se datagram isporuči bilo kom računaru iz neke grupe. (Ova osobina može, na primer, da se upotrebi za slanje HTTP GET poruke najbližem računaru od više identičnih kopija na kome se nalazi traženi dokument).

- *Jednostavnije 40-bajtno zaglavlje.* Kao što ćemo kasnije opisati, više polja iz protokola IPv4 je izbačeno, ili više nisu obavezna. Tako je dobijeno zaglavlje stalne dužine 40 bajtova koje omogućava bržu obradu IP datagrama. Nov način kodovanja opcija obezbeđuje njihovu mnogo lakšu obradu.



Slika 4.24 ◈ Format datagrama IPv6

- *Označavanje i prioriteti toka.* IPv6 koristi prilično dvosmislenu definiciju pojma **tok**. U dokumentima RFC 1752 i RFC 2460 navodi se da to omogućava: „označavanje paketa da pripadaju određenim tokovima za koje pošiljalac zahteva poseban postupak, kao što su kvalitet usluge viši od podrazumevanog, ili usluga u realnom vremenu". Na primer, audio i video prenosi verovatno će se smatrati kao tok. S druge strane, tradicionalnije aplikacije, kao što su prenos datoteka i e-pošta ne moraju se smatrati za tokove. Saobraćaj koji prenosi korisnik sa visokim prioritetom (na primer, neko ko plaća da za svoj saobraćaj dobije bolju uslugu), takođe je moguće smatrati za tok. Jasno je, međutim, da su projektanti protokola IPv6 predvideli moguću potrebu da se pravi razlika među različitim tokovima, mada još nije utvrđeno tačno značenje toka. Zaglavlje 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 datagramima unutar toka ili za davanje prednosti datagramima iz određenih aplikacija (na primer, ICMP paketima) u odnosu na datagrame iz drugih aplikacija (na primer, sa mrežnim novostima).

Kako smo već napomenuli, poređenje slike 4.24 sa slikom 4.13 otkriva uproščenu, mnogo jednostavniju građu datagrama IPv6. U protokolu IPv6 definisana su sledeća polja:

- *Verzija.* Ovo polje od 4 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 protokol IPv6).

- *Klasa saobraćaja.* Ovo 8-bitno polje slična je po duhu 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 neoznačen ceo broj, kojim se daje broj bajtova u IPv6 datagramu koji slede iza 40-bajtnog zaglavlja stalne dužine.
- *Sledeće zaglavlje.* 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 zaglavlju protokola IPv4.
- *Ograničenje broja skokova.* Sadržaj ovog polja umanjuje se za jedan, nasvakom rutera koji prosledi datagram. Kada vrednost ovog polja postane jednaka nuli, datagram se odbacuje.
- *Adrese izvora i odredišta.* Različiti formati 128-bitne adrese protokola IPv6 opisani su u dokumentu RFC 4291.
- *Podaci.* Ovo je deo sa korisnim podacima u IPv6 datagramu. Kada datagram stigne na svoje odredište, korisni podaci se izdvajaju iz IP datagrama i predaju protokolu, navedenom u polju za sledeće zaglavlje.

U prethodnom opisu navedene su svrhe polja iz datagrama IPv6. Ako uporedimo format datagrama IPv6 na slici 4.24 sa formatom datagrama 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 fragmentaciju i ponovno sastavljanje u usputnim ruterima; ti poslovi mogu da se obavljaju samo na izvoru i na odredištu. Ruter koji primi IPv6 datagram koji je previše veliki za prosleđivanje izlaznim linkom, jednostavno odbacuje taj datagram i pošiljaocu vraća ICMP poruku o grešci: „Packet Too Big" (više o tome nešto kasnije). Pošiljalac tada može ponovo da pošalje podatke, koristeći manji IP datagram. Fragmentacija i ponovno sastavljanje troše mnogo vremena; uklanjanje ovih zadataka iz rutera i njihovo prebacivanje na krajnje sisteme znatno ubrzava IP prosleđivanje unutar mreže.
- *Kontrolni zbir zaglavlja.* Pošto protokoli transportnog sloja (na primer, TCP i UDP) i protokoli sloja veze podataka (na primer, Eternet) proveravaju kontrolne zbirove, projektanti protokola IP verovatno su smatrali da je taj zadatak suvišan na mrežnom sloju i da se može ukloniti. I ovde je brza obrada IP paketa bila glavna briga. Sećate se iz opisa protokola IPv4 u odeljku 4.4.1 da kontrolni zbir mora ponovo da se izračunava na svakom ruteru, pošto zaglavlje protokola IPv4 sadrži polje TTL (slično polju za organičenje broja skokova u protokolu IPv6). Za ovo se u protokolu IPv4, slično kao i za fragmentaciju i ponovo sastavljanje, troši prilično vreme.

- *Opcije.* Polje opcija više ne postoji u standardnom IP zaglavlju. Međutim, opcije nisu uklonjene. Umesto toga, polje opcija postalo je jedno od mogućih sledećih zaglavlja na koje se ukazuje iz IPv6 zaglavlja. To jest, kao što zaglavlja protokola TCP ili UDP mogu da budu sledeće zaglavlje u IP paketu, to može da bude i polje opcija. Uklanjanjem polja opcija iz zaglavlja, dobijeno je IP zaglavlje stalne dužine od 40 bajtova.

Sećate iz razmatranja u odeljku 4.4.3 da IP čvorovi koriste protokol ICMP za izveštavanje o greškama i za pružanje ograničenih informacija (na primer, za eho odgovor na ping poruku) nekom krajnjem sistemu. Za IPv6 je u dokumentu RFC 4443 definisana nova verzija protokola ICMP. Osim što su postojeće definicije ICMP tipova i kodova poruka preuređene, u protokol ICMPv6 su dodati novi tipovi i kodovi poruka koji su potrebni za nove mogućnosti koje ima IPv6. Među njima su nov tip poruke „Packet Too Big" (prevelik paket) i novi kôd greške „unrecognized IPv6 options" (nepoznate IPv6 opcije). Osim toga, ICMPv6 obuhvata i mogućnosti koje ima protokol IGMP (Internet Group Management Protocol) koji proučavamo u odeljku 4.7. Protokol IGMP, koji se koristi kada računar pristupa ili napušta takozvane grupe za višestruko slanje, u protokolu IPv4 je zaseban protokol u odnosu na ICMP.

Prelazak sa protokola IPv4 na protokol IPv6

Sada, pošto smo videli tehničke detalje protokola IPv6, razmotrimo jedno veoma praktično pitanje: kako će javni internet koji se zasniva na protokolu IPv4, preći na IPv6? Problem je u tome što su novi sistemi koji podržavaju protokol IPv6 kompatibilni unazad, odnosno mogu da šalju, usmeravaju i primaju IPv4 datagrame, dok postojeći sistemi koji podržavaju protokol IPv4 ne mogu da rukuju datagramima IPv6. Postoji nekoliko rešenja [Huston 2011b].

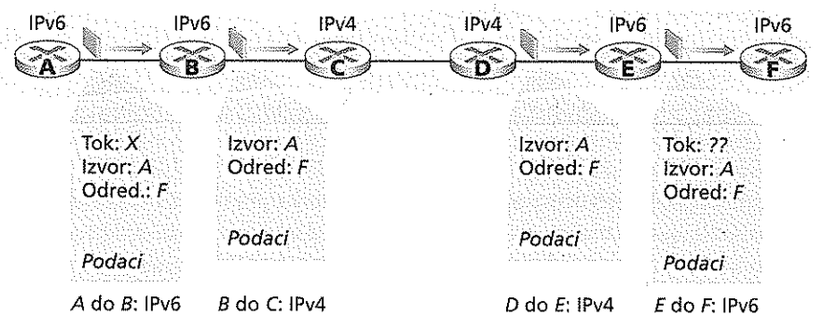
Jedno rešenje bi bilo da se neki dan proglasi kao granični datum – određeni dan i vreme kada bi se sve mašine na internetu isključile i sa protokola IPv4 prešle na protokol IPv6. Poslednja velika promena tehnologije (kada se za uslugu pouzdanog transporta prešlo sa NCP na TCP) dogodila se pre skoro 25 godina. Čak i tada [RFC 801], kada je internet bio u povoju i kada je njime upravljao mali broj „genijalaca", znalo se da takav granični dan nije moguć. Granični dan koji bi obuhvatio na stotine miliona mašina i milione mrežnih administratora i korisnika danas je još teže izvodljiv. RFC 4213 opisuje dva pristupa (koji mogu da se koriste samostalno ili zajedno) 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 uvođenja čvorova koji podržavaju IPv6 jeste rešenje sa **dvostrukim stekom** (eng. dual-stack), po kome IPv6 čvorovi takođe potpuno podržavaju i protokol IPv4. Takav čvor, koji se u dokumentu RFC 4213 naziva IPv6/IPv4 čvor, sposoban je da šalje i prima obe vrste datagrama. Kada operiše sa IPv4 čvorom, IPv6/IPv4 čvor koristi IPv4 datagrame; kada se interakcija obavlja sa IPv6 čvorom, govori jezikom IPv6 datagrama. IPv6/IPv4 čvorovi moraju da imaju kako IPv4, tako i IPv6 adresu. Pored toga moraju da budu u stanju da odrede da li drugi čvor podržava IPv6 ili samo IPv4. Ovaj problem može se rešiti korišćenjem

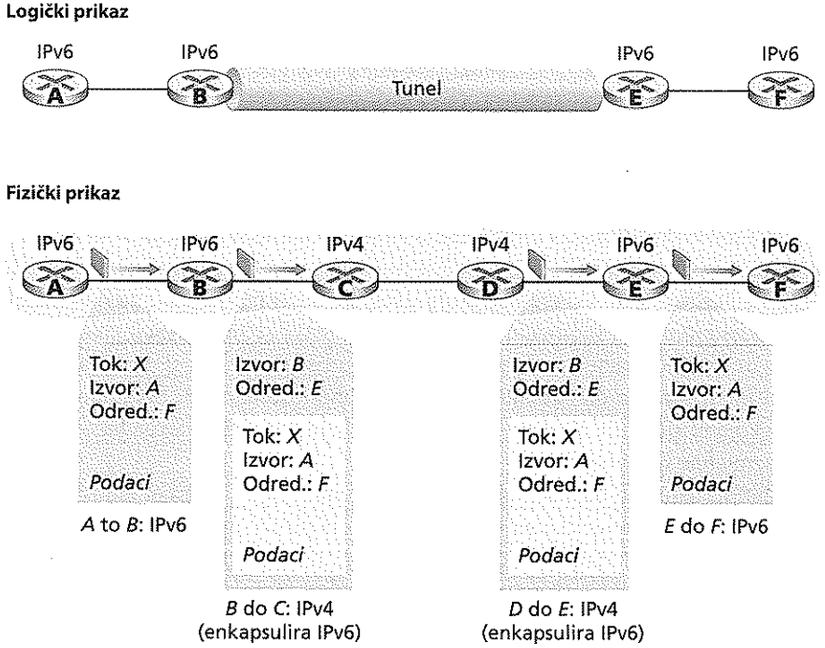
DNS usluge (pogledajte poglavlje 2), koja može da vrati IPv6 adresu, ako traženi čvor podržava IPv6, a inače vraća IPv4 adresu. Naravno, ako čvor koji izdaje DNS zahtev podržava samo IPv4, DNS server vraća samo IPv4 adresu.

U rešenju sa dvostrukim stekom, bilo da pošiljalac ili primalac podržavaju samo IPv4, mora se koristiti IPv4 datagram. Zbog toga je moguće da dva čvora koja podržavaju IPv6 na kraju u suštini jedan drugome šalju IPv4 datagrame. Ovo je prikazano na slici 4.25. Uzmimo da čvor A podržava IPv6 i želi da pošalje IP datagram čvoru F, koji takođe podržava 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 sa podacima iz IPv6 datagrama može da se prekopira u polje sa podacima IPv4 datagrama, a adrese mogu da se preslikaju na odgovarajući način. Međutim, prilikom prevođenja iz protokola IPv6 u IPv4 postoje polja kojih ima samo u IPv6 datagramu (na primer, polje oznake 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 E stignu iz D, ne sadrže sva polja koja su postojala u prvobitnom IPv6 datagramu koji je poslat iz A.

Drugačiji pristup u odnosu na rešenje sa dvostrukim stekom, koji se takođe opisuje u dokumentu RFC 4213, poznat je kao **tunelovanje**. Tunelovanje može da reši gore navedeni problem, jer omogućava, na primer, da E primi IPv6 datagram nastao u čvoru A. Osnovna ideja tunelovanja je sledeća. Pretpostavimo da dva IPv6 čvora (na primer, B i E sa slike 4.25) žele da komuniciraju i koriste IPv6 datagrame, ali su međusobno povezani preko posrednih IPv4 rutera. Usputne IPv4 rutere između dva IPv6 rutera nazivamo **tunel**, kao što je prikazano na slici 4.26. Kada se koristi tunelovanje, IPv6 čvor na predajnoj strani tunela (na primer, B) uzima *ceo* IPv6 datagram i stavlja ga u polje (korisnih) podataka IPv4 datagrama. Taj IPv4 datagram se zatim upućuje do IPv6 čvora na prijemnoj strani tunela (na primer, E) i šalje prvom čvoru u tunelu (na primer, C). Usputni IPv4 ruteri u tunelu prenose ovaj IPv4 datagram međusobno, 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, izvlači IPv6 datagram i zatim ga usmerava dalje baš kao da ga je primio od neposredno povezanog IPv6 suseda.



Slika 4.25 ◆ Rešenje sa dvostrukim stekom



Slika 4.26 ◆ Tunelovanje

Ovaj odeljak zaključujemo napomenom da je prihvatanje protokola IPv6 u početku bilo sporo [Lawton 2001], a da je zamah počeo tek nedavno. Videti [Huston 2008b] za objašnjenja o razmeštanju protokola IPv6 za 2008. godinu; videti [NIST IPv6 2012] za kratak prikaz razmeštanja protokola IPv6 u Sjedinjenim Državama. Širenje uređaja kao što su telefoni sa mogućnošću pristupanja internetu i drugih prenosnih uređaja predstavljaju dodatni podstrek za šire prihvatanje protokola IPv6. U

Evropi, udruženje mobilnih partnera treće generacije (Thrid Generation Partnership Program) [3GPP 2012] propisalo je IPv6 kao standard koji se koristi za mobilne multimedijalne uređaje.

Jedna važna lekcija koju možemo naučiti iz iskustva sa protokolom IPv6 jeste da je užasno teško menjati protokole mrežnog sloja. Još od ranih devedesetih godina za niz novih protokola mrežnog sloja na sav glas se tvrdilo da jesledeća velika revolucija na internetu, ali je većina njih do danas stekla tek ograničen uticaj. Tu spadaju IPv6, protokoli za višeznačno slanje (odeljak 4.7) i protokoli za rezervisanje resursa (poglavlje 7). Zaista, uvođenje novih protokola na mrežni sloj slično je zameni temelja kuće – to se teško postiže bez rušenja cele građevine ili bar privremenog raseljavanja njenih stanara. S druge strane, internet je doživeo nagło uvođenje više novih protokola na aplikativnom sloju. Klasični primeri, naravno, jesu veb, slanje instant poruka i P2P deljenje datotela. Još neki primeri su protok audio i video signala u realnom vremenu i distribuirane igre. Uvođenje novih protokola aplikativnog sloja je kao krečenje kuće – relativno se lako izvodi i ako ste izabrali privlačnu boju, isto će uraditi i neki vaši susedi. Ukratko, u budućnosti se mogu očekivati promene na mrežnom sloju interneta, ali će te promene verovatno biti mnogo sporije od promena na aplikativnom sloju.

4.4.5 Kratak pregled bezbednosti IP protokola

U odeljku 4.4.3 protokol IPv4 je obrađen prilično podrobno, uključujući i usluge koje nudi i to kako se te usluge ostvaruju. Dok ste čitali taj odeljak, možda ste zapazili da nijednom nisu spomenute bezbednosne usluge. I zaista, IPv4 stvoren je u doba (1970. godine) kada su internet uglavnom koristili istraživači u oblasti mreža koji nisu imali razloga da sumnjanju u međusobne loše namere. Stvaranje mreže računara koja bi sjedinila mnoštvo tehnologija bila je dovoljan izazov, tako da niko nije ni razmišljao o bezbednosti.

Međutim, danas je bezbednost jedna od glavnih briga i internet istraživači su morali da se posvete projektovanju novih protokola mrežnog sloja koji nude razne bezbednosne usluge. Jedan od takvih protokola je IPsec, jedan od najpopularnijih bezbednosnih protokola mrežnog sloja i takođe jedan od najviše zastupljenih u virtuelnim privatnim mrežama (engl. *Virtual Private Network*, VPN). Mada se protokol IPsec i osnove kriptografije na kojoj se zasniva podrobnije obrađuju u poglavlju 8, u ovom odeljku ukratko dajemo najopštiji prikaz usluga protokola IPsec.

IPsec je projektovan tako da bude kompatibilan sa protokolima IPv4 i IPv6. Tačnije, da bismo uživali u plodovima protokola IPsec, ne moramo da zamenimo skupove protokola na *svim* ruterima i računarima na internetu. Na primer, ukoliko dva računara žele da bezbedno komuniciraju, koristeći transportni režim rada (jedan od dva „režima rada" protokola IPsec), neophodno je da IPsec bude dostupan samo na ta dva računara. Svi ostali ruteri i računari mogu da nastave da izvršavaju čisti IPv4 protokol.

Da bismo bili jasniji, ovde ćemo se usredsrediti na transportni režim rada protokola IPsec. U ovom režimu, dva računara prvo međusobno uspostavljaju IPsec sesiju. (Stoga je IPsec protokol sa uspostavljanjem veze!). Kada se veza uspostavi, svi TCP i UDP segmenti koji se šalju između ova dva računara uživaju u bezbednosnim uslugama koje nudi IPsec. Na predajnoj strani, transportni sloj predaje segmente do Ipsec protokola. IPsec zatim šifruje segmente, dopunjava ih dodatnim bezbednosnim poljima i dobijene korisne podatke enkapsulira u običan IP datagram. (U suštini se ovo radi nešto složenije, kao što ćemo videti u poglavlju 8). Predajni računar zatim šalje datagrame na internet, koji ih prenosi do odredišnog računara. IPsec tamo dešifruje segmente i nešifrovane segmente prenosi do transportnog sloja.

Usluge koje nudi IPsec obuhvataju:

- *Dogovor za šifrovanje.* Mehanizam koji računarima koji komuniciraju omogućava da se dogovore o algoritmima koji se koriste za šifrovanje i ključevima za šifrovanje.
- *Šifrovanje korisnih podataka IP datagrama.* Kada predajni računar primi segment od transportnog sloja, IPsec šifruje korisne podatke. Dešifrovanje može da obavi samo IPsec na prijemnom računaru.
- *Integritet podataka.* IPsec omogućava da prijemni računar proveri da li su polja zaglavlja i šifrovani korisni podaci promenjeni, dok su datagrami putovali od izvora do odredišta.
- *Autentifikacija izvora.* Kada računar primi IPsec datagram od izvora u koga ima poverenje (sa ključem poverenja – pogledajte poglavlje 8), taj računar je siguran da je IP adresa izvora navedena u datagramu zaista stvarni izvor datagrama.

Kada dva računara međusobno uspostave IPsec sesiju, svi TCP i UDP segmenti koje oni međusobno šalju bivaju šifrovani i autentifikovani. IPsec prema tome nudi potpunu zaštitu, obezbeđujući sve podatke koji se razmenjuju između ta dva računara za sve mrežne aplikacije.

Preduzeća mogu da koriste IPsec za bezbednu komunikaciju preko nebezbednog javnog interneta. Da bismo to pojasnili, ovde ćemo razmotriti jednostavan primer. Uzmimo na primer kompaniju koja ima veliki broj trgovačkih putnika, pri čemu svi oni imaju laptop računare. Pretpostavimo da je tim trgovačkim putnicima često potrebno da proveravaju poverljive informacije kompanije (na primer, informacije o cenama i proizvodima) koje se nalaze na serveru u sedištu kompanije. Dalje, pretpostavimo da trgovački putnici takođe moraju da međusobno šalju poverljiva dokumenta. Kako to mogu da obave koristeći IPsec? Kao što i sami pretpostavljate, instaliraćemo IPsec na server i na svim laptop računarima koje koriste trgovački putnici. Kada se IPsec instalira na svim tim računarima, uvek kada neki trgovački putnik želi da komunicira sa serverom ili sa drugim trgovačkim putnikom, ta komunikacija će biti bezbedna.

4.5 Algoritmi rutiranja

Do sada smo u ovom poglavlju uglavnom istraživali prosleđivanje na mrežnom sloju. Naučili smo da kada neki paket stigne do rutera, ruter pretražuje tabelu prosleđivanja i određuje na koji interfejs linka bi trebalo usmeriti taj paket. Takođe smo

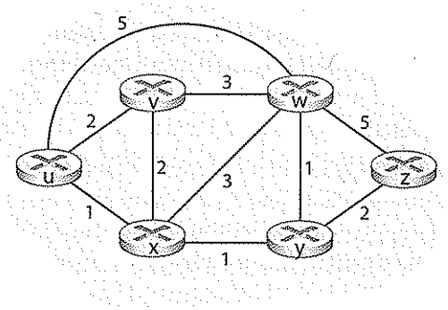
naučili da algoritmi rutiranja, prilikom rada u mrežnim ruterima, razmenjuju i izračunavaju informacije koje se koriste za konfigurisanje tih tabela prosleđivanja. Veza između algoritma rutiranja i tabele prosleđivanja prikazana je na slici 4.2. Pošto smo istražili prosleđivanje, sada posvećujemo pažnju drugoj važnoj temi ovog poglavlja, vrlo važnom poslu koji se obavlja na mrežnom sloju, tačnije, rutiranju. Bez obzira na to da li mrežni sloj pruža uslugu datagrama (kada između datog para izvora i odredišta različiti paketi mogu da idu različitim rutama), ili uslugu virtuelnog kola (kada svi paketi između datog izvora i odredišta idu uvek istom putanjom), mrežni sloj ipak mora da odredi putanju kojom paketi prolaze od pošiljaoca do primaoca. Videćemo da je zadatak rutiranja da se utvrde dobre putanje (tj. rute) od pošiljalaca do primalaca kroz mrežu rutera.

Računar se obično neposredno priključuje na jedan ruter, **podrazumevani ruter** računara (takođe se naziva i **ruter prvog skoka** za taj računar). Kad god računar pošalje paket, taj paket se prenosi do njegovog podrazumevanog rutera. Podrazumevani ruter izvornog računara nazivamo **izvorni ruter**, a podrazumevani ruter odredišnog računara **odredišni ruter**. Problem rutiranja paketa od izvornog do odredišnog računara jasno se svodi na problem rutiranja paketa od izvornog do odredišnog rutera, a to je tema ovog odeljka.

Svrha algoritma rutiranja je jednostavna: za dati skup rutera sa linkovima koji ih povezuju, algoritam rutiranja pronalazi „dobru" putanju od izvornog do odredišnog rutera. Obično je dobra putanja ona koja najmanje košta. Videćemo, međutim, da po zamisli jednostavne i elegantne algoritme na kojima se zasniva rutiranje u današnjim mrežama u praksi komplikuju stvari iz sveta u kome živimo, kao što su pravne smetnje (na primer, pravilo kao što je: „ruter x koji pripada organizaciji Y ne bi trebalo da prosleđuje pakete koji potiču iz mreže u vlasništvu organizacije Z").

Za formulisanje algoritama rutiranja koristi se teorija grafova. Sećate se da je **graf** $G = (N, E)$ skup od N čvorova i skupod E ivica, gde svaka ivica predstavlja par čvorova iz N. U smislu rutiranja na mrežnom sloju, čvorovi u grafu predstavljaju rutere – tačke na kojima se donose odluke o prosleđivanju paketa – a ivice koje povezuju te čvorove predstavljaju fizičke linkove između rutera. Takva grafička predstava računarske mreže prikazana je na slici 4.27. Grafove koji predstavljaju mape stvarnih mreža možete naći u [Dodge 2012, Cheswick 2000]; razmatranje o tome kako se internet može modelovati prilično različitim modelima, zasnovanim na grafovima, nalazi se u [Zegura 1999, Faloutsos 1999, Li 2004].

Kao što je prikazano na slici 4.27, ivica takođe ima vrednost, koja predstavlja njene troškove. Obično, ovi troškovi mogu da zavise od fizičke dužine odgovarajućeg linka (na primer, prekookeanski link može da ima veće troškove od kratkog zemaljskog linka), od brzine linka, ili novčane cene pridružene tom linku. Za naše trenutne potrebe, uzećemo date troškove linka i nećemo brinuti o tome kako se oni utvrđuju. Za bilo koju ivicu (x,y) u E, troškove ivice između čvorova x i y označićemo sa $c(x, y)$. Ako par (x, y) ne pripada E, stavićemo da je $c(x, y) = \infty$. Takođe, sve vreme razmatramo samo neusmerene grafove (tj. grafove čije ivice nemaju smer), tako da je ivica (x, y) isto što i ivica (y, x) i da je $c(x, y) = c(y, x)$. Osim toga, za čvor y kažemo da je **susedan** čvoru x, ako (x, y) pripada E.



Slika 4.27 ◈ Apstraktni model računarske mreže u obliku grafa

Pošto su ivicama u ovom prikazu u obliku grafa pridruženi troškovi, prirodan cilj algoritma rutiranja je da se pronađu putanje sa najmanjim troškovima od izvora do odredišta. Da bi problem bio što određeniji, prisetimo se da **putanju** u grafu $G = (N, E)$ sačinjava niz čvorova $(x_1, x_2, ..., x_p)$, tako da svaki par (x_1, x_2), (x_2, x_3), ..., (x_{p-1}, x_p) predstavlja ivicu iz E. Troškovi putanje $(x_1, x_2, ..., x_p)$ su prosto zbir troškova svih ivica na putanji, odnosno, $c(x_1, x_2) + c(x_2, x_3) + ... + c(x_{p-1}, x_p)$. Za proizvoljan par čvorova x i y obično postoji više putanja između ta dva čvora, pri čemu svaka od njih ima svoje troškove. Jedna od ovih putanja (ili više njih) je **putanja najmanjih troškova**. Problem najmanjih troškova je stoga očigledan: pronaći putanju između izvora i odredišta koja ima najmanje troškove. Na primer, na slici 4.27, putanja sa najmanjim troškovima od izvornog čvora u do odredišnog čvora w je (u, x, y, w) sa troškovima putanje 3. Obratite pažnju na to da, ako su troškovi svih ivica isti, putanja najmanjih troškova je takođe i **najkraća putanja** (tj. putanja sa najmanjim brojem linkova između izvora i odredišta).

Za početak, pokušajte da pronađete putanju najmanjih troškova od čvora u do čvora z na slici 4.27 i razmislite malo o tome kako ste izračunali tu putanju. Ako razmišljate slično većini ljudi, putanju od čvora u do čvora z utvrdili ste tako što ste posmatrali sliku 4.27, isprobali nekoliko putanja od u do z i na neki način došli do zaključka da putanja koju ste izabrali ima najmanje troškove 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 se obavlja na jednom mestu, u Vašoj glavi, gde se nalaze sve informacije o mreži. U opštem slučaju, način na koji možemo razvrstati algoritme rutiranja je po tome da li su globalni ili decentralizovani.

- **Globalni algoritam rutiranja** izračunava putanju najmanjih troškova između izvora i odredišta, koristeći potpuna, globalna saznanja o mreži. Drugim rečima, ovaj algoritam uzima veze između svih čvorova i troškove svih linkova kao ulazne vrednosti. To znači da algoritam mora nekako da pribavi te informacije, pre nego što obavi izračunavanja. Samo izračunavanje može da se

izvršava na jednom mestu (centralizovani globalni algoritam rutiranja), ili da se obavlja uporedo na više mesta. Ključna odlika globalnog algoritma je u tome da ima potpune informacije o vezama i troškovima linkova. U praksi, algoritmi sa potpunim informacijama o stanju mreže obično se nazivaju **algoritmi stanja linkova** (**LS** – link state), pošto algoritam mora da zna troškove svih linkova u mreži. LS algoritme proučavamo u odeljku 4.5.1.

- U **decentralizovanom algoritmu rutiranja**, izračunavanje putanje najmanjih troškova izvršava se iterativno, na više mesta, to jest distribuirano. Nijedan čvor nema potpune informacije o troškovima svih mrežnih linkova. Umesto toga, svaki čvor počinje samo od saznanja o troškovima linkova koji su neposredno povezani sa njim. Zatim, iterativnim postupkom izračunava i razmenjuje informacije sa susednim čvorovima (odnosno, čvorovima koji se nalaze na suprotnoj strani linkova sa kojima je čvor neposredno povezan); čvor postepeno izračunava putanju najmanjih troškova do jednog ili više odredišta. Decentralizovani algoritam rutiranja koji proučavamo u odeljku 4.5.2. poznat je kao algoritam vektora rastojanja (distance vector, DV) zato što svaki čvor održava vektor procenjenih troškova (rastojanja) do svih ostalih čvorova u mreži.

