5

NIVELUL REȚEA

Nivelul rețea are ca sarcină preluarea pachetelor de la sursă și transferul lor către destinație. Ajungerea la destinație poate necesita mai multe salturi prin rutere intermediare de-a lungul drumului. Această funcție contrastează clar cu cea a nivelului legătură de date, care avea scopul mult mai modest de a transfera cadre de la un capăt al unui fir la celălalt. Astfel nivelul rețea este cel mai scăzut nivel care se ocupă de transmisii capăt la capăt.

Pentru realizarea scopurilor propuse, nivelul rețea trebuie să cunoască topologia subrețelei de comunicație (de exemplu mulțimea tuturor ruterelor) și să aleagă calea cea mai potrivită prin aceasta. De asemenea trebuie să aleagă căile de urmat astfel, încât să nu încarce excesiv unele legături de comunicație sau rutere în timp ce altele sunt inactive. În fine, când sursa și destinația fac parte din rețele diferite, apar probleme noi. Este sarcina nivelului rețea să se ocupe de ele. În acest capitol vom studia toate aceste aspecte și le vom exemplifica, in primul rând folosind Internetul și protocolul lui la nivelul rețea, IP, cu toate că vom vorbi și despre rețele fără fir.

5.1 CERINȚELE DE PROIECTARE ALE NIVELULUI REȚEA

Vom prezenta, în continuare, o introducere a cerințelor pe care proiectantul nivelului rețea trebuie să le rezolve. Acestea includ serviciile furnizate nivelului transport și proiectarea internă a subrețelei.

5.1.1 Comutare de pachete de tip Memorează-și-Retransmite (Store-and-Forward)

Dar înainte de a începe explicarea detaliilor nivelului rețea, merită probabil sa reinițializăm contextul în care operează protocoalele de la nivelul rețea. Acest context este prezentat în fig. 5-1. Componentele majore ale sistemului sunt echipamentul companiei de telecomunicații (rutere conectate prin linii de transmisie), prezentat în interiorul ovalului umbrit, și echipamentul clientului, prezentat în afara ovalului. Gazda *H1* este conectată direct la unul dintre ruterele companiei de telecomunicații, *A*, printr-o linie închiriată. În contrast, *H2* este într-o rețea LAN cu un ruter, *F*, deținut și operat de către client. Acest ruter are, deasemeni, și o linie închiriată către echipamentul companiei de telecomunicații. Am prezentat *F* ca fiind în afara ovalului, deoarece nu aparține companiei de telecomunicații, dar în termeni de construcție, software și protocoale, probabil că nu diferă față de ruterele acesteia. Este discutabil dacă aparține subrețelei, dar în contextul acestui capitol ruterele din localul clientului sunt considerate parte a subrețelei deoarece rulează aceeași algoritmi ca și ruterele companiei de telecomunicații (și aici principala noastră preocupare sunt algoritmii).

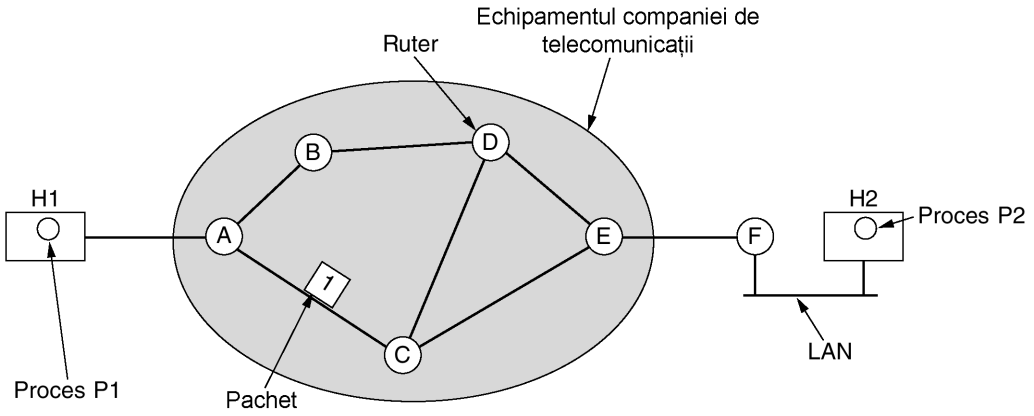


Fig. 5-1. Cadrul protocoalelor nivelului rețea.

Acest echipament este folosit după cum urmează. O gazdă care are de transmis un pachet îl transmite celui mai apropiat ruter, fie in aceeași rețea LAN, fie printr-o legătură punct la punct cu compania de telecomunicații. Pachetul este memorat acolo până ajunge integral, astfel încât să poată fi verificată suma de control. Apoi este trimis mai departe către următorul ruter de pe traseu, până ajunge la gazda destinație, unde este livrat. Acest mecanism reprezintă comutarea de pachete de tip memorează-și-retransmite, așa cum am văzut în capitolele anterioare.

5.1.2 Servicii furnizate nivelului transport

Nivelul rețea furnizează servicii nivelului transport la interfața dintre cele două niveluri. O întrebare importantă este ce fel de servicii furnizează nivelul rețea nivelului transport. Serviciile nivelului rețea au fost proiectate având în vedere următoarele scopuri:

1. Serviciile trebuie să fie independente de tehnologia ruterului.
2. Nivelul transport trebuie să fie independent de numărul, tipul și topologia ruterelor existente.

3. Adresele de rețea disponibile la nivelul transport trebuie să folosească o schemă de numerotare uniformă, chiar în cadrul rețelelor LAN și WAN.

Obiectivele fiind stabilite, proiectantul nivelului rețea are o mare libertate în a scrie specificațiile detaliate ale serviciilor oferite nivelului transport. Această libertate degenerează adesea într-o aprigă bătălie între două tabere opuse. Problema centrală a discuției este dacă nivelul rețea trebuie să furnizeze servicii orientate pe conexiune sau servicii neorientate pe conexiune.

O tabără (reprezentată de comunitatea Internet) afirmă că scopul ruterului este de a transfera pachete și nimic mai mult. În viziunea lor (bazată pe experiența a aproape 30 de ani de exploatare a unei rețele de calculatoare în funcțiune), subrețeaua este inerent nesigură, indiferent cum ar fi proiectată. De aceea calculatoarele gazdă trebuie să accepte faptul că rețeaua este nesigură și să facă controlul erorilor (i.e., detecția și corecția erorii) și controlul fluxului ele însele.

Acest punct de vedere duce rapid la concluzia că serviciul rețea trebuie să fie neorientat pe conexiune, cu două primitive SEND PACKET și RECEIVE PACKET și cu foarte puțin în plus. În particular, nu trebuie făcută nici o operație pentru controlul ordinii sau fluxului pachetelor pentru că oricum calculatorul gazdă va face acest lucru, și, de obicei, dublarea acestor operații aduce un câștig nesemnificativ. În continuare, fiecare pachet va trebui să poarte întreaga adresă de destinație, pentru că fiecare pachet este independent de pachetele predecesoare, dacă acestea există.

Cealaltă tabără (reprezentată de companiile de telefoane) afirmă că subrețeaua trebuie să asigure un serviciu orientat pe conexiune sigur. Ei susțin că 100 de ani de experiență cu sistemul telefonic mondial reprezintă un ghid excelent. În această perspectivă, calitatea serviciului este elementul dominant, și într-o subrețea fără conexiuni, calitatea serviciului este dificil de obținut, în special pentru trafic in timp real cum ar fi voce și imagine.

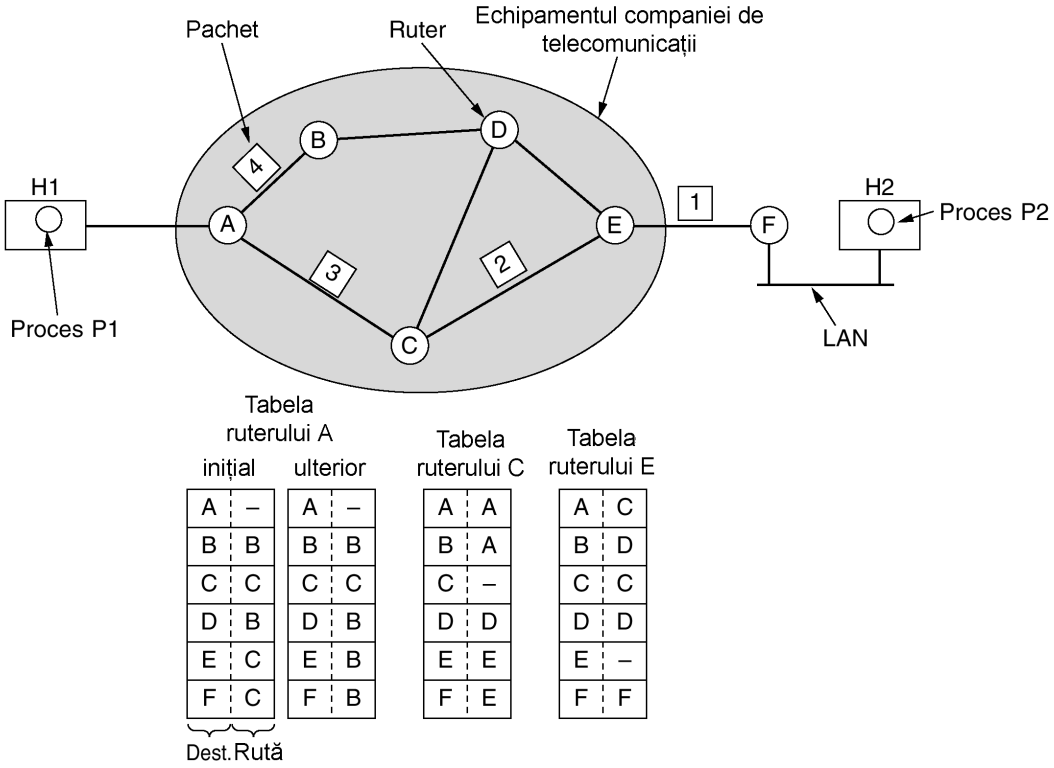
Aceste două tabere sunt cel mai bine exemplificate de Internet și rețele ATM. Rețeaua Internet oferă un serviciu la nivelul rețea neorientat pe conexiune; rețelele ATM oferă un serviciu la nivelul rețea orientat pe conexiune. Totuși, este interesant de notat că cu cât garantarea calității serviciului devine din ce in ce mai importantă, Internetul evoluează. În particular, începe să dobândească proprietăți asociate normal cu serviciile orientate conexiune, așa cum vom vedea mai târziu. De fapt, ne-am făcut o părere despre această evoluție in timpul studiului despre rețele VLAN în Cap. 4.

5.1.3 Implementarea serviciului neorientat pe conexiune

După ce am văzut cele două clase de servicii pe care nivelul rețea le furnizează utilizatorilor săi, este momentul să vedem funcționarea internă a acestui nivel. Sunt posibile două organizări diferite, în funcție de tipul serviciului oferit. Dacă este oferit un serviciu neorientat pe conexiune, atunci pachetele sunt trimise in subrețea individual și dirijate independent de celelalte. Nu este necesară nici o inițializare prealabilă. În acest context, pachetele sunt numite frecvent **datagrame** (datagrams) (prin analogie cu telegramele), iar subrețeaua este numită **subrețea datagramă** (datagram subnet). Dacă este folosit serviciul orientat conexiune, atunci, înainte de a trimite pachete de date, trebuie stabilită o cale de la ruterul sursă la ruterul destinație. Această conexiune este numită **VC** (**virtual circuit**, circuit virtual), prin analogie cu circuitele fizice care se stabilesc în sistemul telefonic, iar subrețeaua este numită **subrețea cu circuite virtuale** (**virtual-circuit subnet**). În această secțiune vom studia subrețele datagramă; în următoarea secțiune vom studia subrețelele cu circuite virtuale.

Să vedem cum funcționează o subrețea datagramă. Să presupunem că procesul *P1* din fig. 5-2 are un mesaj lung pentru procesul *P2*. El transmite mesajul nivelului transport, cu instrucțiunile de livrare către procesul *P2* aflat pe calculatorul gazdă *H2*. Codul nivelului transport rulează pe calculatorul

gazdă *H1*, de obicei în cadrul sistemului de operare. Acesta inserează la începutul mesajului un antet corespunzător nivelului transport și transferă rezultatul nivelului rețea, probabil o altă procedură din cadrul sistemului de operare.



A	–
B	B
C	C
D	B
E	C
F	C

A	–
B	B
C	C
D	B
E	B
F	B

A	A
B	A
C	–
D	D
E	E
F	E

A	C
B	D
C	C
D	D
E	–
F	F

Fig. 5-2. Dirijarea într-o subrețea datagramă.

Să presupunem că mesajul este de patru ori mai lung decât dimensiunea maximă a unui pachet, așa că nivelul rețea trebuie să îl spargă în patru pachete, 1, 2, 3, și 4 și să le trimită pe fiecare în parte ruterului *A*, folosind un protocol punct-la-punct, de exemplu, PPP. Din acest punct controlul este preluat de compania de telecomunicații. Fiecare ruter are o tabelă internă care îi spune unde să trimită pachete pentru fiecare destinație posibilă. Fiecare intrare în tabelă este o pereche compusă din destinație și linia de ieșire folosită pentru acea destinație. Pot fi folosite doar linii conectate direct. De exemplu, în fig. 5-2, *A* are doar două linii de ieșire – către *B* și *C* – astfel că fiecare pachet ce vine trebuie trimis către unul dintre aceste rutere, chiar dacă ultima destinație este alt ruter. Tabela de rutare inițială a lui *A* este prezentată în figură sub eticheta „inițial”.

Cum au ajuns la *A*, pachetele 1, 2 și 3 au fost memorate pentru scurt timp (pentru verificarea sumei de control). Apoi fiecare a fost trimis mai departe către *C* conform tabelei lui *A*. Pachetul 1 a fost apoi trimis mai departe către *E* și apoi către *F*. Când a ajuns la *F*, a fost încapsulat într-un cadru al nivelului legătură de date și trimis către calculatorul gazdă *H2* prin rețeaua LAN.

Totuși, ceva diferit s-a întâmplat cu pachetul 4. Când a ajuns la *A* a fost trimis către ruterul *B,* chiar dacă și el este destinat tot lui *F*. Dintr-un motiv oarecare, *A* a decis să trimită pachetul 4 pe o rută diferită de cea urmată de primele trei. Poate că a aflat despre o congestie undeva pe calea *ACE*

și și-a actualizat tabela de rutare, așa cum apare sub eticheta „mai târziu”. Algoritmul ce administrează tabelele și ia deciziile de rutare se numește **algoritm de rutare** (routing algorithm). Algoritmii de rutare sunt unele dintre principalele elemente pe care le vom studia în acest capitol.

5.1.4 Implementarea serviciilor orientate pe conexiune

Pentru serviciile orientate conexiune, avem nevoie de o subrețea cu circuite virtuale. Să vedem cum funcționează aceasta. Ideea care se stă la baza circuitelor virtuale este evitarea alegerii unei noi căi (rute) pentru fiecare pachet trimis, ca în fig. 5-2. În schimb, atunci când se stabilește o conexiune, se alege o cale între mașina sursă și mașina destinație, ca parte componentă a inițializării conexiunii și aceasta este memorată în tabelele ruterelor. Acea cale este folosită pentru tot traficul de pe conexiune, exact în același mod în care funcționează sistemul telefonic. Atunci când conexiunea este eliberată, este închis și circuitul virtual. În cazul serviciilor orientate conexiune, fiecare pachet poartă un identificator care spune cărui circuit virtual îi aparține.

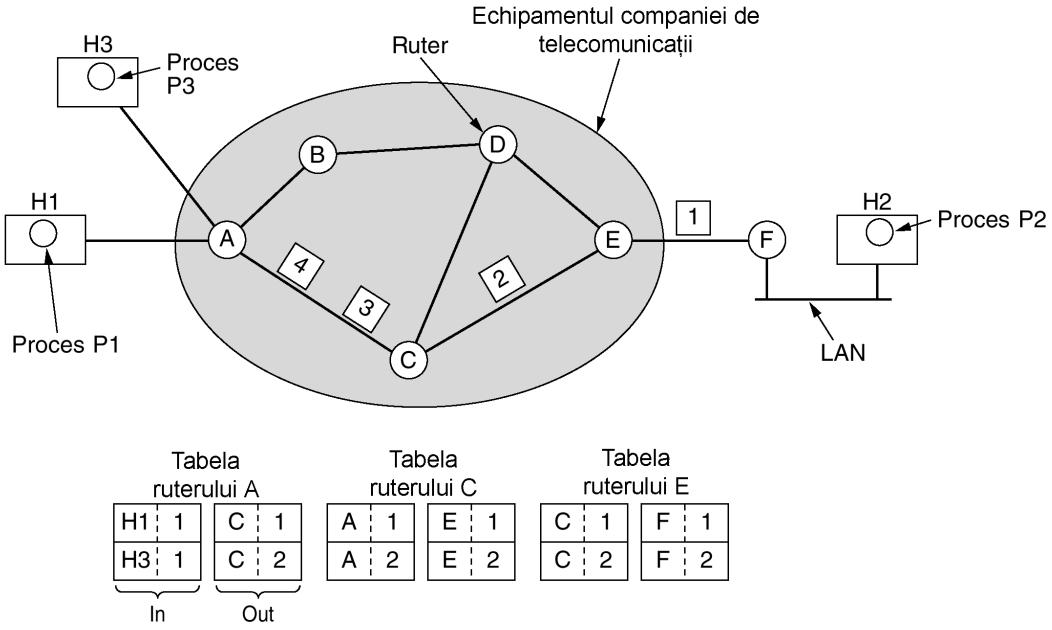


Fig. 5-3. Dirijare în cadrul unei subrețele cu circuite virtuale.

De exemplu, să considerăm situația din fig. 5-3. Aici calculatorul gazdă *H1* a stabilit conexiunea 1 cu calculatorul gazdă *H2*. Aceasta este memorată ca prima intrare în fiecare tabelă de rutare. Prima linie a tabelei lui *A* spune că dacă un pachet purtând identificatorul de conexiune 1 vine de la *H1*, atunci trebuie trimis către ruterul *C*, dându-i-se identificatorul de conexiune 1. Similar, prima intrare a lui C dirijează pachetul către *E*, tot cu identificatorul de conexiune 1.

Acum să vedem ce se întâmplă dacă *H3* vrea, de asemenea, să stabilească o conexiune cu *H2*. Alege identificatorul de conexiune 1 (deoarece inițializează conexiunea și aceasta este singura conexiune) și indică subrețelei să stabilească circuitul virtual. Aceasta conduce la a doua linie din tabele. Observați că apare un conflict deoarece deși *A* poate distinge ușor pachetele conexiunii 1 de la *H1*

de pachetele conexiunii 1 de la *H3*, *C* nu poate face asta. Din acest motiv, *A* asociază un identificator de conexiune diferit pentru traficul de ieșire al celei de a doua conexiuni. Pentru evitarea conflictelor de acest gen ruterele trebuie să poată înlocui identificatorii de conexiune în pachetele care pleacă. În unele contexte, aceasta se numește comutarea etichetelor (label switching).

5.1.5 Comparație între subrețele cu circuite virtuale și subrețele datagramă

Atât circuitele virtuale cât și datagramele au suporteri și oponenți. Vom încerca acum să rezumăm argumentele ambelor tabere. Principalele aspecte sunt prezentate în fig. 5-4, deși cei extrem de riguroși ar putea probabil găsi un contraexemplu pentru toate cele descrise în această figură.

Problemă	Subrețea datagramă	Subrețea cu circuite virtuale (CV)
Stabilirea circuitului	Nu este necesară	Obligatorie
Adresare	Fiecare pachet conține adresa completă pentru sursă și destinație	Fiecare pachet conține un număr mic de CV
Informații de stare	Ruterele nu păstrează informații despre conexiuni	Fiecare CV necesită spațiu pentru tabela ruterului per conexiune
Dirijare	Fiecare pachet este dirijat independent	Calea este stabilită la inițierea CV; toate pachetele o urmează
Efectul defectării ruterului	Nici unul, cu excepția pachetelor pierdute în timpul defectării	Toate circuitele virtuale care trec prin ruterul defect sunt terminate
Calitatea serviciului	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse
Controlul congestiei	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse

Fig. 5-4. Comparație între subrețele datagramă și subrețele cu circuite virtuale.

În interiorul subrețelei există situații în care trebuie să se aleagă între facilități antagoniste specifice fie circuitelor virtuale, fie datagramelor. Un astfel de compromis este acela între spațiul de memorie al ruterului și lățimea de bandă. Circuitele virtuale permit pachetelor să conțină numere de circuite în locul unor adrese complete. Dacă pachetul tinde să fie foarte mic, atunci existența unei adrese complete în fiecare pachet poate reprezenta o supraîncărcare (overhead) importantă și deci o irosire a lățimii de bandă. Prețul plătit pentru folosirea internă a circuitelor virtuale este spațiul necesar păstrării tabelei în ruter. Soluția mai ieftină este determinată de raportul între costul circuitelor de comunicație și cel al memoriei ruterului.

Alt compromis este cel între timpul necesar stabilirii circuitului și timpul de analiză a adresei. Folosirea circuitelor virtuale presupune existența unei faze inițiale de stabilire a căii, care cere timp și consumă resurse. Oricum, este ușor să ne imaginăm ce se întâmplă cu un pachet de date într-o subrețea bazată pe circuite virtuale: ruterul folosește numărul circuitului ca un index într-o tabelă pentru a afla unde merge pachetul. Într-o rețea bazată pe datagrame, pentru a găsi intrarea corespunzătoare destinației se folosește o procedură de căutare mult mai complicată.

O altă problemă este cea a dimensiunii spațiului necesar pentru tabela din memoria ruterului. O subrețea datagramă necesită o intrare pentru fiecare destinație posibilă, în timp ce o rețea cu circuite virtuale necesită o intrare pentru fiecare circuit virtual. Totuși, acest avantaj este relativ iluzoriu deoarece și pachetele de inițializare a conexiunii trebuie rutate, iar ele folosesc adresele destinație, la fel ca și datagramele.

Circuitele virtuale au unele avantaje în garantarea calității serviciului și evitarea congestionării subrețelei, deoarece resursele (de exemplu zone tampon, lărgime de bandă și cicluri CPU) pot fi rezervate în avans, atunci când se stabilește conexiunea. La sosirea pachetelor, lățimea de bandă necesară și capacitatea ruterului vor fi deja pregătite. Pentru o subrețea bazată pe datagrame, evitarea congestionării este mult mai dificilă.

Pentru sistemele de prelucrare a tranzacțiilor (de exemplu apelurile magazinelor pentru a verifica cumpărături realizate cu cărți de credit) overhead-ul implicat de stabilirea și eliberarea unui circuit virtual poate reduce cu ușurință utilitatea circuitului. Dacă majoritatea traficului este de acest tip, folosirea internă a circuitelor virtuale în cadrul subrețelei nu prea are sens. Pe de altă parte, ar putea fi de folos circuite virtuale permanente, stabilite manual și care să dureze luni sau chiar ani.

Circuitele virtuale au o problemă de vulnerabilitate. Dacă un ruter se defectează și își pierde conținutul memoriei, atunci toate circuitele virtuale care treceau prin el sunt suprimate, chiar dacă acesta își revine după o secundă. Prin contrast, dacă se defectează un ruter bazat pe datagrame vor fi afectați doar acei utilizatori care aveau pachete memorate temporar în cozile de așteptare ale ruterului și este posibil ca numărul lor să fie și mai mic, în funcție de câte pachete au fost deja confirmate. Pierderea liniei de comunicație este fatală pentru circuitele virtuale care o folosesc, însă poate fi ușor compensată dacă se folosesc datagrame. De asemenea, datagramele permit ruterului să echilibreze traficul prin subrețea, deoarece căile pot fi modificate parțial în cursul unei secvențe lungi de pachete transmise.

5.2 ALGORITMI DE DIRIJARE

Principala funcție a nivelului rețea este dirijarea pachetelor de la mașina sursă către mașina destinație. În majoritatea subrețelelor pachetele vor face salturi multiple pentru a ajunge la destinație. Singura excepție remarcabilă o reprezintă rețelele cu difuzare, dar chiar și aici dirijarea este importantă, atunci când sursa și destinația nu sunt în aceeași rețea. Algoritmii care aleg calea și structurile de date folosite de aceștia reprezintă un domeniu important al proiectării nivelului rețea.

Algoritmul de dirijare (routing algorithm) este acea parte a software-ului nivelului rețea care răspunde de alegerea liniei de ieșire pe care trebuie trimis un pachet recepționat. Dacă subrețeaua folosește intern datagrame, această decizie trebuie luată din nou pentru fiecare pachet recepționat, deoarece este posibil ca cea mai bună rută să se fi modificat între timp. Dacă subrețeaua folosește circuite virtuale, deciziile de dirijare sunt luate doar la inițializarea unui nou circuit virtual. După aceea pachetele de date vor urma doar calea stabilită anterior. Acest ultim caz este numit uneori **dirijare de sesiune** (**session routing**), deoarece calea rămâne în funcțiune pentru o întreagă sesiune utilizator (de exemplu o sesiune de conectare de la un terminal -login- sau un transfer de fișiere).

Uneori este util să se facă distincția între dirijare, care înseamnă alegerea căii care va fi folosită, și retransmitere, care se referă la ceea ce se întâmplă atunci când sosește un pachet. Se poate spune despre un ruter că rulează intern două procese. Unul dintre ele preia fiecare pachet care sosește, căutând în tabela de dirijare linia de ieșire folosită pentru el. Acesta este procesul de **retransmitere** (**forwarding**). Celălalt proces se ocupă de completarea și actualizarea tabelei de rutare. Aici algoritmul intervine de dirijare.

Indiferent dacă ruta se alege independent pentru fiecare pachet sau doar la stabilirea unei noi conexiuni, un algoritm de dirijare trebuie să aibă anumite proprietăți: corectitudine, simplitate, robustețe, stabilitate, echitate, optimalitate. Corectitudinea și simplitatea nu mai au nevoie de comentarii, dar necesitatea robusteții poate fi mai puțin evidentă la prima vedere. Odată ce apare pe piață o rețea importantă, este de așteptat ca ea să funcționeze continuu ani întregi, fără defecte generale ale sistemului. În acest timp vor exista defecte hardware și software de tot felul. Calculatoare gazdă, rutere, linii de comunicație vor cădea repetat și topologia se va schimba de multe ori. Algoritmul de dirijare trebuie să facă față acestor modificări ale topologiei și traficului, fără a impune ca toate joburile de pe toate calculatoarele să fie abandonate și rețeaua să fie reinițializată de fiecare dată când se defectează un ruter.

Stabilitatea este de asemenea un obiectiv important pentru algoritmul de dirijare. Există algoritmi de dirijare care niciodată nu converg la echilibru, indiferent cât timp ar rula. Un algoritm stabil atinge starea de echilibru și o menține. Echitatea și optimalitatea sunt evidente – este sigur că nici o persoană înțeleaptă nu li se opune - însă, așa cum se va arăta, adeseori acestea sunt obiective contradictorii. Un exemplu simplu al acestui conflict este prezentat în fig. 5-5. Presupunem că între A și A', între B și B' și între C și C' există un trafic suficient pentru a satura legăturile orizontale. Pentru a maximiza fluxul total, traficul între X și X' trebuie oprit. Din păcate acest lucru ar defavoriza pe X și X'. Evident, este necesar un compromis între eficiența globală și echitatea față de fiecare dintre conexiuni.

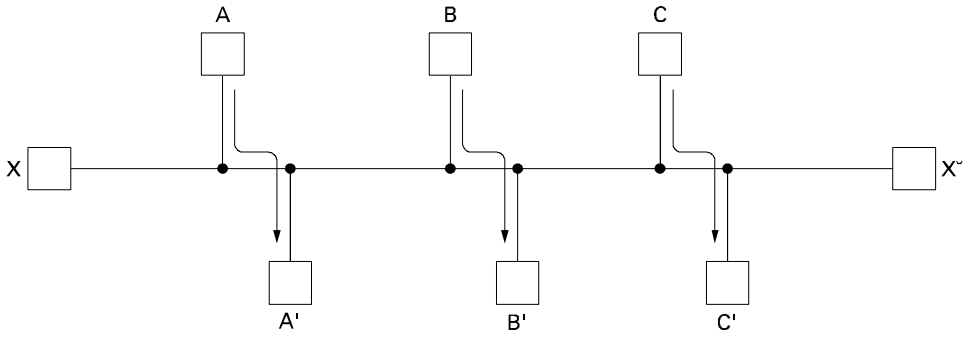


Fig. 5-5. Conflict între echitate și optimalitate.

Înainte de a încerca să găsim rezolvarea acestui conflict între optimalitate și prevenirea defavorizării, trebuie să stabilim ce vrem să optimizăm. Minimizarea întârzierii medii a unui pachet este un candidat evident, însă la fel este și maximizarea productivității (throughput) totale a rețelei. Mai mult, și aceste două obiective sunt în conflict, deoarece funcționarea unui sistem cu cozi de așteptare la limita capacității sale produce întârzieri majore. Pentru a realiza un compromis, în multe rețele se încearcă minimizarea numărului de salturi pe care trebuie să le facă un pachet, deoarece reducerea numărului de salturi tinde să îmbunătățească întârzierea și de asemenea să reducă lățimea de bandă consumată, ceea ce tinde să îmbunătățească și productivitatea.

Algoritmii de dirijare pot fi grupați în două mari clase: neadaptivi și adaptivi. **Algoritmii neadaptivi** (**nonadaptive algorithms**) nu își bazează deciziile de dirijare pe măsurători sau estimări ale traficului și topologiei curente. Astfel, alegerea căii folosite pentru a ajunge de la nodul *I* la nodul *J* (oricare ar fi *I* și *J*) se calculează în avans, off-line și parvine ruterului la inițializarea rețelei. Această procedură se mai numește și **dirijare statică** (**static routing**).

Algoritmii adaptivi (**adaptive algorithms**), prin contrast, își modifică deciziile de dirijare pentru a reflecta modificările de topologie și de multe ori și pe cele de trafic. Algoritmii adaptivi diferă prin locul de unde își iau informația (de exemplu local, de la un ruter vecin sau de la toate ruterele), prin momentul la care schimbă rutele (de exemplu la fiecare ΔT secunde, când se schimbă încărcarea sau când se schimbă topologia) și prin metrica folosită pentru optimizare (de exemplu distanța, numărul de salturi sau timpul estimat pentru tranzit). În secțiunile următoare vom discuta o varietate de algoritmi de dirijare, atât statici cât și dinamici.

5.2.1 Principiul optimalității

Înainte de a intra în algoritmii specifici, ar fi poate folositor să observăm că se poate face o afirmație despre rutele optimale fără a ne referi la topologia rețelei sau la trafic. Această afirmație este cunoscută sub numele de **principiul optimalității** (**optimality principle**). El stabilește că dacă ruterul *J* este pe calea optimă de la ruterul *I* către ruterul *K*, atunci calea optimă de la *J* la *K* este pe aceeași rută. Pentru a vedea aceasta, să notăm cu r_1 partea din cale de la *I* la *J*, iar cu r_2 restul rutei. Dacă ar exista o rută mai bună decât r_2 de la *J* la *K*, ea ar putea fi concatenată cu r_1 și ar îmbunătăți ruta de la *I* la *K*, ceea ce ar contrazice presupunerea că r_1r_2 este optimală.

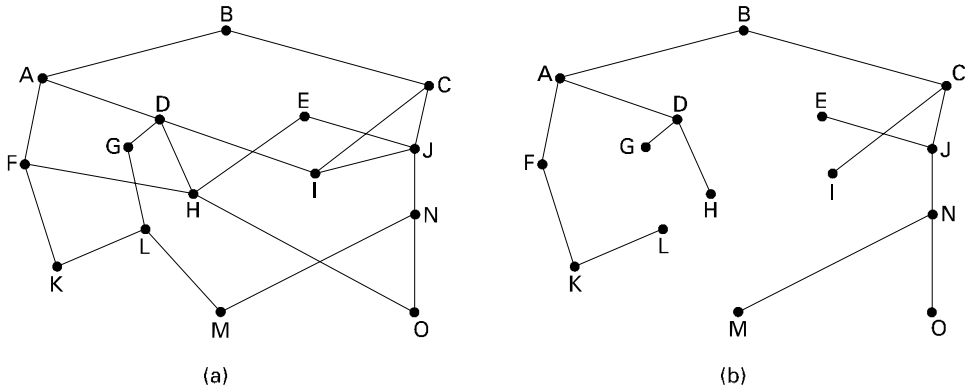


Fig. 5-6. (a) O subrețea. (b) Un arbore de scufundare pentru ruterul *B*.

Ca o consecință directă a principiului optimalității, putem observa că mulțimea rutelor de la toate sursele către o anume destinație formează un arbore având rădăcina în destinație. Acest arbore se numește **arbore de scufundare** (**sink tree**) și este prezentat în fig. 5-6, unde distanța metrica aleasă este numărul de salturi. Observați că arborele de scufundare nu este unic, putând exista și alți arbori cu aceeași lungime a căii. Scopul tuturor algoritmilor de dirijare este de a descoperi și folosi arborii de scufundare pentru toate ruterele.

Deoarece arborele de scufundare este într-adevăr un arbore, el nu conține bucle, deci fiecare pachet va fi livrat într-un număr finit și limitat de salturi. În practică viața nu este chiar așa de ușoară. Legăturile și ruterele pot să se defecteze și să-și revină în timpul operațiilor, astfel încât diferite rutere pot avea imagini diferite asupra topologiei curente. De asemenea, am trecut mai repede peste întrebarea dacă fiecare ruter trebuie să obțină individual informația necesară calculării arborelui de scufundare sau dacă această informație este colectată prin alte mijloace. Vom reveni însă la această

problemă în curând. Cu toate acestea, principiul optimalității și arborele de scufundare furnizează referințe cu care pot fi comparați ceilalți algoritmi de dirijare.

5.2.2 Dirijarea pe calea cea mai scurtă

Să începem studiul algoritmilor de dirijare posibili cu o tehnică des utilizată în multe forme deoarece este simplă și ușor de înțeles. Ideea este de a construi un graf al subrețelei, fiecare nod al grafului fiind un ruter, iar fiecare arc al grafului fiind o linie de comunicație (numită adesea legătură). Pentru a alege o cale între o pereche dată de rutere, algoritmul trebuie să găsească în graf calea cea mai scurtă dintre ele.

Conceptul de **cea mai scurtă cale (shortest path routing)** necesită unele explicații. O modalitate de a măsura lungimea căii este numărul de salturi. Folosind această metrică, căile *ABC* și *ABE* din fig. 5-7 sunt la fel de lungi. O altă metrică este distanța geografică în kilometri, caz în care *ABC* este clar mult mai mare decât *ABE* (presupunând că fig. este desenată la scară).

Oricum, sunt posibile multe alte metrici în afară de salturi și distanța geografică. De exemplu, fiecare arc poate fi etichetat cu valorile medii ale așteptării în coadă și întârzierii de transmisie pentru anumite pachete standard de test, așa cum sunt determinate de măsurători care se fac din oră în oră. Cu această etichetare, cea mai scurtă cale este cea mai rapidă, nu neapărat cea cu mai puține arce sau kilometri.

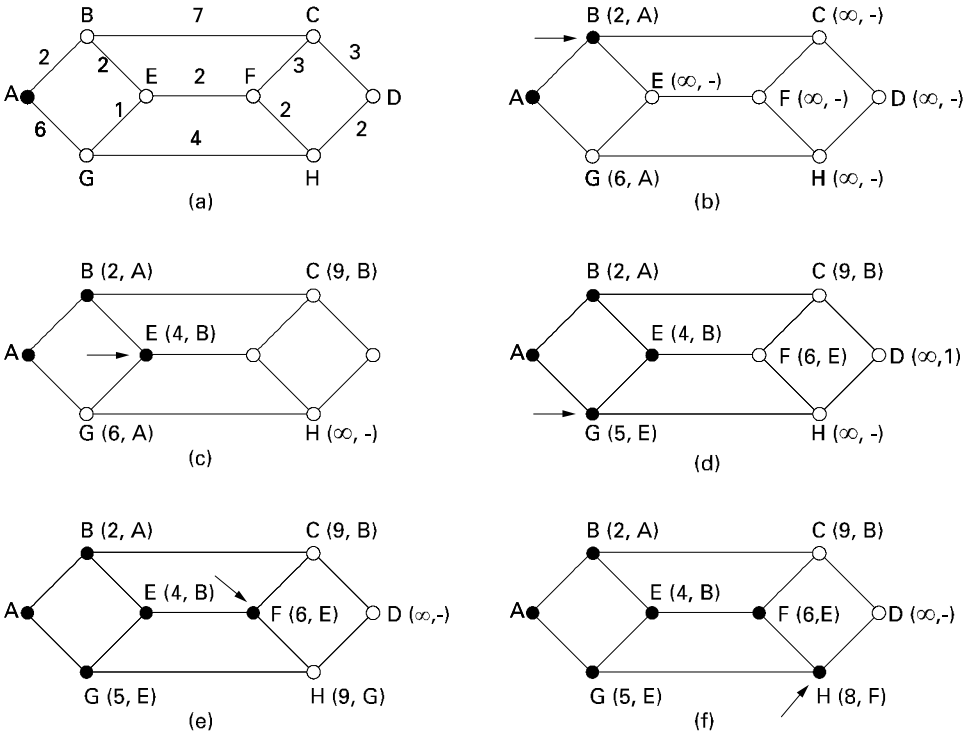


Fig. 5-7. Primii cinci pași folosiți în calcularea celei mai scurte căi de la A la D. Săgețile indică nodul curent.

În cazul cel mai general, etichetele de pe arce ar putea fi calculate ca funcții de distanță, lărgime de bandă, trafic mediu, cost al comunicației, lungime medie a cozilor de așteptare, întârzieri măsurate și alți factori. Prin modificarea ponderilor, algoritmul ar putea calcula cea mai „scurtă" cale, în conformitate cu oricare dintre aceste criterii sau cu combinații ale acestor criterii.

Se cunosc mai mulți algoritmi pentru calculul celei mai scurte căi între două noduri dintr-un graf. Cel mai cunoscut este cel propus de Dijkstra (1959). Fiecare nod este etichetat (în paranteze) cu distanța de la nodul sursă până la el, de-a lungul celei mai bune căi cunoscute. Inițial nu se cunoaște nici o cale, așa că toate nodurile vor fi etichetate cu infinit. Pe măsură ce se execută algoritmul și se găsesc noi căi, etichetele se pot schimba, reflectând căi mai bune. O etichetă poate fi fie temporară, fie permanentă. Inițial toate etichetele sunt temporare. Atunci când se descoperă că o etichetă reprezintă cea mai scurtă cale posibilă de la sursă către acel nod, ea devine permanentă și nu se mai schimbă ulterior.

Pentru a ilustra cum funcționează algoritmul de etichetare, să ne uităm la graful neorientat, etichetat din fig. 5-7(a), unde etichetele reprezintă, de exemplu, distanța. Dorim să aflăm cea mai scurtă cale de la *A* la *D*. Începem prin a marca nodul *A* ca permanent, indicând aceasta printr-un cerc colorat. Apoi vom examina fiecare nod adiacent cu *A* (care este acum nodul curent), reetichetând fiecare nod cu distanța până la nodul *A*. De fiecare dată când un nod este reetichetat, îl vom eticheta și cu nodul de la care s-a făcut încercarea, pentru a putea reface calea ulterior. După ce am examinat toate nodurile adiacente ale lui *A*, vom examina toate nodurile cu etichetă temporară din întregul graf și îl facem permanent pe cel cu eticheta minimă, așa cum se observă din fig. 5-7(b). Acest nod devine noul nod curent.

Acum începem din *B* și examinăm toate nodurile sale adiacente. Dacă suma între eticheta lui *B* și distanța de la *B* la nodul considerat este mai mică decât eticheta acelui nod, înseamnă că am găsit o cale mai scurtă și va trebui făcută reetichetarea nodului.

După ce toate nodurile adiacente nodului curent au fost inspectate și au fost schimbate toate etichetele temporare posibile, se reia căutarea în întregul graf pentru a identifica nodul cu eticheta temporară minimă. Acest nod este făcut permanent și devine nodul curent al etapei următoare. Fig. 5-7 prezintă primii cinci pași ai algoritmului.

Pentru a vedea de ce merge algoritmul, să privim fig. 5-7(c). La momentul respectiv de abia am făcut permanent nodul *E*. Să presupunem că ar exista o cale mai scurtă decât *ABE*, de exemplu *AXYZE*. Există două posibilități: fie nodul *Z* a fost deja făcut permanent, fie încă nu a fost. Dacă a fost, atunci *E* a fost deja examinat (la pasul imediat următor celui la care *Z* a fost făcut permanent), astfel încât calea *AXYZE* nu a fost ignorată și deci nu poate fi cea mai scurtă cale.

Să considerăm acum cazul în care *Z* este încă etichetă temporară. Atunci fie eticheta lui *Z* este mai mare sau egală cu cea a lui *E*, caz în care *ABE* nu poate fi o cale mai scurtă decât *AXYZE*, fie este mai mică decât cea a lui *E*, caz în care *Z* și nu *E* va deveni permanent mai întâi, permițând lui *E* să fie examinat din *Z*.

