Tema 4-1 Nivelul rețea

Lecția prezentă rolul şi funcțiile nivelului rețea în modelele arhitecturale OSI şi Internet. Sunt dezvoltate aspectele referitoare la rutarea pachetelor şi fluxurilor de date, sunt descrişi şi exemplificați mai mulți algoritmi şi protocoale de rutare împreună cu utilizările acestora.

După parcurgerea şi însuşirea acestei lecții, studentul va cunoaşte:

- ***Locul, rolul şi importanța nivelului rețea şi legătura cu nivelele adiacente***
- ***Definirea şi caracteristicile algoritmilor de rutare şi aplicabilitatea lor***
- ***Algoritmi de rutare pe bază de flux***
- ***Algoritmi de rutare bazați pe vectori distanță***
- ***Algoritmi de rutare bazați starea legăturilor***
- ***Dirijarea ierarhică şi dirijarea multiplă***
- ***Principalele protocoale de rutare folosite în Internet (RIP, OSPF, IGRP, BGP, EGP, IS-IS, etc.)***

Cuvinte cheie: ***rută, protocol de rutare, metrică de rutare, algoritm de rutare, algoritmul lui Dijkstra, algoritmul Bellman-Ford, tabel de rutare, stare legături, vectori distanță, difuzare, inundare, arbore de scufundare, RIP, OSPF, IGRP, BGP, EGP, IS-IS***

Bibliografie

1. **Iosif Praoveanu - Rețele de calculatoare, Ed. Universității Titu Maiorescu, Bucureşti 2009 - cap 4 Nivelul rețea**
2. **Andrew S. Tanenbaum - Rețele de calculatoare, ediția a 4-a, Editura Byblos, Bucureşti 2004 – cap. 5 Nivelul rețea**
3. **Tim Parker, Mark Sportack – TCP/IP, Editura Teora, Bucureşti, 2002**

Temă de casă*: Fiecare student va primi o temă de casă care va consta în descrierea şi aplicarea practică a unui algoritm de rutare.*

Lucrările de laborator exemplifică practic aplicarea acestor algoritmi pentru rutarea traficului în diverse rețele locale şi de arie largă. Cu ajutorul programelor de simulare RouteSim Network Visualizer şi PacketTracer se realizează diferite scenarii şi aplicații de rutare. Se vor folosi şi echipamente fizice de nivel rețea (rutere, switch-uri) pentru a pentru a proiecta şi realiza rețele fizice.

Timp de studiu 6 ore

Rolul nivelului rețea

Nivelul rețea trebuie să asigure transportul blocurilor de date (TPDU-urilor) de la un capăt la altul al rețelei de transport indiferent de tehnologia rețelei, de algoritmii de dirijare, de modul de rutare (pachete sau circuite), de lungimea rutei, de topologia rețelei etc.

Transferul se poate face prin unul sau mai multe salturi, spre deosebire de nivelul legătură de date, unde transferul se face doar între două noduri vecine. Rețeaua trebuie să poată face dirijarea pachetelor în noduri, adică să facă **rutarea**. Nodurile de rețea care fac dirijare se numesc **rutere**. Ele trebuie să fie echipamente "inteligente", capabile să ia decizii de rutare optime, să aleagă calea cea mai potrivită de urmat dintre multe variante posibile.

Rutarea este operația de transportare a informațiilor într-o rețea, sau între mai multe rețele, de la o sursă la o destinație.

Rutarea implică două activități de bază:

- **determinrea căilor optime de rutare** şi
- **transportarea fluxurilor de informații** (sau pachetelor) **prin rețea**.

Pentru aceasta, echipamentele de nivel rețea trebuie să folosească **trei categorii de protocoale**:

1. **protocoale de rutare** care descoperă topologia rețelei şi stabileşte rute între oricare perechi sursă destinație între care are loc transfer de date;
2. **protocoale rutabile** care sunt capabile să dirijeze pachetele de date dintr-un nod în altul pe baza rutelor stabilite de protocoalele de rutare;
3. **protocoale de control** care mențin controlul transferului de date şi informează entitățile corespondente despre evenimentele care pot apărea în rețea pe durata transferului.

Protocoalele de rutare folosesc diferite **metrici** pentru a evalua ce drum este optim pentru transportul unui pachet.

O metrică este o măsură standard, ca de exemplu lățimea de bandă a canalului de comunicație, distanța dintre sursă şi destinație etc.

Pentru determinarea drumului, ruterele folosesc protocoalele de rutare, prin care inițializează şi administrează **tabele de rutare**, în care se află informații despre rute.

Pentru aceasta, **ruterele trebuie să cunoască topologia rețelei**, să aibă mereu **informații despre starea rutelor**, să poată folosi diferite **criterii de performanță pentru a compara rutele**, să poată **utiliza algoritmi de rutare** în timp real.

4.1. Servicii asigurate nivelului transport

Serviciul de transport al datelor prin rețea poate să fie **neorientat pe conexiune** sau **orientat pe conexiune.** În primul caz se trimit pachetele folosind două primitive de bază, *send packet* şi *receive packet*, fără operații de verificare a ordinei sosirii pachetelor.

În cel de al doilea caz se stabileşte mai întâi o conexiune la nivel transport între sursă şi destinație, după care se pot negocia unii parametrii ai conexiunii (calitatea serviciilor, banda de transfer etc.) şi apoi are loc transferul datelor. Ambele tipuri de servicii sunt folosite în rețele. Rețelele IP au serviciile de nivel rețea neorientate pe conexiune, iar ATM are nivelul rețea orientat pe conexiune.