Drugi opšti način za razvrstavanje algoritama rutiranja je prema tome da li su statički ili dinamički. U **statičkim algoritmima rutiranja** rute se menjaju veoma retko, obično posredstvom ljudi (na primer, čovek ručno uređuje tabelu prosleđivanja rutera). U **dinamičkim algoritmima rutiranja** putanje rutiranja menjaju se promenom opterećenja saobraćaja na mreži ili promenom topologije mreže. Dinamički algoritam može da se pokrene povremeno ili kao neposredan odgovor na promene topologije ili troškova linkova. Mada se dinamički algoritmi mnogo bolje ponašaju pri promenama u mreži, oni su takođe i podložniji problemima kao što su petlje rutiranja i stalne promene ruta.

Treći način za razvrstavanje algoritma rutiranja je prema tome da li su osetljivi na opterećenje. Kod **algoritma osetljivog na opterećenje**, troškovi linkova menjaju se dinamički, zavisno od trenutnog nivoa zagušenja određenog linka. Ako se linku koji je trenutno zagušen pridruži visoki trošak, algoritam rutiranja teži tome da bira rute kojima zaobilazi taj zagušeni link. Prvobitni algoritmi rutiranja ARPAnet mreže bili su osetljivi na opterećenje [McQuillan 1980], ali se pojavio niz poteškoća [Huitema 1998]. Algoritmi rutiranja današnjeg interneta (kao što su RIP, OSPF i BGP) **nisu osetljivi na opterećenje**, jer troškovi linka ne odražavaju tačno trenutni (ili nedavni) nivo zagušenja.

4.5.1 Algoritam rutiranja zasnovan na stanju linkova (LS)

Sećate se da se u algoritmu stanja linkova znaju topologija mreže i troškovi svih linkova, odnosno da su dostupni kao ulazni podaci u LS algoritam. U praksi se ovo postiže tako što svaki čvor difuzno šalje pakete o stanju linkova *svim* ostalim čvorovima u mreži, pri čemu svaki od ovih paketa sadrži podatke i troškove o linkovima

sa kojima je taj čvor povezan. U praksi (na primer, u protokolu rutiranja OSPF koji obrađujemo u odeljku 4.6.1) ovo se obično postiže algoritmom za **difuzno slanje stanja linkova** [Perlman 1999]. Algoritme za difuzno slanje obrađujemo u odeljku 4.7. Krajnji rezultat difuznog slanja stanja linkova jeste da će svi čvorovi imati istovetan i potpun uvid u stanje mreže. Na taj način, svaki čvor može da pokrene LS algoritam i da izračuna isti skup putanja najmanjih troškova kao i svaki drugi čvor.

Algoritam stanja linkova koji ćemo sada predstaviti poznat je kao *Dijkstrin algoritam*, nazvan prema njegovom izumitelju. Primov algoritam je veoma sličan; u [Cormen 2001] naći ćete opšti opis algoritama iz teorije grafova. Dijkstrin algoritam izračunava putanju najmanjih troškova od jednog čvora (izvora, koji ćemo nazvati u) do svih ostalih čvorova u mreži. Dijkstrin algoritam je iterativan i ima svojstvo da su posle k-te iteracije algoritma poznate putanje najmanjih troškova do k odredišnih čvorova, a među putanjama najmanjih troškova do svih odredišnih čvorova tih k putanja imaju k najmanjih troškova. Definišimo sledeće oznake:

- $D(v)$: troškovi putanje najmanjih troškova od izvornog čvora do odredišta v za određenu iteraciju algoritma;
- $p(v)$: prethodni čvor (sused čvora v) na putanji sa trenutno najmanjim troškovima od izvora do v;
- N': podskup čvorova; v je deo skupa N', ukoliko je putanja najmanjih troškova od izvora do v konačno utvrđena.

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 svih drugih čvorova u mreži.

Algoritam stanja linkova za izvorni čvor u:

```
1  Initialization:
2    N'={u}
3    forallnodesv
4    ifvisaneighborofu
5       thenD(v)=c(u,v)
6    elseD(v)=∞
7
8  Loop
9    findwnotinN'suchthatD(w)isaminimum
10   addwtoN'
11    updateD(v)foreachneighbor v ofw andnotinN':
12          D(v)=min(D(v),D(w)+c(w,v))
13    /*new cost to v is either old cost to v or known
14     least path cost to w plus cost from w to v*/
15 until N'=N
```

Video napomena

Dijkstrin algoritam: iskusija i primer

Kao primer, razmotrimo mrežu na slici 4.27 i izračunajmo putanje najmanjih troškova od u do svih mogućih odredišta. Tabelarni pregled izračunavanja po ovom algoritmu prikazan je u tabeli 4.3, u kojoj svaki red tabele daje vrednosti promenljivih algoritma na kraju odgovarajuće iteracije. Razmotrimo podrobnije prvih nekoliko koraka.

- U početnom koraku, inicijalizaciji, trenutno poznatim putanjama najmanjih troškova od u do suseda sa kojima je neposredno povezan, v, x i w dodeljuju se početne vrednosti 2, 1 odnosno 5. Posebno obratite pažnju na to da su troškovi do w postavljeni na 5 (mada ćemo uskoro videti da postoji putanja sa manjim troškovima), jer je to trošak linka koji neposredno (jednim skokom) povezuje u i w. Troškovi do y i z postavljeni su na beskonačno, zato što oni nisu neposredno povezani sa u.
- U prvoj iteraciji pregledamo čvorove koje još nismo dodali skupu N' i pronalazimo čvor sa najmanjim troškovima na kraju prethodne iteracije. Taj čvor je x sa troškovima 1 i stoga se x dodaje skupu N'. Tada se izvršava 12. red LS algoritma čime se ažurira vrednost $D(v)$ za sve čvorove v, čime se dobijaju rezultati prikazani u drugom redu (korak 1) tabele 4.3. Troškovi putanje do v nisu se promenili. Pronađeno je da troškovi putanje do w (koji su na kraju početnog koraka bili 5) imaju vrednost 4 preko čvora x. Zato se bira ova putanja sa manjim troškovima, a prethodni čvor na najkraćem putu od u do čvora w postaje x. Slično tome, izračunato je da troškovi do čvora y (preko x) iznose 2, a tabela se ažurira u skladu sa tim.
- U drugoj iteraciji pokazalo se da čvorovi v i y imaju putanje sa najmanjim troškovima (2), pa proizvoljno rešavamo nerešeni rezultat i u skup N' dodajemo y, tako da N' sada sadrži u, x i y. Troškovi do preostalih čvorova koji još nisu u N', odnosno, do čvorova v, w i z ažuriraju se preko 12. reda LS algoritma, tako da se dobijaju rezultati prikazani u trećem redu tabele 4.3.
- I tako dalje...

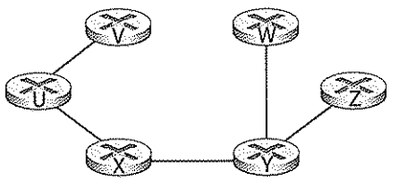
Korak	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Tabela 4.3 ◆ Izvršavanje algoritma stanja linkova za mrežu sa slike 4.27

Kada se LS algoritam završi, imamo, za svaki čvor, njegovog prethodnika duž putanje najmanjih troškova 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 prosleđivanja u nekom čvoru, recimo čvoru u, može da se napravi od ovih informacija, tako što se za svako odredište čuva čvor sledećeg skoka duž putanje najmanjih troškova od u do određenog odredišta. Na slici 4.28 prikazane su dobijene putanje najmanjih troškova i tabela prosleđivanja u čvoru u za mrežu sa slike 4.27.

Kolika je složenost ovog algoritma za izračunavanje? Odnosno, za datih n čvorova (ne računajući izvor), koliko izračunavanja bi trebalo izvršiti u najgorem slučaju, da bi se pronašle putanje najmanjih troškova od izvora do svih odredišta? U prvoj iteraciji moramo da pretražimo svih n čvorova da bismo odredili čvor w koji ima najmanje troškove, a nije u skupu N'. U drugoj iteraciji moramo da proverimo $n - 1$ čvorova, da bismo utvrdili najmanje troškove, u trećoj iteraciji proveravamo $n - 2$ čvorova i tako redom. Sve zajedno, ukupan broj čvorova koje moramo da ispitamo u svim iteracijama je $n(n + 1)/2$, pa stoga kažemo da prethodna realizacijaLS algoritma ima složenost u najgorem slučaju reda n na kvadrat: $O(n^2)$. (Promišljenijom primenom ovog algoritma, u kojoj se koristi struktura podataka poznata kao hip (eng. heap), moguće je pronaći najmanju vrednost u devetom redu u logaritamskom umesto linearnom vremenu, čime se složenost smanjuje).

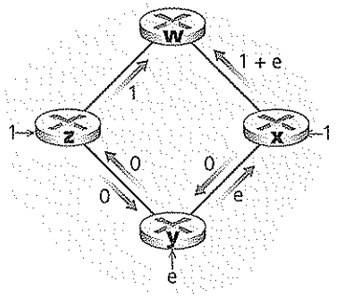
Pre nego što zaključimo opis LS algoritma, razmotrićemo nepoželjan slučaj do koga može doći. Na slici 4.29 prikazana je topologija jednostavne mreže, gde su troškovi linkova jednaki opterećenju koje se tim linkom prenosi, odnosno, odražavaju kašnjenje koje bi moglo da se javi. U ovom primeru, troškovi linkova nisu simetrični; tj. $c(u,v)$ jednako je $c(v,u)$, samo ako su na linku (u,v) opterećenja u oba smera jednaka. U ovom primeru, iz čvora z polazi jedinica saobraćaja upućena čvoru w, iz čvora x takođe kreće jedinica saobraćaja upućena čvoru w, a čvor y ubacuje količinu saobraćaja jednaku e, takođe upućenu čvoru w. Početno rutiranje prikazano je na slici 4.29(a), gde troškovi linkova odgovaraju količini saobraćaja koji se njima prenosi.



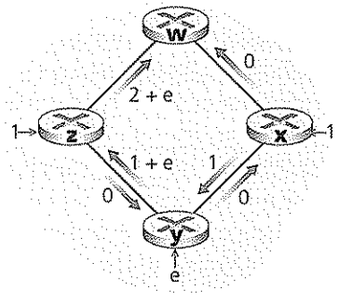
Odredište	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Slika 4.28 ◆ Putanje najmanjih troškova i tabela prosleđivanja za čvor u

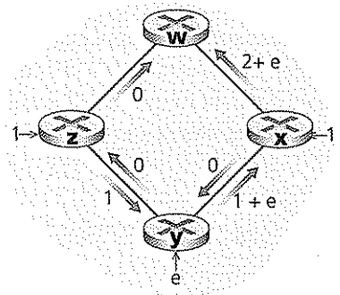
Kada se LS algoritam sledeći put pokrene, čvor y utvrđuje (na osnovu troškova linkova prikazanih na slici 4.29(a)) da putanja do w u smeru kazaljke na satu ima trošak 1, dok putanja do istog odredišta u smeru suprotnom od kazaljke na satu (koja se dotle koristila) ima trošak $1 + e$. Zbog toga je putanja najmanjih troškova od y do w sada u smeru kazaljke na satu. Slično tome, x utvrđuje da je njegova putanja najmanjih troškova do w takođe u smeru kazaljke na satu, pa dobijamo troškove kao na slici 4.29(b). Pri sledećem izvršavanju LS algoritma, čvorovi x, y, i z otkrivaju putanju bez troškova do w u smeru suprotnom od kazaljke na satu i svi rutiraju svoj saobraćaj rutama u smeru suprotnom od kazaljke na satu. Kod sledećeg izvršavanja LS algoritma, čvorovi x, y, i z rutiraju svoj saobraćaj u smeru kazaljke na satu.



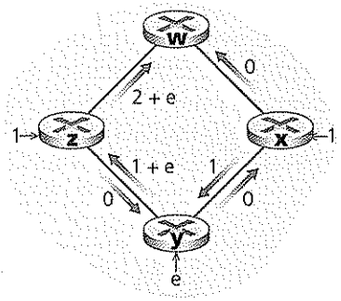
a. Početno rutiranje



b. *x* i *y* 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 4.29 ◆ Oscilacije kod rutiranja osetljivog na zagušenje

Kako ove oscilacije mogu da se spreče (koje se javljaju u svakom algoritmu, ne samo u LS algoritmu, koji koristi metriku zagušenja ili metriku koja se zasniva na kašnjenju)? Jedno rešenje bi bila naredba da troškovi linka ne zavise od količine

prenošenog saobraćaja – neprihvatljivo rešenje s obzirom da je jedan cilj rutiranja izbegavanje visoko zagušenih likova (na primer, zbog velikog kašnjenja). Drugo rešenje je obezbediti da svi ruteri ne izvršavaju LS algoritam u isto vreme. Ovo izgleda kao razumnije rešenje, jer se nadamo da čak iako ruteri izvršavaju LS algoritam u isto vreme, da izvršna instanca algoritma neće biti ista na svakom čvoru. Zanimljivo je da su istraživači otkirli da ruteri na intrenetu mogu da se međusobno samosinhronizuju [Floyd Synchronization 1994]. To znači, da čak i ako u početku izvršavaju algoritam u istom vremenskom periodu, ali u različito vreme, instanca izvršenja alogoritma može možda da postane i ostane sinhronizovana na ruterima. Jedan od načina za izbegavanje samosinhronizacije je da svaki ruter nasumično odredi vreme kada šalje oglašavanje linka.

Pošto smo prostudirali LS algoritam, posvetimo se drugim značajnim algoritmima rutiranja koji se danas koriste u praksi – algoritam rutiranja sa vektorom rastojanja.

4.5.2 Algoritam rutiranja vektora rastojanja

Dok LS algoritam koristi globalne informacije, algoritam **vektora rastojanja** (DV) je iterativan, asinhron i distribuiran. *Distribuiran* je po tome što svaki čvor prima određene informacije od jednog ili od više svojih suseda sa kojima je *neposredno povezan*, obavlja proračune, a zatim distribuira rezultate svojih proračuna svojim susedima. *Iterativan* je po tome što se ovaj postupak ponavlja, sve dok ima informacija za razmenu među susedima. (Zanimljivo je da se ovaj algoritam sâm prekida – ne postoji znak da bi izračunavanje trebalo prekinuti; ono prosto prestaje.) Algoritam je *asinhron* po tome što nije neophodno da svi čvorovi rade usaglašeno. Videćemo da je asinhroni, iterativni, samoprekidajući, distribuirani algoritam daleko zanimljiviji i zabavniji od centralizovanog algoritma!

Pre nego što predstavimo DV algoritam, biće korisno da opišemo važan odnos koji postoji između troškova putanja najmanjih troškova. Neka $d_x(y)$ bude trošak putanje najmanjih troškova od čvora x do čvora y. Odnos između najmanjih troškova dat je čuvenom Belman-Fordovom jednačinom, tačnije:

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}, \qquad (1.1)$$

gde se min_v u jednačini odnosi na sve susede čvora x. Belman-Fordova jednačina je očigledna sama po sebi. Zaista, ako posle puta od x do v krenemo putanjom najmanjih troškova od v do y, troškovi putanje biće $c(x,v) + d_v(y)$. Pošto moramo da počnemo putovanjem do nekog suseda čvora v, najmanji trošak od x do y je najmanja vrednost od $c(x,v) + d_v(y)$, uzeta za sve susede čvora v.

Ali, zbog onih koji možda sumnjaju u tačnost ove jednačine, proverimo je za izvorni čvor u i odredišni čvor z sa slike 4.27. Izvorni čvor u ima tri suseda: v, x i w. Prolazeći različitim putanjama na grafu, očigledno je $d_v(z) = 5$, $d_x(z) = 3$ i $d_w(z) = 3$. Ubacivanjem ovih vrednosti u jednačinu 4.1, uz troškove $c(u,v) = 2$, $c(u,x) = 1$ i $c(u,w) = 5$ dobijamo $d_u(z) = \min\{2 + 5, 5 + 3, 1 + 3\} = 4$, što je očigledno tačno i što nam je dao i Dijkstrin algoritam za ovu mrežu. Ova brza provera trebalo bi da odstrani sve sumnje koje ste možda imali.

Belman-Fordova jednačina nije samo teoretski zanimljiva. Ona u stvari ima izuzetan praktičan značaj. Tačnije, rešenje Belman-Fordove jednačine pruža podatke za tabelu prosleđivanja čvora x. Da bismo se u to uverili, neka v^* bude susedni čvor za koji je postignuta minimalna vrednost u jednačini 4.1. Stoga, ako čvor x želi da pošalje paket čvoru y duž putanje najmanjih troškova, trebalo bi prvo da prosledi paket do čvora v^*. Prema tome, u tabeli prosleđivanja čvora x kao ruter prvog skoka za krajnje odredište y bio bi naveden čvor v^*. Još jedan značajan praktičan doprinos Belman-Fordove jednačine jeste to što se njome nagoveštava kakav će se oblik komunikacije među susedima primeniti u DV algoritmu.

Osnovna zamisao je sledeća. Svaki čvor $x$počinje od $D_x(y)$, procenjenih troškova putanje najmanjih troškova od tog čvora do čvora y, za sve čvorove u N. Neka $\boldsymbol{D}_x = [D_x(y): y \text{ u } N]$ bude vektor rastojanja čvora x, a to je vektor procenjenih troškova od x do svih ostalih čvorova, y, u N. U DV algoritmu, svaki čvor x održava sledeće informacije rutiranja:

- za svakog suseda v, trošak $c(x,v)$ od x do neposredno povezanog suseda v;
- vektor rastojanja čvora x, tj. $\boldsymbol{D}_x = [D_x(y): y \text{ u } N]$, koji sadrži procenjene troškove do svih odredišta y, u N;
- vektore rastojanja svih svojih suseda, tj. $\boldsymbol{D}_v = [Dv(y): y \text{ u } N]$ svih suseda v čvora x.

U distribuiranom, asinhronom algoritmu svaki čvor, s vremena na vreme, šalje kopiju svog vektora rastojanja svim svojim susedima. Kada čvor x primi novi vektor rastojanja od nekog svog suseda v, on čuva taj vektor rastojanja i zatim pomoću Belman-Fordove jednačine ažurira vlastiti vektor rastojanja na sledeći način:

$$D_x(y) = \min_v\{c(x,v) + D_v(y)\} \quad \text{za svaki čvor } y \text{ u } N$$

Ako se vektor rastojanja čvora x prilikom ovog koraka ažuriranja promeni, čvor x šalje svoj ažurirani vektor rastojanja svim svojim susedima, koji sa svoje strane ažuriraju svoje vektore rastojanja. Čudesno je to da, ukoliko svi čvorovi nastave da razmenjuju svoje vektore rastojanja ovako asinhrono, svaki procenjeni trošak $D_x(y)$ teži vrednosti $d_x(y)$, stvarnim troškovima putanje najmanjih troškova od čvora x do čvora y [Bersekas 1991]!

Algoritam vektora rastojanja

Na svakom čvoru, x:

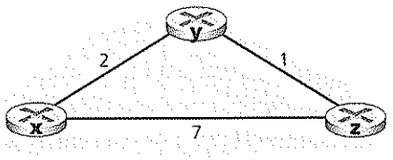
```
1  Initialization:
2    for all destination y in N:
3      Dx(y) = C(x,y)   /* if y is not a neighbor then c(x,y) = ∞+ */
4    for each neighbor w
5      Dw(y) = ∞ for all destination y in N
6    for each neighbor w
7      send distance vector Dx = [Dx(y): y in N] to w
8
9  loop
10   wait (until I see a link cost change to some neighbor w or
11     until I receive a distance vector from some neighbor w)
12
13   for each y in N:
14     Dx(y) = minv{Dx(y + Dv(y)}
15
16   if Dx(y) changed for any destination y
17     send distance vector Dx = [Dx(y): y in N] to all neighbors
18
19 forever
```

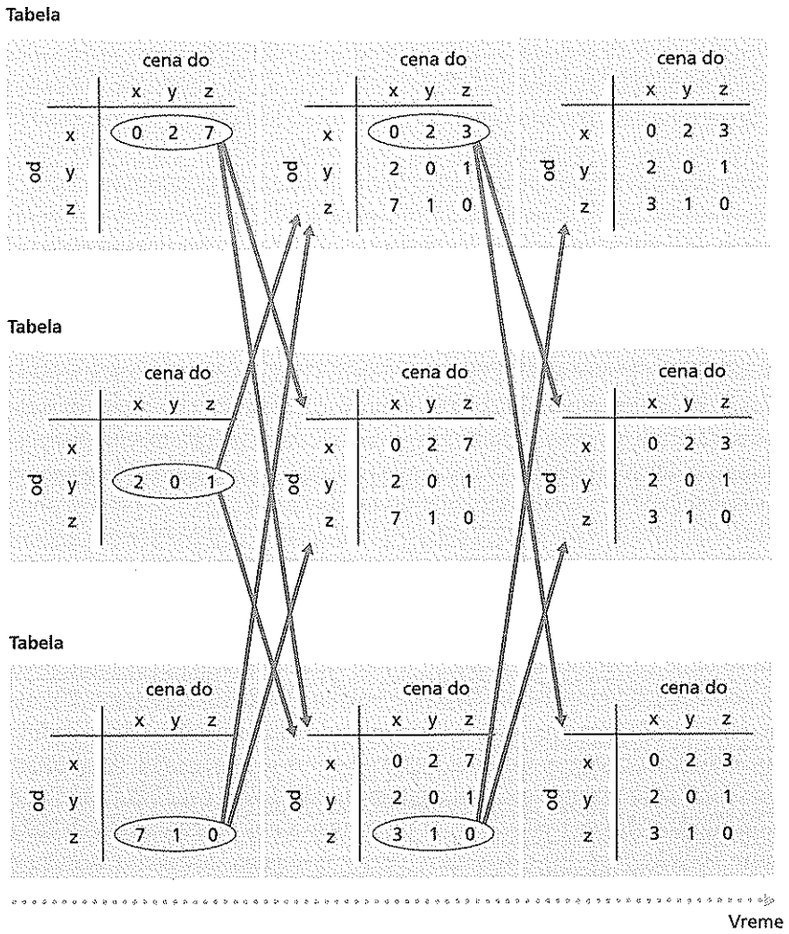
U DV algoritmu, čvor x ažurira svoj procenjeni vektor rastojanja, kada primeti promenu troškova na nekom od linkova sa kojima je neposredno povezan, ili kada od nekog suseda primi ažurirani vektor rastojanja. Ali, da bi vlastitu tabelu prosleđivanja ažurirao za dato odredište y, čvoru x nije potrebno da zna najkraće rastojanje do y, već susedni čvor $v^*(y)$, odnosno ruter prvog skoka duž najkraće putanje do y. Kao što biste i očekivali, ruter prvog skoka $v^*(y)$ je sused v koji postiže minimalnu vrednost u 14. redu DV algoritma. (Ako više suseda v ostvaruje isti minimum, tada $v^*(y)$ može da bude bilo koji od njih). Prema tome, u 13. i 14. redu čvor x, za svako odredište y, takođe utvrđuje $v^*(y)$ i ažurira svoju tabelu prosleđivanja za odredište y.

Sećate se da je LS algoritam globalan u tom smislu da svaki čvor mora prvo da pribavi potpunu mapu mreže, pre nego što pokrene Dijkstrin algoritam. DV algoritam je *decentralizovan* i ne koristi takve globalne informacije. Zaista, jedine informacije u svakom čvoru su troškovi linkova kojima je neposredno povezan sa svojim susedima i informacije koje prima od tih suseda. Svaki čvor čeka ažurirane vrednosti od bilo kog suseda (redovi 10 i 11), izračunava nov vektor rastojanja kada primi ažurne podatke (red 14) i svoj novi vektor rastojanja distribuira susedima (redovi 16 i 17). Algoritmi nalik DV algoritmu u praksi se koriste u mnogim protokolima rutiranja, uključujući protokole interneta RIP i BGP, ISO IDRP, Novelov protokol IPX i prvobitni ARPAnet protokol.

Na slici 4.30 prikazan je rad DV algoritma za jednostavnu mrežu od tri čvora, prikazanu na vrhu slike. Prikazan je sinhronizovani način rada algoritma, pri čemu 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. Kada proučite ovaj primer, trebalo bi da se uverite u to da se algoritam pravilno odvija i za asinhroni način rada, u kome se izračunavanja u čvorovima kao i generisanje i primanje ažurnih informacija događa u bilo kom trenutku.





Slika 4.30 ◆ Algoritam vektora rastojanja – DV algoritam

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 određene tabele rutiranja, svaki red je jedan vektor rastojanja – tačnije, tabela rutiranja svakog čvora sadrži njegov vlastiti vektor rastojanja i vektore rastojanja njegovih suseda. Prema tome, prvi red početne tabele rutiranja čvora x je $\boldsymbol{D}_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$. Drugi i treći red ove tabele su poslednji primljeni vektori rastojanja od čvorova y odnosno z. Pošto u početku čvor x nije još ništa primio od čvorova y i z, stavke u drugom i trećem redu imaju početnu vrednost beskonačno.

Posle prvog koraka, svi čvorovi šalju svoje vektore rastojanja svakom od svoja dva suseda. Ovo je na slici 4.30 prikazano strelicama koje polaze iz prve kolone tabela prema drugoj koloni tabela. Na primer, čvor x šalje svoj vektor rastojanja $\boldsymbol{D}_x = [0, 2, 7]$ čvorovima y i z. Pošto primi ažurirane podatke, svaki čvor ponovo izračunava svoj vektor rastojanja. Na primer, čvor x izračunava:

$$D_x(x) = 0$$

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min 2 + 0{,}7 + 1\} = 2$$

$$D_x(z) = \min\ \{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min 2 + 1{,}7 + 0\} = 3$$

U drugoj koloni su prema tome, za svaki čvor, prikazani nov vektor rastojanja odgovarajućeg čvora i vektori rastojanja koje je upravo primio od suseda. Obratite, na primer, pažnju na to da se procena čvora x za najmanji trošak do čvora z, $D_x(z)$, promenila sa7 na 3. Takođe obratite pažnju na to da za čvor x, susedni čvor y dobija minimalnu vrednost u 14. redu DV algoritma; prema tome, u ovoj fazi algoritma, za čvor x imamo da je $v^*(y) = y$ i $v^*(z) = y$.

Kada čvorovi ponovo izračunaju svoje vektore rastojanja, ažurirane vektore rastojanja ponovo šalju svojim susedima (ukoliko je došlo do promene). Ovo je na slici 4.30 prikazano strelicama iz druge kolone tabela u treću kolonu tabela. Primetićete da samo čvorovi x i z šalju ažurirane vrednosti: vektor rastojanja čvora y nije se promenio i zato on ne šalje ništa. Nakon što prime ažuriranja, čvorovi ponovo izračunavaju svoje vektore rastojanja i ažuriraju svoje tabele rutiranja, što je prikazano u trećoj koloni.

Postupak primanja ažuriranih vektora rastojanja od suseda, ponovnog izračunavanja vrednosti u tabeli rutiranja i obaveštavanja suseda o promenjenim troškovima putanje najmanjih troškova do nekog odredišta se nastavlja, sve dok ima poruka za ažuriranje. U tom trenutku, pošto se ne šalju nikakve poruke sa novim vrednostima, tabele rutiranja se više ne preračunavaju i algoritam ulazi u stanje mirovanja; odnosno, svi čvorovi izvršavaju naredbu za čekanje u 10. i 11. redu DV algoritma. Algoritam ostaje u stanju mirovanja, dok ne dođe do promene troškova nekog linka, o čemu govorimo u nastavku.