Algoritmul este prezentat în fig. 5-8. Variabilele globale *n* și *dist* sunt inițializate înainte să fie apelată *shortest_path*. Singura diferență între program și algoritmul descris mai sus este aceea că în fig. 5-8 calculăm calea cea mai scurtă pornind de la nodul terminal, *t*, în locul nodului sursă, *s*. Deoarece calea cea mai scurtă de la *t* la *s* într-un graf neorientat este exact aceeași cu calea cea mai scurtă de la *s* la *t*, nu contează la care capăt începem (decât dacă există mai multe căi scurte, caz în care, inversând calea, am putea găsi alt drum). Motivul pentru care începem căutarea de la nodul destinație este acela că nodurile sunt etichetate cu predecesorul și nu cu succesorul. Atunci când calea finală este copiată în variabila de ieșire, *path*, calea este inversată. Prin inversarea căutării, cele două efecte se anulează, astfel încât rezultatul se obține în ordinea corectă.

```
#define MAX_NODES 1024                                  /* numărul maxim de noduri */
#define INFINITY 1000000000                          /* un număr mai mare decât orice cale */
int n,dist[MAX_NODES][ MAX_NODES]                    /* dist[i][j] e distanța de la i la j */

void shortest_path(int s, int t, int path[])
{ struct state {                                          /* calea cu care se lucrează */
 int predecessor;                                                 /* nodul anterior */
 int length;                                          /* lungimea de la sursă la acest nod */
 enum {permanent, tentative} label;                               /* etichetă stare */
}state [MAX_NODES];

int i, k, min;
struct state *p;

for (p = &state[0]; p < &state[n]; p++) {                          /* inițializări */
 p->predecessor = -1;
 p->length = INFINITY;
 p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;                                                    /* k este nodul inițial de lucru */
do {                                                   /* există vreo cale mai bună de la k? */
 for (i = 0; i < n; i++)                                          /* graful are n noduri */
   if (dist[k][i] != 0 && state[i].label == tentative) {
     if (state[k].length + dist[k][i] < state[i].length) {
       state[i].predecessor = k;
       state[i].length = state[k].length + dist[k][i];
     }
   }

 /* Găsește nodul etichetat temporar cu cea mai mică etichetă */
 k = 0; min = INFINITY;
 for (i = 0; i < n; i++)
   if (state[i].label == tentative && state[i].length < min) {
     min = state[i].length;
     k = i;
   }
 state[k].label = permanent;
} while (k != s);

/* Copiază calea în vectorul de ieșire */
i = 0; k = s;
do { path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Fig. 5-8. Algoritmul Dijkstra pentru calculul celei mai scurte căi într-un graf.

5.2.3 Inundarea

Un alt algoritm static este **inundarea (flooding)**, în care fiecare pachet recepționat este trimis mai departe pe fiecare linie de ieșire, cu excepția celei pe care a sosit. Este evident că inundarea generează un mare număr de pachete duplicate, de fapt un număr infinit dacă nu se iau unele măsuri pentru a limita acest proces. O astfel de măsură este păstrarea unui contor de salturi în antetul fiecă-

rui pachet, contor care este decrementat la fiecare salt și care face ca pachetul să fie distrus când contorul atinge valoarea zero.

Ideal ar fi ca acest contor să fie inițializat cu lungimea căii de la sursă la destinație. Dacă emițătorul nu cunoaște lungimea căii, poate inițializa contorul la valoarea cea mai defavorabilă, adică diametrul subrețelei.

O metodă alternativă pentru limitarea inundării este identificarea pachetelor care au fost deja inundate, pentru a preîntâmpina trimiterea lor a doua oară. O cale pentru a realiza acest scop este ca ruterul sursă să plaseze un număr de secvență în fiecare pachet pe care îl primește de la calculatorul gazdă asociat. Fiecare ruter necesită menținerea unei liste pentru fiecare ruter sursă, cu numerele de secvență provenite de la acel ruter sursă și care au fost deja trimise mai departe. Dacă sosește un pachet care se află în listă, el nu mai este trimis mai departe.

Pentru a limita creșterea lungimii listei, fiecare listă trebuie însoțită de un contor, *k*, care semnifică faptul că toate numerele de secvență până la *k* au fost deja tratate. La recepția unui pachet este ușor să se verifice dacă este un duplicat, caz în care este distrus. Evident, lista cu numere mai mici decât *k* nu este necesară, deoarece *k* o rezumă.

O variantă a algoritmului de inundare, care este și ceva mai practică, este **inundarea selectivă** (**selective flooding**). În acest algoritm ruterele nu trimit fiecare pachet recepționat pe fiecare legătură de ieșire, ci doar pe acele linii care duc aproximativ în direcția potrivită. De obicei sunt puține motive pentru a trimite un pachet spre partea de vest a rețelei folosind o legătură spre est, decât dacă topologia rețelei este cu totul deosebită și ruterul este sigur de acest lucru.

Inundarea nu este practică pentru majoritatea aplicațiilor, însă are destule utilizări. De exemplu, în aplicațiile militare, unde un mare număr de rutere pot fi scoase din funcționare în orice moment, robustețea extraordinară a inundării este necesară. În aplicațiile de baze de date distribuite, este uneori necesar ca toate bazele de date să fie actualizate simultan, caz în care inundarea poate fi folositoare. În rețelele fără fir, toate mesajele transmise de o gazdă pot fi recepționate de toate celelalte gazde din raza sa radio, ceea ce înseamnă de fapt inundare, și unii algoritmi folosesc această proprietate. O a patra utilizare posibilă a inundării este ca metrică la care să se raporteze toți ceilalți algoritmi de dirijare. Inundarea alege întotdeauna cea mai scurtă cale, deoarece alege în paralel toate căile posibile. În consecință, nici un alt algoritm nu poate produce o întârziere mai redusă (dacă ignorăm supraîncărcarea generată de însuși procesul de inundare).

5.2.4 Dirijare cu vectori distanță

Rețelele moderne de calculatoare folosesc de obicei algoritmi dinamici de dirijare în locul celor statici, descriși anterior, deoarece algoritmii statici nu țin seama de încărcarea curentă a rețelei. Doi dintre cei mai cunoscuți algoritmi dinamici sunt algoritmul de dirijare cu vectori distanță și algoritmul de dirijare bazat pe starea legăturilor. În această secțiune ne vom ocupa de primul algoritm. În secțiunea următoare vom studia cel de-al doilea algoritm.

Algoritmul de **dirijare cu vectori distanță (distance vector routing)** presupune că fiecare ruter menține o tabelă (de exemplu un vector) care păstrează cea mai bună distanță cunoscută spre fiecare destinație și linia care trebuie urmată pentru a ajunge acolo. Aceste tabele sunt actualizate prin schimbul de informații între nodurile vecine.

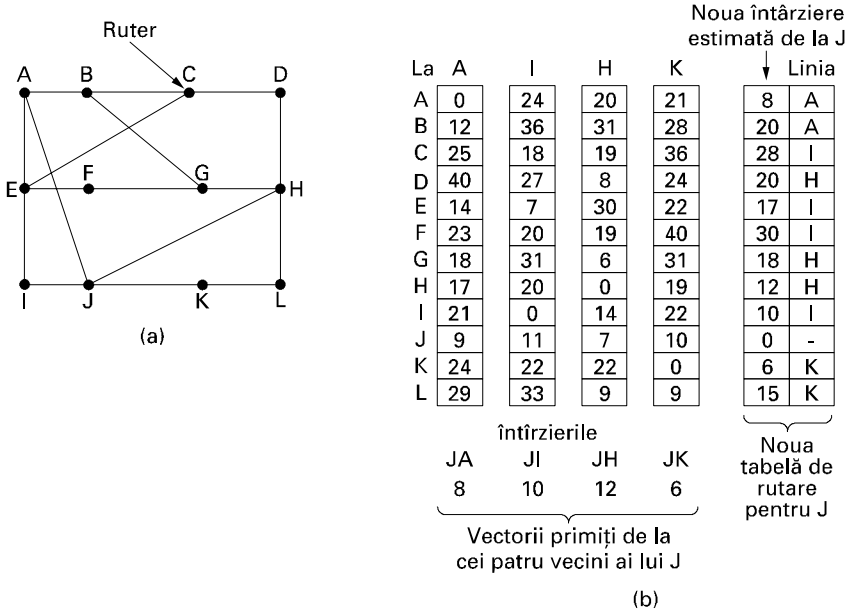
Algoritmul de dirijare cu vectori distanță este cunoscut și sub alte nume, cel mai des algoritmul distribuit de dirijare **Bellman-Ford** și algoritmul **Ford-Fulkerson**, după numele cercetătorilor care l-

au propus (Bellman, 1957; și Ford și Fulkerson, 1962). A fost algoritmul de dirijare folosit inițial în rețeaua ARPANET, a fost folosit de asemenea în Internet sub numele de RIP.

În dirijarea pe baza vectorilor distanță, fiecare ruter păstrează o tabelă de dirijare conținând câte o intrare pentru fiecare ruter din subrețea. Această intrare are două părți: linia de ieșire preferată care se folosește pentru destinația respectivă și o estimare a timpului sau distanței până la acea destinație. Metrica folosită poate fi numărul de salturi, întârzierea în milisecunde, numărul total de pachete care așteaptă în cozi de-a lungul căii, sau ceva asemănător.

Se presupune că ruterul cunoaște „distanța" spre fiecare dintre vecinii săi. Dacă se folosește metrica salturilor, distanța este doar de un salt. Dacă metrica folosită este cea a lungimii cozilor de așteptare, ruterul examinează pur și simplu lungimile acestor cozi. Dacă metrica este cea a întârzierilor, ruterul o poate măsura direct prin pachete speciale ECHO, în care receptorul va marca doar timpul curent (ștampila de timp) și le va trimite înapoi cât mai repede posibil.

Ca exemplu, să presupunem că se folosește metrica întârzierilor și că ruterul cunoaște întârzierea spre fiecare dintre vecinii săi. O dată la fiecare T msec fiecare ruter trimite spre fiecare vecin o listă a estimărilor proprii spre fiecare destinație. De asemenea el recepționează o listă similară de la fiecare vecin. Să presupunem că una dintre aceste tabele tocmai a sosit de la vecinul X, cu X_i fiind estimarea lui X despre cât timp este necesar pentru a ajunge la ruterul i. Dacă ruterul știe că întârzierea spre X este m msec, el știe de asemenea că poate atinge ruterul i trecând prin X în $X_i + m$ msec. Făcând aceste calcule pentru fiecare vecin, un ruter poate stabili care estimare pare a fi cea mai bună, pentru a folosi această estimare, împreună cu linia corespunzătoare în noua tabelă de dirijare. Este de remarcat faptul că vechea tabelă de dirijare nu intervine în calcule.



La	A	I	H	K	Linia
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 -
K	24	22	22	0	6 K
L	29	33	9	9	15 K

Fig. 5-9. (a) O subrețea. (b) Intrări de la A, I, H și K și noua tabelă de dirijare pentru J.

Acest proces de actualizare este ilustrat în fig. 5-9. Partea (a) prezintă o subrețea. Primele patru coloane din partea (b) conțin vectorii de întârzieri primiți de la vecinii ruterului J. A afirmă că are 12 msec întârziere spre B, 25 msec întârziere spre C, 40 msec întârziere spre D etc. Presupunem că

J și-a măsurat sau estimat întârzierea față de vecinii săi *A*, *I*, *H* și *K*, obținând valorile 8, 10, 12 și 16 ms, respectiv.

Să vedem cum calculează *J* noua cale spre ruterul *G*. El știe că poate ajunge la *A* în 8 msec și *A* pretinde că este în stare să ajungă la *G* în 18 msec, astfel încât *J* poate conta pe o întârziere de 26 msec spre G dacă dirijează pachetul spre *A*. Similar, el calculează întârzierea spre *G* prin *I*, *H*, *K* ca fiind 41 (31 + 10), 18 (6 + 12) și 37 (31 + 6) respectiv. Cea mai bună valoare este 18, așa că va crea o intrare în tabela de dirijare cu întârzierea către *G* de 18 msec și ruta de urmat trecând prin *H*. Aceleași calcule se fac pentru toate destinațiile, obținându-se noua tabelă de dirijare, care este prezentată în ultima coloană a figurii.

Problema numărării la infinit

Dirijarea folosind vectori distanță funcționează în teorie, însă în practică are o limitare importantă: deși ea converge spre rezultatul corect, o face foarte lent. În particular, ea reacționează rapid la veștile bune, dar foarte lent la cele rele. Să considerăm un ruter care are un cel mai bun drum spre destinația *X* foarte lung. Dacă la următorul schimb de informații, vecinul său *A* raportează brusc o întârziere mică spre *X*, ruterul va comuta și va folosi linia spre *A* pentru a dirija traficul spre *X*. Astfel, într-o singură schimbare a vectorului, vestea bună a fost luată în considerare.

Pentru a vedea cât de repede se propagă veștile bune, să considerăm subrețeaua (liniară) de cinci noduri din fig. 5-10, unde metrica întârzierilor este numărul de salturi. Să presupunem că inițial nodul *A* nu funcționează și toate celelalte rutere cunosc acest lucru. Cu alte cuvinte, toate celelalte rutere au înregistrat întârzierea spre *A* ca având valoarea infinit.

Când *A* pornește, celelalte rutere află aceasta datorită schimbărilor din vector. Pentru simplificare, vom considera că există un gong uriaș undeva, care bate periodic pentru a iniția schimbul de vectori simultan la toate ruterele. La momentul primului schimb, *B* află că vecinul din stânga are o întârziere nulă spre *A*. Astfel, *B* creează o nouă intrare în tabela sa, marcând faptul că *A* este la un singur salt distanță, spre stânga. Toate celelalte rutere consideră că *A* este încă oprit. Intrările tabelei de dirijare pentru *A*, la acest moment, sunt prezentate în a doua linie din fig. 5-10(a). La următorul schimb, *C* află că *B* are o cale de lungime 1 spre *A*, astfel încât își actualizează tabela de dirijare pentru a indica o cale de lungime 2, însă *D* și *E* nu vor primi vestea cea bună decât mai târziu. Evident, vestea cea bună se răspândește cu viteza de un salt la fiecare schimb. Într-o subrețea având calea cea mai lungă de lungime *N* salturi, după *N* schimburi fiecare ruter va afla despre liniile și ruterele nou apărute.

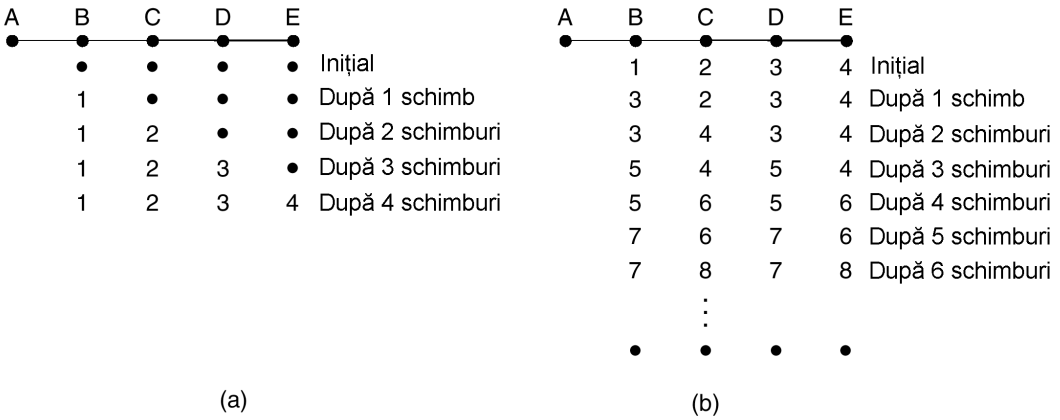


Fig. 5-10. Problema numărării la infinit.

Să considerăm acum situația din fig. 5-10(b), în care toate liniile și ruterele sunt inițial în funcțiune. Ruterele *B*, *C*, *D* și *E* au distanțele spre *A* respectiv de 1, 2, 3, 4. Brusc, *A* se oprește sau, alternativ, linia dintre *A* și *B* este întreruptă, ceea ce reprezintă efectiv același lucru din punctul de vedere al lui *B*.

La primul schimb de pachete, *B* nu primește nimic de la *A*. Din fericire, *C* spune: „Nici o problemă. Eu știu o cale spre *A* de lungime 2." Însă *B* nu știe că această cale a lui *C* trece prin *B* însuși. După cunoștințele lui *B*, *C* ar putea avea zece linii, toate cu căi separate de lungime 2 spre *A*. Prin urmare *B* va crede că poate ajunge la *A* prin *C* pe o cale de lungime 3. *D* și *E* nu își actualizează intrările proprii pentru *A* la primul schimb.

La al doilea schimb, *C* remarcă faptul că fiecare dintre vecinii săi pretinde a avea o cale de lungime 3 spre *A*. El va alege la întâmplare unul dintre acești vecini și va înregistra noua distanță spre *A* ca fiind 4, așa cum se arată în linia a treia din fig. 5-10(b). Schimburile următoare vor produce succesiunea prezentată în continuare în fig. 5-10(b).

Din această figură se poate deduce de ce veștile rele circulă mai lent: orice ruter va avea întotdeauna o valoare cu cel mult unu mai mare decât valoarea minimă a vecinilor săi. Treptat, toate ruterele vor ajunge la infinit, însă numărul de schimburi necesar depinde de valoarea numerică folosită pentru a reprezenta valoarea infinit. Din această cauză este recomandat să se aleagă infinitul, ca fiind lungimea celei mai mari căi, plus 1. Dacă metrica este întârzierea în timp, atunci nu este definită nici o limită superioară, astfel încât este necesară o valoare mare pentru a preveni considerarea unui drum cu întârziere mare ca fiind un drum defect. Nu este deloc surprinzător că această problemă este numită **problema numărării la infinit** (**the count to infinity problem**). Au existat câteva încercări de rezolvare a problemei (cum ar fi orizont împărțit cu revers otrăvit - eng.: split horizon with poisoned reverse - în RFC 1058), dar nici una nu a funcționat bine în general. Miezul problemei este că atunci când *X* îi spune lui *Y* că are o cale spre o destinație, *Y* nu are de unde să știe dacă se află el însuși pe acea cale.

5.2.5 Dirijarea folosind starea legăturilor

Dirijarea folosind vectori distanță a fost folosită în ARPANET până în 1979, când a fost înlocuită prin dirijarea folosind starea legăturilor. Au fost două probleme importante care au cauzat această schimbare. În primul rând, deoarece metrica folosită era lungimea cozilor de așteptare, la stabilirea rutei nu se lua în considerare lățimea de bandă. Inițial toate liniile erau de 56 Kbps, astfel încât lățimea de bandă nu era o problemă, însă după ce câteva linii au fost îmbunătățite la 230 Kbps, iar altele la 1.544 Mbps, neluarea în considerare a lățimii de bandă a devenit o problemă majoră. Evident, era posibil să se schimbe metrica folosită pentru a depinde și de lățimea de bandă, însă exista și o a doua problemă și anume aceea că algoritmul convergea destul de greu (problema numărării la infinit). De aceea, a fost înlocuit cu un algoritm nou, numit algoritm de **dirijare folosind starea legăturilor (link state routing)**. Variante de dirijare folosind starea legăturilor sunt actualmente foarte răspândite.

Ideea algoritmului bazat pe starea legăturilor este simplă și poate fi formulată în 5 puncte. Fiecare ruter trebuie să facă următoarele:

1. Să descopere care sunt vecinii săi și afle adresele de rețea ale acestora.
2. Să măsoare întârzierea sau costul până la fiecare din vecinii săi.
3. Să pregătească un pachet prin care anunță pe toată lumea că tocmai a terminat de cules datele despre vecini.

4. Să trimită acest pachet către toate celelalte rutere.
5. Să calculeze cea mai scurtă cale spre fiecare ruter.

Ca urmare, întreaga topologie și toate întârzierile sunt măsurate experimental și distribuite spre fiecare ruter. Apoi se poate rula algoritmul lui Dijkstra pentru a afla cea mai scurtă cale către fiecare ruter. În continuare vom analiza mai în detaliu acești cinci pași.

Determinarea vecinilor

Când un ruter este pus în funcțiune, prima sa sarcină este să afle care sunt vecinii săi. El realizează aceasta prin trimiterea unui pachet special HELLO pe fiecare linie prin care este legat la alt ruter. Ruterul de la celălalt capăt trebuie să răspundă anunțând într-un pachet identitatea sa. Aceste nume trebuie să fie unice global, pentru că dacă mai târziu un ruter află că trei rutere sunt conectate toate la *F*, este esențial ca acesta să poată determina dacă cele trei se referă la același *F*.

Când două sau mai multe rutere sunt conectate printr-o rețea LAN, situația devine puțin mai complicată. Fig. 5-11(a) ilustrează un LAN la care sunt conectate direct ruterele *A*, *C* și *F*. Fiecare dintre aceste rutere este conectat cu unul sau mai multe alte rutere, așa cum se arată în figură.

O modalitate de a modela rețeaua LAN este de a o considera ca un nod, așa cum se arată în fig. 5-11 (b). Aici am introdus un nod nou, artificial, *N*, la care *A*, *C* și *F* sunt conectate. Faptul că este posibil să se meargă de la *A* la *C* prin LAN este reprezentat aici de calea *ANC*.

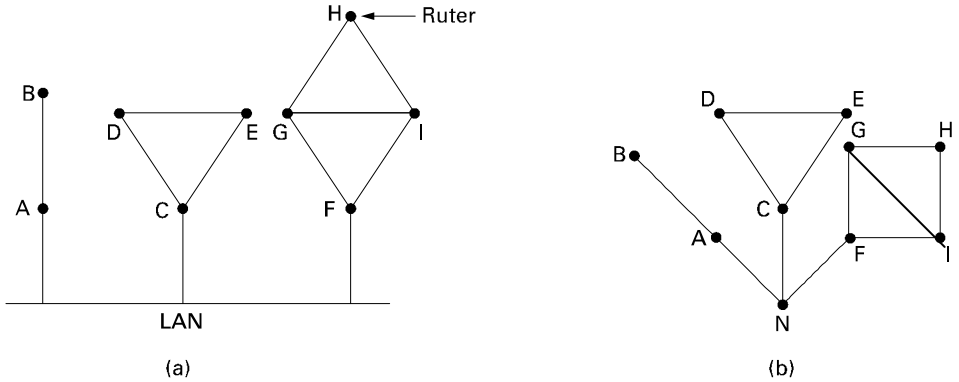


Fig. 5-11. (a) Nouă rutere și o LAN. (b) Graful asociat punctului (a).

Măsurarea costului liniei

Algoritmul de dirijare bazat pe starea legăturilor cere ca fiecare ruter să știe, sau cel puțin să aibă o estimare rezonabilă, a întârzierii către fiecare dintre vecinii săi. Cel mai direct mod de a afla acest lucru este de a trimite un pachet special ECHO pe linie, cerând ca ruterul partener să-l trimită înapoi imediat. Măsurând timpul în care pachetul se întoarce (round-trip time) și împărțindu-l la doi, ruterul inițiator poate avea o estimare rezonabilă a întârzierii. Pentru rezultate și mai bune, testul poate fi repetat de mai multe ori, folosindu-se apoi valoarea medie obținută. Bineînțeles, această metodă presupune implicit că întârzierile sunt simetrice, dar nu mereu este așa.

O problemă interesantă este dacă să se considere sau nu încărcarea rețelei la măsurarea întârzierii. Pentru a ține cont de încărcare, timpul de revenire trebuie măsurat din momentul în care pachetul ECHO este pus în coadă. Pentru a ignora încărcarea, ceasul se poate porni în momentul în care pachetul ECHO ajunge pe prima poziție din coadă.

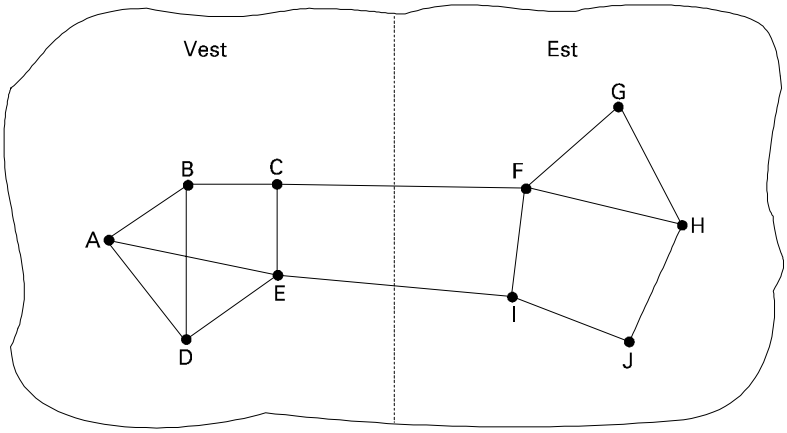


Fig. 5-12. O subrețea în care părțile de Est și Vest sunt conectate prin două linii.

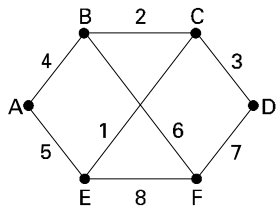
Pot fi aduse argumente în favoarea ambelor variante. Dacă se ține cont de întârzierile provocate de trafic la măsurători înseamnă că dacă un ruter trebuie să aleagă între două linii cu aceeași lățime de bandă, una dintre ele fiind puternic încărcată tot timpul, iar cealaltă nefiind foarte încărcată, atunci ruterul va desemna calea cea mai puțin încărcată ca fiind cea mai scurtă. Această alegere va duce la îmbunătățirea performanțelor.

Din păcate, există și un argument împotriva folosirii încărcării la calculul întârzierii. Să considerăm subrețeaua din fig. 5-12, care este divizată în două zone, Est și Vest, conectate prin două linii, *CF* și *EI*. Să presupunem că majoritatea traficului între Est și Vest folosește linia *CF* și, prin urmare, această linie este puternic încărcată și are întârzieri mari. Folosirea întârzierilor în cozi pentru calculul celei mai scurte căi va face ca drumul *EI* să fie preferat. După ce noile tabele de dirijare au fost instalate, majoritatea traficului Est-Vest va trece acum prin *EI*, supraîncărcând-o. De aceea, la următoarea actualizare, *CF* va părea a fi calea cea mai scurtă. Prin urmare tabelele de dirijare vor oscila puternic, conducând la o dirijare fluctuantă și facilitând apariția multor probleme potențiale. Dacă nu se ține cont de încărcare, luându-se în considerare doar lățimea de bandă, această problemă nu mai apare. Alternativ, încărcarea poate fi distribuită pe ambele linii, dar această soluție nu folosește în întregime calea cea mai bună. Cu toate acestea, pentru a evita oscilațiile în alegerea celei mai bune căi, poate fi înțelept să se distribuie încărcarea pe mai multe linii, în proporții cunoscute pe fiecare linie.

Construirea pachetelor cu starea legăturilor

De îndată ce a fost colectată informația necesară pentru realizarea schimbului, se poate trece la pasul următor, fiecare ruter construind un pachet care conține toate datele. Pachetul începe cu identitatea expeditorului, urmată de un număr de secvență, vârstă (care va fi descrisă în continuare) și o listă a vecinilor. Pentru fiecare vecin se specifică întârzierea asociată. Un exemplu de subrețea este prezentat în fig. 5-13(a), unde etichetele liniilor reprezintă întârzierile asociate. Pachetele cu starea legăturilor asociate tuturor celor șase rutere sunt prezentate în fig. 5-13(b).

Construirea pachetelor cu starea legăturilor se face ușor. Partea mai dificilă este să se determine când să fie construite ele. O posibilitate este ca ele să fie construite periodic, adică la intervale regulate. O altă posibilitate este atunci când se produce un eveniment semnificativ, cum ar fi scoaterea din funcțiune a unui vecin sau a unei linii, sau repunerea lor în funcțiune, sau modificarea semnificativă a proprietăților lor.



(a)

Pachete cu starea legăturilor

A	
Secv.	
Vârstă	
B	4
E	5

B	
Secv.	
Vârstă	
A	4
C	2
F	6

C	
Secv.	
Vârstă	
B	2
D	3
E	1

D	
Secv.	
Vârstă	
C	3
F	7

E	
Secv.	
Vârstă	
A	5
C	1
F	8

F	
Secv.	
Vârstă	
B	6
D	7
E	8

(b)

Fig. 5-13. (a) O subrețea. (b) Pachetele cu starea legăturilor pentru această subrețea.

Distribuirea pachetelor cu starea legăturilor

Cea mai complicată parte a algoritmului este distribuirea sigură a pachetelor cu starea legăturilor. De îndată ce pachetele sunt distribuite și instalate, ruterele care primesc primele pachete își vor schimba rutele. În consecință, rutere diferite ar putea folosi versiuni diferite ale topologiei, ceea ce poate duce la apariția unor inconsistențe, bucle, mașini inaccesibile sau a altor probleme.

Pentru început vom descrie algoritmul de bază folosit pentru distribuție. Apoi vom prezenta unele îmbunătățiri posibile. Ideea fundamentală este folosirea inundării pentru a distribui pachetele cu starea legăturilor. Pentru a avea controlul inundării, fiecare pachet conține un număr de secvență care este incrementat la fiecare nou pachet trimis. Ruterele păstrează evidența tuturor perechilor (ruter sursă, număr secvență) pe care le văd. La sosirea unui nou pachet cu starea legăturilor, el este căutat în lista pachetelor deja văzute. Dacă pachetul este nou, el este retrimis pe toate liniile, cu excepția celei pe care a sosit. Dacă este un duplicat, pachetul este distrus. Dacă pachetul sosit are un număr de secvență mai mic decât cel mai mare număr de secvență detectat, atunci el este rejectat ca fiind învechit.

Acest algoritm are câteva probleme, dar ele sunt tratabile. În primul rând, dacă numerele de secvență ating valoarea maximă posibilă și reîncep de la valori mici, se pot produce confuzii. Soluția este folosirea numerelor de secvență pe 32 biți. Considerând un pachet de starea legăturilor pe secundă, ar trebui 137 ani pentru a se reîncepe de la valori mici, astfel încât această posibilitate poate fi ignorată.

În al doilea rând, dacă un ruter se defectează, el va pierde evidența numerelor de secvență. Dacă va reîncepe de la 0, următorul pachet va fi rejectat ca duplicat.

În al treilea rând, dacă numărul de secvență este alterat și se recepționează 65540 în loc de 4 (eroare de modificare a unui bit), pachetele de la 5 la 65540 vor fi rejectate ca fiind învechite, deoarece se consideră că numărul de secvență curent este 65540.

Soluția tuturor acestor probleme este includerea vârstei pachetului după numărul de secvență și decrementarea sa la fiecare secundă. Când vârsta ajunge la zero, informația de la ruterul respectiv este distrusă. În mod normal un nou pachet apare, să zicem, la fiecare 10 sec., astfel încât informația de la ruter expiră doar dacă ruterul este oprit (sau dacă șase pachete consecutive s-au pierdut, un lucru extrem de improbabil). Câmpul *Age* este decrementat de asemenea de fiecare ruter la începutul procesului de inundare, pentru a se asigura că nici un pachet nu se poate pierde sau nu poate supraviețui o perioadă nelimitată (un pachet având vârsta zero este distrus).

Unele îmbunătățiri ale algoritmului pot să-l facă mai robust. Atunci când un pachet cu starea legăturilor ajunge într-un ruter pentru inundare, acesta nu este pus imediat în coada de transmisie. El este pus într-o zonă de așteptare pentru o scurtă perioadă. Dacă înainte de a fi trimis primul pachet sosește un alt pachet cu starea legăturilor de la aceeași sursă, numerele lor de secvență sunt comparate. Dacă sunt identice, duplicatul este distrus. Dacă sunt diferite, cel mai bătrân este ignorat. Pen-

tru a preîntâmpina erorile pe linia ruter-ruter, toate aceste pachete sunt confirmate. Dacă linia nu este folosită, zona de așteptare este inspectată în manieră round-robin, pentru a selecta un pachet sau o confirmare de trimis.

Structura de date folosită de ruterul *B* pentru subrețeaua din fig. 5-13(a) este descrisă în fig. 5-46. Fiecare linie corespunde unui pachet cu starea legăturilor sosit recent, dar încă neprelucrat în întregime. În tabelă se înregistrează originea pachetului, numărul de secvență, vârsta și datele transportate. În plus mai există unele indicatoare (flags) de trimitere și confirmare pentru fiecare dintre cele trei linii ale lui *B* (respectiv către *A*, *C*, *F*). Indicatorul de trimitere precizează că pachetul trebuie trimis pe linia indicată. Indicatorul de confirmare precizează că respectivul pachet trebuie confirmat.

În fig. 5-14, pachetul cu starea legăturilor de la *A* sosește direct, astfel încât el trebuie trimis către *C* și *F* și trebuie confirmat către *A*, așa cum precizează biții indicatori. Similar, pachetul de la *F* trebuie trimis către *A* și *C* și confirmat către *F*.

			Trimitere			ACK			
Sursa	Secv.	Vârsta	A	C	F	A	C	F	Date
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Fig. 5-14. Zona tampon pentru pachete a ruterului *B* din fig. 5-13.

Oricum, situația celui de-al treilea pachet, de la *E*, este diferită. El sosește de două ori, întâi pe calea *EAB* și apoi pe calea *EFB*. În consecință el trebuie trimis numai către *C*, însă trebuie confirmat atât lui *A* cât și lui *F*, așa cum indică și biții corespunzători.

Dacă sosește un duplicat în timp ce pachetul este încă în zona tampon, biții trebuie modificați. De exemplu, dacă o copie a stării lui *C* sosește de la *F* înainte ca a patra intrare din tabelă să fie trimisă, cei șase biți se vor schimba în 100011, pentru a preciza că pachetul trebuie confirmat către *F*, dar nu trimis acolo.

Calcularea noilor rute

De îndată ce un ruter a acumulat un set complet de pachete cu starea legăturilor, el poate construi graful întregii subrețele, deoarece fiecare legătură este reprezentată. De fapt, fiecare legătură este reprezentată de două ori, o dată pentru fiecare direcție. Cele două valori pot fi mediate sau folosite separat.

Acum poate fi folosit local algoritmul lui Dijkstra, pentru a construi cea mai scurtă cale către toate destinațiile posibile. Rezultatul acestui algoritm poate fi trecut în tabelele de dirijare, iar apoi operațiile normale pot fi reluate.

Pentru o subrețea formată din *n* rutere, fiecare având *k* vecini, memoria necesară pentru a memora datele de intrare este proporțională cu *kn*. Pentru subrețele mari, aceasta poate fi o problemă. De asemenea și timpul de calcul poate fi important. Cu toate acestea, în multe situații, algoritmul bazat pe starea legăturilor funcționează bine.

Oricum, probleme ale hardware-ului sau software-ului pot influența negativ funcționarea acestui algoritm (la fel ca și a altora). De exemplu, dacă un ruter pretinde că are o linie pe care de fapt nu o are, sau uită despre o linie pe care o are, graful subrețelei va fi incorect. Dacă un ruter eșuează în retrimiterea pachetelor sau le alterează în timp ce le retrimite, vor apărea probleme. În fine, dacă un ruter rămâne fără memorie sau calculează greșit rutele, se vor petrece lucruri urâte. Pe măsură ce subrețeaua crește în dimensiune, ajungând la zeci sau sute de mii de noduri, probabilitatea ca un ruter să se defecteze devine neneglijabilă. Trucul care se poate folosi constă în încercarea de limitare a pagubelor atunci când inevitabilul s-a produs. Perlman (1988) tratează în detaliu aceste probleme, precum și soluțiile lor.

Dirijarea bazată pe starea legăturilor este larg folosită în rețelele actuale, astfel încât ar fi potrivite câteva cuvinte despre unele protocoale care o folosesc. Protocolul OSPF, care este extrem de folosit în Internet, utilizează un algoritm bazat pe starea legăturilor. Vom descrie protocolul OSPF în sect. 5.6.4.

Un alt protocol bazat pe starea legăturilor este **IS-IS** (**Intermediate System-Intermediate System,** Sistem Intermediar – Sistem Intermediar), care a fost proiectat pentru DECnet și mai apoi adoptat de ISO pentru a fi folosit cu protocolul neorientat pe conexiune de la nivelul rețea, CLNP. De atunci a fost modificat pentru a se descurca și cu alte protocoale, cel mai important fiind IP. IS-IS este folosit în coloane vertebrale ale Internet-ului (Internet backbones) (inclusiv în vechiul NSFNET) și în unele sisteme digitale celulare cum ar fi CDPD. Novell NetWare folosește o variantă simplificată IS-IS (NLSP) pentru a dirija pachetele IPX.

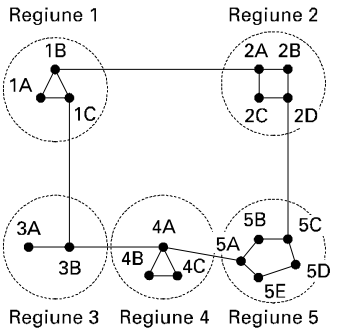
În principiu IS-IS distribuie o imagine a topologiei ruterelor, pe baza căreia se calculează calea cea mai scurtă. Fiecare ruter anunță, în informația de stare a legăturilor sale, ce adrese la nivelul rețea poate să acceseze direct. Aceste adrese pot fi IP, IPX, AppleTalk, sau orice alte adrese. IS-IS poate accepta chiar mai multe protocoale ale nivelului rețea în același timp.

Multe din inovațiile din proiectarea lui IS-IS au fost preluate de OSPF (OSPF a fost proiectat la mulți ani după IS-IS). Acestea includ o metodă auto-stabilizatoare a inundării, folosită la actualizarea stării legăturilor, conceptul de ruter dedicat într-o LAN, metoda de calcul și utilizare a căilor divizate și mai multe metrici. Ca o consecință, vor exista puține diferențe între IS-IS și OSPF. Cea mai importantă diferență este că IS-IS este codificat în așa fel, încât suportă ușor și chiar natural să transporte informațiile mai multor protocoale ale nivelului rețea, o caracteristică pe care OSPF nu o are. Acest avantaj este deosebit de important, în special în mediile mari, multiprotocol.

5.2.6 Dirijare ierarhică

Pe măsură ce rețelele cresc în dimensiune, tabelele de dirijare cresc proporțional. Pe lângă faptul că memoria ruterului este consumată de fiecare nouă creștere a tabelelor, pentru parcurgerea lor este necesar tot mai mult timp de calcul și se folosește tot mai mult din lățimea de bandă pentru a trimite rapoartele de stare despre ele. La un moment dat, rețeaua poate crește până la un punct în care nu mai este posibil ca fiecare ruter să dețină o intrare pentru fiecare alt ruter, astfel încât dirijarea trebuie făcută ierarhic, la fel ca în rețeaua telefonică.

Atunci când se folosește dirijarea ierarhică, ruterele sunt împărțite în ceea ce vom numi **regiuni** (**regions**), fiecare ruter știind toate detaliile necesare pentru a dirija pachete spre destinație în cadrul regiunii sale, dar neștiind nimic despre organizarea internă a celorlalte regiuni. Când rețele diferite sunt interconectate, este natural să privim fiecare rețea ca pe o regiune, pentru a elibera ruterele dintr-o rețea de sarcina de a cunoaște structura topologică a celorlalte.



(a)

Tabela completă pentru 1A

Dest.	Linie	Salturi
1A	—	—
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Tabela ierarhică pentru 1A

Dest.	Linie	Salturi
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Fig. 5-15. Dirijare ierarhică.

Pentru rețele uriașe, o ierarhie pe două niveluri ar putea fi insuficientă, ar putea fi necesar să grupăm regiunile în asociații (clusters), asociațiile în zone, zonele în grupuri și așa mai departe până ce nu mai avem nume pentru aceste aglomerări. Ca un exemplu de ierarhie multinivel, să considerăm cum poate fi dirijat un pachet din Berkeley, California spre Malindi, Kenya. Ruterul din Berkeley cunoaște în detaliu topologia din California, așa că va trimite tot traficul pentru exteriorul statului către ruterul din Los Angeles. Ruterul din Los Angeles este capabil să dirijeze traficul spre alte rutere interne, dar va trimite tot traficul extern spre New York. Ruterul din New York este programat să dirijeze traficul către ruterul din țara de destinație care se ocupă de traficul extern, de exemplu în Nairobi. În final, pachetul va urma calea sa, coborând în arborele din Kenya până ce ajunge la Malindi.

Fig. 5-15 sugerează un exemplu cantitativ al dirijării într-o ierarhie pe două niveluri, având cinci regiuni. Tabela de dirijare completă a ruterului 1*A* are 17 intrări, așa cum se arată în fig. 5-15 (b). Atunci când dirijarea se face ierarhic, așa ca în fig. 5-15(c), există intrări pentru toate ruterele locale, la fel ca și până acum, însă toate celelalte regiuni au fost condensate într-un singur ruter, astfel încât tot traficul pentru regiunea 2 va merge pe linia 1*B*-2*A*, iar restul traficului la distanță va merge pe linia 1*C*-3*B*. Dirijarea ierarhică a redus dimensiunea tabelei de la 17 intrări la 7. Pe măsură ce raportul între numărul de regiuni și numărul de rutere dintr-o regiune crește, se economisește tot mai mult spațiu de memorie.

Din păcate acest câștig de spațiu nu este gratuit. Trebuie plătit un preț și acesta este concretizat în creșterea lungimii căilor. De exemplu, cea mai bună cale de la 1*A* la 5*C* este prin regiunea 2, însă în dirijarea ierarhică tot traficul către regiunea 5 este trimis spre regiunea 3, deoarece așa este mai bine pentru majoritatea destinațiilor din regiunea 5.

Dacă o rețea unică devine prea mare, o întrebare interesantă este: Câte niveluri trebuie să aibă ierarhia? De exemplu, să considerăm o subrețea cu 720 rutere. În absența oricărei ierarhii, tabela de

dirijare a fiecărui ruter trebuie să conțină 720 intrări. Dacă subrețeaua este partiționată în 24 regiuni cu 30 de rutere fiecare, fiecare ruter necesită 30 de intrări locale plus 23 de intrări pentru celelalte regiuni, deci un total de 53 intrări. Dacă se alege o ierarhie pe trei niveluri, cu opt asociații (clusters), fiecare având 9 regiuni a câte 10 rutere fiecare, fiecare ruter are nevoie de 10 intrări pentru ruterele locale, 8 intrări pentru celelalte regiuni din asociația sa și de 7 intrări pentru celelalte asociații, deci un total de 25 intrări. Kamoun și Kleinrock (1979) au descoperit că numărul optim de niveluri pentru o subrețea cu N rutere este $\ln N$, ceea ce necesită un total de $e \ln N$ intrări pentru fiecare ruter. De asemenea ei au arătat că creșterea efectivă a lungimii medii a căilor provocată de dirijarea ierarhică este suficient de mică pentru a fi acceptată.

5.2.7 Dirijarea prin difuzare

În unele aplicații, calculatoarele gazdă au nevoie să trimită mesaje către mai multe sau către toate celelalte calculatoare gazdă. De exemplu, un serviciu de distribuire a rapoartelor meteorologice, de actualizare a cursului acțiunilor sau transmisiuni radio în direct ar putea funcționa mai bine prin difuzarea datelor către toate mașinile, fiind la latitudinea celor interesate de date să le recepționeze. Trimiterea simultană a unui pachet către toate destinațiile se numește **difuzare** (**broadcast**); mai multe metode pentru realizarea ei au fost propuse.

O metodă de difuzare care nu are cerințe speciale pentru subrețea este ca sursa să trimită un pachet distinct către fiecare destinație. Metoda este nu numai consumatoare de lățime de bandă, dar ea cere ca sursa să dețină o listă completă a tuturor destinațiilor. S-ar putea ca în practică aceasta să fie singura metodă utilizabilă, însă ea este metoda cea mai puțin dorită.

Inundarea este un alt candidat evident. Deși inundarea este nepotrivită pentru comunicația obișnuită capăt la capăt, pentru difuzare ar trebui luată serios în considerare, mai ales dacă nici una dintre metodele ce vor fi descrise în continuare nu este aplicabilă. Problema folosirii inundării ca metodă de difuzare este aceeași cu problema ivită la folosirea sa ca algoritm de dirijare capăt la capăt: generează prea multe pachete și consumă lățime de bandă prea mare.

Al treilea algoritm este **dirijarea multidestinație** (**multidestination routing**). Dacă se folosește această metodă, fiecare pachet conține fie o listă a destinațiilor, fie o hartă de biți care indică destinațiile dorite. Atunci când un pachet ajunge la un ruter, ruterul verifică toate destinațiile pentru a determina setul liniilor de ieșire pe care trebuie trimis pachetul. (O linie de ieșire este selectată dacă este calea cea mai bună pentru cel puțin o destinație.) Ruterul generează o nouă copie a pachetului pentru fiecare linie de ieșire folosită și include în fiecare pachet doar acele destinații care folosesc linia respectivă. Efectul este partiționarea mulțimii destinațiilor între liniile de ieșire. După un număr suficient de salturi, fiecare pachet va conține o singură destinație și poate fi tratat ca un pachet normal. Dirijarea multidestinație este asemănătoare trimiterii separate a mai multor pachete către adresele destinație, cu excepția faptului că atunci când mai multe pachete trebuie să urmeze aceeași cale, unul dintre ele plătește tot drumul, iar celelalte călătoresc gratuit.

Al patrulea algoritm de difuzare utilizează explicit arborele de scufundare al ruterului care inițiază difuzarea sau orice alt arbore de acoperire util. Un **arbore de acoperire** (**spanning tree**) este un subset al subrețelei care include toate ruterele și nu conține bucle. Dacă fiecare ruter cunoaște care din liniile sale participă la arborele de acoperire, el poate copia un pachet de difuzare recepționat pe toate liniile de ieșire care fac parte din arborele de acoperire, cu excepția celei pe care a fost recepționat. Această metodă asigură o utilizare deosebit de eficientă a lățimii de bandă, generând numărul minim de pachete necesare pentru a rezolva problema. Singura problemă este că, pentru a fi aplica-

bilă, fiecare ruter trebuie să dețină cunoștințe despre un anume arbore de acoperire. Uneori această informație este disponibilă (de exemplu la dirijarea bazată pe starea legăturilor), iar alteori nu este disponibilă (de exemplu la dirijarea cu vectori distanță).

Ultimul nostru algoritm cu difuzare este o încercare de a aproxima comportamentul precedentului, chiar și atunci când ruterele nu știu absolut nimic despre arborele de acoperire. Ideea, numită **trimiterea pe calea inversă** (**reverse path forwarding**) este remarcabil de simplă odată ce a fost indicată. Când un pachet de difuzare ajunge la un ruter, acesta verifică dacă pachetul a sosit pe linia pe care se trimit de obicei pachete către sursa difuzării. Dacă este așa, este o șansă foarte mare ca însuși pachetul de difuzare să fi urmat cea mai bună cale, fiind astfel prima copie care ajunge la ruter. Aceasta fiind situația, ruterul trimite pachetul pe toate liniile de ieșire, cu excepția celei pe care a sosit. Dacă însă pachetul a sosit pe altă linie decât cea preferată pentru a ajunge la sursă, pachetul este distrus, fiind considerat un posibil duplicat.

Un exemplu de **trimitere pe calea inversă**, este prezentat în fig. 5-16. Partea (a) prezintă subrețeaua, partea (b) arată arborele de scufundare pentru ruterul *I* al subrețelei, iar partea (c) prezintă cum funcționează algoritmul căii inverse. La primul salt, *I* trimite pachete către *F*, *H*, *J* și *N*, așa cum se vede din al doilea nivel al arborelui. Fiecare din aceste pachete ajunge pe calea preferată către *I* (presupunând că ruta preferată face parte din arborele de scufundare), lucru care este indicat printr-un cerc în jurul literei. La saltul următor, se generează opt pachete, două de către fiecare ruter care a primit un pachet la primul salt. Așa cum se vede, toate aceste opt pachete ajung la rutere încă nevizitate și cinci dintre ele ajung pe linia preferată. Dintre cele șase pachete generate la al treilea salt, doar trei sosesc pe linia preferată (în *C, E* și *K*); celelalte sunt duplicate. După cinci salturi și 24 pachete, difuzarea se termină, spre deosebire de patru salturi și 14 pachete necesare dacă arborele de scufundare ar fi fost urmat integral.

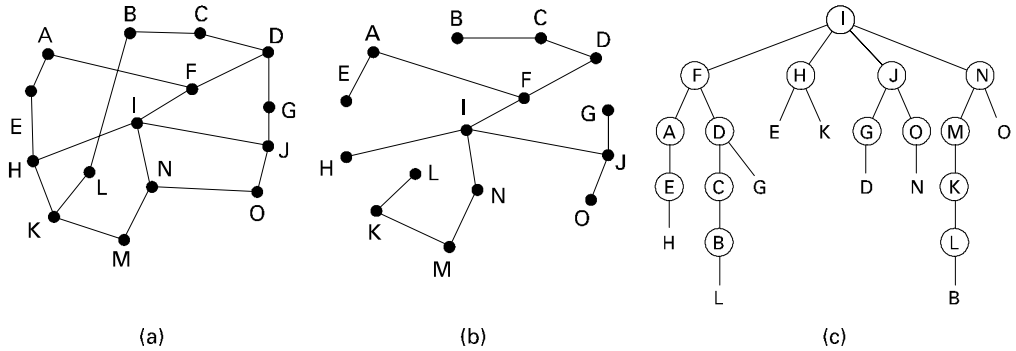


Fig. 5-16. Algoritmul trimiterii pe calea inversă. (a) O subrețea. (b) Un arbore de scufundare. (c) Arborele construit de algoritmul căii inverse.

Principalul avantaj al folosirii căii inverse este acela că este rezonabil de eficient și este ușor de implementat. El nu cere ruterelor să cunoască arborele de acoperire și nici nu are overhead-ul algoritmului de adresare multidestinație, care folosește o listă de destinații sau hartă de biți la fiecare difuzare. De asemenea, el nu necesită nici un mecanism special pentru a opri procesul, ca în cazul inundării (unde se folosește fie un contor al salturilor în fiecare pachet și o cunoaștere a priori a diametrului subrețelei, fie o listă de pachete deja văzute pentru fiecare sursă).

5.2.8 Dirijarea cu trimitere multiplă (multicast)

Unele aplicații necesită ca procese aflate la mari distanțe unele de altele să lucreze în grup, de exemplu, un grup de procese care implementează o bază de date distribuită. În aceste situații, este adesea necesar ca un proces să trimită un mesaj către toți ceilalți membri ai grupului. Dacă grupul este mic, el poate trimite fiecărui partener un mesaj capăt la capăt. Dacă grupul este mare, această strategie este costisitoare. Uneori se poate folosi difuzarea, dar folosirea difuzării pentru a anunța 1000 de mașini dintr-o rețea cu un milion de noduri este ineficientă, deoarece majoritatea receptorilor nu sunt interesați de mesaj (sau chiar mai rău, sunt foarte interesați, dar nu trebuie să vadă mesajul). De aceea, avem nevoie de o modalitate de a trimite mesaje spre grupuri bine definite, care conțin un număr mare de noduri, dar totuși redus față de dimensiunea întregii rețele.

Trimiterea unui mesaj către un astfel de grup se numește **multicasting**, iar algoritmul de dirijare asociat se numește **dirijare multicast** (**multicast routing**). În această secțiune, vom descrie o modalitate de a realiza dirijarea multicast. Pentru informații suplimentare, vezi (Chu et al., 2000; Costa et al. 2001; Kasera et al., 2000; Madruga and Garcia-Luna-Aceves, 2001; Zhang and Ryu, 2001).

Dirijarea multicast necesită managementul grupului. Trebuie să existe modalități de a crea și distruge grupuri și de a permite proceselor să intre în grupuri sau să le părăsească. Cum se realizează aceste funcții nu este treaba algoritmului de dirijare. Important pentru algoritmul de dirijare este că atunci când un proces se atașează unui grup, el trebuie să informeze gazda sa despre aceasta. Este important ca ruterele să știe căror grupuri le aparțin calculatoarele gazdă asociate. Fie calculatoarele gazdă trebuie să anunțe ruterul asociat la producerea unei modificări în alcătuirea grupurilor, fie ruterul trebuie să interogheze periodic aceste calculatoare gazdă. În ambele cazuri, ruterul află din ce grupuri fac parte calculatoarele gazdă. Ruterele își informează vecinii, astfel că informația se propagă prin subrețea.

Pentru a realiza dirijarea multicast, fiecare ruter calculează arborele de acoperire care acoperă toate celelalte rutere din subrețea. De exemplu, în fig. 5-17(a) avem două grupuri, 1 și 2. Unele rutere sunt atașate la calculatoare gazdă care aparțin unuia sau ambelor grupuri, așa cum se indică în figură. Un arbore de acoperire pentru cel mai din stânga ruter este prezentat în fig. 5-17(b).

Atunci când un proces trimite un pachet multicast către un grup, primul ruter își examinează arborele de acoperire și îl retează, eliminând toate liniile care nu conduc către calculatoare gazdă, membre ale grupului. În exemplul nostru, fig. 5-17(c) arată arborele de acoperire retezat pentru grupul 1. Similar, fig. 5-17(d) arată arborele de acoperire retezat pentru grupul 2. Pachetele multicast sunt dirijate doar de-a lungul arborelui de acoperire corespunzător.

Sunt posibile mai multe moduri de retezare a arborelui de acoperire. Cel mai simplu se poate folosi dacă se utilizează dirijarea bazată pe starea legăturilor și fiecare ruter cunoaște întreaga topologie a subrețelei, inclusiv apartenența calculatoarelor gazdă la grupuri. Atunci arborele poate fi retezat pornind de la sfârșitul fiecărei căi, mergând spre rădăcină și eliminând toate ruterele care nu aparțin grupului respectiv.

În cazul dirijării folosind vectori distanță, poate fi aplicată o altă strategie de retezare a arborelui. Algoritmul de bază folosit este trimiterea pe calea inversă. Oricum, ori de câte ori un ruter fără nici un calculator gazdă interesat de un anume grup și fără nici o conexiune la alte rutere primește un mesaj multicast pentru acel grup, el va răspunde cu un mesaj PRUNE (retezare), anunțând expeditorul să nu îi mai trimită mesaje multicast pentru acel grup. Când un ruter care nu are printre calculatoarele gazdă nici un membru al vreunui grup primește astfel de mesaje pe toate liniile sale, el poate de asemenea să răspundă cu un mesaj PRUNE. În acest fel subrețeaua este retezată recursiv.

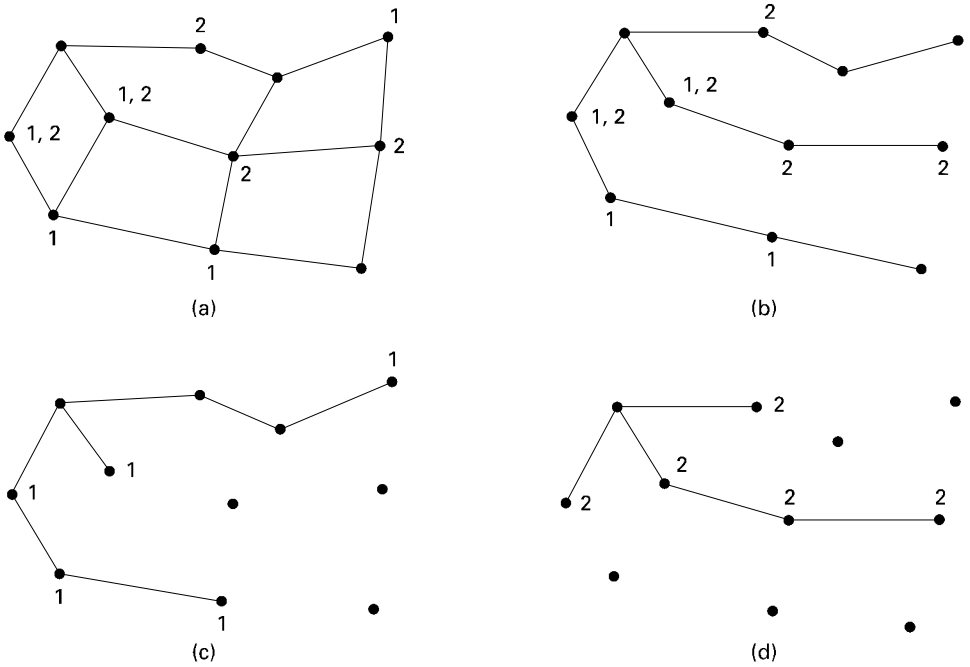


Fig. 5-17. (a) O rețea. (b) Un arbore de acoperire pentru cel mai din stânga ruter. (c) Un arbore multicast al grupului 1. (d) Un arborele multicast al grupului 2.

Un potențial dezavantaj al acestui algoritm este acela că se comportă deficitar în cazul rețelelor extinse. Să presupunem că o rețea are *n* grupuri, fiecare cu un număr mediu de *m* membri. Pentru fiecare grup, trebuie memorați *m* arbori de acoperire retezați, deci un total de *mn* arbori. Dacă există multe grupuri mari, atunci, pentru a memora toți arborii, este necesar un spațiu de memorie considerabil.

O alternativă de proiectare folosește **arbori de bază** (**core-base trees**) (Ballardie ș.a., 1993). Aici se calculează un singur arbore de acoperire pentru fiecare grup, având rădăcina (core - inima, nucleu) lângă mijlocul grupului. Pentru a trimite un mesaj multicast, un calculator gazdă trimite mesajul către rădăcină, care apoi îl trimite de-a lungul arborelui de acoperire. Deși acest arbore nu va fi optim pentru toate sursele, reducerea costului memorării de la *m* arbori la unul singur pe grup reprezintă o economie semnificativă.

5.2.9 Dirijarea pentru calculatoare gazdă mobile

Milioane de oameni dețin astăzi calculatoare portabile și, de obicei, doresc să-și citească poșta electronică și să-și acceseze sistemele de fișiere uzuale din orice punct al lumii s-ar afla. Aceste calculatoare mobile introduc o nouă complicație: pentru a dirija un pachet către un calculator mobil, rețeaua trebuie mai întâi să-l localizeze. Problema integrării calculatoarelor mobile într-o rețea este foarte recentă, dar în această secțiune vom sugera unele abordări și vom da o posibilă soluție.

Modelul lumii folosit de obicei de proiectanții de rețele este cel prezentat în fig. 5-18. Aici avem o rețea WAN formată din rutere și calculatoare gazdă. La WAN sunt conectate LAN-uri, MAN-uri și celule de comunicație fără fir de tipul celor descrise în Cap. 2.

Calculatoarele gazdă care nu se mișcă niciodată se numesc **staționare**. Ele sunt conectate la rețea prin fire de cupru sau fibre optice. Prin contrast, putem distinge alte două categorii de calculatoare gazdă. Calculatoarele gazdă **migratoare** sunt de fapt calculatoare gazdă staționare, dar care, din când în când, se mută dintr-un loc fixat în altul și care folosesc rețeaua doar atunci când sunt conectate fizic. Calculatoarele gazdă **călătoare** efectuează calcule în timp ce se deplasează și doresc să mențină legătura în timpul deplasării. Vom folosi termenul **gazdă mobilă** pentru a desemna fiecare dintre ultimele două categorii, adică toate calculatoarele gazdă care sunt departe de casă și doresc totuși să fie conectate.

Se presupune că toate calculatoarele gazdă au o **locație de domiciliu** permanentă (home location), care nu se modifică niciodată. Calculatoarele gazdă au de asemenea o adresă personală permanentă, care poate fi folosită pentru a determina locația de domiciliu, într-o manieră similară celei în care numărul de telefon 1-212-5551212 indică Statele Unite (codul de țară 1) și Manhattan (212). Scopul dirijării în sistemele cu calculatoare gazdă mobile este de a face posibilă trimiterea pachetelor spre gazde mobile folosind adresa lor personală permanentă și să se asigure o trimitere eficientă a pachetelor spre ele, oriunde ar fi acestea. Problema este, bineînțeles, modul de localizare a lor.