În cazul serviciului neorientat pe conexiune, nivelul transport al gazdei transmițătoare trimite blocul de date (TPDU) la primul ruter la care este conectat la intrarea în rețea. Acesta îi adaugă antetul de rețea care conține printre altele adresa de rețea pe baza căreia, folosind o tabelă de rutare, îl trimite pe un port de ieşire al ruterului, spre un alt nod al rețelei. Acesta primind pachetul îl memorează, analizează adresa de rețea şi folosind tabela de rutare proprie îl trimite printr-un port de ieşire spre următorul ruter. Pentru aceasta, fiecare nod memorează şi retransmite (*Store-and-forrward)* pachetele după o prelucrare specifică. Ruterele fiind şi capete ale nivelului legătură de date, pachetele suferă procesări specifice acestui nivel (existența erorilor, suma de control, etc.) înainte de a fi livrate nivelului rețea. Prin urmare, procesările din nodurile de rețea sunt destul de complexe şi necesită uneori timp apreciabil.

Sarcina principală a ruterului este de a dirija pachetele folosind **tabele de rutare.**

Un tabel de rutare al unui nod conține, sub formă de perechi, lista tuturor destinațiilor care pot fi atinse din nodul dat şi portul de ieşire corespunzător.

Tabelele de rutare se pot schimba în timp, în funcție de starea rețelei, de algoritmul de dirijare folosit etc.

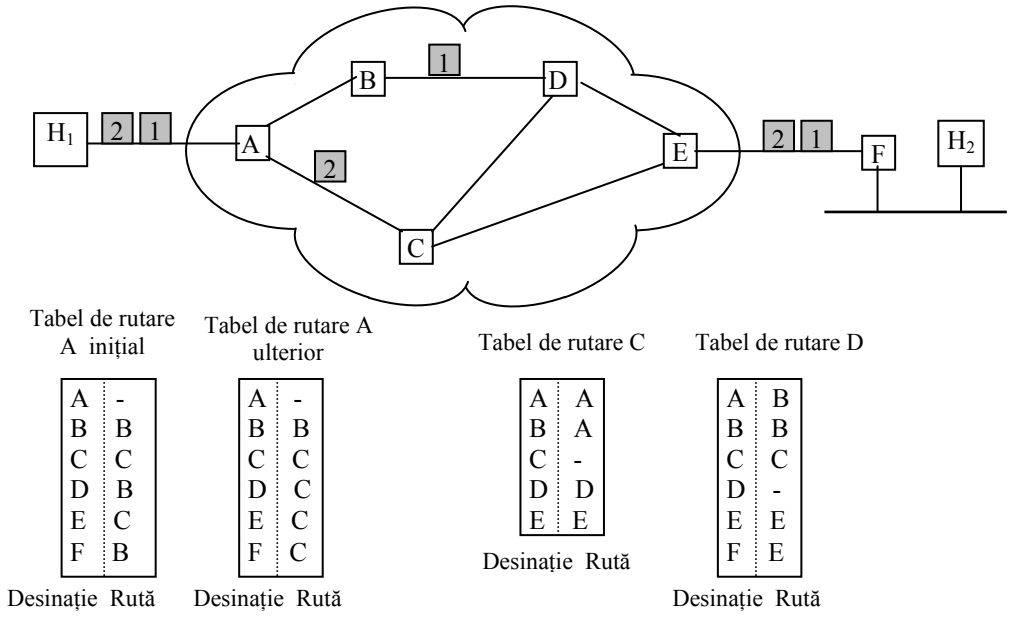


Fig. 4.1 Dirijarea pachetelor într-o rețea neorientată pe conexiune

Pentru a înțelege mecanismul rutării pe bază de tabele se poate analiza rețeaua din fig. 4.1. Gazda H_1 emite pachete pentru destinația H_2. Gazda H_1 este conectată direct la ruterul A din rețeaua de transport, iar gazda H_2 face parte dintr-un LAN care este conectat la rețeaua de transport prin ruterul E. Pachetele gazdei H_1 sunt livrate ruterului A care urmează să le livreze următorului nod din rețea. Pachetul 1 va fi dirijat de la A spre ieşirea B conform tabelului de rutare inițial care arată că destinația F se poate atinge din A prin ieşirea B. Ruterul B nu are decât o ieşire, spre D şi pachetul va fi livrat lui D. Ruterul D vede că pchetul 1 este destinat lui F, se uită în tabela de rutare şi vede că F poate fi atins din D pe ieşirea E. Prin urmare, îl trimite lui E şi aşa mai departe. Pentru al doilea pachet ruta prin rețea poate fi alta, de exemplu A,C,E,F. Aceast rută este calculată de algoritmul de rutare care, la momentul transmiterii pachetului 2, o găseşte ca fiind mai bună decât ruta anterioară. Algoritmul va schimba tabelele de rutare din noduri, precizând că ieşirea din A spre F este prin C. Calcularea rutelor în fiecare nod şi pentru fiecare pachet este un proces consumator de timp şi resurse: el necesită memorii tampon suficiente în noduri şi blochează procesorul ruterului pe timpul rulării algoritmului de rutare. Tehnica rutării independente a pachetelor are şi avantaje prin faptul că elimină fazele stabilirii şi desfacerii conexiunii şi poate asigura o încărcare uniformă a rutelor fără a le supraîncărca pe unele şi subîncărca pe altele.

4.2 Algoritmi de rutare

Pentru stabilirea rutelor urmate de pachete în rețea, fie că este vorba de comutare de pachete, fie că este vorba de stabilire de circuite virtuale, se folosesc algoritmi de rutare.