Algoritam vektora rastojanja: promene troškova linkova i otkazivanje linkova

Kada čvor koji izvršava DV algoritam otkrije promenu u trošku linka između sebe i suseda (redovi 10 i 11), on ažurira svoj vektor rastojanja (redovi 13 i 14) i, ako je došlo do promene u troškovima putanje najmanjih troškova, obaveštava svoje susede (redovi 16 i 17) o svom novom vektoru rastojanja. Na slici 4.31(a) prikazan je slučaj kada se troškovi linka od y do x promene od 4 na 1. Ovde ćemo se usredsrediti samo na vrednosti tabela rastojanja čvorova y i z do odredišta x. DV algoritam dovodi do sledećeg niza događaja:

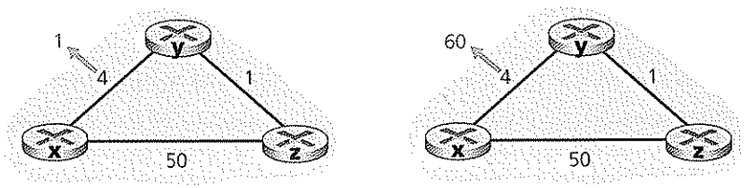
- u trenutku t_0, y otkriva promenu troška linka (trošak se promenio od 4 na 1), y ažurira svoj vektor rastojanja i obaveštava svoje susede o toj promeni, nakon što promeni svoj vektor rastojanja;
- u trenutku t_1, z prima ovu ažuriranu vrednost od y i ažurira svoju tabelu, izračuna novi najmanji trošak do x (smanjio se od 5 na 2) i svojim susedima šalje svoj novi vektor rastojanja;
- u trenutku t_2, y prima ažuriranu vrednost od z i ažurira svoju tabelu rastojanja, pošto se u čvoru y najmanji troškovi nisu promenili, y ne šalje nikakvu poruku čvoru z; algoritam prelazi u stanje mirovanja.

Potrebne su, znači, samo dve iteracije DV algoritma, da bi se dostiglo stanje mirovanja. Radosne vesti o smanjenim troškovima između x i y brzo su se proširile kroz mrežu.

Razmotrimo sada šta se događa ako trošak linka *poraste*. Pretpostavimo da troškovi linka između x i y porastu od 4 na 60, kao što je prikazano na slici 4.31(b).

1. Pre promene troška linka imamo $D_y(x) = 4$, $D_y(z) = 1$, $D_z(y) = 1$ i $D_y(x) = 5$. U trenutku t_0, y otkriva promenu troškova linka (trošak se promenio od 4 na 60). Čvor y izračunava da njegova nova putanja najmanjih troškova do x ima trošak od

$$D_y(x) = \min \{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min \{60+0, 1+5\} = 6$$



a. b.

Slika 4.31 ◆ Promene troškova linkova

Naravno, pošto imamo celovit uvid u mrežu, vidimo da ti novi troškovi preko z jesu *pogrešni*. Ali, čvor y jedino ima informaciju da neposredni troškovi do x iznose 60, a da je zadnje obaveštenje koje je y dobio od čvora z glasilo da se od z do x može stići sa troškovima 5. Zato, da bi stigao do x, y sada rutira preko z, smatrajući da od z do x može da se stigne sa troškovima 5. Od trenutka t_1 imamo **petlju rutiranja** – da bi dosegli 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 lutaće između ta dva čvora zauvek (ili dok se njihove tabele prosleđivanja ne promene).

2. Pošto je y izračunao nove najmanje troškove prema x, on o svom novom vektoru rastojanja obaveštava z u trenutku t_1.
3. Nedugo posle t_1, z prima novi vektor rastojanja čvora y, u kome je navedeno da su najmanji troškovi y do x jednaki 6; z zna da može stići do čvora y sa troškovima 1 i izračunava da novi najmanji trošak do x iznosi $D_z(x) = \min \{50+0, 1+6\} = 7$. Pošto su se najmanji troškovi čvora z do x povećali, on o tome obaveštava y u trenutku t_2.
4. Na sličan način, kada primi novi vektor rastojanja čvora z, y tada utvrđuje da je $D_y(x) = 8$ i čvoru z šalje svoj vektor rastojanja. Čvor z zatim utvrđuje da je $D_z(x) = 9$ i šalje svoj vektor rastojanja čvoru y i tako redom.

Dokle će taj proces trajati? Videćete da će petlja imati 44 iteracije (razmena poruka između y i z) – sve dok z u jednom trenutku ne izračuna da su troškovi njegove putanje preko y veći od 50. U tom trenutku, z će (konačno!) utvrditi da je njegova putanja najmanjih troškova do x preko njegove neposredne veze sa x. Čvor y će tada prema čvoru x rutirati preko čvora z. Loše vesti o povećanju troškova linka zaista sporo putuju! Šta bi se dogodilo da se trošak linka $c(y,x)$ promenio od 4 na 10 000, a da je trošak $c(z,x)$ bio 9 999? Zbog takvih slučajeva, problem koji smo sada videli ponekad se naziva problem brojanja do beskonačnosti.

Algoritam vektora rastojanja: dodavanje lažnog rastojanja

Poseban slučaj sa petljom koju smo upravo opisali može da se izbegne korišćenjem tehnike poznate kao *lažno rastojanje* (engl. *poisoned reverse*). Zamisao je jednostavna – ako z rutira preko y, da bi se stiglo do odredišta x, onda z obaveštava y da je njegovo rastojanje do x beskonačno, odnosno z obaveštava y da je $D_z(x) = \infty$ (iako z zna da je u stvari $D_z(x) = 5$); z nastavlja da obmanjuje y, sve dok prema x rutira preko y. Pošto y veruje da od z nema putanje prema x, y neće ni da pokuša da rutira do x preko z, dok z nastavlja da rutira do x preko y (i laže dok to radi).

Da pogledamo sada kako lažno rastojanje rešava određen problem petlje koji smo imali na slici 4.31(b). Zahvaljujući lažnom rastojanju, tabela rastojanja u čvoru y ukazuje da je $D_z(x) = \infty$. Kada se trošak linka (x,y) promeni od 4 na 60 u trenutku t_0, y ažurira svoju tabelu i nastavlja da rutira neposredno prema x, uprkos višim troškovima od 60, a obaveštava z o toj novoj ceni prema x, odnosno, $D_y(x) = 60$. Nakon što primi tu ažuriranu vrednost u trenutku t_1, z odmah prebacuje rutu do x

tako da bude neposredno povezano preko linka (z, x) sa troškovima od 50. Pošto je ovo nova putanja najmanjih troškova do 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 ažuriranu vrednost od z, y ažurira svoju tabelu rastojanja sa $D_y(x) = 51$. Isto tako, pošto se z sada nalazi na putanji najmanjih troškova čvora y prema x, y lažno predstavlja rastojanje od z prema x, tako što obaveštava z u trenutku t_3 da je $D_y(x) = \infty$ (iako y zna da je u stvari $D_y(x) = 51$).

Da li lažno rastojanje rešava problem brojanja do beskonačnosti u opštem slučaju? Ne. Potrebno je da pokažete da petlje u kojima učestvuju tri ili više čvorova (a ne samo dva neposredno susedna čvora) nije moguće otkriti tehnikom lažnog rastojanja.

Poređenje LS i DV algoritama rutiranja

DV i LS algoritmi imaju potpuno različite pristupe za izračunavanje rutiranja. U DV algoritmu, svaki čvor razgovara *samo* sa svojim, neposredno povezanim susedima, ali svojim susedima pruža procene putanja najmanjih troškova od njega do *svih* ostalih čvorova u mreži (za koje zna). U LS algoritmu, svaki čvor razgovara sa *svim* ostalim čvorovima (difuznim slanjem poruka), ali im saopštava *samo* troškove linkova sa kojima je neposredno povezan. Proučavanje LS i DV algoritama zaključićemo kraćim poređenjem nekih njihovih svojstava. Podsećamo da je N skup čvorova (rutera), a da je E skup ivica (linkova).

- *Složenost poruka.* Videli smo da je za LS algoritam potrebno da svaki čvor zna troškove svih linkova u mreži. Da bi se to postiglo, potrebno je da se pošalje O(|N| |E|) poruka. Takođe, kad god se promeni trošak nekog linka, novi trošak linka mora da se pošalje svim čvorovima. DV algoritam pri svakoj iteraciji zahteva razmenu poruka između neposredno povezanih suseda. Videli smo da vreme potrebno da se algoritam dovede do kraja zavisi od mnogo činilaca. Kada se promene troškovi linka, DV algoritam širi posledice te promene, samo ako novi trošak linka menja putanju najmanjih troškova prema nekom od čvorova povezanih tim linkom.
- *Brzina konvregencije.* Videli smo da naša realizacija LS algoritma jeste algoritam složenosti O(|N|2), koji zahteva O(|N| |E|) poruka. DV algoritam može sporo da se dovede do kraja, to jest sporo da konvergira, i može da ima petlje rutiranja dok se ne završi. U DV algoritmu takođe može doći do problema brojanja do beskonačnosti.
- *Robusnost.* Šta može da se dogodi ako ruter otkaže, pogrešno se ponaša, ili je napadnut? Ako se koristi LS algoritam, ruter može da difuzno pošalje pogrešne troškove za jedan link povezan sa njim (ali ne i za druge linkove). Čvor takođe može da ošteti ili odbaci pojedine LS pakete koje je primio o stanju linkova. Ali, LS čvor izračunava samo vlastite tabele prosleđivanja; ostali čvorovi vrše slična izračunavanja sami za sebe. To znači da su izračunavanja ruta u LS algoritmu donekle razdvojena, što im daje određeni stepen robusnosti. U DV algoritmu, čvor može da objavi pogrešne putanje najmanjih troškova do bilo kojeg ili do svih odredišta. (To se zaista i dogodilo 1997. godine, kada je pokvareni ruter u malom posredniku za internet usluge ruterima državne okosnice pružio pogrešne informacije o rutiranju. Zbog toga su drugi ruteri preplavili neispravni ruter saobraćajem, što je dovelo do toga da veliki delovi interneta nekoliko sati ostanu nepovezani [Neumann 1997]). Uopšteno govoreći, primećujemo da se, pri svakoj iteraciji, rezultati izračunavanja koja izvrši jedan čvor u DV algoritmu predaju njegovom susedu, a zatim posredno i susedu njegovog suseda pri sledećoj iteraciji. Na taj način, korišćenjem DV algoritma neispravan rezultat iz jednog čvora može da se proširi kroz čitavu mrežu.

Konačni zaključak je da nijedan od ovih algoritama nije bolji od drugog; zaista, na internetu se koriste oba algoritma.

Ostali algoritmi rutiranja

Algoritmi LS i DV koje smo upravo obradili ne samo da su u širokoj upotrebi, već su u suštini *jedini* algoritmi rutiranja koji se danas praktično koriste na internetu. Ipak, tokom poslednjih 30 godina istraživači su predlagali mnoge algoritme rutiranja, od krajnje jednostavnih do veoma složenih i zamršenih. Jedna velika klasa algoritama rutiranja zasniva se na tome da se saobraćaj paketa posmatra kao tok između izvora i odredišta u mreži. Sa takvim pristupom, problem rutiranja može matematički da se formuliše kao problem ograničene optimizacije, poznat kao problem mrežnog toka [Bertsekas 1991]. Pomenućemo ovde još jedan skup algoritama rutiranja koji potiče iz sveta telefonije. Ovi **algoritmi rutiranja sa komutiranjem vodova** zanimljivi su za mreže sa komutiranjem paketa podataka u slučajevima kada se za svaku vezu koja se rutira preko nekog linka rezervišu resursi na tom linku (na primer, privremena memorija ili deo propusnog opsega linka). Iako formulisanje problema rutiranja možda izgleda potpuno različito od formulacije rutiranja prema najnižim troškovima koju smo videli u ovom poglavlju, postoji niz sličnosti, bar što se tiče algoritma za pronalaženje putanje (algoritma rutiranja). Pogledajte [Ash 1998; Ross 1995; Girard 1990] za podrobniji prikaz ovog područja istraživanja.

4.5.3 Hijerarhijsko rutiranje

U proučavanju LS i DV algoritama, mrežu smo posmatrali kao prost skup međusobno povezanih rutera. Nije pravljena razlika između rutera u smislu da svi ruteri izvršavaju isti algoritam rutiranja za izračunavanje putanja rutiranja kroz celu mrežu. U praksi je ovakav model koji modeluje rutere kao skup istovetnih rutera koji izvršavaju isti algoritam rutiranja previše uprošćen bar iz dva značajna razloga:

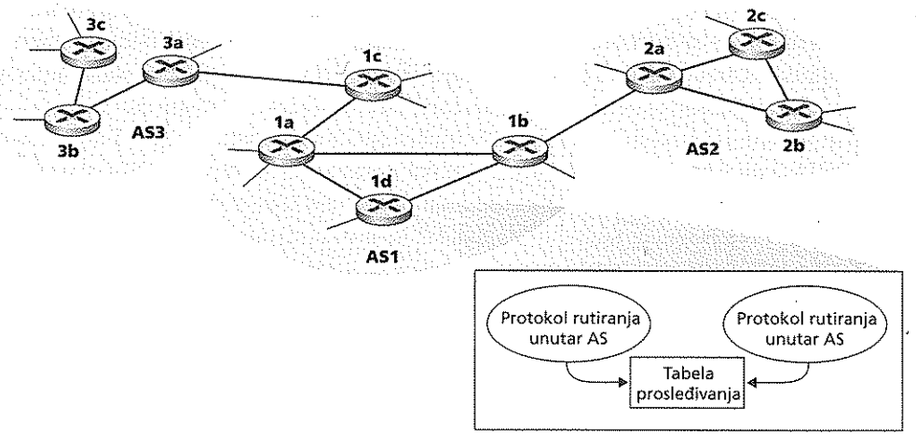
- *Proširivost.* Porastom broja rutera, deo resursa koji se troši na izračunavanje, čuvanje i razmenjivanje informacija o rutiranju (na primer, zbog ažuriranja u LS algoritmu ili zbog promena putanja najmanjih troškova) postaje zabrinja-

vajući. Današnji javni internet sastoji se od stotina miliona računara. Jasno je da bi za čuvanje informacija o rutiranju na svakom od njih bile potrebne ogromne količine memorije. Na resurse neophodne za difuzno slanje ažurnog stanja linkova u LS algoritmu svim ruterima u javnom internetu utrošio bi se sav propusni opseg, pa ništa ne bi ostalo za slanje paketa podataka! Algoritam vektora rastojanja koji bi svoje iterircije obavljao između tako velikog broja rutera sigurno nikada ne bi završio svoj posao! Jasno je da bi trebalo nešto uraditi da bi se smanjila složenost izračunavanja ruta u tako velikim mrežama kakav je javni internet.

- *Samostalnost u upravljanju.* Iako istraživači obično ne vode mnogo računa o želji kompanija da koriste rutere po vlastitoj želji (na primer, da same biraju algoritme rutiranja), ili da sakriju odnose unutar organizacije svoje mreže od spoljnog sveta, ovo su značajna pitanja. Idealno bi bilo da organizacija koristi i upravlja svojom mrežom kako želi, a da ipak može da je povezuje sa ostalim spoljašnjim mrežama.

Oba ova problema mogu da se reše tako što se ruteri organizuju u **autonomne sisteme** (engl. *autonomus systems*, **AS**) pri čemu se svaki AS sastoji od grupe rutera kojima se obično administrira sa jednog mesta (npr. njima upravlja isti posrednik za internet usluge ili pripadaju istoj organizaciji). Svi ruteri unutar istog AS sistema izvršavaju isti algoritam rutiranja (na primer, LS ili DV algoritam) i imaju informacije jedni o drugima – tačno kao u našem idealizovanom modelu iz prethodnog odeljka. Algoritam rutiranja koji se izvršava unutar autonomnog sistema nazivamo **protokol rutiranja unutar autonomnog sistema**. Naravno, neophodno je međusobno povezati AS sisteme, pa će jedan ruter ili više njih u AS sistemu imati dodatni zadatak prosleđivanja paketa na odredišta van AS sistema; ove rutere zovemo **ruteri mrežnog prolaza**.

Na slici 4.32 prikazan je jednostavan primer sa tri autonomna sistema – AS1, AS2 i AS3. Na ovoj slici, deblje linije predstavljaju neposredne linkove između parova rutera. Tanje linije koje se spuštaju od rutera predstavljaju podmreže koje su neposredno povezane sa ruterima. AS1 ima četiri rutera – 1a, 1b, 1c i 1d – koji izvršavaju protokol rutiranja unutar autonomnog sistema AS1. Prema tome, sva ova četiri rutera znaju kako da prosleđuju pakete duž optimalne putanje do svakog odredišta unutar AS1. Slično tome, autonomni sistemi AS2 i AS3 imaju po tri rutera. Obratite pažnju na to da protokoli rutiranja unutar AS sistema koji se izvršavaju u sistemima AS1, AS2 i AS3 ne moraju biti isti. Obratite takođe pažnju i na to da su ruteri 1b, 1c, 2a i 3a ruteri mrežnog prolaza.



Slika 4.32 ◆ Primer međusobno povezanih autonomnih sistema

Jasno je kako ruteri u AS sistemima utvrđuju putanje rutiranja za parove izvor-odredište koji se nalaze unutar autonomnog sistema. Ali, još uvek nije potpuno jasno kako se vrši rutiranje sa kraja na kraj. Kako ruter unutar nekog AS sistema zna kako da rutira paket na odredište izvan tog AS sistema? Lako je odgovoriti na ovo pitanje, ukoliko bi AS imao samo jedan ruter mrežnog prolaza kojim je povezan sa samo jednim od preostalih AS sistema. U tom slučaju, pošto je algoritam rutiranja unutar AS sistema utvrdio putanju najmanjih troškova od svih unutrašnjih rutera do rutera mrežnog prolaza, svi unutrašnji ruteri znaju kako da prosleđuju pakete. Ruter mrežnog prolaza, nakon što primi paket, prosleđuje paket na jedini link koji vodi izvan tog AS sistema. AS na suprotnoj strani tog linka zatim preuzima odgovornost za rutiranje paketa do njegovog konačnog odredišta. Kao primer, pretpostavimo da ruter 2b sa slike 4.32 primi paket čije se odredište nalazi izvan sistema AS2. Ruter 2b zatim prosleđuje paket, bilo ruteru 2a ili ruteru 2c, zavisno od tabele prosleđivanja rutera 2b, što se konfiguriše protokolom rutiranja unutar autonomnog sistema AS2. Paket stiže do rutera mrežnog prolaza 2a, koji će ga proslediti ruteru 1b. Pošto paket napusti ruter 2a, posao AS2 sistema je gotov, što se tiče ovog paketa.

Stoga, problem je lak, ako izvorni AS ima samo jedan link koji vodi izvan tog AS sistema. Ali, šta ako izvorni AS ima više od dva linka (kroz dva ili više rutera mrežnog prolaza) koji vode van AS sistema? Tada je mnogo teže utvrditi gde bi trebalo proslediti paket. Na primer, uzmimo ruter u sistemu AS1 i pretpostavimo da je primio paket čije se odredište nalazi izvan ovog AS sistema. Jasno je da bi ruter trebalo da prosledi paket nekom od dva mrežna prolaza, 1b ili 1c, ali kojem? Da bi se rešio ovaj problem, AS1 mora: (1) da zna do kojih se odredišta može stići preko sistema AS2, a do kojih preko sistema AS3 i (2) da proširi te informacije o dostupnosti odredišta svim ruterima u AS1, tako da svaki ruter konfiguriše svoju tabelu prosleđivanja za odredišta izvan AS sistema. Ova dva zadatka – pribavljanje

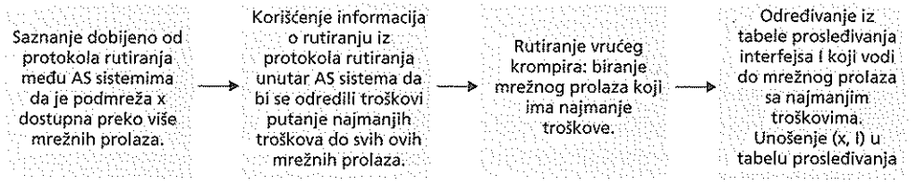
informacija o dostupnosti odredišta od susednih AS sistema i širenje tih informacija o dostupnosti svim ruterima unutar AS sistema – obavlja **protokol rutiranja među autonomnim sistemima**. Pošto protokol rutiranja među autonomnim sistemima obuhvata komunikaciju između dva AS sistema, ta dva AS sistema moraju da izvršavaju isti protokol rutiranja među autonomnim sistemima. U stvari, na internetu svi AS sistemi izvršavaju isti protokol rutiranja među autonomnim sistemima, nazvan BGP4, koji razmatramo u sledećem odeljku. Kao što se vidi na slici 4.32, svaki ruter prima informacije od protokola rutiranja unutar autonomnog sistema i protokola rutiranja među autonomnim sistemima i koristi informacije iz oba ta protokola, da bi konfigurisao svoju tabelu prosleđivanja.

Uzmimo, kao primer, jednu podmrežu x (koja se raspoznaje po adresi određenoj CIDR šemom) i pretpostavimo da AS1 sazna od protokola rutiranja među autonomnim sistemima da je podmreža x dostupna iz sistema AS3, ali da *nije* dostupna iz sistema AS2. AS1 ovu informaciju prenosi svim svojim ruterima. Kada ruter 1d sazna da do mreže x može stići preko sistema AS3, pa prema tome preko mrežnog prolaza 1c, on iz informacija dobijenih od protokola rutiranja unutar AS sistema određuje interfejs rutera, koji se nalazi na putanji najmanjih troškova od rutera 1d do rutera mrežnog prolaza 1c. Recimo da je to interfejs I. Ruter 1d u svoju tabelu prosleđivanja dodaje stavku (x, I). (Ovaj primer, kao i ostali iz ovog odeljka, u suštini tačno, ali prilično uprošćeno prikazuju ono što se zaista događa na internetu. U sledećem odeljku daćemo podrobniji, puno složeniji, opis protokola BGP).

Nadovezujući se na prethodni primer, pretpostavimo sada da se AS2 i AS3 povezuju sa drugim AS sistemima koji nisu prikazani na slici. Pretpostavimo takođe da AS1 sazna od protokola rutiranja među autonomnim sistemima da je podmreža x dostupna iz sistema AS2 preko mrežnog prolaza 1b i iz sistema AS3 preko mrežnog prolaza 1c. AS1 bi zatim proširio ovu informaciju svim svojim ruterima, među njima i ruteru 1d. Da bi konfigurisao svoju tabelu prosleđivanja, ruter 1d mora da odredi kom bi ruteru mrežnog prolaza, 1b ili 1c, trebalo da usmeri pakete koji su upućeni u podmrežu x. Jedan pristup, koji se često koristi u praksi je da se primeni tzv. **rutiranje vrućeg krompira** (eng. hot-potato routing). U rutiranju vrućeg krompira, AS pokušava da se oslobodi paketa (vrućeg krompira) što pre može (tačnije, sa što manjim troškovima). To se postiže slanjem paketa onom ruteru mrežnog prolaza koji ima najmanje troškove od rutera do mrežnog prolaza u odnosu na sve ostale mrežne prolaze koji imaju putanju prema odredištu. Za naš trenutni primer, rutiranje vrućeg krompira, koje se izvršava u ruteru 1d, na osnovu informacija dobijenih od protokola rutiranja unutar AS sistema bi utvrdilo troškove putanja do 1b i 1c i zatim izabralo putanju sa manjim troškovima. Pošto izabere putanju, ruter 1d dodaje stavku za podmrežu x u svoju tabelu prosleđivanja. Na slici 4.33 ukratko su prikazni postupci koji se preduzimaju u ruteru 1d za dodavanje nove stavke za x u tabelu prosleđivanja.

Kada neki AS od susednog AS sazna za neko odredište, on može da objavi tu informaciju za rutiranje nekim od svojih ostalih susednih AS . Na primer, pretpostavimo da AS1 sazna od sistema AS2 da je podmreža x dostupna preko sistema AS2.

AS1 bi mogao da obavesti AS3 da je podmreža x dostupna preko sistema AS1. Na ovaj način, ako bi AS3 trebalo da rutira paket sa odredištem x, AS3 bi prosledio paket do AS1, koji bi zatim prosledio paket do AS2. Kao što ćemo videti pri razmatranju protokola BGP, autonomni sistemi prilično slobodno odlučuju o tome o kojim će odredištima obavestiti susedne autonomne sisteme. To je *strateška* odluka, koja obično više zavisi od ekonomskih nego od tehničkih činilaca.



Slika 4.33 ◆ Postupak dodavanja odredišta izvan AS u tabelu prosleđivanja rutera

Sećate se da smo u odeljku 1.5 rekli da se internet sastoji od hijerarhijski međusobno povezanih posrednika za internet usluge. Kakve veze imaju ovi posrednici i autonomni sistemi? Možda mislite da ruteri jednog posrednika i linkovi koji ih međusobno povezuju čine jedan AS. Iako je obično tako, mnogi posrednici dele svoju mrežu na više autonomnih sistema. Na primer, neki posrednici prvog reda koriste jedan AS za celu svoju mrežu, dok drugi dele svoje mreže na desetine međusobno povezanih AS.

Ukratko, problemi proširivosti i samostalnosti u upravljanju rešavaju se uspostavljanjem autonomnih sistema. Unutar AS svi ruteri izvršavaju isti protokol rutiranja unutar AS. Autonomni sistemi između sebe izvršavaju isti protokol rutiranja među AS. Problem proširivanja sistema rešen je time što bi ruter unutar AS trebalo da zna samo za ostale rutere unutar svog AS. Problem samostalnosti u upravljanju rešen je time što organizacije mogu da izvršavaju protokol rutiranja unutar AS po svom izboru; međutim, parovi povezanih AS moraju da izvršavaju isti protokol rutiranja među AS kako bi mogli da razmenjuju informacije o dostupnosti.

U sledećem odeljku ispitujemo dva protokola rutiranja unutar AS (RIP i OSPF) i protokol rutiranja među AS (BGP) koji se koriste na današnjem internetu. Ovi primeri će lepo upotpuniti naša proučavanja hijerarhijskog rutiranja.

4.6 Rutiranje na internetu

Pošto smo proučili adresiranje na internetu i protokol IP, sada svoju pažnju usmeravamo na protokole rutiranja na internetu; njihov zadatak je da utvrde putanju kojom datagrami stižu od izvora do odredišta. Videćemo da protokoli rutiranja na internetu primenjuju većinu principa koje smo proučili ranije u ovom poglavlju. Rešenja sa stanjem linkova i vektorima rastojanja koje smo proučili u odeljcima 4.5.1 i 4.5.2 kao i pojam autonomnih sistema koji smo razmatrali u odeljku 4.5.3. ključni su za to kako se rutiranje obavlja na današnjem internetu.

Sećate se iz odeljka 4.5.3 da je autonomni sistem (AS) skup rutera koji su pod istom administrativnom i tehničkom kontrolom pri čemu svi oni između sebe koriste isti protokol rutiranja. Svaki AS, sa svoje strane, obično obuhvata više podmreža (ovde koristimo izraz podmreža u tačnom značenju u smislu adresiranja iz odeljka 4.4.2).

4.6.1 Rutiranje unutar autonomnih sistema na internetu: protokol RIP

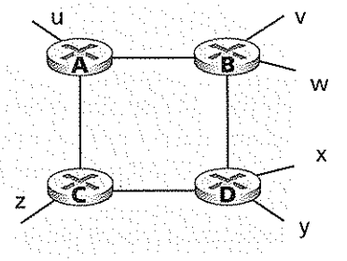
Protokol rutiranja unutar AS koristi se za to da bi se odredilo kako se vrši rutiranje unutar određenog autonomnog sistema. Protokoli rutiranja unutar AS poznati su i kao **unutrašnji protokoli mrežnih prolaza**. Na internetu se za rutiranje unutar autonomnih sistema najviše koriste dva protokola rutiranja: od ranije **RIP (Routing Information Protokol)** i **OSPF (Open Shortest Path First)** koji se pojavio nešto kasnije. Protokol rutiranja, blisko povezan sa protokolom OSPF je protokol **IS-IS** [RFC 1142, Perlman 1999]. Prvo razmatramo protokol RIP, a zatim protokol OSPF.

RIP je jedan od prvih protokola rutiranja unutar AS na internetu i danas je još uvek u širokoj upotrebi. Njegovi koreni i ime potiču iz arhitekture XNS (Xerox Network Systems). Za široku rasprostranjenost protokola RIP najvećim delom je zaslužno to što je 1982. godine bio uključen u BSD (Berkeley Software Distribution) distribuciju verzije UNIX-a koja je podržavala TCP/IP. Prva verzija protokola RIP definisana je u [RFC 1058], a druga verzija, kompatibilna sa prvom verzijom, definisana je u [RFC 2453].

RIP je protokol vektora rastojanja koji radi na način veoma sličan idealizovanom DV protokolu, koji smo razmotrili u odeljku 4.5.2. Verzija protokola RIP opisana u dokumentu RFC 1058 troškove utvrđuje prema broju skokova; odnosno, svi linkovi imaju troškove 1. U algoritmu DV u odeljku 4.5.2, zbog jednostavnosti, troškovi su određivani između parova rutera. U protokolu RIP (a takođe i u protokolu OSPF) stvarni troškovi su od izvornog rutera do odredišne podmreže. RIP koristi izraz *skok* što je u stvari broj podmreža kroz koje se prolazi duž najkraće putanje od izvornog rutera do odredišne podmreže, uključujući odredišnu podmrežu. Na slici 4.34 je prikazan AS sa šest grana podmreža. Tabela na ovoj slici prikazuje broj skokova od izvora A do svih grana podmreža.