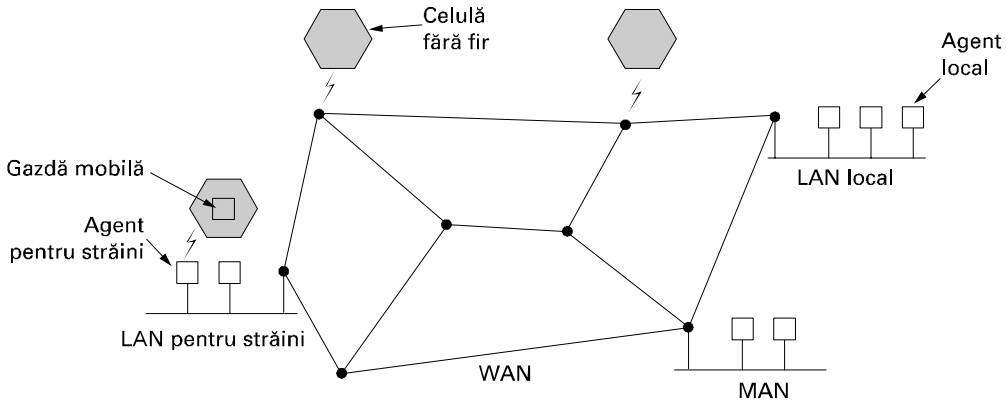


Fig. 5-18. O rețea WAN la care sunt conectate rețele LAN, MAN și celule de comunicație fără fir.

În modelul din fig. 5-18, lumea este divizată (geografic) în unități de dimensiune redusă. Să denumim aceste unități domenii, unde un domeniu este de obicei un LAN sau o celulă de comunicație fără fir. Fiecare domeniu are unul sau mai mulți **agenți pentru străini (foreign agent)**, procese ce țin evidența tuturor gazdelor mobile care vizitează domeniul. În plus, fiecare domeniu are un **agent local (local agent)** care ține evidența calculatoarelor gazdă cu domiciliul în domeniul respectiv, dar care momentan vizitează alte domenii.

Când un nou calculator gazdă pătrunde într-un domeniu, fie prin conectarea la acesta (atașarea fizică la LAN), fie prin plimbarea printr-o celulă, trebuie să se înregistreze la agentul pentru străini din domeniul respectiv. Procedura de înregistrare se desfășoară de obicei astfel:

1. Periodic, fiecare agent pentru străini difuzează un pachet anunțându-și existența și adresa. Un utilizator mobil nou sosit trebuie să aștepte unul dintre aceste mesaje, însă dacă nici

unul nu sosește într-un interval rezonabil de timp, calculatorul mobil poate difuza un pachet care spune: „Este vreun agent pentru străini prin zonă?”

2. Calculatorul mobil se înregistrează la agentul pentru străini precizând adresa sa permanentă, adresa actuală a nivelului legătură de date, precum și unele informații de securitate.
3. Agentul pentru străini contactează apoi agentul local al domeniului din care provine utilizatorul mobil și îi spune: „Unul dintre calculatoarele tale gazdă se află chiar aici.” Mesajul agentului pentru străini către agentul local conține adresa de rețea a agentului pentru străini. El mai conține de asemenea și informația de securitate, pentru a convinge agentul local că gazda mobilă este într-adevăr acolo.
4. Agentul local examinează informația de securitate, care conține și o ștampilă de timp, pentru a dovedi că a fost generată în ultimele câteva secunde. Dacă este mulțumit, atunci anunță agentul pentru străini că poate trece la acțiune.
5. Când agentul pentru străini primește confirmarea de la agentul local, el creează o nouă intrare în tabela sa și anunță utilizatorul mobil că a fost înregistrat.

Ideal, atunci când un calculator gazdă părăsește un domeniu, acest lucru trebuie anunțat, pentru a fi scos din evidență, însă mulți utilizatori pur și simplu închid calculatorul când termină.

Când se trimite un pachet către o gazdă mobilă, el este dirijat către domiciliul calculatorului gazdă, deoarece adresa sugerează că așa trebuie făcut, după cum se ilustrează și în pasul 1 din fig. 5-19. Aici emițătorul, aflat în orașul de nord-vest Seattle, dorește să trimită un pachet către un calculator gazdă aflat în New York, traversând Statele Unite. Pachetele trimise către calculatorul gazdă mobil în LAN-ul său de domiciliu, în New York, sunt interceptate de agentul local de acolo. Apoi agentul local caută noua locație (temporară) a gazdei mobile și află adresa agentului pentru străini care se ocupă de gazdele mobile, în Los Angeles.

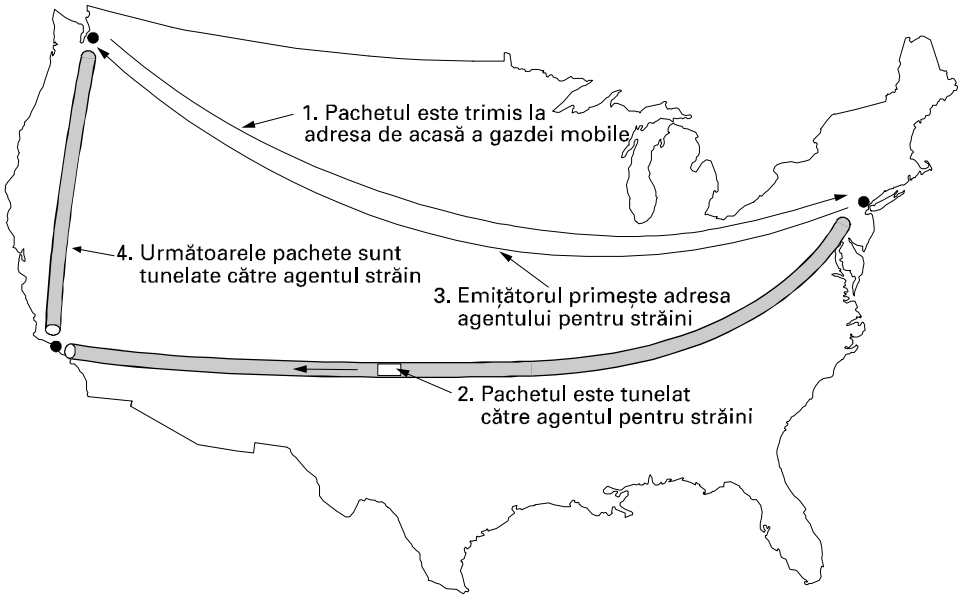


Fig. 5-19. Dirijarea pachetelor pentru gazde mobile.

Agentul local face două lucruri. În primul rând, încapsulează pachetul în câmpul de informație utilă (payload) al unui pachet de trimis, pe care îl expediază apoi către agentul pentru străini (pasul 2

din fig. 5-19). Acest mecanism se numește tunelare; îl vom studia în detaliu puțin mai târziu. După ce recepționează pachetul încapsulat, agentul pentru străini extrage pachetul inițial din câmpul payload și îl trimite către calculatorul gazdă mobil drept cadru al nivelului legătură de date.

În al doilea rând, agentul local anunță expeditorul mesajului ca de acum încolo să trimită pachetele adresate gazdei mobile încapsulându-le în câmpul de informație utilă (payload) al unor pachete trimise explicit agentului pentru străini, în loc să le trimită la adresa de domiciliu a calculatorului gazdă mobil (pasul 3). Pachetele următoare vor putea fi dirijate direct către utilizator, prin intermediul agentului pentru străini (pasul 4), evitând trecerea prin locația de domiciliu.

Diversele scheme propuse diferă prin mai multe aspecte. În primul rând, cât din acest protocol este realizat de rutere și cât de calculatoarele gazdă și, în această ultimă situație, pe care nivel de pe calculatorul gazdă. În al doilea rând, în unele scheme există rutere de-a lungul traseului care înregistrează adresele modificate, astfel încât să fie posibilă interceptarea și redirectarea traficului chiar înainte ca acesta să ajungă la locația de domiciliu. În al treilea rând, în unele variante fiecare vizitator primește o adresă temporară unică; în altele adresa temporară se referă la un agent care se ocupă de toți vizitatorii.

În al patrulea rând, variantele propuse diferă prin modul de rezolvare a situației în care pachetele adresate unei destinații trebuie livrate în altă parte. O variantă este schimbarea adresei destinație și retransmiterea pachetului modificat. Ca o alternativă, întregul pachet, adresa de domiciliu și toate celelalte pot fi încapsulate în câmpul de informație utilă (payload) al altui pachet care este trimis spre adresa temporară. În fine, schemele diferă prin aspectele legate de securitate. În general, când un ruter sau calculator gazdă primește un mesaj de forma „Începând din acest moment, vă rog să trimiteți toate mesajele de poștă electronică ale lui Stephany către mine," el ar putea avea dubii în legătură cu partenerul de dialog și dacă este o idee bună. Diferite protocoale pentru calculatoare gazdă mobile sunt discutate și comparate în (Hac și Guo, 2000; Perkins, 1998a; Snoeren și Balakrishan, 2000; Solomon, 1998; și Wang și Chen, 2001).

5.2.10 Dirijarea în rețele AD HOC

Am văzut cum se face dirijarea atunci când calculatoarele gazdă sunt mobile, dar ruterele sunt fixe. Un caz aparte apare atunci când însăși ruterele sunt mobile. Printre situațiile posibile sunt:

1. Vehicule militare pe un câmp de luptă atunci când nu există o infrastructură.
2. O flotă aflată în larg.
3. Echipe de intervenție la un cutremur ce a distrus infrastructura.
4. O adunare de persoane cu calculatoare portabile într-o zonă fără 802.11.

În toate aceste cazuri, și altele, fiecare nod este compus dintr-un ruter și o gazdă, de obicei pe același calculator. Rețelele de noduri ce întâmplător se află aproape unul de celălalt sunt numite **rețele ad hoc** (**ad hoc network**) sau **MANET** (**Mobile Ad hoc NETworks**, rețele mobile ad hoc). Să le studiem pe scurt. Mai multe informații pot fi găsite în (Perkins, 2001).

Diferența între rețelele ad hoc și rețelele cablate (eng.: wired) este că toate regulile despre topologii fixe, vecini ficși și cunoscuți, relații fixe stabilite între adresa IP și locație, și altele, sunt complet suprimate. Cât ai clipi ruterele pot să apară și să dispară sau să apară în alte locuri. În cazul rețelelor cablate, dacă un ruter are o cale validă către o destinație, aceasta continuă sa fie validă un timp nedefinit (suportând o defecțiune în altă parte a sistemului). Cu o rețea ad hoc, topologia se poate schimba mereu, așa că oportunitatea și chiar validitatea cailor se poate schimba spontan, fără averti-

zare. Nu mai trebuie spus că în aceste circumstanțe dirijarea în rețelele ad hoc diferă foarte mult față de dirijarea în echivalentul lor fix.

Au fost propuși diferiți algoritmi de dirijare pentru rețelele ad hoc. Unul dintre cei mai interesanți este algoritmul de dirijare **AODV** (**Ad hoc On-demand Distance Vector**, vectori distanță ad hoc la cerere) (Perkins și Royer, 1999). Este o rudă îndepărtată a algoritmului cu vectori distanță Bellman-Ford dar adaptat pentru un mediu mobil și care ia în considerare limitele lărgimii de bandă și durata scurtă de funcționare a unei baterii în acest mediu. O altă caracteristică neobișnuită este faptul că acesta este un algoritm la cerere, adică determină o cale către o anumită destinație doar atunci când cineva dorește să trimită un pachet către acea destinație. Să vedem acum ce înseamnă asta.

Descoperirea rutei

La orice moment de timp, o rețea ad hoc poate fi descrisă de un graf de noduri (rutere + gazde). Două noduri sunt conectate (de exemplu, există un arc ce le unește în graf) dacă pot comunica direct folosind radioul. Cum unul dintre ele poate avea un transmițător mai puternic decât celălalt, este posibil ca *A* să fie conectat cu *B* dar *B* să nu fie conectat cu *A*. Totuși, pentru a simplifica, presupunem că toate conexiunile sunt simetrice. De asemenea ar trebui observat că simplul fapt că fiecare din cele două noduri este în raza radio a celuilalt nu înseamnă că ele sunt conectate. Pot exista clădiri, dealuri sau alte obstacole care să blocheze comunicația.

Pentru a descrie algoritmul, considerăm rețeaua ad hoc din fig. 5-20, în care un proces al nodului *A* dorește să transmită un pachet nodului *I*. Algoritmul AODV menține o tabelă în fiecare nod, în care cheia este destinația, și care dă informații despre acea destinație, inclusiv cărui vecin trebuie trimise pachetele pentru a ajunge la destinație. Să presupunem că *A* se uită în tabelă și nu găsește nici o intrare pentru *I*. Trebuie deci să descopere o rută până la *I*. Această proprietate de descoperire a rutelor doar atunci când sunt necesare este cea care face din acest algoritm un algoritm „la cerere”.

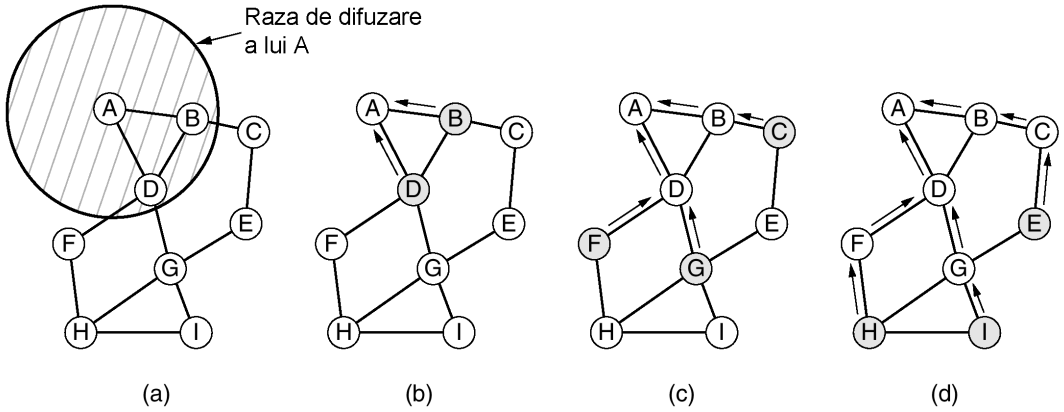


Fig. 5-20. (a) Raza de difuzare a lui *A*. (b) După ce *B* și *D* au primit difuzarea lui *A*. (c) După ce *C, F* și *G* au primit difuzarea lui *A*. (d) După ce *E, H* și *I* au primit difuzarea lui *A*. Nodurile hașurate sunt noii destinatari. Săgețile arată calea inversă posibilă.

Pentru a îl localiza pe *I*, *A* construiește un pachet special ROUTE REQUEST și îl difuzează. Pachetul ajunge la *B* și la *D,* așa cum este ilustrat în fig. 5-20(a). De fapt, motivul pentru care *B* și *D* sunt conectate cu *A* în graf este acela că pot comunica cu *A*. *F*, de exemplu, nu este prezentat ca fiind legat printr-un arc cu *A* deoarece nu poate recepționa semnalul radio emis de *A*.

Formatul pachetului ROUTE REQUEST este prezentat în fig. 5-21. El conține adresele sursă și destinație, bineînțeles adrese IP, care specifică cine caută pe cine. Conține de asemenea și un *ID Cerere* (Request ID), care este un contor local ținut de fiecare nod și incrementat la fiecare difuzare a unui pachet ROUTE REQUEST. Împreună, câmpurile *adresă* Sursă și *ID Cerere* identifică unic pachetul ROUTE REQUEST pentru a permite nodurilor să înlăture orice duplicat pe care-l primesc.

Adresa sursei	ID cerere	Adresa destinației	Numărul de secvență al sursei	Numărul de secvență al destinației	Contorul de salturi

Fig. 5-21. Formatul unui pachet ROUTE REQUEST

În plus, pe lângă contorul *ID Cerere*, fiecare nod menține un al doilea contor de secvență incrementat de fiecare dată când este trimis un pachet ROUTE REQUEST (sau un răspuns la un ROUTE REQUEST). Acesta funcționează asemănător unui ceas și este folosit pentru a deosebi rutele noi de cele vechi. Cel de-al patrulea câmp din fig. 5-21. este contorul de secvențe al lui *A*; al cincilea câmp este cea mai recentă valoare a contorului de secvențe a lui *I* întâlnită de *A* (0 dacă nu a întâlnit-o niciodată). Rolul acestor câmpuri se va clarifica în scurt timp. Ultimul câmp, *contorul de salturi* (Hop count), va memora câte salturi a făcut pachetul. Acesta este inițializat cu 0.

Când un pachet ROUTE REQUEST ajunge la un nod (în acest caz *B* și *D*), este prelucrat astfel:

1. Perechea (Adresă sursă, ID Cerere) este căutată într-o tabelă locală cu istoria cererilor pentru a vedea dacă această cerere a mai fost întâlnită și procesată. Dacă este un duplicat, atunci este înlăturat și procesarea se oprește. Dacă nu este un duplicat, atunci perechea este introdusă tabelă pentru ca viitoarele duplicate să poată fi rejectate, și procesarea continuă.
2. Receptorul caută destinația în propria tabela de dirijare. Dacă a aflat o nouă cale către destinație, atunci sursei îi este trimis un pachet ROUTE REPLY, spunându-i cum se ajunge la destinație. Nouă înseamnă că *numărul de secvență al destinației* memorat în tabela de dirijare este mai mare sau egal cu numărul de secvență al destinației din pachetul ROUTE REQUEST. Dacă este mai mic, calea memorată este mai veche decât calea anterioară pe care sursa o avea pentru destinație, așa că se execută pasul 3.
3. Deoarece receptorul nu știe o rută nouă către destinație, incrementează câmpul *contor de salturi* și redifuzează pachetul ROUTE REQUEST. De asemenea extrage datele din pachet și le memorează ca o nouă intrare în tabela de căi inverse. Această informație va fi folosită la construcția caii inverse pentru ca răspunsul să se poată întoarce mai târziu la sursă. Săgețile din fig. 5-20 sunt folosite pentru construirea căii inverse. De asemenea este pornit un cronometru asociat intrării corespunzătoare căii inverse nou create. Dacă expiră, intrarea este ștearsă.

Nici B și nici D nu știu unde este I, așa că fiecare creează o intrare pentru calea inversă indicând către A, așa cum arată săgețile din fig. 5-20, și difuzează pachetul cu contorul de salturi setat pe 1. Difuzarea de la B ajunge la C și D. C creează o intrare pentru el în tabela de căi inverse și-l redifuzează. În contrast, *D* îl rejectează ca pe un duplicat. Similar, difuzarea făcută de *D* este rejectată de *B*. Totuși, difuzarea făcută de *D* este acceptată de *F* și *G* și memorată, așa cum este prezentat în fig. 5-20(c). După ce *E*, *H* și *I* primesc difuzarea, pachetul ROUTE REQUEST ajunge în final la o destinație care știe unde se află *I*, de fapt, chiar *I*, așa cum este ilustrat în fig. 5-20(d). Observați că deși am prezentat difuzările în trei pași discreți, difuzările de la noduri diferite nu sunt coordonate în nici un fel.

Ca răspuns la cererea venită, *I* construiește un pachet ROUTE REPLY, prezentat în fig. 5-22. *Adresa sursă, Adresa destinație* și *Contorul de salturi* sunt copiate din cererea venită, dar *Numărul de secvență al destinației e*ste luat din contorul său din memorie. Contorul de salturi este setat la 0. Câmpul *Durată de viață* controlează perioada în care ruta este validă. Acest pachet este trimis doar către nodul de la care a venit pachetul ROUTE REQUEST, în acest caz, *G*. Apoi urmează calea inversă către *D* și în final către *A*. La fiecare nod, *Contorul de salturi* este incrementat astfel încât nodul să poată vedea cât de departe de destinație (*I*) este.

Adresa sursei	Adresa destinației	Numărul de secvență al destinației	Contorul de salturi	Durata de viață

Fig. 5-22. Formatul pachetului ROUTE REPLY

Pachetul este analizat la fiecare nod intermediar de pe drumul înapoi. El este introdus în tabela locală de dirijare ca rută către *I* dacă una sau mai multe din cele trei condiții de mai jos este îndeplinită:

1. Nu se cunoaște nici o rută către *I*.
2. Numărul de secvență pentru *I* din pachetul ROUTE REPLY este mai mare decât valoarea din tabela de dirijare.
3. Numerele de secvență sunt egale, dar noua rută este mai scurtă.

În acest mod, toate nodurile de pe calea inversă află și ruta către *I*, ca o consecință a descoperirii rutei de către *A.* Nodurile care au primit pachetul ROUTE REQUEST original, dar nu sunt pe calea inversă (*B, C, E, F,* și *H* în acest exemplu) vor înlătura intrarea din tabela de căi inverse la expirarea timpului asociat.

Într-o rețea mare, algoritmul generează multe difuzări, chiar pentru destinații aflate foarte aproape. Numărul de difuzări poate fi redus după cum urmează. *Durata de viață* a pachetului IP este inițializată de emițător cu valoarea diametrului rețelei și decrementată la fiecare salt. Dacă ajunge la valoarea 0, pachetul este înlăturat în loc să fie difuzat.

Procesul descoperirii este modificat după cum urmează. Pentru a localiza o destinație, emițătorul difuzează un pachet ROUTE REQUEST cu *Durata de viață* setată la 1. Dacă nu primește nici un răspuns într-un timp rezonabil, este trimis un alt pachet, de această dată cu *Durata de viață* setată la 2. La încercările următoare se utilizează 3, 4, 5 etc. În acest fel se încearcă întâi local, apoi în cercuri din ce în ce mai largi.

Întreținerea rutei

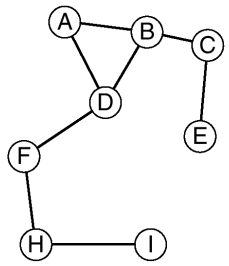
Deoarece nodurile se pot muta sau pot fi oprite, topologia se poate schimba spontan. De exemplu, în fig. 5-20, dacă *G* este oprit, *A* nu își va da seama că ruta pe care o folosea către *I* (*ADGI*) nu mai este validă. Algoritmul trebuie să poată rezolva această situație. Periodic, fiecare nod difuzează un mesaj *Hello.* Fiecare vecin ar trebui să răspundă. Dacă nu este primit nici un răspuns, atunci emițătorul știe că respectivul vecin s-a mutat din raza sa și nu mai este conectat cu el. Similar, dacă încearcă să-i trimită un pachet unui vecin ce nu răspunde, află că vecinul nu mai este disponibil.

Această informație este folosită pentru a elimina rutele care nu mai funcționează. Pentru fiecare destinație posibilă, fiecare nod, *N*, memorează vecinii care au trimis un pachet către acea destinație în ultimele ΔT secunde. Aceștia sunt numiți **vecinii activi** (**active neighbors**) ai lui *N* . *N* realizează acest lucru prin intermediul unei tabele de dirijare în care cheia este destinația și care conține nodul de ieșire ce trebuie utilizat pentru a ajunge la destinație, contorul de salturi până la destinație, cel mai recent

număr de secvență al destinației și lista vecinilor activi pentru acea destinație. O tabelă de dirijare posibilă pentru nodul *D* pentru topologia din exemplul nostru este prezentată în fig. 5-23(a).

Dest.	Următorul salt	Distanța	Vecini activi	Alte câmpuri
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

Fig. 5-23. (a) Tabela de dirijare a lui *D* înaintea opririi lui *G*. (b) Graful după ce *G* s-a oprit.

Când oricare dintre vecinii lui *N* devine inaccesibil, el verifică tabela de dirijare pentru a vedea ce destinații au rute ce trec prin vecinul dispărut. Pentru fiecare din aceste rute, vecinii activi sunt informați că ruta ce trece prin *N* este acum inutilizabilă și trebuie ștearsă din tabela lor de dirijare. Vecinii activi transmit mai departe informația vecinilor lor activi, și așa mai departe, recursiv, până ce toate rutele ce depindeau de nodul devenit inaccesibil sunt șterse din toate tabelele de dirijare.

Ca exemplu de întreținere a rutelor, să considerăm exemplul anterior, dar cu *G* oprit brusc. Topologia schimbată este ilustrată în fig. 5-23(b). Când *D* descoperă că *G* nu mai este în rețea, se uită în tabela de dirijare și observă că *G* era folosit pentru rutele către *E, G,* și *I.* Mulțimea vecinilor activi pentru aceste destinații este {*A, B*}. Cu alte cuvinte, *A* și *B* depind de *G* pentru câteva dintre rutele lor, așa că ele trebuie informate că aceste rute nu mai sunt valide. *D* le informează prin trimiterea de pachete care le determină să-și actualizeze tabelele de dirijare. De asemenea, *D* șterge intrările pentru *E, G,* și *I* din tabela sa de dirijare.

Poate nu a fost evident din descrierea noastră, dar o diferență critică între AODV și Bellman-Ford este că nodurile nu difuzează periodic întreaga lor tabelă de dirijare. Această diferență economisește atât lățimea de bandă, cât și durata de viață a bateriei.

De asemenea, AODV este capabil de dirijare prin difuzare și de dirijare cu trimitere multiplă. Pentru detalii, consultați (Perkins și Royer, 2001). Dirijarea ad hoc este un domeniu de cercetare fierbinte. S-a publicat mult despre acest subiect. Câteva dintre lucrări sunt (Chen et. al, 2002; Hu și Johnson, 2001; Li et. al, 2001; Raju și Garcia-Luna-Aceves, 2001; Ramanathan și Redi, 2002; Royer și Toh, 1999; Spohn și Garcia-Luna-Aceves, 2001; Tseng et. al, 2001; și Zadeh et. al, 2002).

5.2.11 Căutarea nodurilor în rețele punct la punct

Un fenomen relativ nou este reprezentat de rețelele punct la punct, în care un număr mare de persoane, de obicei cu conexiuni permanente la Internet, sunt în contact pentru a partaja resurse. Prima aplicație larg răspândită a tehnologiei punct la punct a fost un delict de anvergură: 50 de milioane de utilizatori ai Napster-ului făceau schimb de cântece cu copyright fără permisiunea proprietarilor până când utilizarea Napster-ului a fost interzisă de tribunal în vâltoarea unor mari controverse. Totuși, tehnologia punct la punct are multe utilizări interesante și legale. De asemenea, are

ceva asemănător cu problema dirijării, deși nu este chiar la fel cu cele studiate până acum. Cu toate acestea, merită să-i acordăm puțină atenție.

Ceea ce face ca sistemele punct la punct să fie interesante este faptul că ele sunt în întregime distribuite. Toate nodurile sunt simetrice și nu există un control central sau o ierarhie. Într-un sistem punct la punct tipic fiecare utilizator are o informație ce poate fi de interes pentru alți utilizatori. Această informație poate fi software gratuit, muzică (din domeniul public), fotografii și așa mai departe. Dacă numărul de utilizatori este mare, aceștia nu se știu între ei și nu știu unde să găsească ceea ce caută. O soluție este o bază de date centrală, dar așa ceva ar putea fi irealizabil din diverse motive (de exemplu, nimeni nu dorește să o găzduiască și s-o administreze). Astfel, problema se reduce la felul în care un utilizator găsește un nod ce conține ceea ce caută el, în absența unei baze de date centralizate sau a unui index centralizat.

Să presupunem că un utilizator are una sau mai multe unități de date cum ar fi cântece, fotografii, programe, fișiere ș.a.m.d, pe care ar dori să le citească alți utilizatori. Fiecare unitate are ca nume un șir de caractere ASCII. Un posibil utilizator cunoaște numai șirul de caractere ASCII și vrea să afle dacă una sau mai multe persoane au copii, și în caz că au, care sunt adresele IP ale acestora.

De exemplu, să considerăm o bază de date genealogică distribuită. Fiecare specialist în genealogie are o serie de înregistrări on-line despre strămoșii și rudele sale, posibil și cu fotografii, clipuri audio sau chiar video ale persoanei. Mai multe persoane pot avea același străbunic, astfel că un strămoș poate avea înregistrări în mai multe noduri. Numele înregistrării este numele persoanei într-o formă canonică. La un moment dat, un specialist în genealogie descoperă testamentul străbunicului său într-o arhivă, în care străbunicul lasă moștenire nepotului ceasul de buzunar de aur. Genealogistul știe numele nepotului și dorește să afle dacă alt genealogist are o înregistrare pentru el. Cum aflăm, fără o bază de date centrală, cine are, dacă are cineva, aceste înregistrări?

Pentru a rezolva această problemă au fost propuși diverși algoritmi. Cel pe care îl vom studia este Chord (Dabek ș.a.,2001; și Stoica ș.a.,2001). O explicație simplificată a felului în care funcționează este următoarea. Sistemul Chord constă din n utilizatori participanți, fiecare dintre ei având câteva înregistrări memorate și fiecare dintre ei fiind pregătit să memoreze fragmente din index pentru uzul celorlalți utilizatori. Fiecare nod utilizator are o adresă IP care poate fi convertită într-un număr pe m biți, folosind o funcție de dispersie, *hash*. Pentru *hash* Chord folosește SHA-1. SHA-1 este folosit în criptografie; îl vom analiza în Cap. 8. Pentru moment, este doar o funcție care primește ca argument un șir de caractere de lungime variabilă și generează un număr aleator pe 160 de biți. Deci, putem converti orice adresă IP într-un număr pe 160 de biți numit **identificatorul nodului** (**node identifier**).

Conceptual, toți cei 2^{160} identificatori de noduri sunt așezați în ordine crescătoare într-un mare cerc. Câțiva corespund nodurilor participante, dar majoritatea nu. În fig. 5-24(a) prezentăm un cerc de identificatori de noduri pentru $m = 5$ (pentru moment ignorați arcele din centru). În acest exemplu, nodurile cu identificatorii 1, 4, 7, 12, 15, 20 și 27 corespund nodurilor reale și sunt hașurate; restul nodurilor nu există.

Să definim acum funcția *successor(k), ca* identificator al primului nod real ce urmează după k pe cerc, în sensul acelor de ceasornic. De exemplu, successor(6) = 7, successor(8) = 12 și successor(22) = 27.

Numele înregistrărilor (nume de cântece, numele strămoșilor etc.) sunt de asemenea convertite cu funcția *hash* (adică SHA-1) pentru a genera un număr pe 160 de biți, numit **cheie (key)**. Astfel, pentru a converti un *nume* (numele ASCII al înregistrării) la cheia sa, folosim *key = hash (name)*. Acest calcul este doar un apel local al procedurii *hash*. Dacă o persoană deținând o înregistrare genealogică pentru *nume* dorește să o facă publică, atunci construiește mai întâi un tuplu (*nume, adresă*

IP) și apoi îl roagă pe *successor(hash(nume))* să-l memoreze. Dacă există mai multe înregistrări (în noduri diferite) pentru același nume, toate tuplurile lor vor fi memorate în același nod. În acest mod, index-ul este distribuit aleator pe noduri. Pentru toleranța la defecte, pentru memorarea fiecărui tuplu în *p* noduri pot fi folosite *p* funcții de dispersie diferite, dar nu ne vom referi aici la acest aspect.

Dacă ulterior un utilizator caută *nume*, îi aplică funcția de dispersie pentru a afla cheia și apoi folosește *successor(cheie)* pentru a afla adresa IP a nodului ce memorează tuplurile respective din index. Primul pas este ușor; dar al doilea nu. Pentru a face posibilă găsirea adresei IP corespunzătoare unei anumite chei, fiecare nod trebuie să mențină structuri de date administrative. Una dintre acestea este adresa IP a nodului succesor, de-a lungul cercului de identificatori de noduri. De exemplu, în fig. 5-24, succesorul nodului 4 este 7 și succesorul nodului 7 este 12.

Căutarea se poate desfășura după cum urmează. Nodul care face cererea trimite succesorului său un pachet conținând adresa sa IP și cheia pe care o caută. Pachetul se propagă prin inel până ce localizează succesorul identificatorului căutat. Nodul verifică dacă deține vreo informație cu cheia respectivă și, dacă este așa, o returnează direct nodului care a făcut cererea, a cărui adresă IP o are.

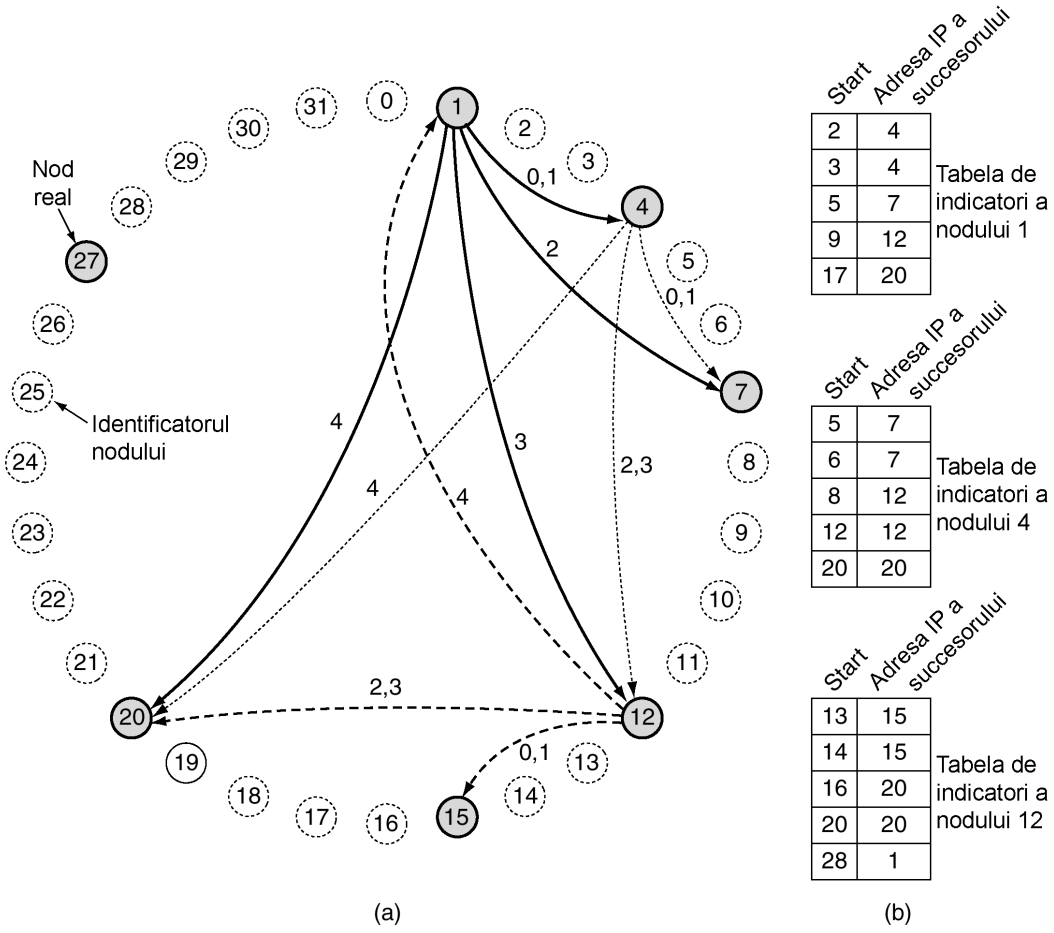


Fig. 5-24. (a) Un set de 32 de identificatori de noduri aranjați în cerc. Nodurile hașurate corespund mașinilor existente. Arcele arată indicatorii de la nodurile 1, 4 și 12. Etichetele arcelor reprezintă indici în tabele. (b) Exemple de tabele de indicatori.

Ca o primă optimizare, fiecare nod ar putea memora adresele IP ale succesorului și predecesorului, astfel încât interogările ar putea fi trimise fie în sensul acelor de ceasornic, fie în sens invers, pe calea considerată mai scurtă. De exemplu, nodul 7 din fig. 5-24 poate să trimită în sensul acelor de ceasornic pentru a găsi nodul cu identificatorul 10, dar în sens invers pentru a găsi nodul cu identificatorul 3.

Chiar și cu două posibilități pentru direcție, căutarea liniară prin toate nodurile este foarte ineficientă într-un sistem punct la punct mare, deoarece numărul mediu de noduri implicat într-o căutare este de *n/2*. Pentru a mări considerabil viteza de căutare, fiecare nod memorează de asemenea ceea ce Chord numește **tabelă de indicatori (finger table)**. Tabela de indicatori are *m* intrări, indexate de la 0 la *m-1,* fiecare indicând către un nod real diferit. Fiecare dintre intrări are două câmpuri: *start* și adresa IP a *successor(start)*, așa cum este prezentat pentru cele trei noduri din fig. 5-24(b). Valoarea acestor câmpuri pentru intrarea *i* a nodului *k* sunt:

start = $k + 2^i$ *(modulo* 2^m*)*
Adresa IP a successor(start[i])

Observați că fiecare nod memorează adresa IP a unui număr relativ mic de noduri și că majoritatea acestora sunt foarte apropiate din punctul de vedere al identificatorilor de noduri.

Folosind tabela de indicatori, căutarea *cheii* la nodul *k* se desfășoară după cum urmează. Dacă *cheia* este între *k* și *successor*(*k*), atunci nodul care deține informația despre *cheie* este *succesor(k)* și căutarea se termină. Altfel, se caută în tabela de indicatori pentru a găsi intrarea al cărui câmp *start* este predecesorul cel mai apropiat al *cheii.* Apoi se trimite direct la adresa IP din intrarea din tabela de indicatori o cerere în care se cere continuarea căutării. Deoarece aceasta este mai apropiată de cheie, dar inferioară, sunt mari șanse să fie capabilă să returneze un răspuns după un număr mic de interogări suplimentare. De fapt, deoarece fiecare căutare înjumătățește distanța rămasă până la țintă, se poate demonstra că numărul mediu de căutări este $\log_2 n$.

Ca prim exemplu, să considerăm căutarea *key* = 3 la nodul 1. Deoarece nodul 1 știe că 3 se află între el și succesorul său, 4, nodul dorit este 4 și căutarea s-a terminat, returnându-se adresa IP a nodului 4.

Ca un al doilea exemplu, să considerăm căutarea *key* = 14 la nodul 1. Deoarece 14 nu este între 1 și 4, este analizată tabela de indicatori. Cel mai apropiat predecesor al lui 14 este 9, așa că cererea este trimisă către adresa IP din intrarea 9, și anume, cea a nodului 12. Nodul 12 vede că 14 se află între el și successor(15), așa că returnează adresa IP a nodului 15.

Ca un al treilea exemplu, să considerăm căutarea *key* = 16 la nodul 1. Din nou se trimite o interogare la nodul 12, dar de această dată nodul 12 nu cunoaște răspunsul. Caută nodul cel mai apropiat predecesor al lui 16 și îl găsește pe 14, care furnizează adresa IP a nodului 15. Acestuia îi este trimisă o interogare. Nodul 15 observă că 16 se află între el și succesorul său (20), așa că returnează apelantului adresa IP a lui 20, care se întoarce către nodul 1.

Deoarece nodurile apar și dispar mereu, Chord trebuie să trateze cumva aceste operații. Presupunem că atunci când sistemul a început să funcționeze, el era îndeajuns de mic pentru ca nodurile să poată schimba informații direct, pentru a construi primul cerc și tabelele de indicatori. După aceea este necesară o procedură automată, după cum urmează. Când un nod nou, *r*, vrea să se alăture, trebuie să contacteze un nod existent și să-i ceară să caute adresa IP a successor(r). După aceea noul nod îl întreabă pe successor(r) cine este predecesorul său. Apoi noul nod cere ambelor noduri să îl insereze pe *r* între ele în cerc. De exemplu, dacă 24 din fig. 5-24 vrea să se alăture, roagă orice nod să caute *successor(24)*, care este 27. Apoi îl întreabă pe 27 cine este predecesorul său (20). După ce le

înștiințează pe amândouă de existența sa, 20 îl folosește pe 24 ca succesor, iar 27 îl folosește pe 24 ca predecesor. În plus, nodul 27 predă cheile din intervalul 21-24, care acum îi aparțin lui 24. În acest moment, 24 este inserat pe deplin.

Totuși, multe tabele de indicatori sunt acum greșite. Pentru a le corecta, fiecare nod rulează un proces în fundal care calculează periodic fiecare indicator prin apelarea funcției *successor*. Când una dintre aceste interogări ajunge la un nod nou, este actualizată intrarea corespunzătoare indicatorului.

Când un nod pleacă normal, predă cheile succesorului său și informează predecesorul de plecarea sa, astfel încât predecesorul poate să se lege la succesorul nodului care a plecat. Când se defectează un nod apare o problemă, deoarece predecesorul său nu are un succesor valid. Pentru a soluționa problema, fiecare nod păstrează date nu numai despre succesorul său direct, dar și despre *s* succesori direcți, pentru a-i permite să sară peste *s-1* noduri consecutive defecte și să se reconecteze la cerc.

Chord a fost folosit la construirea unui sistem de fișiere distribuit (Dabek ș.a.,2001) și alte aplicații, iar cercetările continuă. Un sistem punct la punct diferit, Pastry, și aplicațiile sale sunt descrise în (Rowstron și Druschel, 2001a; și Rowstron și Druschel, 2001b). Un al treilea sistem punct la punct, Freenet, este discutat în (Clarke ș.a., 2002). Un al patrulea sistem de acest tip este descris în (Ratnasamy ș.a., 2001).

5.3 ALGORITMI PENTRU CONTROLUL CONGESTIEI

Atunci când foarte multe pachete sunt prezente într-o subrețea (sau o parte a unei subrețele), performanțele se degradează. Această situație se numește **congestie** (**congestion**). Fig. 5-25 prezintă simptomele. Când numărul de pachete emise în subrețea de calculatoarele gazdă nu depășește capacitatea de transport, ele sunt livrate integral (cu excepția celor care sunt afectate de erori de transmisie), iar numărul celor livrate este proporțional cu numărul celor emise. Totuși, atunci când traficul crește prea mult, ruterele încep să nu mai facă față și să piardă pachete. Aceasta tinde să înrăutățească lucrurile. La un trafic foarte intens performanțele se deteriorează complet și aproape nici un pachet nu mai este livrat.

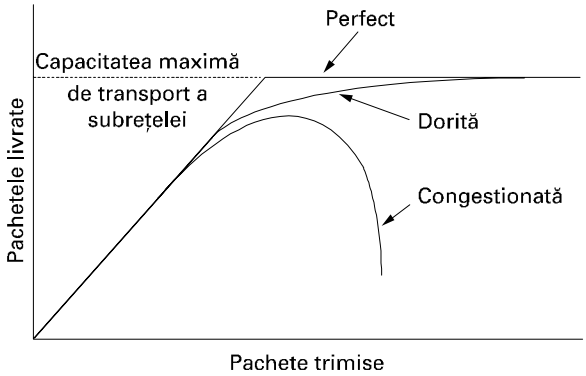


Fig. 5-25. Dacă traficul este prea intens, apare congestia și performanțele se degradează puternic.

Congestia poate fi produsă de mai mulți factori. Dacă dintr-o dată încep să sosească șiruri de pachete pe trei sau patru linii de intrare și toate necesită aceeași linie de ieșire, atunci se va forma o coadă. Dacă nu există suficientă memorie pentru a le păstra pe toate, unele se vor pierde. Adăugarea de memorie poate fi folositoare până la un punct, dar Nagle (1987) a descoperit că dacă ruterele ar avea o cantitate infinită de memorie, congestia s-ar înrăutăți în loc să se amelioreze, deoarece până să ajungă la începutul cozii pachetele au fost deja considerate pierdute (timeout repetat) și s-au trimis duplicate. Toate aceste pachete vor fi trimise cu conștiinciozitate către următorul ruter, crescând încărcarea de-a lungul căii către destinație.

Și procesoarele lente pot cauza congestia. Dacă unitatea centrală (CPU) a ruterului este lentă în execuția funcțiilor sale (introducerea în cozi, actualizarea tabelelor etc.), se pot forma cozi, chiar dacă linia de comunicație nu este folosită la capacitate. Similar și liniile cu lățime de bandă scăzută pot provoca congestia. Schimbarea liniilor cu unele mai performante și păstrarea aceluiași procesor sau vice-versa de obicei ajută puțin, însă de cele mai multe ori doar deplasează punctul critic. De asemenea, îmbunătățirea parțială și nu totală a sistemului cel mai adesea mută punctul critic în altă parte. Adevărata problemă este de multe ori o incompatibilitate între părți ale sistemului. Ea va persista până ce toate componentele sunt în echilibru.

Este important să subliniem diferența dintre controlul congestiei și controlul fluxului, deoarece relația este subtilă. Controlul congestiei trebuie să asigure că subrețeaua este capabilă să transporte întreg traficul implicat. Este o problemă globală, implicând comportamentul tuturor calculatoarelor gazdă, al tuturor ruterelor, prelucrarea de tip memorează-și-retransmite (store-and-forward) din rutere și toți ceilalți factori care tind să diminueze capacitatea de transport a subrețelei.

Controlul fluxului, prin contrast, se referă la traficul capăt la capăt între un expeditor și un destinatar. Rolul său este de a împiedica un expeditor rapid să trimită date continuu, la o viteză mai mare decât cea cu care destinatarul poate consuma datele. Controlul fluxului implică frecvent existența unui feed-back de la receptor către emițător, pentru a spune emițătorului cum se desfășoară lucrurile la celălalt capăt.

Pentru a vedea diferența dintre aceste două concepte, să considerăm o rețea cu fibre optice cu o capacitate de 1000 gigabiți/sec pe care un supercalculator încearcă să transfere un fișier către un calculator personal la 1 Gbps. Deși nu există nici o congestie (rețeaua nu are nici un fel de probleme), controlul fluxului este necesar pentru a forța supercalculatorul să se oprească des, pentru a permite calculatorului personal să și respire.

La cealaltă extremă, să considerăm o rețea de tip memorează-și-retransmite (store-and-forward), cu linii de 1 Mbps și 1000 de calculatoare mari, din care jumătate încearcă să transfere fișiere la 100 Kbps către cealaltă jumătate. Aici problema nu apare datorită unui emițător rapid care surclasează un receptor lent, ci pentru că traficul cerut depășește posibilitățile rețelei.

Motivul pentru care controlul congestiei și controlul fluxului sunt adesea confundate este acela că unii algoritmi pentru controlul congestiei funcționează trimițând mesaje înapoi către diferitele surse, spunându-le să încetinească atunci când rețeaua are probleme. Astfel, un calculator gazdă poate primi un mesaj de încetinire fie din cauză că receptorul nu suportă încărcarea, fie pentru că rețeaua este depășită. Vom reveni asupra acestui punct mai târziu.

Vom începe studiul algoritmilor pentru controlul congestiei prin studiul unui model general de tratare a acesteia. Apoi ne vom ocupa de principalele măsuri pentru prevenirea sa. După aceea, vom analiza o serie de algoritmi dinamici pentru tratarea congestiei odată ce a apărut.

5.3.1 Principii generale ale controlului congestiei

Multe din problemele care apar în sistemele complexe, cum ar fi rețelele de calculatoare, pot fi privite din punctul de vedere al unei teorii a controlului. Această abordare conduce la împărțirea tuturor soluțiilor în două grupe: în buclă deschisă și în buclă închisă. Soluțiile în buclă deschisă încearcă să rezolve problema printr-o proiectare atentă, în esență să se asigure că problema nu apare. După ce sistemul este pornit și funcționează, nu se mai fac nici un fel de corecții.

Instrumentele pentru realizarea controlului în buclă deschisă decid când să se accepte trafic nou, când să se distrugă pachete și care să fie acestea, realizează planificarea deciziilor în diferite puncte din rețea. Toate acestea au ca numitor comun faptul că iau decizii fără a ține cont de starea curentă a rețelei.

Prin contrast, soluțiile în buclă închisă se bazează pe conceptul de reacție inversă (feedback loop). Această abordare are trei părți, atunci când se folosește pentru controlul congestiei:

1. Monitorizează sistemul pentru a detecta când și unde se produce congestia.
2. Trimite aceste informații către locurile unde se pot executa acțiuni.
3. Ajustează funcționarea sistemului pentru a corecta problema.

În vederea monitorizării subrețelei pentru congestie se pot folosi diverse de metrici. Cele mai utilizate sunt procentul din totalul pachetelor care au fost distruse din cauza lipsei spațiului temporar de memorare, lungimea medie a cozilor de așteptare, numărul de pachete care sunt retransmise pe motiv de timeout, întârzierea medie a unui pachet, deviația standard a întârzierii unui pachet. În toate cazurile, valorile crescătoare indică creșterea congestiei.

Al doilea pas în bucla de reacție este transferul informației legate de congestie de la punctul în care a fost depistată la punctul în care se poate face ceva. Varianta imediată presupune trimiterea unor pachete de la ruterul care a detectat congestia către sursa sau sursele de trafic, pentru a raporta problema. Evident, aceste pachete suplimentare cresc încărcarea rețelei exact la momentul în care acest lucru era cel mai puțin dorit, subrețeaua fiind congestionată.

Există însă și alte posibilități. De exemplu, poate fi rezervat un bit sau un câmp în fiecare pachet, pentru a fi completat de rutere dacă congestia depășește o anumită valoare de prag. Când un ruter detectează congestie, el completează câmpurile tuturor pachetelor expediate, pentru a-și preveni vecinii.

O altă abordare este ca ruterele sau calculatoarele gazdă să trimită periodic pachete de probă pentru a întreba explicit despre congestie. Aceste informații pot fi apoi folosite pentru a ocoli zonele cu probleme. Unele stații de radio au elicoptere care zboară deasupra orașelor pentru a raporta congestiile de pe drumuri și a permite ascultătorilor mobili să își dirijeze pachetele (mașinile) astfel încât să ocolească zonele fierbinți.

În toate schemele cu feedback se speră că informarea asupra producerii congestiei va determina calculatoarele gazdă să ia măsurile necesare pentru a reduce congestia. Pentru ca o schemă să funcționeze corect, duratele trebuie reglate foarte atent. Dacă un ruter strigă STOP de fiecare dată când sosesc două pachete succesive și PLEACĂ (eng.: GO) de fiecare dată când este liber mai mult de 20 μsec, atunci sistemul va oscila puternic și nu va converge niciodată. Pe de altă parte, dacă va aștepta 30 de minute pentru a fi sigur înainte de a spune ceva, mecanismul pentru controlul congestiei reacționează prea lent pentru a fi de vreun folos real. Pentru a funcționa corect sunt necesare unele medieri, dar aflarea celor mai potrivite constante de timp nu este o treabă tocmai ușoară.

Se cunosc numeroși algoritmi pentru controlul congestiei. Pentru a oferi o modalitate de organizare a lor, Yang și Reddy (1995) au dezvoltat o taxonomie pentru algoritmii de control al congestiei. Ei încep prin a împărți algoritmii în cei în buclă deschisă și cei în buclă închisă, așa cum s-a precizat anterior. În continuare împart algoritmii cu buclă deschisă în unii care acționează asupra sursei și alții care acționează asupra destinației. Algoritmii în buclă închisă sunt de asemenea împărțiți în două subcategorii, cu feedback implicit și cu feedback explicit. În algoritmii cu feedback explicit, pachetele sunt trimise înapoi de la punctul unde s-a produs congestia către sursă, pentru a o avertiza. În algoritmii impliciți, sursa deduce existența congestiei din observații locale, cum ar fi timpul necesar pentru întoarcerea confirmărilor.

Prezența congestiei înseamnă că încărcarea (momentană) a sistemului este mai mare decât cea pe care o pot suporta resursele (unei părți a sistemului). Pentru rezolvare vin imediat în minte două soluții: sporirea resurselor sau reducerea încărcării. De exemplu subrețeaua poate începe să folosească linii telefonice pentru a crește temporar lățimea de bandă între anumite puncte. În sistemele bazate pe sateliți, creșterea puterii de transmisie asigură de regulă creșterea lățimii de bandă. Spargerea traficului pe mai multe căi în locul folosirii doar a celei mai bune poate duce efectiv la creșterea lățimii de bandă. În fine, ruterele suplimentare, folosite de obicei doar ca rezerve pentru copii de siguranță (backups) (pentru a face sistemul tolerant la defecte), pot fi folosite pentru a asigura o capacitate sporită atunci când apar congestii serioase.

Totuși, uneori nu este posibilă creșterea capacității sau aceasta a fost deja crescută la limită. Atunci singura cale de a rezolva congestia este reducerea încărcării. Sunt posibile mai multe metode pentru reducerea încărcării, cum ar fi refuzul servirii anumitor utilizatori, degradarea serviciilor pentru o parte sau pentru toți utilizatorii și planificarea cererilor utilizatorilor într-o manieră mai previzibilă.

Unele dintre aceste metode, pe care le vom studia pe scurt, pot fi aplicate cel mai bine circuitelor virtuale. Pentru subrețelele care folosesc intern circuite virtuale aceste metode pot fi utilizate la nivelul rețea. Pentru subrețele bazate pe datagrame ele pot fi totuși folosite uneori pentru conexiuni la nivelul transport. În acest capitol, ne vom concentra pe folosirea lor în cadrul nivelului rețea. În următorul, vom vedea ce se poate face la nivelul transport pentru a controla congestia.

5.3.2 Politici pentru prevenirea congestiei

Să începem studiul asupra metodelor pentru controlul congestiei prin analiza sistemelor cu buclă deschisă. Aceste sisteme sunt proiectate astfel încât să minimizeze congestia, în loc să o lase să se producă și apoi să reacționeze. Ele încearcă să-și atingă scopul folosind politici corespunzătoare, la diferite niveluri. În fig. 5-26 sunt prezentate diferite politici pentru nivelul legătură de date, rețea și transport, care pot influența congestia (Jain, 1990).

Să începem cu nivelul legătură de date și să ne continuăm apoi drumul spre niveluri superioare. Politica de retransmisie stabilește cât de repede se produce timeout la un emițător și ce transmite acesta la producerea timeout-ului. Un emițător vioi, care produce repede timeout și retransmite toate pachetele în așteptare folosind retrimiterea ultimelor *n*, va produce o încărcare mai mare decât un emițător calm, care folosește retrimiterea selectivă. Strâns legată de acestea este politica de memorare în zonă tampon. Dacă receptorii distrug toate pachetele în afara secvenței, acestea vor trebui retransmise ulterior, introducând o încărcare suplimentară. În ceea ce privește controlul congestiei, repetarea selectivă este, în mod sigur, mai bună decât retrimiterea ultimelor *n*.

Nivel	Politică
Transport	Politica de retransmisie Politica de memorare temporară a pachetelor în afară de secvență (out-of-order caching) Politica de confirmare Politica de control al fluxului Determinarea timeout-ului
Rețea	Circuite virtuale contra datagrame în interiorul subrețelei Plasarea pachetelor în cozi de așteptare și politici de servire Politica de distrugere a pachetelor Algoritmi de dirijare Gestiunea timpului de viață al pachetelor
Legătură de date	Politica de retransmitere Politica de memorare temporară a pachetelor în afară de secvență (out-of-order caching) Politica de confirmare Politica de control al fluxului

Fig. 5-26. Politici care influențează congestia.

Și politica de confirmare afectează congestia. Dacă fiecare pachet este confirmat imediat, pachetele de confirmare vor genera un trafic suplimentar. Totuși, în cazul în care confirmările sunt preluate de traficul de răspuns, se pot produce timeout-uri și retransmisii suplimentare. O schemă prea strânsă pentru controlul fluxului (de exemplu fereastră mică) reduce volumul de date și ajută în lupta cu congestia.

La nivelul rețea, alegerea între folosirea circuitelor virtuale și datagrame influențează congestia, deoarece mulți algoritmi pentru controlul congestiei funcționează doar pe subrețele cu circuite virtuale. Plasarea în cozi de așteptare a pachetelor și politicile de servire specifică dacă ruterele au o coadă pentru fiecare linie de intrare, o coadă pentru fiecare linie de ieșire sau ambele. Mai precizează ordinea în care se prelucrează pachetele (de exemplu round robin sau bazată pe priorități). Politica de distrugere a pachetelor este regula care stabilește ce pachet este distrus dacă nu mai există spațiu. O politică bună va ajuta la eliminarea congestiei, pe când una greșită o va accentua.

Un algoritm de dirijare bun poate ajuta la evitarea congestiei prin răspândirea traficului de-a lungul tuturor liniilor, în timp ce un algoritm neperformant ar putea trimite toate pachetele pe aceeași linie, care deja este congestionată. În fine, gestiunea timpului de viață asociat pachetelor stabilește cât de mult poate trăi un pachet înainte de a fi distrus. Dacă acest timp este prea mare, pachetele pierdute vor încurca pentru mult timp activitatea, iar dacă este prea mic, este posibil să se producă timeout înainte de a ajunge la destinație, provocând astfel retransmisii.

La nivelul transport apar aceleași probleme ca la nivelul legăturii de date dar, în plus, determinarea intervalului de timeout este mai dificil de realizat, deoarece timpul de tranzit prin rețea este mai greu de prezis decât timpul de tranzit pe un fir între două rutere. Dacă intervalul de timeout este prea mic, vor fi trimise inutil pachete suplimentare. Dacă este prea mare, congestia se va reduce, însă timpul de răspuns va fi afectat de pierderea unui pachet.

5.3.3 Controlul congestiei în subrețelele bazate pe circuite virtuale

Metodele pentru controlul congestiei descrise anterior sunt în principiu în buclă deschisă: ele încearcă, în primul rând, să prevină apariția congestiei, în loc să o trateze după ce a apărut. În această

secțiune, vom descrie unele abordări ale controlului dinamic al congestiei în subrețelele bazate pe circuite virtuale. În următoarele două, vom studia tehnici ce pot fi folosite în orice subrețea.

O tehnică larg răspândită pentru a împiedica agravarea unei congestii deja apărute este **controlul admisiei**. Ideea este simplă: odată ce s-a semnalat apariția congestiei, nu se mai stabilesc alte circuite virtuale până ce problema nu s-a rezolvat. Astfel, încercarea de a stabili o nouă conexiune la nivel transport eșuează. Lăsând tot mai mulți utilizatori să stabilească conexiuni, nu ar face decât să agraveze lucrurile. Deși această abordare este dură, ea este simplă și ușor de realizat. În sistemul telefonic, dacă o centrală devine supraaglomerată, ea practică controlul admisiei, nemaifurnizând tonul.

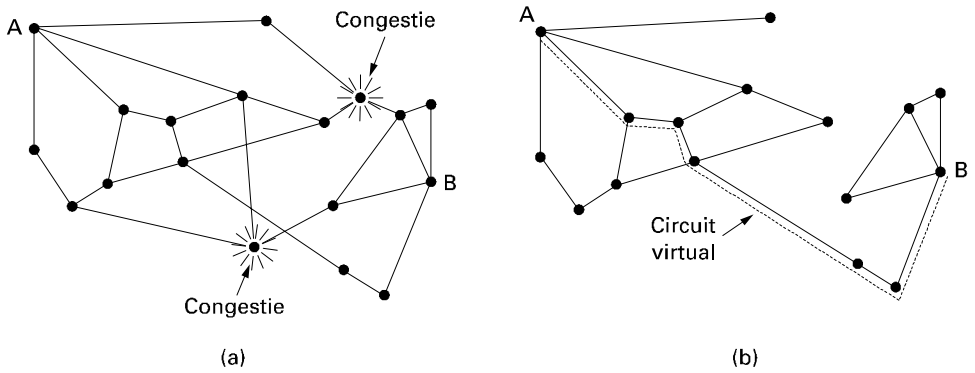


Fig. 5-27. (a) O subrețea congestionată. (b) O subrețea reconfigurată care elimină congestia. Se prezintă și un circuit virtual de la *A* la *B*.

O alternativă este de a permite stabilirea de noi circuite virtuale, dar de a dirija cu atenție aceste noi circuite, ocolind zonele cu probleme. De exemplu, să considerăm subrețeaua din fig. 5-27(a), în care două rutere sunt congestionate, așa cum se poate observa.