Ei sunt acea parte a softului de rețea care trebuie să aleagă o rută (cale) optimă între un nod sursă şi unul destinație. Uneori este util să se facă distincție între două procese care au loc în ruter:

1. **procesul de dirijare** care înseamnă stabilirea căii prin rețea când are loc completarea sau actualizarea tabelei de dirijare
2. **procesul de retransmitere** (forwarding) a pachetelor sosite în ruter.

Observație

Stabilirea unei rute optime prin rețea nu este un proces simplu. De multe ori trebuie să se aleagă sau să se facă un compromis între optimalitate şi încărcarea rutelor. Calea cea mai scurtă dintre două noduri de rețea poate fi şi prea încărcată, depăşind chiar capacitatea de transfer de date. Atunci trebuie aleasă o cale mai lungă, dar mai puțin încărcată.

Principiul optimalității spune 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 aceeaşi rută. O consecință directă a principiului optimalității este că mulțimea rutelor de la un nod sursă către toate nodurile destinație formează un **arbore de scufundare** (sink tree) cu rădăcina în sursă. Arborele de scufundare poate să nu fie unic. El nu conține bucle şi, ca urmare, un pachet poate fi livrat din sursă în oricare alt nod printr-un număr limitat de salturi. Scopul algoritmilor de rutare este de a descoperi şi folosi arbori de scufundare pentru toate nodurile.

Algoritmii de dirijare pot fi:

1. **Algoritmii neadaptivi (statici)** nu îşi bazează deciziile pe măsurători sau estimări ale traficului sau ale topologiei curente ale rețelei. Ei sunt algoritmi statici deoarece se aplică o singură dată la inițializarea rețelei şi nu țin seama de modificările apărute în timp în rețea..
2. **Algoritmii adaptivi (dinamici)** îşi modifică deciziile de rutare în funcție de schimbările care apar în rețea. Prin aceasta ei sunt mereu adaptați la starea curentă a rețelei.

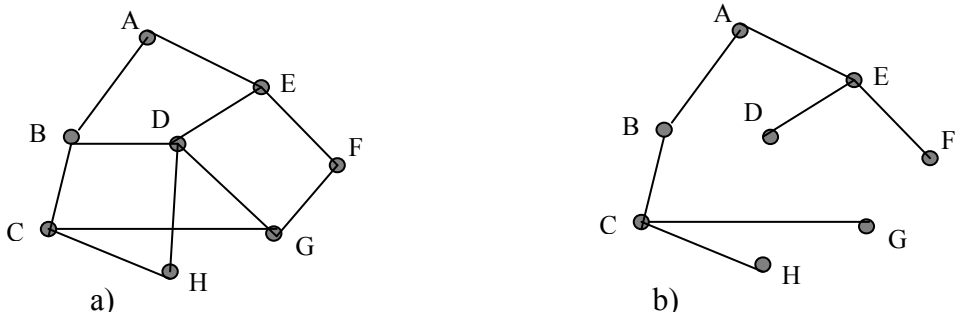


Fig. 4.3 a) O subrețea b) Un arbore de scufundare pentru nodul A

Inundarea (flooding) este acel algoritm care trimite un pachet sosit pe o intrare pe fiecare ieşire (mai puțin cea pe care a venit). În felul acesta, se produce o multiplicare exponențială a numărului de pachete şi se poate ajunge repede la blocarea rețelei dacă nu se iau măsuri speciale de prevenire a acestei situații.

Evitarea multiplicării infinite a pachetelor se poate face ataşând un contor în antetul fiecărui pachet inițializat cu o anumită valoare şi care se decrementează la fiecare multiplicare (salt prin rețea). Când contorul ajunge la zero, pachetul este eliminat.

4.2.1 Definiții şi caracteristici ale algoritmilor

Definiția 1:

Numim *algoritm* o prescripție care determină un anumit proces de calcul şi care este precisă, perfect inteligibilă şi nu admite nici un fel de interpretări din partea celui care o duce la îndeplinire. Înțelesul modern de algoritm este destul de apropiat de cel de rețetă, proces, metodă, procedură, rutină.

Definiția 2:

***Algoritmul* reprezintă un set de reguli care dau o secvență de operații pentru soluționarea unui tip specific de probleme.**

Caracteristicile unui algoritm:

- ***generalitate*** = algoritmul nu trebuie să rezolve numai o problemă, ci toate problemele din clasa respectivă;
- ***finititudine*** = numărul de transformări intermediare aplicate asupra informației inițiale pentru a obține informația finală este finit;
- ***unicitate*** = toate transformările intermediare aplicate asupra informației inițiale sunt unic determinate de regulile algoritmului; acesta trebuie să precizeze ordinea strictă a transformărilor; de asemenea, regulile precizează în ce caz se obține informația finală, după care activitatea algoritmului se întrerupe;
- ***claritate*** = fiecare pas al unui algoritm trebuie definit în mod precis; procedurile şi etapele de calcul trebuie specificate în mod riguros şi fără ambiguități;
- ***eficacitate*** = orice algoritm trebuie să ne conducă la rezultatul scontat în timp optim; toate operațiile ce urmează a fi executate în algoritm trebuie să fie suficient de fundamentate încât, în principiu, să poată fi făcute exact şi într-un interval finit de timp;
- *intrarea* = un algoritm are una sau mai multe intrări constituite din cantitățile inițiale care îi sunt date înainte ca algoritmul să înceapă; aceste intrări sunt luate dintr-un set specific de obiecte;
- ***ieşirea*** = un algoritm are una sau mai multe ieşiri, adică acele cantități ce sunt într-o relație specifică cu intrările.