Najveći trošak putanje ograničen je na 15, zbog čega je upotreba protokola RIP ograničena na autonomne sisteme sa manje od 15 skokova u prečniku. Sećate se da u DV protokolu, susedni ruteri međusobno razmenjuju vektore rastojanja. Vektor rastojanja za bilo koji ruter je trenutno procenjena vrednost najkraće putanje od tog rutera do podmreža u AS. U RIP protokolu se ažurirani podaci o rutiranju razmenjuju između suseda približno svakih 30 sekundi, korišćenjem **RIP poruke odgovora**. Poruka odgovora koju šalje ruter ili računar sadrži spisak od najviše 25 odredišnih podmreža unutar AS, kao i udaljenost pošiljaoca do svake od tih podmreža. Poruke odgovora takođe su poznate i kao **RIP objave** (eng. advertisment).

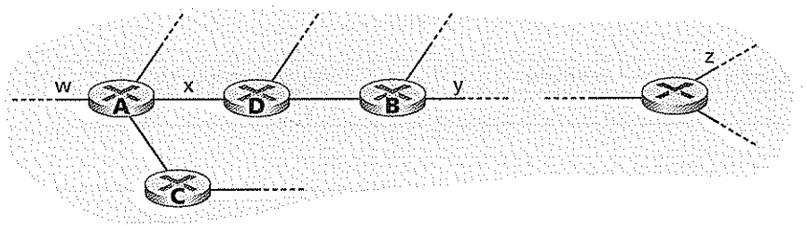
Pogledajmo jedan jednostavan primer o tome kako rade RIP objave. Uzmite na primer deo AS prikazanog na slici 4.35. Na ovoj slici linije koje povezuju rutere predstavljaju podmreže. Označeni su samo izabrani ruteri (*A*, *B*, *C* i *D*) i podmreže (*w*, *x*, *y* i *z*). Isprekidane linije označavaju da se AS širi i da ovaj autonomni sistem ima mnogo više rutera i linkova nego što je prikazano.



Odredište	Broj skokova
u	1
v	2
w	2
x	3
y	3
z	2

Slika 4.34 ◈ Broj skokova od izvornog rutera A do raznih podmreža

Svi ruteri održavaju RIP tabele poznate kao **tabele rutiranja**. Tabela rutiranja rutera sadrži vektor rastojanja odgovarajućeg rutera i njegovu tabelu prosleđivanja. Na slici 4.36 prikazana je tabela rutiranja rutera *D*. Obratite pažnju na to da tabela rutiranja ima tri kolone. Prva kolona je za odredišnu podmrežu, druga kolona navodi sledeći ruter na najkraćoj putanji prema odredišnoj mreži, a u trećoj se nalazi broj skokova (tj. broj podmreža koje bi trebalo preći, uključujući odredišnu podmrežu), da bi se najkraćim putem stiglo u odredišnu podmrežu. U ovom primeru tabela pokazuje da bi za slanje datagrama od rutera *D* do odredišne podmreže *w*, datagram trebalo prvo proslediti susednom ruteru *A*; tabela takođe pokazuje da je odredišna podmreža *w* udaljena dva skoka duž najkraće putanje. Slično tome, tabela pokazuje da je podmreža *z* udaljena sedam skokova preko rutera *B*. Po pravilu, tabela rutiranja ima po jedan red za svaku podmrežu unutar AS, mada verzija 2 protokola RIP dozvoljava udruživanje ulaznih podataka o podmrežama, korišćenjem tehnika za sažimanje ruta koje smo ispitali u odeljku 4.4. Tabela na slici 4.36 i sledeće tabele su samo delimično popunjene.



Slika 4.35 ◈ Deo autonomnog sistema

Odredišna podmreža	Sledeći ruter	Broj skokova do odredišta
w	A	2
y	B	2
z	B	7
x	—	1
....	....	....

Slika 4.36 ◈ Tabela rutiranja u ruteru *D* pre prijema objave od rutera *A*

Pretpostavimo sada da 30 sekundi kasnije, ruter *D* primi od rutera *A* objavu prikazanu na slici 4.37. Obratite pažnju na to da ova objava nije ništa drugo nego informacija iz tabele rutiranja rutera *A*! Ove informacije između ostalog ukazuju na to da je podmreža *z* udaljena samo 4 skoka od rutera *A*. Ruter *D*, nakon što primi ovu objavu, pripaja ovu objavu (slika 4.37) staroj tabeli rutiranja (slika 4.36). U ovom slučaju to znači da ruter *D* saznaje da do podmreže *z* sada postoji putanja preko rutera *A*, kraća od putanje preko rutera *B*. Stoga, ruter *D* ažurira svoju tabelu rutiranja, tako što u nju uključuje kraću najkraću putanju, kao što je prikazano na slici 4.38. Možda se pitate kako to da je najkraća putanja do podmreže *z* postala kraća? Moguće je da je decentralizovani algoritam sa vektorom rastojanja još uvek u procesu konvergencije (pogledajte odeljak 4.5.2), a moguće je i da su novi linkovi i/ili ruteri dodati u ovaj AS, pa su se stoga promenile najkraće putanje unutar AS.

Razmotrimo sada nekoliko stvari koje se dešavaju pri realizaciji protokola RIP. Sećate se da RIP ruteri razmenjuju objave približno svakih 30 sekundi. Ako ruter nema vesti od nekog svog suseda najmanje bar jednom tokom 180 sekundi, smatra da taj sused više nije dostupan; odnosno, ili je sused prestao sa radom, ili se prekinuo link koji ih povezuje. Kada se to desi, RIP menja lokalnu tabelu rutiranja i zatim tu informaciju širi dalje, tako što šalje objave susednim ruterima (onima koji su još uvek dostupni). Ruter korišćenjem RIP poruke zahteva može takođe da zahteva informaciju o troškovima svojih suseda do datog odredišta. Ruteri jedni drugima šalju RIP poruke zahteva i RIP poruke odgovora preko protokola UDP, koristeći broj porta 520. UDP segment se između rutera prenosi uobičajenim IP datagramom. Činjenica da RIP koristi protokol transportnog sloja (UDP) iznad protokola mrežnog sloja (IP), da bi ostvario zadatak koji se obavlja na mrežnom sloju (algoritam rutiranja) može da izgleda prilično uvrnuto (i jeste!). Kada malo podrobnije proučimo to kako se RIP realizuje, biće i ovo jasnije.

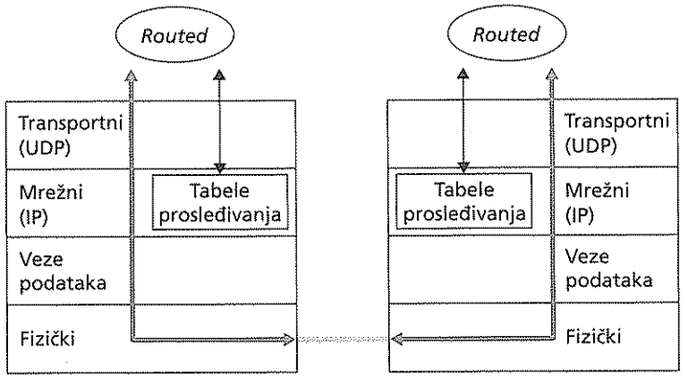
Odredišna mreža	Sledeći ruter	Broj skokova do odredišta
z	C	4
w	–	1
x	–	1
....	....	....

Slika 4.37 ◈ Objava iz rutera A

Odredišna podmreža	Sledeći ruter	Broj skokova do odredišta
w	A	2
y	B	2
z	A	5
....	....	....

Slika 4.38 ◈ Tabela rutiranja u ruteru *D* posle prijema objave od rutera *A*

Na slici 4.39 prikazana je skica kako se RIP uobičajeno implementira u UNIX sistemu, na primer, UNIX radnoj stanici koja služi kao ruter. Proces nazvan *routed* (izgovara se „rut di") izvršava protokol RIP, odnosno održava informacije o rutiranju i razmenjuje poruke sa *routed* procesima koji se izvršavaju na susednim ruterima. Pošto se RIP izvršava kao proces aplikativnog sloja (mada ne kao običan proces, već onaj koji može da upravlja tabelama rutiranja unutar UNIX kernela), može da šalje i prima poruke preko uobičajenog soketa i koristi uobičajeni transportni protokol. Kao što se vidi, RIP je ostvaren kao protokol aplikativnog sloja (pogledajte poglavlje 2) koji se izvršava iznad protokola UDP. Ako Vas interesuje implementacija protokola RIP (ili protokola OSPF i BGP koje ćemo uskoro objasniti), pogledajte [Quagga 2012].



Slika 4.39 ◈ Implementacija protokola RIP kao procesa (demon, u terminologiji UNIX sistema) *routed*

4.6.2 Rutiranje unutar autonomnih sistema na internetu: protokol OSPF

Slično protokolu RIP, i protokol OSPF se dosta koristi za rutiranje unutar AS na internetu. OSPF i njegovog bliskog srodnika, protokol IS-IS, obično koriste posrednici internet usluga višeg reda, dok se RIP koristi u mrežama posrednika nižeg reda i u mrežama preduzeća. Reč Open (eng. otvoren) u nazivu protokola OPSF znači da je specifikacija protokola rutiranja javno dostupna (za razliku, na primer, od Cisco protokola EIGRP). Najnovija verzija protokola OSPF, verzija 2, definisana je u javno dostupnom dokumentu RFC 2328.

OSPF je zamišljen kao naslednik protokola RIP, pa kao takav ima niz naprednih svojstava. U suštini, međutim, OSPF je protokol stanja linkova koji koristi plavljenje informacijama o stanju linkova i Dijkstrin algoritam putanje najmanjih troškova. Ruter koji koristi OSPF pravi potpunu topološku mapu (odnosno, graf) čitavog autonomnog sistema. Ruter zatim lokalno izvršava Dijkstrin algoritam najkraće putanje kako bi odredio najkraće putanje do *svih* podmreža, pri čemu sebe smatra za koreni čvor. Troškove pojedinačnih linkova konfiguriše administrator mreže (pogledajte izdvojeni tekst „Principi u praksi: postavljanje OSPF vrednosti linkova"). Administrator može da postavi troškove svih linkova na 1, čime se rutiranje vrši prema minimalnom broju skokova, a može da postavi tako da vrednosti linkova budu obrnuto srazmerne kapacitetu linkova, kako bi obeshrabrio da se za saobraćaj koriste linkovi manjih propusnih opsega. OSPF ne propisuje pravila za to kako se postavljaju vrednosti linkova (to je posao administratora mreže), već umesto toga obezbeđuje mehanizme (protokol) kojim se određuje putanja rutiranja najmanjih troškova za date vrednosti linkova.

Ruter koji koristi OSPF difuzno šalje informacije o rutiranju *svim* ostalim ruterima u autonomnom sistemu, a ne samo susednim ruterima. Ruter difuzno šalje informacije o stanju linkova, uvek kad dođe do promene stanja nekog linka (na primer, do promene troškova ili promene njegovog statusa). Ruter takođe povremeno difuzno šalje stanje linkova (najmanje svakih 30 minuta), bez obzira na to što se stanje linkova nije promenilo. RFC 2328 ističe da: „ovo povremeno ažuriranje objava stanja linkova povećava robusnost ovog algoritma". OSPF objave smeštaju se unutar OSPF poruka koje se neposredno prenose IP protokolom, sa kodom protokola gornjeg sloja 89 za OSPF. Zbog toga, protokol OSPF mora sâm da ostvaruje zadatke, kao što su pouzdan prenos poruka i difuzno slanje stanja linkova. Protokol OSPF takođe proverava da li su linkovi ispravni i da li rade (pomoću poruke: „HELLO", koja se šalje između susednih rutera) i omogućava OSPF ruteru da preko svog susednog rutera pribavi bazu podataka o stanju linkova u celoj mreži.

Evo nekoliko unapređenja koje donosi OSPF:

- *Bezbednost.* Moguće je proveravati autentičnost razmene poruka između OSPF rutera (na primer, slanje ažurnih stanja linkova). To znači da samo ovlašćeni ruteri mogu da učestvuju u OSPF protokolu unutar AS, čime se sprečava da zlonamerni uljezi (ili studenti umrežavanja koji svoje novostečeno znanje koriste za šale) ubace netačne informacije u tabele rutera. OSPF paketi između rutera se podrazumevano ne autentifikuju pa ih je moguće falsifikovati. Moguće je konfigurisati dve vrste provere autentičnosti – jednostavnu ili MD5 (u poglavlju 8 možete naći opšti opis provere autentičnosti i posebno MD5). Sa jednostavnom proverom autentičnosti na svim ruterima se postavlja ista lozinka. OSPF paket koji ruter šalje uključuje lozinku u obliku običnog teksta. Jasno je da jednostavna provera autentičnosti nije posebno bezbedna. Provera autentičnosti MD5 zasniva se na deljenim tajnim ključevima koji se postavljaju na svim ruterima. Ruter izračunava MD5 heš sadržaja svih OSPF paketâ i pridruženog tajnog ključa. (Pogledajte razmatranje o kodovima za autentifikaciju poruka u poglavlju 7.) Dobijenu heš vrednost ruter dodaje u OSPF paket. Prijemni ruter, koristeći unapred postavljeni tajni ključ, izračunava MD5 heš ovog paketa i upoređuje ga heš vrednošću koju nosi paket, čime proverava autentičnost paketa. Kao zaštita od napada ponavljanjem u MD5 proveri autentičnosti takođe se koriste redni brojevi.
- *Više putanja sa istim troškovima.* Kada više putanja do odredišta ima iste troškove, OSPF dozvoljava da se koristi više putanja (odnosno, ne mora da se izabere samo jedna putanja za prenošenje čitavog saobraćaja, kada postoji više putanja sa jednakim troškovima).
- *Integrisana podrška za jednoznačno i višeznačno rutiranje.* Višeznačni OSPF (Multicast OSPF, MOSPF) [RFC 1584] sadrži jednostavna proširenja protokola OSPF koja omogućavaju višeznačno rutiranje (tema koju podrobnije obrađujemo u odeljku 4.7.2). MOSPF dodaje novu vrstu obaveštavanja linkova u postojeći mehanizam protokola OSPF za difuzno slanje stanja linkova, koristeći postojeću OSPF bazu podataka o linkovima.
- *Podrška za hijerarhiju unutar istog domena rutiranja.* Možda najznačajniji napredak u protokolu OSPF jeste mogućnost da se autonomni sistem hijerarhijski struktuira. U odeljku 4.5.3 već smo uvideli mnoge prednosti hijerarhijskih struktura za rutiranje. U ostatku ovog odeljka obrađujemo OSPF hijerarhijsko rutiranje.

OSPF autonomni sistem može se hijerarhijski konfigurisati u zone. Svaka zona izvršava vlastiti OSPF algoritam rutiranja stanja linkova, pri čemu svi ruteri unutar neke zone difuzno šalju stanja svojih linkova svim ostalim ruterima u toj zoni. Unutar svih zona, jedan **granični ruter zone** (ili više njih) zadužen je za rutiranje paketa izvan zone. Jedna jedina OSPF zona u autonomnom sistemu postavlja se da bude zona **okosnice**. Prvenstvena uloga zone okosnice je da rutira saobraćaj između ostalih zona u autonomnom sistemu. Okosnica uvek obuhvata sve granične rutere zona u autonomnom sistemu, a može da sadrži i rutere koji nisu granični za zone. Rutiranje između zona unutar autonomnog sistema zahteva da se paket prvo rutira do nekog graničnog rutera zone (rutiranje unutar zona), zatim kroz okosnicu do graničnog rutera odredišne zone, a zatim do konačnog odredišta.

OSPF je relativno složen protokol i naš je opis morao da bude skraćen; više o tome možete da nađete u [Huitema 1998; Moy 1998; RFC 2328].

PRINCIPI U PRAKSI

POSTAVLJANJE OSPF VREDNOSTI LINKOVA

U razmatranju rutiranja prema stanju linkova prećutno smo polazili od pretpostavke da su vrednosti linkova postavljene, da se izvršava algoritam rutiranja kao što je OSPF i da saobraćaj teče prema tabelama rutiranja koje su izračunate LS algoritmom. U smislu uzroka i posledice, vrednosti linkova su date (odnosno, one su na početku), a iz njih proizlaze (preko Dijkstrinog algoritma) putanje rutiranja kojima se ukupni troškovi svode na najmanju meru. Tako posmatrano, vrednost linka predstavlja troškove korišćenja linka (npr. ako su vrednosti linkova obrnuto srazmerne njihovoj propusnoj moći, onda korišćenje linkova većih propusnih moći ima manju vrednost i stoga je mnogo privlačnije sa stanovišta rutiranja), a Dijkstrin algoritam služi za svođenje ukupnih troškova na najmanju meru.

U praksi, odnos između vrednosti linkova i putanja rutiranja kao uzroka i posledice može da se preokrene, pri čemu mrežni operateri postavljaju vrednosti linkova tako da postignu određene ciljeve u upravljanju saobraćajem [Fortz 2000, Fortz 2002]. Na primer, pretpostavite da mrežni operater za sve ulazne tačke ima procenu toka saobraćaja koji ulazi u mrežu sa odredištem ka svim izlaznim tačkama. Operater bi mogao da postavi određeno rutiranje tokova od ulaza ka izlazu, tako da maksimalno opterećenje svih linkova u mreži bude što manje. Za algoritam rutiranja kakav je OSPF, vrednosti linkova su osnovno „dugme" kojim operator podešava rutiranje tokova kroz mrežu. Prema tome, da bi postigao cilj da maksimalno opterećenje svih linkova u mreži bude što manje, operater mora da pronađe skup vrednosti linkova kojima postiže taj cilj. Ovo je obrnut odnos uzroka i posledice - željeno rutiranje tokova je poznato, a moraju se pronaći vrednosti OSPF linkova, takve da OSPF algoritam rutiranja proizvede željeno rutiranje tokova.

4.6.3 Rutiranje među autonomnim sistemima: protokol BGP

Upravo smo naučili kako posrednici za internet usluge koriste RIP i OSPF za utvrđivanje optimalnih putanja između izvora i odredišta koji se nalaze unutar istog autonomnog sistema. Razmotrimo sada kako se utvrđuju optimalne putanje između izvora i odredišta koji se nalaze u različitim autonomnim sistemima. **Protokol graničnog mrežnog prolaza** (Border Gateway Protocol, BGP) verzija 4, definisan u dokumentu RFC 4271 (videti takođe [RFC 4274]), na današnjem internetu je *de facto* standardni protokol za rutiranje među autonomnim sistemima. Obično se pominje kao BGP4 ili jednostavno kao **BGP**. Kao protokol rutiranja među autonomnim sistemima (videti odeljak 4.5.3), BGP autonomnim sistemima omogućava:

1. pribavljanje informacija o dostupnosti podmreža od susednih autonomnih sistema;
2. prenošenje tih informacija o dostupnosti do svih rutera unutar autonomnog sistema;
3. određivanje „dobrih" ruta do podmreža na osnovu informacija o dostupnosti i pravila autonomnog sistema.

Što je najvažnije, BGP omogućava svakoj podmreži da objavi svoja postojanja ostalima na internetu. Podmreža viče: „Postojim i ovde sam", a BGP se brine za to da svi autonomni sistemi na internetu saznaju za tu podmrežu i kako da stignu do nje. Da nema protokola BGP, svaka podmreža bi ostala izolovana – sama i nepoznata ostalima na internetu.

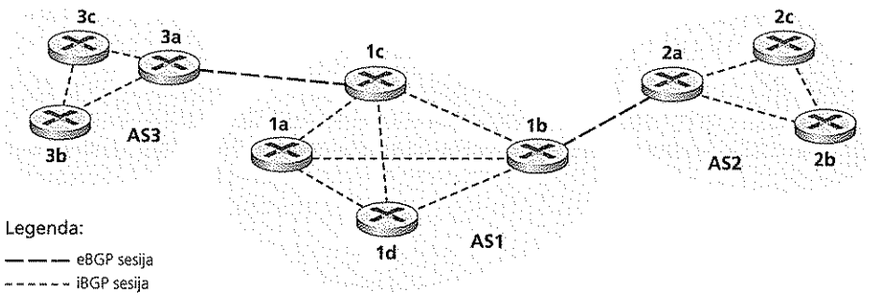
Osnove protokola BGP

Video napomena

Držanje interneta na kupu

BGP je izuzetno složen; čitave knjige posvećene su toj temi i mnogo toga je još uvek nejasno [Yannuzzi 2005]. Štaviše, čak i posle čitanja tih knjiga i odgovarajućih RFC dokumenata, teško je potpuno ovladati protokolom BGP bez višemesečnog (ako ne višegodišnjeg) praktičnog korišćenja protokola BGP u ulozi projektanta ili administratora mreža posrednika za internet usluge višeg reda. Ipak, pošto je BGP nesumnjivo najvažniji protokol za internet – u suštini, to je protokol koji drži celu stvar na okupu – moramo da steknemo bar najosnovnije pojmove o tome kako radi. Počinjemo opisom kako bi BGP mogao da radi u okviru primera jednostavne mreže, koju smo već poučavali sa slike 4.32. Ovim opisom nadovezujemo se na razmatranje hijerarhijskog rutiranja iz odeljka 4.5.3; preporučujemo da ga ponovo pročitate.

U protokolu BGP parovi rutera razmenjuju informacije o rutiranju preko polutrajnih TCP veza, korišćenjem porta 179. Polutrajne TCP veze za mrežu sa slike 4.32 prikazane su na slici 4.40. Obično postoji po jedna takva BGP TCP veza za svaki link koji neposredno povezuje dva rutera iz različitih autonomnih sistema; prema tome, na slici 4.40 imamo jednu TCP vezu između rutera mrežnih prolaza 3a i 1c i drugu TCP vezu između rutera mrežnih prolaza 1b i 2a. Postoje takođe polutrajne BGP TCP veze između rutera unutar autonomnog sistema. U ovom slučaju, na slici 4.40 prikazana je uobičajena konfiguracija od jedne TCP veze za svaki par rutera unutar AS, tako da imamo mrežu TCP veza unutar svih autonomnih sistema. Za sve TCP veze, dva rutera na krajevima veze nazivaju se **BGP ravnopravni učesnici** (eng. BGP peers), a TCP veza zajedno sa svim BGP porukama koje se šalju tom vezom naziva se **BGP sesija**. Osim toga, BGP sesija koja obuhvata dva AS naziva se **spoljašnja BGP (eBGP) sesija**, a BGP sesija između rutera u istom autonomnom sistemu naziva se **unutrašnja BGP (iBGP) sesija**. Na slici 4.40, eBGP sesije prikazane su dužim crticama; iBGP sesije prikazane su kraćim crticama. Obratite pažnju na to da se linije BGP sesija na slici 4.40 ne podudaraju uvek sa fizičkim linkovima na slici 4.32.



Slika 4.40 ◆ eBGP i iBGP sesije

PRINCIPI U PRAKSI

OBEZBEĐENJE INTERNET PRISUSTVA: SLAGANJE SLAGALICE

Pretpostavimo da ste upravo kreirali malu serversku farmu koja ima brojne servere, uključujući i javni veb server koji opisuje proizvode i usluge Vaše kompanije, server za e-poštu sa kog Vaši mogu da dobijaju svoje poruke e-pošte, kao i DNS server. Želeli biste, naravno, da ceo svet može da surfuje vašom veb lokacijom, kako bi se informisali o vašim uzbudljivim proizvodima i uslugama. Štaviše, želeli biste da Vaši zaposleni mogu da šalju i primaju e-poštu od potencijanih klijenata širom sveta.

Da biste ostvarili ove ciljeve, potrebno je da prvo uspostavite internet vezu, koja se omogućava ugovorom i povezivanjem sa lokalnim posrednikom za internet usluge. Vaša kompanija će imati ruter mrežnog prolaza, koji će biti povazan sa ruterom vašeg posrednika za internet usluge. Ova veza može biti DSL veza preko postojeće telefonske infrastrukture, iznajmljena linija do rutera posrednika za interent usluge, ili jedan od mnogih drugih pristupnih rešenja opisanih u poglavlju 1. Vaš lokalni posrednik za internet usluge će vam obezbediti jedan opseg IP adrese, na primer a/24 opseg adrese koji se sastoji iz 256 adresa. Kada uspostavite fizičku povezanost i opseg IP adrese, biće dodeljena jedna od IP adresa (u vašem opsegu adresa) vašem veb serveru, jedna sereveru e-pošte, jedna DNS serveru, jedna ruteru mrežnog prolaza, a ostale IP adrese ostalim serverima i mrežnim uređajima u kompanijskoj mreži.

Pored ugovora sa posrednikom za internet usluge, biće Vam potreban i ugovor sa internet registrarom domena, kako biste dobili ime domena za vašu kompaniju, kao što je opisano u poglavlju 2. Na primer, ako je ime kompanije, recimo, Xanadu INC., normalno je da dobijete ime domena xanadu.com. Vaša kompanija mora takođe da dobije prisustvo u DNS sistemu. S obzirom da ljudi iz spoljnog sveta žele da kontakiraju vaš DNS server i dobiju IP adrese vaših servera, biće potrebno da obezbedite registrar sa IP adresama za vaš DNS server. Registrar će tada postaviti zapis za vaš DNS server (ime domena i odgovarajuća IP adresa) na .com domenima servera višeg niova, na način opisan u poglavlju 2. Kada se ovaj korak izvrši, svaki korisnik koji zna ime vašeg domena (npr. xanadu.com) će moći da nabavi IP adresu vašeg DNS servera putem DNS sistema.

Stoga, da bi ljudi mogli da otkriju IP adrese vašeg veb servera, na vašem DNS serveru, biće potrebno da uključite zapis koji preslikava naziv računara vašeg veb servera (npr. www.xanadu.com) u njegovu IP adresu. Želećete slične zapise za ostale javno dostupne servere u vašoj kompaniji, uključujući i server e-pošte. Na ovaj način, ako Alisa želi da pretraži vaš veb server, DNS sistem će kontaktirati vaš DNS server, pronaći će IP adresu vašeg veb servera, i daće je Alisi. Alisa će tada moći da uspostavi TCP vezu direktno sa vašim veb serverom.

Međutim, još uvek nedostaje jedan neophodan i ključan korak kojim se dozvoljava spoljnim korisnicima iz čitavog sveta pristup veb serveru. Pogledajmo šta se dešava kada Alisa, koja zna IP adresu vašeg veb servra, pošalje jedan IP datagram (npr. segment TCP SYN) na ovu IP adresu. Ovaj datagram će biti rutiran preko interneta, i posećivaće serije rutera u različitim autonomnim sistemima, i konačno će doći do vašeg veb servera. Kada jedan od rutera primi datagram, tražiće odrednicu u tabeli prosleđivanja, kako bi se utvrdio na koji bi izlazni port trebalo da prosledi datagram. Stoga, svi ruteri bi trebalo da znaju za postojanje kompanijskog / 24 prefiksa (ili neki grupisanu odrednicu). Kako ruter postaje svestan kompanijskog prefiksa? Kao što smo upravo videli, postaje ga svestan iz protokola BGP (tj. opsega adrese)! Kada vaša kompanija potpiše ugovor sa lokalnim posrednikom za internet usluge i dodeli joj se prefiks (tj. opseg adrese), vaš lokalni posrednik za internet usluge će koristiti protokol BGP da objavi ovaj prefiks posrednicima za internet usluge sa kojima se povezuje. Ovi posrednici će tada, redom da koriste protokol BGP, kako bi dalje širili ovu objavu. Konačno, svi internet ruteri će znati vaš prefiks (ili neki grupisani unos koji uključuje vaš prefiks) i stoga će biti u mogućnosti da proslede datagrame na vaš veb i servere e-pošte.

BGP omogućava da svaki AS sazna koja su odredišta dostupna preko njemu susednih autonomnih sistema. Kada se koristi BGP, odredišta nisu računari već **prefiksi** određeni CIDR šemom, gde svaki prefiks predstavlja podmrežu ili skup podmreža. Tako, na primer, pretpostavimo da su za AS2 vezane četiri podmreže: 138.16.64/24, 138.16.65/24, 138.16.66/24 i 138.16.67/24. Stoga AS2 može da sažme prefikse te četiri podmreže i iskoristi BGP, kako bi autonomnom sistemu AS1 objavio jedinstveni prefiks 138.16.64/22. Kao drugi primer, pretpostavimo da se samo prve tri podmreže nalaze u autonomnom sistemu AS2, a da je četvrta, 138.16.67/24, u autonomnom sistemu AS3. Tada, kao što je objašnjeno u tekstu „Principi u praksi" u odeljku 4.4.2, pošto ruteri za prosleđivanje datagrama koriste pravilo preklapanja najdužeg prefiksa, AS3 bi mogao da objavi autonomnom sistemu AS1 određeniji prefiks 138.16.67/24, a AS2 bi *i dalje* mogao da objavljuje autonomnom sistemu AS1 grupisani prefiks 138.16.64/22.