Să presupunem că un calculator gazdă atașat ruterului *A* dorește să stabilească o conexiune cu un altul, atașat ruterului *B*. În mod normal, această conexiune ar trece prin unul dintre cele două rutere congestionate. Pentru a evita această situație, putem redesena subrețeaua așa cum se arată în fig. 5-27(b), omițând ruterele congestionate și toate liniile asociate. Linia punctată arată o posibilă cale pentru circuitul virtual, care evită ruterele congestionate.

O altă strategie specifică circuitelor virtuale este negocierea unei înțelegeri între calculatorul gazdă și subrețea la stabilirea unui circuit virtual. Această înțelegere specifică în mod normal volumul și forma traficului, calitatea serviciului cerut și alți parametri. De obicei, pentru a-și respecta partea din înțelegere, subrețeaua își rezervă resurse de-a lungul căii în momentul stabilirii circuitului virtual. Resursele pot include spațiu pentru tabele și zone tampon în rutere și lățime de bandă pe linii. În acest fel, congestia nu prea are șanse să se producă pe noul circuit virtual, deoarece toate resursele sunt garantate a fi disponibile.

Acest tip de rezervare poate fi aplicat tot timpul ca o procedură standard sau doar atunci când rețeaua este congestionată. Dezavantajul încercării de a-l face tot timpul este risipa de resurse. Dacă șase circuite virtuale care pot folosi 1 Mbps trec toate prin aceeași legătură fizică de 6 Mbps, linia trebuie marcată ca plină, chiar dacă este puțin probabil ca toate cele șase circuite să transmită la nivel maxim în același timp. În consecință, prețul controlului congestiei este lățime de bandă nefolosită (adică risipită) în cazuri normale.

5.3.4 Controlul congestiei în subrețele datagramă

Să ne oprim acum asupra unor abordări care pot fi folosite în subrețelele bazate pe datagrame (dar și în cele bazate pe circuite virtuale). Fiecare ruter poate gestiona cu ușurință folosirea liniilor sale de ieșire precum și alte resurse. De exemplu, el poate asocia fiecărei linii o variabilă reală, u, a cărei valoare, între 0.0 și 1.0, reflectă utilizarea recentă a acelei linii. Pentru a menține o estimare cât mai bună a lui u, se poate face periodic eșantionarea utilizării instantanee a liniei f (fie 0, fie 1) și apoi actualizarea lui u astfel:

$$u_{new} = au_{old} + (1 - a)f$$

unde constanta a determină cât de repede uită ruterul istoria recentă.

Ori de câte ori u depășește pragul, linia de ieșire intră într-o stare de „avertisment". Fiecare pachet nou-venit este verificat pentru a se vedea dacă linia de ieșire asociată este în starea de avertisment. Dacă este așa, atunci se iau măsuri. Acțiunea executată poate fi una dintr-o mulțime de alternative, pe care le vom discuta acum.

Bitul de avertizare

Vechea arhitectură DECNET semnala starea de avertizare prin setarea unui bit special în antetul pachetului. La fel procedează și retransmisia cadrelor (frame relay). Când pachetul ajunge la destinația sa, entitatea transport copiază bitul în următoarea confirmare trimisă înapoi la sursă. În acel moment, sursa își reduce traficul.

Atât timp cât ruterul a fost în starea de avertizare, acesta a continuat să seteze bitul de avertizare, ceea ce înseamnă că sursa a primit în continuare confirmări cu acest bit setat. Sursa a monitorizat fracțiunea de confirmări cu bitul setat și și-a ajustat rata de transmisie corespunzător. Atâta timp cât au continuat să sosească biți de avertizare, sursa și-a micșorat continuu rata de transmisie. Odată cu încetinirea sosirii acestora, sursa și-a mărit rata de transmisie. De observat că, deoarece fiecare ruter de-a lungul căi ar fi putut seta bitul de avertizare, traficul a crescut numai atunci când nici unul dintre rutere nu a avut probleme.

Pachete de șoc

Algoritmul anterior pentru controlul congestiei este destul de ingenios. Folosește o modalitate ocolită de a înștiința sursa să încetinească transmisia. De ce nu-i spune direct? În această abordare, ruterul trimite un **pachet de șoc** către calculatorul gazdă sursă, dându-i destinația găsită în pachet. Pachetul original este marcat (un bit din antet este comutat) pentru a nu se mai genera pachete de șoc pe calea aleasă, apoi este retrimis în mod obișnuit.

Un calculator gazdă sursă care primește un pachet de șoc trebuie să reducă traficul trimis spre destinația specificată cu X procente. Deoarece alte pachete trimise către aceeași destinație sunt deja pe drum și vor genera alte pachete de șoc, calculatorul gazdă ar trebui să ignore pachetele de șoc referitoare la destinația respectivă o anumită perioadă de timp. După ce perioada s-a scurs, calculatorul gazdă așteaptă alte pachete de șoc un alt interval. Dacă sosește un astfel de pachet, linia este încă congestionată, astfel încât calculatorul va reduce fluxul și mai mult și va reîncepe să ignore pachetele de șoc. Dacă pe perioada de ascultare nu sosesc pachete de șoc, atunci calculatorul gazdă poate să crească din nou fluxul. Reacția implicită a acestui protocol poate ajuta la prevenirea congestiei, neștrangulând fluxul decât dacă au apărut probleme.

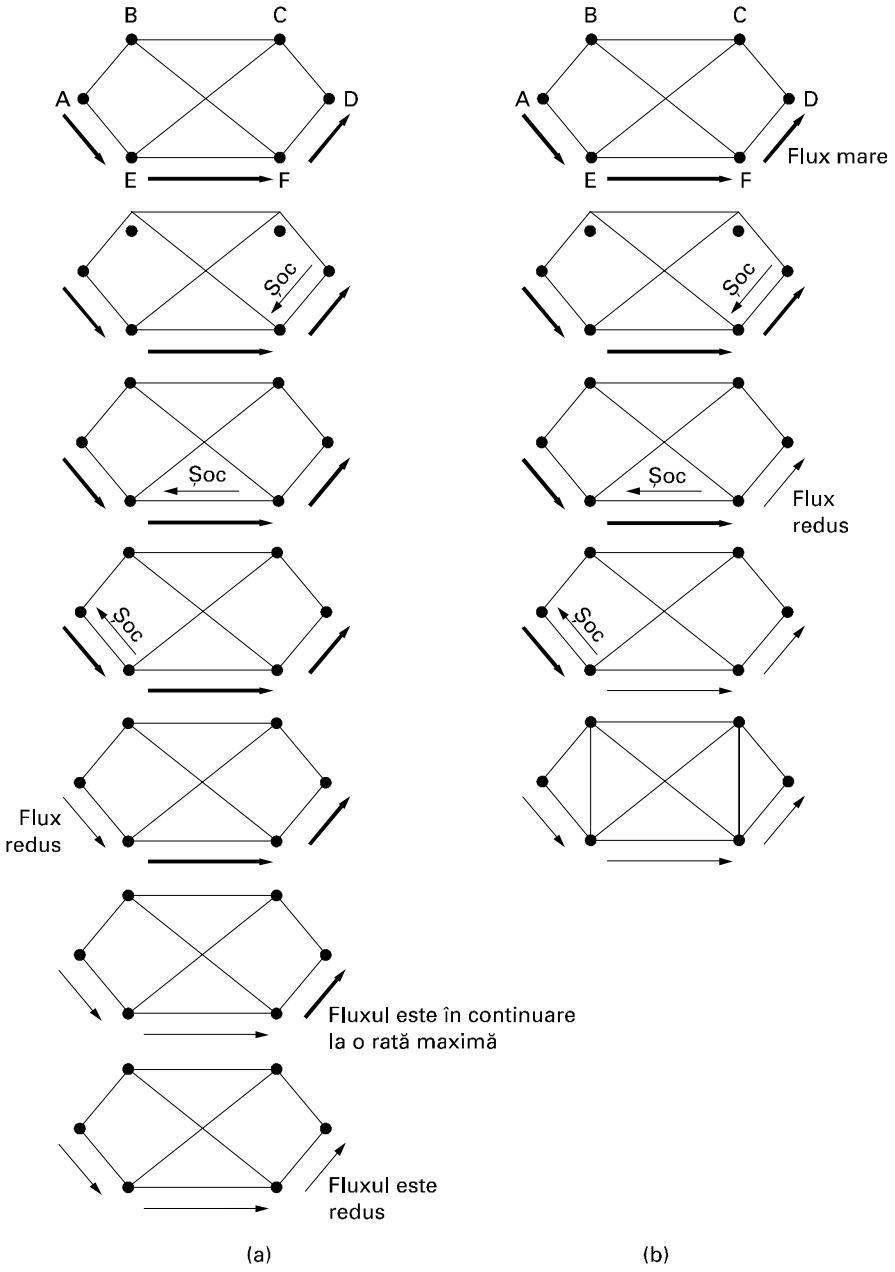


Fig. 5-28. (a) Un pachet de șoc care afectează doar sursa.
(b) Un pachet de șoc care afectează fiecare salt prin care trece.

Calculatorul gazdă poate reduce traficul prin ajustarea parametrilor asociați politicii folosite, de exemplu dimensiunea ferestrei. De obicei, primul pachet de șoc determină scăderea ratei datelor la

0.50 din valoarea anterioară, următoarea reduce traficul la 0.25 și așa mai departe. Creșterea valorilor are loc cu rate mai mici pentru a preveni reinstalarea rapidă a congestiei.

Au fost propuse mai multe variante ale acestui algoritm pentru controlul congestiei. În una dintre acestea, fiecare ruter poate menține mai multe valori de referință (praguri). În funcție de pragul depășit, pachetul de șoc poate conține un avertisment blând, unul sever sau un ultimatum.

Altă variantă prevede folosirea lungimilor cozilor sau a utilizării zonelor tampon în locul utilizării liniilor, ca semnal de comutare. Evident, pentru această metrică se poate folosi aceeași ponderare exponențială ca și pentru u.

Pachete de șoc salt cu salt

La viteze mari sau pe distanțe mari, trimiterea unui pachet de șoc către calculatorul sursă nu funcționează normal, reacția fiind întârziată. Să considerăm, de exemplu, un calculator în San-Francisco (ruterul A din fig. 5-28) care trimite trafic către un calculator din New York (ruterul D din fig. 5-28) la 155 Mbps. Dacă ruterul din New York rămâne fără zone tampon, atunci pachetului de șoc îi vor trebui circa 30 msec pentru ca să revină la San Francisco solicitând încetinirea. Propagarea pachetului de șoc este prezentată ca al doilea, al treilea și al patrulea pas din fig. 5-28(a). În aceste 30 msec, se pot trimite 4.6 MB. Chiar atunci când calculatorul gazdă din San Franciso se oprește imediat, cei 4.6 MB din conductă (eng.: pipe) vor continua să sosească și trebuie luați în seamă. De-abia în a șaptea diagramă din fig. 5-28(a) ruterul din New York va observa reducerea fluxului.

O abordare alternativă este ca pachetele de șoc să aibă efect în fiecare salt prin care trec, așa cum se arată în secvența din fig. 5-28(b). În acest caz, de îndată ce pachetul de șoc ajunge la F, F trebuie să reducă fluxul spre D. Procedând în acest fel, se cere din partea lui F alocarea mai multor zone tampon pentru flux, în timp ce sursa continuă să emită la viteză maximă, însă D simte imediat o ușurare, la fel ca într-o reclamă TV pentru înlăturarea durerii de cap. La pasul următor, pachetul de șoc va ajunge la E, spunându-i lui E să reducă fluxul spre F. Această acțiune solicită puternic zonele tampon ale lui E, însă are un efect benefic imediat asupra lui F, ușurându-i munca. În fine, pachetul de șoc ajunge la A, și fluxul scade efectiv.

Efectul acestei scheme salt cu salt asupra rețelei este asigurarea unei eliberări imediate la locul congestiei cu prețul folosirii mai multor zone tampon. În acest fel congestia poate fi înăbușită imediat ce se manifestă, fără a se pierde nici un pachet. Ideea este discutată în amănunt în (Mishra și Kanakia, 1992), unde se furnizează și rezultatele unei simulări.

5.3.5 Împrăștierea încărcării

Când nici una dintre metodele anterioare nu reduc congestia, ruterele pot să-și scoată la bătaie artileria grea: împrăștierea încărcării. **Împrăștierea încărcării** este un mod simpatic de a spune că atunci când ruterele sunt inundate de pachete pe care nu le mai pot gestiona, pur și simplu le aruncă. Termenul provine din domeniul distribuției energiei electrice, unde se referă la practica serviciilor publice care lasă intenționat fără energie electrică anumite zone, pentru a salva întreaga rețea de la colaps, în zilele călduroase de vară când cererea de energie electrică depășește puternic oferta.

Un ruter care se îneacă cu pachete poate alege la întâmplare pachetele pe care să le arunce, însă de obicei el poate face mai mult decât atât. Alegerea pachetelor care vor fi aruncate poate depinde de aplicațiile care rulează. Pentru transferul de fișiere, un pachet vechi este mai valoros decât unul nou, deoarece aruncarea pachetului 6 și păstrarea pachetelor de la 7 la 10 va cauza producerea unei „spărturi” în fereastra receptorului, care poate forța retrimiterea

pachetelor de la 6 la 10 (dacă receptorul distruge automat pachetele care sunt în afara secvenței). În cazul unui fișier de 12 pachete, aruncarea pachetului 6 poate necesita retransmiterea pachetelor de la 7 la 12, în timp ce aruncarea lui 10 va necesita doar retrimiterea pachetelor de la 10 la 12. În contrast, pentru multimedia, un pachet nou este mult mai important decât un pachet vechi. Prima politică (vechiul este mai bun decât noul) este numită adesea **a vinului**, iar ultima (noul este mai bun decât vechiul) este numită **a laptelui**.

Un pas inteligent mai departe presupune cooperare din partea expeditorilor. Pentru multe aplicații, unele pachete sunt mai importante decât altele. De exemplu, unii algoritmi de compresie a imaginilor transmit periodic un cadru întreg și apoi trimit următoarele cadre ca diferențe față de ultimul cadru complet. În acest caz, aruncarea unui pachet care face parte dintr-o diferență este de preferat aruncării unuia care face parte dintr-un cadru întreg. Ca un alt exemplu, să considerăm trimiterea documentelor care conțin text ASCII și imagini. Pierderea unei linii de pixeli dintr-o imagine este de departe mai puțin dăunătoare decât pierderea unei linii dintr-un text.

Pentru a implementa o politică inteligentă de distrugere a pachetelor, aplicațiile trebuie să înscrie pe fiecare pachet clasa de prioritate din care face parte, pentru a indica cât de important este. Dacă ele fac aceasta, atunci când trebuie distruse unele pachete, ruterele vor începe cu cele din clasa cea mai slabă, vor continua cu cele din următoarea clasă și așa mai departe. Evident, dacă nu ar fi nici un stimulent pentru a marca pachetele și altfel decât FOARTE IMPORTANT - A NU SE DISTRUGE NICIODATĂ, SUB NICI UN MOTIV, nimeni nu ar face acest lucru.

Stimulentul ar putea fi sub formă de bani, adică pachetele mai puțin prioritare să fie mai ieftin de trimis decât cele cu prioritate mare. Ca alternativă, expeditorilor li s-ar putea permite să trimită pachete cu priorități mari cu condiția ca încărcarea să fie mică, dar odată cu creșterea încărcării acestea ar fi aruncate, încurajând astfel utilizatorii să înceteze trimiterea lor.

O altă opțiune ar fi să se permită calculatoarelor gazdă să depășească limitele specificate în contractul negociat la inițializarea circuitului virtual (să folosească o lățime de bandă mai mare decât li s-a permis), dar cu condiția ca tot traficul în exces să fie marcat ca fiind de prioritate scăzută. O astfel de strategie nu este de fapt o idee rea, pentru că utilizează în mod mai eficient resursele mai puțin folosite, permițând astfel calculatoarelor gazdă să le utilizeze atâta timp cât nimeni altcineva nu este interesat, dar fără să stabilească dreptul la ele atunci când situația devine mai dură.

Detecția aleatoare timpurie

Este bine cunoscut faptul că a trata congestia imediat ce a fost detectată este mai eficient decât să o lași să-ți încurce treburile și apoi să încerci să te descurci cu ea. Această observație a dus la ideea de a arunca pachete înainte ca toată zona tampon să fie într-adevăr epuizată. Un algoritm popular pentru a face acest lucru se numește **RED** (**Random Early Detection** – rom.: Detecția aleatoare timpurie) (Floyd și Jacobson, 1993). În unele protocoale de transport (incluzând TCP-ul), răspunsul la pachetele pierdute este ca sursa să încetinească. Raționamentul din spatele acestei logici este acela că TCP a fost proiectat pentru rețele cablate, iar acestea sunt foarte sigure, astfel încât pachetele pierdute se datorează mai degrabă depășirii spațiului tampon decât erorilor de transmisie. Acest lucru poate fi exploatat pentru a ajuta la reducerea congestiei.

Prin aruncarea pachetelor de către rutere înainte ca situația să devină fără speranță (de unde și termenul “timpuriu” din denumire), ideea este de a lua măsuri înainte de a fi prea târziu. Pentru a determina când să înceapă aruncarea, ruterele mențin neîntrerupt o medie a lungimilor cozilor lor. Când lungimea medie a cozii unei linii depășește o limită, se spune că linia este congestionată și se iau măsuri.

Având în vedere că ruterul probabil că nu poate spune care sursă produce necazul cel mai mare, alegerea la întâmplare a unui pachet din coada care a determinat această acțiune este probabil cel mai bun lucru pe care îl poate face.

Cum ar trebui ruterul să informeze sursa despre problemă? O cale este aceea de a-i trimite un pachet de șoc, după cum am descris. O problemă a acestei abordări este că încarcă și mai mult rețeaua deja congestionată. O strategie diferită este să arunce pur și simplu pachetul selectat și să nu raporteze acest lucru. Sursa va observa în cele din urmă lipsa confirmării și va lua măsuri. Deoarece știe că pachetele pierdute sunt în general determinate de congestie și aruncări, va reacționa prin încetinire în loc să încerce mai abitir. Această formă implicită de reacție funcționează doar atunci când sursele răspund la pachetele pierdute prin reducerea ratei lor de transmisie. În rețele fără fir, unde majoritatea pierderilor se datorează zgomotelor legăturii aeriene, această abordare nu poate fi folosită.

5.3.6 Controlul fluctuațiilor

Pentru aplicații de genul transmisiilor audio sau video, nu prea contează dacă pachetele au nevoie de 20 msec sau de 30 msec pentru a fi distribuite, atât timp cât durata de tranzit este constantă. Variația (adică deviația standard) timpului de sosire a pachetului se numește **fluctuație.** O fluctuație mare, de exemplu dacă unele pachete au nevoie de 20 msec și altele de 30 msec pentru a ajunge, va produce sunet sau imagine de calitate inegală. Fluctuația este ilustrată în fig. 5-29. În contrast, o înțelegere ca 99 procente din pachete să fie livrate cu o întârziere de la 24.5 msec până la 25.5 msec ar putea fi acceptabilă.

Limita aleasă trebuie să fie, bineînțeles, reală. Ea trebuie să ia în calcul durata de tranzit a vitezei luminii și întârzierea minimă prin rutere și poate să-și lase o mică rezervă pentru unele întârzieri inevitabile.

Fluctuațiile pot fi limitate prin calcularea timpului de tranzit estimat pentru fiecare salt de-a lungul căii. Când un pachet ajunge la ruter, ruterul verifică să vadă cât de mult este decalat pachetul față de planificarea sa. Această informație este păstrată în pachet și actualizată la fiecare salt. Dacă pachetul a sosit înaintea planificării, el este ținut suficient pentru a reveni în grafic. Dacă pachetul este întârziat față de planificare, ruterul încearcă să-l trimită cât mai repede.

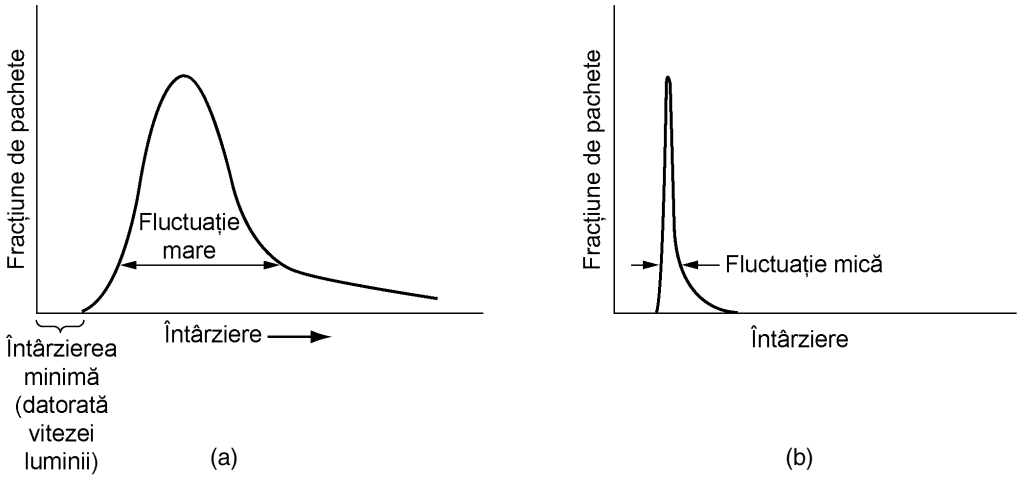


Fig. 5-29. (a) Fluctuație mare. (b) Fluctuație scăzută.

De fapt, algoritmul pentru determinarea pachetului care va merge mai departe dintr-un set de pachete care concurează pentru o linie de ieșire, va alege întotdeauna pachetul cu cea mai mare întârziere față de program. În acest fel, pachetele care au ajuns mai repede sunt încetinite, iar cele care sunt întârziate vor fi accelerate, în ambele cazuri reducându-se fluctuația.

În unele aplicații, cum ar fi aplicații video la cerere, fluctuațiile pot fi eliminate prin adăugarea unei zone tampon la receptor și apoi aducerea datelor pentru a fi afișate din zona tampon în loc de a fi luate din rețea în timp real. Totuși, pentru alte aplicații, în special acelea care necesită interacțiune în timp real între oameni, cum ar fi convorbirile telefonice și videoconferințele prin Internet, întârzierea inerentă utilizării zonelor tampon este inacceptabilă. Controlul congestiei este o arie activă a cercetării. Nivelul actual este rezumat în (Gevros et al., 2001).

5.4 CALITATEA SERVICIILOR

Tehnicile pe care le-am analizat în secțiunile precedente sunt proiectate pentru a reduce congestia și a îmbunătăți performanțele rețelei. Cu toate acestea, odată cu creșterea lucrului cu aplicații multimedia în rețea, de multe ori aceste măsuri luate ad-hoc nu sunt de ajuns. Este nevoie de preocupări serioase în proiectarea rețelelor și a protocoalelor pentru garantarea calității serviciilor. În secțiunile următoare ne vom continua studiul asupra performanței rețelei, dar acum ne vom concentra mai mult asupra modalităților de a furniza o calitate a serviciilor în conformitate cu necesitățile fiecărei aplicații. Ar trebui stipulat de la început faptul că, totuși, multe dintre aceste idei sunt în formare și sunt supuse schimbărilor.

5.4.1 Cerințe

Un șir de pachete de la o sursă la o destinație se numește **flux**. Într-o rețea orientată pe conexiune, toate pachetele aparținând unui flux merg pe aceeași rută; într-o rețea neorientată pe conexiune, acestea pot urma rute diferite. Necesitățile fiecărui flux pot fi caracterizate prin patru parametri primari: fiabilitatea, întârzierea, fluctuația și lățimea de bandă. Împreună, acestea determină **QoS** (**Quality of Service** – rom.: Calitatea serviciilor) pe care o necesită fluxul. În fig. 5-30 sunt listate câteva aplicații obișnuite precum și cerințele stringente ale acestora.

Aplicația	Fiabilitatea	Întârzierea	Fluctuația	Lățimea de bandă
Poșta electronică	Mare	Mică	Mică	Mică
Transfer de fișiere	Mare	Mică	Mică	Medie
Acces Web	Mare	Medie	Mică	Medie
Conectare la distanță	Mare	Medie	Medie	Mică
Audio la cerere	Mică	Mică	Mare	Medie
Video la cerere	Mică	Mică	Mare	Mare
Telefonie	Mică	Mare	Mare	Mică
Videoconferințe	Mică	Mare	Mare	Mare

Fig. 5-30. Cât de stringente sunt cerințele referitoare la calitatea serviciilor

Primele patru aplicații au cerințe stringente de fiabilitate. Nu se accepta livrarea de biți incorecți. Acest obiectiv este atins de obicei prin crearea unei sume de control pentru fiecare pachet și verifica-

rea acestei sume de control la destinație. Dacă un pachet este alterat pe parcurs, nu este confirmat și în cele din urmă va fi retransmis. Această strategie oferă o fiabilitate mare. Ultimele patru aplicații (audio/video) pot tolera erori, așa că nu este calculată sau verificată nici o sumă de control.

Aplicațiile de transfer de fișiere, inclusiv poșta electronică și video nu sunt sensibile la întârzieri. Dacă toate pachetele sunt întârziate uniform cu câteva secunde, nu este nici o problemă. Aplicațiile interactive, cum ar fi navigarea pe Web și conectarea la distanță, sunt mult mai sensibile la întârzieri. Aplicațiile de timp real, cum ar fi telefonia și videoconferințele au cerințe stricte de întârziere. Dacă toate cuvintele dintr-o convorbire telefonică sunt fiecare întârziate cu 2.000 secunde, utilizatorii vor găsi conexiunea inacceptabilă. Pe de altă parte, rularea de fișiere audio sau video de pe server nu cere întârzieri scăzute.

Primele trei aplicații nu sunt sensibile la sosirea pachetelor la intervale neregulate. Conectarea la distanță este într-un fel sensibilă la aceasta deoarece caracterele vor apărea pe ecran în rafale dacă conexiunea are multe fluctuații. Aplicațiile video și în special cele audio sunt extrem de sensibile la fluctuații. Dacă un utilizator urmărește un fișier video din rețea și cadrele sunt toate întârziate cu exact 2.000 secunde, nu este nici o problemă. Dar dacă timpul de transmisie variază aleator între 1 și 2 secunde, rezultatul va fi îngrozitor. Pentru audio, o fluctuație chiar și de câteva milisecunde este perfect sesizabilă.

În cele din urmă, aplicațiile diferă la cerințele de lățime de bandă, fără ca poșta electronică și conectarea la distanță să necesite o bandă prea largă, iar aplicațiile video de orice fel necesitând o bandă foarte largă. Rețelele ATM clasifică fluxul în patru mari categorii, în funcție de cerințele lor de QoS, după cum urmează:

1. Rată de transfer constantă (de exemplu telefonie).
2. Rată de transfer variabilă în timp real (de exemplu videoconferențiere compresată).
3. Rată de transfer variabilă, dar nu în timp real (de exemplu vizionarea unui film pe Internet).
4. Rată de transfer disponibilă (de exemplu transfer de fișiere).

Aceste categorii sunt de asemenea folositoare pentru alte scopuri și pentru alte tipuri de rețele. Rata de transfer constantă este o încercare de a simula un cablu, furnizând o lățime de bandă uniformă și o întârziere uniformă. Rata de transfer variabilă apare pentru video compresat, unele cadre fiind mai comprimate decât altele. Astfel, transmisia unui cadru cu o mulțime de detalii în el ar putea necesita transmisia mai multor biți, în timp ce transmisia imaginii unui perete alb s-ar putea comprima extrem de bine. Rata de transfer disponibilă este pentru aplicații care nu sunt sensibile la întârzieri sau la fluctuații, cum ar fi poșta electronică.

5.4.2 Tehnici pentru obținerea unei bune calități a serviciilor

Acum că știm câte ceva despre cerințele QoS, cum le obținem? Ei bine, pentru început, nu există o rețetă magică. Nici o tehnică nu furnizează într-un mod optim o calitate a serviciilor eficientă și pe care să te poți baza. În schimb, au fost dezvoltate o varietate de tehnici, cu soluții practice care adeseori combină mai multe tehnici. Vom examina acum câteva dintre tehnicile folosite de proiectanții de sisteme pentru obținerea calității serviciilor.

Supraaprovizionarea

O soluție ușoară este aceea de a furniza ruterului suficientă capacitate, spațiu tampon și lățime de bandă încât pachetele să zboare pur și simplu. Problema cu această soluție este aceea că este costisi-

toare. Odată cu trecerea timpului și cu faptul că proiectanții au o idee mai clară despre cât de mult înseamnă suficient, această tehnică ar putea deveni chiar realistă. Până la un punct, sistemul telefonic este supraapovizionat Se întâmplă rar să ridici receptorul telefonului și să nu ai ton de formare instantaneu. Pur și simplu există atâta capacitate disponibilă încât cererea va fi întotdeauna satisfăcută.

Memorarea temporară

Fluxurile pot fi reținute în zone tampon ale receptorului înainte de a fi livrate. Aceasta nu afectează fiabilitatea sau lățimea de bandă și mărește întârzierea, dar uniformizează fluctuația. Pentru audio și video la cerere, fluctuațiile reprezintă problema principală, iar această tehnică este de mare ajutor.

Am văzut diferența dintre fluctuațiile mari și fluctuațiile mici în fig. 5-29. În fig. 5-31 vedem un flux de pachete care sunt livrate cu o fluctuație considerabilă. Pachetul 1 este transmis de către server la momentul $t = 0$ sec și ajunge la client la $t = 1$ sec. Pachetul 2 acumulează o întârziere mai mare și îi sunt necesare 2 sec pentru a ajunge. Pe măsură ce pachetele sosesc, ele sunt reținute în zona tampon pe mașina client.

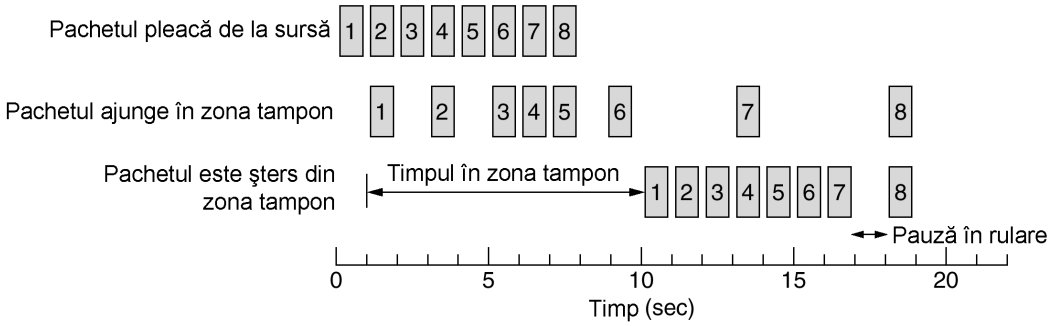


Fig. 5-31. Uniformizarea fluxului de ieșire prin memorarea temporară a pachetelor.

La $t = 10$ sec începe rularea. La acest moment, pachetele de la 1 la 6 au fost introduse în zona tampon așa că pot fi scoase de acolo la intervale egale pentru o rulare uniformă. Din păcate, pachetul 8 a avut o întârziere așa de mare încât nu este disponibil când îi vine timpul de rulare, așa că rularea trebuie întreruptă până la sosirea acestuia, creând astfel o pauză supărătoare în muzică sau în film. Această problemă poate fi alinată prin întârzierea și mai mare a momentului începerii, deși acest lucru necesită și un spațiu tampon mai mare. Sit-urile Web comerciale care conțin fluxuri audio sau video folosesc toate programe de rulare care au timpul de reținere în zona tampon de aproximativ 10 secunde până să înceapă să ruleze aplicația.

Formarea traficului

În exemplul de mai sus, sursa trimite pachetele la intervale de timp egale între ele, dar în alte cazuri ele pot fi emise neregulat, ceea ce poate duce la apariția congestiei în rețea. Neuniformitatea ieșirii este un lucru obișnuit dacă server-ul se ocupă de mai multe fluxuri la un moment dat și de asemeni permite și alte acțiuni cum ar fi derularea rapidă înainte sau înapoi, autentificarea utilizatorilor și așa mai departe. De asemenea, abordarea pe care am folosit-o aici (memorarea temporară) nu este întotdeauna posibilă, de exemplu în cazul videoconferințelor. Cu toate acestea, dacă s-ar putea face ceva pentru ca serverul (și calculatoarele gazdă în general) să transmită cu rate de transfer uniforme, calitatea serviciilor ar fi mai bună. Vom examina acum o tehnică numită **formarea traficului (traffic shaping)**, care uniformizează traficul mai degrabă pentru server decât pentru client.

Formarea traficului se ocupă cu uniformizarea ratei medii de transmisie a datelor (atenuarea rafalelor). În contrast, protocoalele cu fereastră glisantă pe care le-am studiat anterior limitează volumul de date în tranzit la un moment dat și nu rata la care sunt transmise acestea. La momentul stabilirii unei conexiuni, utilizatorul și subrețeaua (clientul și furnizorul) stabilesc un anumit model al traficului (formă) pentru acel circuit. În unele cazuri aceasta se numește **înțelegere la nivelul serviciilor (service level agreement)**. Atât timp cât clientul își respectă partea sa de contract și trimite pachete conform înțelegerii încheiate, furnizorul promite livrarea lor în timp util. Formarea traficului reduce congestia și ajută furnizorul să-și țină promisiunea. Astfel de înțelegeri nu sunt foarte importante pentru transferul de fișiere, însă sunt deosebit de importante pentru datele în timp real, cum ar fi conexiunile audio sau video, care au cerințe stringente de calitate a serviciilor.

Pentru formarea traficului clientul spune furnizorului: „Modelul meu de transmisie arată cam așa. Poți să te descurci cu el?" Dacă furnizorul este de acord, problema care apare este cum poate spune furnizorul dacă clientul respectă înțelegerea și ce să facă dacă nu o respectă. Monitorizarea fluxului traficului se numește **supravegherea traficului (traffic policing**.) Stabilirea unei forme a traficului și urmărirea respectării ei se fac mai ușor în cazul subrețelelor bazate pe circuite virtuale decât în cazul subrețelelor bazate pe datagrame. Cu toate acestea, chiar și în cazul subrețelelor bazate pe datagrame, aceleași idei pot fi aplicate la conexiunile nivelului transport.

Algoritmul găleții găurite

Să ne imaginăm o găleată cu un mic orificiu în fundul său, așa cum este prezentată în fig. 5-32(a). Nu contează cu ce rată curge apa în găleată, fluxul de ieșire va fi la o rată constantă, ρ, dacă există apă în găleată și zero dacă găleata este goală. De asemenea, odată ce găleata s-a umplut, orice cantitate suplimentară de apă se va revărsa în afara pereților și va fi pierdută (adică nu se va regăsi în fluxul de ieșire de sub orificiu).

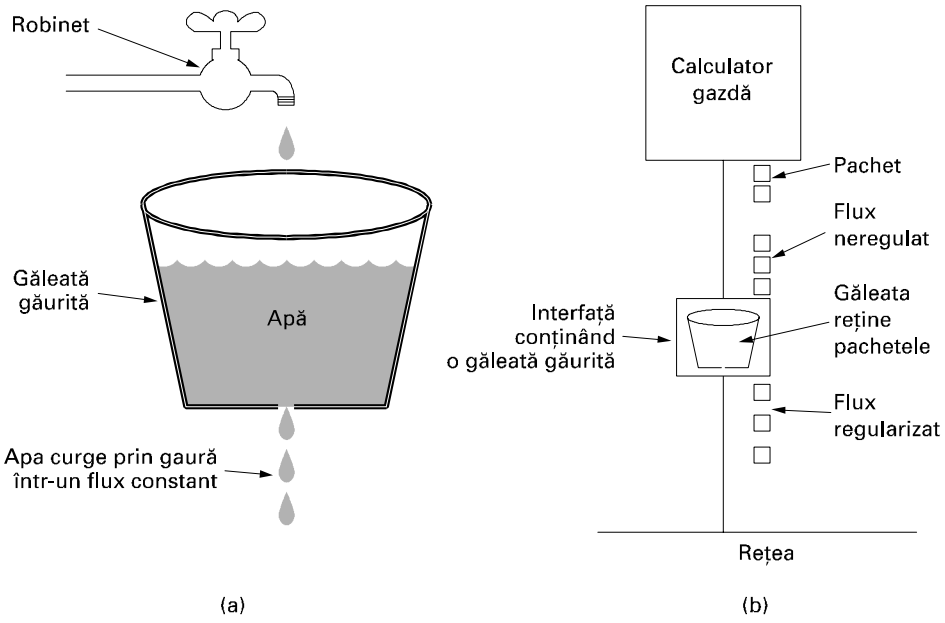


Fig. 5-32. (a) O găleată găurită umplută cu apă (b) O găleată găurită, cu pachete.

Aceeași idee poate fi aplicată și în cazul pachetelor, așa cum se arată în fig. 5-32 (b). Conceptual, fiecare calculator gazdă este conectat la rețea printr-o interfață conținând o găleată găurită, în fapt o coadă internă cu capacitate finită. Dacă un pachet sosește în coadă atunci când aceasta este plină, el este distrus. Cu alte cuvinte, dacă unul sau mai multe procese de pe calculatorul gazdă încearcă trimiterea unui pachet atunci când coada conține deja numărul maxim de pachete, pachetele noi vor fi distruse fără menajamente. Acest aranjament poate fi implementat în interfața hardware sau poate fi simulat de către sistemul de operare gazdă. A fost propus pentru prima dată de către Turner (1986) și este numit **algoritmul găleții găurite (the leaky bucket algorithm)**. De fapt nu este altceva decât un sistem de cozi cu un singur server și cu timp de servire constant.

Calculatorul gazdă poate pune în rețea câte un pachet la fiecare tact al ceasului. Și acest lucru poate fi realizat de placa de interfață sau de către sistemul de operare. Acest mecanism transformă un flux neregulat de pachete de la procesele de pe calculatorul gazdă într-un flux uniform de pachete care se depun pe rețea, netezind rafalele și reducând mult șansele de producere a congestiei.

Dacă toate pachetele au aceeași dimensiune (de exemplu celule ATM), algoritmul poate fi folosit exact așa cum a fost descris. Dacă se folosesc pachete de lungimi variabile, este adesea mai convenabil să se transmită un anumit număr de octeți la fiecare tact și nu un singur pachet. Astfel, dacă regula este 1024 octeți la fiecare tact, atunci se pot transmite un pachet de 1024 octeți, două de 512 octeți, sau patru de 256 octeți ș.a.m.d. Dacă numărul rezidual de octeți este prea mic, următorul pachet va trebui să aștepte următorul tact.

Implementarea algoritmului inițial al găleții găurite este simplă. Găleata găurită constă de fapt dintr-o coadă finită. Dacă la sosirea unui pachet este loc în coadă, el este adăugat la sfârșitul cozii, în caz contrar, este distrus. La fiecare tact se trimite un pachet din coadă (bineînțeles dacă aceasta nu este vidă).

Algoritmul găleții folosind contorizarea octeților este implementat aproximativ în aceeași manieră. La fiecare tact un contor este inițializat la n. Dacă primul pachet din coadă are mai puțini octeți decât valoarea curentă a contorului, el este transmis și contorul este decrementat cu numărul corespunzător de octeți. Mai pot fi transmise și alte pachete adiționale, atât timp cât contorul este suficient de mare. Dacă contorul scade sub lungimea primului pachet din coadă, atunci transmisia încetează până la următorul tact, când contorul este reinițializat și fluxul poate continua.

Ca un exemplu de găleată găurită, să ne imaginăm un calculator care produce date cu rata de 25 milioane octeți/sec (200 Mbps) și o rețea care funcționează la aceeași viteză. Cu toate acestea, ruterele pot să gestioneze datele la această rată doar pentru un timp foarte scurt (practic, până când li se umple spațiul tampon). Pentru intervale mai mari ele lucrează optim pentru rate care nu depășesc valoarea de 2 milioane octeți/sec. Să presupunem acum că datele vin în rafale de câte 1 milion de octeți, câte o rafală de 40 msec în fiecare secundă. Pentru a reduce rata medie la 2 MB/sec, putem folosi o găleată găurită având $\rho = 2$ MB/sec și o capacitate, C, de 1 MB. Aceasta înseamnă că rafalele de până la 1 MB pot fi gestionate fără pierderi de date și că acele rafale sunt împrăștiate de-a lungul a 500 msec, indiferent cât de repede sosesc.

În fig. 5-33(a) vedem intrarea pentru găleata găurită funcționând la 25 MB/sec pentru 40 msec. În fig. 5-33(b) vedem ieșirea curgând la o rată uniformă de 2 MB/sec pentru 500 msec.

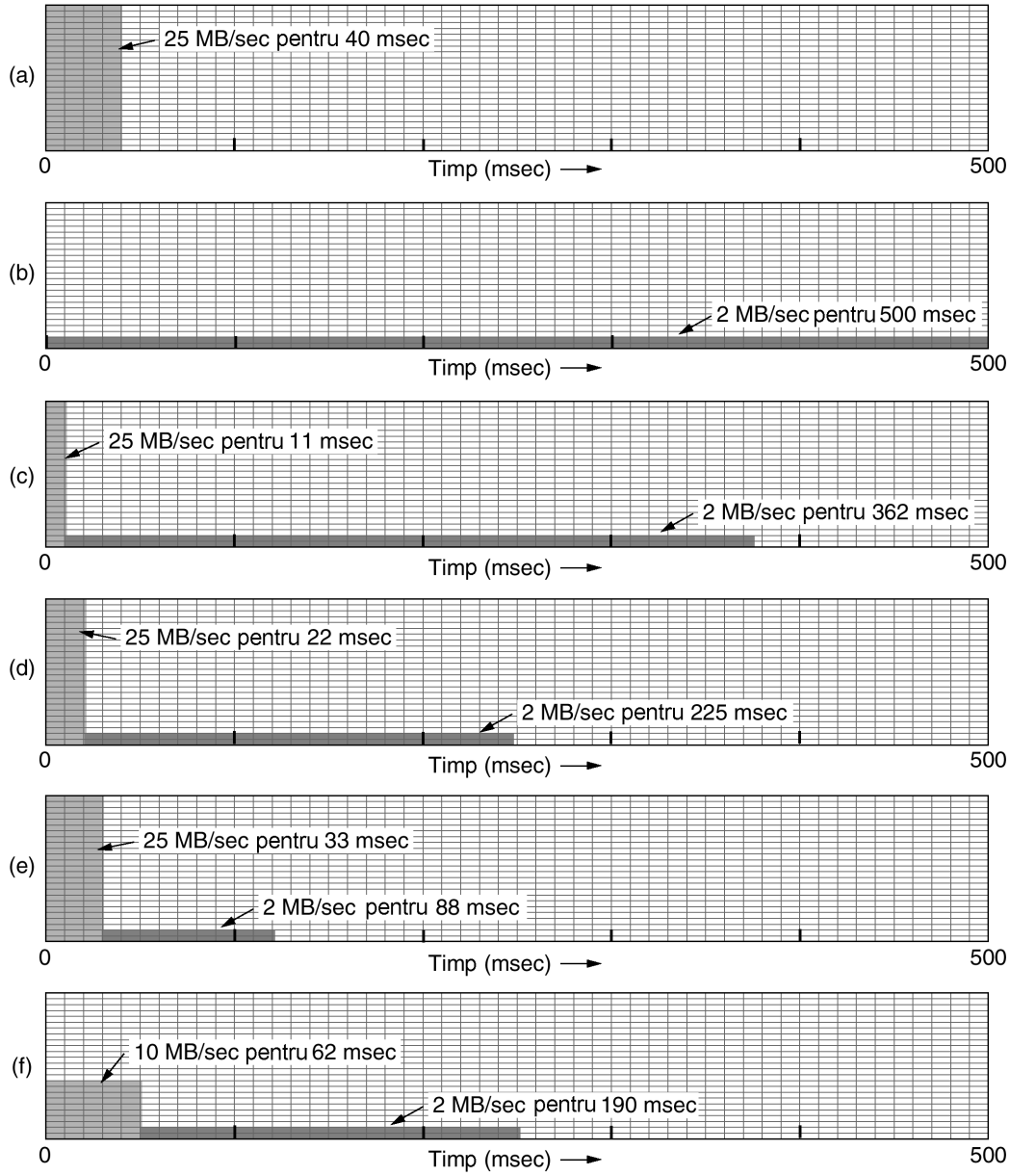


Fig. 5-33. (a) Intrarea pentru o găleată găurită. (b) Ieșirea pentru o găleată găurită. (c) - (e) Ieșirea pentru o găleată cu jeton de capacitate 250 KB, 500 KB, 750 KB. (f) Ieșirea pentru o găleată cu jeton de capacitate 500 KB care alimentează o găleată găurită de 10 MB/sec.

Algoritmul găleții cu jeton

Algoritmul găleții găurite impune un model rigid al ieșirii, din punct de vedere al ratei medii, indiferent de cum arată traficul. Pentru numeroase aplicații este mai convenabil să se permită o creștere a vitezei de ieșire la apariția unor rafale mari, astfel încât este necesar un algoritm mai flexibil, de preferat unul care nu pierde date. Un astfel de algoritm este **algoritmul găleții cu jeton (the token bucket algorithm)**. În acest algoritm, găleata găurită păstrează jetoane, generate de un ceas cu rata de un jeton la fiecare ΔT sec. În fig. 5-34(a) vedem o găleată păstrând trei jetoane și având cinci pachete care așteaptă să fie transmise. Pentru ca un pachet să fie transmis, el trebuie să captureze și să distrugă un jeton. În fig. 5-34(b) vedem că trei din cele cinci pachete au trecut mai departe, în timp ce celelalte două așteaptă să fie generate alte două jetoane.

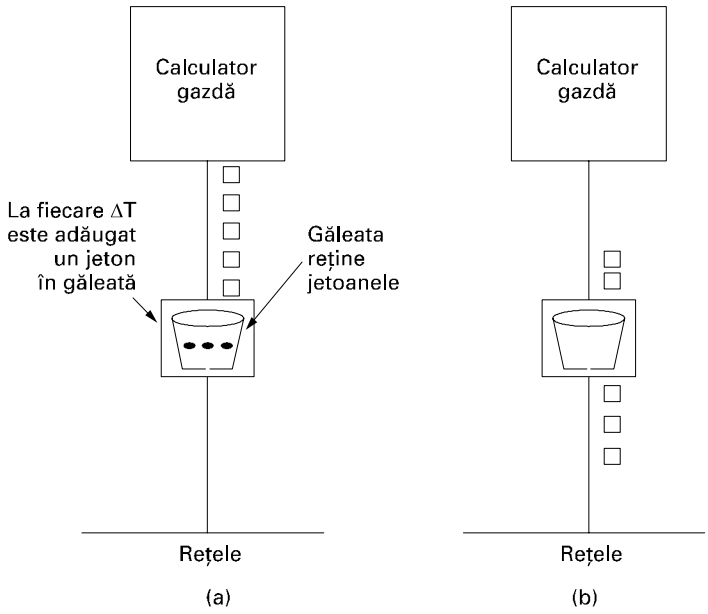


Fig. 5-34. Algoritmul găleții cu jeton. (a) Înainte. (b) După.

Algoritmul găleții cu jeton asigură o formare diferită a traficului, comparativ cu algoritmul găleții găurite. Algoritmul găleții găurite nu permite calculatoarelor gazdă inactive să acumuleze permisiunea de a trimite rafale mari ulterior. Algoritmul găleții cu jeton permite această acumulare, mergând până la dimensiunea maximă a găleții, n. Această proprietate permite ca rafale de până la n pachete să fie trimise dintr-o dată, permițând apariția unor rafale la ieșire și asigurând un răspuns mai rapid la apariția bruscă a unor rafale la intrare.

O altă diferență între cei doi algoritmi este aceea că algoritmul găleții cu jeton aruncă jetoanele (capacitatea de transmisie) la umplerea găleții, dar niciodată nu distruge pachete. Prin contrast, algoritmul găleții găurite distruge pachete la umplerea găleții.

Și aici este posibilă o variantă, în care fiecare jeton reprezintă dreptul de a trimite nu un pachet, ci k octeți. Un pachet va putea fi trimis doar dacă se dețin suficiente jetoane pentru a-i acoperi lungimea exprimată în octeți. Jetoanele fracționare sunt păstrate pentru utilizări ulterioare.

Algoritmul găleții găurite și algoritmul găleții cu jeton pot fi folosite și pentru a uniformiza traficul între rutere, la fel de bine cum pot fi folosite pentru a regla informația de ieșire a unui calculator

gazdă, ca în exemplul prezentat. Oricum, o diferență evidentă este aceea că algoritmul găleții cu jeton folosit pentru a regla ieșirea unui calculator gazdă îl poate opri pe acesta să trimită date atunci când apare vreo restricție. A spune unui ruter să oprească transmisiile în timp ce datele de intrare continuă să sosească, poate duce la pierderea de date.

Implementarea de bază a algoritmului găleții cu jeton se face printr-o variabilă care numără jetoane. Acest contor crește cu 1 la fiecare ΔT și scade cu 1 ori de câte ori este trimis un pachet. Dacă acest contor atinge valoarea zero, nu mai poate fi trimis nici un pachet. În varianta la nivel de octeți, contorul este incrementat cu k octeți la fiecare ΔT și decrementat cu lungimea pachetului trimis.

Caracteristica esențială a algoritmului găleții cu jeton este că permite rafalele, dar până la o lungime maximă controlată. Priviți de exemplu fig. 5-33(c). Aici avem o găleată cu jeton, de capacitate 250 KB. Jetoanele sosesc cu o rată care asigură o ieșire de 2 MB/sec. Presupunând că găleata este plină când apare o rafală de 1 MB, ea poate asigura scurgerea la întreaga capacitate de 25 MB/sec pentru circa 11 milisecunde. Apoi trebuie să revină la 2 MB/sec până ce este trimisă întreaga rafală.

Calculul lungimii rafalei de viteză maximă este puțin mai special. Nu este doar 1 MB împărțit la 25 MB/sec deoarece, în timp ce rafala este prelucrată, apar alte jetoane. Dacă notăm S cu lungimea rafalei în secunde, cu C capacitatea găleții în octeți, cu ρ rata de sosire a jetoanelor în octeți/secundă, cu M rata maximă de ieșire în octeți/secundă, vom vedea că o rafală de ieșire conține cel mult $C + \rho S$ octeți. De asemenea, mai știm că numărul de octeți într-o rafală de viteză maximă de S secunde este MS. De aici avem:

$$C + \rho S = MS$$

Rezolvând această ecuație obținem $S = C / (M - \rho)$. Pentru parametrii considerați $C = 250$ KB, $M = 25$ MB/sec și $\rho = 2$ MB/sec vom obține o durată a rafalei de circa 11 msec. Fig. 5-33(d) și fig. 5-33(e) prezintă găletile cu jeton de capacități 500 KB și respectiv 750 KB.

O problemă potențială a algoritmului găleții cu jeton este aceea că și el permite apariția unor rafale mari, chiar dacă durata maximă a unei rafale poate fi reglată prin selectarea atentă a lui ρ și M. De multe ori este de dorit să se reducă valoarea de vârf, dar fără a se reveni la valorile scăzute permise de algoritmul original, al găleții găurite.

O altă cale de a obține un trafic mai uniform este de a pune o găleată găurită după cea cu jeton. Rata găleții găurite va trebui să fie mai mare decât parametrul ρ al găleții cu jeton, însă mai mică decât rata maximă a rețelei. Fig. 5-25(f) ilustrează comportarea unei găleți cu jeton de 500 KB, urmată de o găleată găurită de 10 MB/sec.

Gestionarea tuturor acestor scheme poate fi puțin mai specială. În esență, rețeaua trebuie să simuleze algoritmul și să se asigure că nu se trimit mai multe pachete sau mai mulți octeți decât este permis. Totuși, aceste instrumente furnizează posibilități de a aduce traficul rețelei la forme mai ușor de administrat pentru a ajuta la îndeplinirea condițiilor necesare calității serviciilor.

Rezervarea resurselor

Un bun început pentru garantarea calității serviciilor este capabilitatea de a regla forma traficului oferit. Totuși, folosirea efectivă a acestei informații înseamnă implicit să ceri ca toate pachetele dintr-un flux să urmeze aceeași rută. Răspândirea lor aleatoare pe rutere face dificil de garantat orice. Ca o consecință, trebuie ca între sursă și destinație să se creeze ceva similar unui circuit virtual și ca toate pachetele care aparțin fluxului să urmeze această cale.

Odată ce avem o rută specifică pentru un flux, devine posibil să rezervi resurse de-a lungul acelei rute pentru a te asigura că este disponibilă capacitatea necesară. Pot fi rezervate trei tipuri diferite de resurse:

1. Lățimea de bandă
2. Zona tampon
3. Ciclurile procesorului

Prima, lățimea de bandă este cea mai evidentă. Dacă un flux cere 1 Mbps și linia pe care se iese are capacitatea de 2 Mbps, încercarea de a dirija trei fluxuri prin acea linie nu va reuși. Astfel, a rezerva lățime de bandă înseamnă să nu se supraîncarce vreo linie de ieșire.

O a doua resursă care este adeseori insuficientă este spațiul tampon. Când sosește un pachet, acesta este de obicei depozitat pe placa de rețea de către hardware. Software-ul ruterului trebuie să copieze apoi pachetul într-o zonă tampon din RAM și să adauge acest tampon în coada pentru transmisie pe linia de ieșire aleasă. Dacă nu este disponibil nici un tampon, pachetul trebuie aruncat deoarece nu există spațiu în care să poată fi pus. Pentru o bună calitate a serviciilor, unele zone tampon pot fi rezervate pentru un anumit flux în așa fel încât acel flux să nu trebuiască să concureze pentru tampoane cu alte fluxuri. Întotdeauna va exista o zonă tampon disponibilă atunci când fluxul va avea nevoie, dar până la o anumită limită.

În cele din urmă, ciclurile procesorului sunt de asemenea resurse rare. Ruterul are nevoie de timp de procesor pentru a prelucra un pachet, deci un ruter poate procesa doar un anumit număr de pachete pe secundă. Este necesar să ne asigurăm că procesorul nu este supraîncărcat pentru a asigura prelucrarea în timp util a fiecărui pachet.

La prima vedere ar putea părea că dacă, să zicem, un ruter are nevoie de 1μsec pentru a procesa un pachet, atunci el poate procesa 1 milion de pachete/sec. Această observație nu este adevărată pentru că vor exista întotdeauna perioade nefolosite datorate fluctuațiilor statistice ale încărcării. Dacă procesorul are nevoie de fiecare ciclu de ceas pentru a-și face treaba, pierderea chiar și a câtorva cicluri din cauza perioadelor de nefolosire ocazionale creează un decalaj de care nu se poate scăpa.

Oricum, chiar și cu o încărcare puțin sub capacitatea teoretică, se pot forma cozi și pot apărea întârzieri. Să considerăm o situație în care pachetele sosesc aleator, cu rată medie de sosire de λ pachete/sec. Timpul de procesor cerut de fiecare pachet este de asemenea aleator, cu o capacitate medie de procesare de μ pachete/sec. Presupunând că distribuțiile sosirii și servirii sunt distribuții Poisson, folosind teoria cozilor se poate demonstra că întârzierea medie a unui pachet, *T*, este:

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda / \mu} = \frac{1}{\mu} \times \frac{1}{1 - \rho}$$

unde $\rho = \lambda/\mu$ este utilizarea procesorului. Primul factor, $1/\mu$, reprezintă timpul de servire în absența competiției. Al doilea factor este încetinirea cauzată de competiția cu alte fluxuri. De exemplu, dacă λ = 950,000 de pachete/sec și μ = 1,000,000 de pachete/sec, atunci ρ = 0.95 și întârzierea medie a unui pachet va fi de 20 μsec în loc de 1 μsec Acest timp ia în considerare atât timpul de așteptare în coadă, cât și timpul de servire, după cum se poate vedea atunci când încărcarea este foarte mică ($\lambda/\mu \approx 0$). Dacă presupunem că pe traseul fluxului există 30 de rutere, numai datorită întârzierilor în cozi vor rezulta 600μsec de întârziere.

Controlul accesului

Acum suntem în punctul în care traficul dintr-un flux oarecare este bine format și poate să urmeze o singură rută în care capacitatea poate fi rezervată în avans pe ruterele ce se găsesc de-a lungul căii. Când un asemenea flux este oferit unui ruter, acesta trebuie să decidă, pe baza capacității sale și a numărului de angajamente pe care le-a făcut deja cu alte fluxuri, dacă să primească sau să respingă fluxul.

Decizia de a accepta sau de a respinge fluxul nu este o simplă chestiune de comparație între (lățime de bandă, zone tampon, cicluri) cerute de către flux și capacitatea în exces a ruterului în aceste trei dimensiuni. Este puțin mai complicat decât atât. În primul rând, deși unele aplicații și-ar putea cunoaște necesitățile de lățime de bandă, puține știu despre zonele tampon sau despre ciclurile de procesor, deci, la nivel minim, este necesar un alt mod de descriere a fluxurilor. Apoi, unele aplicații sunt de departe mai tolerante la ratarea ocazională a unui termen limită. În cele din urmă, unele aplicații ar putea fi dornice să negocieze în legătură cu parametrii fluxului, iar altele nu. De exemplu, un server video care rulează în mod normal la 30 de cadre/sec ar putea fi dispus să coboare la 25 de cadre /sec dacă nu există suficientă lățime de bandă liberă pentru a suporta 30 de cadre/sec. În mod similar pot fi ajustați numărul de pixeli pe cadru, lățimea de bandă audio și alte proprietăți.

Deoarece în negocierea fluxului pot fi implicate mai multe părți (emițătorul, receptorul și toate ruterele aflate de-a lungul căii dintre aceștia), fluxurile trebuie descrise exact în termenii parametrilor specifici ce pot fi negociați. Un set de astfel de parametri se numește **specificarea fluxului (flow specification)**. Tipic, emițătorul (adică serverul video) produce o specificație a fluxului propunând parametrii pe care ar vrea să îi folosească. Pe măsură ce specificația se propagă de-a lungul căii, fiecare ruter o examinează și îi modifică parametrii după cum are nevoie. Modificările pot doar să reducă fluxul, nu să îl și crească (de exemplu o rată mai mică a datelor, nu mai mare). Când ajunge la capătul celălalt, parametrii pot fi stabiliți.

Ca un exemplu de ce poate exista într-o specificație a fluxului, să considerăm exemplul din fig. 5-35, care se bazează pe RFC-urile 2210 și 2211. Are cinci parametri, dintre care primul, *rata găleții cu jeton*, reprezintă numărul de octeți pe secundă care sunt puși în găleată. Aceasta este rata maximă suportată la care poate transmite emițătorul, mediată de-a lungul unui interval mare de timp.

Parametru	Unitate de măsură
Rata găleții cu jeton	Octeți/sec
Dimensiunea găleții cu jeton	Octeți
Rata de vârf a datelor	Octeți/sec
Dimensiunea minimă a pachetului	Octeți
Dimensiunea maximă a pachetului	Octeți

Fig. 5-35. Un exemplu de specificație a fluxului

Al doilea parametru este dimensiunea găleții în octeți. Dacă, de exemplu, *rata găleții cu jeton* este de 1 Mbps și *dimensiunea găleții cu jeton* este de 500 KB, găleata se poate umplută continuu timp de 4 sec până să fie plină (în absența oricărei transmisii). Orice jeton trimis după aceasta este pierdut.

Al treilea parametru, *Rata de vârf a datelor* este rata de transmisie maximă tolerată, chiar și pentru intervale scurte de timp. Emițătorul nu trebuie să depășească niciodată această valoare.

Ultimii doi parametri specifică dimensiunile minimă și maximă ale pachetului, incluzând antetele nivelelor transport și rețea (de exemplu TCP și IP). Dimensiunea minimă este importantă deoarece durata procesării fiecărui pachet este fixată, indiferent cât este de mic pachetul. Un ruter poate fi pregătit să se ocupe de 10,000 de pachete/sec de câte 1 KB fiecare, dar nepregătit să se ocupe de 100000 de pachete/sec de câte 50 de octeți fiecare, chiar dacă acestea reprezintă mai puține date. Dimensiu-

nea maximă a pachetului este importantă datorită limitărilor interne ale rețelei, care nu pot fi depășite. De exemplu, dacă o parte a căii trece prin Ethernet, dimensiunea maximă a pachetului va fi restricționată la nu mai mult de 1500 de octeți, indiferent de cât poate să suporte restul rețelei.

O întrebare interesantă este cum transformă un ruter o specificație a fluxului într-un set de rezervări specifice de resurse. Această mapare este implementată specific și nu este standardizată. Să presupunem că un ruter poate procesa 100000 pachete/sec. Dacă îi este oferit un flux de 1MB/sec cu dimensiunile minimă și maximă a pachetului de 512 octeți, ruterul poate calcula că ar putea lua 2048 de pachete/sec din acel flux. În acest caz, el trebuie să rezerve 2% din procesor pentru acel flux, preferabil mai mult pentru a evita întârzierile cauzate de așteptarea în coadă. Dacă politica unui ruter este de a nu aloca niciodată mai mult de 50 % din procesor (ceea ce implică o întârziere dublă) și este deja solicitat 49%, atunci acest flux trebuie respins. Calcule asemănătoare sunt necesare și pentru celelalte resurse.

Cu cât este mai exigentă specificația, cu atât ea este mai folositoare ruterelor. Dacă o specificație de flux spune că are nevoie de *o rată a găleții cu jeton* de 5MB/sec dar pachetele pot varia de la 50 de octeți la 1500 de octeți, atunci rata pachetelor va varia de la circa 3500 de pachete/sec la 105000 de pachete/sec. Ruterul s-ar putea panica la acest număr și ar putea respinge fluxul, deși la o dimensiune minimă a pachetului de 1000 de octeți, fluxul de 5MB/sec ar fi fost acceptat.

Dirijarea proporțională

Cei mai mulți algoritmi de dirijare încearcă să găsească cea mai bună cale pentru fiecare destinație și îndreaptă tot traficul spre acea destinație pe cea mai bună cale. O abordare diferită care a fost propusă pentru a furniza o mai bună calitate a serviciilor este de a împărți traficul pentru fiecare destinație pe mai multe căi. Din moment ce ruterele nu au în general o imagine de ansamblu completă asupra traficului din rețea, singura posibilitate reală de a împărți traficul pe mai multe rute este de a folosi informațiile disponibile local. O metodă simplă este de a împărți traficul în mod egal sau proporțional cu capacitatea legăturilor de ieșire. Totuși, sunt disponibili și algoritmi mai sofisticați (Nelakuditi și Zhang, 2002).

Planificarea pachetelor

Dacă un ruter se ocupă de mai multe fluxuri, există pericolul ca un flux să ia prea mult din capacitate, înghețând toate celelalte fluxuri. Procesare pachetelor în ordinea sosirii lor înseamnă că un emițător agresiv poate acapara cea mai mare parte din capacitatea ruterelor pe care le traversează pachetele sale, micșorând calitatea serviciilor pentru celelalte. Pentru a împiedica asemenea tentative au fost inventați diverși algoritmi de planificare a pachetelor (Bhatti și Crowcroft, 2000).

Unul dintre primii a fost algoritmul **așteptare echitabilă (fair queuing)** (Nagle, 987). Esența algoritmului este că ruterele au cozi separate pentru fiecare linie de ieșire, câte una pentru fiecare flux. Când o linie se eliberează, ruterul scanează cozile după metoda round robin, luând primul pachet din coada următoare. În acest fel, cu *n* calculatoare gazdă care concurează pentru o anumită linie de ieșire, fiecare gazdă reușește să expedieze un pachet din fiecare *n*. Trimiterea mai multor pachete nu va îmbunătăți rata.

Deși este un început, algoritmul are o problemă: dă mai multă lățime de bandă gazdelor care folosesc pachete mari decât celor care folosesc pachete mici. Demers ș.a. (1990) a sugerat o îmbunătățire în care parcurgerea round robin este realizată astfel încât să simuleze un round-robin octet-cu-octet în locul unui round robin pachet-cu-pachet. În concluzie, el scanează cozile în mod repetat, octet cu octet, până găsește momentul la care fiecare pachet va fi terminat. Apoi pachetele sunt sortate în ordine terminării lor și trimise în acea ordine. Algoritmul este ilustrat în fig. 5-36.

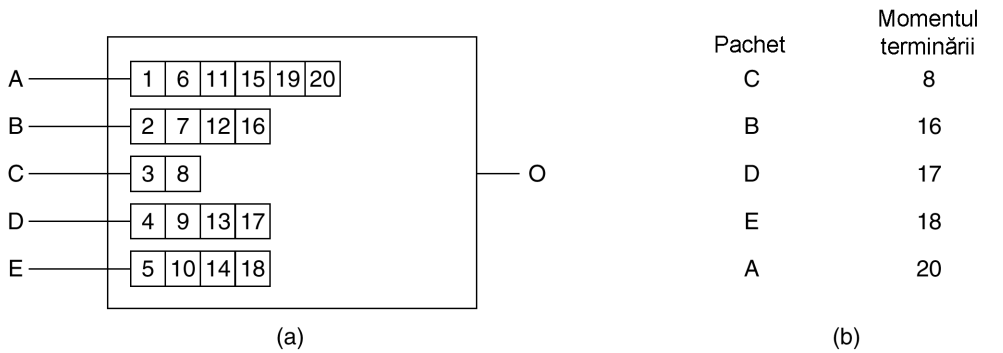


Fig. 5-36. (a) Un ruter cu 5 pachete așteptând pentru linia *O*.
(b) Momentul terminării pentru cele 5 pachete.