Algoritmii de rutare utilizează diverse **metrici** pentru determinarea **rutei optime**. Algoritmi sofisticați de rutare pot să facă selecția rutelor pe baza mai multor metrici, combinându-le într-o singură metrică hibridă. Tipuri de metrici:

- **Lungimea drumului** (length path) este cea mai folosită metrică; de obicei este suma costurilor legăturilor drumului.
- **Siguranță** (reliability) - rata de erori, cât de repede se restabileşte o legătură căzută.
- **Întârziere** (delay) - cât durează să ajungă un pachet de la o sursă la o destinație. Depinde de lățimea de bandă, congestie, distanța fizică parcursă.
- **Lățimea de bandă** (bandwidth) - cât trafic poate să suporte o legătură.
- **Încărcare** (load) - se referă la gradul în care o resursă a rețelei este folosită, de exemplu un router.
- **Costul comunicației** (communication cost) - este important mai ales în companii care pot folosi liniile proprii (cost scăzut) față de folosirea altor linii (probabil un cost mai mare).

4.2.2 Algoritm de rutare pe bază de flux

În condițiile în care fluxurile de date dintr-o rețea sunt relativ constante şi cunoscute în timp şi spațiu, se poate calcula întârzierea medie din rețea pe baza teoriei cozilor. Minimizarea întârzierii medii a pachetelor în rețea poate fi un criteriu de optimizare pe baza căruia se iau deciziile de rutare. Pentru aplicarea algoritmului dirijării pe bază de flux trebuie cunoscute:

- topologia rețelei;

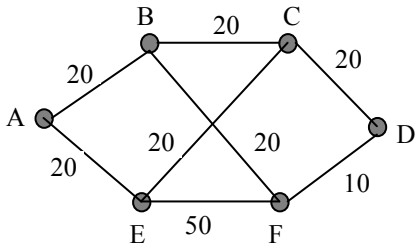
- matricea traficului (T_{ij})
- capacitățile liniilor (C_i)

Matricea traficului (fig 4.4) arată numărul mediu de pchete care circulă între sursa ***i*** şi destinația ***j*** pe ruta menționată în fiecare căsuță din matrice.

Ea poate fi simetrică ca în figură (simetrie față de diagonala principală) sau, într-un caz mai general, nesimetrică. De exemplu de la nodul sursă D pleacă spre nodul destinație B în medie 3 pachete pe secundă pe ruta DFB şi invers, de la D spre B tot 3 pchete pe secundă pe ruta BFD. Specificarea unei rute presupune aplicarea unui algoritm de dirijare.

	A	B	C	D	E	F
A	-	*9 AB*	*4 ABC*	*1 ABFD*	*7 AE*	*4 AEF*
B	*9 BA*	-	*8 BC*	*3 BFD*	*2 BFE*	*4 BF*
C	*4 CBA*	*8 CB*	-	*3 CD*	*3 CE*	*2 CEF*
D	*1 DFBA*	*3 DFB*	*3 DC*	-	*3 DCE*	*4 DF*
E	*7 EA*	*2 EFB*	*3 EC*	*3 ECD*	-	*5 EF*
F	*4 FEA*	*4 FB*	*2 FCE*	*4 FD*	*5 FE*	-

a)



b)

Fig. 4.4 Dirijarea pe bază de flux a) matricea traficului b) topologia rețelei

Având matricea traficului, se poate calcula încărcarea sau traficul λ_i pe fiecare link prin însumarea traficului dat de fiecare rută pe linkul respectiv. De exemplu, traficul pe linkul CD este de 6 pachete/s dat astfel: 3 unități trafic direct între C şi D şi 3 unități trafic între E şi D pe ruta ECD.

Traficul total în fiecare linie orientată de la stânga la dreapta este prezentat în Tabelul 4.2.

Tabelul 4.2

i	*linia*	*λ_i [pachete/s]*	*C_i [kbps]*	*μC_i[pachete/s]*	*T_i [ms]*	*P_i*
1	AB	14 *(AB+ABC+ABEF)*	20	25	91	0,171
2	BC	12 *(BC+ABC)*	20	25	77	0,146
3	CD	6 *(CD+ECD)*	10	12,5	154	0,073
4	AE	11 *(AE+AEF)*	20	25	71	0,134
5	EF	13 *(EF+AEF+EFB+CEF)*	50	62,5	20	0,159
6	FD	8 *(FD+ABFD+BFD)*	10	12,5	222	0,098
7	BF	10 *(ABFD+BFD+BFE+BF)*	20	25	67	0,122
8	EC	8 *(EC+ECD+FEC)*	20	25	59	0,098

$\Sigma\lambda_i$ =82

Fig. 4.4 Dirijarea pe bază de flux a) matricea traficului b) topologia rețelei

Considerând lungimea medie a unui pachet de 800 biți (μ=1/800), rata medie a pachetelor pe linie este μC_i [pachete/s]. Întârzierea medie pe fiecare link se calculează astfel:

$$T_i = \frac{1}{\mu C_i - \lambda_i} \text{ [ms]} \quad (4.1)$$

Ponderea unei linii în traficul total din rețea este:

$$P_i = \frac{\lambda_i}{\sum \lambda_i} \quad (4.2)$$

Întârzierea medie a pachetelor în rețea este suma ponderată a întârzierilor pe fiecare link:

$$\Delta T = \sum T_i P_i = 86ms \quad (4.3)$$

Luând în calcul o altă dirijare posibilă se obține o altă întâziere medie ΔT.

Calculând toate întârzierile posibile corespunzătoare diferitelor rutări posibile se va lua în considerare întărzierea minimă care va da şi schema de rutare optimă din punct de vedere al întârzierii minime.