Ispitajmo sada kako bi BGP distribuirao informacije o dostupnosti prefiksa preko BGP sesija, prikazanih na slici 4.40. Kao što i pretpostavljate, koristeći eBGP sesije između rutera mrežnih prolaza 3a i 1c, AS3 šalje u AS1 listu prefiksa koji su dostupni iz autonomnog sistema AS3; a AS1 šalje u AS3 listu prefiksa dostupnih iz autonomnog sistema AS1. Slično tome, AS1 i AS2 razmenjuju informacije o dostupnosti prefiksa preko svojih rutera mrežnog prolaza 1b i 2a. Takođe prema očekivanju, kada ruter mrežnog prolaza (u bilo kom AS) sazna prefikse preko eBGP sesija, on pomoću svojih iBGP sesija distribuira prefikse ostalim ruterima u autonomnom sistemu. Na taj način, svi ruteri u autonomnom sistemu AS1 znaju prefikse autonomnog sistema AS3, uključujući i ruter mrežnog prolaza 1b. Ruter mrežnog prolaza 1b (u AS1) može zatim da prefikse autonomnog sistema AS3 objavi autonomnom sistemu AS2. Kada ruter (mrežnog prolaza ili običan) sazna za neki novi prefiks, on u svojoj tabeli prosleđivanja pravi stavku za taj prefiks, kao što je opisano u odeljku 4.5.3.

Atributi putanje i BGP rute

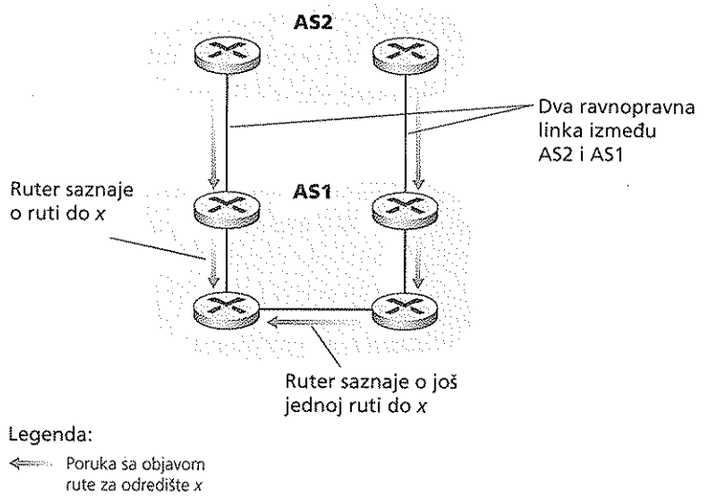
Pošto smo stekli osnovna saznanja o protokolu BGP, krenimo malo dublje (ipak ćemo i dalje pod tepih gurati neke manje važne detalje!). U protokolu BGP autonomni sistem prepoznaje se po svom globalno jedinstvenom **broju autonomnog sistema** (autonomous system number, ASN) [RFC 1930]. (Nije baš sasvim tačno da svaki AS ima svoj ASN. Tačnije, takozvani završni (eng. stub) AS koji prenosi isključivo

saobraćaj čiji je on izvor ili odredište obično nema ASN; u daljem razmatranju zanemarujemo ovu tehničku nedoslednost da nam drveće ne bi zaklonilo šumu.) AS brojevi, kao i IP adrese, dodeljuju se u regionalnim registrima ICANN [ICANN 2012].

Kada ruter kroz BGP sesiju objavljuje prefiks, on uz taj prefiks stavlja i izvestan broj **BGP atributa**. U BGP žargonu, prefiks zajedno sa svojim atributima se naziva **ruta**. Prema tome, ravnopravni BGP učesnici objavljuju jedni drugima rute. Dva najvažnija atributa su AS-PATH i NEXT-HOP.

- *AS-PATH*. Ovaj atribut sadrži autonomne sisteme kroz koje je prošla objava za odgovarajući prefiks. Kada se prefiks prenese unutar autonomnog sistema, taj AS dodaje svoj ASN u atribut AS-PATH. Na primer, pogledajmo sliku 4.40 i pretpostavimo da je prefiks 138.16.64/24 iz autonomnog sistema AS2 prvo objavljen autonomnom sistemu AS1; ako AS1 zatim objavi taj prefiks autonomnom sistemu AS3, atribut AS-PATH bi bio AS2 AS1. Ruteri koriste atribut AS-PATH za otkrivanje i sprečavanje višestrukih ponavljanja objava; drugim rečima, ako ruter vidi da se njegov AS već nalazi na spisku putanje, on odbacuje objavu. Kao što ćemo uskoro objasniti, ruteri koriste atribut AS-PATH i prilikom biranja jedne od više putanja za isti prefiks.
- Atribut NEXT-HOP ima malu, ali veoma važnu ulogu u ostvarivanju suštinski važne veze između protokola rutiranja unutar autonomnog sistema i protokola rutiranja među autonomnim sistemima. *Atribut NEXT-HOP je interfejs rutera kojim počinje putanja AS-PATH kroz autonomne sisteme*. Da bismo bolje upoznali ovaj atribut, pogledajmo ponovo sliku 4.40. Razmotrimo šta se događa kada ruter mrežnog prolaza 3a u AS3 objavi rutu ruteru mrežnog prolaza 1c u AS1, koristeći eBGP sesiju. Ruta obuhvata objavljeni prefiks, koji ćemo nazvati x i atribut AS-PATH tog prefiksa. Ova objava takođe obuhvata i atribut NEXT-HOP, koji je IP adresa interfejsa rutera 3a, koji vodi do rutera 1c. (Sećate se da ruter ima više IP adresa, po jednu za svaki interfejs.) Sada razmotrimo šta se događa kada ruter 1d od iBGP sesije sazna za tu rutu. Pošto sazna za tu rutu do prefiksa x, ruter 1d možda želi da prosleđuje pakete do x duž te rute, odnosno, ruter 1d može da u svoju tabelu prosleđivanja unese vrednost (x, I), gde je I njegov interfejs kojim počinje putanja najmanjih troškova od rutera 1d prema ruteru mrežnog prolaza 1c. Da bi odredio I, 1d obezbeđuje IP adresu u NEXT-HOP atributu svom modulu za rutiranje unutar autonomnog sistema. Obratite pažnju na to da je algoritam za rutiranje unutar autonomnog sistema već odredio putanje najmanjih troškova za sve podmreže, povezane sa ruterima u AS1, među njima i za podmrežu za link između rutera 1c i 3a. Iz putanje najmanjih troškova od 1d do podmreže 1c-3a, ruter 1d određuje svoj interfejs I kojim počinje ova putanja i zatim u svoju tabelu prosleđivanja dodaje unos (x, I). Uf! Ukratko, atribut NEXT-HOP koriste ruteri za ispravno konfigurisanje svojih tabela prosleđivanja.
- Na slici 4.41 prikazan je drugi slučaj u kome je neophodan atribut NEXT-HOP. Na ovoj slici, autonomni sistemi AS1 i AS2 su povezani preko dva ravnopravna linka. Ruter u AS1 može da sazna za dve različite rute do istog prefiksa x. Ove dve rute mogu da imaju isti atribut AS-PATH do x, ali mogu da imaju različite vrednosti atributa NEXT-HOP koje odgovaraju različitim ravnopravnim linkovima. Koristeći vrednosti atributa NEXT-HOP i algoritam rutiranja unutar autonomnog sistema, ruter može da odredi troškove putanje za oba ravnopravna linka, a da zatim primeni rutiranje vrućeg krompira (pogledajte odeljak 4.5.3), da bi odredio odgovarajući interfejs.

BGP takođe uključuje atribute koji ruterima omogućavaju da metrikama ruta dodele prioritet i atribut koji označava kako je prefiks umetnut u BGP na početnom autonomnom sistemu. Potpuni opis atributa ruta možete naći u [Griffin 2012; Stewart 1999; Halabi 2000; Feamster 2004; RFC 4271].



Slika 4.41 ◆ Atributi NEXT-HOP u objavama koriste se za određivanje koji ravnopravni link se koristi

Kada ruter mrežnog prolaza primi objavu od nekog rutera, on koristi svoja **pravila uvoza** za odlučivanje da li da odbaci ili prihvati odgovarajuću rutu i da li da postavi određene atribute kao što je prioritet metrike rutera. Pravila uvoza mogu da odbace neku rutu zato što AS ne želi da šalje saobraćaj preko nekog od autonomnih sistema iz atributa AS-PATH te rute. Ruter mrežnog prolaza može takođe da odbaci rutu, zato što već zna za bolju rutu prema istom prefiksu.

Biranje BGP ruta

Kao što je u ovom odeljku već rečeno, BGP koristi eBGP i iBGP sesije za distribuiranje ruta svim ruterima unutar autonomnih sistema. Iz ove distribucije ruter može da sazna da postoji više ruta prema istom prefiksu, u kom slučaju mora da izabere za

jednu od mogućih ruta. U ovom postupku izbora ruter polazi od svih ruta za koje je saznao i koje je prihvatio. Ako postoje dve rute ili više njih do istog prefiksa, BGP redom primenjuje sledeća pravila eliminacije, sve dok ne ostane samo jedna ruta.

- Svakoj ruti je dodeljen atribut vrednost lokalnog prioriteta. Lokalni prioritet rute može da postavi odgovarajući ruter ili može da se sazna od drugog rutera u istom autonomnom sistemu. Ovo je strateška odluka i prepušta se administratoru mreže autonomnog sistema. (Uskoro nešto podrobnije razmatramo strateška pitanja protokola BGP). Biraju se rute sa najvišim vrednostima lokalnog prioriteta.
- Od preostalih ruta (sve sa jednakim vrednostima lokalnog prioriteta) bira se ruta sa najkraćim atributom AS-PATH. Da je ovo jedino pravilo za biranje ruta, BGP bi za određivanje putanja koristio algoritam DV, gde se kao merilo razdaljine koristi broj AS skokova, a ne broj skokova rutera.
- Od preostalih ruta (sve sa jednakim vrednostima lokalnog prioriteta i jednakim dužinama atributa AS-PATH) bira se ruta sa najbližim NEXT-HOP ruterom. Ovde se najbližim smatra ruter za koga su troškovi putanje najmanjih troškova, određeni algoritmom rutiranja unutar autonomnog sistema, najniži. Kao što je rečeno u odeljku 4.5.3, ovaj postupak se naziva rutiranje vrućeg krompira.
- Ako i dalje postoji više ruta, ruteri koriste BGP identifikatore za biranje rute; pogledajte [Stewart 1999].

Pravila eliminacije mnogo su složenija, nego što je to gore opisano. Da bi se izbegle noćne more o protokolu BGP, bolje ga je učiti u manjim dozama!

PRINCIPI U PRAKSI

SASTAVLJANJE: KAKO SE UBACUJE STAVKA U TABELU PROSLEĐIVANJA RUTERA

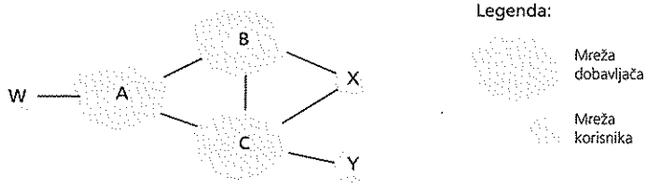
Setite se da se jedna stavka u tabeli prosleđivanja rutera sastoji iz prefiksa (npr. 138.16.64/22) i odgovarajućeg izlaznog porta rutera (npr. port 7). Kada paket stigne do rutera, odredišna IP adresa paketa se poredi sa prefiksima u tabeli prosleđivanja, kako bi se pronašao onaj prefiks sa najdužim podudaranjem. Paket se tada prosleđuje (unutar rutera) do porta rutera koji se povzuje sa tim prefiksom. Sada ćemo ukratko prikazati kako se stavka rutiranja (prefiks i odgovarajući port) unose u tabelu prosleđivanja. Ova jednostavna vežba će povezati ono što smo upravo naučili o rutiranju i prosleđivanju. Da bi stvari bile interesantnije, pretpostavićemo da je prefiks „strani prefiks", odnosno da ne pripada autonomnom sistemu rutera, već nekom drugom autonomnom sistemu.

Da bi prefiks uneli u tabelu prosleđivanja rutera, ruter mora *prvo da postane svestan* prefiksa (koji odgovara podmreži ili agregiranim podmrežama). Kao što smo upravo naučili, ruter postaje svestan prefiksa pomoću BGP objave putanje. Takva objava može da se pošalje preko eBGP sesije (sa rutera iz nekog drugog autonomnog sistema), ili preko iBGP sesije (sa rutera u istom autonomnom sistemu).

Kada ruter postane svestan prefiksa, trebalo bi da utvrdi odgovarajući izlazni port sa kog se datagrami, upućeni na taj prefiks, prosleđuju, pre unošenja tog prefiksa u tabelu prosleđivanja. Ako ruter primi više od jedne objave putanje za taj prefiks, ruter koristi BGP proces izbora putanje, na način ranije prikazan u ovom pododeljku, kako bi pronašao „najbolju" putanju za taj prefiks. Pretpostavimo da je izabrana takva najbolja putanja. Kao što je ranije opisano, izabrana putanja sadrži NEXT-HOP atribut, koji predstavlja IP adresu prvog rutera izvan AS rutera duž ove najbolje putanje. Gore smo objasnili, da ruter tada koristi svoj protokol rutiranja unutar AS (obično OSPF), da bi utvrdio najkraći put do NEXT-HOP rutera. Ruter na kraju utvrđuje broj porta koji bi doveo u vezu sa prefiksom identifikovanje prvog linka duž tog najkraćeg puta. Ruter tada može (konačno!) da unese par prefiksa i porta u svoju tabelu prosleđivanja! Tabela prosleđivanja koja se izračunava pomoću procesora rutiranja (videti sliku 4.6) se tada ubacuje u linijsku karticu ulaznog porta rutera.

Politika rutiranja

Neke od osnovnih pojmova politike BGP rutiranja prikazaćemo jednim jednostavnim primerom. Na slici 4.42 prikazano je šest međusobno povezanih autonomnih sistema: A, B, C, W, X i Y. Važno je da obratite pažnju na to da su: A, B, C, W, X i Y autonomni sistemi, a ne ruteri. Pretpostavimo da su autonomni sistemi W, X i Y završne mreže, a da su A, B i C mreže posrednika okosnica. Takođe, pretpostavljamo da su A, B i C međusobno ravnopravni autonomni sistemi i da svojim mrežama korisnicima obezbeđuju sve potrebne BGP informacije. Sav saobraćaj koji ulazi u **završnu mrežu** mora da ima odredište u toj mreži, a sav saobraćaj koji napušta završnu mrežu mora da potiče iz nje. Jasno je da su W i Y takve mreže. X je takozvana **višedomna završna mreža**, pošto je povezana sa ostatkom mreže preko dva različita posrednika (situacija koja je u praksi sve češća). Međutim, kao i W i Y, i mreža X mora da bude odredište, odnosno izvor svog saobraćaja koji stiže u mrežu X ili izlazi iz nje. Ali, kako se ostvaruje i postiže 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. U ovom slučaju, X će funkcionisati kao završna mreža, ako (svojim susedima B i C) objavi da nema putanje ni za jedno odredište osim svojih. To jest, iako X zna za putanju, na primer XCY, kojom se stiže do mreže Y, ona *neće* objaviti tu putanju mreži 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 prikazuje kako se politika selektivnog objavljivanja ruta može iskoristiti za ostvarivanje odnosa korisnika i posrednika prilikom rutiranja.



Slika 4.42 ◈ Jednostavan primer BGP rutiranja

ZAŠTO POSTOJE RAZLIČITI PROTOKOLI ZA RUTIRANJE UNUTAR AUTONOMNIH SISTEMA I RUTIRANJE MEĐU AUTONOMNIM SISTEMIMA?

Pošto smo podrobno proučili pojedine protokole za rutiranje unutar autonomnih sistema i među autonomnim sistemima koji se koriste na današnjem internetu, razmatranje možemo zaključiti možda najfundamentalnijim pitanjem koje se uopšte može postaviti o tim protokolima (dobro je da se stalno to pitate i da niste zbog drveća iz vida izgubili šumu!): zašto se koriste različiti protokoli rutiranja unutar autonomnih sistema i među autonomnim sistemima?

Odgovor na ovo pitanje ukazuje na osnovnu razliku između ciljeva rutiranja unutar autonomnih sistema i rutiranja među autonomnim sistemima.

- *Pravila.* U odnosu između autonomnih sistema vrlo su bitna određena pravila. Ponekad je važno da se nipošto ne dozvoli prolaženje saobraćaja koji potiče iz nekog autonomnog sistema kroz drugi određeni autonomni sistem. Slično, neki AS možda sa pravom želi da kontroliše saobraćaj koji prenosi između drugih autonomnih sistema. Videli smo da BGP prenosi atribute putanje i omogućava kontrolisanu distribuciju informacija o rutiranju, tako da se mogu donositi odluke rutiranja, zasnovane na uspostavljenim međusobnim odnosima. Unutar istog autonomnog sistema sve je bar uslovno pod istom administrativnom kontrolom, pa ova pravila ponašanja imaju daleko manji značaj za biranje rute unutar određenog autonomnog sistema.
- *Proširivost.* Mogućnost da se algoritam rutiranja i strukture njegovih podataka prilagode tako da mogu da se izbore sa rutiranjem prema većem broju mreža ili sa rutiranjem između velikog broja mreža, izuzetno je značajna za rutiranje među autonomnim sistemima. Unutar nekog autonomnog sistema prilagodljivost je manje bitna. U svakom slučaju, ukoliko neki autonomni sistem kojim se upravlja sa jednog mesta previše naraste, uvek je moguće podeliti ga na dva autonomna sistema i sprovoditi rutiranje između ta dva nova autonomna sistema. (Sećate se da OSPF dozvoljava uspostavljanje takvih odnosa podelom autonomnog sistema u manje oblasti).
- *Performanse.* Pošto rutiranje među autonomnim sistemima prvenstveno zavisi od odnosa uspostavljenih između njih, kvalitet (odnosno, performanse) ruta koje se koriste, često je drugorazredna briga (tj. bira se duža i skuplja ruta koja zadovoljava određene kriterijume međusobno uspostavljenih odnosa, umesto kraće rute koja te kriterijume ne zadovoljava). Zaista, videli smo da između autonomnih sistema nema čak ni nagoveštaja o troškovima određenih ruta (osim broja AS skokova). Unutar pojedinačnih autonomnih sistema, međutim, o tim odnosima se manje vodi računa, što omogućava da se rutiranje obavlja tako da se ostvaruju bolje performanse.

Usredsredimo se sada na mrežu posrednika, recimo autonomni sistem B. Pretpostavimo da je B saznao (od A) da A ima putanju AW prema W. To znači da B može da ugradi rutu BAW u svoju bazu informacija za rutiranje. Očigledno, B takođe želi da objavi putanju BAW svom korisniku X, tako da X zna da prema W može da rutira preko B. Međutim, da li bi B trebalo da objavi putanju BAW i posredniku C? Ako to uradi, onda bi C mogao da rutira saobraćaj prema W putanjom CBAW.

Pošto su A, B i C posrednici okosnice, onda B opravdano može da smatra da on ne bi trebalo da deli teret (i troškove!) prenošenja tranzitnog saobraćaja između posrednika A i C. B može opravdano da smatra da je obaveza (i troškovi!) posrednika A i C da obezbede da C rutira prema korisnicima posrednika A ili od njih, koristeći neposrednu vezu između A i C. Trenutno među posrednicima, okosnicama za internet usluge, ne postoje zvanični standardi za međusobno rutiranje. Međutim, osnovno pravilo kojeg se drže komercijalni posrednici za internet usluge jeste da saobraćaj koji teče njihovom mrežom mora da ima izvor ili odredište (ili oboje) u mreži koja je korisnik tog posrednika za internet usluge; inače bi taj saobraćaj besplatno koristio mrežu tog posrednika. Pojedinačni ugovori o saradnji (kojima se rešavaju problemi kao što su ovi o kojima pričamo) obično nastaju pregovorima između samih posrednika i često su poverljivi; [Huston 1999] nudi zanimljivo razmatranje ovakvih ugovora o saradnji. Za detaljan opis uticaja pravila rutiranja na komercijalni odnos između posrednika za internet usluge pogledajte [Gao 2001; Dmitiropoulos 2007]. Za najnovija razmatranja pravila BGP rutiranja sa stanovišta posrednika za internet usluge pogledajte [Caesar 2005].

Kao što smo već rekli, BGP je *de facto* standard za rutiranje među autonomnim sistemima za javni internet. Ako Vas zanima sadržaj različitih BGP tabela rutiranja (velike su!) iz rutera posrednika prvog reda, posetite adresu 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 [Potaroo 2012].

Ovim zaključujemo kratak uvod u protokol BGP. Važno je razumeti BGP zbog njegovog značaja na internetu. Savetujemo čitaocima da pregledaju reference [Griffin 2012; Stewart 1999; Labovitz 1997; Halabi 2000; Huitema 1998; Gao 2001; Feamster 2004; Caesar 2005b; Li 2007], da bi saznali više o protokolu BGP.

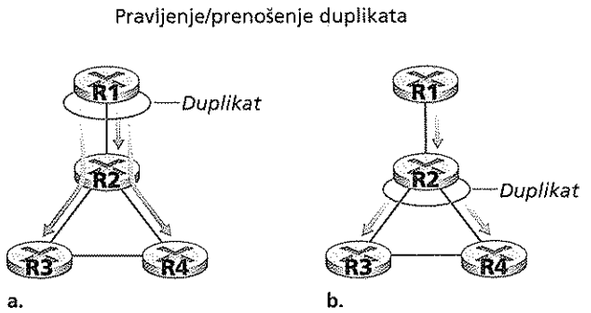
4.7 Difuzno i višeznačno rutiranje

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 pažnju usmeravamo na protokole za difuzno i višeznačno rutiranje. Kod **difuznog rutiranja** mrežni sloj pruža uslugu isporuke paketâ koji se iz izvornog čvora šalje svim ostalim čvorovima u mreži; **višeznačno rutiranje** omogućava da jedan izvorni čvor pošalje po primerak paketa određenom broju ostalih čvorova u mreži. U odeljku 4.7.1 razmatramo algoritme difuznog rutiranja i njihovo otelotvorenje u protokolima rutiranja. Višeznačno rutiranje istražujemo u odeljku 4.7.2.

4.7.1 Algoritmi difuznog rutiranja

Možda najprostiji način za ostvarivanje komunikacije sa difuznim slanjem jeste da otpremni čvor pošalje zaseban primerak paketa na svako odredište, kao što je prikazano na slici 4.43(a). Za N odredišnih čvorova, izvorni čvor jednostavno pravi N kopija paketa, adresira svaki primerak na drugo odredište i zatim šalje N primeraka na N odredišta, koristeći jednoznačno rutiranje. Ovakvo rešenje za difuzno slanje sa **N-tostrukim jednoznačnim** slanjem je jednostavno – nije potreban novi protokol rutiranja mrežnog sloja, nije potrebno dupliranje paketa, niti nova funkcionalnst prosleđivanja. Ovo rešenje, međutim, ima nekoliko nedostataka. Prvi nedostatak je njegova neefikasnost. Ako je izvorni čvor vezan za ostatak mreže samo jednim linkom, kroz taj link će proći N primeraka (istog) paketa. Jasno je da bi bilo efikasnije da se u prvom skoku pošalje samo jedan primerak, a da zatim čvor na drugoj strani prvog skoka napravi i prosledi potreban broj dodatnih kopija. Drugim rečima, bilo bi efikasnije kada bi sami mrežni čvorovi (a ne samo izvorni čvor) pravili kopije paketa. Na primer, na slici 4.43(b) kroz link R1-R2 prolazi samo jedan primerak paketa. Taj paket se zatim umnožava u R2, a zatim se linkovima R2-R3 i R2-R4 šalje po jedan primerak.

Još jedan nedostatak pristupa sa N-tostrukim jednoznačnim slanjem možda je teže uočljiv, ali nije manje značajan. Prećutno se podrazumeva da pri N-tostrukom jednoznačnom slanju pošiljalac zna sve difuzne primaoce i njihove adrese. Ali, kako da pribavi te informacije? Najverovatnije bi bili potrebni dodatni mehanizmi protokola (kao što su učlanjavanje u grupe za difuzno slanje ili protokol za registrovanje odredišta). Time bi se protokol opteretio dodatnim zadacima, a što je još važnije, protokol koji je na početku izgledao sasvim jednostavan postao bi neuporedivo složeniji. Konačni nedostatak rešenja sa N-tostrukim jednoznačnim slanjem odnosi se na ono čemu je difuzno slanje namenjeno. U odeljku 4.5 naučili smo da protokoli rutiranja prema stanju linkova koriste difuzno slanje za objavljivanje informacija o stanju linkova, koje se koriste za izračunavanje jednoznačnih ruta. Jasno je da, u okolnostima u kojima se difuzno slanje koristi za izradu i ažuriranje jednoznačnih ruta, ne bi bilo mudro (u najmanju ruku!) osloniti se na infrastrukturu za jednoznačno rutiranje, da bi se postiglo difuzno slanje.



Slika 4.43 ◆ Umnožavanje u izvoru u odnosu na umnožavanje u mreži

S obzirom na nekoliko navedenih nedostataka sa difuznim N-tostrukim jednoznačnim slanjem, jasno je zanimanje za rešenja u kojima mrežni čvorovi i sami imaju aktivnu ulogu u umnožavanju paketa, prosleđivanju paketa i izračunavanju ruta za difuzno slanje. Ispitaćemo nekoliko takvih rešenja i pri tome ćemo ponovo koristiti grafove koje smo uveli u odeljku 4.5. Ponovo modelujemo mrežu kao graf, $G = (N,E)$, gde je N skup čvorova, a E je skup ivica, pri čemu svaka ivica povezuje par čvorova iz skupa N. Bićemo malo nedosledni prilikom obeležavanja, pa ćemo koristiti N za označavanje skupa čvorova i za označavanje broja članova ($|N|$) ili veličinu skupa.

Nekontrolisano plavljenje

Najočigledniji način da se ostvari difuzno slanje je rešenje sa **plavljenjem** (eng. flooding) u kojem izvorni čvor šalje kopiju paketa svim svojim susedima. Kada čvor primi difuzno poslat paket, on ga umnožava i prosleđuje svim svojim susedima (osim suseda od kojeg je primio paket). Očigledno je da će na ovaj način, ako je graf povezan, svaki čvor u grafu pre ili kasnije primiti kopiju difuzno poslatog paketa. Iako je ova šema jednostavna i elegantna, ona ima jednu fatalnu grešku (pre nego što nastavite sa čitanjem, pokušajte sami da otkrijete tu fatalnu grešku). Ako je više čvorova u grafu povezano u krug, jedna ili više kopija difuzno poslatih paketa će beskonačno kružiti. Na primer, na slici 4.43, R2 šalje prema R3, R3 šalje prema R4, R4 šalje prema R2, a R2 šalje (ponovo!) prema R3 i tako dalje. U ovom jednostavnom primeru dolazi do beskonačnog kruženja dva difuzno poslata paketa, jednog u smeru kretanja kazaljke na satu, a drugog u smeru suprotnom smeru kretanja kazaljke na satu. Međutim, može se javiti jedna još razornija fatalna greška: kada je čvor povezan sa više od dva čvora, on će napraviti i proslediti više primeraka difuzno poslatog paketa, pri čemu će se od njih napraviti više primeraka (u drugim čvorovima sa više od dva suseda) i tako dalje. Takva **difuzna oluja**, koja potiče od beskrajnog umnožavanja difuzno poslatih paketa na kraju bi proizvela toliko difuzno poslatih paketa da bi mreža postala neupotrebljiva. (Među domaćim zadacima na kraju ovog poglavlja se nalazi problem u kome se analizira brzina kojom nastaje takva oluja difuznog slanja).

Kontrolisano plavljenje

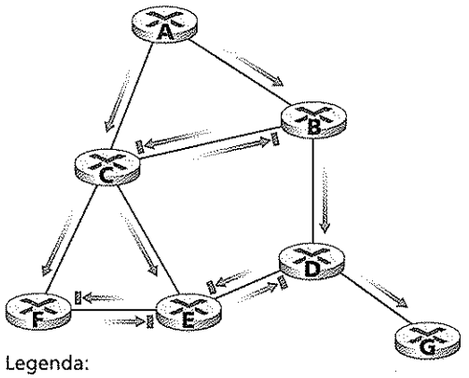
Rešenje za izbegavanje difuzne oluje je u tome da čvor pravilno proceni kada je potrebno da prosledi pakete svim svojim susedima, to jest da izvrši plavljenje, a kada ne (npr. ukoliko je već primio i prosledio raniji primerak istog paketa). U praksi se to obavlja na jedan od nekoliko načina.