În fig. 5-36(a) observăm pachete cu lungimi între 2 și 6 octeți. La momentul (virtual) 1 este trimis primul octet al pachetului de pe linia A. Apoi urmează primul octet al pachetului de pe linia B și așa mai departe. Primul pachet care este terminat este C, după opt perioade. Ordinea sortată este dată în fig. 5-36(b). În absența unor noi sosiri, pachetele vor fi trimise în ordinea listată, de la C la A.

O problemă a acestui algoritm este aceea că acordă tuturor calculatoarelor gazdă aceeași prioritate. În multe situații, este de dorit să se acorde mai multă lățime de bandă serverelor video decât serverelor de fișiere normale în așa fel încât să poată transmită doi sau mai mulți octeți pe perioadă. Acest algoritm modificat se numește **așteptare echitabilă ponderată** (**weighted fair queueing**) și este foarte folosit. Uneori ponderea este egală cu numărul de fluxuri care ies dintr-un calculator, astfel încât toate procesele primesc lățime de bandă egală. O implementare eficientă a acestui algoritm este discutată în (Shreedhar și Varghese, 1995). Din ce în ce mai mult, retransmiterea efectivă a pachetelor printr-un ruter sau comutator se face prin hardware (Elhananz et al., 2001).

5.4.3 Servicii integrate

Între 1995 și 1997, IETF a depus mult efort pentru a inventa o arhitectură pentru fluxurile de tip multimedia. Această muncă s-a concretizat în peste doua duzini de RFC-uri, începând cu RFC-urile 2205-2210. Numele generic al acestui rezultat este **algoritmi bazați pe flux** (**flow-based algorithms**) sau **servicii integrate** (**integrated services**). A fost gândit atât pentru aplicații cu trimitere unică (eng.: unicast) cât și pentru cele cu trimitere multiplă (eng.: multicast). Un exemplu pentru cele dintâi este cazul unui singur utilizator rulând o secvență video de pe un sit de știri. Un exemplu pentru cel din urmă este o colecție de stații de televiziune prin cablu difuzându-și programele ca șiruri de pachete IP mai multor receptori din diverse locații. În cele ce urmează ne vom concentra pe multicast, având în vedere că unicastul este un caz particular al multicast-ului.

În numeroase aplicații multicast, grupurile își pot schimba dinamic componența, de exemplu ca participanții la o videoconferință care se plictisesc și comută pe un canal cu un serial ușurel sau pe canalul de crochet. În aceste condiții, abordarea în care emițătorii rezervă lățime de bandă în avans nu mai funcționează corespunzător, deoarece va cere fiecărui emițător să urmărească toate intrările și ieșirile celor din audiența sa. Pentru un sistem destinat transmisiilor televiziune cu milioane de abonați, nu va merge deloc.

RSVP - Protocol de rezervare a resurselor

Principalul protocol IETF pentru arhitectura serviciilor integrate este **RSVP (Resource reSerVation Protocol)**. El este descris în RFC 2205 și altele. Acest protocol este folosit pentru a face rezervări; pentru transmisia datelor sunt folosite alte protocoale. RSVP permite emițătorilor multipli să transmită spre grupuri multiple de receptori, permite fiecărui receptor să schimbe canalul la alegere și optimizează lățimea de bandă folosită, eliminând în același timp congestia.

În forma sa cea mai simplă, protocolul folosește dirijarea multicast cu arbori de acoperire, așa cum s-a discutat anterior. Fiecare grup are asociată o adresă de grup. Pentru a trimite unui grup, emițătorul pune adresa grupului în pachetele pe care le trimite. În continuare, algoritmul standard de dirijare multicast va construi un arbore de acoperire care acoperă toți membrii grupului. Algoritmul de dirijare nu este parte a RSVP. Singura diferență față de multicast-ul normal este o mică informație suplimentară distribuită periodic grupului pentru a spune ruterelor de-a lungul arborelui să mențină anumite structuri de date.

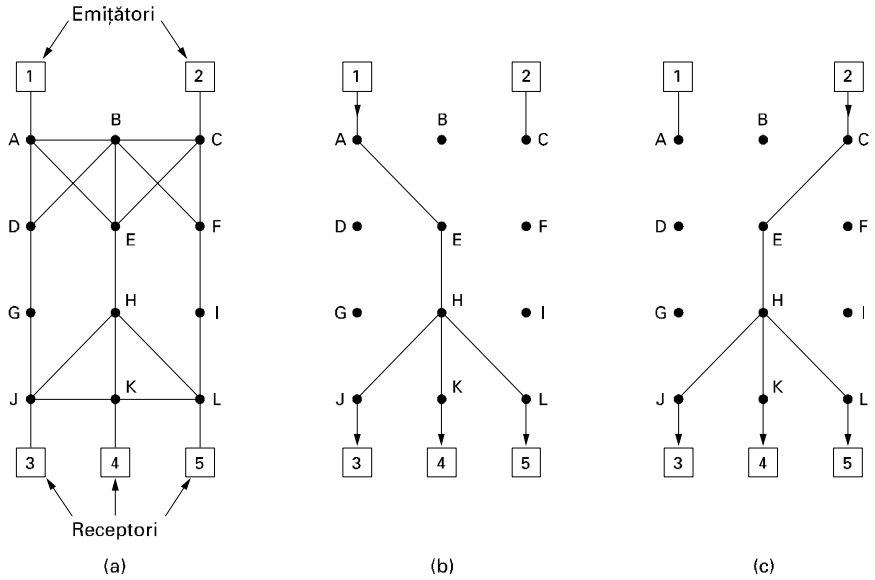


Fig. 5-37. (a) O rețea. (b) Arborele de acoperire multicast pentru calculatorul 1. (c) Arborele de acoperire multicast pentru calculatorul 2.

Ca exemplu, să considerăm rețeaua din fig. 5-37(a). Calculatoarele gazdă 1 și 2 sunt emițători multicast, iar calculatoarele gazdă 3, 4 și 5 sunt receptori multicast. În acest exemplu emițătorii și receptorii sunt disjuncți, dar în general cele două mulțimi se pot suprapune. Arborii multicast pentru mașinile 1 și 2 sunt prezentați în fig. 5-37(b), respectiv fig. 5-37(c).

Pentru a avea o recepție mai bună și pentru a elimina congestia, fiecare dintre receptorii dintr-un grup poate să trimită un mesaj de rezervare în sus pe arbore spre emițător. Mesajul este propagat folosind algoritmul căii inverse discutat anterior. La fiecare salt, ruterul notează rezervarea și rezervă lățimea de bandă necesară. Dacă lățimea de bandă este insuficientă, se raportează eroare. Atunci când mesajul ajunge la sursă, lățimea de bandă a fost rezervată pe tot drumul de la emițător spre receptorul care a făcut rezervarea de-a lungul arborelui de acoperire.

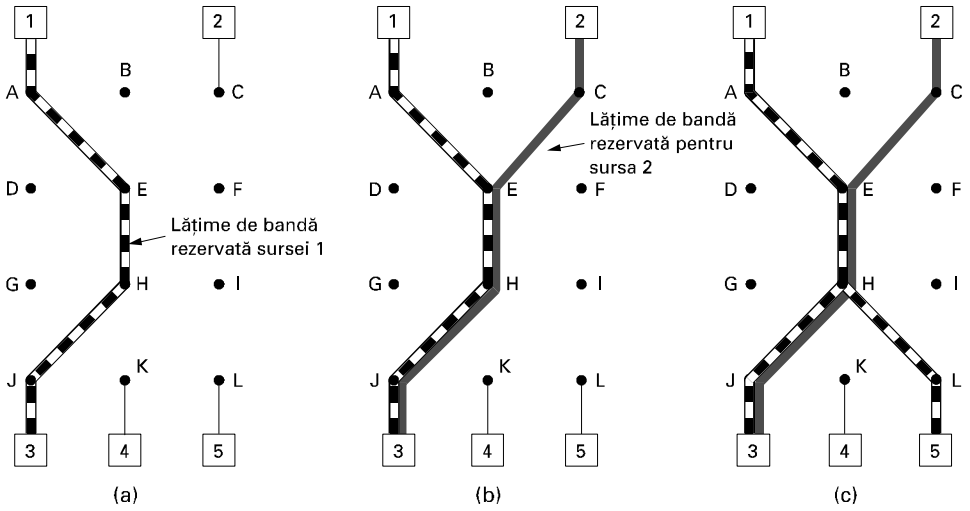


Fig. 5-38. (a) Calculatorul gazdă 3 cere un canal către calculatorul gazdă 1. (b) Calculatorul gazdă 3 cere un al doilea canal către calculatorul gazdă 2. (c) Calculatorul gazdă 5 cere un canal către calculatorul gazdă 1.

Un exemplu de astfel de rezervare este prezentat în fig. 5-38(a). Aici calculatorul gazdă 3 a cerut un canal către calculatorul gazdă 1. Odată ce acesta a fost stabilit, pachetele pot ajunge de la 1 la 3 fără congestie. Să vedem acum ce se întâmplă dacă gazda 3 rezervă și un canal către celălalt emițător, calculatorul gazdă 2, astfel încât utilizatorul să poată urmări două posturi de televiziune simultan. Se rezervă o a doua cale, așa cum se ilustrează în fig. 5-38(b). Observați că sunt necesare două canale distincte de la calculatorul gazdă 3 către ruterul *E*, deoarece se transmit două fluxuri independente.

În fine, în fig. 5-38 (c) calculatorul gazdă 5 decide să urmărească programul de televiziune transmis de 1 și face de asemenea o rezervare. Pentru început, se rezervă lățime de bandă spre ruterul *H*. Acest ruter observă că are deja o rezervare către 1, astfel încât dacă lățimea de bandă necesară a fost rezervată, nu mai trebuie să rezerve nimic. Se poate întâmpla ca 3 și 5 să ceară lățimi de bandă diferite (de exemplu, 3 are un sistem de televiziune alb-negru, astfel încât nu are nevoie de informația de culoare), caz în care capacitatea rezervată trebuie să fie suficientă pentru a satisface cererea cea mai mare.

La realizarea unei rezervări, un receptor poate (opțional) specifica una sau mai multe surse de la care vrea să recepționeze. De asemenea, el mai poate specifica dacă aceste alegeri sunt fixe pe durata rezervării sau dacă receptorul dorește să-și păstreze opțiunea de a modifica sursele ulterior. Ruterele folosesc această informație pentru a optimiza planificarea lățimii de bandă. În particular, doi receptori pot să partajeze o cale doar dacă ambii stabilesc să nu modifice sursele ulterior.

Motivul acestei strategii în cazul dinamic sută la sută este că rezervarea lățimii de bandă este decuplată de alegerea sursei. Odată ce un receptor a rezervat lățime de bandă, el poate comuta către altă sursă și să păstreze porțiunea din calea existentă care este comună cu calea către noua sursă. Când calculatorul 2 transmite mai multe fluxuri video, de exemplu, calculatorul 3 poate comuta între ele după dorință, fără a-și schimba rezervarea: ruterelor nu le pasă la ce program se uită utilizatorul.

5.4.4 Servicii diferențiate

Algoritmii bazați pe flux au potențialul de a oferi o bună calitate a serviciilor unuia sau mai multor fluxuri deoarece acestea își rezervă resursele necesare de-a lungul rutei. Totuși ei au și un dezavantaj: au nevoie să stabilească în avans caracteristicile fiecărui flux, ceea ce nu este optim în cazul în care există milioane de fluxuri. De asemenea, ei memorează caracteristicile fiecărui flux ca date interne ale ruterului, ceea ce le face vulnerabile în cazul căderii ruterului. În fine, modificările necesare codului ruterului sunt substanțiale și implică schimburi complexe de informații între rutere pentru stabilirea fluxurilor. Ca o consecință, există foarte puține implementări ale RSVP-ului sau a ceva asemănător.

Din aceste motive, IETF a propus și o abordare mai simplă a calității serviciilor, care poate fi implementată local în fiecare ruter fără inițializări prealabile și fără a implica întreaga cale. Această abordare este cunoscută sub numele de calitatea serviciilor **orientate pe clase (class-based)** (opusă abordării bazate pe flux). IETF a standardizat o arhitectură numită **servicii diferențiate** (**differentiated services**), care este descrisă în RFC-urile 2474, 2475 și altele. O vom descrie în continuare.

Serviciile diferențiate (DS) pot fi oferite de către un set de rutere care formează un domeniu administrativ (de exemplu un ISP sau telco). Administrația definește un set de clase de servicii cu regulile de rutare corespunzătoare. Dacă un client se înscrie pentru DS, pachetele clientului care intră în domeniu pot avea un câmp *Type of Service (Tipul serviciului)*, cu servicii mai bine furnizate unor clase (de exemplu serviciu premium) decât altora. Traficului dintr-o clasă i se poate cere să se conformeze unei anumite forme, cum ar fi găleata găurită cu o anumită rată decurgere. Un operator cu fler ar putea să ceară mai mult pentru fiecare pachet premium transportat sau ar putea să permită maximum N pachete premium pe lună pentru o taxă lunară suplimentară fixă. Observați că această schemă nu presupune inițializarea în avans, nici rezervarea resurselor și nici negocierea capăt-la-capăt pentru fiecare flux, care consumă timp, ca în cazul serviciilor integrate. Aceasta face ca serviciile diferențiate să fie relativ ușor de implementat.

Serviciile bazate pe clase se întâlnesc și în alte domenii. De exemplu, companiile care livrează pachete oferă de obicei servicii de tip “peste noapte”, “în două zile” sau “în trei zile”. Companiile aeriene oferă clasa întâi, clasa de afaceri și clasa a doua. Trenurile pe distanțe lungi oferă adeseori multiple clase de servicii. Chiar și metroul din Paris are două tipuri de servicii. Pentru pachete, clasele pot să difere, printre altele, prin valorile întârzierii, fluctuației și a probabilității pachetului de a fi aruncat la apariția unei congestii.

Pentru a evidenția diferența dintre calitatea serviciilor bazate pe flux și calitatea serviciilor bazate pe clase, să considerăm un exemplu: telefonia Internet. În cazul organizării pe flux, fiecare apel telefonic are propriile resurse și garanții. În cazul organizării pe clase, toate apelurile telefonice beneficiază de resursele rezervate pentru clasa telefoniei. Aceste resurse nu pot fi preluate de pachete din clasa transferului de fișiere sau din alte clase, dar nici un apel telefonic nu beneficiază de resurse particulare, rezervate doar pentru acesta.

Retransmitere expeditivă (Expedited Forwarding)

Alegerea claselor de servicii este la dispoziția fiecărui operator, dar, având în vedere că pachetele sunt adesea expediate între subrețele administrate de operatori diferiți, IETF lucrează la definirea unor clase de servicii independente de rețea. Cea mai simplă clasă este **retransmiterea expeditivă (expedited forwarding)**, deci vom începe cu aceasta. Descrierea ei este dată în RFC 3246.

Ideea care stă la baza retransmiterii expeditive este foarte simplă. Sunt disponibile două clase de servicii: normale și rapide (expeditive). Marea majoritate a traficului se presupune că este de tip normal, dar o mică parte se face și expeditiv. Pachetele din această clasă ar trebui să poată traversa subrețeaua ca și cum nu ar mai exista și alte pachete. O reprezentare simbolică a acestui sistem "bi-canal" este dată în fig. 5-39. Observați că există doar o singură linie fizică. Cele două canale logice din figură reprezintă o cale de a rezerva lățime de bandă și nu o a doua linie fizică.

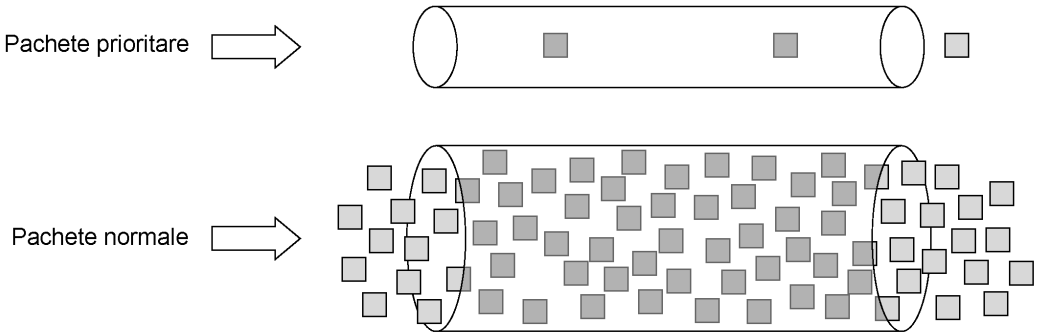


Fig. 5-39. Pachetele prioritare trec printr-o rețea cu trafic redus.

O modalitate de a implementa această strategie este de a programa ruterele astfel încât să aibă două cozi de așteptare pentru fiecare linie de ieșire, una pentru pachete prioritare și cealaltă pentru pachete normale. Când sosește un pachet este introdus în coada corespunzătoare. Planificarea pachetelor ar trebui să utilizeze ceva de genul așteptării echitabile ponderate. De exemplu, dacă 10% din trafic este de tip expeditiv și 90% de tip normal, atunci 20% din lățimea de bandă ar putea fi repartizată traficului expeditiv, iar restul traficului normal. În acest fel traficul expeditiv ar avea de două ori mai multă lățime de bandă decât îi este necesar pentru a asigura întârziere mică. Această repartizare poate fi obținută transmițând câte un pachet prioritar la fiecare patru pachete normale (presupunând că distribuția dimensiunilor pachetelor ambelor clase este similară). În acest fel se speră că pachetele prioritare nu văd subrețeaua ca fiind aglomerată, chiar dacă există o încărcare substanțială.

Rutare garantată

O metodă mai elaborată de a administra clasele de servicii este **rutarea garantată (assured forwarding)**. Descrierea acesteia se găsește în RFC 2597. Ea precizează că ar trebui să existe patru clase de priorități, fiecare cu propriile resurse. În plus definește trei probabilități de aruncare a pachetelor care sunt supuse congestiei: mică, medie și mare. Luate împreună, aceste două criterii definesc 12 clase de servicii.

Fig. 5-40 prezintă o modalitate în care pachetele ar putea fi procesate în cazul rutării garantate. Primul pas este acela de a repartiza pachetele într-una din cele patru clase de priorități. Acest pas se poate face pe calculatorul emițător (după cum se vede și din figură) sau în primul ruter. Avantajul realizării clasificării pe calculatorul gazdă care realizează transmisia este acela că are la dispoziție mai multe informații despre fiecare pachet și direcția în care se îndreaptă.

Al doilea pas este de a marca pachetele în conformitate cu clasa din care fac parte. Pentru aceasta este necesar un câmp antet. Din fericire în antetul IP este disponibil un câmp pe 8 biți numit *Tipul serviciului*, după cum vom vedea în continuare. RFC 2597 specifică faptul că șase dintre acești biți se vor folosi pentru clasa de serviciu, lăsând loc pentru codificarea claselor istorice și a celor viitoare.

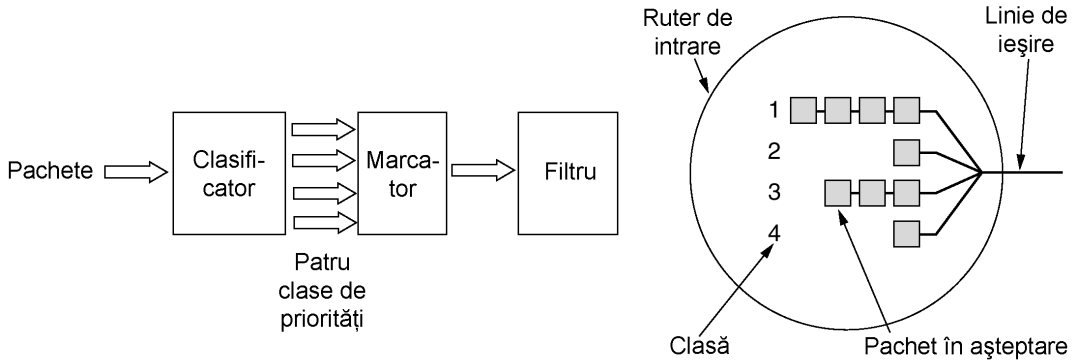


Fig. 5-40. O posibilă implementare a fluxului de date pentru rutarea garantată.

Pasul al treilea presupune trecerea pachetelor printr-un filtru (eng.: shaper/dropper filter) care ar putea întârzia sau arunca o parte dintre ele pentru a forma cele patru fluxuri într-o manieră acceptabilă, folosind de exemplu algoritmul găleții găurite sau al găleții cu jeton. Dacă sunt prea multe pachete, unele dintre ele ar putea fi aruncate în această etapă, în funcție de categoria în care au fost plasate de filtru. Sunt posibile și scheme mai elaborate care implică măsurători și reacții inverse

În acest exemplu, pașii specificați sunt efectuați pe calculatorul emițător, deci fluxul rezultat este trimis primului ruter. Merită observat faptul că acești pași pot fi efectuați de un software de rețea specializat sau chiar de către sistemul de operare, pentru a evita modificarea aplicației existente.

5.4.5 Comutarea etichetelor și MPLS

În timp ce IETF lucra la serviciile integrate și diferențiate, unii dintre vânzătorii de rutere căutau metode mai bune de rutare. Munca lor s-a axat pe adăugarea unei etichete în fața fiecărui pachet și pe efectuarea rutării mai degrabă pe baza acestei etichete decât a adresei destinație. Utilizarea etichetei ca index într-o tabelă internă a ruterului face ca găsirea liniei de ieșire corecte să se transforme într-o simplă căutare în tabel. Folosind această metodă, rutarea se poate face foarte repede iar resursele necesare pot fi rezervate pe traseu.

Bineînțeles, etichetarea fluxurilor în acest fel se apropie foarte mult de circuitele virtuale. X.25, ATM, releu de cadre (eng.: frame-relay) și toate celelalte rețele cu subrețele cu circuite virtuale pun și ele câte o etichetă (identificatorul circuitului virtual) în fiecare pachet, o caută într-o tabelă și rutarea se face pe baza intrării din tabelă. În ciuda faptului că mulți dintre membrii comunității Internet au o puternică antipatie față de rutarea orientată pe conexiune, se pare că nu se renunță la idee, de data aceasta pentru a furniza o rutare rapidă și calitatea serviciilor. Totuși există diferențe fundamentale între modul în care se ocupă Internetul de construcția rutelor și modul în care o fac rețelele orientate pe conexiune, deci în mod sigur tehnica nu este tradițională comutare de circuite.

"Noua" idee de comutare este cunoscută sub mai multe denumiri, inclusiv **comutarea etichetelor** (eng.: **label switching**) și **comutarea marcajelor** (eng.: **tag switching**). În cele din urmă, IETF a început să standardizeze ideea sub numele de **multiprotocol de comutare a etichetelor (MPLS – MultiProtocol Label Switching)**. În continuare o vom numi MPLS. Aceasta este descrisă în RFC 3031 și în multe alte RFC-uri.

Pe de altă parte, unii fac diferența între *rutare* și *comutare*. Rutarea este procesul de căutare a adresei destinație în tabel pentru a găsi unde trebuie trimis. În schimb, comutarea folosește o eti-

chetă luată dintr-un pachet ca index într-o tabelă de rutare. Aceste definiții sunt totuși departe de a fi universale.

Prima problemă care apare este unde se pune eticheta. Din moment ce pachetele IP nu au fost proiectate pentru circuite virtuale, în cadrul antetelor IP nu există nici un câmp disponibil pentru numerele circuitelor virtuale. Din acest motiv, în fața antetului IP a trebuit adăugat un nou antet MPLS. Pe o linie ruter-la-ruter folosind PPP ca protocol de încadrare, formatul cadrului, inclusiv antetele PPP, MPLS, IP și TCP, arată ca în fig. 5-41. Într-un fel, MPLS este nivelul 2.5.

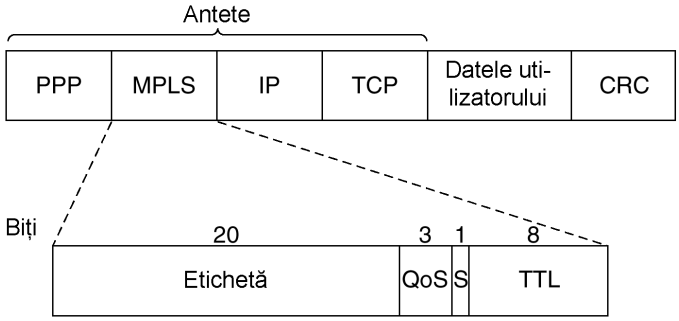


Fig. 5-41. Transmiterea unui segment TCP utilizând IP, MPLS și PPP.

Antetul generic MPLS are 4 câmpuri, dintre care cel mai important este câmpul *Etichetă (Label)* care deține indexul. Câmpul *QoS (Quality of Service; rom.: Calitatea serviciilor)* precizează clasa de servicii. Câmpul *S* se referă la memorarea mai multor etichete în rețelele ierarhice (care vor fi discutate mai târziu). Dacă ajunge la valoarea 0, pachetul este aruncat. Această facilitate previne buclarea infinită în cazul instabilității rutării.

Deoarece antetele MPLS nu fac parte din pachetul format la nivelul rețea și nici din cadrele nivelului legătură de date, MPLS reprezintă o extindere a acestor două nivele. Printre altele aceasta înseamnă că este posibilă construirea de comutatoare MPLS care să poată ruta atât pachete IP cât și celule ATM, după necesități. Această caracteristică a dus la apariția termenului "multiprotocol" din denumirea de MPLS.

Când un pachet îmbunătățit MPLS (sau o celulă) ajunge la un ruter ce suportă MPLS, eticheta este folosită ca index într-un tabel pentru a determina linia de ieșire ce va fi folosită precum și noua etichetă. Această schimbare de etichete se folosește în toate subrețelele cu circuite virtuale deoarece etichetele au doar semnificație locală și două rutere diferite pot trimite pachete care nu au legătură unul cu altul dar au aceeași etichetă unui alt ruter pentru a fi transmise pe aceeași linie de ieșire. Pentru a putea fi diferențiate la capătul celălalt, etichetele trebuie să fie remapate la fiecare salt. Am văzut cum funcționează acest mecanism în fig. 5-3. MPLS folosește aceeași tehnică.

O diferență față de circuitele virtuale tradiționale este nivelul de agregare. În mod sigur este posibil ca fiecare flux să aibă propriul set de etichete în cadrul subrețelei. Totuși se obișnuiește mai mult ca ruterele să grupeze mai multe fluxuri care au ca destinație un anumit ruter sau LAN și să folosească pentru ele o singură etichetă. Despre fluxurile care sunt grupate sub o aceeași etichetă se spune că aparțin aceleiași **Clase echivalente de rutare (FEC – Forwarding Equivalence Class)**. Această clasă acoperă nu numai pachetele care pleacă, dar și clasele lor de servicii (în sensul claselor diferențiate de servicii) deoarece toate pachetele pe care le conțin sunt tratate la fel din motive de expediere.

La rutarea tradițională de tip circuit virtual nu este posibil să se grupeze mai multe căi cu destinații diferite pe același identificator de circuit virtual, deoarece la destinația finală nu ar mai fi posibil

să se facă distincție între ele. Cu MPLS, pachetele conțin, pe lângă etichetă, și adresa lor finală de destinație, în așa fel încât, la capătul rutei etichetate, antetul poate fi eliminat și rutarea poate continua în mod obișnuit, folosind adresa de destinație a nivelului rețea.

O diferență majoră dintre MPLS și circuitele virtuale convenționale este modul în care este construită tabela de rutare. În rețelele cu circuite virtuale tradiționale, când un utilizator dorește să stabilească o conexiune, este lansat în rețea un pachet de inițializare pentru a crea calea și intrările tabelelor de rutare. MPLS nu funcționează astfel deoarece nu există fază de inițializare pentru fiecare conexiune (deoarece ar stopa prea mult din software-ul existent în Internet).

În schimb există două modalități în care pot fi create intrările tabelelor de rutare. În abordarea **bazată pe date**, **(data-driven)** atunci când un pachet ajunge la primul ruter, acesta contactează ruterul din aval, la care trebuie să ajungă pachetul, și îi cere să genereze o etichetă pentru flux. Această metodă se aplică recursiv. Practic, aceasta este crearea circuitelor virtuale la cerere.

Protocoalele care realizează această răspândire au foarte mare grijă să evite buclele. Ele folosesc adeseori tehnica numită a **firelor colorate (colored threads)**. Propagarea înapoi a unei FEC poate fi comparată cu tragerea unui fir de o singură culoare înapoi în subrețea. Dacă un ruter întâlnește o culoare pe care o are deja, el știe că există o buclă și ia măsuri pentru remedierea situației. Abordarea bazată pe date este folosită în rețelele unde sub nivelul transport se găsește ATM-ul (ca în cazul sistemului telefonic).

Cealaltă modalitate, folosită în rețele care nu sunt bazate pe ATM, este abordarea bazată pe control (eng.: **control-driven**). Ea are mai multe variante, dintre care una funcționează în felul următor. Când este pornit un ruter, verifică pentru ce rute el reprezintă destinația finală (de exemplu ce calculatoare gazdă sunt în rețeaua locală). Apoi creează una sau mai multe clase echivalente pentru acestea, alocă pentru fiecare câte o etichetă și trimite etichetele vecinilor săi. Aceștia, la rândul lor, introduc etichetele în tabelele de rutare și trimit noi etichete vecinilor lor, până când toate ruterele și-au însușit calea. De asemenea, pentru a garanta o calitate corespunzătoare a serviciilor, resursele pot fi alocate pe măsura construirii căii.

MPLS poate opera la mai multe niveluri deodată. La nivelul cel mai înalt, fiecare companie de telecomunicații poate fi privită ca un fel de metaruter, existând o cale prin metarutere de la sursă la destinație. Această cale poate folosi MPLS. Totuși, în cadrul rețelei fiecărei companii de telecomunicații poate fi folosit de asemenea MPLS, ducând la un al doilea nivel de etichetare. De fapt, un pachet poate transporta o întreagă stivă de etichete. Bitul *S* din fig. 5-41 permite unui ruter care șterge o etichetă să afle dacă au mai rămas alte etichete. Acesta are valoarea 1 pentru eticheta de la bază și 0 pentru toate celelalte etichete. În practică, această facilitate este folosită în principal la implementarea rețelelor virtuale private și a tunelelor recursive.

Deși ideile care au stat la baza MPLS-ului sunt simple, detaliile sunt extrem de complicate, cu multe variații și optimizări, așa că nu vom mai dezvolta acest subiect. Pentru mai multe informații, vezi (Davie și Rekhter, 2000; Lin et al., 2002; Pepelnjak și Guichard, 2001; și Wang, 2001).

5.5 INTERCONECTAREA REȚELELOR

Până în acest moment, am presupus implicit că există o singură rețea omogenă, în care fiecare mașină folosește același protocol la fiecare nivel. Din păcate, această presupunere este prea optimistă. Există mai multe tipuri de rețele, incluzând LAN-uri, MAN-uri și WAN-uri. La fiecare nivel se

folosesc pe larg numeroase protocoale. În secțiunile următoare vom analiza în detaliu problemele care apar când două sau mai multe rețele sunt conectate pentru a forma o **inter-rețea** (internet).

Există controverse considerabile pe tema întrebării dacă abundența actuală de tipuri de rețele este o condiție temporară, care va dispărea de îndată ce toată lumea își va da seama ce minunată este [completați cu tipul preferat de rețea], sau dacă este o caracteristică inevitabilă, dar permanentă, a lumii care va persista. Existența rețelelor de tipuri diferite înseamnă, invariabil, a avea protocoale diferite.

Credem că întotdeauna vor coexista o varietate de rețele diferite (și implicit de protocoale) din următoarele motive. În primul rând, numărul de rețele diferite instalate este mare. Aproape toate calculatoarele personale folosesc TCP/IP. Multe întreprinderi mari încă se bazează pe sisteme de calcul care folosesc SNA de la IBM. O parte importantă din companiile telefonice operează rețele ATM. Unele LAN-uri de calculatoare personale folosesc încă Novell NCP/IPX sau AppleTalk. Și, în final, în domeniul în plină dezvoltare al comunicațiilor fără fir există o mare varietate de protocoale. Această tendință va continua mult timp datorită problemelor de continuitate, noilor tehnologii și faptului că nu toți producătorii percep a fi în interesul lor posibilitatea clienților de a migra cu ușurință spre sistemul altui producător.

În al doilea rând, pe măsură ce calculatoarele și rețelele devin mai ieftine, nivelul la care se iau deciziile se mută în jos în organizație. Multe companii au o politică care stabilește că achizițiile de peste un milion de dolari trebuie să fie aprobate de conducerea superioară, achizițiile de peste 100.000 de dolari trebuie să fie aprobate de conducerea medie, dar achizițiile de până la 100.000 de dolari pot fi făcute de șefii de departamente fără nici o aprobare de mai sus. Aceasta poate duce ușor la instalarea unor stații de lucru UNIX care rulează TCP/IP în departamentul inginerie și a unor calculatoare Macintosh care folosesc AppleTalk în departamentul de marketing.

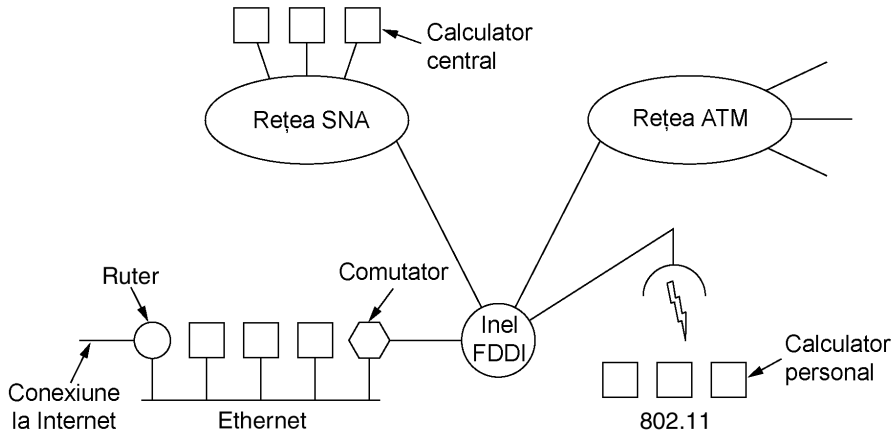


Fig. 5-42. Colecție de rețele interconectate.

În al treilea rând, rețele diferite (de exemplu ATM și fără fir) sunt bazate pe tehnologii radical diferite și nu ar trebui să surprindă că pe măsură ce apar dezvoltări hardware, se vor crea noi programe care să se potrivească cu noul hardware. De exemplu, casa medie de astăzi este asemănătoare cu biroul mediu de acum 10 ani; este plină de calculatoare care nu comunică unul cu celălalt. În viitor, ar putea fi un lucru obișnuit ca telefonul, televizorul și alte aparate electrocasnice să fie legate în rețea, pentru a permite controlul de la distanță. Această nouă tehnologie va aduce, fără dubii, noi rețele și noi protocoale.

Ca un exemplu al modului în care ar putea fi conectate rețele diferite, să considerăm exemplul din fig. 5-42. Aici vedem rețeaua unei companii cu mai multe sedii interconectate printr-o rețea ATM de mare întindere. La unul dintre sedii este folosită o coloană vertebrală (eng.: backbone) optică FDDI, prin care sunt conectate: o rețea Ethernet, un LAN fără fir 802.11 și rețeaua SNA de calculatoare centrale a centrului de date al corporației.

Scopul interconectării acestor rețele este de a permite utilizatorilor din orice rețea să comunice cu utilizatorii celorlalte rețele și de asemenea de a permite unui utilizator din orice rețea să acceseze date pe orice rețea. Realizarea acestui scop înseamnă trimiterea pachetelor dintr-o rețea în alta. Cum rețelele diferă deseori în puncte esențiale, transmiterea pachetelor dintr-o rețea în alta nu este întotdeauna ușoară, după cum vom vedea în continuare.

5.5.1 Prin ce diferă rețelele

Rețelele pot diferi în multe moduri. Unele dintre deosebiri, cum ar fi diferite tehnici de modulare sau formate de cadre, se găsesc la nivelul fizic și legătură de date. Aceste diferențe nu ne preocupă la acest nivel. În schimb, în fig. 5-43, enumerăm câteva din diferențele care pot apărea la nivelul rețea. Trecerea peste aceste diferențe face interconectarea rețelelor mult mai dificilă decât operarea într-o singură rețea.

Element	Câteva posibilități
Serviciu oferit	Orientat pe conexiuni față de cel fără conexiune
Protocol	IP, IPX, SNA, ATM, MPLS, AppleTalk etc.
Adresare	Plată (802) opusă celei ierarhice (IP)
Trimitere multiplă	Prezentă sau absentă (de asemenea, difuzarea totală)
Dimensiune pachet	Fiecare rețea își are propriul maxim
Calitatea serviciului	Poate fi prezentă sau absentă; multe tipuri diferite
Tratarea erorilor	Livrare fiabilă, ordonată sau neordonată
Controlul fluxului	Fereastră glisantă, controlul ratei, altele sau nimic
Controlul congestiei	Algoritmul găleții găurite, algoritmul găleții cu jeton, RED, pachete de șoc etc.
Securitate	Reguli de secretizare, criptare etc.
Parametri	Diferite limitări de timp, specificări ale fluxului etc.
Contabilizare	După timpul de conectare, după pachet, după octeți sau fără.

Fig. 5-43. Câteva din multele moduri în care pot diferi rețelele.

Când pachetele trimise de o sursă dintr-o rețea trebuie să tranziteze una sau mai multe rețele străine înainte de a ajunge în rețeaua destinație (care, de asemenea, poate fi diferită față de rețeaua sursă), pot apărea multe probleme la interfețele dintre rețele. Pentru început, atunci când pachetele dintr-o rețea orientată pe conexiuni trebuie să tranziteze o rețea fără conexiuni, poate interveni o reordonare a acestora, lucru la care emițătorul nu se așteaptă și căruia receptorul nu este pregătit să-i facă față. Vor fi necesare frecvente conversii de protocol, care pot fi dificile dacă funcționalitatea cerută nu poate fi exprimată. De asemenea, vor fi necesare conversii de adresă, ceea ce poate cere un sistem de catalogare. Trecerea pachetelor cu trimitere multiplă printr-o rețea care nu oferă trimitere multiplă necesită generarea de pachete separate pentru fiecare destinație.

Diferența dintre dimensiunile maxime ale pachetelor folosite de diferite rețele poate produce mari neplăceri. Cum veți trece un pachet de 8000 de octeți printr-o rețea a cărei dimensiune maximă este de 1500 de octeți? Diferențele în calitatea serviciilor sunt o problemă în momentul în care un

pachet care are constrângeri de livrare în timp real traversează o rețea care nu oferă nici o garanție de timp real.

Controlul erorilor, al fluxului și al congestiei diferă frecvent între rețele diferite. Dacă sursa și destinația așteaptă amândouă ca toate pachetele să fie livrate în ordine fără erori, dar o rețea intermediară elimină pachete ori de câte ori întrevede congestie la orizont, multe aplicații se vor comporta neprevăzut. Diferențele în ceea ce privește mecanismele de securitate, stabilirea parametrilor, regulile de contabilizare și chiar legile naționale referitoare la secrete pot, de asemenea, cauza probleme.

5.5.2 Cum pot fi conectate rețelele

Rețelele pot fi interconectate prin diferite dispozitive, așa cum s-a arătat în cap.4. Să revedem pe scurt acest material. La nivelul fizic, rețelele pot fi conectate prin repetoare sau noduri (eng.: hubs), care doar transferă biții între două rețele identice. Acestea sunt în marea lor majoritate dispozitive analogice și nu cunosc protocoalele numerice (doar regenerează semnale).

Cu un nivel mai sus întâlnim punțile și comutatoarele, care operează la nivelul legăturii de date. Acestea acceptă cadre, examinează adresele MAC și retransmit cadrele către o rețea diferită, efectuând traduceri de protocol minore, ca de exemplu de la Ethernet la FDDI sau la 802.11.

La nivelul rețea avem rutere care pot conecta două rețele. Dacă două rețele au niveluri rețea diferite, ruterul poate fi capabil să transforme formatul pachetelor, cu toate că astfel de situații sunt din ce în ce mai rare. Un ruter care poate trata mai multe protocoale este numit **ruter multiprotocol** (eng. multiprotocol router).

La nivelul transport întâlnim porți de transport, care por realiza interfața între două conexiuni de transport. De exemplu, o poartă de transport poate permite pachetelor să treacă între o rețea TCP și o rețea SNA, care are un protocol de transport diferit, făcând legătura între o conexiune TCP și o conexiune SNA.

În sfârșit, la nivelul aplicație, porțile de aplicație traduc semnificația mesajelor. De exemplu, punțile dintre poșta electronică din Internet (RFC 822) și poșta electronică X 400 trebuie să analizeze mesajele și să schimbe diferite câmpuri din antete.

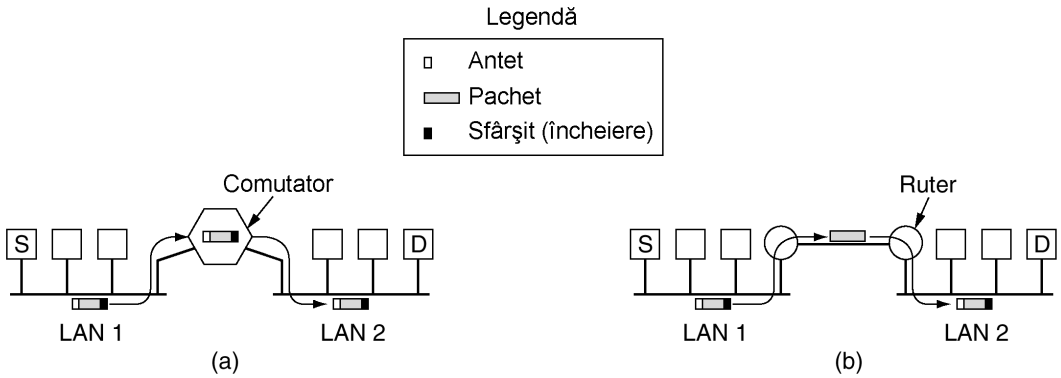


Fig. 5-44. (a) Două rețele Ethernet conectate printr-un comutator.
(b) Două rețele Ethernet conectate prin rutere.

În acest capitol ne vom concentra pe interconectarea rețelelor la nivelul rețea. Pentru a vedea prin ce diferă de comutarea la nivelul legăturii de date, să examinăm fig. 5-44. În fig. 5-44(a) mașina

sursă, *S*, dorește să trimită un pachet mașinii destinație, *D*. Aceste mașini se află în rețele Ethernet diferite, conectate printr-un comutator. *S* încapsulează pachetul într-un cadru și îl trimite spre destinație. Cadrul ajunge la comutator, care determină, analizând adresa MAC, destinația cadrului, reprezentată de LAN 2. Comutatorul nu face decât să preia cadrul din LAN 1 și să îl pună pe LAN 2.

Să considerăm acum aceeași situație, dar cu două rețele Ethernet conectate printr-o pereche de rutere, în loc de comutator. Ruterele sunt conectate printr-o linie punct-la-punct, posibil o linie închiriată de mai mulți kilometri lungime. Acum cadrul este preluat de ruter, iar pachetul este extras din câmpul de date al cadrului. Ruterul examinează adresa din pachet (de exemplu o adresă IP) și o caută în tabela sa de rutare. Pe baza acestei adrese decide să trimită pachetul la ruterul aflat la distanță, eventual încapsulat într-un alt tip de cadru, în funcție de protocolul liniei. La celălalt capăt pachetul este pus în câmpul de date al unui cadru Ethernet și este depus pe LAN 2.

O diferență esențială între cazul cu comutator (sau punte) și cazul cu rutere este următoarea. În cazul cu comutator (sau punte) este transportat întregul cadru, pe baza adresei MAC. În cazul unui ruter pachetul este extras din cadru, iar adresa din pachet este utilizată pentru a decide unde să fie trimis. Comutatoarele nu trebuie să înțeleagă protocolul nivelului rețea, dar ruterele da.

5.5.3 Circuite virtuale concatenate

Sunt posibile două stiluri de interconectare a rețelelor: o concatenare orientată pe conexiuni a subrețelelor cu circuite virtuale și un stil datagrame inter-rețea. Vom examina pe rând aceste variante, dar înainte de aceasta, un avertisment. În trecut, marea majoritate a rețelelor (publice) erau orientate pe conexiune (și releul de cadre, SNA, 802.16 și ATM încă sunt). Apoi, odată cu acceptarea rapidă a Internetului, datagramele au venit la modă. Totuși ar fi o greșeală să credem că datagramele vor fi folosite la nesfârșit. În acest domeniu singurul lucru etern este schimbarea. Odată cu creșterea importanței rețelelor multimedia, este probabil ca orientarea pe conexiune să revină într-o formă sau alta, deoarece este mai ușor să se garanteze calitatea serviciului cu conexiuni, decât fără ele. De aceea în continuare vom dedica spațiu orientării pe conexiune.

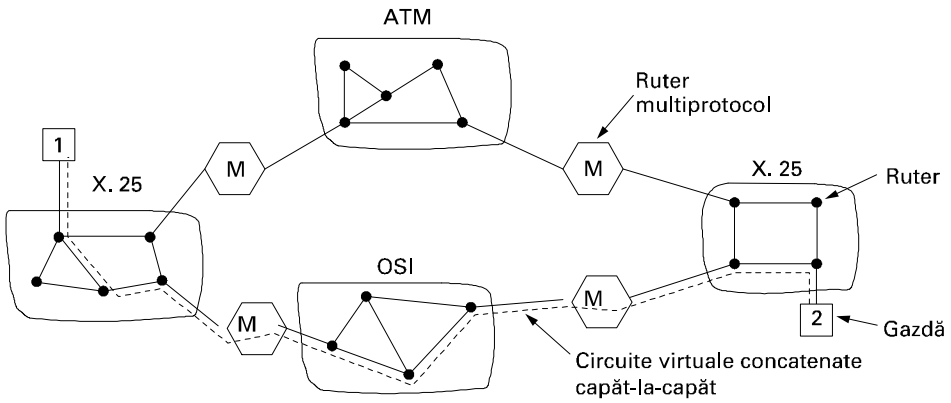


Fig. 5-45. Interconectarea rețelelor folosind circuite virtuale concatenate.

În modelul circuitelor virtuale concatenate, arătat în fig. 5-45, o conexiune către o gazdă dintr-o rețea îndepărtată este stabilită într-un mod similar cu modul în care sunt stabilite conexiunile în cazul normal. Subrețeaua observă că destinația este îndepărtată și construiește un circuit virtual spre

ruterul aflat cel mai aproape de rețeaua destinație. Apoi construiește un circuit virtual de la acel ruter la o **poartă** externă (un ruter multiprotocol). Această poartă înregistrează existența circuitului virtual în tabelele sale și continuă să construiască un alt circuit virtual către un ruter din următoarea subrețea. Acest proces continuă până când se ajunge la gazda destinație.

O dată ce pachetele de date încep să circule de-a lungul căii, fiecare poartă retransmite pachetele primite, făcând, după nevoie, conversia între formatele pachetelor și numerele de circuite virtuale. Evident, toate pachetele de date trebuie să traverseze aceeași secvență de porți, deci ajung în ordine.

Caracteristica esențială a acestei abordări este că se stabilește o secvență de circuite virtuale de la sursă, prin una sau mai multe porți, până la destinație. Fiecare poartă menține tabele spunând ce circuite virtuale o traversează, unde vor fi dirijate și care este numărul noului circuit virtual.

Această schemă funcționează cel mai bine atunci când toate rețelele au, în mare, aceleași proprietăți. De exemplu, dacă toate garantează livrarea sigură a pachetelor de la nivelul rețea, atunci, exceptând un accident de-a lungul căii, fluxul de la sursă la destinație va fi de asemenea sigur. Similar, dacă nici una din ele nu garantează livrarea sigură, atunci concatenarea circuitelor virtuale nu este nici ea sigură. Pe de altă parte, dacă mașina sursă este într-o rețea care garantează livrarea sigură, dar una din rețelele intermediare poate pierde pachete, concatenarea schimbă fundamental natura serviciului.

Circuitele virtuale concatenate sunt, de asemenea, uzuale la nivelul transport. În particular, este posibil să se construiască o conductă de biți folosind, să spunem, SNA, care se termină într-o poartă și având o conexiune TCP de la această poartă la poarta următoare. În acest mod, un circuit virtual capăt-la-capăt poate fi construit acoperind diferite rețele și protocoale.

5.5.4 Interconectarea rețelelor fără conexiuni

Modelul alternativ de interconectare este modelul datagramă, prezentat în fig. 5-46. În acest model, singurul serviciu pe care nivelul rețea îl oferă nivelului transport este capacitatea de a injecta datagrame în subrețea în speranța că totul va merge bine. Nu există nici o noțiune de circuit virtual la nivelul rețea și uitați de concatenarea lor. Acest model nu necesită ca toate pachetele care aparțin unei conexiuni să traverseze aceeași secvență de porți. În fig. 5–46 sunt ilustrate datagramele de la gazda 1 pentru gazda 2, care urmează rute diferite prin rețeaua de interconectare. O decizie de dirijare este luată separat pentru fiecare pachet, eventual în funcție de traficul din momentul în care este trimis pachetul. Această strategie poate utiliza rute multiple și atinge astfel o capacitate mai mare decât modelul circuitelor virtuale concatenate. Pe de altă parte, nu există nici o garanție că pachetele ajung la destinație în ordine, presupunând că vor ajunge.

Modelul din fig. 5-46 nu este așa de simplu precum pare. Pe de o parte, dacă fiecare rețea are propriul protocol de nivel rețea, nu este posibil ca un pachet dintr-o rețea să tranziteze alta. Se pot imagina rutere multiprotocol care încearcă să convertească dintr-un format în altul, dar, în afara cazului în care cele două formate sunt strâns înrudite având aceleași câmpuri de informație, aceste conversii vor fi incomplete și deseori sortite eșecului. Din acest motiv, arareori se încearcă conversii.

O a doua problemă și mai serioasă este adresarea. Să ne imaginăm un caz simplu: o gazdă din Internet încearcă să trimită un pachet IP către o gazdă dintr-o rețea adiacentă SNA. Adresele IP și SNA sunt diferite. Ar trebui stabilită o corespondență între adresele IP și SNA și invers. În plus, ceea ce se poate adresa este diferit. În IP, gazdele (de fapt plăcile de interfață) au adrese. În SNA, pot avea adrese și alte entități în afară de gazde (de exemplu echipamente hardware). În cel mai bun caz, cineva ar trebui să mențină o bază de date a tuturor corespondențelor, dar ar fi o sursă constantă de necazuri.

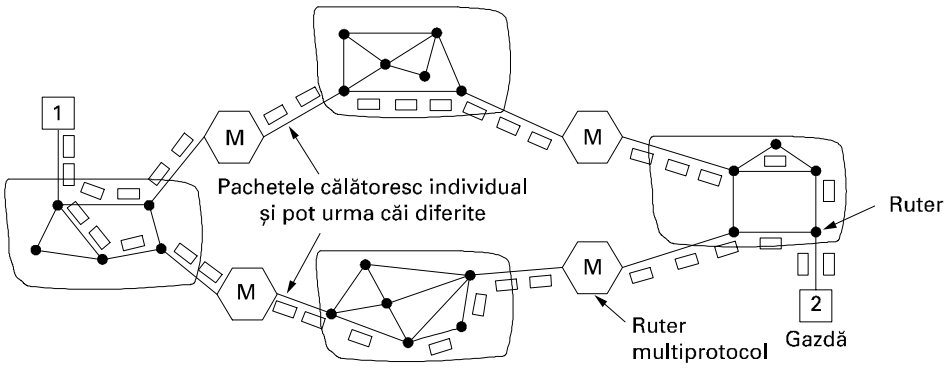


Fig. 5-46. O interconectare fără conexiuni.

O altă idee este de a proiecta un pachet universal „inter-rețea” și a obliga toate ruterele să-l recunoască. Această abordare este, de fapt, chiar IP-ul - un pachet proiectat ca să fie purtat prin mai multe rețele. Bineînțeles se poate întâmpla ca IPv4 (protocolul actual din Internet) să scoată de pe piață toate celelalte protocoale, IPv6 (viitorul protocol Internet) să nu prindă și să nu mai fie inventat nimic nou, dar istoria sugerează altceva. Este dificil ca toată lumea să fie de acord cu un singur format, atunci când companiile consideră că este în interesul lor să aibă formate proprii pe care să le controleze.

Să recapitulăm pe scurt cele două moduri prin care se poate aborda interconectarea rețelelor. Modelul circuitelor virtuale concatenate are, în esență, aceleași avantaje ca folosirea circuitelor virtuale într-o singură subrețea: zonele tampon pot fi rezervate în prealabil, poate fi garantată secvențialitatea, pot fi folosite antete scurte, iar necazurile cauzate de pachetele duplicate întârziate pot fi evitate.

El are, de asemenea, și dezavantaje: spațiul în tabele necesar în fiecare ruter pentru fiecare conexiune deschisă, lipsa unei dirijări alternative pentru a evita zonele congestionate și vulnerabilitatea la defectarea ruterelor de pe parcurs. De asemenea, are dezavantajul de a fi dificil, dacă nu imposibil, de implementat în cazul în care una din rețelele implicate este rețea nesigură de tip datagramă.

Proprietățile abordării interconectării rețelelor prin datagrame sunt în mare parte aceleași ca și cele ale subrețelelor de tip datagramă: potențial de congestionare mai mare, dar de asemenea potențial mai mare de adaptare la congestionări, robustețe în cazul defectării ruterelor și lungime mai mare necesară pentru antete. Într-o inter-rețea sunt posibili diferiți algoritmi de dirijare adaptivă, așa cum sunt în cadrul unei singure rețele de tip datagramă. Un avantaj major al abordării interconectării rețelelor prin datagrame este că aceasta poate fi folosită peste subrețele care nu folosesc circuite virtuale în interior. Multe LAN-uri, rețele mobile (de exemplu flotele aeriene și navale) și chiar unele WAN-uri intră în această categorie. Când o inter-rețea include una dintre acestea, apar probleme serioase dacă strategia de interconectare a rețelelor este bazată pe circuite virtuale.

5.5.5 Trecerea prin tunel

Rezolvarea cazului general de interconectare a două rețele diferite este extrem de dificilă. Cu toate acestea, există un caz special uzual care este gestionabil. Acest caz apare când gazdele sursă și destinație sunt în același tip de rețea, dar între ele există o rețea diferită. Ca exemplu, gândiți-vă la o bancă internațională cu o rețea TCP/IP bazată pe Ethernet la Paris, o rețea TCP/IP bazată pe Ethernet la Londra și o rețea non-IP de mare întindere (de exemplu ATM), așa cum este prezentat în fig. 5-47.

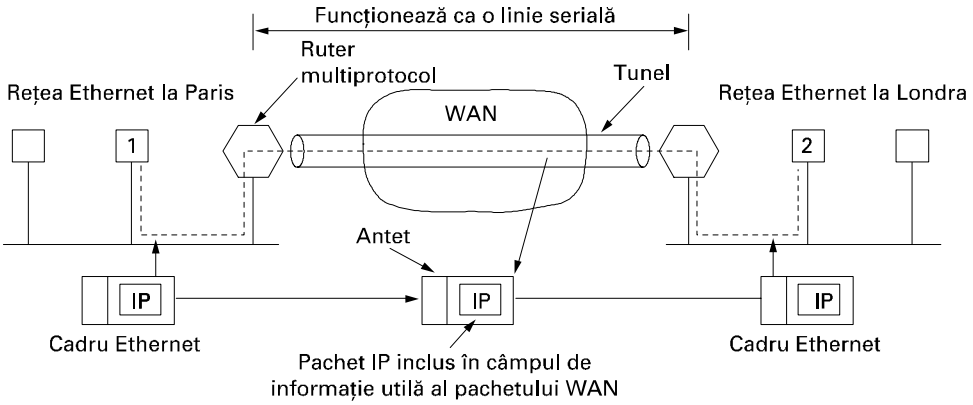


Fig. 5-47. Utilizarea tunelului pentru un pachet trimis de la Paris la Londra.

Soluția acestei probleme este o tehnică numită **trecerea prin tunele**. Pentru a trimite un pachet IP la gazda 2, gazda 1 construiește pachetul conținând adresa IP a gazdei 2, îl inserează într-un cadru Ethernet adresat ruterului multiprotocol parizian și apoi îl trimite în rețeaua Ethernet. Când ruterul multiprotocol primește cadrul, extrage pachetul IP, îl inserează în câmpul informație utilă al pachetului de nivel rețea WAN, pachet pe care îl adresează cu adresa ruterului multiprotocol londonez. Când ajunge acolo, ruterul londonez extrage pachetul IP și îl trimite gazdei 2 în interiorul unui cadru Ethernet.

WAN-ul poate fi văzut ca un mare tunel ce se întinde de la un ruter multiprotocol la altul. Pachetul IP doar traversează tunelul de la un capăt la altul, așezat confortabil în frumosul său lăcaș. El nu trebuie să aibă deloc grijă de comportarea la nivel WAN. Și nici gazdele din oricare Ethernet. Numai ruterul multiprotocol trebuie să înțeleagă și pachete IP și WAN. Ca rezultat, întreaga distanță de la mijlocul unui ruter multiprotocol până la mijlocul celuilalt acționează ca o linie serială.

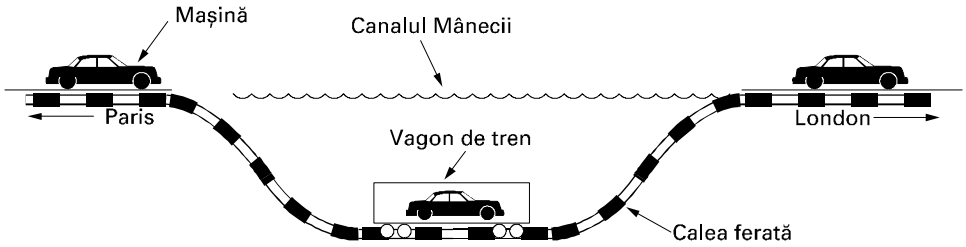


Fig. 5-48. Trecerea unui automobil din Franța în Anglia, prin tunel.

O analogie poate face utilizarea tunelelor mai clară. Considerați o persoană conducând mașina de la Paris la Londra. În Franța, mașina se deplasează sub acțiunea propriei puteri, dar când ajunge la Canalul Mânecii, este încărcată într-un tren de mare viteză și transportată în Anglia prin Chunnel[6] (mașinile nu pot circula prin Chunnel). Efectiv, mașina este purtată ca informație utilă, așa cum este prezentat în fig. 5-48. La capătul englez, mașina este eliberată pe drumurile engleze și continuă să se

[6] N.T. Chunnel, ce provine de la Channel (canal) și Tunnel (tunel) este denumirea dată în engleză tunelului de sub Canalul Mânecii.

deplaseze, din nou cu propria putere. Utilizarea tunelelor printr-o rețea necunoscută funcționează în același mod.

5.5.6 Dirijarea în rețele interconectate

Dirijarea printr-o rețea interconectată este similară cu dirijarea într-o singură subrețea, dar cu câteva complicații în plus. Să considerăm, de exemplu, interconectarea rețelelor din fig. 5-49(a) în care cinci rețele sunt conectate prin șase rutere (posibil multiprotocol). Crearea unui graf ca model al acestei situații este complicată de faptul că fiecare ruter poate accesa direct (mai clar, poate trimite pachete la) orice alt ruter conectat la orice rețea cu care este conectat. De exemplu, B din fig. 5-49(a) poate accesa direct A și C prin rețeaua 2 și de asemenea D prin rețeaua 3. Aceasta duce la graful din fig. 5-49(b).

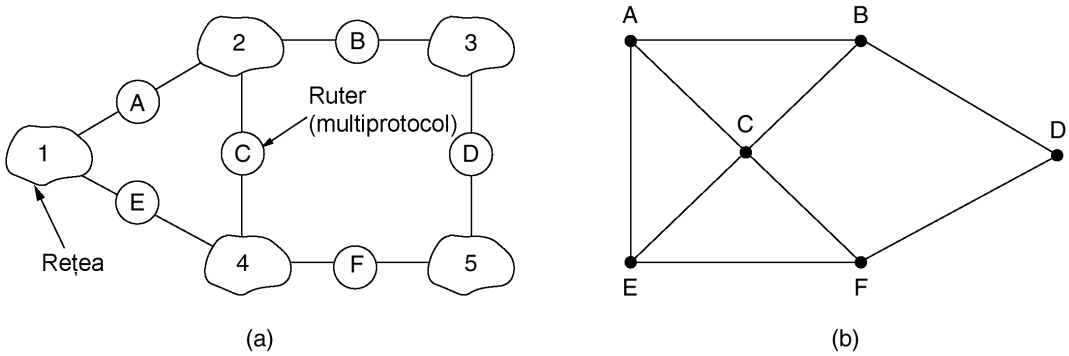


Fig. 5-49. (a) O interconectare de rețele. (b) Un graf al interconectării rețelelor.

O dată ce graful a fost construit, pe mulțimea de rutere multiprotocol pot fi aplicați algoritmii de dirijare cunoscuți, cum ar fi algoritmii de tip vectori distanță sau bazați pe starea legăturii. Aceasta duce la un algoritm de dirijare în doi pași: în interiorul fiecărei rețele se folosește un **protocol de poartă interioară (internal gateway protocol)**, dar între rețele se folosește un **protocol de poartă exterioară (exterior gateway protocol)** („poartă" este un termen mai vechi pentru „ruter„). De fapt, din moment ce fiecare rețea este independentă, ele pot folosi algoritmi diferiți. Deoarece fiecare rețea dintr-o interconectare de rețele este independentă de toate celelalte, este deseori referită ca un **sistem autonom (autonomous system-AS)**.

Un pachet tipic inter-rețele pornește din LAN-ul său adresat ruterului multiprotocol local (în antetul nivelului MAC). După ce ajunge acolo, codul de la nivelul rețea decide cărui ruter multiprotocol să-i trimită pachetul, folosind propriile tabele de dirijare. Dacă la acel ruter se poate ajunge folosind protocolul de rețea nativ al pachetului, atunci este trimis direct acelui ruter. Altfel, este trimis utilizând tunele, încapsulat în protocolul cerut de rețeaua intermediară. Acest proces este repetat până când pachetul ajunge în rețeaua destinație.

Una din diferențele dintre dirijarea inter-rețele și dirijarea intra-rețele este că dirijarea inter-rețele poate necesita deseori traversarea granițelor internaționale. Intră în joc legi diferite, cum ar fi legile suedeze despre secretul strict care se referă la exportarea din Suedia de date personale referitoare la cetățenii suedezi. Un alt exemplu este legea canadiană care spune că traficul de date care pornește din Canada și se termină în Canada nu poate părăsi țara. Această lege înseamnă că traficul

din Windsor, Ontario spre Vancouver nu poate fi dirijat prin apropiatul Detroit, chiar dacă această rută este cea mai rapidă și cea mai ieftină.

O altă diferență între dirijarea internă și cea externă este costul. Într-o singură rețea, se aplică în mod normal un singur algoritm de taxare. Cu toate acestea, diferite rețele pot avea administrații diferite și o cale poate fi mai ieftină decât alta. Similar, calitatea serviciului oferit de rețele diferite poate fi diferită și acesta poate fi un motiv pentru alegerea unei căi în defavoarea alteia.

5.5.7 Fragmentarea

Fiecare rețea impune o anumită dimensiune maximă pentru pachetele sale. Aceste limite au diferite cauze, printre care:

1. Hardware (de exemplu, dimensiunea unui cadru Ethernet).
2. Sistemul de operare (de exemplu, toate zonele tampon au 512 octeți).
3. Protocoale (de exemplu, numărul de biți din câmpul lungime al pachetului).
4. Concordanța cu unele standarde (inter)naționale.
5. Dorința de a reduce la un anumit nivel retransmisiile provocate de erori.
6. Dorința de a preveni ocuparea îndelungată a canalului de către un singur pachet.

Rezultatul tuturor acestor factori este că proiectanții de rețele nu au libertatea de a alege dimensiunea maximă a pachetelor oricum ar dori. Informația utilă maximă variază de la 8 octeți (celulele ATM) la 65515 octeți (pachetele IP), cu toate că dimensiunea pachetelor la nivelurile mai înalte este deseori mai mare.