Acest algoritm are două dezavantaje majore:

- necesită calculul întârzierilor pe toate rutele posibile, ceea ce implică un volum mare de calcule
- este un algoritm static, care nu ține seama de schimbările posibile din rețea.

4.2.3 Dirijarea cu vectori distanță

Rețele moderne de calculatoare folosesc algoritmi dinamici de dirijare. Cei mai cunoscuți sunt cei bazați pe **vectori distanță** şi cei bazați pe **starea legăturilor (LSA-Link State Algorithm)**.

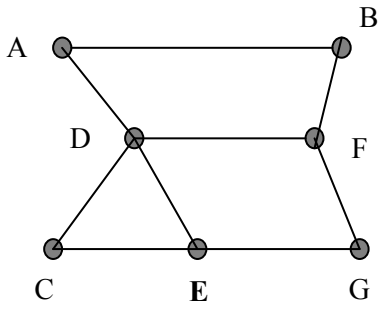
Algoritmul bazat pe vectori distanță presupune că fiecare ruter menține o tabelă (un vector) care păstrează cea mai bună distanță cunoscută spre fiecare destinație şi linia (ieşirea) care trebuie urmată pentru a ajunge acolo.

Aceste tabele sunt actualizate prin schimbul de informații între ruterele vecine. Algoritmul mai este cunoscut şi sub numele de **algoritmul lui Ford – Fulkerson** sau **algoritmul lui Bellman – Ford**. Fiecare înregistrare din tabelă este o pereche de date:

- ieşirea preferată spre destinația specificată
- estimarea timpului (distanței) de ajungere la acea destinație.

Ca metrică se pot folosi numărul de salturi, întârzierea, lungimea cozii etc. Pentru metrica salturilor distanța până la vecini este doar de un salt. Pentru metrica lungimea cozilor, ruterul examinează lungimea cozii pachetelor aflate în aşteptare pentru servire în ruterul vecin etc. Să examinăm cazul în care algoritmuil foloseşte metrica întârzierilor Atunci ruterul cunoaşte întârzierea spre fiecare din vecinii săi. De asemenea el recepționează o listă similară de la vecini. Dacă un ruter A ştie că până la vecinul B distanța este m iar vecinul i-a trimis estimarea sa până la ruterul X_i ca fiind t_i, atunci ruterul A stie că poate ajunge la X_i în $m+t_i$ secunde ieşind prin B. Făcând aceste calcule pentru fiecare vecin, ruterul A poate afla cea mai scurtă cale spre ruterul X_i.

Modul de aplicare al algoritmului se poate explica pe fig. 4.5 Se cunoaşte topologia rețelei şi nodul E are tabela de estimare a distanțelor nodurilor vecine față de oricare din celelalte noduri din rețea. La anumite intervale de timp nodurile recalculează distanțele față de nodurile vecine. De exemplu **nodul E** recalculează întârzierile față de nodurile vecine şi găseşte valorile: $t_{EC}=8$, $t_{ED}=6$, $t_{EG}=15$ diferite de valorile anterioare de 10, 7 şi respectiv 13.



	C	D	G
A	10	8	20
B	14	15	12
C	-	8	16
D	8	-	12
E	10	7	13
F	12	10	8
G	11	13	-

Dest	t_i	linia
A	14	D
B	21	D
C	8	C
D	6	D
E	-	-
F	16	D
G	15	G

Fig. 4.5 Model pentru algoritm bazat pe vectori distanță

Cu aceste noi valori şi cu tabela distanțelor vecinilor față de noduri (tabela b)), el îşi actualizează tabela proprie a distanțelor față de nodurile din rețea (tabela c)), pe care o comunică ulterior nodurilor vecine. Tabela c) este este rezultatul unor calcule succesive. Astfel, de la nodul E se poate ajunge în A prin C, D sau G. Pentru fiecare variantă se calculează întârzierile:

$$\begin{aligned} t_{EA} &= t_{EC} + t_{CA} = 8 + 10 = 18 \\ t_{EA} &= t_{ED} + t_{DA} = 6 + 8 = 14 \\ t_{EA} &= t_{EG} + t_{GA} = 15 + 20 = 35 \end{aligned} \qquad (4.4)$$

În noul tabel de rutare al lui E pentru destinația A se va trece valoarea minimă 14, cu linia de ieşire D. Calcule asemănătoare se fac pentru toate destinațiile care pot fi atinse din E.

Algoritmul rutării pe bază de vectori distanță are o limitare serioasă: deşi converge spre rezultatul corect, o face foarte lent. Lungimea mare a tabelelor de rutare, ca şi numărul mare de calcule cerute, face ca acest algoritm să fie aplicabil în timp real doar pentru rețele mici şi care nu reclamă actualizări dese ale tabelelor de rutare.

4.2.4 Algoritmul de rutare folosind starea legăturilor

Un algoritm adaptiv des utilizat în rețelele actuale este cel bazat pe **starea legăturilor**. Aplicarea acestuia presupune ca ruterele să facă următoarele acțiuni:

1. ***Să descopere vecinii săi din rețea şi să afle adresele de rețea ale acestora;***
2. ***Să măsoare costurile până la fiecare din vecinii săi;***
3. ***Să anunțe toți vecinii că s-a instalat în rețea şi are date despre ei;***
4. ***Să trimită pachete de înştiințare către toate ruterele din rețea;***
5. ***Să calculeze cea mai scurtă cale spre fiecare ruter***

Determinarea vecinilor

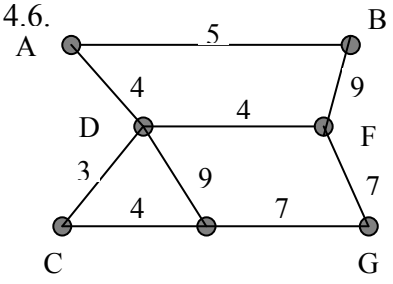
Când un ruter se instalează în rețea primul lucru pe care trebuie să-l facă este să afle care sunt vecinii săi. Pentru aceasta el trimite un pachet special de salut **HELLO** pe fiecare linie pe care este conectat la un alt ruter. Ruterul apelat trebuie să-i răspundă anunțându-şi identitatea sa.