U **plavljenju koje se kontroliše rednim brojevima** izvorni čvor svoju adresu (ili neki drugi jedinstveni znak raspoznavanja) kao i **difuzni redni broj** stavlja u paket za difuzno slanje, a zatim taj paket šalje svojim susedima. Svaki čvor održava spisak izvornih adresa i rednih brojeva za sve difuzno poslate pakete koje je već primio, umnožio i prosledio. Kada čvor primi difuzno poslati paket, prvo proverava da li se taj paket nalazi na tom spisku. Ako je tako, paket se odbacuje; ako ga nema, paket se umnožava i prosleđuje susedinim čvorovima (osim čvoru od kojeg je paket primljen). U protokolu *Gnutella*, opisanom u poglavlju 2, koristi se plavljenje koje se kontroliše rednim broje-

vima za difuzno slanje upita u preklopljenu mrežu. (U protokolu *Gnutella* umnožavanje i prosleđivanje poruka se obavlja na aplikativnom, a ne namrežnom sloju).

Drugo rešenje za kontrolu plavljenja poznato je kao **prosleđivanje na osnovu obrnute putanje** (reverse path forwarding, RPF) [Dalal 1978], poznato i pod nazivom difuzno slanje na osnovu obrnute putanje (reverse path broadcasting, RPB). Zamisao na kojoj se zasniva RPF je istovremeno jednostavna i elegantna. Kada ruter primi difuzno poslati paket sa datom adresom izvora, on taj paket prenosi na sve svoje izlazne linkove (osim onog po kojem je paket primljen), samo ako je paket stigao linkom koji predstavlja njegovu najkraću jednoznačnu povratnu putanju prema izvoru. Inače, ruter jednostavno odbacuje pristigli paket i ne prosleđuje ga ni na jedan od svojih izlaznih linkova. Takav paket može da se odbaci, jer ruter zna da je linkom koji se nalazi na njegovoj najkraćoj putanji prema pošiljaocu već primio ili će primiti primerak istog paketa. (Pokušajte da proverite da li se to zaista i događa i da li se na ovaj način ne dolazi do kruženja niti do difizne oluje). Obratite pažnju na to da RPF ne koristi jednoznačno rutiranje za isporuku paketa na odredište, a i ne zahteva se da ruter zna celu najkraću putanju između njega i izvora. Jedino je potrebno da zna ko je sledeći sused na njegovoj najkraćoj jednoznačnoj putanji prema pošiljaocu, zato što identitet tog suseda koristi samo da bi odlučio da li da vrši ili ne vrši plavljenje difuzno primljenog paketa.

Na slici 4.44 je prikazano prosleđivanje na osnovu obrnute putanje. Pretpostavimo da linkovi, prikazani debljim linijama, predstavljaju putanje najmanjih troškova od primalaca do izvora (*A*). Čvor *A* na početku difuzno šalje paket izvora *A* čvorovima *C* i *B*. Čvor *B* prosleđuje paket izvora *A* koji je primio od *A* (pošto je *A* na njegovoj putanji najmanjih troškova prema *A*) čvorovima *C* i *D*. Čvor *B* zanemaruje (odbacuje, bez prosleđivanja) pakete izvora *A*, koje primi od ostalih čvorova (na primer, od rutera *C* ili *D*). Razmotrimo sada čvor *C* koji prima paket izvora *A* neposredno od *A*, ali i od *B*. Pošto B nije na njegovoj najkraćoj putanji do *A*, čvor *C* ne uzima u obzir pakete izvora *A* koje prima od *B*. S druge strane, paket izvora *A*, koje *C* primi neposredno od *A*, prosleđuje čvorovima *B*, *E* i *F*.

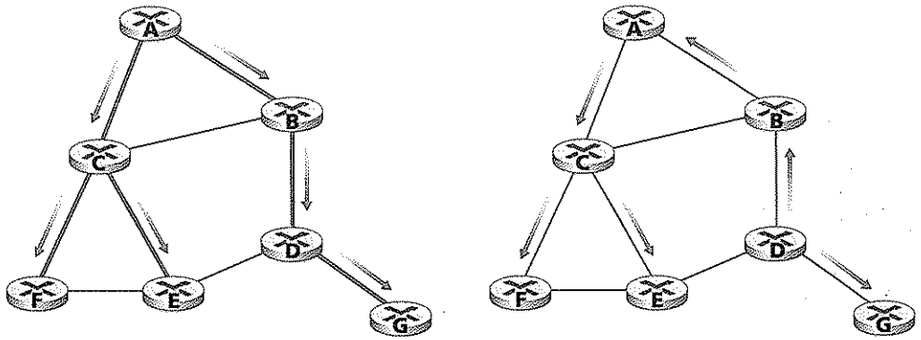


Slika 4.44 ◆ Prosleđivanje na osnovu obrnute putanje

Difuzno slanje sveobuhvatnim stablom

Mada se plavljenjem koje se kontroliše rednim brojevima i prosleđivanjem na osnovu obrnute putanje izbegavaju difuzne oluje, njima se ne izbegava prenošenje suvišnih difuzno poslatih paketa u potpunosti. Na primer, na slici 4.44 čvorovi: *B*, *C*, *D*, *E* i *F* primaju jedan ili dva suvišna paketa. Idealno bi bilo da svaki čvor primi samo jedan primerak difuzno poslatog paketa. Proučavanjem stabla koje se sastoji od čvorova povezanih debljim linijama, na slici 4.45(a), jasno je da, kada bi se difuzno poslati paketi prosleđivali samo duž linkova unutar ovog stabla, svaki čvor u mreži primio bi samo jedan primerak difuzno poslatog paketa – a to je rešenje kakvo i tražimo! Ovo stablo je primer **sveobuhvatnog stabla** (eng. spanning tree) – stabla koje sadrži sve čvorove u grafu. Formalnije rečeno, sveobuhvatno stablo grafa $G = (N,E)$ je graf $G' = (N,E')$ takav da je E' podskup od E, a G' je povezan graf koji ne obuhvata kružne tokove i sadrži sve prvobitne čvorove iz G. Ako se svim linkovima pridruže troškovi, a trošak stabla je zbir troškova linkova, tada se sveobuhvatno stablo čiji su troškovi najmanji od svih sveobuhvatnih stabala tog grafa naziva (sasvim očekivano) **minimalno sveobuhvatno stablo**.

Na osnovu ovoga nameće se još jedno rešenje za difuzno slanje u kome se od čvorova mreže prvo pravi sveobuhvatno stablo. Kada izvorni čvor želi da difuzno pošalje neki paket, šalje ga samo preko linkova koji pripadaju sveobuhvatnom stablu. Čvor koji primi difuzno poslati paket zatim prosleđuje taj paket svim svojim susedima u sveobuhvatnom stablu (osim susedu od kojeg je primio paket). U sveobuhvatnom stablu ne samo da nema suvišnih difuzno poslatih paketa, nego i, nakon što se uspostavi, svi čvorovi koji otpočinju difuzno slanje mogu da ga koriste, kao što je prikazano na slikama 4.45(a) i 4.45(b). Obratite pažnju na to da čvor ne mora da zna čitavo stablo; sve što je potrebno da zna je koji su njegovi susedi u *G* istovremeno i susedi u sveobuhvatnom stablu.

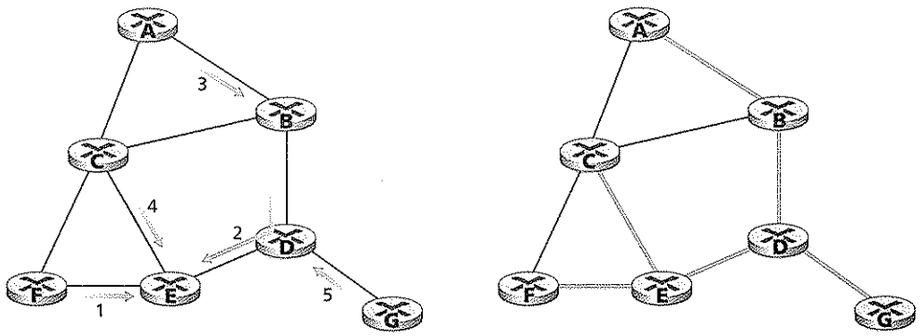


a. Difuzno emitovanje iz izvora A **b. Difuzno emitovanje iz izvora D**

Slika 4.45 ◆ Difuzno slanje duž sveobuhvatnog stabla

Najviše kompleksnosti u rešenju sa sveobuhvatnim stablom stvara pravljenje i održavanje sveobuhvatnog stabla. Razvijeno je više distribuiranih algoritama sveobuhvatnog stabla [Gallager 1983, Gartner 2003]. Ovde razmatramo samo jedan jednostavan algoritam. U **rešenju sa centralnim čvorom** za izgradnju sveobuhvatnog stabla najpre se određuje centralni čvor (ponekad se naziva **tačka susreta** ili **jezgro**). Čvorovi zatim jednoznačno šalju poruke o pristupanju stablu, koje su upućene tom centralnom čvoru. Poruka o pristupanju stablu prosleđuje se korišćenjem jednoznačnog rutiranja prema centru, sve dok ne stigne, ili do čvora koji već pripada sveobuhvatnom stablu, ili do centra. U svakom slučaju, putanja kojom je išla poruka o pristupanju stablu određuje granu sveobuhvatnog stabla između rubnog čvora koji je pokrenuo poruku o pristupanju stablu i centra. Može se zamisliti kao da se ta nova putanja kalemi na postojeće stablo.

Na slici 4.46 prikazana je izgradnja sveobuhvatnog stabla sa centralnim čvorom. Pretpostavimo da je čvor *E* izabran za centar stabla. Pretpostavimo da čvor *F* prvi pristupa stablu, tako što poruku o pristupanju prosleđuje čvoru *E*. Link *EF* postaje začetak sveobuhvatnog stabla. Stablu zatim pristupa čvor *B*, tako što svoju poruku o pristupanju šalje čvoru *E*. Pretpostavimo da se ova poruka jednoznačno rutira iz *B* u *E* preko *D*. U ovom slučaju, na osnovu ove poruke o pristupanju na stablo se kalemi putanja *BDE*. Čvor *A* prosleđivanjem svoje poruke o pristupanju prema *E*, sledeći pristupa grupi. Ako putanja za jednoznačno rutiranje iz *A* prema *E* prolazi kroz *B* onda, pošto je čvor *B* već pristupio sveobuhvatnom stablu, dolazak poruke o pristupanju čvora *A* u *B* znači da se link *AB* odmah kalemi na sveobuhvatno stablo. Sledeći sveobuhvatnom stablu pristupa čvor *C* – prosleđivanjem poruke o pristupanju neposredno u *E*. Na kraju, pošto jednoznačno rutiranje iz *G* u *E* mora da prolazi preko čvora *D*, kada *G* pošalje svoju poruku o pristupanju u *E*, link *GD* se kalemi na sveobuhvatno stablo u čvoru D.



a. Redosled izgradnje sveobuhvatnog stabla

b. Izgrađeno sveobuhvatno stablo

Slika 4.46 ◇ Izgradnja sveobuhvatnog stabla sa centralnim čvorom

Algoritmi difuznog rutiranja u praksi

Protokoli za difuzno rutiranje u praksi se koriste na aplikativnom i na mrežnom sloju. Protokol *Gnutella* [Gnutella 2009] koristi difuzno slanje na nivou aplikativnog sloja, kako bi se svim primaocima poslali upiti o sadržaju koji dele *Gnutella* ravnopravni učesnici. Ovde je link između dva distribuirana ravnopravna procesa na aplikativnom nivou u *Gnutella* mreži u suštini TCP veza. *Gnutella* koristi poseban vid plavljenja, kontrolisan rednim brojevima u kojem se 16-bitni identifikator i 16-bitni indeks korisnih podataka (kojim se određuje tip *Gnutella* poruke) koriste za utvrđivanje da li je upravo primljeni difuzno poslati upit ranije već bio primljen, umnožen i prosleđen. *Gnutella* takođe koristi polje TTL (rok trajanja) kojim se ograničava broj skokova preko kojih se plavljeni upit prosleđuje. Kada primi i umnoži upit, Gnutella proces pre prosleđivanja upita smanjuje vrednost polja TTL. Na taj način, plavljeni upit protokola *Gnutella* stiže samo do onih ravnopravnih učesnika koji su od pokretača upita udaljeni manje od datog broja (početne vrednosti TTL) skokova na aplikativnom nivou. Mehanizam plavljenja protokola *Gnutella* se zato ponekad naziva *plavljenje sa ograničenim dometom*.

Još jedan vid plavljenja kontrolisan rednim brojevima koristi se za difuzno slanje objava o stanju linkova (Link State Advertisment, LSA) u algoritmu rutiranja OSPF [RFC 2328, Perlman 1999] i u algoritmu rutiranja IS-IS (Intermediate-System-to-Intermediate-System [RFC 1142, Perlman 1999]. Protokol OSPF za identifikovanje LSA objava koristi 32-bitni redni broj kao i 16-bitno polje za rok važnosti. Sećate se da OSPF čvor difuzno šalje LSA objave za linkove povezane sa njim povremeno, kada se promene troškovi nekog linka do suseda, ili kada se link uključi/isključi. Redni brojevi LSA objava koriste se za otkrivanje duplikata, ali služe i za obavljanje još jednog važnog zadatka u protokolu OSPF. Prilikom plavljenja je moguće da LSA objava koja je napravljena na izvoru u trenutku t stigne *posle* novije LSA objave koja je na istom izvoru napravljena u trenutku $t + \delta$. Redni brojevi koje pravi izvorni čvor omogućavaju da se starija LSA objava razlikuje od novije LSA objave. Polje za rok važnosti služi u slične svrhe čemu služi vrednost polja TTL. Početna vrednost polja za rok važnosti postavljena je na nulu, a tokom plavljenja povećava se pri svakom skoku, ali se povećava i dok u memoriji rutera čeka na to da se prosledi dalje. Čak i iz ovog kratkog opisa LSA algoritma plavljenja primećujemo da je projektovanje LSA protokola za difuzno slanje zaista složen posao. U [RFC 789; Perlman 1999] opisan je nemio slučaj kada je zbog neispravno prenete LSA objave između dva rutera u kvaru došlo do toga da jedna od prvih verzija LSA algoritma plavljenja prekine rad čitave ARPAnet mreže.

4.7.2 Višeznačno rutiranje

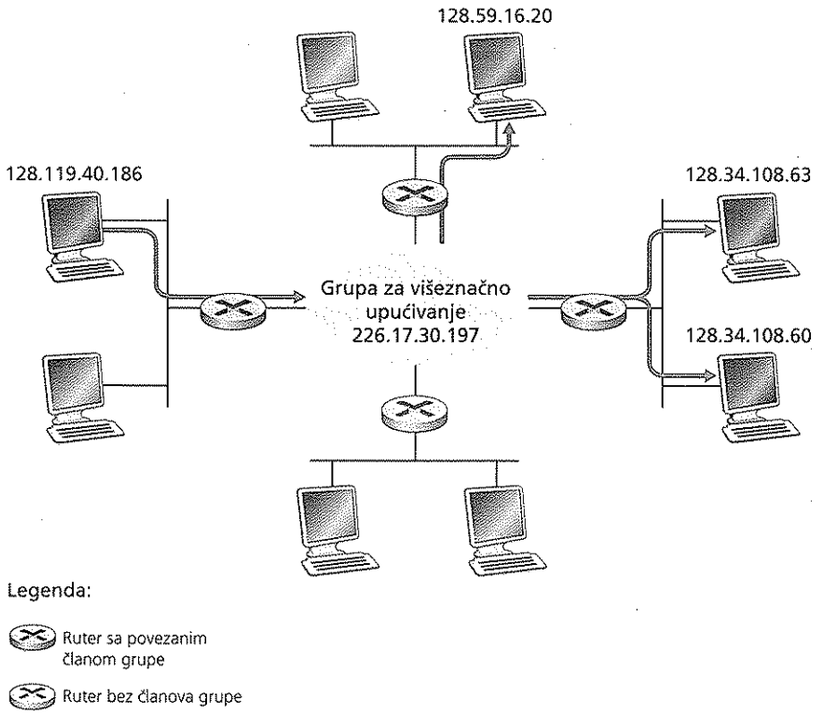
U prethodnom odeljku videli smo da se uslugom difuznog rutiranja paketi isporučuju svim postojećim čvorovima u mreži. U ovom odeljku svoju pažnju usmeravamo na uslugu **višeznačnog** rutiranja u kojoj se višeznačno poslati paket isporučuje samo *podskupu* mrežnih čvorova. Sve veći broj novih mrežnih aplikacija zahteva isporu-

ku paketa od jednog ili više pošiljalaca do grupe primalaca. Te aplikacije obuhvataju slanje obimnih podataka (na primer, prenos nadogradnje softvera od programera do više korisnika kojima je potrebna ta nadogradnja), protok kontinulanih multimedijalnih sadržaja (na primer, prenos audio, video i tekstualnih zapisa sa predavanja uživo do grupe raštrkanih učesnika predavanja), aplikacije sa deljenim podacima (na primer, elektronska tabla za telekonferencije koja je zajednička za više raštrkanih učesnika), aplikacije za slanje novosti (na primer, berzanski izveštaji), ažuriranje veb keša i interaktivne igre (na primer, distribuirano interaktivno virtuelno okruženje ili igre sa više igrača).

U višeznačnoj komunikaciji se na samom početku suočavamo sa dva problema – kako prepoznati primaoce višeznačno poslatog paketa i kako se adresira paket koji se šalje tim primaocima. U slučaju jednoznačne komunikacije, IP adresa primaoca (odredišta) nalazi se u svakom jednoznačno poslatom IP datagramu i određuje jedinog primaoca; u slučaju difuznog slanja, *svi* čvorovi bi trebalo da prime difuzno poslati paket, pa adrese odredišta nisu ni potrebne. Međutim, u slučaju višeznačnog slanja imamo više primalaca. Ima li smisla da svaki višestruko poslati 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 pojedinačno da naznači sve primaoce, morao bi da zna njihove identitete i adrese. Ubrzo ćemo videti da postoje slučajevi kada tako nešto nije poželjno.

Iz tih razloga, u arhitekturi interneta (i u drugim mrežnim arhitekturama, kao što je ATM mreža [Black 1995]), višeznačno poslati paket adresira se korišćenjem **posrednog adresiranja** (eng. address indirection). Drugim rečima, za grupu primalaca koristi se samo jedan identifikator, a primerak paketa koji je adresiran na određenu grupu pomoću tog identifikatora isporučuje se svim primaocima koji su pridruženi toj grupi. Identifikator koji na internetu predstavlja grupu primalaca je višeznačna IP adresa klase D. Grupa primalaca pridružena adresi klase D naziva se **višeznačna grupa** (eng. multicast group). Pojam višeznačne grupe prikazan je na slici 4.47. Na ovoj slici četiri računara (prikazana osenčeno), kojima je pridružena adresa višeznačne grupe 226.17.30.197, primaju sve datagrame upućene na tu višeznačnu adresu. Poteškoća koja preostaje da se reši jeste činjenica da svi računari imaju jedinstvenu jednoznačnu IP adresu, koja je potpuno nezavisna od adrese višeznačne grupe kojoj pripadaju.

Pojam višeznačne grupe je jednostavan, mada otvara sijaset 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 iz grupe), ili je članstvo u grupi ograničeno, i ako je tako, ko ga ograničava? Da li članovi grupe mogu da saznaju ko su ostali članovi grupe u okviru protokola mrežnog sloja? Kako čvorovi mreže međusobno sarađuju prilikom isporučivanja višeznačno poslatog datagrama svim članovima grupe? Za internet, odgovori na sva ova pitanja nalaze se u protokolu za upravljanje grupama na internetu (engl. *Internet Group Management Protocol*, IGMP) [RFC 3376]. Zato, prvo ukratko razmotramo IGMP, a zatim se vraćamo ovim opštijim pitanjima.



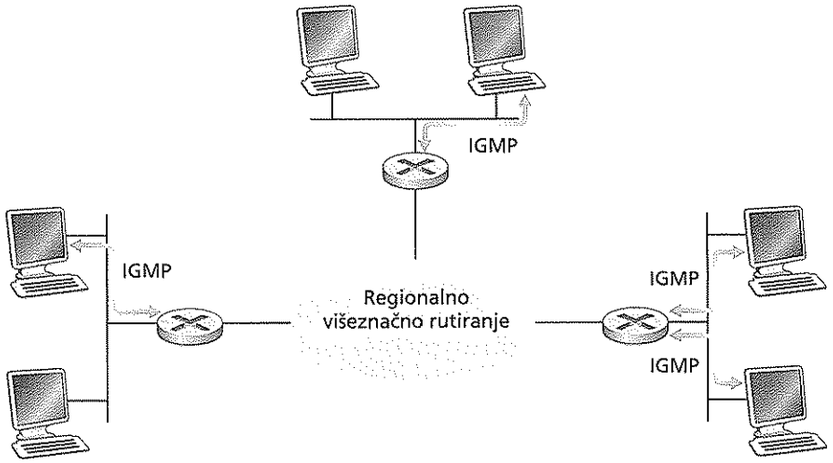
Slika 4.47 ◆ Višeznačna grupa: datagram adresiran na ovu grupu isporučuje se svim članovima višeznačne grupe

Protokol IGMP

Protokol IGMP (Internet Group Management Protocol) verzija 3 [RFC 3376] izvršava se između računara i sa njim neposredno povezanog rutera (neposredno povezani ruter može se, mada nije baš uvek tako, zamisliti kao ruter prvog skoka na koji bi računar prvo naišao na putanji prema bilo kom drugom računaru izvan svoje lokalne mreže, ili kao ruter poslednjeg skoka na bilo kojoj putanji prema ovom računaru), kao što je prikazano na slici 4.48. Na slici 4.48 prikazana su tri višeznačna rutera prvog skoka, pri čemu je svaki povezan sa svojim računarima preko izlaznog lokalnog interfejsa. Taj lokalni interfejs je u ovom primeru povezan sa LAN mrežom i iako svaki LAN ima više povezanih računara, tek nekoliko ovih računara u isto vreme pripada istoj višeznačnoj grupi.

IGMP nudi način na koji računar obaveštava svoj povezani ruter da aplikacija koja se izvršava na tom računaru želi da pristupi određenoj višeznačnoj grupi. Kako je delokrug protokola IGMP ograničen na računar i njegov neposredno povezani ruter, potreban je drugi protokol za usklađivanje višeznačnih rutera (uključujući i povezane rutere) na čitavom internetu, tako da se višeznačno poslati datagrami rutiraju do svojih konačnih odredišta. Ovaj poslednji zadatak postiže se algoritmima za više-

značno rutiranje na mrežnom sloju, o kojima ćemo uskoro saznati nešto više. Stoga se višeznačno rutiranje na mrežnom sloju interneta sastoji se od dve komponente koje se međusobno dopunjuju: protokola IGMP i protokola za višeznačno rutiranje.



Slika 4.48 ◈ Dve komponente višeznačnog rutiranja na mrežnom sloju interneta: protokol IGMP i protokol za višeznačno rutiranje

Protokol IGMP ima samo tri vrste poruka. Slično porukama protokola ICMP i poruke protokola IGMP prenose se (enkapsuliraju) unutar IP datagrama sa IP brojem protokola 2. Ruter šalje poruku `membership_query` svim računarima na priključnom interfejsu (na primer, svim računarima lokalne mreže), da bi utvrdio skup svih višeznačnih grupa kojima su pristupili računari sa tog interfejsa. Računari na poruku `membership_query` odgovaraju porukom protokola IGMP `membership_report`. Računar može da napravi poruku `membership_report`, kada neka aplikacija prvi put pristupa višeznačnoj grupi, ne čekajući poruku `membership_query` od rutera. Poslednja vrsta poruke protokola IGMP je poruka `leave_group`. Zanimljivo je da ova poruka nije obavezna. Ali, ako nije obavezna, kako ruter da otkrije kada neki računar napusti višeznačnu grupu? Odgovor na ovo pitanje je da ruter *zaključuje* da određeni računar nije više u višeznačnoj grupi ukoliko više ne odgovara na poruku `membership_query` sa datom adresom grupe. Ovo je primer onoga što se ponekad naziva **meko stanje** (eng. soft state) u internet protokolu. U protokolu sa mekim stanjima, neko stanje (u ovom slučaju sa protokolom IGMP, činjenica da postoje računari koji pripadaju višeznačnoj grupi) prekida se događajem isteka određenog vremena (u ovom slučaju, porukom `membership_query`, koju ruter periodično šalje), ukoliko se izričito ne obnovi (u ovom slučaju, porukom `membership_report`, koju šalje povezani računar).

Izraz meko stanje izmislio je Klark [Clark 1988], koji je opisao periodične poruke za osvežavanje stanja koje šalje krajnji sistem, i utvrdio da sa takvim porukama za osvežavanje, stanje može da se izgubi prilikom pada sistema i da se automatski

obnovi sledećom porukom za osvežavanje – sve je transparentno u krajnjem sistemu i nema pozivanja bilo kakve procedure za oporavak sistema:

> *„...informacije o stanju neće biti kritične u održavanju željene vrste usluge koja je povezana sa tokom. Umesto toga, vrsta usluge će biti podstaknuta krajnjim tačkama, koje će periodično slati poruke, kako bi se obezbedila odgovarajuća vrsta usluge koja se povezuje sa tokom. Na ovaj način, informacije o stanju, povezane sa tokom, mogu da se izgube prilikom pada sistema bez trajnog prekida usluge koja se koristi. Ovaj koncept zovem „meko stanje" i ono nam veoma dobro može dozvoliti da postignemo naše primarne ciljeve održavanja i fleksibilnosti..."*

Veruje se da je protokole sa mekim stanjima jednostavnije kontrolisati od protokola sa čvrstim stanjima koji, ne samo da zahtevaju da se ta stanja izričito dodaju i uklanjaju, već takođe zahtevaju i mehanizme za oporavak u slučajevima kada se entitet zadužen za uklanjanje stanja isključi pre vremena ili otkaže sa radom. Zanimljiva rasprava o mekom stanju može da se pronađe u [Raman 1999; Ji 2003; Lui 2004].

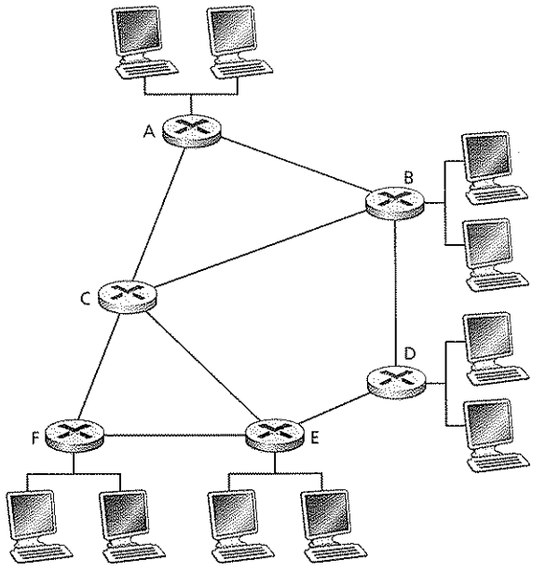
Algoritmi za višeznačno rutiranje

Na slici 4.49 prikazan je **problem višeznačnog rutiranja**. Računari pridruženi višeznačnoj grupi su osenčeni; njihovi neposredno povezani ruteri takođe su prikazani osenčeno. Kao što je prikazano na slici 4.49, samo podskup rutera (oni povezani sa računarima koji su pridruženi višeznačnoj grupi) bi u stvari trebalo da primaju vešeznačno rutirani saobraćaj. Na slici 4.49, samo ruteri: *A*, *B*, *E* i *F* bi trebalo da primaju višeznačno rutiran saobraćaj. Pošto nijedan računar, povezan sa ruterom *D*, nije pridružen višeznačnoj grupi, a pošto sa ruterom *C* nije povezan nijedan računar, ruteri *C* i *D* ne bi trebalo da primaju saobraćaj višeznačne grupe. Stoga, cilj višeznačnog rutiranja je da se pronađe stablo linkova koje povezuje sve rutere na koje su priključeni računari koji pripadaju višeznačnoj grupi. Višeznačno poslati paketi rutiraju se duž tog stabla, od pošiljaoca do svih računara koji pripadaju višeznačnom stablu. Naravno, ovo stablo može da obuhvata i rutere bez priključenih računara koji pripadaju višeznačnoj grupi (na primer, na slici 4.49, nemoguće je povezati rutere *A*, *B*, *E* i *F* u stablo, bez učešća bilo rutera *C* ili rutera *D*).