O problemă evidentă apare când un pachet mare vrea să traverseze o rețea în care dimensiunea maximă a pachetului este prea mică. O soluție este să ne asigurăm din capul locului că problema nu apare. Cu alte cuvinte, inter-rețeaua trebuie să utilizeze un algoritm de dirijare care evită transmiterea pachetelor prin rețele în care pachetele nu pot fi manevrate. Cu toate acestea, această soluție nu este de fapt nici o soluție. Ce se întâmplă dacă pachetul sursă original este prea mare pentru a fi manevrat de rețeaua destinație? Algoritmul de dirijare nu are cum să evite destinația.

În esență, singura soluție a problemei este de a permite porților să spargă pachetele în **fragmente**, trimițând fiecare pachet ca un pachet inter-rețea separat. Cu toate acestea, așa cum știe orice părinte al unui copil mic, convertirea unui obiect mare în fragmente mici este considerabil mai ușoară decât procesul invers. (Fizicienii au dat chiar un nume acestui efect: legea a doua a termodinamicii.) Rețelele cu comutare de pachete au, de asemenea, probleme în îmbinarea fragmentelor.

Există două strategii opuse pentru reconstituirea pachetului original din fragmente. Prima strategie este de a face fragmentarea cauzată de o rețea cu „pachete mici” transparentă pentru toate rețelele succesive prin care pachetul trebuie să treacă pe calea către destinația finală. Această opțiune este prezentată în fig. 5-50(a). În această strategie, rețeaua cu pachete mici are porți (cel mai probabil, rutere specializate) către celelalte rețele. Când un pachet supradimensionat ajunge la poartă, poarta îl sparge în fragmente. Fiecare fragment este adresat aceleiași porți de ieșire, unde piesele sunt recombinate. În acest mod, trecerea printr-o rețea cu pachete mici a devenit transparentă. Rețelele următoare nici măcar nu sunt conștiente de fragmentarea făcută. Rețelele ATM, de exemplu, au hardware special pentru a oferi fragmentarea transparentă a pachetelor în celule și apoi reasamblarea celulelor în pachete. În lumea ATM, fragmentarea este numită segmentare; conceptul este același, dar diferă unele detalii.

Fragmentarea transparentă este simplă, dar are câteva probleme. Un motiv este că poarta de ieșire trebuie să știe când a primit toate piesele, așa încât în fiecare pachet trebuie inclus fie un câmp contor, fie un bit „sfârșit-de-pachet„. Un alt motiv este că toate pachetele trebuie să iasă prin aceeași poartă. Performanțele se pot degrada nepermițând ca unele pachete să urmărească o cale către destinația finală și alte pachete o cale diferită. O ultimă problemă este timpul suplimentar necesar pentru reasamblarea și apoi refragmentarea repetată a unui pachet mare care traversează o serie de rețele cu pachete mici. ATM necesită fragmentare transparentă.

Cealaltă strategie de fragmentare este de a nu recombina fragmentele la nici o poartă intermediară. O dată ce un pachet a fost fragmentat, fiecare fragment este tratat ca și cum ar fi un pachet original. Toate fragmentele sunt trecute printr-o poartă (sau porți) de ieșire, așa cum se arată în fig. 5-50(b). Recombinarea are loc doar la gazda destinație. Așa funcționează IP-ul.

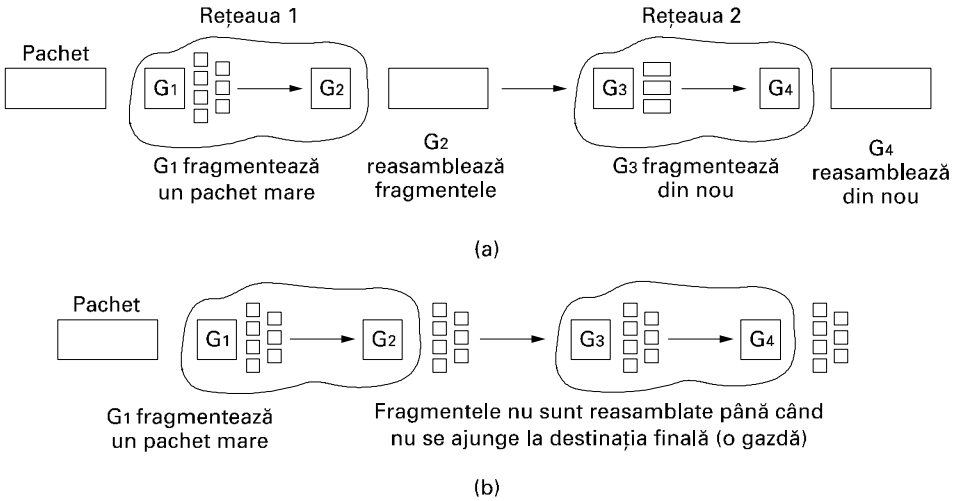


Fig. 5-50. (a) Fragmentare transparentă. (b) Fragmentare netransparentă.

Fragmentarea netransparentă are, de asemenea, unele probleme. De exemplu, necesită ca fiecare gazdă să fie capabilă să facă reasamblarea. Încă o problemă este că atunci când se fragmentează un pachet mare, supraîncărcarea crește, deoarece fiecare fragment trebuie să aibă un antet. Pe câtă vreme în prima metodă această supraîncărcare dispare de îndată ce se iese din rețeaua cu pachete mici, în această metodă supraîncărcarea rămâne pentru restul călătoriei. Cu toate acestea, un avantaj al acestei metode este că se pot folosi mai multe porți de ieșire și se pot obține performanțe mai bune. Desigur, dacă se folosește modelul circuitelor virtuale concatenate, acest avantaj nu este de nici un folos.

Când un pachet este fragmentat, fragmentele trebuie numerotate astfel încât șirul inițial de date să poată fi reconstituit. O metodă de numerotare a fragmentelor este bazată pe un arbore. Dacă pachetul 0 trebuie descompus, bucățile sunt numite 0.0, 0.1, 0.2 etc. Dacă aceste fragmente trebuie la rândul lor, să fie fragmentate mai târziu, bucățile sunt numerotate 0.0.0, 0.0.1, 0.0.2, ..., 0.1.0, 0.1.1, 0.1.2 etc. Dacă în antet au fost rezervate suficiente câmpuri pentru cel mai rău caz și nu sunt generate duplicate în altă parte, această schemă este suficientă pentru a asigura că toate bucățile pot fi corect reasamblate la destinație, neavând importanță ordinea în care ajung.

Cu toate acestea, dacă o singură rețea pierde sau elimină pachete, apare necesitatea unor retransmisii capăt-la-capăt, cu efecte nefericite pentru schema de numerotare. Să presupunem că un pa-

chet de 1024 biți este inițial fragmentat în patru fragmente de dimensiune egală, 0.0, 0.1, 0.2 și 0.3. Fragmentul 0.1 este pierdut, dar toate celelalte părți ajung la destinație. În cele din urmă, la sursă apare o depășire de timp și aceasta retransmite pachetul original. Numai că de această dată intervine legea lui Murphy și calea trece printr-o rețea cu limita de 512 octeți, deci sunt generate două fragmente. Când noul fragment 0.1 ajunge la destinație, receptorul va considera că toate cele patru bucăți au ajuns și va reconstrui pachetul incorect.

Un sistem de numerotare diferit (și mai bun) este ca protocolul de interconectare a rețelelor să definească o dimensiune de fragment elementar suficient de mică ca fragmentul elementar să poată trece prin orice rețea. Când un pachet este fragmentat, toate bucățile sunt egale cu dimensiunea fragmentului elementar, cu excepția ultimului, care poate fi mai scurt. Un pachet inter-rețea poate conține mai multe fragmente, din motive de eficiență. Antetul inter-rețea trebuie să ofere numărul original al pachetului și numărul fragmentului (sau primului fragment) elementar conținut în pachet. Ca de obicei, trebuie să existe un bit care să indice că ultimul fragment elementar conținut în pachetul inter-rețea este ultimul din pachetul original.

Această abordare necesită două câmpuri de secvență în antetul inter-rețea: numărul pachetului original și numărul fragmentului. Există clar un compromis între dimensiunea fragmentului elementar și numărul de biți din numărul fragmentului. Deoarece dimensiunea fragmentului elementar este presupusă a fi acceptabilă pentru toate rețelele, fragmentările ulterioare ale unui pachet inter-rețea conținând câteva fragmente nu cauzează probleme. În acest caz, limita extremă este reprezentată de un fragment elementar de un bit sau octet, numărul de fragment fiind reprezentat de deplasamentul bitului sau octetului în cadrul pachetului original, așa cum se arată în fig. 5-51.

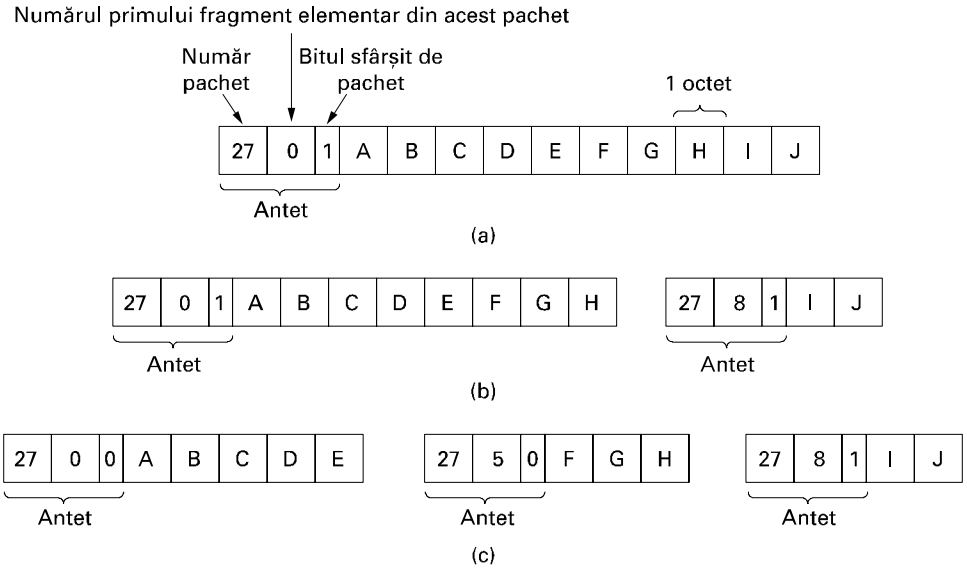


Fig. 5-51. Fragmentarea când dimensiunea datelor elementare este 1 octet. (a) Pachetul original, conținând 10 octeți de date. (b) Fragmente după trecerea printr-o rețea cu dimensiunea maximă a pachetului de 8 octeți. (c) Fragmente după trecerea printr-o poartă cu dimensiunea de 5.

Unele protocoale inter-rețea extind această metodă și consideră întreaga transmisie pe un circuit virtual ca fiind un pachet gigant, așa încât fiecare fragment conține numărul absolut de octet al primului octet din fragment.

5.6 NIVELUL REȚEA ÎN INTERNET

Înainte de a intra în detaliile nivelului rețea din Internet merită studiate principiile care au ghidat proiectarea lui în trecut și l-au făcut succesul care este astăzi. În ziua de azi, în prea multe cazuri oamenii par să le fi uitat. Aceste principii sunt enumerate și discutate în RFC 1958, care merită citit (și ar trebui să fie obligatoriu pentru toți proiectanții de protocoale – cu un examen final la sfârșit). Acest RFC se bazează în mare parte pe idei care se găsesc în (Clark, 19 și Saltzer, 1984). Vom rezuma ceea ce considerăm a fi cele 10 principii (de la cel mai important la cel mai puțin important).

1. **Fiți siguri că funcționează**. Nu finalizați proiectarea sau standardul până când mai multe prototipuri nu au comunicat cu succes unele cu altele. De prea multe ori proiectanții întâi scriu un standard de 1000 de pagini, primesc aprobarea pentru el și apoi descoperă că acest standard este eronat și nu funcționează. Apoi scriu versiunea 1.1 a standardului. Nu așa se face.
2. **Menține-l simplu**. Când există îndoieli, folosiți soluția mai simplă. William de Occam a enunțat acest principiu (briciul lui Occam) în secolul al 14-lea. Reformulat în termeni moderni: împotriviți-vă caracteristicilor suplimentare. Dacă o caracteristică nu este absolut necesară, nu o includeți, mai ales dacă același efect poate fi obținut prin combinarea altor caracteristici.
3. **Faceți alegeri clare**. Dacă există mai multe moduri de a realiza același lucru, alegeți unul. A avea două sau mai multe moduri de a rezolva un lucru înseamnă să se caute necazul cu lumânarea. Standardele conțin de obicei multe opțiuni sau moduri sau parametri pentru că anumite grupări importante susțin că modul lor este cel mai bun. Proiectanții ar trebui să reziste la această tendință. Spuneți nu.
4. **Exploatați modularitatea**. Acest principiu conduce direct la ideea de a avea stive de protocoale, fiecare nivel fiind independent de toate celelalte. În acest mod, dacă circumstanțele cer ca un modul sau nivel să fie schimbat, celelalte nu vor fi afectate.
5. **Așteptați-vă la medii eterogene**. În orice rețea mare vor apărea diferite tipuri de hardware, posibilități de transmisie și aplicații. Pentru a le trata pe toate, proiectarea rețelei trebuie să fie simplă, generală și flexibilă.
6. **Evitați opțiuni sau parametri statici**. Dacă un parametru nu poate fi evitat (de exemplu dimensiunea maximă a unui pachet), este mai bine ca transmițătorul și receptorul să negocieze o valoare decât să se definească valori fixe.
7. **Căutați o proiectare bună; nu este necesar să fie perfectă**. Deseori proiectanții au un proiect bun care nu poate trata un caz mai ciudat. Decât să strice tot proiectul, proiectanții ar trebui să-l păstreze și să transfere povara rezolvării acelui caz celor cu cerințe ciudate.

8. **Fiți stricți atunci când trimiteți și toleranți atunci când recepționați**. Cu alte cuvinte, trimiteți numai pachete care sunt în deplină conformitate cu standardul, dar așteptați-vă ca pachetele recepționate să nu fie în deplină conformitate cu standardul și încercați să le folosiți.
9. **Gândiți-vă la scalabilitate**. Dacă sistemul trebuie să suporte milioane de gazde și efectiv miliarde de utilizatori, nu se poate accepta nici un fel de bază de date centralizată, iar încărcarea trebuie distribuită cât mai uniform posibil folosind resursele disponibile.
10. **Luați în considerare performanțele și costurile**. Dacă o rețea are performanțe slabe sau prețuri exorbitante, nu o va folosi nimeni.

Să lăsăm acum principiile generale și să începem să studiem detaliile nivelului rețea din Internet. La nivelul rețea, Internet-ul poate fi văzut ca o colecție de subrețele sau **sisteme autonome** (eng.: **Autonomous Systems - AS**) care sunt interconectate. Nu există o structură reală, dar există câteva coloane vertebrale majore. Acestea sunt construite din linii de înaltă capacitate și rutere rapide. La coloanele vertebrale sunt atașate rețelele regionale (de nivel mediu), iar la aceste rețele regionale sunt atașate LAN-urile din multe universități, companii și furnizori de servicii Internet. O schiță a acestei organizări cvasi-ierarhice este dată în fig. 5-52.

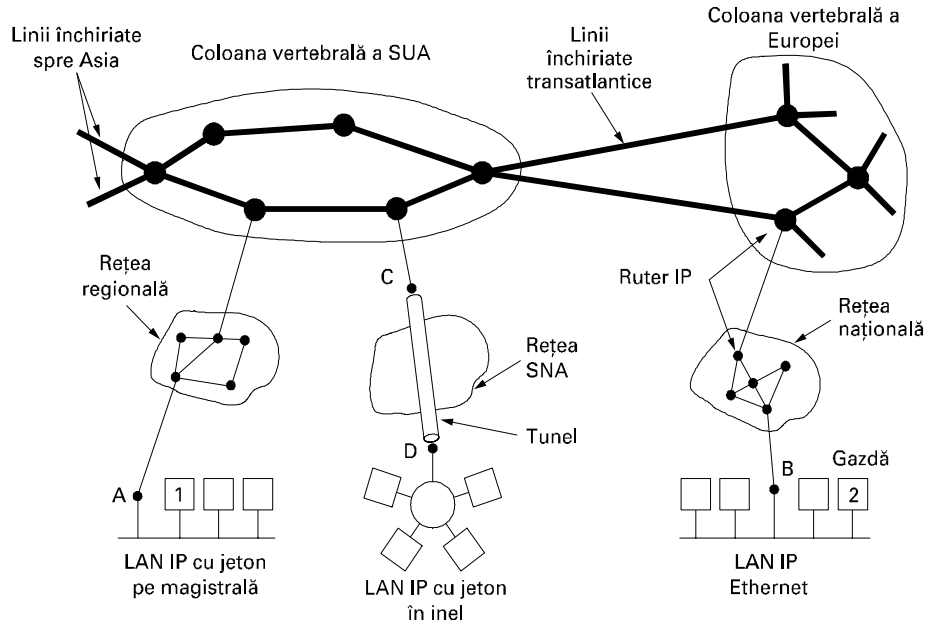


Fig. 5-52. Internet-ul este o colecție de multe rețele interconectate.

Liantul care ține Internet-ul la un loc este protocolul de nivel rețea, numit **IP** (**Internet Protocol** - protocolul Internet). Spre deosebire de protocoalele mai vechi de nivel rețea, acesta a fost proiectat de la început având în vedere interconectarea rețelelor. O metodă bună de a gândi nivelul rețea este aceasta. Sarcina lui este de a oferi cel mai bun mod posibil (adică negarantat) de a transporta datagrame de la sursă la destinație, fără a ține seama dacă aceste mașini sunt sau nu în aceeași rețea sau dacă între ele există sau nu alte rețele.

Comunicația în Internet funcționează după cum urmează. Nivelul transport preia șiruri de date și le sparge în datagrame. În teorie, datagramele pot avea fiecare până la 64 KB, dar în practică, ele nu depășesc 1500 octeți (pentru a intra într-un cadru Ehternet) . Fiecare datagramă este transmisă prin Internet, fiind eventual fragmentată în unități mai mici pe drum. Când toate bucățile ajung în sfârșit la mașina destinație, ele sunt reasamblate de nivelul rețea în datagrama originală. Datagrama este apoi pasată nivelului transport, care o inserează în șirul de intrare al procesului receptor. După cum se poate vedea în fig. 5-52 un pachet transmis de gazda 1 trebuie să traverseze șase rețele pentru a ajunge la gazda 2. În practică se trece de obicei prin mult mai mult decât șase rețele.

5.6.1 Protocolul IP

Un loc potrivit pentru a porni studiul nostru despre nivelul rețea în Internet este însuși formatul datagramelor IP. O datagramă IP constă dintr-o parte de antet și o parte de text. Antetul are o parte fixă de 20 de octeți și o parte opțională cu lungime variabilă. Formatul antetului este prezentat în fig. 5-53. El este transmis în ordinea *big endian* (cel mai semnificativ primul): de la stânga la dreapta, începând cu bitul cel mai semnificativ al câmpului *Versiune*. (Procesorul SPARC este de tip *big endian*; Pentium este de tip *little endian* - cel mai puțin semnificativ primul). Pe mașinile de tip *little endian*, este necesară o conversie prin program atât la transmisie cât și la recepție.

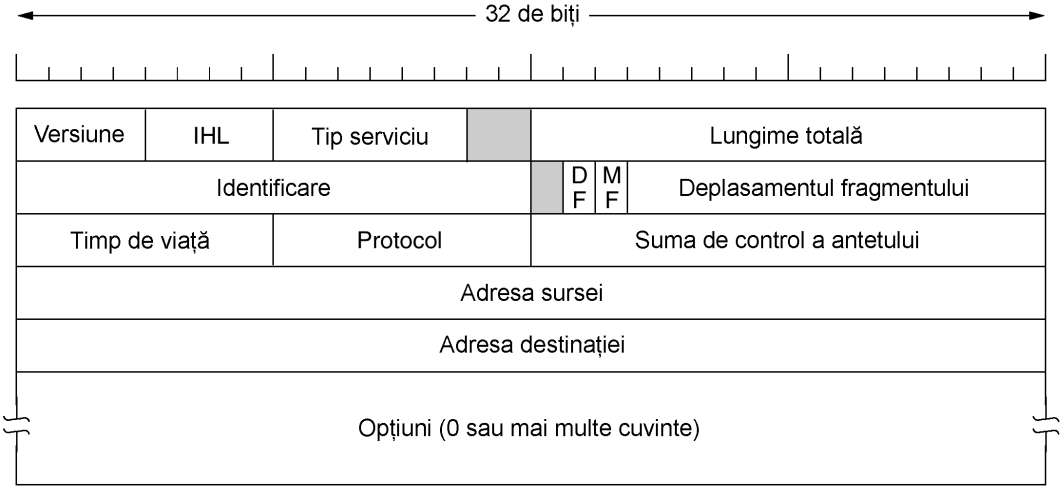


Fig. 5-53. Antetul IPv4 (protocolul Internet).

Câmpul *Versiune* memorează cărei versiuni de protocol îi aparține datagrama. Prin includerea unui câmp versiune în fiecare datagramă, devine posibil ca tranziția între versiuni să dureze ani de zile, cu unele mașini rulând vechea versiune, iar altele rulând-o pe cea nouă. La ora actuală are loc o tranziție de la IPv4 la IPv6, care deja durează de câțiva ani și în nici un caz nu s-a apropiat de final (Durand, 2001; Wilijakka, 2002; and Waddington and Chang 2002). Anumite persoane consideră chiar că nu se va întâmpla niciodată (Weiser, 2001). Referitor la numerotare, IPv5 a fost un protocol experimental de flux în timp real care nu a fost folosit pe scară largă.

Din moment ce lungimea antetului nu este constantă, un câmp din antet, *IHL*, este pus la dispoziție pentru a spune cât de lung este antetul, în cuvinte de 32 de octeți. Valoarea minimă este 5, care se aplică atunci când nu sunt prezente opțiuni. Valoarea maximă a acestui câmp de 4 biți este 15,

ceea ce limitează antetul la 60 de octeți și, astfel, câmpul de opțiuni la 40 de octeți. Pentru unele opțiuni, cum ar fi cea care înregistrează calea pe care a mers un pachet, 40 de octeți sunt mult prea puțini, făcând această opțiune nefolositoare.

Câmpul *Tip serviciu* este unul dintre puținele câmpuri care și-a schimbat sensul (oarecum) de-a lungul anilor. A fost și este în continuare menit să diferențieze diferitele clase de servicii. Sunt posibile diferite combinații de fiabilitate și viteză. Pentru vocea digitizată, livrarea rapidă are prioritate față de transmisia corectă. Pentru transferul de fișiere, transmisia fără erori este mai importantă decât transmisia rapidă.

La început, câmpul de 6 biți conținea (de la stânga la dreapta), un câmp *Precedență* de trei biți și trei indicatori, *D*, *T* și *R*. Câmpul *Precedență* reprezintă prioritatea, de la 0 (normal) la 7 (pachet de control al rețelei). Cei trei biți indicatori permiteau calculatorului gazdă să specifice ce îl afectează cel mai mult din mulțimea {Delay (Întârziere), Throughput (Productivitate), Reliability (Fiabilitate)}. În teorie, aceste câmpuri permit ruterelor să aleagă între, de exemplu, o legătură prin satelit cu o productivitate mare și o întârziere mare sau o linie dedicată cu o productivitate scăzută și o întârziere mică. În practică, ruterele curente ignoră adesea întregul câmp *Tip serviciu*.

Până la urmă, IETF a cedat și a modificat puțin câmpul pentru a-l adapta la servicii diferențiate. Șase dintre biți sunt folosiți pentru a indica căreia dintre clasele de servicii descrise mai devreme îi aparține pachetul. Aceste clase includ patru priorități de introducere în coadă, trei probabilități de respingere și clasele istorice.

Lungimea totală include totul din datagramă - atât antet cât și date. Lungimea maximă este de 65535 octeți. În prezent, această limită superioară este tolerabilă, dar în viitoarele rețele cu capacități de gigaocteți vor fi necesare datagrame mai mari. Câmpul *Identificare* este necesar pentru a permite gazdei destinație să determine cărei datagrame îi aparține un nou pachet primit. Toate fragmentele unei datagrame conțin aceeași valoare de *Identificare*.

Urmează un bit nefolosit și apoi două câmpuri de 1 bit. *DF* însemnă Don't Fragment (A nu se fragmenta). Acesta este un ordin dat ruterelor ca să nu fragmenteze datagrama pentru că destinația nu este capabilă să reasambleze piesele la loc. De exemplu, când un calculator pornește, memoria sa ROM poate cere să i se trimită o imagine de memorie ca o singură datagramă. Prin marcarea datagramei cu bitul *DF*, emițătorul știe că aceasta va ajunge într-o singură bucată, chiar dacă aceasta înseamnă că datagrama trebuie să evite o rețea cu pachete mai mici pe calea cea mai bună și să aleagă o rută suboptimală. Toate mașinile trebuie să accepte fragmente de 576 octeți sau mai mici.

MF însemnă More Fragments (mai urmează fragmente). Toate fragmentele, cu excepția ultimului, au acest bit activat. El este necesar pentru a ști când au ajuns toate fragmentele unei datagrame.

Deplasamentul fragmentului spune unde este locul fragmentului curent în cadrul datagramei. Toate fragmentele dintr-o datagramă, cu excepția ultimului, trebuie să fie un multiplu de 8 octeți - unitatea de fragmentare elementară. Din moment ce sunt prevăzuți 13 biți, există un maxim de 8192 de fragmente pe datagramă, obținându-se o lungime maximă a datagramei de 65536 octeți, cu unul mai mult decât câmpul *Lungime totală*.

Câmpul *Timp de viață* este un contor folosit pentru a limita durata de viață a pachetelor. Este prevăzut să contorizeze timpul în secunde, permițând un timp maxim de viață de 255 secunde. El trebuie să fie decrementat la fiecare salt (hop - trecere dintr-o rețea în alta) și se presupune că este decrementat de mai multe ori când stă la coadă un timp îndelungat într-un ruter. În practică, el contorizează doar salturile. Când ajunge la valoarea zero, pachetul este eliminat și se trimite înapoi la gazda sursă un pachet de avertisment. Această facilitate previne hoinăreala la infinit a datagramelor, ceea ce se poate întâmpla dacă tabelele de dirijare devin incoerente.

Când nivelul rețea a asamblat o datagramă completă, trebuie să știe ce să facă cu ea. Câmpul *Protocol* spune cărui proces de transport trebuie să o predea. TCP este o posibilitate, dar tot așa sunt și UDP și alte câteva. Numerotarea protocoalelor este globală la nivelul întregului Internet. Protocoalele și alte numere alocate erau anterior definite în RFC 1700, dar astăzi ele sunt conținute într-o bază de date on-line, care se găsește la adresa *www.iana.org*.

Suma de control a antetului verifică numai antetul. O astfel de sumă de control este utilă pentru detectarea erorilor generate de locații de memorie proaste din interiorul unui ruter. Algoritmul este de a aduna toate jumătățile de cuvinte, de 16 biți, atunci când acestea sosesc, folosind aritmetică în complement față de unu și păstrarea complementului față de unu al rezultatului. Pentru scopul acestui algoritm, se presupune că la sosire *suma de control a antetului* este zero. Acest algoritm este mai robust decât folosirea unei adunări normale. Observați că *suma de control a antetului* trebuie recalculată la fiecare salt, pentru că întotdeauna se schimbă cel puțin un câmp (câmpul *timp de viață*), dar se pot folosi trucuri pentru a accelera calculul.

Adresa sursei și *Adresa destinației* indică numărul de rețea și numărul de gazdă. Vom discuta adresele Internet în secțiunea următoare. Câmpul *Opțiuni* a fost proiectat pentru a oferi un subterfugiu care să permită versiunilor viitoare ale protocolului să includă informații care nu sunt prezente în proiectul original, pentru a permite cercetătorilor să încerce noi idei și pentru a evita alocarea unor biți din antet pentru informații folosite rar. Opțiunile sunt de lungime variabilă. Fiecare începe cu un cod de un octet care identifică opțiunea. Unele opțiuni sunt urmate de un câmp de un octet reprezentând lungimea opțiunii, urmat de unul sau mai mulți octeți de date. Câmpul *Opțiuni* este completat până la un multiplu de 4 octeți. Inițial erau definite cinci opțiuni, așa cum sunt listate în fig. 5-54, dar de atunci au mai fost adăugate și altele. Lista completă se găsește la adresa *www.iana.org/assignements/ip-parameters*.

Opțiune	Descriere
Securitate	Menționează cât de secretă este datagrama
Dirijare strictă de la sursă	Indică calea completă de parcurs
Dirijare aproximativă de la sursă	Indică o listă a ruterelor care nu trebuie sărite
Înregistrează calea	Determină fiecare ruter să-și adauge adresa IP
Amprentă de timp	Determină fiecare ruter să-și adauge adresa și o amprentă de timp.

Fig. 5-54. Unele dintre opțiunile IP.

Opțiunea *Securitate* menționează cât de secretă este informația. În teorie, un ruter militar poate folosi acest câmp pentru a menționa că nu se dorește o dirijare prin anumite țări pe care militarii le consideră a fi „băieții răi”. În practică, toate ruterele îl ignoră, deci singura sa funcție practică este să ajute spionii să găsească mai ușor lucrurile de calitate.

Opțiunea *Dirijare strictă de la sursă* dă calea completă de la sursă la destinație ca o secvență de adrese IP. Datagrama este obligată să urmărească această cale precisă. Ea este deosebit de utilă pentru administratorii de sistem pentru a trimite pachete de urgență atunci când tabelele de dirijare sunt distruse sau pentru a realiza măsurători de timp.

Opțiunea *Dirijare aproximativă de la sursă* cere ca pachetul să traverseze o listă specificată de rutere și în ordinea specificată, dar este permisă trecerea prin alte rutere pe drum. În mod normal, această opțiune ar putea oferi doar câteva rutere, pentru a forța o anumită cale. De exemplu, pentru a forța un pachet de la Londra la Sydney să meargă spre vest în loc de est, această opțiune poate specifica rutere în New York, Los Angeles și Honolulu. Această opțiune este foarte utilă atunci când motive politice sau economice dictează trecerea prin anumite țări sau evitarea lor.

Opțiunea *Înregistrează calea* indică ruterelor de pe cale să-și adauge adresele IP la câmpul opțiune. Aceasta permite administratorilor de sistem să localizeze pene în algoritmii de dirijare („De ce pachetele de la Houston la Dallas trec mai întâi prin Tokio?"). Când rețeaua ARPANET a fost înființată, nici un pachet nu trecea vreodată prin mai mult de nouă rutere, deci 40 de octeți pentru opțiuni au fost destui. Așa cum s-a menționat anterior, acum dimensiunea este prea mică.

În sfârșit, opțiunea *Amprentă de timp* este similară opțiunii *Înregistrează ruta*, cu excepția faptului că, în plus față de înregistrarea adresei sale de 32 de biți, fiecare ruter înregistrează și o amprentă de timp de 32 de biți. Și această opțiune este folosită în special pentru depanarea algoritmilor de dirijare.

5.6.2 Adrese IP

Fiecare gazdă și ruter din Internet are o adresă IP, care codifică adresa sa de rețea și de gazdă. Combinația este unică: în principiu nu există două mașini cu aceeași adresă IP. Toate adresele IP sunt de 32 de biți lungime și sunt folosite în câmpurile *Adresă sursă* și *Adresă destinație* ale pachetelor IP. Este important de observat că o adresă IP nu se referă de fapt la o gazdă. Se referă de fapt la o interfață de rețea, deci dacă o gazdă este în două rețele, trebuie să folosească două adrese IP. Totuși în practică, cele mai multe gazde sunt conectate la o singură rețea și deci au o adresă IP.

Timp de mai multe decenii, adresele IP erau împărțite în cinci categorii ilustrate în fig. 5-55. Acest model de alocare a fost denumit **clase de adrese**. Nu mai este folosit, dar referințele la acest model sunt în continuare des întâlnite în literatură. Vom discuta puțin mai târziu ce a înlocuit modelul claselor de adrese.

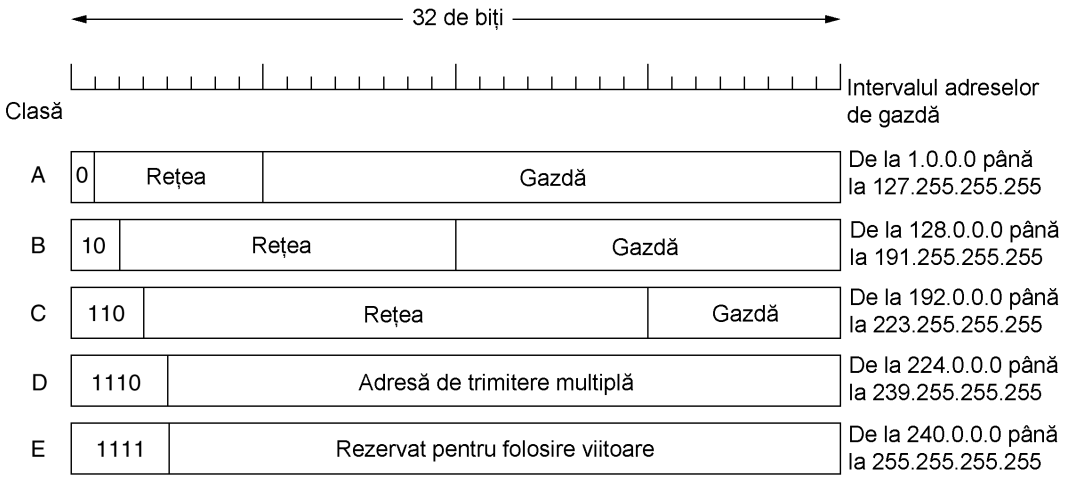


Fig. 5-55. Formatul adreselor IP.

Formatele de clasă A, B, C și D permit până la 128 rețele cu 16 milioane de gazde fiecare, 16.384 rețele cu până la 64K gazde, 2 milioane de rețele (de exemplu, LAN-uri) cu până la 256 gazde fiecare (deși unele dintre acestea sunt speciale). De asemenea este suportată și trimiterea multiplă (multicast), în care fiecare datagramă este direcționată mai multor gazde. Adresele care încep cu 1111 sunt rezervate pentru o folosire ulterioară. Peste 500 000 de rețele sunt conectate acum la Internet și numărul acestora crește în fiecare an. Pentru a evita conflictele numerele de rețea sunt atribuite de **ICANN (Internet Corporation for Assigned NAMES and Numbers** – Corporația Internet

pentru numere și nume atribuite). La rândul său, ICANN a împuternicit diverse autorități regionale să administreze părți din spațiul de adrese și acestea, la rândul lor, au împărțit adrese ISP-urilor și altor companii.

Adresele de rețea, care sunt numere de 32 de biți, sunt scrise în mod uzual în **notația zecimală cu punct**. În acest format, fiecare din cei 4 octeți este scris în zecimal, de la 0 la 255. De exemplu, adresa hexazecimală C0290614 este scrisă ca 192.41.6.20. Cea mai mică adresă IP este 0.0.0.0 și cea mai mare este 255.255.255.255.

Valorile 0 și -1 au semnificații speciale, așa cum se arată în fig. 5-56. Valoarea 0 înseamnă rețeaua curentă sau gazda curentă. Valoarea -1 este folosită ca o adresă de difuzare pentru a desemna toate gazdele din rețeaua indicată.

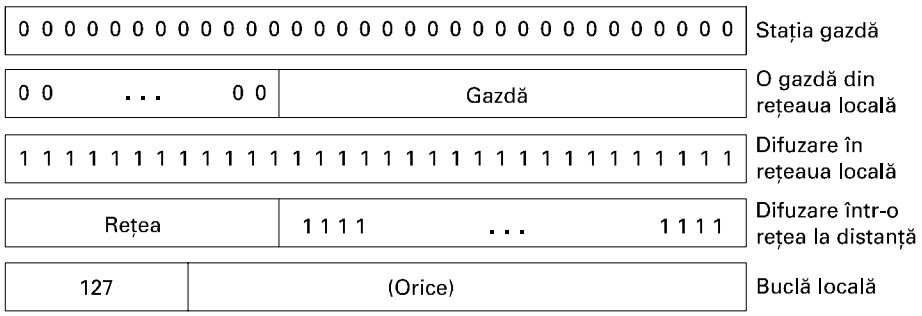


Fig. 5-56. Adrese IP speciale.

Adresa IP 0.0.0.0 este folosită de gazde atunci când sunt pornite. Adresele IP cu 0 ca număr de rețea se referă la rețeaua curentă. Aceste adrese permit ca mașinile să refere propria rețea fără a cunoaște numărul de rețea (dar ele trebuie să cunoască clasa adresei pentru a ști câte zerouri să includă). Adresele care constau numai din 1-uri permit difuzarea în rețeaua curentă, în mod uzual un LAN. Adresele cu un număr exact de rețea și numai 1-uri în câmpul gazdă permit mașinilor să trimită pachete de difuzare în LAN-uri la distanță, aflate oriunde în Internet (deși mulți administratori de sistem dezactivează această opțiune). În final, toate adresele de forma 127.xx.yy.zz sunt rezervate pentru testări în buclă locală (loopback). Pachetele trimise către această adresă nu sunt trimise prin cablu; ele sunt prelucrate local și tratate ca pachete sosite. Aceasta permite trimiterea pachetelor către rețeaua locală fără ca emițătorul să-i cunoască numărul.

Subrețele

Așa cum am văzut, toate gazdele dintr-o rețea trebuie să aibă același număr de rețea. Această proprietate a adresării IP poate crea probleme când rețeaua crește. De exemplu, să considerăm o universitate care a folosit la început o rețea de clasă B pentru calculatoarele din rețeaua Ethernet a Departamentului de Calculatoare. Un an mai târziu, departamentul de Inginerie Electrică a vrut acces la Internet, deci a cumpărat un repetor pentru a extinde Ethernetul Departamentului de Calculatoare în clădirea lor. Cu timpul, multe alte departamente au achiziționat calculatoare și limita de patru repetoare per Ethernet a fost rapid atinsă. Era nevoie de o altă organizare.

Achiziționarea altei adrese de rețea ar fi fost dificilă deoarece adresele sunt insuficiente și universitatea avea deja adrese suficiente pentru peste 60,000 de stații. Problema provine de la regula care afirmă că o singură adresă de clasă A, B sau C se referă la o singură rețea, nu la o mul-

țime de rețele. Cum tot mai multe organizații s-au lovit de această problemă, sistemul de adresare a fost puțin modificat pentru a o rezolva

Soluția este să se permită ca o rețea să fie divizată în mai multe părți pentru uz intern, dar pentru lumea exterioară să se comporte tot ca o singură rețea. În ziua de azi o rețea de campus tipică poate arăta ca în fig. 5-57, cu un ruter principal conectat la un ISP sau la rețeaua regională și numeroase rețele Ethernet împrăștiate prin campus în diferite departamente. Fiecare Ehternet are propriul ruter conectat la ruterul principal (posibil printr-un LAN coloană vertebrală, dar natura conexiunii interruter nu este relevantă aici)

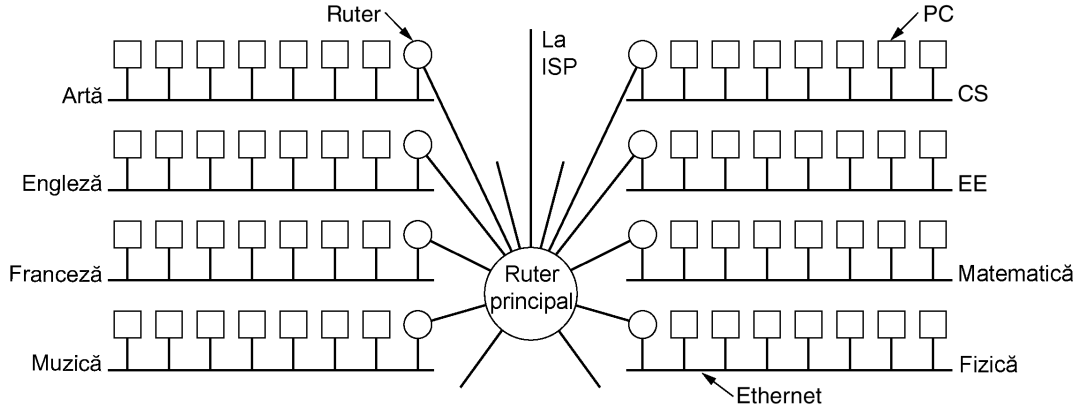


Fig. 5-57. O rețea de campus compusă din mai multe LAN-uri ale diferitelor departamente

În literatura Internet, aceste părți sunt numite **subrețele**. Așa cum am menționat în Cap. 1, această utilizare intră în conflict cu termenul „subrețea", care înseamnă mulțimea tuturor ruterelor și liniilor de comunicație dintr-o rețea. Din fericire, va fi clar din context care semnificație este atribuită. În această secțiune, și în următoarea definiția folosită va fi cea nouă.

Atunci când un pachet ajunge la ruterul principal, de unde știe acesta către ce subrețea (Ethernet) să-l trimită? O soluție ar fi să existe o tabelă cu 65,536 înregistrări în ruterul principal care să-i spună acestuia ce ruter să folosească pentru fiecare stație din campus. Această idee ar funcționa, dar ar fi nevoie de o tabelă foarte mare în ruterul principal și de multă muncă manuală de întreținere pe măsură ce stațiile ar fi adăugate, mutate sau scoase din uz.

În locul ei a fost inventată o altă soluție. În esență, în loc să existe o singură adresă de clasă B, cu 14 biți pentru numărul rețelei și 16 biți pentru gazdă, un număr de biți din numărul gazdei sunt folosiți pentru a crea un număr de subrețea. De exemplu, dacă o universitate are 35 de departamente, ar putea folosi un număr de subrețea de 6 biți și un număr de 10 biți pentru gazde, ceea ce ar permite un număr de 64 de rețele Ethernet, fiecare cu un maxim de 1022 de gazde (așa cum am menționat mai devreme, 0 și -1 nu sunt disponibile). Această împărțire ar putea fi modificată ulterior în caz că a fost o greșeală.

Pentru a se putea folosi subrețele, ruterul principal trebuie are nevoie de o **mască de subrețea,** care indică separarea dintre numărul rețea + subrețea și gazdă, așa cum este ilustrat în fig. 5-58. Măștile de subrețea sunt scrise de asemenea în notație zecimală cu punct, cu adăugarea unui caracter / (slash) urmat de numărul de biți din partea cu rețea + subrețea. Pentru exemplul din fig. 5-58, masca de subrețea poate fi scrisă ca 255.255.255.0 . O notație alternativă este /22 pentru a indica că masca subrețelei are lungimea de 22 de biți.

Fig. 5-58. O rețea de clasă B împărțită în 64 de subrețele

În afara rețelei, împărțirea în subrețele nu este vizibilă, astfel încât alocarea unei noi subrețele nu necesită contactarea ICANN sau schimbarea unor baze de date externe. În acest exemplu, prima subrețea poate folosi adrese IP începând de la 130.50.4.1, cea de-a doua poate începe de la 130.50.8.1, cea de-a treia poate începe de la 130.50.12.1 și așa mai departe. Pentru a se înțelege de ce numărul subrețelei crește cu patru, să observăm că adresele binare corespunzătoare sunt următoarele:

Subrețea 1 :	10000010	00110010	000001|00	00000001
Subrețea 2 :	10000010	00110010	000010|00	00000001
Subrețea 3 :	10000010	00110010	000011|00	00000001

Aici, bara verticală (|) arată granița dintre numărul subrețelei și numărul gazdei. La stânga se găsește numărul de 6 biți al subrețelei, la dreapta numărul de 10 biți al gazdei.

Pentru a vedea cum funcționează subrețelele, este necesar să explicăm cum sunt prelucrate pachetele IP într-un ruter. Fiecare ruter are o tabelă ce memorează un număr de adrese IP de forma (rețea, 0) și un număr de adrese IP de forma (această-rețea, gazdă). Primul tip indică cum se ajunge la rețelele aflate la distanță. Al doilea tip spune cum se ajunge la gazdele locale. Fiecărei tabele îi este asociată interfața de rețea care se folosește pentru a ajunge la destinație și alte câteva informații.

Când sosește un pachet IP, adresa destinație este căutată în tabela de dirijare. Dacă pachetul este pentru o rețea aflată la distanță, el este trimis ruterului următor prin interfața specificată în tabelă. Dacă este o gazdă locală (de exemplu în LAN-ul ruterului), pachetul este trimis direct către destinație. Dacă rețeaua nu este prezentă, pachetul este trimis unui ruter implicit care are tabele mai extinse. Acest algoritm înseamnă că fiecare ruter trebuie să memoreze numai rețele și gazde, nu perechi (rețea, gazdă), reducând considerabil dimensiunea tabelelor de dirijare.

Când este introdusă împărțirea în subrețele, tabelele de dirijare sunt schimbate, adăugând intrări de forma (această-rețea, subrețea, 0) și (această-rețea, această-subrețea, gazdă). Astfel un ruter din subrețeaua *k* știe cum să ajungă la toate celelalte subrețele și, de asemenea, cum să ajungă la toate gazdele din subrețeaua *k*. El nu trebuie să cunoască detalii referitoare la gazde din alte subrețele. De fapt, tot ceea ce trebuie schimbat este de a impune fiecărui ruter să facă un ȘI logic cu **masca de subrețea** a rețelei pentru a scăpa de numărul de gazdă și a căuta adresa rezultată în tabelele sale (după ce determină cărei clase de rețele aparține). De exemplu, asupra unui pachet adresat către 130.50.15.6 care ajunge la ruterul principal se face un ȘI logic cu masca de subrețea 255.255.252.0/22 pentru a obține adresa 130.50.12.0. Această adresă este căutată în tabelele de dirijare pentru a se găsi cum se ajunge la gazdele din subrețeaua 3. Ruterul din subrețeaua 5 este astfel ușurat de munca de a memora toate adresele de nivel legătură de date ale altor gazde decât cele din subrețeaua 5. Împărțirea în subrețele reduce astfel spațiul tabelelor de dirijare prin crearea unei ierarhii pe trei niveluri, alcătuită din rețea, subrețea și gazdă.

CIDR - Dirijarea fără clase între domenii

IP este folosit intens de câteva decenii. A funcționat extrem de bine, așa cum a fost demonstrat de creșterea exponențială a Internet-ului. Din nefericire, IP devine rapid o victimă a propriei popularități: își epuizează adresele. Acest dezastru fantomatic a generat foarte multe discuții și controverse în cadrul comunității Internet referitor la cum să fie tratat. În această secțiune vom descrie atât problema cât și câteva soluții propuse.

Prin 1987, câțiva vizionari au prezis că într-o zi Internet-ul poate crește până la 100.000 de rețele. Cei mai mulți experți s-au exprimat cu dispreț că aceasta va fi peste decenii, dacă va fi vreodată. Cea de-a 100.000-a rețea a fost conectată în 1996. Problema, expusă anterior, este că Internet-ul își epuizează rapid adresele IP. În principiu, există peste 2 miliarde de adrese, dar practica organizării spațiului de adrese în clase (vezi fig. 5-55) irosește milioane din acestea. În particular, adevărații vinovați sunt de rețelele de clasă B. Pentru cele mai multe organizații, o adresă de clasă A, cu 16 milioane de adrese este prea mare, iar o rețea de clasă C, cu 256 de adrese, este prea mică. O rețea de clasă B, cu 65.536 adrese, este numai bună. În folclorul Internet, această situație este cunoscută ca **problema celor trei urși** (ca din *Goldilocks and the Three Bears*).

În realitate, o adresă de clasă B este mult prea mare pentru cele mai multe organizații. Studiile au arătat că mai mult de jumătate din toate rețelele de clasă B au mai puțin de 50 de gazde. O rețea de clasă C ar fi fost suficientă, dar fără îndoială că fiecare organizație care a cerut o adresă de clasă B a crezut că într-o zi ar putea depăși câmpul gazdă de 8 biți. Privind retrospectiv, ar fi fost mai bine să fi avut rețele de clasă C care să folosească 10 biți în loc de opt pentru numărul de gazdă, permițând 1022 gazde pentru o rețea. În acest caz, cele mai multe organizații s-ar fi decis, probabil, pentru o rețea de clasă C și ar fi existat jumătate de milion dintre acestea (comparativ cu doar 16.384 rețele de clasă B).

Este greu să fie învinuiți proiectanții Internetului pentru că nu au pus la dispoziție mai multe adrese de clasă B (și mai mici). În momentul în care s-a luat decizia să fie create cele trei clase, Internetul era o rețea de cercetare care conecta marile universități din SUA (plus un număr mic de companii și sit-uri militare care făceau cercetări în domeniul rețelelor). Pe atunci nimeni nu a perceput Internetul ca fiind un sistem de comunicație în masă care va rivaliza cu rețeaua telefonică. Fără îndoială că în acel moment cineva a spus: „SUA are aproximativ 2000 de universități și colegii. Chiar dacă toate se conectează la Internet și se mai conectează și multe universități din alte țări, nu o să ajungem niciodată la 16.000 pentru că nu există atâtea universități în întreaga lume. În plus faptul că numărul gazdei este un număr întreg de octeți, mărește viteza de prelucrare a pachetelor."

Totuși, dacă s-ar fi alocat 20 de biți numărului de rețea pentru rețele de clasă B, ar fi apărut o altă problemă: explozia tabelelor de dirijare. Din punctul de vedere al ruterelor, spațiul de adrese IP este o ierarhie pe două niveluri, cu numere de rețea și numere de gazde. Ruterele nu trebuie să știe despre toate gazdele, dar ele trebuie să știe despre toate rețelele. Dacă ar fi fost în folosință jumătate de milion de rețele de clasă C, fiecare ruter din întregul Internet ar fi necesitat o tabelă cu o jumătate de milion de intrări, una pentru fiecare rețea, spunând care linie se folosește pentru a ajunge la respectiva rețea, împreună cu alte informații.

Memorarea fizică efectivă a tabelelor cu jumătate de milion de intrări este probabil realizabilă, deși costisitoare pentru ruterele critice care trebuie să mențină tabelele în RAM static pe plăcile I/O. O problemă mai serioasă este reprezentată de creșterea complexității diferiților algoritmi referitori la gestiunea tabelelor, care este mai rapidă decât liniară. Și mai rău, o mare parte din programele și firmware-ul ruterelor existente a fost proiectat pe vremea când Internet-ul avea 1000 de rețele co-

nectate și 10.000 de rețele păreau la depărtare de decenii. Deciziile de proiectare făcute atunci sunt departe de a fi optime acum.

În plus, diferiți algoritmi de dirijare necesită ca fiecare ruter să-și transmită tabelele periodic (de exemplu protocolul vectorilor distanță). Cu cât tabelele sunt mai mari, cu atât este mai probabil ca anumite părți să se piardă pe drum, ducând la date incomplete la celălalt capăt și, posibil, la instabilități de dirijare.

Problema tabelelor de dirijare ar fi putut fi rezolvată prin adoptarea unei ierarhii mai adânci. De exemplu, ar fi mers dacă s-ar fi impus ca fiecare adresă să conțină un câmp țară, județ/provincie, oraș, rețea și gazdă. Atunci fiecare ruter ar fi trebuit să știe doar cum să ajungă la fiecare țară, la județele sau provinciile din țara sa, orașele din propriul județ sau provincie și rețelele din orașul său. Din nefericire, această soluție ar necesita mai mult de 32 de biți pentru adrese IP și ar folosi adresele ineficient (Liechtenstein ar fi avut tot atâția biți ca Statele Unite).

Pe scurt, cele mai multe soluții rezolvă o problemă, dar creează o alta. Soluția care a fost implementată acum și care a dat Internet-ului un pic de spațiu de manevră este **CIDR** (**Classless InterDomain Routing** - Dirijarea fără clase între domenii). Ideea de la baza CIDR, descrisă în RFC 1519, este de a aloca adresele IP rămase, în blocuri de dimensiune variabilă, fără a se ține cont de clase. Dacă un sit are nevoie, să zicem, de 2000 de adrese, îi este dat un bloc de 2048 adrese la o graniță de 2048 de octeți.

Renunțarea la clase face rutarea mai complicată. În vechiul sistem cu clase, rutarea se efectua în felul următor. Atunci când un pachet ajungea la un ruter, o copie a adresei IP era deplasată la dreapta cu 28 de biți pentru a obține un număr de clasă de 4 biți. O ramificație cu 16 căi sorta pachetele în A, B, C și D (dacă era suportat), cu 8 cazuri pentru clasa A, patru cazuri pentru clasa B, două cazuri pentru clasa C și câte unul pentru D și E. Programul pentru fiecare clasă masca numărul de rețea de 8, 16 sau 24 de biți și îl alinia la dreapta într-un cuvânt de 32 de biți. Numărul de rețea era apoi căutat în tabela pentru A, B sau C, de obicei indexat pentru rețelele A și B și folosind dispersia (hashing) pentru rețelele C. Odată intrarea găsită, se putea găsi linia de ieșire și pachetul era retransmis.

Cu CIDR, acest algoritm simplu nu mai funcționează. În loc de aceasta, fiecare intrare din tabela de rutare este extinsă cu o mască de 32 de biți. Din acest motiv, acum există o singură tabelă de rutare pentru toate rețelele, constituită dintr-un un vector de triplete (adresă IP, mască subrețea, linie de ieșire). Când sosește un pachet IP, mai întâi se extrage adresa IP a destinației. Apoi (conceptual) tabela de rutare este scanată intrare cu intrare, mascând adresa destinație și comparând cu intrarea din tabelă, în căutarea unei potriviri. Este posibil ca mai multe intrări (cu măști de subrețea de lungimi diferite) să se potrivească, caz în care este folosită cea mai lungă mască. Astfel, dacă există potrivire pentru o mască /20 și pentru o mască /24, este folosită intrarea cu /24.

Pentru a accelera procesul de găsire a adreselor au fost imaginați algoritmi complecși (Ruiz-Sanchez et al., 2001). Ruterele comerciale folosesc chip-uri VLSI proprietare cu acești algoritmi implementați în hardware.

Pentru a face algoritmul de retransmitere mai ușor de înțeles, să considerăm un exemplu în care sunt disponibile milioane de adrese, începând de la 194.24.0.0. Să presupunem că Universitatea Cambridge are nevoie de 2048 de adrese și are atribuite adresele de la 194.24.0.0 până la 194.24.7.255, împreună cu masca 255.255.248.0. Apoi, Universitatea Oxford cere 4096 de adrese. Deoarece un bloc de 4096 de adrese trebuie să fie aliniat la o frontieră de 4096 octeți, acestea nu pot fi numerotate începând de la 194.8.0.0. În loc, se primesc adrese de la 194.24.16.0 până la 194.24.31.255, împreună cu o mască de 255.255.240.0. Acum Universitatea din Edinburgh cere

1024 de adrese și îi sunt atribuite adresele de la 194.24.8.0 până la 194.24.11.255 și masca 255.255.252.0. Alocările sunt rezumate în fig. 5-59.

Universitatea	Prima adresă	Ultima adresă	Număr adrese	Notație
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(disponibil)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Fig. 5-59. Un set de atribuiri de adrese IP.

Tabelele de dirijare din toată lumea sunt acum actualizate cu cele trei intrări atribuite. Fiecare intrare conține o adresă de bază și o mască de subrețea. Aceste intrări (în binar) sunt:

	Adresă	Mască
C	11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E	11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000
C	11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

Să vedem acum ce se întâmplă când sosește un pachet adresat pentru 194.24.17.4, care este reprezentat ca următorul șir de 32 de biți :

11000010 00011000 00010001 00000100

Mai întâi, se face un ȘI logic cu masca de la Cambridge pentru a obține:

11000010 00011000 00010000 00000000

Această valoare nu se potrivește cu adresa de bază de la Cambridge, așa că adresei originale i se aplică un ȘI logic cu masca de la Edinburgh pentru a se obține:

11000010 00011000 00010000 00000000

Această valoare nu se potrivește cu adresa de bază de la Edinburgh, așa că se încearcă în continuare Oxford, obținându-se:
11000010 00011000 00010000 00000000

Această valoare se potrivește cu adresa de bază de la Oxford. Dacă în restul tabelei nu sunt găsite potriviri mai lungi, atunci este folosită intrarea pentru Oxford și pachetul va fi trimis pe linia corespunzătoare.

Acum să privim cele trei universități din punctul de vedere al unui ruter din Omaha, Nebraska, care are doar patru linii de ieșire: Minneapolis, New York, Dallas și Denver. Atunci când software-ul ruterului primește cele trei noi intrări, realizează că le poate combina pe toate trei într-o singură **intrare agregată** (eng,: **aggregate entry**) 194.24.0.0/19 cu adresa binară și o masca de subrețea următoare :

11000010 00000000 00000000 00000000	11111111 11111111 11100000 00000000

Această intrare trimite toate pachetele destinate uneia dintre cele trei universități la New York. Reunind cele trei intrări ruterul din Omaha și-a redus mărimea tabelei de rutare cu doi.

Dacă New York-ul are o singură conexiune cu Londra pentru tot traficul spre Marea Britanie, poate folosi de asemenea o intrare agregată. Totuși, dacă are două conexiuni separate pentru Londra și Edinburgh, atunci trebuie să aibă trei intrări.

Ca o notă finală despre acest exemplu, intrarea agregată a ruterului din Omaha va trimite și pachetele către adresele nealocate tot spre New York. Cât timp adresele sunt cu adevărat nealocate, aceasta nu contează pentru că așa ceva nu ar trebui să se întâmple. Totuși, dacă mai târziu sunt alocate unei companii din California, pentru a le trata va fi nevoie de o înregistrare separată, 194.24.12.0/22.

NAT – Translatarea adreselor de rețea

Adresele IP sunt insuficiente. Un ISP ar putea avea o adresă /16 (anterior de clasă B) oferindu-i 65.534 numere de stații. Dacă are mai mulți clienți, are o problemă. Pentru utilizatorii casnici cu conexiuni pe linie telefonică (eng.: dial-up), o soluție ar fi să se aloce dinamic o adresă IP fiecărui calculator în momentul în care sună și se conectează și să o dealoce în momentul în care se termină sesiunea. În acest fel o singură adresă /16 poate fi folosită pentru 65.534 utilizatori activi, ceea ce este probabil suficient pentru un ISP cu mai multe sute de mii de utilizatori. În momentul în care sesiunea se termină, adresa IP este alocată altui calculator care sună. Deși această strategie funcționează bine pentru un ISP cu un număr moderat de utilizatori casnici, eșuează pentru ISP-uri care deservesc preponderent companii.

Problema este că o companie se așteaptă să fie on-line în mod continuu în timpul orelor de program. Atât companiile mici, cum ar fi o companie de turism compusă din trei persoane, cât și mari corporații au mai multe calculatoare conectate de un LAN. Unele sunt calculatoarele personale ale angajaților; altele pot fi servere Web. În general în LAN există un ruter care este conectat cu ISP-ul printr-o linie închiriată pentru a avea conexiune continuă. Această situație impune ca fiecărui calculator să îi fie asociat un IP propriu pe parcursul întregii zile. De fapt, numărul total al calculatoarelor companiilor care sunt clienți ai ISP-ului nu poate depăși numărul de adrese IP pe care le posedă acesta. Pentru o adresă /16, aceasta limitează numărul total de calculatoare la 65.534. Pentru un ISP cu zeci de mii de companii cliente, această limită va fi repede depășită.

Pentru a agrava situația, din ce în ce mai mulți utilizatori se abonează de acasă la ADSL sau Internet prin cablu. Două dintre facilitățile oferite de aceste servicii sunt (1) utilizatorul primește o adresă IP permanentă și (2) nu există taxă de conectare (doar o taxă lunară), așa că mulți utilizatori ai ADSL-ului și ai Internetului prin cablu rămân conectați permanent. Aceasta contribuie la insuficiența adreselor IP. Alocarea dinamică a adreselor IP așa cum se face cu utilizatorii dial-up nu este utilă pentru că numărul adreselor IP folosite la un moment dat poate fi de multe ori mai mare decât numărul adreselor deținute de ISP.

Și pentru a complica și mai mult lucrurile, mulți utilizatori de ADSL și Internet prin cablu au două sau mai multe calculatoare acasă, deseori câte unul pentru fiecare membru al familiei, și toți vor să fie online tot timpul folosind singura adresă IP care le-a fost alocată de către ISP. Soluția este să se conecteze toate PC-urile printr-un LAN și să se pună un ruter. Din punctul de vedere al ISP-ului, familia este acum ca o companie cu câteva calculatoare. Bine-ați venit la Jones, Inc.

Problema epuizării adreselor IP nu este o problemă teoretică care ar putea apărea cândva în viitorul îndepărtat. A apărut aici și acum. Soluția pe termen lung este ca tot Internetul să migreze la IPv6, care are adrese de 128 de biți. Tranziția se desfășoară încet, dar vor trece ani până la finalizarea procesului. În consecință, anumite persoane au considerat că este nevoie de o rezolvare rapidă pe termen scurt. Rezolvarea a venit în forma NAT (Network Address Translation –

Translatarea adreselor de rețea), care este descrisă în RFC 3022 și pe care o vom rezuma mai jos. Pentru informații suplimentare consultați (Dutcher, 2001).

Ideea de bază a NAT-ului este de a aloca fiecărei companii o singură adresă IP (sau cel mult un număr mic de adrese) pentru traficul Internet. În interiorul companiei, fiecare calculator primește o adresă IP unică, care este folosită pentru traficul intern. Totuși, atunci când un pachet părăsește compania și se duce la ISP, are loc o translatare de adresă. Pentru a face posibil acest lucru, au fost declarate ca fiind private trei intervale de adrese IP. Companiile le pot folosi intern așa cum doresc. Singura regulă este ca nici un pachet conținând aceste adrese să nu apară pe Internet. Cele trei intervale rezervate sunt :