Măsurarea costului liniei

Pentru a afla costul, distanța sau întârzierea ruterului față de vecinii săi, acesta trimite un pachet de sondare de tip **ECHO** pe linie, cerând ruterului să trimită imediat răspunsul înapoi. Marcarea timpului dus-întors constituie o bună estimare a întârzierii pe linia respectivă. Pentru o estimare mai bună se repetă operația de sondare şi se face o medie aritmetică a rezultatelor obținute. O problemă care trebuie lută în consideare este dacă în acest timp trebuie inclusă şi aşteptarea în coadă în cazul că ruterul apelat nu poate răspunde imediat.

Construirea pachetelor cu starea legăturii

După ce ruterul s-a conectat în rețea, a anunțat vecinii şi a calculat distanțele față de ei, urmează să construiască un pachet care să conțină starea legăturilor. Un exemplu este prezentat în fig. 4.6.



a) topologia rețelei

A	
Secv.	
Vârstă	
B	5
D	4

B	
Secv.	
Vârstă	
A	5
F	9

C	
Secv.	
Vârstă	
D	3
E	4

D	
Secv.	
Vârstă	
A	4
C	3
E	9
F	7

E	
Secv.	
Vârstă	
C	4
D	9
G	7

F	
Secv.	
Vârstă	
B	9
D	7
G	7

G	
Secv.	
Vârstă	
E	7
F	7

b) pachete cu starea legăturilor

Fig. 4.6 Algoritm de rutare bazat pe starea legăturilor

Distribuirea pachetelor cu starea legăturilor

Partea cea mai delicată a algoritmului este distribuirea sigură a pachetelor cu starea legăturilor. De îndată ce pachetele sunt distribuite şi recepționate de rutere, acestea încep să-şi calculeze rutele. Pentru ca pachetele expediate să ajungă la toate ruterele se foloseşte algoritmul inundării. Pentru a controla mecanismul inundării, fiecare pachet creat are un număr de secvență care este incrementat la fiecare nou pachet transmis. Ruterele păstrează evidența tuturor perechilor *ruter_sursă, număr_secvență* pe care le-a văzut deja. La sosirea unui nou pachet cu starea legăturilor, el este căutat în lista cu pachete deja văzute. Dacă pachetul este nou, el este trimis pe toate ieşirile, mai puțin pe portul pe care a venit. Dacă este duplicat, el este şters. Dacă pachetul sosit are număr de secvență mai mic decât cel mai mare număr de secvență detectat, el este rejectat ca fiind învechit.

Analiza atentă a acestui algoritm scoate în evidență unele probleme. De exemplu, dacă un ruter se defectează, el va pierde evidența numerelor de secvență. Dacă va începe de la 0, următorul pachet va fi rejectat ca duplicat. Soluția acestei probleme este includerea vârstei pachetului după numărul de secvență şi decrementarea lui la fiecare secundă. Când vârsta ajunge la 0 pachetul este distrus.

Calcularea noilor rute

După ce un ruter a recepționat un set complet de pachete cu starea legăturilor, el poate construi graful întregii subrețele, deoarece fiecare legătură este reprezentată. Se poate folosi algoritmul lui Dijkstra pentru a găsi calea cea mai scurtă către toate destinațiile posibile. Rezultatele acestui algoritm sunt trecute în tabelul de dirijare care vor conține distanța până la fiecare destinație şi primul ruter prin care se poate atinge acea destinație. Memoria necesară stocării tabelelor de ruare pentru a rețea cu *n* noduri şi *m* vecini este proporțională cu produsul $n \times m$. Pentru rețele foarte mari dimensiunea memoriei ca şi timpul de calcul al rutelor pot constitui probleme serioase. Totuşi dirijarea pe baza stării legăturilor este larg folosită în rețelele actuale. **OSPF** (**Open Shortest Path First**) şi **IS-IS** (**Intermediate System to Intermediate System**) sunt doar doi algoritmi bazați pe starea legărturilor. IS-IS este folosit în coloanele vertebrale ale Internetului (Internet backbone). El a fost creat de DECNET şi apoi adoptat de ISO pentru a fi folosit cu protocolul de nivel rețea neorientat pe conexiune, CLNP.