U praksi se za određivanje stabla višeznačnog rutiranja koriste dva rešenja, pri čemu su oba već razmatrana u slučaju difuznog rutiranja, tako da ćemo ih ovde samo ukratko spomenuti. Ta dva rešenja razlikuju se po tome da li se za distribuiranje saobraćaja od *svih* pošiljalaca u grupi koristi jedno stablo zajedničko za grupu, ili se za svakog pojedinačnog pošiljaoca pravi zasebno stablo rutiranja.

- *Višeznačno rutiranje pomoću stabla zajedničkog za grupu.* Kao u slučaju difuznog rutiranja sveobuhvatnim stablom, višeznačno rutiranje preko stabla zajedničkog za grupu zasniva se na izgradnji stabla koje obuhvata sve rubne rutere sa priključenim računarima koji pripadaju višeznačnoj grupi. U praksi se za pravljenje stabla za višeznačno rutiranje koristi rešenje sa centralnim čvorom, u

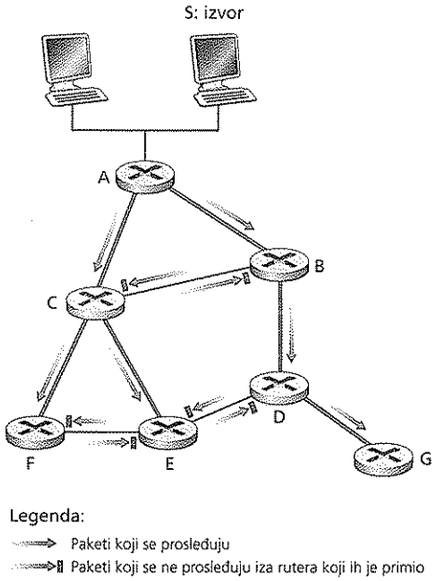
kome rubni ruteri sa priključenim računarima koji pripadaju višeznačnoj grupi šalju (jednoznačne) poruke pristupanja, adresirane na centralni čvor. Kao i u slučaju difuznog slanja, poruka o pristupanju prosleđuje se pomoću jednoznačnog rutiranja prema centru, sve dok ne stigne do rutera koji već pripada višeznačnom stablu, ili ne stigne u centralni čvor. Svi ruteri duž putanje kojom prolazi poruka o pristupanju posle toga prosleđuju primljene višeznačne pakete prema rubnom ruteru koji je pokrenuo pridruživanje višeznačnoj grupi. Najvažnije pitanje kod stabla za višeznačno rutiranje sa centralnim čvorom je postupak koji se koristi za biranje centralnog čvora. Algoritmi za biranje centralnog čvora razmatraju se u [Wall 1980; Thaler 1997; Estrin 1997].



Slika 4.49 ◆ Višeznačni računari, ruteri sa kojima su povezani i ostali ruteri

- *Višeznačno rutiranje pomoću zasebnog stabla za svaki izvor.* Dok se u slučaju zajedničkog stabla za višeznačno rutiranje pravi jedno, zajedničko stablo rutiranja, kojim se paketi rutiraju od *svih* pošiljaoca, u drugom rešenju pravi se višeznačno stablo rutiranja za *svaki* izvor u višeznačnoj grupi. U praksi se za izgradnju stabla za višeznačno prosleđivanje višeznačnih datagrama koji potiču od izvora x koristi RPF algoritam (sa izvornim čvorom x). Algoritam RPF za difuzno rutiranje, koji smo ranije proučili, potrebno je malo prilagoditi, kako bi mogao da se koristi za višeznačno rutiranje. Da bismo shvatili zašto, posmatrajmo ruter D na slici 4.50. Algoritam RPF za difuzno rutiranje prosleđuje pakete ruteru G, mada sa ruterom G nije povezan nijedan računar koji je pristupio višeznačnoj grupi. To i 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 rutera! Svaki od tih rutera primio bi neželjene višeznačno poslate

pakete. (Tako nešto nije sasvim nemoguće. Za prvobitnu verziju mreže *MBone* [Casner 1992; Macedonia 1994], prve globalne mreže za višeznačno rutiranje, ovo je bio prvenstveni problem). Rešenje problema sa primanjem neželjenih višeznačno poslatih paketa u algoritmu RPF naziva se **orezivanje** (eng. pruning). Višeznačni ruter koji primi višeznačne pakete, a nema priključene računare koji su pristupili određenoj grupi, svom uzvodnom ruteru šalje poruku kojom se orezuje. Ruter koji primi poruke orezivanja od svih svojih nizvodnih rutera može uzvodno da prosledi poruku kojom se orezuje.



Slika 4.50 ◆ Prosleđivanje na osnovu obrnute putanje u slučaju višeznačnog rutiranja

Višeznačno rutiranje na internetu

Prvi protokol za višeznačno rutiranje koji se koristio na internetu bio je **protokol za višeznačno rutiranje sa vektorom rastojanja** (Distance-Vector Multicast Routing Protokol, DVMRP) [RFC 1075]. U protokolu DVMRP primenjuju se zasebna stabla za svaki izvor sa prosleđivanjem na osnovu obrnute putanje i orezivanje. DVMRP koristi algoritam RPF sa orezivanjem, o čemu smo ranije govorili. Možda najviše korišćeni protokol za višeznačno rutiranje na internetu je **protokol višeznačnog rutiranja nezavisno od protokola** (Protocol-Independent Multicast, PIM routing protocol) u kome se jasno razlikuju dva slučaja za višeznačno rutiranje. U gustom režimu rada (eng. dense mode) [RFC 3973], članovi višeznačne grupe su gusto smešteni, odnosno mnoge rutere ili većinu njih, u određenom području bi trebalo uključiti u rutiranje višeznačnih datagrama. Gusti režim rada protokola PIM je tehnika prosleđivanja sa plavljenjem i orezivanjem na osnovu obrnute putanje, po mnogo čemu slična protokolu DVMRP.

U retkom režimu rada (eng. sparse mode) [RFC 4601], broj rutera sa povezanim članovima grupe je mali u odnosu na ukupan broj rutera, odnosno članovi grupe su raštrkani. U retkom režimu rada protokola PIM koriste se tačke susreta za postavljanje višeznačnog stabla rutiranja. U **višeznačnom rutiranju određenog izvora** (source-specific multicast, SSM) [RFC 3569, RFC 4607], samo se zasebnom pošiljaocu omogućava da šalje saobraćaj u višeznačno stablo, čime se izgradnja i održavanje stabla znatno pojednostavljuje.

Kada se unutar nekog domena koriste PIM i DVMP, administrator mreže može da konfiguriše višeznačne rutere unutar domena, po mnogo čemu na isti način na koji se konfigurišu jednoznačni protokoli rutiranja unutar domena, kao što su RIP, IS-IS i OSPF. Međutim, šta se događa kada su između različitih domena neophodne višeznačne rute? Da li postoji višeznačni protokol sličan BGP protokolu za rutiranje između domena? Odgovor je (bukvalno) da postoji. Dokument [RFC 4271] definiše proširenje protokola BGP za više protokola kojim se omogućava prenošenje informacija rutiranja drugih protokola, uključujući i informacije višeznačnog rutiranja. Protokol MSDP (Multicast Source Discovery Protocol) [RFC 3618, RFC 4611] može da se koristi za povezivanje tačaka susreta različitih domena u kojima se koristi redak režim rada protokola PIM. Odličan primer trenutnog stanja višeznačnog rutiranja na internetu je dokument [RFC 5110].

Zaključimo našu raspravu o višeznačnom rutiranju napomenom da je pred njim dugačak put. Zanimljivu raspravu o trenutnom modelu usluga višeznačnog rutiranja na internetu i problemima primene moguće je pronaći u [Diot 2000, Sharma 2003]. Ipak, uprkos tome da se još uvek ne koristi puno, višeznačno rutiranje na nivou mrežnog sloja je daleko od toga da je „mrtvo". Višeznačni saobraćaj se već godinama prenosi na internetu 2 i u mrežama sa kojima je ravnopravno deli [Internet 2 Multicast 2012]. U Velikoj Britaniji, bilo je pokušaja da BBC svoj program emituje preko višeznačnog IP rutiranja [BBC Multicast 2012]. Istovremeno, višeznačno rutiranje na aplikativnom nivou, kao što je slučaj sa internet televizijom *PPLive*, što smo videli sa u poglavlju 2 i drugim sistemima ravnopravnih računara, kao što je višeznačno rutiranje krajnjeg sistema (engl. End System Multicast) [Chu 2002], obezbeđuju višeznačno rutiranje sadržaja između ravnopravnih korisnika, korišćenjem protokola za višeznačno rutiranje na aplikativnom sloju (a ne na mrežnom sloju). Da li će buduća usluga višeznačnog rutiranja biti prvenstveno ostvarivana na mrežnom sloju (u jezgru mreže) ili na aplikativnom sloju (po obodu mreže)? Mada trenutna moda da se za distribuciju sadržaja koriste rešenja sa ravnopravnim korisnicima neznatnu prednost daje višeznačnom rutiranju na aplikativnom sloju, bar u bliskoj budućnosti, čini se stalni napredak u IP višeznačnom rutiranju, a ponekad je ta trka spora i ujednačena.

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 skupu protokola.

Saznali smo da ruter možda mora istovremeno da obradi na milione protoka paketa između različitih parova izvora i odredišta. Da bi ruter mogao da obradi tako veliki broj protoka, projektanti su tokom godina zaključili da bi zadaci rutera trebalo da ostanu što jednostavniji. Da bi se pojednostavio posao rutera, preduzimaju se različite mere, uključujući korišćenje datagrama na mrežnom sloju umesto mrežnog sloja sa virtuelnim kolima, upotrebu pojednostavljenog zaglavlja nepromenljive dužine (kao u protokolu IPv6), onemogućavanje fragmentacije (što je takođe urađeno u protokolu IPv6), kao i pružanje samo usluge najboljeg pokušaja. U tome je možda najvažnije to da se *ne* vodi računa o pojedinačnim tokovima, već da se odluke o rutiranju donose jedino na osnovu hijerarhijski struktuiranim odredišnim adresama u datagramima. Zanimljivo je primetiti da poštanska služba godinama koristi upravo ovakvo rešenje.

U ovom poglavlju takođe smo razmotrili osnovne principe algoritama rutiranja. Naučili smo da algoritmi rutiranja mrežu računara predstavljaju u obliku grafa sa čvorovima i linkovima. Polazeći od takve predstave, moguće je iskoristiti bogatu teoriju pronalaženja najkraćih putanja za rutiranje u grafovima, koja se preko 40 godina razvija u oblasti operacionih istraživanja i algoritama. Videli smo da postoje dva šira pristupa: centralizovani (globalni) pristup u kojem svaki čvor dobija kompletnu mapu mreže i nezavisno primenjuje algoritam za rutiranje najkraćom putanjom i decentralizovani pristup u kojem pojedinačni čvorovi imaju samo delimičnu sliku cele mreže, a čvorovi sarađuju da bi isporučili pakete duž najkraćih ruta. Takođe smo proučavali kako se hijerarhijski rešava problem narastanja mreže tako što se velike mreže dele u nezavisne administrativne domene, koji se nazivaju autonomni sistemi (AS). Svaki AS nezavisno rutira svoje datagrame kroz autonomni sistem, kao što svaka zemlja nezavisno raznosi svoju poštu po državi. Saznali smo kako se centralizovana, decentralizovana i hijerarhijska rešenja ostvaruju u glavnim protokolima rutiranja na internetu: RIP, OSPF i BGP. Proučavanje algoritama rutiranja završili smo razmatranjem difuznog i višeznačnog rutiranja.

Zaključujemo proučavanje mrežnog sloja i prelazimo korak niže u skupu protokola, tačnije, na sloj veze. Slično kao i mrežni sloj i sloj veze deo je jezgra mreže. Ali, u sledećem poglavlju videćemo da je uloga sloja veze ograničena samo na zadatak prenošenja paketa između čvorova na istom linku ili LAN mreži. Mada se ovaj zadatak naizgled čini manje važnim u poređenju sa zadacima mrežnog sloja, videćemo da sloj veze postavlja niz značajnih i zanimljivih pitanja, koja će nas dugo zaokupljati.

Domaći zadatak: problemi i pitanja

Poglavlje 4 Kontrolna pitanja

ODELJCI 4.1 – 4.2

R1. Podsetimo se terminologije koja se koristi u ovoj knjizi. Sećate se da se paket transportnog sloja naziva *segment*, a da se paket sloja veze naziva *okvir*. Kako se naziva paket mrežnog sloja? Sećate da se i ruteri i komutatori sloja veze nazivaju *komutatori 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.

R2. Koje su dve najvažnije funkcije mrežnog sloja u mrežama sa datagramima? Koja su tri najvažnija zadatka mrežnog sloja u mrežama sa virtuelnim kolima?

R3. Kakva je razlika između rutiranja i prosleđivanja?

R4. 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 vrste mreža.

R5. Opišite neku hipotetičku uslugu koju bi mrežni sloj mogao da obezbedi pojedinačnom paketu. Uradite to isto za tok paketa. Da li neke od usluga koje ste zamislili već postoje na mrežnom sloju interneta? Da li postoje u CBR modelu usluge ATM mrežâ? Da li postoje u ABR modelu usluge ATM mrežâ?

R6. Navedite neke aplikacije koje koriste CBR model usluge ATM mrežâ.

ODELJAK 4.3

R7. Objasnite zašto svaki ulazni port u ruteru velike brzine čuva kopiju tabele prosleđivanja?

R8. U odeljku 4.3 razmatraju se tri vrste komutatorske mreže. Navedite koje su i ukratko ih opišite. Koja bi, ako postoji, mogla da šalje više paketa duž mreže istovremeno?

R9. Opišite kako dolazi do gubitka paketa u ulaznim portovima. Opišite kako se može izbeći gubitak paketa u ulaznom portu (bez korišćenja beskonačno velikog bafera).

R10. Opišite kako dolazi do gubitka paketa u izlaznim portovima. Da li povećanje brzine komutatorske mreže može da spreči ovaj gubitak?

R11. Šta je to HOL blokiranje? Da li se ono dešava u ulaznim ili u izlaznim portovima?

ODELJAK 4.4

R12. Da li ruteri imaju IP adrese? Ako imaju, koliko ih imaju?

R13. Koji je 32-bitni binarni ekvivalent IP adrese 223.1.3.27?

R14. Posetite računar koji koristi DHCP za dobijanje IP adrese, mrežne maske, podrazumevanog rutera i IP adrese svog lokalnog DNS servera. Navedite te vrednosti.

R15. Pretpostavimo da između izvornog i odredišnog računara postoje tri rutera. Ako zanemarimo fragmentaciju, preko koliko interfejsa prelazi IP datagram koji se pošalje od izvornog do odredišnog računara? Koliko tabela prosleđivanja će se indeksirati, da bi se datagram preneo od izvora do odredišta?

R16. Pretpostavimo da neka aplikacija svakih 20 ms proizvodi odsečke od po 40 bajtova podataka i da se svaki takav odsečak enkapsulira u TCP segment, a zatim u IP datagram. Koji procenat takvog datagrama pripada dopunskim podacima zaglavlja, a koji podacima aplikacije?

R17. Pretpostavimo da računar A šalje TCP, segment enkapsuliran u IP datagram, računaru B. Kada računar B primi datagram, kako mrežni sloj računara B zna da bi taj segment (odnosno, korisne podatke datagrama) trebalo da preda protokolu TCP, a ne protokolu UDP ili nekom drugom?

R18. Pretpostavimo da ste nabavili bežični ruter i povezali ga sa kablovskim modemom. Pretpostavimo takođe da vaš posrednik za internet usluge dinamički dodeli vašem povezanom uređaju (tj. bežičnom ruteru) jednu IP adresu. Takođe, pretpostavimo da u kući imate pet PC računara koji koriste 802.11 za bežično povezivanje sa bežičnim ruterom. Kako se IP adrese dodeljuju PC računarima? Da li bežični ruter koristi NAT? Zašto ga koristi, ili zašto ga ne koristi?

R19. Opišite po čemu se razlikuju poljâ zaglavlja kod protokola IPv4 i IPv6. Da li postoje neka zajednička poljâ?

R20. Kaže se da kada se IPv6 tuneluje kroz IPv4 rutere, IPv6 posmatra IPv4 tunele kao protokole sloja veze. Da li se slažete sa tim tvrđenjem? Zašto se slažete, ili zašto se ne slažete?

ODELJAK 4.5

R21. Opišite po čemu se razlikuju algoritmi rutiranja prema stanju linkova i algoritmi sa vektorom rastojanja.

R22. Objasnite kako je hijerarhijsko organizovanje interneta omogućilo proširivost na milione korisnika.

R23. Da li je neophodno da svi autonomni sistemi koriste isti algoritam rutiranja unutar autonomnih sistema? Zašto?

ODELJAK 4.6

R24. Posmatrajmo sliku 4.37. Uzmimo prvobitnu tabelu u *D* i pretpostavimo da *D* primi od *A* sledeću objavu:

Odredišna podmreža	Sledeći ruter	Broj skokova do odredišta
z	C	10
w	–	1
x	–	1
....	....	....

Da li će se tabela u *D* promeniti? Ako hoće, kako?

R25. Opišite po čemu se razlikuju objave koje koristi protokol RIP i protokol OSPF.

R26. Popunite praznu liniju: RIP objave obično objavljuju broj skokova do različitih odredišta. S druge strane, BGP ažuriranja objavljuju _____ do različitih odredišta.

R27. Zašto se na internetu unutar i između autonomnih sistema koriste različiti protokoli?

R28. Zašto je politika rutiranja toliko značajna u protokolima unutar autonomnih sistema (kao što su OSPF i RIP), koliko je značajna i u protokolima između autonomnih sistema (kao što je BGP)?

R29. Definišite sledeće pojmove i opišite razlike između njih: *podmreža*, *prefiks* i *BGP ruta*.

R30. Kako protokol BGP koristi atribut NEXT-HOP? Kako koristi atribut AS-PATH?

R31. Opišite kako mrežni administrator posrednika višeg reda ostvaruje poštovanje određenih pravila ponašanja prilikom konfigurisanja protokola BGP.

ODELJAK 4.7

R32. Koja je najvažnija razlika između toga kada se difuzno slanje ostvaruje pomoću višestrukog jednoznačnog rutiranja i difuznog rutiranja koje podržavaju mreža i ruter?

R33. Za svaki od tri pristupa za difuzno slanje datagrama (nekontrolisano plavljenje, kontrolisano plavljenje i difuzno rutiranje sveobuhvatnim stablom) koje smo obradili, odgovori da li su sledeće izjave tačne ili netačne ? Može se poći od pretpostavke da se paketi ne gube zbog prepunjavanja bafera i da se svi paketi na linku isporučuju po redosledu po kome su poslati.

a. Čvor može da primi više kopija istog paketa.

b. Čvor može da prosledi više kopija paketa preko istog izlaznog linka.

R34. Kada računar pristupi višeznačnoj grupi, da li mora da promeni svoju IP adresu u adresu višeznačne grupe kojoj pristupa?

R35. Koje uloge igraju protokol IGMP i protokol za višeznačno rutiranje širokopojasne mreže?

R36. Koja je razlika između stabla koje je zajedničko za grupu i stabla koje zavisi od izvora u smislu višeznačnog rutiranja?

Problemi

P1. Navedite neke od razloga za i protiv mrežâ sa datagramima i mrežâ sa virtuelnim kolima.

a. Pretpostavimo da su ruteri podvrgnuti otežanim uslovima zbog kojih češće otkazuju. Da li to ide u prilog arhitekturi sa datagramima ili sa virtuelnim kolima? Zašto?

b. Pretpostavimo da čvorovi izvora i odredišta zahtevaju fiksnu količinu kapaciteta koji će uvek biti raspoloživ svim ruterima na putanji između izvornog i odredišnog čvora, za isključivu upotrebu protoka saobraćaja između ovog izvornog i odredišnog čvora. Da li se takvo rešenje lakše ostvaruje u arhitekturi sa datagramima ili sa virtuelnim kolima? Zašto?

c. Pretpostavimo da linkovi i ruteri u mreži nikada ne otkazuju i da su putanje rutiranja koje se koriste između svih parova izvor/odredište konstante. U ovom scenariju, da li arhitektura virtuelnog kola ili datagrama ima veću kontrolu nad vršnom brzinom saobraćaja? Zašto?

P2. Uzmimo za primer mrežu sa virtuelnim kolima. Pretpostavimo da je VC broj 8-bitno polje.

a. Koji je najveći broj virtuelnih kola koja mogu da se prenose preko linka?

b. Pretpostavimo da centralni čvor određuje putanje i VC brojeve prilikom uspostavljanja veze. Pretpostavimo da se na svakom linku duž VC putanje koristi isti VC broj. Opišite kako bi centralni čvor mogao da utvrdi VC broj prilikom uspostavljanja veze. Da li je moguće da je broj aktivnih virtuelnih kola manji od najvećeg broja koji je određen u delu (a), zato što nema slobodnih VC brojeva?

c. Pretpostavimo da je dozvoljena upotreba različitih VC brojeva na linkovima duž VC putanje. Prilikom uspostavljanja veze, nakon što se utvrdi putanja sa kraja na kraj, opišite kako linkovi decentralizovano biraju VC brojeve i konfigurišu svoje tabele prosleđivanja bez oslanjanja na centralni čvor.

P3. Osnovna tabela prosleđivanja u mreži sa virtuelnim kolima ima četiri kolone. Šta znače vrednosti u tim kolonama? Osnovna tabela prosleđivanja u mreži sa datagramima ima dve kolone. Šta znače vrednosti u tim kolonama?

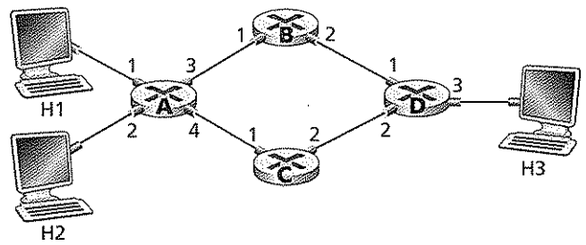
P4. Uzmimo za primer mrežu dole navedenu.

a. Pretpostavimo da je ova mreža datagramska mreža. Prikažite tabelu prosleđivanja u ruteru A, tako da se sav saobraćaj, upućen na računar H3, prosleđuje preko interfejsa 3.

b. Pretpostavimo da je ova mreža datagramska mreža. Da li možete da upišete tabelu prosleđivanja u ruter A, tako da sav saobraćaj od H1 koji se upućuje na računar H3, bude prosleđen preko interfejsa 3; a sav saobaraćaj usmeren sa računara H2 do računara H3, bude prosleđen preko interfejsa 4? (Savet: ovo je trik pitanje.)

c. Pretpostavimo da je ova mreža, mreža virtuelnog kola, i da se između računara H1 i H3 odvija jedna veza, a druga između računara H2 i H3. Upišite tabelu prosleđivanja u ruter A, tako što se sav saobraćaj, koji je usmeren od računara H1 do računara H3, prosleđuje preko interfejsa 3, dok se sav saobraćaj, usmeren od računara H2 od računara H3, prosleđuje preko interfejsa 4.

d. Pretpostavite isti scenario kao pod stavkom (c), upišite tabelu prosleđivanja u čvorove B, C i D.



P5. Uzmimo za primer mrežu sa virtuelnim kolima sa 2-bitnim poljem za VC broj. Pretpostavimo da ta mreža želi da uspostavi virtuelno kolo preko četiri

linka: A, B, C i D. Pretpostavimo 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 jedan od navedenih linkova.

a. Ako se zahteva da svako virtuelno koristi isti VC broj na svim linkovima duž svoje putanje, koji VC broj bi mogao da se dodeli novom virtuelnom kolu?

b. Ako je dopušteno da svako virtuelno kolo ima različite VC brojeve na raznim linkovima duž svoje putanje (tako da tabela prosleđivanja mora da obavlja prevođenje VC brojeva), koliko različitih kombinacija četiri VC broja (po jedan za svaki od četiri linka) može da se iskoristi?

P6. U tekstu smo koristili izraz *usluga sa vezom* za opisivanje odgovarajuće usluge transportnog sloja i izraz *usluga veze* za opisivanje usluge mrežnog sloja. Objasnite tu suptilnu razliku u terminologiji?

P7. Pretpostavimo da dva paketa stignu do dva različita ulazna porta rutera u isto vreme. Takođe, pretpostavimo da u ruteru nema nijednog drugog paketa.

a. Pretpostavimo da bi dva paketa trebalo da se proslede do dva *različita* izlazna porta. Da li je moguće proslediti dva paketa preko komutatorske mreže u isto vreme kada se kao komutatorska mreža koristi *zajednička magistrala*?

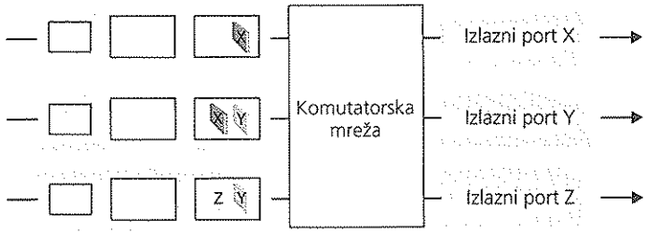
b. Pretpostavimo da bi dva paketa trebalo da se proslede do dva *različita* izlazna porta. Da li je moguće proslediti dva paketa preko komutatorske mreže u isto vreme kada se kao komutatorska mreža koristi *unakrsna mreža*?

c. Pretpostavimo da bi dva paketa trebalo da se proslede do *istog* izlaznog porta. Da li je moguće proslediti dva paketa preko komutatorske mreže u isto vreme kada se kao mreža koristi *unakrsna mreža*?

P8. U odeljku 4.3 napomenuli smo da je maksimalno čekanje u redu $(n - 1)D$, ako je komutatorska mreža n puta brža od ulazne linije. Pretpostavimo da su svi paketi iste dužine, n paketa stigne u isto vreme do n ulaznih portova, a svih n paketa bi trebalo da budu prosleđeni do *različitih* izlaznih portova. Koliko je maksimalno kašnjenje za paket u slučaju komutatorske mreže, zasnovane na: (a) memoriji, (b) magistrali i (c) unakrsnoj komutatorskoj mreži?

P9. Uzmimo u obzir komutator prikazan na slici ispod. Pretpostavimo da svi datagrami imaju istu fiksiranu dužinu, da komutatori funkcionišu na vremenski raspodeljen, sinhronizovan način i da u jednom vremenskom periodu datagram može da se prenese od ulaznog do izlaznog porta. Komutatorska

mreža je unakrsna, tako da najviše jedan datagram može da se prenese do zadatog izlaznog porta u vremenskom periodu, ali drugi izlazni portovi mogu da prime datagrame sa različitih ulaznih portova u jednom vremenskom intervalu. Koji je minimalan broj vremenskih perioda potreban za prenošenje prikazanih paketa od ulaznih portova do njihovih izlaznih portova, ako pretpostavite bilo koji raspored ulaznog reda za čekanje (tj. ne mora da ima HOL blokiranje)? Koji je najveći broj vremenskih perioda potreban, ako pretpostavite najgori mogući rapsored koji možete da zamislite, pretpostavljajući da nijedan ulazni red za čekanje koji nije prazan nije besposlen?



P10. Zamislimo mrežu sa datagramima u kojoj se koristi 32-bitno adresiranje. Pretpostavimo da ruter ima četiri linka, označena od 0 do 3, a da bi pakete trebalo proslediti interfejsima linkova na sledeći način:

Raspon odredišnih adresa	**Interfejs linka**
11100000 00000000 00000000 00000000 do 11100000001111111111111111111111	0
11100000 01000000 00000000 00000000 do 11100000 01000000 11111111111111111 11100000 01000001 00000000 00000000 do 11100001011111111111111111111111	1 2
ostalo	3

a. Napravite tabelu prosleđivanja koja ima četiri stavke, koristi preklapanje najdužeg prefiksa i prosleđuje pakete na odgovarajuće interfejse linkova.

b. Opišite kako bi vaša tabela prosleđivanja odredila odgovarajući interfejs linka za datagrame sa sledećim odredišnim adresama:

11001000 10010001 01010001 01010101
11100001010000001100001100111100
11100001 10000000 00010001 01110111

P11. Zamislimo mrežu sa datagramima u kojoj se koristi 8-bitno adresiranje. Pretpostavimo da ruter koristi preklapanje najdužeg prefiksa i sledeću tabelu prosleđivanja:

Preklapanje prefiksa	Interfejs
00	0
010	1
011	2
10	2
11	3

Za sva četiri interfejsa navedite raspon odredišnih adresa računara i broj adresa u odgovarajućim rasponima.