10.0.0.0	- 10.255.255.255/8	(16.777.216 gazde)
172.16.0.0	- 172.31.255.255/12	(1 .048.576 gazde)
192.168.0.0	- 192.168.255.255/16	(65.536 gazde)

Primul interval pune la dispoziție 16.777.216 adrese (cu excepția adreselor 0 și –1, ca de obicei) și este alegerea obișnuită a majorității companiilor, chiar dacă nu au nevoie de așa de multe adrese.

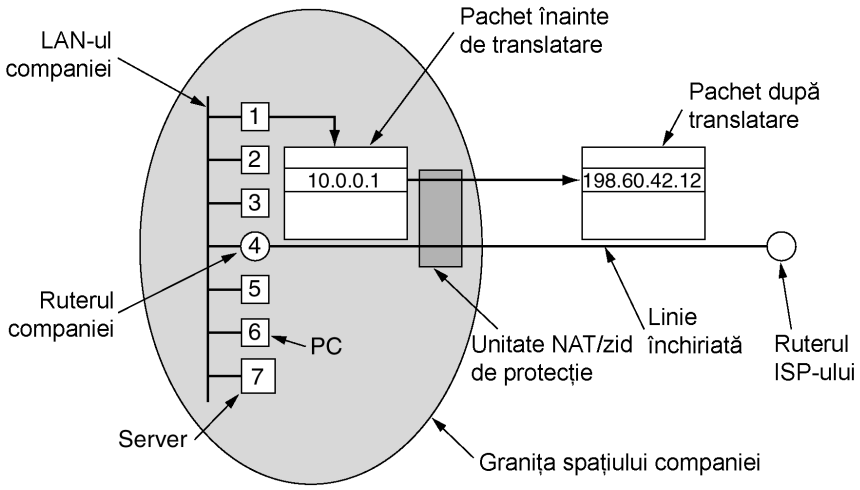


Fig. 5-60. Amplasarea și funcționarea unei unități NAT.

Funcționarea NAT este ilustrată în fig. 5-60. În interiorul companiei fiecare mașină are o adresă unică de forma 10.x.y.z. Totuși, când un pachet părăsește compania, trece printr-o **unitate NAT** (eng.: **NAT box**)care convertește adresa IP internă, 10.0.0.1 în figură, la adresa IP reală a companiei, 198.60.42.12, în acest exemplu. Unitatea NAT este deseori combinată într-un singur dispozitiv cu un zid de protecție (eng.: firewall), care asigură securitatea controlând cu grijă ce intră și iese din companie. Vom studia zidurile de protecție în Cap. 8. De asemenea este posibilă integrarea unității NAT în ruterul companiei.

Până acum am trecut cu vederea un detaliu: atunci când vine răspunsul (de exemplu de la un server Web), este adresat 198.60.42.12, deci de unde știe unitatea NAT cu care adresă să o înlocuiască pe aceasta? Aici este problema NAT-ului. Dacă ar fi existat un câmp liber în antetul IP, acel câmp ar fi putut fi folosit pentru a memora cine a fost adevăratul emițător, dar numai 1 bit este încă nefolosit. În principiu, o nouă opțiune ar putea fi creată pentru a reține adevărata adresă a

sursei, dar acest lucru ar necesita schimbarea codului din protocolul IP de pe toate stațiile din întregul Internet pentru a putea prelucra noua opțiune. Aceasta nu este o alternativă promițătoare pentru o rezolvare rapidă.

În cele ce urmează se prezintă ceea ce s-a întâmplat cu adevărat. Proiectanții NAT au observat că marea majoritate a pachetelor IP aveau conținut TCP sau UDP. Atunci când vom studia protocoalele TCP și UDP în Cap. 6, vom vedea că ambele au antete conțin un port sursă și un port destinație. În continuare vom discuta doar despre porturi TCP, dar exact aceleași lucruri se aplică și la UDP. Porturile sunt numere întregi pe 16 biți care indică unde începe și unde se termină o conexiune TCP. Aceste câmpuri pun la dispoziție câmpul necesar funcționării NAT.

Atunci când un proces dorește să stabilească o conexiune TCP cu un proces de la distanță, se atașează unui port TCP nefolosit de pe mașina sa. Acesta se numește **portul sursă** și spune codului TCP unde să trimită pachetele care vin și aparțin acestei conexiuni. Procesul furnizează și un **port destinație** pentru a spune cui să trimită pachetele în cealaltă parte. Porturile 0-1023 sunt rezervate pentru servicii bine cunoscute. De exemplu portul 80 este folosit de serverele Web, astfel încât clienții să le poată localiza. Fiecare mesaj TCP care pleacă conține atât un port sursă cât și un port destinație. Cele două porturi permit identificarea proceselor care folosesc conexiunea la cele două capete.

S-ar putea ca o analogie să clarifice mai mult utilizarea porturilor. Imaginați-vă o companie cu un singur număr de telefon principal. Atunci când cineva sună la acest număr, vorbește cu un operator care întreabă ce interior dorește și apoi face legătura la acel interior. Numărul principal este analog cu adresa IP a companiei iar interioarele de la ambele capete sunt analoage cu porturile. Porturile reprezintă 16 biți suplimentari de adresare identificând procesul care primește un pachet sosit.

Folosind câmpul *Port sursă* rezolvăm problema de corespondență. De fiecare dată când un pachet pleacă, el intră în unitatea NAT și adresa sursă 10.x.y.z este înlocuită cu adresa reală a companiei. În plus, câmpul TCP *Port sursă* este înlocuit cu un index în tabela de translatare a unității NAT, care are 65.536 intrări. Această tabelă conține adresa IP inițială și portul inițial. În final, sunt recalculate și inserate în pachet sumele de control ale antetelor IP și TCP. Câmpul *Port sursă* trebuie înlocuit pentru că s-ar putea întâmpla, de exemplu, ca stațiile 10.0.0.1 și 10.0.0.2 să aibă amândouă conexiuni care să folosească portul 5000, deci câmpul *Port sursă* nu este suficient pentru a identifica procesul emițător.

Atunci când la unitatea NAT sosește un pachet de la ISP, *Portul sursă* din antetul TCP este extras și folosit ca index în tabela de corespondență a unității NAT. Din intrarea localizată sunt extrase și inserate în pachet adresa IP internă și *Portul sursă* TCP inițial. Apoi sunt recalculate sumele de control TCP și IP și inserate în pachet. După aceea pachetul este transferat ruterului companiei pentru transmitere normală folosind adresa 10.x.y.z.

NAT poate fi folosit pentru rezolva insuficiența adreselor IP pentru utilizatori de ADSL și Internet prin cablu. Atunci când un ISP alocă fiecărui utilizator o adresă, folosește adrese 10.x.y.z. Atunci când pachete de la stațiile utilizatorilor ies din ISP și intră în Internet trec printr-o cutie NAT care le translatează în adresele Internet adevărate ale ISP-ului. La întoarcere pachetele trec prin maparea inversă. Din această perspectivă, pentru restul Internetului, ISP-ul și utilizatorii săi ADSL/cablu arată la fel ca o mare companie.

Deși această schemă rezolvă într-un fel problema, multe persoane din comunitatea IP o consideră un monstru pe fața pământului. Rezumate succint, iată câteva dintre obiecții. În primul rând, NAT violează modelul arhitectural al IP-ului, care afirmă că fiecare adresă IP identifică unic o sin-

gură mașină din lume. Întreaga structură software a Internetului este construită pornind de la acest fapt. Cu NAT, mii de mașini pot folosi (și folosesc) adresa 10.0.0.1.

În al doilea rând, NAT schimbă natura Internetului de la o rețea fără conexiuni la un fel de rețea cu conexiuni. Problema este că o unitate NAT trebuie să mențină informații (corespondențe) pentru fiecare conexiune care trece prin ea. Faptul că rețeaua păstrează starea conexiunilor este o proprietate a rețelelor orientate pe conexiune, nu a celor neorientate pe conexiune. Dacă o unitate NAT se defectează și tabela de corespondență este pierdută, toate conexiunile TCP sunt distruse. În absența NAT, căderea ruterelor nu are nici un efect asupra TCP. Procesul care transmite ajunge la limita de timp în câteva secunde și retransmite toate pachetele neconfirmate. Cu NAT, Internetul devine la fel de vulnerabil ca o rețea cu comutare de circuite.

În al treilea rând, NAT încalcă cea mai fundamentală regulă a protocoalelor pe niveluri: nivelul *k* nu poate face nici un fel de presupuneri referitor la ceea ce a pus nivelul *k+1* în câmpul de informație utilă. Principiul de bază este să se păstreze niveluri independente. Dacă mai târziu TCP este înlocuit de TCP-2, cu un antet diferit (de exemplu porturi pe 32 de biți), NAT va eșua. Ideea protocoalelor pe niveluri este de a asigura că modificările la un nivel nu necesită modificări la celelalte niveluri. NAT distruge această independență.

În al patrulea rând, procesele din Internet nu sunt obligate să folosească TCP sau UDP. Dacă un utilizator de pe mașina *A* se decide să folosesc un nou protocol de transport pentru a comunica cu un utilizator de pe mașina *B* (de exemplu, pentru o aplicație multimedia), introducerea unei unități NAT va determina eșecul aplicației, deoarece cutia NAT nu va fi în stare să localizeze corect câmpul TCP *Port sursă*.

În al cincilea rând, anumite aplicații introduc adrese IP în corpul mesajului. Receptorul extrage aceste adrese și le folosește. De vreme ce NAT nu știe nimic despre aceste adrese, nu le poate înlocui, deci orice încercare de a le folosi de către cealaltă parte va eșua. **FTP**, standardul **File Transfer Protocol** (rom.: protocol de transfer de fișiere) funcționează în acest mod și poate eșua în prezența NAT dacă nu se iau măsuri speciale. În mod similar protocolul pentru telefonie Internet H.323 (care va fi studiat în Cap. 8) are această proprietate și nu va funcționa în prezența NAT. Ar fi posibil să se modifice NAT-ul pentru a funcționa cu H.323, dar modificarea codului unității NAT de fiecare dată când apare o aplicație nouă nu este o idee bună.

În al șaselea rând, deoarece câmpul TCP *Port sursă* are 16 biți, o adresă IP poate fi pusă în corespondență cu cel mult 65.536 mașini. De fapt acest număr este puțin mai mic, deoarece primele 4096 porturi sunt rezervate pentru utilizări speciale. Totuși dacă sunt disponibile mai multe adrese IP fiecare poate trata 61.440 mașini.

Acestea și alte probleme ale NAT sunt discutate în RFC 2993. În general, opozanții NAT spun că rezolvând problema insuficienței adreselor IP cu o soluție temporară și urâtă, presiunea pentru a implementa soluția reală, adică tranziția la IPv6, este redusă și acesta este un lucru rău.

5.5.4 Protocoale de control în Internet

Pe lângă IP, care este folosit pentru transferul de date, Internet-ul are câteva protocoale de control la nivelul rețea, incluzând ICMP, ARP, RARP, BOOTP și DHCP. În această secțiune vom arunca o privire asupra fiecăruia dintre ele.

Protocolul mesajelor de control din Internet

Operarea Internet-ului este strâns monitorizată de către rutere. Atunci când se întâmplă ceva neobișnuit, evenimentul este raportat prin **ICMP** (**Internet Control Message Protocol** - protocolul mesajelor de control din Internet), care este folosit și pentru testarea Internet-ului. Sunt definite aproape o duzină de tipuri de mesaje ICMP. Cele mai importante sunt enumerate în fig. 5-61. Fiecare tip de mesaj ICMP este încapsulat într-un pachet IP.

Tipul mesajului	Descriere
Destinație inaccesibilă	Pachetul nu poate fi livrat
Timp depășit	Câmpul timp de viață a ajuns la 0
Problemă de parametru	Câmp invalid în antet
Oprire sursă	Pachet de șoc
Redirectare	Învață un ruter despre geografie
Cerere de ecou	Întreabă o mașină dacă este activă
Răspuns ecou	Da, sunt activă
Cerere de amprentă de timp	La fel ca cererea de ecou, dar cu amprentă de timp
Răspuns cu amprentă de timp	La fel ca răspunsul ecou, dar cu amprentă de timp

Fig. 5-61. Tipurile principale de mesaje ICMP.

Mesajul DESTINAȚIE INACCESIBILĂ (DESTINATION UNREACHABLE) este folosit atunci când subrețeaua sau un ruter nu pot localiza destinația, sau un pachet cu bitul DF nu poate fi livrat deoarece o rețea cu „pachete mici” îi stă în cale.

Mesajul TIMP DEPĂȘIT (TIME EXCEEDED) este trimis când un pachet este eliminat datorită ajungerii contorului său la zero. Acest mesaj este un simptom al buclării pachetelor, al unei enorme congestii sau al stabilirii unor valori prea mici pentru ceas.

Mesajul PROBLEMĂ DE PARAMETRU (PARAMETER PROBLEM) indică detectarea unei valori nepermise într-un câmp din antet. Această problemă indică o eroare în programele IP ale gazdei emițătoare sau eventual în programele unui ruter tranzitat.

Mesajul OPRIRE SURSĂ (SOURCE QUENCH) a fost folosit pe vremuri pentru a limita traficul gazdelor care trimiteau prea multe pachete. Când o gazdă primea acest mesaj, era de așteptat să încetinească ritmul de transmisie. Este folosit arareori, deoarece când apare congestie, aceste pachete au tendința de a turna mai mult gaz pe foc. Controlul congestiei în Internet este făcut acum pe larg la nivelul transport și va fi studiat în detaliu în Cap. 6.

Mesajul REDIRECTARE (REDIRECT) este folosit atunci când un ruter observă că un pachet pare a fi dirijat greșit. Este folosit de ruter pentru a spune gazdei emițătoare despre eroarea probabilă.

Mesajele CERERE ECOU (ECHO REQUEST) și RĂSPUNS ECOU (ECHO REPLY) sunt folosite pentru a vedea dacă o anumită destinație este accesibilă și activă. Este de așteptat ca la recepția mesajului ECOU, destinația să răspundă printr-un mesaj RĂSPUNS ECOU. Mesajele CERERE AMPRENTĂ DE TIMP (TIMESTAMP REQUEST) și RĂSPUNS AMPRENTĂ DE TIMP (TIMESTAMP REPLY) sunt similare, cu excepția faptului că în răspuns sunt înregistrate timpul de sosire a mesajului și de plecare a răspunsului. Această facilitate este folosită pentru a măsura performanțele rețelei.

În plus față de aceste mesaje, au fost definite și altele. Lista se află on-line la adresa www.iana.org/assignements/icmp-parameters.

Protocolul de rezoluție a adresei: ARP

Deși fiecare mașină din Internet are una sau mai multe adrese IP, acestea nu pot fi folosite de fapt pentru trimiterea pachetelor deoarece hardware-ul nivelului legăturii de date nu înțelege adresele Internet. Actualmente, cele mai multe gazde sunt atașate la un LAN printr-o placă de interfață care înțelege numai adresele LAN. De exemplu, fiecare placă Ethernet fabricată până acum vine cu o adresă Ethernet de 48 biți. Fabricanții plăcilor Ethernet cer un spațiu de adrese de la o autoritate centrală pentru a se asigura că nu există două plăci cu aceeași adresă (pentru a evita conflictele care ar apărea dacă cele două plăci ar fi în același LAN). Plăcile trimit și primesc cadre pe baza adresei Ethernet de 48 biți. Ele nu știu absolut nimic despre adresele IP pe 32 de biți.

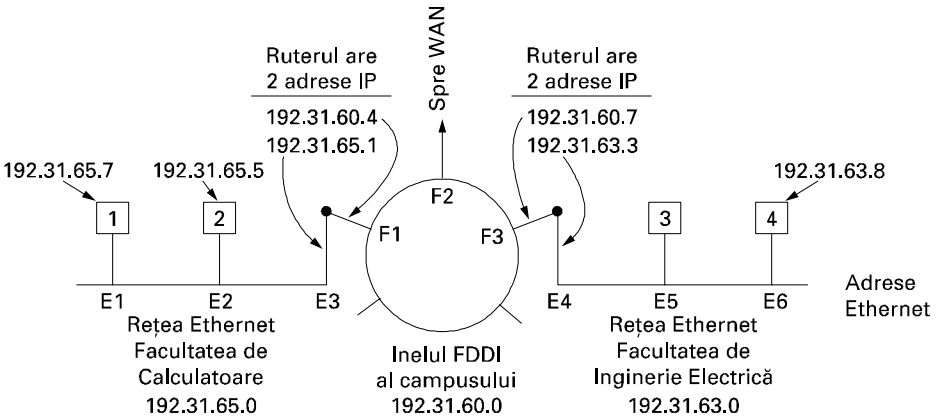


Fig. 5-62. Trei rețele /24 interconectate: două rețele Ethernet și un inel FDDI.

Se pune atunci întrebarea: Cum sunt transformate adresele IP în adrese la nivelul legăturii de date, ca de exemplu Ethernet? Pentru a explica care este funcționarea, vom folosi exemplul din fig. 5-62, în care este ilustrată o universitate mică ce are câteva rețele de clasă C (denumită acum /24). Avem două rețele Ethernet, una în facultatea de Calculatoare, cu adresa IP 192.31.65.0 și una în facultatea de Inginerie Electrică, cu adresa IP 192.31.63.0. Acestea sunt conectate printr-un inel FDDI la nivelul campusului, care are adresa IP 192.31.60.0. Fiecare mașină dintr-o rețea Ethernet are o adresă Ethernet unică, etichetată de la E1 la E6, iar fiecare mașină de pe inelul FDDI are o adresă FDDI, etichetată de la F1 la F3.

Să începem prin a vedea cum trimite un utilizator de pe gazda 1 un pachet unui utilizator de pe gazda 2. Presupunem că expeditorul știe numele destinatarului, ceva de genul *ary@eagle.cs.uni.edu*. Primul pas este aflarea adresei IP a gazdei 2, cunoscută ca *eagle.cs.uni.edu*. Această căutare este făcută de sistemul numelor de domenii (DNS), pe care îl vom studia în Cap. 7. Pentru moment, vom presupune că DNS-ul întoarce adresa IP a gazdei 2 (192.31.65.5).

Programele de la nivelurile superioare ale gazdei 1 construiesc un pachet cu 192.31.65.5 în câmpul *adresa destinatarului,* pachet care este trimis programelor IP pentru a-l transmite. Programele IP se uită la adresă și văd că destinatarul se află în propria rețea, dar au nevoie de un mijloc prin care să determine adresa Ethernet a destinatarului. O soluție este să avem undeva în sistem un fișier de configurare care transformă adresele IP în adrese Ethernet. Această soluție este posibilă, desigur, dar pentru organizații cu mii de mașini, menținerea fișierelor actualizate este o acțiune consumatoare de timp și care poate genera erori.

O soluție mai bună este ca gazda 1 să trimită un pachet de difuzare în rețeaua Ethernet întrebând: „Cine este proprietarul adresei IP 192.31.65.5?”. Pachetul de difuzare va ajunge la toate mașinile din rețeaua Ethernet 192.31.65.0 și fiecare își va verifica adresa IP. Numai gazda 2 va răspunde cu adresa sa Ethernet (*E2*). În acest mod gazda 1 află că adresa IP 192.31.65.5 este pe gazda cu adresa Ethernet *E2*. Protocolul folosit pentru a pune astfel de întrebări și a primi răspunsul se numește **ARP** (**Address Resolution Protocol** - Protocolul de rezoluție a adresei). Aproape toate mașinile din Internet îl folosesc. El este definit în RFC 826.

Avantajul folosirii ARP față de fișierele de configurare îl reprezintă simplitatea. Administratorul de sistem nu trebuie să facă prea multe, decât să atribuie fiecărei mașini o adresă IP și să hotărască măștile subrețelelor. ARP-ul face restul.

În acest punct, programele IP de pe gazda 1 construiesc un cadru Ethernet adresat către *E2*, pun pachetul IP (adresat către 193.31.65.5) în câmpul informație utilă și îl lansează pe rețeaua Ethernet. Placa Ethernet a gazdei 2 detectează acest cadru, recunoaște că este un cadru pentru ea, îl ia repede și generează o întrerupere. Driverul Ethernet extrage pachetul IP din informația utilă și îl trimite programelor IP, care văd că este corect adresat și îl prelucrează.

Pentru a face ARP-ul mai eficient sunt posibile mai multe optimizări. Pentru început, la fiecare execuție a ARP, mașina păstrează rezultatul pentru cazul în care are nevoie să contacteze din nou aceeași mașină în scurt timp. Data viitoare va găsi local corespondentul adresei, evitându-se astfel necesitatea unei a doua difuzări. În multe cazuri, gazda 2 trebuie să trimită înapoi un răspuns, ceea ce o forțează să execute ARP, pentru a determina adresa Ethernet a expeditorului. Această difuzare ARP poate fi evitată obligând gazda 1 să includă în pachetul ARP corespondența dintre adresa sa IP și adresa Ethernet. Când pachetul ARP ajunge la gazda 2, perechea (192.31.65.7, E1) este memorată local de ARP pentru o folosire ulterioară. De fapt, toate mașinile din rețeaua Ethernet pot memora această relație în memoria ARP locală.

Altă optimizare este ca fiecare mașină să difuzeze corespondența sa de adrese la pornirea mașinii. Această difuzare este realizată în general printr-un pachet ARP de căutare a propriei adrese IP. Nu ar trebui să existe un răspuns, dar un efect lateral al difuzării este introducerea unei înregistrări în memoria ascunsă ARP a tuturor. Dacă totuși sosește un răspuns (neașteptat), înseamnă că două mașini au aceeași adresă IP. Noua mașină ar trebui să-l informeze pe administratorul de sistem și să nu pornească.

Pentru a permite schimbarea relației, de exemplu, când o placă Ethernet se strică și este înlocuită cu una nouă (și astfel apare o nouă adresă Ethernet), înregistrările din memoria ascunsă ARP ar trebui să expire după câteva minute.

Să privim din nou fig. 5-62, numai că de această dată gazda 1 vrea să trimită un pachet către gazda 4 (192.31.63.8). Folosirea ARP va eșua pentru că gazda 4 nu va vedea difuzarea (ruterele nu trimit mai departe difuzările de nivel Ethernet). Există două soluții. Prima: ruterul facultății de Calculatoare poate fi configurat să răspundă la cererile ARP pentru rețeaua 193.31.63.0 (și posibil și pentru alte rețele locale). În acest caz, gazda 1 va memora local perechea (193.31.63.8, E3) și va trimite tot traficul pentru gazda 4 către ruterul local. Această soluție se numește **ARP cu intermediar** (**proxy ARP**). A doua soluție este ca gazda 1 să-și dea seama imediat că destinația se află pe o rețea aflată la distanță și să trimită tot traficul către o adresă Ethernet implicită care tratează tot traficul la distanță, în acest caz *E3*. Această soluție nu necesită ca ruterul facultății de Calculatoare să știe ce rețele la distanță deservește.

În ambele cazuri, ceea ce se întâmplă este că gazda 1 împachetează pachetul IP în câmpul informație utilă dintr-un cadru Ethernet adresat către *E3*. Când ruterul facultății de Calculatoare primeș-

te cadrul Ethernet, extrage pachetul IP din câmpul informație utilă și caută adresa IP din tabelele sale de dirijare. Descoperă că pachetele pentru rețeaua 193.31.63.0 trebuie să meargă către ruterul 192.31.60.7. Dacă nu cunoaște încă adresa FDDI a lui 193.31.60.7, difuzează un pachet ARP pe inel și află că adresa din inel este *F3*. Apoi inserează pachetul în câmpul informație utilă al unui cadru FDDI adresat către *F3* și îl transmite pe inel.

Driverul FDDI al ruterului facultății de Inginerie Electrică scoate pachetul din câmpul informație utilă și îl trimite programelor IP care văd că trebuie să trimită pachetul către 192.31.63.8. Dacă această adresă IP nu este în memoria ascunsă ARP, difuzează o cerere ARP pe rețeaua Ethernet a facultății de Inginerie Electrică și află că adresa destinație este *E6*, astfel încât construiește un cadru Ethernet adresat către *E6*, pune pachetul în câmpul informație utilă și îl trimite în rețeaua Ethernet. Când cadrul Ethernet ajunge la gazda 4, pachetul este extras din cadru și trimis programelor IP pentru procesare.

Transferul între gazda 1 și o rețea la distanță peste un WAN funcționează în esență asemănător, cu excepția că de data aceasta tabelele ruterului facultății de Calculatoare îi vor indica folosirea ruterului WAN, a cărui adresă FDDI este *F2*.

RARP, BOOTP și DHCP

ARP-ul rezolvă problema aflării adresei Ethernet corespunzătoare unei adrese IP date. Câteodată trebuie rezolvată problema inversă: dându-se o adresă Ethernet, care este adresa IP corespunzătoare? În particular, această problemă apare când se pornește o stație de lucru fără disc. O astfel de mașină va primi, în mod normal, imaginea binară a sistemului său de operare de la un server de fișiere la distanță. Dar cum își află adresa IP?

Prima soluție proiectată a fost **RARP** (**Reverse Address Resolution Protocol** - Protocol de rezoluție inversă a adresei) (definit în RFC 903). Acest protocol permite unei stații de lucru de-abia pornită să difuzeze adresa sa Ethernet și să spună: „Adresa mea Ethernet de 48 de biți este 14.04.05.18.01.25. Știe cineva adresa mea IP?" Serverul RARP vede această cerere, caută adresa Ethernet în fișierele sale de configurare și trimite înapoi adresa IP corespunzătoare.

Folosirea RARP este mai bună decât introducerea unei adrese IP în imaginea de memorie, pentru că permite ca aceeași imagine să fie folosită pe toate mașinile. Dacă adresa IP ar fi fixată înăuntrul imaginii, atunci fiecare stație de lucru ar necesita imaginea sa proprie.

Un dezavantaj al RARP este că, pentru a ajunge la serverul RARP, folosește o adresă destinație numai din 1-uri (difuzare limitată). Cu toate acestea, asemenea difuzări nu sunt propagate de rutere, așa încât este necesar un server RARP în fiecare rețea. Pentru a rezolva această problemă, a fost inventat un protocol alternativ de pornire, numit BOOTP (vezi RFC-urile 951, 1048 și 1084). Spre deosebire de RARP, acesta folosește mesaje UDP, care sunt propagate prin rutere. De asemenea furnizează unei stații de lucru fără disc informații suplimentare, care includ adresa IP a serverului de fișiere care deține imaginea de memorie, adresa IP a ruterului implicit și masca de subrețea care se folosește. BOOTP-ul este descris în RFC 951, 1048 și 1084.

O problemă serioasă cu BOOTP este că necesită configurarea manuală a corespondențelor între adresele IP și adresele Ethernet. Atunci când o nouă gazdă este adăugată la LAN, nu poate folosi BOOTP decât atunci când un administrator îi alocă o adresă IP și introduce manual (adresă Ethernet, adresă IP) în tabelele de configurare BOOTP. Pentru a elimina acest pas predispus la erori, BOOTP a fost extins și i-a fost dat un nou nume: **DHCP** (**Dynamic Host Configuration Protocol** – Protocol dinamic de configurare a gazdei). DHCP permite atât atribuirea manuală

de adrese IP, cât și atribuirea automată. Este descris în RFC-urile 2131 și 2132. În majoritatea sistemelor, a înlocuit în mare parte RARP și BOOTP.

Ca și RARP și BOOTP, DHCP este bazat pe ideea unui server special care atribuie adrese IP gazdelor care cer una. Acest server nu trebuie să se afle în același LAN cu gazda care face cererea. Deoarece serverul DHCP s-ar putea să nu fie accesibil prin difuzare, este nevoie ca în fiecare LAN să existe un **agent de legătură DHCP** (**DHCP relay agent**), așa cum se vede în fi fig. 5-63.

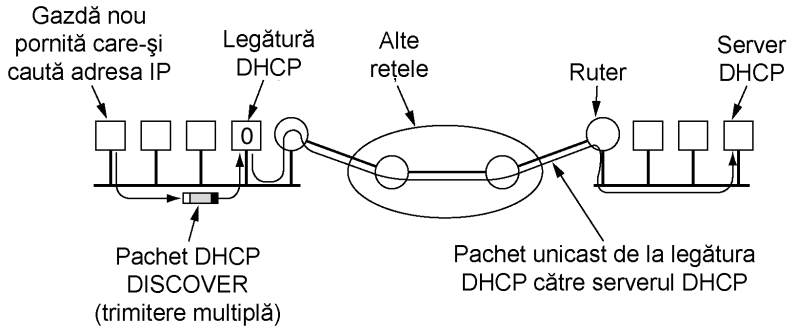


Fig. 5-63. Funcționarea DHCP.

Pentru a-și afla adresa IP, o mașină tocmai pornită difuzează un pachet DHCP DISCOVER. Agentul de legătură DHCP din LAN interceptează toate difuzările DHCP. Atunci când găsește un pachet DHCP DISCOVER, îl trimite ca pachet unicast serverului DHCP, posibil într-o rețea depărtată. Singura informație de care are nevoie agentul este adresa IP a serverului DHCP.

O problemă care apare cu atribuirea automată a adreselor IP dintr-o rezervă comună este cât de mult ar trebui alocată o adresă IP . Dacă o gazdă părăsește rețeaua și nu returnează adresa sa IP serverului DHCP, acea adresă va fi pierdută permanent. După o perioadă de timp vor fi pierdute multe adrese. Pentru a preveni aceasta, atribuirea adresei IP va fi pentru o perioadă fixă de timp, o tehnică numită închiriere. Chiar înainte ca perioada de închiriere să expire, gazda trebuie să îi ceară DHCP-ului o reînnoire. Dacă nu reușește să facă cererea sau dacă cererea este respinsă, gazda nu va mai putea folosi adresa IP care îi fusese dată mai devreme.

5.5.5 Protocolul de dirijare folosit de porțile interioare: OSPF

Am terminat acum studiul protocoalelor de control ale Internetului. Este timpul să trecem la următorul subiect : rutarea în Internet. Așa cum am menționat anterior, Internet-ul este construit dintr-un număr mare de sisteme autonome. Fiecare AS este administrat de o organizație diferită și poate folosi propriul algoritm de dirijare în interior. De exemplu, rețelele interne ale companiilor *X*, *Y* și *Z* ar fi văzute ca trei AS-uri dacă toate ar fi în Internet. Toate trei pot folosi intern algoritmi de dirijare diferiți. Cu toate acestea, existența standardelor, chiar și pentru dirijarea internă, simplifică implementarea la granițele dintre AS-uri și permite reutilizarea codului. În această secțiune vom studia dirijarea în cadrul unui AS. În următoarea, vom examina dirijarea între AS-uri. Un algoritm de dirijare în interiorul unui AS este numit **protocol de porți interioare**; un algoritm de dirijare utilizat între AS-uri este numit **protocol de porți exterioare**.

Protocolul de porți interioare inițial în Internet a fost un protocol de dirijare pe baza vectorilor distanță (RIP) bazat pe algoritmul Bellman-Ford moștenit de la ARPANET. El funcționa bine în sisteme mici, dar mai puțin bine pe măsură ce AS-urile s-au extins. El suferea de asemenea de pro-

blema numărării la infinit și de o convergență slabă în general, așa că în mai 1979 a fost înlocuit de un protocol bazat pe starea legăturilor. În 1988, Internet Engineering Task Force a început lucrul la un succesor. Acel succesor, numit **OSPF** (**Open Shortest Path First** - protocol public (deschis) bazat pe calea cea mai scurtă) a devenit un standard în 1990. Mulți producători de rutere oferă suport pentru el și acesta a devenit principalul protocol de porți interioare. În cele ce urmează vom face o schiță a funcționării OSPF. Pentru informații complete, vedeți RFC 2328.

Dată fiind experiența îndelungată cu alte protocoale de dirijare, grupul care a proiectat noul protocol a avut o listă lungă de cerințe care trebuiau satisfăcute. Mai întâi, algoritmul trebuia publicat în literatura publică (open), de unde provine „O" din OSPF. O soluție în proprietatea unei companii nu era un candidat. În al doilea rând, noul protocol trebuia să suporte o varietate de metrici de distanță, incluzând distanța fizică, întârzierea ș.a.m.d. În al treilea rând, el trebuia să fie un algoritm dinamic, care să se adapteze automat și repede la schimbările în topologie.

În al patrulea rând și nou pentru OSPF, trebuia să suporte dirijarea bazată pe tipul de serviciu. Noul protocol trebuia să fie capabil să dirijeze traficul de timp real într-un mod, iar alt tip de trafic în alt mod. Protocolul IP are câmpul *Tip serviciu*, dar nici un protocol de dirijare nu-l folosea. Acest câmp a fost introdus în OSPF dar tot nu îl folosea nimeni și în cele din urmă a fost eliminat.

În al cincilea rând și în legătură cu cele de mai sus, noul protocol trebuia să facă echilibrarea încărcării, divizând încărcarea pe mai multe linii. Majoritatea protocoalelor anterioare trimit toate pachetele pe cea mai bună cale. Calea de pe locul doi nu era folosită de loc. În multe cazuri, divizarea încărcării pe mai multe linii duce la performanțe mai bune.

În al șaselea rând, era necesar suportul pentru sisteme ierarhice. Până în 1988, Internet a crescut atât de mult, încât nu se poate aștepta ca un ruter să cunoască întreaga topologie. Noul protocol de dirijare trebuia să fie proiectat astfel, încât nici un ruter să nu fie nevoit să cunoască toată topologia.

În al șaptelea rând, se cerea un minim de măsuri de securitate, pentru a evita ca studenții iubitori de distracții să păcălească ruterele trimițându-le informații de dirijare false. În fine, a fost necesară luarea de măsuri pentru a trata ruterele care au fost conectate la Internet printr-un tunel. Protocoalele precedente nu tratau bine acest caz.

OSPF suportă trei tipuri de conexiuni și rețele:

1. Linii punct-la-punct între exact două rutere.
2. Rețele multiacces cu difuzare (de exemplu, cele mai multe LAN-uri).
3. Rețele multiacces fără difuzare (de exemplu, cele mai multe WAN-uri cu comutare de pachete).

O rețea **multiacces** este o rețea care poate să conțină mai multe rutere, fiecare dintre ele putând comunica direct cu toate celelalte. Toate LAN-urile și WAN-urile au această proprietate. Fig. 5-64(a) prezintă un AS care conține toate cele trei tipuri de rețele. Observați că gazdele, în general, nu joacă un rol în OSPF.

OSPF funcționează prin abstractizarea colecției de rețele, rutere și linii reale într-un graf orientat în care fiecărui arc îi este atribuit un cost (distanță, întârziere etc.). Apoi calculează cea mai scurtă cale bazându-se pe ponderile arcelor. O conexiune serială între două rutere este reprezentată de o pereche de arce, câte unul în fiecare direcție. Ponderile lor pot fi diferite. O rețea multiacces este reprezentată de un nod pentru rețeaua însăși plus câte un nod pentru fiecare ruter. Arcele de la nodul rețea la rutere au pondere 0 și sunt omise din graf.

Fig. 5-64(b) prezintă graful pentru rețeaua din fig. 5-64(a). Ponderile sunt simetrice, dacă nu se specifică altfel. Fundamental, ceea ce face OSPF este să reprezinte rețeaua reală printr-un graf ca acesta și apoi să calculeze cea mai scurtă cale de la fiecare ruter la fiecare alt ruter.

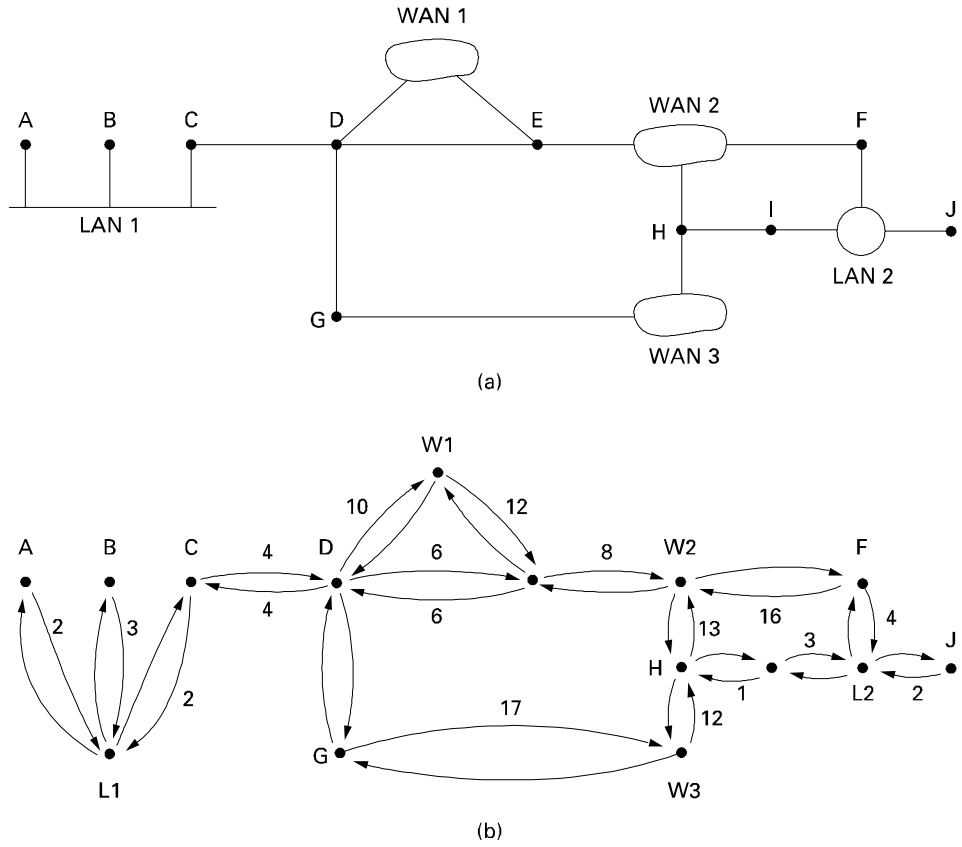


Fig. 5-64. (a) Un sistem autonom. (b) O reprezentare de tip graf a lui (a).

Multe din AS-urile din Internet sunt foarte mari și nu sunt simplu de administrat. OSPF le permite să fie divizate în **zone** numerotate, unde o zonă este o rețea sau o mulțime de rețele învecinate. Zonele nu se suprapun și nu este necesar să fie exhaustive, în sensul că unele rutere pot să nu aparțină nici unei zone. O zonă este o generalizare a unei subrețele. În afara zonei, topologia și detaliile sale nu sunt vizibile.

Orice AS are o zonă **de coloană vertebrală**, numită zona 0. Toate zonele sunt conectate la coloana vertebrală, eventual prin tunele, astfel încât este posibil să se ajungă din orice zonă din AS în orice altă zonă din AS prin intermediul coloanei vertebrale. Un tunel este reprezentat în graf ca un arc și are un cost. Fiecare ruter care este conectat la două sau mai multe zone aparține coloanei vertebrale. Analog cu celelalte zone, topologia coloanei vertebrale nu este vizibilă din afara coloanei vertebrale.

În interiorul unei zone, fiecare ruter are aceeași bază de date pentru starea legăturilor și folosește același algoritm de cea mai scurtă cale. Principala sa sarcină este să calculeze cea mai scurtă cale de la sine la fiecare alt ruter din zonă, incluzând ruterul care este conectat la coloana vertebrală, din care trebuie să existe cel puțin unul. Un ruter care conectează două zone are nevoie de bazele de date pentru ambele zone și trebuie să folosească algoritmul de cale cât mai scurtă separat pentru fiecare zonă.

În timpul operării normale pot fi necesare trei tipuri de căi: intrazonale, interzonale și interAS-uri. Rutele intrazonale sunt cele mai ușoare, din moment ce ruterul sursă știe întotdeauna calea cea

mai scurtă spre ruterul destinație. Dirijarea interzonală se desfășoară întotdeauna în trei pași: drum de la sursă la coloana vertebrală; drum de-a lungul coloanei vertebrale până la zona destinație; drum la destinație. Acest algoritm forțează o configurație de tip stea pentru OSPF, coloana vertebrală fiind concentratorul (hub), iar celelalte zone fiind spițele. Pachetele sunt dirijate de la sursă la destinație „ca atare". Ele nu sunt încapsulate sau trecute prin tunel, cu excepția cazului în care merg spre o zonă a cărei unică conexiune la coloana vertebrală este un tunel. Fig. 5-65 arată o parte a Internetului cu AS-uri și zone.

OSPF distinge patru clase de rutere:

1. Ruterele interne sunt integral în interiorul unei zone.
2. Ruterele de la granița zonei conectează două sau mai multe zone.
3. Ruterele coloanei vertebrale sunt pe coloana vertebrală.
4. Ruterele de la granița AS-urilor discută cu ruterele din alte AS-uri.

Aceste clase pot să se suprapună. De exemplu, toate ruterele de graniță fac parte în mod automat din coloana vertebrală. În plus, un ruter care este în coloana vertebrală, dar nu face parte din nici o altă zonă este de asemenea un ruter intern. Exemple din toate cele patru clase de rutere sunt ilustrate în fig. 5-65.

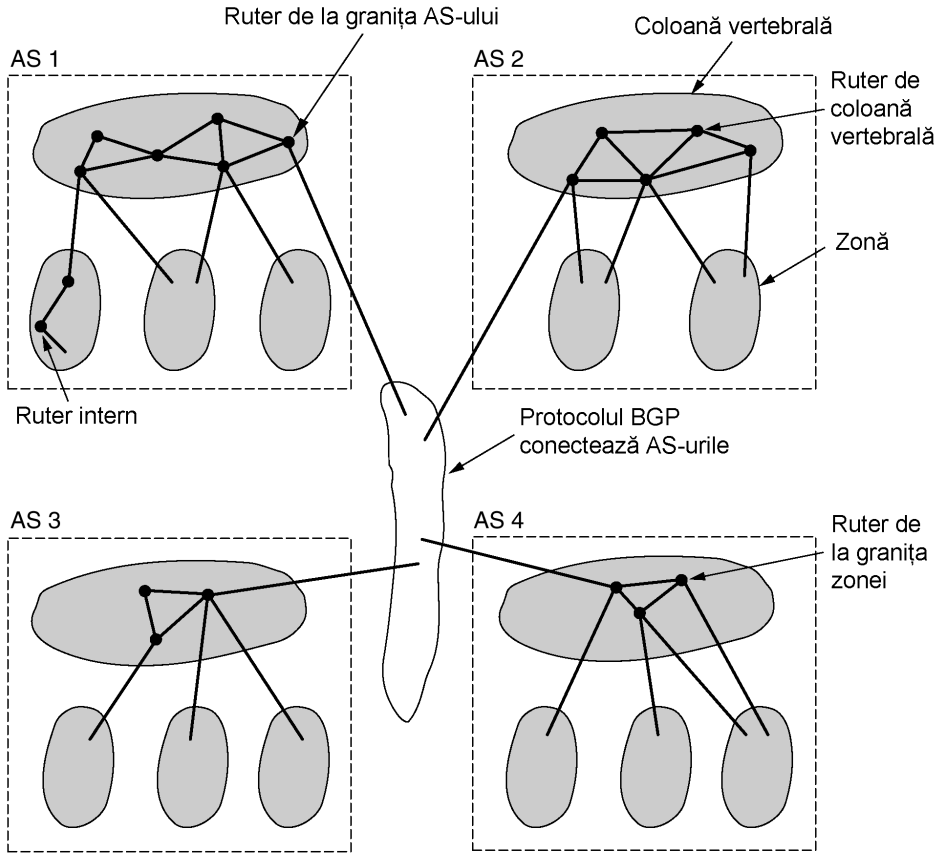


Fig. 5-65. Relația dintre AS-uri, coloane vertebrale și zone în OSPF.

Când un ruter pornește, trimite mesaje HELLO pe toate liniile sale punct-la-punct și trimite multiplu (multicast) în LAN-urile grupului compus din toate celelalte rutere. În WAN-uri, are nevoie de anumite informații de configurație, pentru a ști pe cine să contacteze. Din răspunsuri, fiecare ruter află care sunt vecinii săi. Ruterele din același LAN sunt toate vecine.

OSPF funcționează prin schimb de informații între rutere **adiacente**, care nu este același lucru cu schimbul de informații între ruterele vecine. În particular, este ineficient ca fiecare ruter dintr-un LAN să discute cu fiecare alt ruter din LAN. Pentru a evita această situație, un ruter este ales ca **ruter desemnat**. Se spune că el este adiacent cu toate celelalte rutere din LAN-ul său și schimbă informații cu ele. Ruterele vecine care nu sunt adiacente nu schimbă informații între ele. De asemenea, este actualizat în permanență și un ruter desemnat de rezervă pentru a ușura tranziția dacă ruterul desemnat primar se defectează și trebuie să fie înlocuit imediat.

În timpul funcționării normale, fiecare ruter inundă periodic cu mesaje ACTUALIZARE STARE LEGĂTURĂ (Link State Update) fiecare ruter adiacent. Acest mesaj indică starea sa și furnizează costurile folosite în baza de date topologică. Mesajele de inundare sunt confirmate pentru a le face sigure. Fiecare mesaj are un număr de secvență, astfel încât un ruter poate vedea dacă un mesaj ACTUALIZARE STARE LEGĂTURĂ este mai vechi sau mai nou decât ceea ce are deja. De asemenea, ruterele trimit aceste mesaje când o linie cade sau își revine sau când costul acesteia se modifică.

Mesajele DESCRIERE BAZA DE DATE (Database Description) dau numerele de secvență pentru toate intrările de stare a liniei deținute actual de emițător. Prin compararea valorilor proprii cu acelea ale emițătorului, receptorul poate determina cine are cea mai recentă valoare. Aceste mesaje sunt folosite când o linie este refăcută.

Fiecare partener poate cere informații de stare a legăturii de la celălalt folosind mesaje CERERE STARE LEGĂTURĂ (Link State Request). Rezultatul concret al acestui algoritm este că fiecare pereche de rutere adiacente verifică să vadă cine are cele mai recente date și astfel, noua informație este răspândită în zonă. Toate aceste mesaje sunt trimise ca simple pachete IP. Cele cinci tipuri de mesaje sunt rezumate în fig. 5-66.

Tip mesaj	Descriere
Hello	Folosit pentru descoperirea vecinilor
Actualizare Stare Legătură	Emițătorul furnizează vecinilor săi costurile sale
Confirmare Stare Legătură	Confirmă actualizarea stării legăturii
Descriere Bază de Date	Anunță ce actualizări are emițătorul
Cerere Stare Legătură	Cere informații de la partener

Fig. 5-66. Cele cinci tipuri de mesaje OSPF.

În final, putem să asamblăm toate piesele. Folosind inundarea, fiecare ruter informează toate celelalte rutere din zona sa despre vecinii și costurile sale. Această informație permite fiecărui ruter să construiască graful zonei (zonelor) sale și să calculeze cea mai scurtă cale. Zona de coloană vertebrală face și ea același lucru. În plus, ruterele de coloană vertebrală acceptă informația de la ruterele zonei de graniță cu scopul de a calcula cea mai bună cale de la fiecare ruter de coloană vertebrală către fiecare alt ruter. Această informație este propagată înapoi către ruterele zonei de graniță, care o fac publică în zonele lor. Folosind această informație, un ruter gata să trimită un pachet interzonal poate selecta cel mai bun ruter de ieșire către coloana vertebrală.

5.6.5 Protocolul de dirijare pentru porți externe: BGP

În cadrul unui singur AS, protocolul de dirijare recomandat este OSPF (deși, desigur, nu este singurul folosit). Între AS-uri se folosește un protocol diferit, **BGP** (**Border Gateway Protocol** - Protocolul porților de graniță). Între AS-uri este necesar un protocol diferit pentru că scopurile unui protocol pentru porți interioare și ale unui protocol pentru porți exterioare sunt diferite. Tot ce trebuie să facă un protocol pentru porți interioare este să mute pachetele cât mai eficient de la sursă la destinație. El nu trebuie să-și facă probleme cu politica.

Ruterele ce folosesc protocolul de porți exterioare trebuie să țină cont într-o mare măsură de politică (Metz, 2001). De exemplu, un AS al unei corporații poate dori facilitatea de a trimite pachete oricărui sit Internet și să recepționeze pachete de la orice sit Internet. Cu toate acestea, poate să nu dorească să asigure tranzitarea pentru pachetele originare dintr-un AS străin destinate unui AS străin diferit, chiar dacă prin AS-ul propriu trece cea mai scurtă cale dintre cele două AS-uri străine („Asta este problema lor, nu a noastră."). Pe de altă parte, poate fi dispus să asigure tranzitarea pentru vecinii săi, sau chiar pentru anumite AS-uri care au plătit pentru acest serviciu. Companiile telefonice, de exemplu, pot fi fericite să acționeze ca un purtător pentru clienții lor, dar nu și pentru alții. Protocoalele pentru porți externe, în general și BGP în particular, au fost proiectate pentru a permite forțarea multor tipuri de politici de dirijare pentru traficul între AS-uri.

Politicile tipice implică considerații politice, de securitate sau economice. Câteva exemple de constrângeri de dirijare sunt:

1. Nu se tranzitează traficul prin anumite AS-uri.
2. Nu se plasează Irak-ul pe o rută care pornește din Pentagon.
3. Nu se folosesc Statele Unite pentru a ajunge din Columbia Britanică în Ontario.
4. Albania este tranzitată numai dacă nu există altă alternativă către destinație.
5. Traficul care pleacă sau ajunge la IBM nu trebuie să tranziteze Microsoft.

În mod obișnuit politicile sunt configurate manual în fiecare ruter BGP (sau sunt incluse folosind un anumit tip de script). Ele nu sunt parte a protocolului însuși.

Din punctul de vedere al unui ruter BGP, lumea constă din AS-uri și liniile care le conectează. Două AS-uri sunt considerate conectate dacă există o linie între două rutere de graniță din fiecare. Dat fiind interesul special al BGP-ului pentru traficul în tranzit, rețelele sunt grupate în trei categorii. Prima categorie este cea a **rețelelor ciot (stub networks)**, care au doar o conexiune la graful BGP. Acestea nu pot fi folosite pentru traficul în tranzit pentru că nu este nimeni la capătul celălalt. Apoi vin **rețelele multiconectate**. Acestea pot fi folosite pentru traficul în tranzit, cu excepția a ceea ce ele refuză. În final, sunt **rețele de tranzit**, cum ar fi coloanele vertebrale, care sunt doritoare să manevreze pachetele altora, eventual cu unele restricții, și de obicei pentru o plată.

Perechile de rutere BGP comunică între ele stabilind conexiuni TCP. Operarea în acest mod oferă comunicație sigură și ascunde toate detaliile rețelelor traversate.

BGP este la bază un protocol bazat pe vectori distanță, dar destul de diferit de majoritatea celorlalte cum ar fi RIP. În loc să mențină doar costul până la fiecare destinație, fiecare ruter BGP memorează calea exactă folosită. Similar, în loc să trimită periodic fiecărui vecin costul său estimat către fiecare destinație posibilă, fiecare ruter BGP comunică vecinilor calea exactă pe care o folosește.

Ca exemplu, să considerăm ruterele BGP din fig. 5-67(a). În particular, să considerăm tabela de dirijare a lui *F*. Să presupunem că el folosește calea *FGCD* pentru a ajunge la *D*. Când vecinii îi dau

informațiile de dirijare, ei oferă căile lor complete, ca în fig. 5-67(b) (pentru simplitate, este arătată doar destinația *D*).

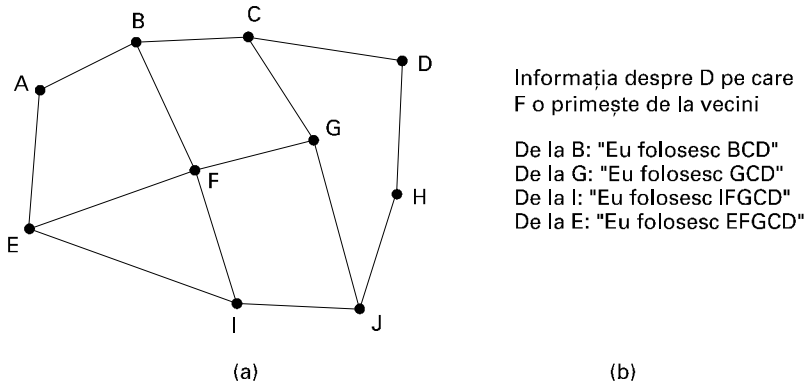


Fig. 5-67. (a) O mulțime de rutere BGP. (b) Informația trimisă lui *F*.

După ce sosesc toate căile de la vecini, *F* le examinează pentru a vedea care este cea mai bună. El elimină imediat căile de la *I* și *E*, din moment ce aceste căi trec chiar prin F. Alegerea este apoi între B și G. Fiecare ruter BGP conține un modul care examinează căile către o destinație dată și le atribuie scoruri, întorcând, pentru fiecare cale, o valoare pentru „distanța" către acea destinație. Orice cale care violează o constrângere politică primește automat un scor infinit. Apoi, ruterul adoptă calea cu cea mai scurtă distanță. Funcția de acordare a scorurilor nu este parte a protocolului BGP și poate fi orice funcție doresc administratorii de sistem.

BGP rezolvă ușor problema numără-la-infinit care chinuie alți algoritmi bazați pe vectori distanță. De exemplu, să presupunem că *G* se defectează sau că linia *FG* cade. Atunci *F* primește căi de la ceilalți trei vecini rămași. Aceste rute sunt *BCD*, *IFGCD* și *EFGCD*. El poate observa imediat că ultimele două căi sunt inutile, din moment ce trec chiar prin *F*, așa că alege *FBCD* ca noua sa cale. Alți algoritmi bazați pe vectori distanță fac de multe ori alegerea greșită, pentru că ei nu pot spune care din vecinii lor au căi independente către destinație și care nu. Definirea BGP-ului se găsește în RFC 1771 și 1774.

5.6.6 Trimiterea multiplă în Internet

Comunicația IP normală este între un emițător și un receptor. Cu toate acestea, pentru unele aplicații, este util ca un proces să fie capabil să trimită simultan unui număr mare de receptori. Exemple sunt actualizarea bazelor de date distribuite multiplicate, transmiterea cotărilor de la bursă mai multor agenți de bursă, tratarea convorbirilor telefonice de tipul conferințelor digitale (așadar cu mai mulți participanți).

IP-ul suportă trimiterea multiplă (multicast), folosind adrese de clasă D. Fiecare adresă de clasă D identifică un grup de gazde. Pentru identificarea grupurilor sunt disponibili douăzeci și opt de biți, așa încât pot exista în același moment peste 250 milioane de grupuri. Când un proces trimite un pachet unei adrese de clasă D, se face cea mai bună încercare pentru a-l livra tuturor membrilor grupului adresat, dar nu sunt date garanții. Unii membri pot să nu primească pachetul.

Sunt suportate două tipuri de adrese de grup: adrese permanente și adrese temporare. Un grup permanent există întotdeauna și nu trebuie configurat. Fiecare grup permanent are o adresă de grup permanentă. Câteva exemple de adrese de grup permanente sunt:

224.0.0.1 Toate sistemele dintr-un LAN.
224.0.0.2 Toate ruterele dintr-un LAN.
224.0.0.5 Toate ruterele OSPF dintr-un LAN.
224.0.0.6 Toate ruterele desemnate dintr-un LAN.

Grupurile temporare trebuie create înainte de a fi utilizate. Un proces poate cere gazdei sale să intre într-un anume grup. De asemenea, el poate cere gazdei sale să părăsească grupul. Când ultimul proces părăsește un grup, acel grup nu mai este prezent pe calculatorul său gazdă. Fiecare gazdă memorează căror grupuri aparțin la un moment dat procesele sale.

Trimiterea multiplă este implementată de rutere speciale de trimitere multiplă, care pot sau nu coabita cu ruterele standard. Cam o dată pe minut, fiecare ruter de trimitere multiplă face o trimitere multiplă hardware (la nivel legătură de date) pentru gazdele din LAN-ul său (adresa 224.0.0.1), cerându-le să raporteze căror grupuri aparțin procesele lor la momentul respectiv. Fiecare gazdă trimite înapoi răspunsuri pentru toate adresele de clasă D de care este interesată.

Aceste pachete de întrebare și răspuns folosesc un protocol numit **IGMP** (**Internet Group Management Protocol** - Protocol de gestiune a grupurilor Internet), care se aseamănă întrucâtva cu ICMP. El are numai două tipuri de pachete: întrebare și răspuns, fiecare cu un format fix, simplu, care conține unele informații de control în primul cuvânt al câmpului informație utilă și o adresă de clasă D în al doilea cuvânt. El este descris în RFC 1112.

Dirijarea cu trimitere multiplă este realizată folosind arbori de acoperire. Fiecare ruter de dirijare multiplă schimbă informații cu vecinii săi, folosind un protocol modificat bazat pe vectori distanță cu scopul ca fiecare să construiască pentru fiecare grup un arbore de acoperire care să acopere toți membrii grupului. Pentru a elimina ruterele și rețelele neinteresate de anumite grupuri, se utilizează diverse optimizări de reducere a arborelui. Pentru evitarea deranjării nodurilor care nu fac parte din arborele de acoperire, protocolul folosește intensiv trecerea prin tunel.

5.6.7 IP mobil

Mulți utilizatori ai Internet-ului au calculatoare portabile și vor să rămână conectați la Internet atunci când vizitează un sit Internet aflat la distanță, și chiar și pe drumul dintre cele două. Din nefericire, sistemul de adresare IP face lucrul la depărtare de casă mai ușor de zis decât de făcut. În această secțiune vom examina problema și soluția. O descriere mai detaliată este dată în (Perkins, 1998a).

Problema apare chiar în schema de adresare. Fiecare adresă IP conține un număr de rețea și un număr de gazdă. De exemplu, să considerăm mașina cu adresa IP 160.80.40.20/16. Partea 160.80 indică numărul de rețea (8272 în notație zecimală). Ruterele din toată lumea au tabele de dirijare care spun ce linie se folosește pentru a ajunge la rețeaua 160.80. Oricând vine un pachet cu adresa IP destinație de forma 160.80.xxx.yyy, pachetul pleacă pe respectiva linie.

Dacă, dintr-o dată, mașina cu adresa respectivă este transferată într-un alt loc din Internet, pachetele vor continua să fie dirijate către LAN-ul (sau ruterul) de acasă. Proprietarul nu va mai primi poștă electronică și așa mai departe. Acordarea unei noi adrese IP mașinii, adresă care să corespundă cu noua sa locație, nu este atractivă pentru că ar trebui să fie informate despre schimbare un mare număr de persoane, programe și baze de date.

O altă abordare este ca ruterele să facă dirijarea folosind adresa IP completă, în loc să folosească numai adresa rețelei. Cu toate acestea, această strategie ar necesita ca fiecare ruter să aibă milioane de intrări în tabele, la un cost astronomic pentru Internet.

Când oamenii au început să ceară posibilitatea de a-și conecta calculatoarele portabile oriunde s-ar duce, IETF a constituit un Grup de Lucru pentru a găsi o soluție. Grupul de Lucru a formulat rapid un număr de obiective considerate necesare în orice soluție. Cele majore au fost:

1. Fiecare gazdă mobilă trebuie să fie capabilă sa folosească adresa sa IP de baza oriunde.
2. Nu au fost permise schimbări de programe pentru gazdele fixe.
3. Nu au fost permise schimbări pentru programele sau tabelele ruterelor.
4. Cele mai multe pachete pentru gazdele mobile nu ar trebui să facă ocoluri pe drum.
5. Nu trebuie să apară nici o suprasolicitare când o gazdă mobilă este acasă.

Soluția aleasă este cea descrisă în secțiunea 5.2.8. Pentru a o recapitula pe scurt, fiecare sit care dorește să permită utilizatorilor săi să se deplaseze trebuie să asigure un agent local. Fiecare sit care dorește să permită accesul vizitatorilor trebuie să creeze un agent pentru străini. Când o gazdă mobilă apare într-un sit străin, ea contactează gazda străină de acolo și se înregistrează. Gazda străină contactează apoi agentul local al utilizatorului și îi dă o **adresă a intermediarului**, în mod normal adresa IP proprie a agentului pentru străini.

Când un pachet ajunge în LAN-ul de domiciliu al utilizatorului, el vine la un ruter atașat la LAN. Apoi ruterul încearcă să localizeze gazda în mod uzual, prin difuzarea unui pachet ARP întrebând, de exemplu: „Care este adresa Ethernet a lui 160.80.40.20?” Agentul local răspunde la această întrebare dând propria adresă Ethernet. Apoi ruterul trimite pachetele pentru 160.80.40.20 la agentul local. El, în schimb, le trimite prin tunel la adresa intermediarului prin încapsularea lor în câmpul informație utilă al unui pachet IP adresat agentului pentru străini. După aceasta, agentul pentru străini le desface și le livrează la adresa de nivel legătură de date a gazdei mobile. În plus, agentul local dă emițătorului adresa intermediarului, așa încât viitoarele pachete pot fi trimise prin tunel direct la agentul pentru străini. Această soluție răspunde tuturor cerințelor expuse mai sus.