4.2.5 Dirijarea ierarhică (Suplimentar)

*Pe măsură ce rețelele cresc în dimensiune, tabelele de rutare devin tot mai mari, memoria necesară stocării tabelelor creşte, iar timpul de calcul devine prohibitiv. O soluție pentru asemenea rețele mari este **dirijarea ierarhică**. În acest caz, rețeaua este împărțită pe **regiuni** (sau **zone**), un ruter trebuind să aibă informații de rutare doar pentru rețeaua din regiunea sa. Recunoaşterea regiunilor trebuie să se poată face pe bază de adresă. În câmpul de adresă trebuie să existe spațiu rezervat pentru regiune şi spațiu rezervat pentru ruterele locale. Pentru expedierea pachetelor pe regiuni, ruterul examinează mai întâi spațiul de adresă de regiune pentru a vedea unde expediază pachetul. Dacă adresa de regiune indică regiunea în care este ruterul, acesta va examina doar adresa locală pentru a identifica ruterul destinație. Rutarea ierarhică se poate face pe mai multe nivele. Reducerea tabelelor de dirijare în rețelele ierarhice este considerabilă. De exemplu, dacă o rețea are 720 de rutere, în cazul că nu este organizată ierarhic, tabelul de rutare va avea 720 de înregistrări. Dacă este organizată ierarhic pe două nivele cu 24 de regiuni şi 30 de rutere în fiecare, atunci tabela de rutare a unui ruter dat va avea 23 de înregistrări pentru celealte regiuni şi 30 de înregistrări pentru ruterele locale din regiunea sa, deci în total 23 plus 30, adică 53 de înregistrări. Dacă se alege o organizare ierarhică pe 3 nivele, cu 8 zone, fiecare zonă cu 9 regiuni şi fiecare regiune cu 10 rutere, atunci tabela de rutare va avea doar 10 înregistrări pentru ruterele locale, 8 pentru regiuni şi 7 pentru zone, în total 25 de înregistrări. Prețul plătit pentru dirijarea ierarhică este lungimea mai mare a rutelor deoarece ele sunt forțate să treacă rutere de regiune sau de zonă, chiar dacă o cale directă ar fi mai scurtă.*

4.2.6 Dirijarea prin difuzare

*In unele rețele de calculatoare este nevoie ca un mesaj să fie difuzat simultan mai multor calculatoare sau tuturor dintr-o rețea. Trimiterea simultană a unui pachet către toate destinațiile se numeşte **difuzare (broadcast)**. Realizarea difuzării se poate face în diferite moduri. O primă modalitate este de a trimite câte un pachet la fiecare destinație. Metoda este mare consumatoare de bandă şi cere ca sursa să aibă lista completă a tuturor destinațiilor.*

*O altă variantă este **inundarea**. Neajunsul metodei constă în multiplicarea peste nevoi a pachetelor, ceea ce atrage după sine şi consum mare de bandă.*

*O variantă mai bună este **dirijarea multidestinație**. Prin 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, acesta verifică toate destinațiile la care trebuie trimis pentru a determina setul de ieşiri pe care trebuie dirijat. Apoi ruterul generează o copie a pachetului pentru fiecare linie de ieşire şi include în fiecare pachet doar acele destinații care folosesc linia respectivă. Efectul este partiționarea mulțimii destinațiilor între liniile de ieşire. După un număr suficient de salturi, pachetul ajunge să aibă o singură destinație ca un pachet normal. Altfel spus, dacă pe o linie se trimit pachete spre mai multe destinații, pe porțiunea comună de linie se trimite doar un singur pachet şi nu câte unul pentru fiecare destinație. Un singur pachet le substituie pe toate celelalte pe porțiunea comună şi multiplicarea se produce doar atunci când linia nu mai este comună pentru mai multe destinații.*

*Un al patrulea mod de difuzare este cel care foloseşte **arborele de acoperire** (spanning tree). Un arbore de acoperire este un subgraf al rețelei care conține toate ruterele (şi evident, nu are bucle). Dacă un rutere cunoaşte care din liniile sale de ieşire participă la arborele de acoperire, el poate copia şi trimite un pachet de difuzare recepționat doar pe liniile de ieşire care fac parte din arborele de acoperire. Este metoda cea mai eficientă din punct de vedere al consumului de bandă. Dificultatea este în găsirea arborelui de acoperire pentru nodurile spre care trebuie să se facă difuzarea.*

4.3 Alte Protocoale de rutare

Cele mai cunoscute protocoale de rutare sunt:

- Routing Information Protocol (RIP)
- Interior Gateway Routing Protocol (IGRP)
- Enhanced Interior Gateway Routing Protocol (Enhanced IGRP)
- Open Shortest Path First (OSPF)
- Intermediate System to Intermediate System (IS-IS)
- Border Gateway Protocol (BGP)
- Exterior Gateway Protocol (EGP)
- Simple Multicast Routing Protocol (SMRP)
- Novell RIP / Service Advertisement Protocol (SAP)

4.3.1 Routing Information Protocol (RIP)

Protocolul RIP este unul dintre cele mai vechi şi mai durabile protocoale. Sunt o mare varietate de protocoale asemănătoare sau bazate pe RIP. Protocolul foloseşte **vectori de distanță** pentru a calcula rutele şi a alege ruta optimă. Algoritmii folosiți în implementare au fost descoperiți prin cercetare academică ce datează din 1957.

Standardul versiunii inițiale (**RIPv1**) este definit în două documente: RFC 1058 (în 1988) şi Internet Standard (STD) 56. Ulterior a fost definită versiunea **RIP 2** (RFC 1388 în 1993, iar în 1994 a publicat RFC 1723) care suportă şi **măşti de rețea de lungime variabilă**, o caracteristică foarte importantă pe care RIP nu o suporta. RIP 2 a mărit şi cantitatea de informații transportată de mesajele RIP, şi a permis folosirea unei metode simple de autentificare pentru securizarea operației de *update* a tabelelor de rutare.

Mesajele *update*

RIP trimite mesaje update la intervale regulate şi în momentul în care topologia rețelei se schimbă. Când un ruter primeşte un mesaj update care include schimbări pentru o rută, îşi schimbă propria tabelă de rutare pentru a reflecta schimbările din rețea. Metrica pentru lungimea acelei rute este incrementată cu 1 şi expeditorul mesajului update devine noul salt în acea rută. Routerele RIP mențin numai ruta cea mai bună (cu metrica lungimii minimă) către o destinație. Apoi ruterul trimite mesaje update pentru a informa alte rutere din rețea de schimbări. Aceste mesaje se trimit independent de mesajele regulate trimise la un interval de timp.