P12. Zamislimo mrežu sa datagramima u kojoj se koristi 8-bitno adresiranje. Pretpostavimo da ruter koristi preklapanje najdužeg prefiksa i sledeću tabelu prosleđivanja:

Preklapanje prefiksa	Interfejs
1	0
10	1
111	2
ostalo	3

Za sva četiri interfejsa navedite raspon odredišnih adresa računara i broj adresa u odgovarajućim rasponima.

P13. Zamislimo ruter koji povezuje tri podmreže: podmreža 1, podmreža 2 i podmreža 3. Pretpostavimo da se zahteva da svi interfejsi, u sve tri podmreže, imaju prefiks 223.1.17/24. Takođe pretpostavimo da se zahteva da podmreža 1 podržava minimalno 60 interfejsa, podmreža 2 minimalno 90 interfejsa, a podmreža 3 najmanje 12 intrefejsa. Odredite tri mrežne adrese (oblika a.b.c/x) koje zadovoljavaju postavljene zahteve.

P14. U odeljku 4.2.2 dat je primer tabele prosleđivanja (koja koristi preklapanje najdužeg prefiksa). Prepišite tu tabelu prosleđivanja u obliku a.b.c/x, umesto u obliku binarnih nizova.

P15. U problemu P10 traženo je da napravite tabelu prosleđivanja (koristeći preklapanje najdužeg prefiksa). Prepišite tu tabelu prosleđivanja u obliku a.b.c/x, umesto u obliku binarnih nizova.

P16. Zamislimo podmrežu sa prefiksom 128.119.40.128/26. Navedite primer IP adrese (oblika xxx.xxx.xxx.xxx) koja se može dodeliti u ovoj mreži. Pretpostavimo da neki posrednik za internet usluge poseduje blok adresa oblika 128.119.40.64/26. Pretpostavimo da on želi da od tog bloka napravi četiri podmreže sa podjednakim brojem IP adresa. Koji su prefiksi (oblika a.b.c.d/x) te četiri podmreže.

P17. Uzmimo na primer topologiju prikazanu na slici 4.17. Označimo tri podmreže sa računarima (počevši 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 šest podmreža, uz sledeća ograničenja: sve adrese bi trebalo dodeliti iz opsega 214.97.254/23; podmreža A bi trebalo da ima dovoljno adresa da podrži 250 interfejsa; podmreža B bi trebalo da ima dovoljno adresa da podrži 120 interfejsa; podmreža C bi trebalo 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 adrese treba dodeliti u obliku a.b.c.d/x ili a.b.c.d/x - e.f.g.h/y.

b. Na osnovu odgovora pod (a), napravite tabele prosleđivanja (koristeći preklapanje najdužeg prefiksa) za sva tri rutera.

P18. Koristite uslugu imena domena Američkog registra internet brojeva (http://www.arin.net/whois) da odredite blokove IP adresa za tri univerziteta. Da li usluga imena domena može da se koristi za utvrđivanje tačne geografkse lokacije određene IP adrese? Pomoću adrese www.maxmind.com utvrdite lokacije veb servera za svaki od ovih univerziteta.

P19. Uzmimo na primer slanje datagrama od 2 400 bajtova na link čiji je MTU jednak 700 bajtova. Pretpostavimo da je izvorni datagram označen identifikacionim brojem 422. Koliko fragmenata bi trebalo napraviti? Koje su vrednosti u raznim poljima dobijenih IP datagrama vezanim za fragmentaciju?

P20. Pretpostavimo da su između izvornog računara A i odredišnog računara B datagrami ograničeni na 1 500 bajtova (uključujući zaglavlje). Ako pretpostavimo da je IP zaglavlje 20 bajtova, koliko bi datagrama bilo potrebno da se pošalje jedan MP3 od 5 miliona bajtova? Objasnite kako ste izračunali odgovor.

P21. Uzmimo za primer postavku mreže sa slike 4.22. Pretpostavimo da je posrednik za internet usluge dodelio ruteru adresu 24.34.112.235, a da je mrežna adresa kućne mreže 192.168.1/24.

a. Dodelite adrese svim interfejsima u kućnoj mreži.

b. Pretpostavimo da svi računari imaju dve izlazne TCP veze, obe sa portom 80 računara 128.119.40.86. Navedite šest odgovarajućih stavki za NAT tabelu prevođenja.

P22. Pretpostavimo da ste zainteresovani za određivanje broja računara u NAT sistemu. Primetićete da se IP sloj označava identifikacionim brojem, redom na svaki IP paket. Identifikacioni broj prvog IP paketa koji je proizveo računar je slučajan broj, a identifikacioni brojevi narednih IP paketa se redom dodeljuju. Pretpostavimo da su svi IP paketi koje je proizveo računar iza NAT rutera poslati u spoljni svet.

a. Na osnovu ovog zapažanja i ako pretpostavimo da možemo da nadgledamo sve pakete koje je NAT sistem poslao u spoljni svet, da li možete da izdvojite jednostavnu tehniku koja otkriva broj jedinstvenih računara iza NAT rutera? Obrazložite svoj odgovor.

b. Ako se identifikacioni brojevi ne dodeljuju redom već nasumično, da li će ova tehnika funkcionisati? Obrazložite svoj odgovor.

P23. U ovom problemu analiziramo uticaj NAT sistema na P2P aplikacije. Pretpostavimo da ravnopravni učesnik sa korisničkim imenom Arnold, koristeći upit, otkrije da učesnik sa korisničkim imenom Bernard ima datoteku koju želi da preuzme. Pretpostavimo takođe da se Bernard i Arnold nalaze iza NAT rutera. Pokušajte da osmislite tehniku koja bi Arnoldu omogućila sa uspostavi TCP vezu sa Bernardom bez posebnog konfigurisanja NAT rutera za određenu aplikaciju. Ukoliko ne možete da osmislite takav način, objasnite zašto.

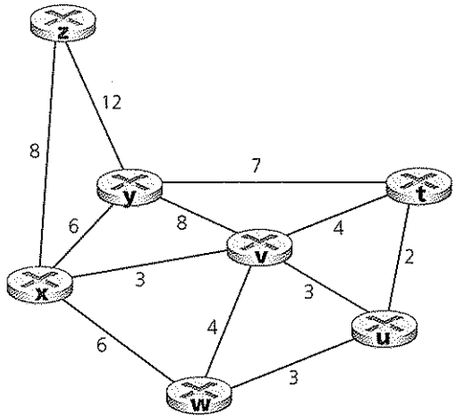
P24. Na slici 4.27 navedite putanje od v do y koje ne sadrže petlje.

P25. Ponovite problem P24 za putanje od x do z, z do u i z do w.

Video napomena

Dijkstrin alg ritam: iskusija i primer

P26. Uzmimo na primer sledeću mrežu. Sa naznačenim troškovima linkova, iskoristite Dijkstrin algoritam najkraće putanje da biste izračunali najkraću putanju od x do svih čvorova u mreži. Svoje proračune prikažite u tabeli, sličnoj tabeli 4.3.



P27. Posmatrajmo mrežu prikazanu u problemu P26. Koristeći Dijsktrin algoritam i prikazujući svoj rad, koristeći tabelu sličnu tabeli 4.3, uradite sledeće:

a. izračunajte najkraću putanju od t do svih ostalih čvorova u mreži;

b. izračunajte najkraću putanju od u do svih ostalih čvorova u mreži;

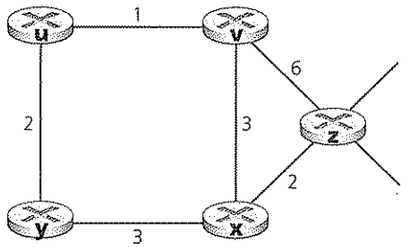
c. izračunajte najkraću putanju od v do svih ostalih čvorova u mreži;

d. izračunajte najkraću putanju od w do svih ostalih čvorova u mreži;

e. izračunajte najkraću putanju od y do svih ostalih čvorova u mreži;

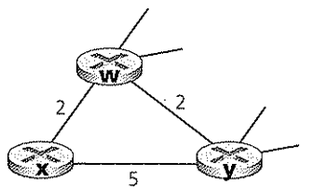
f. izračunajte najkraću putanju od z do svih ostalih čvorova u mreži.

P28. Posmatrajmo mrežu prikazanu ispod i pretpostavimo da svaki čvor na početku zna troškove do svakog od svojih suseda. Koristeći algoritam vektora rastojanja, prikažite stavke tabele rastojanja u čvoru z.



P29. Uzmimo za primer opštu topologiju (odnosno, ne određenu mrežu prikazanu gore) i sinhronu verziju algoritma sa vektorom rastojanja. Pretpostavimo da pri svakoj iteraciji čvor razmenjuje svoje vektore rastojanja sa susedima i prima njihove vektore rastojanja. Ako pretpostavimo da algoritam u svakom čvoru počinje znajući jedino troškove do svojih neposrednih suseda, koji je najveći broj iteracija potreban da se distribuirani algoritam završi? Dokažite svoj odgovor.

P30. Posmatrajmo deo mreže prikazan na slici ispod. Čvor x povezan je samo sa dva suseda, w i y. Putanja najmanjih troškova čvora w do odredišta u (nije prikazano) iznosi 5, a putanja najmanjih troškova čvora y do odredišta u iznosi 6. Nisu prikazane čitave putanje od w i y do u (i između w i y). Svi troškovi linkova u mreži imaju isključivo celobrojne pozitivne vrednosti.



a. Navedite vektor rastojanja čvora x do odredišta w, y i u.

b. Navedite koliko bi trebalo da se promene troškovi linkova $c(x,w)$ ili $c(x,y)$ tako da x obaveštava svoje susede o novoj putanji najmanjih troškova do odredišta u, do koje je došlo nakon izvršavanja algoritma vektora rastojanja.

c. Navedite koliko mogu da se promene troškovi linkova $c(x,w)$ ili $c(x,y)$ tako da x *ne* obaveštava svoje susede o novoj putanji najmanjih troškova do odredišta u, do koje je došlo nakon izvršavanja algoritma vektora rastojanja.

P31. Uzmimo za primer topologiju sa tri čvora, prikazanu na slici 4.30. Umesto troškova linkova, prikazanih na slici 4.30, troškovi linkova su $c(x,y) = 3$, $c(y,z) = 6$, $c(z,x) = 4$. Izračunajte tabele rastojanja nakon početnog koraka i nakon svake iteracije sinhrone verzije algoritma vektora rastojanja (kao što smo već uradili prilikom razmatranja slike 4.30).

P32. Uzmite u obzir problem brojanja do beskonačnosti u rutiranju vektora rastojanja. Da li će se pojaviti problem brojanja do beskonačnoti ukoliko smanjimo troškove linka? Zašto? Šta bi se desilo, ako povežemo dva čvora koji nemaju link?

P33. Prodiskutujte da se za algoritam vektora rastojanja na slici 4.30, nijedna vrednost u vektoru rastojanja $D(x)$ ne povećava i da će se eventualno stabilizovati u konačnom broju koraka.

P34. Uzmite u obzir sliku 4.31. Pretpostavite da postoji još jedan ruter w, povezan sa ruterom y i z. Troškovi svih linkova su zadati na sledeći način: $c(x,y) = 4$, $c(x,z) = 50$, $c(y,w) = 1$, $c(z,w) = 1$, $c(y,z) = 3$. Pretpostavimo da se u algoritmu rutiranja vektora rastojanja koristi lažno rastjanje.

a. Kada se vektor rastojanja stabilizuje, ruteri w, y i z prenose svoja rastojanja do x jedan drugom. Koje vrednosti rastojanja saopštavaju jedni drugima?

b. Pretpostavimo sada da su se troškovi linka između x i y povećali na 60. Da li će se pojaviti problem brojanja do beskonačnosti ukoliko se koristi lažno rastojanje? Zašto da, ili zašto ne? Ako postoji problem brojanja do beskonačnosti, koliko je onda potrebno iteracija za rutiranje vektora rastojanja, kako bi se ponovo postiglo stabilno stanje? Obrazložite svoj odgovor.

c. Kako ćete promeniti $c(y,z)$ da se uopšte ne pojavi problem brojanja do beskonačnosti, ukoliko se $c(y,x)$ promeni sa 4 na 60.

P35. Opišite kako se u protokolu BGP otkrivaju petlje u putanjama.

P36. Da li će BGP ruter uvek birati putanju bez petlje sa najkaraćom dužinom AS putanje? Obrazložite svoj odgovor.

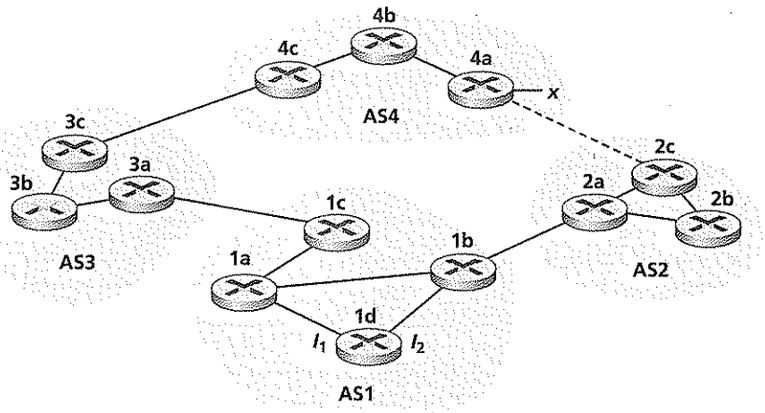
P37. Posmatrajmo mrežu prikazanu ispod. Pretpostavimo da AS3 i AS2 izvršavaju OSPF kao svoj protokol rutiranja unutar autonomnog sistema. Pretpostavimo da AS1 i AS4 izvršavaju RIP kao svoj protokol rutiranja unutar autonomnog sistema. Pretpostavimo da se eBGP i iBGP koriste za protokol rutiranja između autonomnih sistema. Za početak pretpostavimo da između AS2 i AS4 *ne postoji* fizička veza.

a. Od kog protokola rutiranja: OSPF, RIP, eBGP ili iBGP ruter *3c* saznaje o prefiksu x?

b. Od kog protokola rutiranja ruter 3a saznaje o prefiksu x?

c. Od kog protokola rutiranja ruter 1c saznaje o prefiksu x?

d. Od kog protokola rutiranja ruter 1d saznaje o prefiksu x?



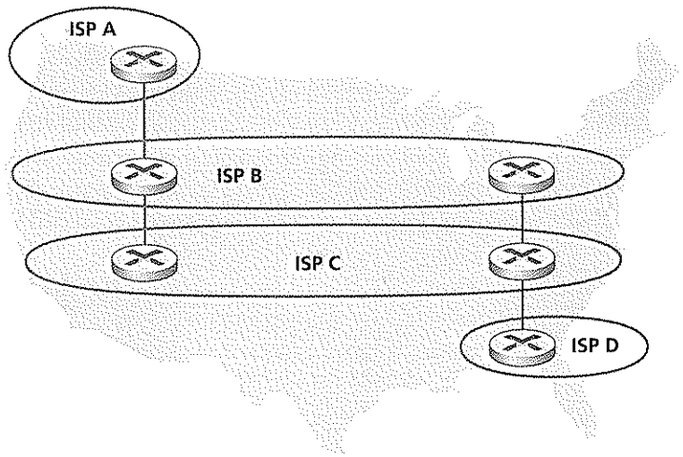
P38. U vezi sa prethodnim problemom, nakon što ruter 1d sazna za x, on u svoju tabelu prosleđivanja unosi stavku (x,I).

a. Da li će vrednost stavke I biti jednaka $I1$ ili $I2$? U jednoj rečenici objasnite zašto.

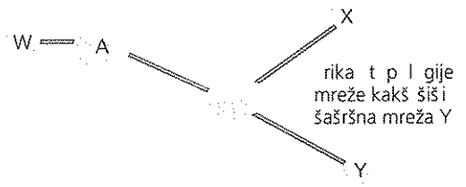
b. Sada pretpostavimo da postoji fizički link između AS2 i AS4, prikazan isprekidanom linijom. Pretpostavimo da ruter 1d sazna da je prefiksx dostupan preko AS2 kao i preko AS3. Da li će stavka I biti postavljena na I1 ili I2? U jednoj rečenici objasnite zašto.

c. Sada pretpostavimo da postoji još jedan autonomni sistem, nazvan AS5, koji leži na putanji između AS2 i AS4 (nije prikazan na crtežu). Pretpostavimo da ruter 1d sazna da je x dostupan preko AS2, AS5, AS4 kao i preko AS3, AS4. Da li će stavka I biti postavljena na I_1 ili I_2? U jednoj rečenici objasnite zašto.

P39. Posmatrajmo mrežu prikazanu ispod. Posrednik za internet usluge B nudi usluge državne okosnice regionalnom posredniku A. Posrednik C pruža usluge državne okosnice regionalnom posredniku D. Svi posrednici se sastoje od po jednog autonomnog sistema. Posrednici B i C su međusobno ravnopravno povezani na dva mesta i koriste protokol BGP. Posmatramo saobraćaj koji se kreće od A do D. Posrednik B bi više voleo da saobraćaj preda posredniku C na Zapadnoj obali (tako da C mora da snosi troškove prenošenja saobraćaja kroz celu zemlju), dok bi C više voleo da saobraćaj preuzme preko priključne tačke na Istočnoj obali (tako da B snosi troškove saobraćaja kroz celu zemlju). Koji bi mehanizam protokola BGP mogao da upotrebi C, tako da se saobraćaj od A za D preuzima u tački B na Istočnoj obali? Da biste odgovorili na ovo pitanje, moraćete dobro da proučite specifikaciju protokola BGP.



P40. Na slici 4.42 posmatramo informacije o putanjama koje stižu u završne mreže W, X i Y. Na osnovu informacija dostupnih u W i X, kako izgleda topologija mreže iz njihovog ugla? Objasnite svoj odgovor. Prikaz topologije mreže posmatran iz Y je prikazan ispod.



P41. Posmatrajte sliku 4.42 gde B nikada neće proslediti saobraćaj usmeren na Y preko X na osnovu rutiranja protokola BGP. Međutim, postoje neke veoma popularne aplikacije za koje paketi sa podacima prvo idu do X, a zatim teku do Y. Identifikujte jednu takvu aplikaciju, i opišite kako paketi sa podacima prate putanju koja nije zadata rutiranjem protokola BGP.

P42. Pretpostavite da na slici 4.42 postoji još jedna završna mreža V, koja je klijent posrednika za internet usluge A. Pretpostavimo da B i C imaju ravnopravni odnos, i da je posrednik A klijent i posrednika B i posrednika C. Pretpostavimo da A želi da ima saodraćaj do W, koji dolazi samo od B, i saobraćaj usmeren do V od B ili C. Kako bi A trebalo da objavi svoje putanje do B i C? Koje putanje autonomnih sistema C prima?

P43. Pretpostavimo da autonomni sistemi X i Z nisu direktno povezani, već su povezani pomoću autnomnog sistema Y. Pretpostavimo dalje da X ima ugovor o ravnopravnom odnosu sa Y, a da Y ima takav ugovor sa Z. Konačno, pretpostavimo da Z želi da prenese sav saobraćaj autonomnog sistema Y, ali ne želi da prenese saobraćaj autonomnog sistema X. Da li protokol BGP dozvoljava autonomnom sistemu Z da primeni ovakav postupak?

P44. Posmatrajmo mrežu sa sedam čvorova (sa čvorovima označenim od *t* do *z*) iz problema P26. Prikažite stablo najmanjih troškova koje počinje u čvoru *z*, koje obuhvata (kao krajnje računare) čvorove *u*, *v*, *w* i *y*. Svojim rečima objasnite zašto je stablo koje ste izabrali zaista stablo najmanjih troškova.

P45. Posmatrajmo dva osnovna načina kojima se postiže difuzno rutiranje: emulacijom jednoznačnog rutiranja i difuzno rutiranje na mrežnom sloju (tj. pomoću rutera) i pretpostavimo da se difuzno rutiranje na mrežnom sloju postiže difuznim rutiranjem, korišćenjem sveobuhvatnog stabla. Posmatramo slučaj sa jednim pošiljaocem i 32 primaoca. Pretpostavimo da je pošiljalac binarnim stablom rutera povezan sa primaocima. Koliki su troškovi slanja difuzno rutiranog paketa u slučaju emulacije jednoznačnim rutiranjem i u slučaju difuznog rutiranja na mrežnom sloju za ovu topologiju? Pri čemu uvek prilikom slanja paketa (ili kopije paketa) preko pojedinih linkova postoje određeni troškovi. Kakva topologija međusobnog povezivanja pošiljaoca, primalaca i rutera dovodi do najveće razlike između troškova emulacijom jednoznačnog rutiranja i troškova pravog difuznog rutiranja na mrežnom sloju? Možete izabrati onoliko rutera koliko god želite.

P46. Posmatrajmo način rada algoritma prosleđivanja na osnovu obrnute putanje (RPF) sa slike 4.44. Koristeći istu topologiju, pronađite skup putanja od svih čvorova do izvornog čvora *A* (i označite te putanje na grafu deblje osenčenim linijama, kao na slici 4.44) tako da, ako su to putanje najmanjih troškova, onda čvor *B*, korišćenjem protokola RPF, prima kopiju difuzno poslate poruke iz A od čvorova *A*, *C* i *D*.

P47. Posmatrajmo topologiju prikazanu na slici 4.44. Pretpostavimo da svi linkovi imaju jednake troškove i da je čvor *E* izvor difuzno poslate poruke. Koristeći strelice slične onima prikazanim na slici 4.44, označite linkove preko kojih će se paketi prosleđivati protokolom RPF i linkove preko kojih se paketi neće prosleđivati, ako se uzme da čvor *E* predstavlja izvor.

P48. Ponovite problem P47 pomoću grafikona za problem P26. Pretpostavite da je *z* izvor difuzne emisije i da su troškovi linka prikazani u problemu P26.

P49. Posmatrajmo topologiju prikazanu na slici 4.46 i pretpostavimo da svi linkovi imaju jednake troškove. Pretpostavimo da je čvor *C* izabran za centar u algoritmu za višeznačno rutiranje koje se zasniva na centralnom čvoru. Pod pretpostavkom da svi povezani ruteri za slanje poruka o pristupanju čvoru *C* koriste svoje putanje najmanjih troškova do čvora *C*, nacrtajte dobijeno stablo za višeznačno rutiranje, zasnovano na centralnom čvoru. Da li je dobijeno stablo jednako stablu najmanjih troškova? Obrazložite svoj odgovor.

P50. Ponovite problem P49 pomoću grafikona iz problema P26. Pretpostavimo da je *v* centralni čvor.

P51. U odeljku 4.5.1 proučavali smo Dijkstrin algoritam rutiranja stanja linkova za izračunavanje jednoznačnih putanja, koje su pojedinačno putanje najmanjih troškova od izvora do svih odredišta. Može se smatrati da unija tih putanja stvara **stablo putanja najmanjih jednoznačnih troškova** (ili stablo najkraćih jednoznačnih putanja, ako su troškovi svih linkova istovetni).

Izgradnjom suprotnog primera pokažite da stablo putanja najmanjih troškova *nije* uvek isto što i najmanje sveobuhvatno stablo.

P52. Zamislite mrežu u kojoj su svi čvorovi povezani sa tri druga čvora. U jednom vremenskom koraku čvor može da primi difuzno poslate pakete od svojih suseda, umnoži te pakete i pošalje ih svim svojim susedima (osim čvora od kojeg je primio odgovarajući paket). U sledećem koraku, susedni čvorovi mogu da ih prime, umnože, proslede te pakete itd. Pretpostavimo da se u takvoj mreži za difuzno rutiranje koristi nekontrolisano plavljenje. Koliko će primeraka difuzno poslatih paketa biti preneto u koraku t, pod pretpostavkom da je u koraku 1 prenet samo jedan difuzni paket iz izvornog čvora do njegova tri suseda.

P53. Videli smo u odeljku 4.7 da ne postoji protokol mrežnog sloja koji bi se mogao upotrebiti za prepoznavanje računara koji učestvuju u višeznačnoj grupi. Polazeći od toga, kako aplikacije koje koriste višeznačno rutiranje mogu da saznaju koji računari učestvuju u višeznačnoj grupi?

P54. Projektujte (napravite opis u obliku pseudokoda) protokol na aplikativnom nivou koji održava adrese računara za sve računare koji učestvuju u višeznačnoj grupi. Pojedinačno navedite mrežnu uslugu (jednoznačnu ili višeznačnu) koju vaš protokol koristi i navedite da li vaš protokol šalje poruke unutar ili izvan opsega (u odnosu na tok podataka aplikacije između učesnika višeznačne grupe) i zašto je tako.

P55. Koliki je višeznačni adresni prostor? Pretpostavimo da dve različite višeznačne grupe biraju višeznačne adrese na slučajan način. Kolika je verovatnoća da će izabrati istu adresu? Pretpostavite sada da 1 000 grupa za višeznačno upućivanje istovremeno bira adresu grupe na slučajan način. Kolika je verovatnoća da dođe do preklapanja?

Zadaci iz programiranja soketa

Na kraju poglavlja 2, bila su četiri zadatka iz programiranja soketa. Ispod ćete naći peti zadatak koji upošljava protokol ICMP o kome smo govorili u ovom poglavlju.

Zadatak 5: Aplikacija Ping protokola ICMP

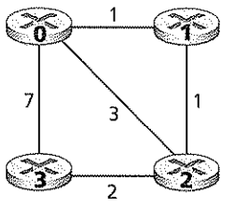
Ping je popularna mrežna aplikacija koja se koristi za testiranje sa udaljenje lekcije da li je određeni računar uključen i dostupan. Često se koristi i za merenje kašnjenja između računara klijenta i odredišnog računara. Funkcioniše tako što protokol ICMP šalje pakete „eho zahtev" (tj. ping pakete) do odredišnog računara i osluškuje „eho odgovor" protokola ICMP (tj. pong pakete). Ping meri RTT, beleži gubitke paketâ i izračunava statistički zbir ping-pong razmena (minimum, prosek, maksimum i standardnu devijaciju vremena kružnog puta).

U ovoj laboratorijskoj vežbi napisaćete sopstvenu Ping aplikaciju u programu *Python*. Vaša aplikacija će koristiti protokol ICMP. Međutim, kako bi program bio jednostavan, neće u potpunosti slediti zvaničnu specifikaciju iz dokumenta RFC 1739. Obratite pažnju da bi trebalo samo da napišete klijentsku stranu programa, jer su potrebne karakteristike serverske strane ugrađene u govoto sve operativne sisteme. Detaljan prikaz ovog zadatka kao i važne delove koda programa *Python* naće ćete na veb adresi http://www.awl.com/kurose-ross.

Programerski zadatak

U programerskom zadatku, napisaćete „distribuirani" skup precedura koje implementiraju distribuirano asinhorno rutiranje vektora rastojanja za mrežu prikazanu ispod.

Trebalo bi da napišete sledeće procedure koje će se „izvršavati" asinhrono unutar simuliranog okruženja koje je obezbeđeno za ovaj zadatak. Za čvor 0, napisaćete sledeće procedure:



- *rtinit0()*. Ova procedura će se pozvati jednom na početku simulacije. *rtinit0()* nema argumente. Ona bi trebalo da inicijalizuje tabelu rastojanja u čvoru 0, kako bi se predstavili direktni troškovi 1, 3 i 7 do čvorova 1, 2 i 3 redom. Na slici iznad svi linkovi su dvosmerni, a troškovi su identični u oba pravca. Posle inicijalizacije tabele rastojanja i drugih struktura podataka koje zahteva vaša procedura čvora 0, potrebno je zatim direktno poslati svojim susedima (u ovom slučaju 1, 2 i 3) troškove minimalne putanje čvora do ostalih čvorova mreže. Ova informacija o minimalnim troškovima se šalje do susednih čvorova u paketu za ažuriranje rutiranja pozivom procedure *tolayer 2 ()*, kao što je opisano u potpunom zadatku. Format paketa za ažuriranje rutiranja je takođe opisan u potpunom zadatku.
- *rtupdate0(struct rtpkt *rcvdpkt)*. Ova procedura će se pozvati kada čvor 0 primi paket rutiranja koji mu je poslao jedan od njegovih direktnih suseda. Parametar **rcvdpkt* je pokazivač na primljen paket. *rtupdate 0()* je „srce" algoritma vektora rastojanja. Vrednosti koje primi u paketu za ažuriranje rutiranja od nekog drugog čvora i sadrže trenutne troškove najkraće putanje čvora i do svih drugih čvorova mreže. *rtupdate 0()* koriste ove primljene vrednosti za ažuriranje sopstvene tabele rastojanja (kao što je odredio algoritam vektora rastojanja). Ako se njegovi minimalni troškovi do drugog čvora promene usled ažuriranja,