Probabil că merită menționat un mic amănunt. În momentul în care gazda mobilă se mută, probabil că ruterul are adresa ei Ethernet (care în curând va fi invalidă) memorată în memoria ascunsă. Pentru a înlocui această adresă Ethernet cu cea a agentului local, se folosește un truc numit **ARP gratuit**. Acesta este un mesaj special, nesolicitat, către ruter pe care îl determină să schimbe o anumită intrare din memoria ascunsă, în acest caz cea a gazdei mobile care urmează să plece. Când, mai târziu, gazda mobilă se întoarce, este folosit același truc pentru a actualiza din nou memoria ascunsă a ruterului.

Nu există nimic în proiect care să împiedice o gazdă mobilă să fie propriul său agent pentru străini, dar această abordare funcționează numai dacă gazda mobilă (în postura sa de agent pentru străini) este conectată logic în Internet la situl său curent. De asemenea, ea trebuie să fie capabilă să obțină pentru folosire o adresă (temporară) de intermediar. Acea adresă IP trebuie să aparțină LAN-ului în care este atașată în mod curent.

Soluția IETF pentru gazde mobile rezolvă un număr de alte probleme care nu au fost menționate până acum. De exemplu, cum sunt localizați agenții? Soluția este ca fiecare agent să-și difuzeze periodic adresa și tipul de serviciu pe care dorește să-l ofere (de exemplu, agent local, pentru străini sau amândouă). Când o gazdă mobilă ajunge undeva, ea poate asculta așteptând aceste difuzări, numite **anunțuri**. Ca o alternativă, ea poate difuza un pachet prin care își anunță sosirea și să spere că agentul pentru străini local îi va răspunde.

O altă problemă care a trebuit rezolvată este cum să se trateze gazdele mobile nepoliticoase, care pleacă fără să spună la revedere. Soluția este ca înregistrarea să fie valabilă doar pentru un interval de timp fixat. Dacă nu este reîmprospătată periodic, ea expiră și ca urmare gazda străină poate să-și curețe tabelele.

O altă problemă este securitatea. Când un agent local primește un mesaj care-i cere să fie amabil să retrimită toate pachetele Robertei la o anume adresă IP, ar fi mai bine să nu se supună decât dacă este convins că Roberta este sursa acestei cereri și nu altcineva încercând să se dea drept ea. În acest scop sunt folosite protocoale criptografice de autentificare. Vom studia asemenea protocoale în cap. 8.

Un punct final adresat de Grupul de Lucru se referă la nivelurile de mobilitate. Să ne imaginăm un avion cu o rețea Ethernet la bord folosită de către calculatoarele de navigare și de bord. În această rețea Ethernet există un ruter standard, care discută cu Internet-ul cablat de la sol printr-o legătură radio. Într-o bună zi, unui director de marketing isteț îi vine ideea să instaleze conectoare Ethernet în toate brațele fotoliilor, astfel încât și pasagerii cu gazde mobile să se poată conecta.

Acum avem două niveluri de mobilitate: calculatoarele proprii ale aeronavei, care sunt staționare în raport cu rețeaua Ethernet și calculatoarele pasagerilor, care sunt mobile în raport cu ea. În plus, ruterul de la bord este mobil în raport cu ruterele de la sol. Mobilitatea în raport cu un sistem care este la rândul său mobil este tratată folosind recursiv tunele.

5.6.8 IPv6

În timp ce CIDR și NAT îi mai pot acorda câțiva ani, toată lumea își dă seama că zilele IP-ului în forma curentă (IPv4) sunt numărate. În plus față de aceste probleme tehnice, există un alt aspect întrezărit în fundal. La începuturile sale, Internet-ul a fost folosit în mare măsură de universități, industria de vârf și de guvernul Statelor Unite (în mod special de Departamentul Apărării). O dată cu explozia interesului față de Internet începând de la mijlocul anilor 1990, a început să fie utilizat de un grup diferit de persoane, în special persoane cu cerințe diferite. Pe de o parte, numeroase persoane cu calculatoare portabile fără fir îl folosesc pentru a ține legătura cu baza de acasă. Pe de altă parte, o dată cu iminenta convergență a industriilor calculatoarelor, comunicațiilor și a distracțiilor, s-ar putea să nu mai fie mult până când fiecare telefon sau televizor din lume va fi un nod Internet, producând un miliard de mașini folosite pentru audio și video la cerere. În aceste condiții, a devenit clar că IP-ul trebuie să evolueze și să devină mai flexibil.

Observând aceste probleme la orizont, IETF a început să lucreze în 1990 la o nouă versiune de IP, una care să nu își epuizeze niciodată adresele, să rezolve o gamă largă de alte probleme și să fie totodată mai flexibilă și mai eficientă. Obiectivele majore au fost:

1. Să suporte miliarde de gazde, chiar cu alocare ineficientă a spațiului de adrese.
2. Să reducă dimensiunea tabelelor de dirijare.
3. Să simplifice protocolul, pentru a permite ruterelor să proceseze pachetele mai rapid.
4. Să asigure o securitate mai bună (autentificare și confidențialitate) față de IP-ul curent.
5. Să acorde o mai mari atenție tipului de serviciu, în special pentru datele de timp real.
6. Să ajute trimiterea multiplă, permițând specificarea de domenii.
7. Să creeze condițiile pentru ca o gazdă să poată migra fără schimbarea adresei sale.
8. Să permită evoluția protocolului în viitor.
9. Să permită coexistența noului și vechiului protocol pentru câțiva ani.

Pentru a găsi un protocol care să îndeplinească toate aceste cerințe, IETF a emis o cerere de propuneri și discuții în RFC 1550. Au fost primite douăzeci și unu de răspunsuri, nu toate din ele propuneri complete. Până în decembrie 1992, au ajuns pe masa discuțiilor șapte propuneri serioase. Ele variau de la efectuarea de mici cârpeli la IP până la renunțarea completă la el și înlocuirea cu un protocol complet nou.

O propunere a fost folosirea TCP peste CLNP, care cu cei 160 de biți de adresă ai săi ar fi oferit spațiu de adrese suficient pentru totdeauna și ar fi unificat două protocoale majore de nivel rețea. Cu toate acestea, multe persoane au simțit că aceasta ar fi fost o acceptare a faptului că, de fapt, a fost făcut ceva chiar bine în lumea OSI, o afirmație considerată incorectă din punct de vedere politic în cercurile Internet. CLNP a fost modelat apropiat de IP, așa încât cele două nu sunt chiar atât de diferite. De fapt, protocolul care a fost ales în final diferă de IP cu mult mai mult decât diferă CLNP. O altă lovitură împotriva CLNP a fost slabul suport pentru tipuri de servicii, ceva necesar pentru transmiterea eficientă de multimedia.

Trei din cele mai bune propuneri au fost publicate în *IEEE Network* (Deering, 1993; Francis, 1993 și Katz și Ford, 1993). După multe discuții, revizii și manevre de culise, a fost selectată o versiune combinată modificată a propunerilor lui Deering și Francis, până atunci numită **SIPP** (**Simple Internet Protocol Plus** - Protocol simplu, îmbunătățit, pentru Internet) și i s-a dat numele de **IPv6**.

IPv6 îndeplinește obiectivele destul de bine. El menține caracteristicile bune ale IP-ului, le elimină sau atenuează pe cele rele și adaugă unele noi acolo unde este nevoie. În general, IPv6 nu este compatibil cu IPv4, dar el este compatibil cu celelalte protocoale Internet auxiliare, incluzând TCP, UDP, ICMP, IGMP, OSPF, BGP și DNS, câteodată fiind necesare mici modificări (majoritatea pentru a putea lucra cu adrese mai lungi). Principalele trăsături ale IPv6 sunt discutate mai jos. Mai multe informații despre el pot fi găsite în RFC 2460 până la RFC 2466.

În primul rând și cel mai important, IPv6 are adrese mai lungi decât IPv4. Ele au o lungime de 16 octeți, ceea ce rezolvă problema pentru a cărei soluționare a fost creat IPv6: să furnizeze o sursă efectiv nelimitată de adrese Internet. În curând vom spune mai multe despre adrese.

A doua mare îmbunătățire a lui IPv6 este simplificarea antetului. El conține numai 7 câmpuri (față de 13 în IPv4). Această schimbare permite ruterelor să prelucreze pachetele mai rapid, îmbunătățind astfel productivitatea și întârzierea. De asemenea, vom discuta în curând și antetul.

A treia mare îmbunătățire a fost suportul mai bun pentru opțiuni. Această schimbare a fost esențială în noul antet, deoarece câmpurile care erau necesare anterior sunt acum opționale. În plus, modul în care sunt reprezentate opțiunile este diferit, ușurând ruterelor saltul peste opțiunile care nu le sunt destinate. Această caracteristică accelerează timpul de prelucrare a pachetelor.

Un al patrulea domeniu în care IPv6 reprezintă un mare progres este în securitate. IETF a avut porția sa de povești de ziar despre copii precoce de 12 ani care își folosesc calculatoarele personale pentru a sparge bănci sau baze militare în tot Internet-ul. A existat un sentiment puternic că ar trebui făcut ceva pentru a îmbunătăți securitatea. Autentificarea și confidențialitatea sunt trăsături cheie ale noului IP. Ulterior ele au fost adaptate și în IPv4, astfel că în domeniul securității diferențele nu mai sunt așa de mari.

În final, a fost acordată o mai mare atenție calității serviciilor. În trecut s-au făcut eforturi, fără prea mare tragere de inimă, dar acum, o dată cu creșterea utilizării multimedia în Internet, presiunea este și mai mare.

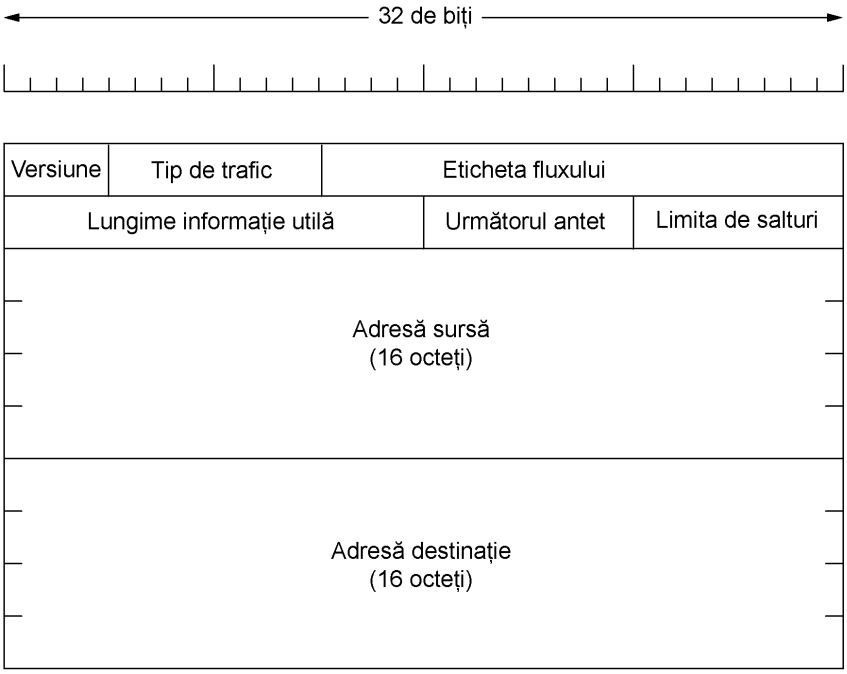


Fig. 5-68. Antetul fix IPv6 (obligatoriu).

Antetul principal IPv6

Antetul IPv6 este prezentat în fig. 5-68. Câmpul *Versiune* este întotdeauna 6 pentru IPv6 (și 4 pentru IPv4). În timpul perioadei de tranziție de la IPv4, care va lua probabil un deceniu, ruterele vor fi capabile să examineze acest câmp pentru a spune ce tip de pachet analizează. Ca un efect lateral, acest test irosește câteva instrucțiuni pe drumul critic, așa încât multe implementări vor încerca să-l evite prin folosirea unui câmp din antetul legăturii de date ca să diferențieze pachetele IPv4 de pachetele IPv6. În acest mod, pachetele pot fi transmise direct rutinei de tratare de nivel rețea corecte. Cu toate acestea, necesitatea ca nivelul legătură de date să cunoască tipurile pachetelor nivelului rețea contravine complet principiul de proiectare care spune că fiecare nivel nu trebuie să cunoască semnificația biților care îi sunt dați de către nivelul de deasupra. Discuțiile dintre taberele „Fă-o corect” și „Fă-o repede” vor fi, fără îndoială, lungi și virulente.

Câmpul *Tip de trafic* (*Traffic class*) este folosit pentru a distinge între pachetele care au diverse cerințe de livrare în timp real. Un câmp cu acest scop a existat în IP de la început, dar a fost implementat sporadic de către rutere. În acest moment se desfășoară experimente pentru a determina cum poate fi utilizat cel mai bine pentru transmisii multimedia.

Câmpul *Eticheta fluxului* este încă experimental, dar va fi folosit pentru a permite unei surse și unei destinații să stabilească o pseudo-conexiune cu proprietăți și cerințe particulare. De exemplu, un șir de pachete de la un proces de pe o anumită gazdă sursă către un anumit proces pe o anumită gazdă destinație poate avea cerințe de întârziere stricte și din acest motiv necesită capacitate de transmisie rezervată. Fluxul poate fi stabilit în avans și poate primi un identificator. Când apare un pachet cu o *Etichetă a fluxului* diferită de zero, toate ruterele pot să o caute în tabelele interne pentru

a vedea ce tip de tratament special necesită. Ca efect, fluxurile sunt o încercare de a combina două moduri: flexibilitatea unei subrețele cu datagrame și garanțiile unei subrețele cu circuite virtuale.

Fiecare flux este desemnat de adresa sursă, adresa destinație și numărul de flux, așa încât, între o pereche dată de adrese IP pot exista mai multe fluxuri active în același timp . De asemenea, în acest mod, chiar dacă două fluxuri venind de la gazde diferite, dar cu același număr de flux trec prin același ruter, ruterul va fi capabil să le separe folosind adresele sursă și destinație. Se așteaptă ca numerele de flux să fie alese aleator, în loc de a fi atribuite secvențial începând cu 1, pentru că se așteaptă ca ruterele să le folosească în tabele de dispersie.

Câmpul *Lungimea informației utile* spune câți octeți urmează după antetul de 40 de octeți din fig. 5-68. Numele a fost schimbat față de câmpul *Lungime totală* din IPv4 deoarece semnificația este ușor modificată: cei 40 de octeți nu mai sunt parte a lungimii (așa cum erau înainte).

Câmpul *Antetul următor* dă de gol proiectanții. Motivul pentru care antetul a putut fi simplificat este că există antete de extensie suplimentare (opționale). Acest câmp spune care din cele șase antete (actuale) de extensie, dacă există vreunul, urmează după cel curent. Dacă acest antet este ultimul antet IP, câmpul *Antetul următor* spune cărui tip de protocol (de exemplu TCP, UDP) i se va transmite pachetul.

Câmpul *Limita salturilor* este folosit pentru a împiedica pachetele să trăiască veșnic. El este, în practică, identic cu câmpul *Timp de viață* din IPv4, și anume un câmp care este decrementat la fiecare salt dintr-o rețea în alta. În teorie, în IPv4 era un timp în secunde, dar nici un ruter nu-l folosea în acest mod, așa încât numele a fost modificat pentru a reflecta modul în care este de fapt folosit.

Apoi urmează câmpurile *Adresă sursă* și *Adresă destinație*. Propunerea originală a lui Deering, SIP, folosea adrese de 8 octeți, dar în timpul procesului de evaluare, multe persoane au simțit că adresele de 8 octeți s-ar putea epuiza în câteva decenii, în timp ce adresele de 16 octeți nu s-ar epuiza niciodată. Alte persoane au argumentat că 16 octeți ar fi un exces, în timp ce alții încurajau folosirea adreselor de 20 de octeți pentru a fi compatibile cu protocolul datagramă OSI. O altă grupare dorea adrese de dimensiune variabilă. După multe discuții, s-a decis că adresele cu lungime fixă de 16 octeți sunt cel mai bun compromis.

Pentru scrierea adreselor de 16 octeți a fost inventată o nouă notație. Ele sunt scrise ca opt grupuri de câte patru cifre hexazecimale cu semnul : (două puncte) între grupuri, astfel:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Din moment ce multe adrese vor avea multe zerouri în interiorul lor, au fost autorizate trei optimizări. Mai întâi, zerourile de la începutul unui grup pot fi omise, astfel încât 0123 poate fi scris ca 123. În al doilea rând, unul sau mai multe grupuri de 16 zerouri pot fi înlocuite de o pereche de semne două puncte (:). Astfel, adresa de mai sus devine acum

8000::123:4567:89AB:CDEF

În final, adresele IPv4 pot fi scrise ca o pereche de semne două puncte și un număr zecimal în vechea formă cu punct, de exemplu

::192.31.20.46

Probabil că nu este necesar să fim atât de expliciți asupra acestui lucru, dar există o mulțime de adrese de 16 octeți. Mai exact, sunt 2^{128} adrese, care reprezintă aproximativ $3x10^{38}$. Dacă întreaga planetă, pământ și apă, ar fi acoperite cu calculatoare, IPv6 ar permite $7x10^{23}$ adrese IP pe metru pătrat. Studenții de la chimie vor observa că acest număr este mai mare decât numărul lui Avogadro. Deși nu a

existat intenția de a da fiecărei molecule de pe suprafața planetei adresa ei IP, nu suntem chiar așa de departe de aceasta.

În practică, spațiul de adrese nu va fi folosit eficient, așa cum nu este folosit spațiul de adrese al numerelor de telefon (prefixul pentru Manhattan, 212, este aproape plin, dar cel pentru Wyoming, 307, este aproape gol). În RFC 3194, Durand și Huitema a calculat că, folosind ca referință alocarea numerelor de telefon, chiar și în cel mai pesimist scenariu, vor fi totuși mult peste 1000 de adrese IP pe metru pătrat de suprafață planetară (pământ sau apă). În orice scenariu credibil, vor fi trilioane de adrese pe metru pătrat. Pe scurt, pare improbabil că vom epuiza adresele în viitorul previzibil.

Este instructiv să comparăm antetul IPv4 (fig. 5-53) cu antetul IPv6 (fig. 5-68) pentru a vedea ce a fost eliminat în IPv6. Câmpul *IHL* a dispărut pentru că antetul IPv6 are o lungime fixă. Câmpul *Protocol* a fost scos pentru că în câmpul *Antetul următor* se indică ce urmează după ultimul antet IP (de exemplu, un segment TCP sau UDP).

Toate câmpurile referitoare la fragmentare au fost eliminate, deoarece IPv6 are o abordare diferită a fragmentării. Pentru început, toate gazdele și ruterele care sunt conforme cu IPv6 trebuie să determine dinamic mărimea datagramei care va fi folosită. Această regulă face ca, de la început, fragmentarea să fie mai puțin probabilă. De asemenea minimul a fost mărit de la 576 la 1280 pentru a permite date de 1024 de octeți și mai multe antete. În plus, când o gazdă trimite un pachet IPv6 care este prea mare, ruterul care este incapabil să îl retransmită trimite înapoi un mesaj de eroare în loc să fragmenteze pachetul. Acest mesaj de eroare îi spune gazdei să spargă toate pachetele viitoare către acea destinație. Este mult mai eficient să obligi gazdele să trimită de la bun început pachete corecte dimensional, decât să obligi ruterele să le fragmenteze din mers.

În sfârșit, câmpul *Sumă de control* este eliminat deoarece calculul acesteia reduce mult performanțele. Datorită rețelelor fiabile folosite acum, combinate cu faptul că nivelurile de legătură de date și de transport au în mod normal propriile sume de control, valoarea a încă unei sume de control nu merita prețul de performanță cerut. Eliminarea tuturor acestor caracteristici a avut ca rezultat un protocol de nivel rețea simplu și eficient. Astfel, obiectivul lui IPv6 – un protocol rapid, dar flexibil, cu o bogăție de spațiu de adrese – a fost atins prin acest proiect.

Antete de extensie

Câteva din câmpurile care lipsesc sunt încă necesare ocazional, astfel încât IPv6 a introdus conceptul de **antet de extensie** (opțional). Aceste antete pot fi furnizate pentru a oferi informații suplimentare, dar codificate într-un mod eficient. În prezent sunt definite șase tipuri de antete de extensie, prezentate în fig. 5-69. Fiecare este opțional, dar dacă sunt prezente mai multe, ele trebuie să apară imediat după antetul fix și, preferabil, în ordinea prezentată.

Antet de extensie	Descriere
Opțiuni salt-după-salt	Diverse informații pentru rutere
Opțiuni pentru destinație	Informații suplimentare pentru destinație
Dirijare	Calea, parțială sau totală, de urmat
Fragmentare	Gestiunea fragmentelor datagramelor
Autentificare	Verificarea identității emițătorului
Informație de siguranță criptată	Informații despre conținutul criptat

Fig. 5-69. Antetele de extensie IPv6.

Unele dintre antete au un format fix; altele conțin un număr variabil de câmpuri de lungime variabilă. Pentru acestea, fiecare element este codificat ca un tuplu (Tip, Lungime, Valoare). *Tipul* este

un câmp de 1 octet care spune ce opțiune este aceasta. Valorile *Tipului* au fost alese astfel, încât primii 2 biți spun ruterelor care nu știu cum să proceseze opțiunea ce anume să facă. Variantele sunt: sărirea opțiunii; eliminarea pachetului; eliminarea pachetului și trimiterea înapoi a unui pachet ICMP; la fel ca mai înainte, doar că nu se trimit pachete ICMP pentru adrese de trimitere multiplă (pentru a preveni ca un pachet eronat de multicast să genereze milioane de răspunsuri ICMP).

Câmpul *Lungime* este de asemenea un câmp de lungime 1 octet. El precizează cât de lungă este valoarea (de la 0 la 255). *Valoarea* este orice informație necesară, până la 255 octeți.

Antetul salt-după-salt este folosit pentru informații ce trebuie examinate de toate ruterele de pe cale. Până acum a fost definită o opțiune: suportul pentru datagrame ce depășesc de 64K. Formatul acestui antet este prezentat în fig. 5-70. Atunci când este folosit, câmpul *Lungimea informației utile* din antetul fix este zero.

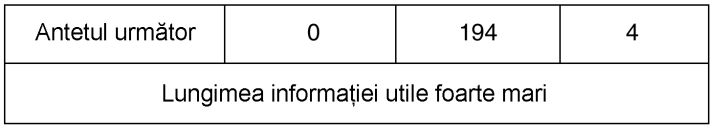


Fig. 5-70. Antetul de extensie salt-după-salt pentru datagrame mari (jumbograme).

Ca toate antetele de extensie, acesta începe cu un octet care spune ce tip de antet este următorul. Acest octet este urmat de unul care spune cât de lung este antetul salt-după-salt în octeți, excluzând primii 8 octeți, care sunt obligatorii. Toate extensiile încep la fel.

Următorii 2 octeți arată că această opțiune definește dimensiunea datagramei (codul 194) ca un număr de 4 octeți. Ultimii 4 octeți dau dimensiunea datagramei. Dimensiunile mai mici de 65.536 nu sunt permise și au ca rezultat eliminarea pachetului de către primul ruter, care va trimite înapoi un mesaj ICMP de eroare. Datagramele care folosesc acest antet de extensie sunt numite **jumbograme**. Folosirea jumbogramelor este importantă pentru aplicațiile supercalculatoarelor care trebuie să transfere gigaocteți de date eficient prin intermediul Internet-ului.

Antetul opțiunilor pentru destinație este prevăzut pentru câmpuri care trebuie interpretate numai de către gazda destinație. În versiunea inițială a IPv6 singura opțiune definită este cea de completare a acestui antet până la un multiplu de 8 octeți, așa că pentru început nu va fi folosit. El a fost inclus pentru a asigura posibilitatea ca un nou software al ruterului sau al gazdei să îl poate trata, în cazul în care cineva, cândva, se gândește la o opțiune pentru destinație.

Antetul de dirijare enumeră unul sau mai multe rutere care trebuie să fie vizitate pe calea spre destinație. Este foarte asemănător cu dirijarea aproximativă din IPv4 prin aceea că toate adresele listate trebuie vizitate în ordine, dar între acestea pot fi vizitate alte rutere nelistate. Formatul antetului de dirijare este prezentat în fig. 5-71.

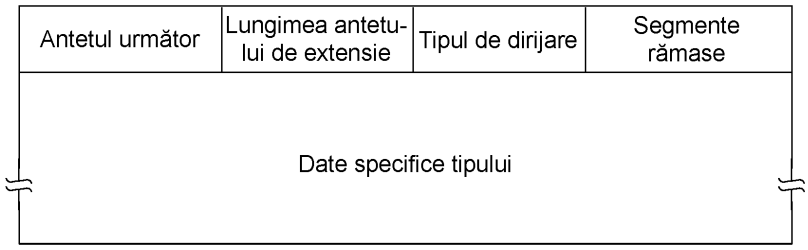


Fig. 5-71. Antetul de extensie pentru dirijare.

Primii 4 octeți ai antetului de extensie de dirijare conțin patru întregi de 1 octet. Câmpurile *Antet următor* și *Lungimea antetului* extensie au fost descrise anterior. Câmpul *Tipul dirijării* precizează formatul părții rămase din header. Tipul 0 arată că după primul cuvânt urmează un cuvânt rezervat pe 32 de biți, urmat de un număr de adrese IPv6. În viitor, în funcție de necesități, ar putea fi inventate alte tipuri. În final, câmpul *Segmente rămase* reține câte adrese din listă nu au fost vizitate și este decrementat de fiecare dată când este vizitată o adresă. Atunci când se ajunge la 0 pachetul este pe cont propriu, fără nici o indicație referitoare la ruta de urmat. De obicei, în acest punct este atât de aproape de destinație încât calea cea mai bună este evidentă.

Antetul fragment tratează fragmentarea într-un mod similar cu cel al IPv4. Antetul menține identificatorul datagramei, numărul de fragment și un bit care spune dacă mai urmează fragmente. În IPv6, spre deosebire de IPv4, numai gazda sursă poate fragmenta un pachet. Ruterele de pe cale nu pot face acest lucru. Deși această schimbare este o rupere filozofică majoră cu trecutul, ea simplifică munca ruterelor și permite ca dirijarea să se facă mai rapid. Așa cum s-a menționat mai sus, dacă un ruter este confruntat cu un pachet care este prea mare, el elimină pachetul și trimite un pachet ICMP înapoi la sursă. Această informație îi permite gazdei sursă să fragmenteze pachetul în bucăți mai mici folosind acest antet și apoi să reîncerce.

Antetul de autentificare oferă un mecanism prin care receptorul unui mesaj poate fi sigur de cel care l-a trimis. Informația utilă de siguranță criptată face posibilă criptarea conținutului unui pachet, astfel încât doar receptorul căruia îi este destinat poate să-l citească. Pentru a-și realiza misiunea aceste antete folosesc tehnici criptografice.

Controverse

Dat fiind procesul de proiectare deschisă și opiniile ferm susținute ale multora dintre persoanele implicate, nu ar trebui să fie o surpriză că multe din deciziile luate pentru IPv6 au fost foarte controversate. În cele ce urmează vom rezuma câteva dintre acestea. Pentru toate amănuntele tăioase, vezi RFC-urile.

Am menționat deja disputa legată de lungimea adresei. Rezultatul a fost un compromis: adrese de lungime fixă de 16 octeți.

O altă dispută s-a dus în jurul lungimii câmpului *Limita salturilor*. O tabără a simțit că limitarea numărului maxim de salturi la 255 (implicită în cazul folosirii unui câmp de 8 biți) ar fi o greșeală grosolană. Până la urmă, căi de 32 de salturi sunt obișnuite în zilele noastre, iar peste 10 ani pot fi obișnuite căi mult mai lungi. Aceste persoane au argumentat că folosirea unui spațiu de adrese enorm a fost clarviziune, dar folosirea unui contor minuscul de salturi a fost miopie. După părerea lor, cel mai mare păcat pe care îl poate comite un informatician este să ofere prea puțini biți într-un loc.

Răspunsul a fost că pot fi aduse argumente pentru lărgirea fiecărui câmp, conducând la un antet umflat. De asemenea, funcția câmpului *Limita salturilor* este de a împiedica hoinăreala pachetelor pentru un timp îndelungat și 65.536 de salturi este mult prea mult. În cele din urmă, pe măsură ce Internet-ul crește, se vor construi din ce în ce mai multe legături de mare distanță, făcând posibilă ajungerea dintr-o țară în alta în cel mult o jumătate de duzină de salturi. Dacă este nevoie de mai mult de 125 de salturi pentru a ajunge de la sursă sau destinație la porțile lor internaționale, ceva este în neregulă cu coloanele vertebrale naționale. Adepții celor 8 biți au câștigat această luptă.

O altă problemă spinoasă a fost dimensiunea maximă a pachetului. Comunitatea supercalculatoarelor a dorit pachete mai mari de 64 KB. Când un supercalculator începe să transfere, aceasta înseamnă într-adevăr lucru serios și nu dorește să fie întrerupt după fiecare 64KB. Argumentul împotri-

va pachetelor mari este că dacă un pachet de 1 MB ajunge la o linie T1 de 1.5 Mbps, acel pachet va monopoliza linia pentru mai mult de 5 secunde, producând o întârziere semnificativă pentru utilizatorii interactivi care partajează linia. Aici s-a ajuns la un compromis: pachetele normale au fost limitate la 64 KB, dar antetul de extensie salt-după-salt poate fi folosit pentru a permite jumbograme.

Un al treilea punct fierbinte a fost eliminarea sumei de control IPv4. Unele persoane au asimilat această mutare cu eliminarea frânelor de la mașină. Făcând acest lucru, mașina devine mai ușoară, astfel încât poate merge mai repede, dar dacă intervine ceva neașteptat, apar probleme.

Argumentul împotriva sumei de control a fost că orice aplicație care are într-adevăr grijă de integritatea datelor trebuie oricum să aibă o sumă de control la nivelul transport, așa încât menținerea a încă o sumă în IP (în plus față de suma de control a nivelului legătură de date) este un exces. Mai mult, experiența a arătat că în IPv4 calculul sumei de control IP era foarte costisitoare. Tabăra împotriva sumei de control a învins de această dată, deci IPv6 nu are o sumă de control.

Gazdele mobile au fost de asemenea un punct de conflict. Dacă un calculator portabil face jumătate din ocolul lumii, poate continua el să opereze la destinație cu aceeași adresă IPv6 sau trebuie să folosească o schemă cu agenți locali și agenți pentru străini? De asemenea, gazdele mobile introduc asimetrii în sistemul de dirijare. Se poate foarte bine întâmpla ca un mic calculator mobil să audă semnalul puternic trimis de un ruter staționar, dar ruterul staționar nu poate auzi semnalul slab trimis de gazda mobilă. În consecință, unele persoane au dorit să includă în IPv6 suport explicit pentru gazde mobile. Acest efort a eșuat pentru că nu s-a putut ajunge la un consens pentru o propunere concretă.

Probabil că cea mai mare bătălie a fost pentru securitate. Toată lumea a fost de acord că este necesară. Războiul a fost pentru unde și cum. Mai întâi unde. Argumentul pentru plasarea la nivelul rețea este că devine un serviciu standard pe care toate aplicațiile îl pot folosi fără o planificare prealabilă. Argumentul contra este că, în general, aplicațiile cu adevărat sigure doresc cel puțin criptare capăt-la-capăt, în care aplicația sursă criptează și aplicația destinație decriptează. Altfel, utilizatorul este la mila unor implementări potențial pline de pene a nivelurilor rețea, implementări asupra cărora nu are nici un control. Răspunsul la acest argument este că aceste aplicații pot să se abțină de la folosirea facilităților de securitate IP și să-și facă treaba ele însele. Replica la aceasta este că persoanele care nu au încredere că rețeaua face treaba cum trebuie, nu doresc să plătească prețul unor implementări de IP lente, greoaie, care au această facilitate, chiar dacă este dezactivată.

Un alt aspect al disputei unde să fie pusă securitatea este legat de faptul că multe țări (dar nu toate) au legi de export severe referitoare la criptografie. Unele, notabil în Franța și Irak, reduc în mare măsură folosirea internă a criptografiei, așa încât oamenii nu pot avea secrete față de poliție. Ca rezultat, orice implementare de IP care utilizează un sistem criptografic suficient de puternic pentru a fi de mare valoare nu poate fi exportat din Statele Unite (și multe alte țări) clienților din lumea întreagă. Necesitatea menținerii a două seturi de programe, unul pentru uz intern și altul pentru export, este un fapt ce întâmpină o opoziție viguroasă din partea firmelor de calculatoare.

Un punct asupra căruia nu au existat controverse este că nimeni nu se așteaptă ca Internet-ul bazat pe IPv4 să fie închis într-o duminică dimineața și să repornească ca un Internet bazat pe IPv6 luni dimineață. În schimb, vor fi convertite „insule” izolate de IPv6, inițial comunicând prin tunele. Pe măsură ce insulele IPv6 cresc, ele vor fuziona în insule mai mari. Până la urmă, toate insulele vor fuziona și Internet-ul va fi convertit complet. Dată fiind investiția masivă în rutere IPv4 în folosință curentă, procesul de conversie va dura probabil un deceniu. Din acest motiv s-a depus o enormă cantitate de efort pentru a asigura că această tranziție va fi cât mai puțin dureroasă posibil. Pentru mai multe informații despre IPv6, vezi (Loshin, 1999).

5.7 REZUMAT

Nivelul rețea furnizează servicii nivelului transport. El se poate baza fie pe circuite virtuale, fie pe datagrame. În ambele cazuri, principala sa sarcină este dirijarea pachetelor de la sursă la destinație. În subrețelele bazate pe circuite virtuale, decizia de dirijare se ia atunci când este stabilit circuitul. În subrețelele bazate pe datagrame decizia este luată pentru fiecare pachet.

În rețelele de calculatoare sunt folosiți mulți algoritmi de dirijare. Algoritmii statici includ dirijarea pe calea cea mai scurtă și inundarea. Algoritmii dinamici include dirijarea după vectorul distanțelor și dirijarea după starea legăturii. Majoritatea rețelelor actuale folosesc unul dintre acești algoritmi. Alte teme importante referitoare la dirijare sunt dirijarea ierarhică, dirijarea pentru gazde mobile, dirijarea pentru difuzare, dirijarea multidestinație și cea în rețele punct-la-punct.

Subrețelele pot deveni cu ușurință congestionate, mărind întârzierea și micșorând productivitatea pentru pachete. Proiectanții rețelelor încearcă să evite congestia printr-o proiectare adecvată. Tehnicile includ politica de retransmitere, folosirea memoriei ascunse, controlul fluxului și altele. Dacă apare congestia, ea trebuie să fie tratată. Pot fi trimise înapoi pachete de șoc, încărcarea poate fi eliminată sau se pot aplica alte metode.

Următorul pas dincolo de simpla tratare a congestiei este încercarea de a se ajunge la calitatea promisă a serviciului. Metodele care pot fi folosite pentru aceasta includ folosirea zonelor tampon la client, modelarea traficului, rezervarea resurselor și controlul accesului. Abordările care au fost proiectate pentru o bună calitate a serviciului includ servicii integrate (ca RSVP), servicii diferențiate și MPLS.

Rețelele diferă prin multe caracteristici, așa că atunci când se conectează mai multe rețele, pot să apară probleme. Uneori problemele pot fi evitate prin trecerea prin tunel a unui pachet ce traversează o rețea ostilă, dar dacă rețeaua sursă și cea destinație diferă, această abordare eșuează. Atunci când rețele diferite au dimensiunile maxime ale pachetelor diferite, se poate produce fragmentarea.

Internet-ul posedă o mare varietate de protocoale legate de nivelul rețea. Acestea includ protocolul de transport al datelor, IP, dar și protocoalele de control ICMP, ARP și RARP și protocoalele de dirijare OSPF și BGP. Internet-ul va rămâne foarte repede fără adrese IP, așa că s-a dezvoltat o nouă versiune de IP, IPv6.

5.8 PROBLEME

1. Dați două exemple de aplicații pentru care este adecvat un serviciu orientat pe conexiune. Apoi dați două exemple pentru care un serviciu fără conexiuni este cel mai potrivit.

2. Există vreo situație în care un serviciu cu circuit virtual va (sau cel puțin ar putea) livra pachetele în altă ordine? Explicați.

3. Subrețelele bazate pe datagrame dirijează fiecare pachet ca pe o unitate separată, independentă de toate celelalte. Subrețelele bazate pe circuite virtuale nu trebuie să facă acest lucru, pentru că fiecare pachet de date urmează o cale predeterminată. Oare această observație înseamnă că

subrețelele bazate pe circuite virtuale nu au nevoie de capacitatea de a dirija pachetele izolate de la o sursă arbitrară către o destinație arbitrară? Explicați răspunsul dat.

4. Dați trei exemple de parametri ai protocolului care ar putea fi negociați atunci când este inițiată o conexiune.

5. Considerați următoarea problemă de proiectare, privind implementarea unui serviciu cu circuit virtual. Dacă în interiorul unei subrețele sunt folosite circuite virtuale, fiecare pachet de date trebuie să conțină un antet de 3 octeți, iar fiecare ruter trebuie să aloce 8 octeți de memorie pentru identificarea circuitelor. Dacă intern sunt folosite datagrame, sunt necesare antete de 15 octeți, dar nu este nevoie de spațiu pentru tabela ruterului. Capacitatea de transmisie costă 1 cent per 10^6 octeți, per salt. Memoria foarte rapidă pentru ruter poate fi cumpărată la prețul de 1 cent per octet și se depreciază peste doi ani (considerând numai orele de funcționare). Din punct de vedere statistic, o sesiune medie durează 1000 de secunde, iar în acest timp sunt transmise 200 de pachete. Un pachet mediu are nevoie de patru salturi. Care implementare este mai ieftină și cu cât?

6. Presupunând că toate ruterele și gazdele funcționează normal și că întregul software din rutere și gazde nu conține nici o eroare, există vreo șansă, oricât de mică, ca un pachet să fie livrat unei destinații greșite?

7. Considerați rețeaua din fig. 5-7, dar ignorați ponderile de pe linii. Presupuneți că algoritmul de rutare utilizat este cel de inundare. Listați toate rutele pe care le va parcurge un pachet trimis de la *A* la *D,* al cărui număr maxim de salturi este 3. De asemenea precizați câte noduri consumă inutil bandă de transmisie.

8. Formulați o euristică simplă pentru găsirea a două căi de la o sursă dată la o destinație dată care pot supraviețui pierderii oricărei linii de comunicație (presupunând că există două astfel de căi). Ruterele sunt considerate suficient de fiabile, deci nu este necesar să ne îngrijoreze posibilitatea căderii ruterelor.

9. Considerați subrețeaua din fig. 5-13(a). Se folosește dirijarea după vectorul distanțelor și următorii vectori tocmai au sosit la ruterul C: de la B: (5, 0, 8, 12, 6, 2); de la D: (16, 12, 6, 0, 9, 10); și de la E: (7, 6, 3, 9, 0, 4). Întârzierile măsurate către B, D și E, sunt 6, 3 și respectiv 5. Care este noua tabelă de dirijare a lui C? Precizați atât linia de ieșire folosită, cât și întârzierea presupusă.

10. Dacă întârzierile sunt înregistrate ca numere de 8 octeți într-o rețea cu 50 de rutere și vectorii cu întârzieri sunt schimbați de două ori pe secundă, cât din lărgimea de bandă a unei linii (duplex integral) este consumată de algoritmul distribuit de dirijare? Presupuneți că fiecare ruter are trei linii către alte rutere.

11. În fig. 5-14 rezultatul operației SAU logic a celor două mulțimi de biți ACF este 111 în fiecare linie. Este acesta doar un accident întâmplat aici sau este valabil pentru toate subrețelele, în toate împrejurările?

12. La dirijarea ierarhică cu 4800 de rutere, ce dimensiuni ar trebui alese pentru regiune și grup, astfel încât să se minimizeze dimensiunea tabelei de dirijare pentru o ierarhie cu trei niveluri?

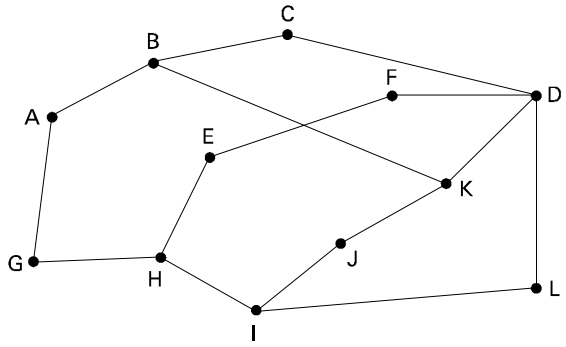
Un punct de pornire este ipoteza că o soluție cu k clustere de k regiuni de k rutere este aproape de optim, ceea ce înseamnă că valoarea k este aproximativ rădăcina cubică a lui 4800 (aproximativ 16). Folosiți încercări repetate pentru a verifica combinațiile cu toți cei trei parametri în vecinătatea lui 16.

13. În text s-a afirmat că atunci când un sistem gazdă mobil nu este acasă, pachetele trimise către LAN-ul de domiciliu sunt interceptate de agentul său local. Pentru o rețea IP pe un LAN 802.3, cum va realiza agentul local această interceptare?

14. Privind subrețeaua din fig. 5-6, câte pachete sunt generate de o difuzare de la B, folosind:
a) urmărirea căii inverse?
b) arborele de scufundare?

15. Fie rețeaua din fig. 5-16(a). Să ne imaginăm că între F și G este adăugată o nouă linie, dar arborele de scufundare din fig. 5-16(b) rămâne neschimbat. Ce modificări survin în fig. 5-16(c) ?

16. Calculați un arbore de acoperire pentru trimitere multiplă pentru ruterul C din rețeaua de mai jos pentru un grup cu membrii la ruterele A, B, C, D, E, F, I și K.



17. În fig. 5-20, difuzează vreodată nodurile H și I la căutarea pornită din A ?

18. Să presupunem că nodul B din fig. 5-20 tocmai a pornit și nu are nici o informație de dirijare în tabelele sale. Brusc, are nevoie de o cale către H. El va difuza pachete cu *TTL* setat la 1, 2, 3 și așa mai departe. De câte runde are nevoie pentru a găsi o cale ?

19. În cea mai simplă variantă a algoritmului Chord pentru căutarea punct-la-punct, căutările nu folosesc tabela de indicatori. În loc de aceasta, ele sunt lineare în jurul cercului în oricare direcție. Poate un nod determina cu precizie în ce direcție trebuie să caute ? Discutați răspunsul.

20. Fie cercul Chord din fig. 5-24. Să presupunem că nodul 10 pornește brusc. Afectează aceasta tabela de indicatori a nodului 1, și dacă da, cum ?

21. Ca un posibil mecanism de control al congestiei într-o subrețea ce folosește intern circuite virtuale, un ruter poate amâna confirmarea unui pachet primit până când (1) știe că ultima sa transmisie de-a lungul circuitului virtual a fost primită cu succes și (2) are un tampon liber. Pentru simplitate, să presupunem că ruterele utilizează un protocol stop-and-wait (pas-cu-pas) și că fie-

care circuit virtual are un tampon dedicat pentru fiecare direcție a traficului. Dacă este nevoie de T sec pentru a trimite un pachet (date sau confirmare) și sunt n rutere de-a lungul căii, care este viteza cu care pachetele sunt livrate gazdei destinație? Presupunem că erorile de transmisie sunt rare, iar conexiunea gazdă-ruter este infinit de rapidă.

22. O subrețea de tip datagramă permite ruterelor să elimine pachete de câte ori este necesar. Probabilitatea ca un ruter să renunțe la un pachet este p. Considerăm cazul unei gazde sursă conectate cu un ruter sursă, care este conectat cu un ruter destinație și apoi cu gazda destinație. Dacă unul dintre rutere elimină un pachet, până la urmă gazda sursă va depăși limita de timp și va încerca din nou. Dacă liniile gazdă-ruter și ruter-ruter sunt ambele numărate ca salturi, care este numărul mediu de:
a) salturi per transmisie pe care le face un pachet?
b) transmisii determinate de un pachet?
c) salturi necesare pentru un pachet primit?

23. Descrieți două diferențe majore dintre metoda bitului de avertizare și metoda RED.

24. Dați o explicație pentru faptul că algoritmul găleții găurite permite un singur pachet per tact, indiferent de cât de mare este pachetul.

25. Într-un sistem oarecare este utilizată varianta cu numărarea octeților a algoritmului găleții găurite. Regula este că pot fi trimise la fiecare tact un pachet de 1024 de octeți, două pachete de 512 octeți etc. Formulați o limitare serioasă a acestui sistem care nu a fost menționată în text.

26. O rețea ATM utilizează pentru modelarea traficului o schemă de tip găleată cu jetoane (token bucket). La fiecare 5 μsec în găleată este introdus un nou jeton. Fiecare jeton este asociat unei singure celule, care conține 48 octeți de date. Care este viteza maximă a datelor care poate fi asigurată?

27. Un calculator dintr-o rețea de 6 Mbps este guvernat de o schemă de tip găleată cu jetoane. Aceasta se umple cu viteza de 1 Mbps. Ea este umplută inițial la capacitatea maximă, cu 8 megabiți. Cât timp poate calculatorul să transmită cu întreaga viteză de 6 Mbps?

28. Să ne imaginăm o specificație de flux care are dimensiunea maximă a pachetului de 1000 de octeți, viteza găleții cu jetoane de 10 milioane de octeți/sec, capacitatea găleții de 1 milion de octeți și viteza maximă de transmisie de 50 de milioane de octeți/sec. Cât timp poate dura o rafală la viteza maximă?

29. Rețeaua din fig. 5-37 folosește RSVP cu arbori multidestinație pentru gazdele 1 și 2, după cum este ilustrat. Să presupunem că gazda 3 cere un canal cu lățimea de bandă de 2MB/sec pentru un flux de la gazda 1 și alt canal cu lărgimea de bandă de 1MB/sec pentru un flux de la gazda 2. În același timp gazda 4 cere un canal cu lărgimea de bandă de 2MB/sec pentru un flux de la gazda 1 și gazda 5 cere un alt cu lărgimea de bandă de 1MB/sec pentru un flux de la gazda 2. Ce lărgime de bandă totală va fi rezervată la ruterele A, B, C, E, H, J, K, L pentru aceste cereri?

30. Procesorul dintr-un ruter poate prelucra 2 milioane de pachete/sec. Încărcarea oferită lui este de 1,5 milioane pachete/sec. Dacă o rută de la sursă la destinație trece prin 10 rutere, cât timp se consumă în așteptare și pentru servirea de către procesoare?

31. Fie utilizatorul unor servicii diferențiate cu rutare expeditivă. Există o garanție că pachetele prioritare vor suferi o întârziere mai mică decât pachetele normale? De ce sau de ce nu?

32. Este nevoie de fragmentare în rețele concatenate bazate pe circuite virtuale sau numai în sisteme cu datagrame?

33. Trecerea prin tunel printr-o subrețea de circuite virtuale concatenate este simplă: ruterul multiprotocol de la un capăt stabilește circuitul virtual către celălalt capăt și trece pachetele prin el. Poate această trecere prin tunel să fie folosită și în subrețelele bazate pe datagrame? Dacă da, cum?

34. Să presupunem că gazda *A* este conectată la ruterul *R1*, *R1* este conectat la alt ruter *R2*, și *R2* este conectat la gazda *B*. Să presupunem că un mesaj TCP care conține 900 octeți de date și 20 de octeți de antet TCP este transmis codului IP aflat pe gazda *A* pentru a fi transmis lui *B*. Arătați câmpurile *Lungimea totală, Identificare*, *DF* ,*MF* și *Deplasamentul fragmentului* din antetul IP din fiecare pachet transmis prin cele trei legături. Se presupune că legătura *A-R1* poate suporta o lungime maximă de cadru de 1024 de octeți incluzând un antet de cadru de 14 octeți, legătura *R1-R2* poate suporta o lungime maximă de cadru de 512 de octeți incluzând un antet de cadru de 8 octeți și legătura *R2-B* poate suporta o lungime maximă de cadru de 512 octeți incluzând un antet de cadru de 12 octeți.

35. Un ruter distruge pachetele IP a căror lungime totală (date plus antet) este de 1024 octeți. Presupunând că pachetele trăiesc pentru 10 sec, care este viteza maximă a liniei la care poate opera ruterul fără a fi în pericol să cicleze prin spațiul numerelor de ID al datagramelor IP.

36. O datagramă IP care folosește opțiunea *Dirijare strictă de la sursă* trebuie să fie fragmentată. Credeți că opțiunea este copiată în fiecare fragment, sau este suficient să fie pusă numai în primul fragment? Explicați răspunsul.

37. Să presupunem că pentru o adresă de clasă B, partea care specifică rețeaua utilizează 20 de biți în loc de 16 biți. Câte rețele de clasă B se pot obține?

38. Transformați adresa IP a cărei reprezentare zecimală este C22F1582 într-o notație zecimală cu puncte.

39. O rețea din Internet are masca de subrețea 255.255.240.0. Care este numărul maxim de gazde din subrețea?

40. Un număr mare de adrese IP consecutive sunt disponibile începând cu 198.16.0.0. Să presupunem că patru organizații, *A*, *B*, *C*, *D*, cer câte 4000, 2000, 4000 și 8000 adrese, în această ordine. Precizați, pentru fiecare dintre ele, prima și ultima adresă IP atribuită, precum și masca în notația w.x.y.z/s.

41. Un ruter tocmai a primit următoarele noi adrese IP: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 și 57.6.129.0/21. Dacă toate folosesc aceeași linie de ieșire, pot fi ele compuse? Dacă da, ce va rezulta? Dacă nu, de ce nu ?

42. Setul de adrese IP de la 29.18.0.0 la 19.18.128.255 au fost reunite la 29.18.9.9/17. Totuși există un spațiu de 1024 de adrese nealocate, de la 29.18.60.0 la 29.18.63.255, care sunt brusc alocate

unei gazde care folosește altă linie de ieșire. Este nevoie acum ca adresa agregată să fie spartă în blocurile constituente, să se adauge blocurile noi la tabelă și să se vadă apoi dacă este posibilă o altă reunire? Dacă nu, ce se poate face ?

43. Un ruter are următoarele intrări (CIDR) în tabela sa de dirijare :

Adresă/mască	**Următorul salt**
135.46.56.0/22	Interfața 0
135.46.60.0/22	Interfața 1
192.53.40.0/23	Ruter 1
Implicit	Ruter 2

Pentru fiecare dintre următoarele adrese IP, ce face ruterul dacă primește un pachet cu respectiva adresă?

a) 135.46.63.10
b) 135.46.57.14
c) 135.46.52.2
d) 192.53.40.7
e) 192.53.56.7

44. Multe companii au politica de a avea două (sau mai multe) rutere conectate la Internet pentru a avea redundanță în caz că unul dintre ele nu mai funcționează. Mai este această politică posibilă cu NAT ? Explicați răspunsul.

45. Tocmai i-ați explicat unui prieten protocolul ARP. Când ați terminat, el spune: „Am înțeles. ARP oferă un serviciu nivelului rețea, deci face parte din nivelul legăturii de date.” Ce îi veți spune?

46. Atât ARP cât și RARP realizează corespondența adreselor dintr-un spațiu în altul. Din acest punct de vedere cele două protocoale sunt similare. Totuși, implementările lor sunt fundamental diferite. Care este diferența esențială dintre ele?

47. Descrieți un procedeu pentru reasamblarea fragmentelor IP la destinație.

48. Cei mai mulți algoritmi de reasamblare a datagramelor IP au un ceas pentru a evita ca un fragment pierdut să țină ocupate pentru totdeauna tampoanele de reasamblare. Să presupunem că o datagramă este împărțită în patru fragmente. Primele trei sosesc, dar ultimul este întârziat. În cele din urmă timpul expiră și cele trei fragmente sunt eliminate din memoria receptorului. Puțin mai târziu, sosește și ultimul fragment. Ce ar trebui făcut cu el?

49. Atât la IP cât și la ATM, suma de control acoperă numai antetul, nu și datele. De ce credeți că s-a ales această soluție?

50. O persoană care locuiește în Boston călătorește la Minneapolis, luându-și calculatorul portabil cu sine. Spre surprinderea sa, LAN-ul de la destinația din Minneapolis este un LAN IP fără fir, deci nu trebuie să se conecteze. Este oare necesar să se recurgă la întreaga poveste cu agenți locali și agenți străini pentru ca mesajele de poștă electronică și alte tipuri de trafic să-i parvină corect?

51. IPv6 folosește adrese de 16 octeți. Dacă la fiecare picosecundă este alocat câte un bloc de 1 milion de adrese, cât timp vor exista adrese disponibile?

52. Câmpul *Protocol* folosit în antetul IPv4 nu este prezent în antetul fix pentru IPv6. De ce?

53. Când se introduce protocolul IPv6, protocolul ARP trebuie să fie modificat? Dacă da, modificările sunt conceptuale sau tehnice?

54. Scrieți un program care să simuleze dirijarea prin inundare. Fiecare pachet ar conține un contor care este decrementat la fiecare salt. Când contorul ajunge la zero, pachetul este eliminat. Timpul este discret, iar fiecare linie manevrează un pachet într-un interval de timp. Realizați trei versiuni ale acestui program: toate liniile sunt inundate, sunt inundate toate liniile cu excepția liniei de intrare, sau sunt inundate numai cele mai bune k linii (alese statistic). Comparați inundarea cu dirijarea deterministă ($k = 1$) în termenii întârzierii și lărgimii de bandă folosite.

55. Scrieți un program care simulează o rețea de calculatoare ce folosește un timp discret. Primul pachet din coada de așteptare a fiecărui ruter face un salt per interval de timp. Fiecare ruter are numai un număr finit de zone tampon. Dacă un pachet sosește și nu este loc pentru el, el este eliminat și nu mai este retransmis. În schimb, există un protocol capăt-la-capăt, complet, cu limită de timp și pachete de confirmare, care va regenera în cele din urmă pachetul de la ruterul sursă. Reprezentați grafic productivitatea rețelei ca funcție de limita de timp, parametrizată de rata erorilor.

56. Scrieți o funcție pentru retransmiterea într-un ruter IP. Procedura are un parametru, o adresă IP. De asemenea are acces la o tabelă globală constând dintr-un vector de tripleți. Fiecare triplet conține trei întregi: o adresă IP, o mască de subrețea și linia de ieșire ce trebuie folosită. Funcția caută adresa IP în tabelă folosind CIDR și întoarce ca valoare linia ce trebuie folosită.

57. Folosiți programele *traceroute* (UNIX) sau *tracert* (Windows) pentru a urmări calea de la calculatorul personal până la diverse universități de pe alte continente. Creați o listă a legăturilor transoceanice pe care le-ați descoperit. Câteva sit-uri de încercat sunt: www.berkely.edu (California), www.u-tokyo.ac.jp (Tokyo), www.mit.edu (Massachusetts), www.vu.nl (Amsterdam), www.usyd.edu.au (Sydney), www.ucl.ac.uk (Londra), www.uct.ac.za (Cape Town).