Metrica RIP

RIP foloseşte o singură metrică (numărul de salturi) pentru a măsura distanța dintre sursă şi destinație. Fiecare hop are asociată o valoare (în mod uzual 1). Când un ruter primeşte un update adaugă 1 la metricile destinațiilor cu rute schimbate. Adresa IP a expeditorului este folosită ca următorul hop pentru acele rute.

Stabilitatea RIP

RIP previne buclele de rutare (routing loops) prin implementarea unei limite a numărului maxim de hopuri permise într-o rută. Acest maxim este 15. Dacă un ruter primeşte un mesaj update care conține o rută cu o metrică 15, destinația este considerată neaccesibilă (*unreachable*). Deficiența acestei trăsături este limita maximă a diametrului unei rețele RIP. RIP mai implementează mecanisme de stabilitate care sunt comnune multor protocoale de routare: despicarea orizontului (*split horizon*) şi menținerea (*holddown*). Acestea asigură stabilitate chiar dacă sunt posibile schimbări rapide în topologia rețelei.

RIP *Timers*

RIP foloseşte timere (cronometre) pentru a-şi regla parametrii. Acestea includ *routing-update timer*, *route-timeout timer*, *route-flush timer*. Routing-update timer măsoară intervalul de timp scurs între update-uri. De obicei este 30 secunde. Fiecare intrare dintr-o tabelă de rutare are un route-timeout timer asociat. Când acesta expiră, ruta este marcată ca invalidă, dar rămâne în tabelă până când expiră şi route-flush timer.

Formatul pachetelor RIP 1

1-byte comandă	1-byte versiune	2-bytes zero	2-bytes AFI	2-bytes zero	4-bytes adresă IP	4-bytes zero	4-bytes zero	4-bytes metrică

Pachetul IP RIP poate avea mai multe intrări, care vor conține câmpurile din tabelul de mai sus.

comandă - indică dacă pachetul este o cerere (*request*) sau un răspuns (*response*), request se trimite pentru a primi înapoi de la router un update, iar response se primeşte nesolicitat regulat sau ca răspuns la un request;

versiune - versiunea RIP;

zero - nu este folosit, are valoarea 0;

Address-family identifier (AFI) - specifică familia de protocoale folosită, pentru IP valoarea este 2;

adresă IP - adresa IP pentru intrare;

metrică - numărul de hopuri sau infinit pentru unreachable (adică valoarea 16);

Formatul pachetelor RIP 2

1-byte comandă	1-byte versiune	2-bytes nefolosit	2-bytes AFI	2-bytes etichetă rută	4-bytes adresă IP	4-bytes mască rețea	4-bytes următorul hop	4-bytes metrică

comandă - indică dacă pachetul este o cerere (*request*) sau un răspuns (*response*), request se trimite pentru a primi înapoi de la router un update, iar response se primeşte nesolicitat regulat sau ca răspuns la un request;

versiune - versiunea RIP;

nefolosit - are valoarea 0;

Address-family identifier (AFI) - specifică familia de protocoale folosită cu o singură excepție: Dacă valoarea câmpului AFI pentru prima intrare din pachetul IP RIP este 0xFFFF, restul intrării conține informații de autentificare (adică o parolă);

etichetă rută - furnizează o metodă de a distinge între rute interne şi externe;

adresă IP - adresa IP pentru intrare;

mască rețea - conține masca de rețea pentru intrare sau 0 dacă nu a fost specificată o mască;

metrică - numărul de hopuri sau infinit pentru unreachable (adică valoarea 16);

RIPv2

RIP v2 este o variantă perfecționată a RIP v1 caracterizată de următoarele:

- este un protocol bazat pe vectori distanță folosind ca metrică numărul de salturi;
- foloseşte un timer pentru a evita transmiterea în buclă a pachetelor. Valoarea implicită este 180 secunde;
- foloseşte despicarea orizontului pentru evitarea rutării în buclă;
- numărul maxim (infinit) de salturi este 16.

RIP v2 conține un prefix de rutare care permite transmiterea (evidențierea) unei măşti de subrețea odată cu actualizarea rutelor. Ca urmare, RIP v2 suportă rutarea fără clase în care diferitele subrețele din aceeaşi rețea pot folosi măşti diferite, ca în VLSM (Variable Length Subnet Mask).

RIP v2 furnizează autentificare în timpul actualizării tabelelor. Pentru aceasta, se poate folosi un set de chei de autentificare pe interfață. Cheile se pot alege şi pot fi de tipul text clar (modul implicit) ori criptate cu Message-Digest 5 (MD5). MD5 poate fi folosit pentru a autentifica o sursă care transmite actualizarea tabelei de rutare.

Comparație RIP v1 cu RIP v2

RIP v1	RIP v2
Suportă numai protocoale cu clasă de rutare Nu dă informații despre subrețea în procesul actualizării Nu suportă rutare cu prefix. Toate gazdele dintr-o rețea trebuie să aibă acceaşi mască Nu asigură autentificarea în procesul de actualizare Face difuzarea pe 255.255.255.2555	Suportă protocoale fără clasă Dă informații despre subrețea în procesul actualizării Suoportă rutarea cu prefix. Subrețelele dintr-o rețea pot avea măşti diferite (VLSM) Permite autentificarea Actualizarea rutării se face prin adresă multicast clasă D, 224.0.0.9, ceea ce este mai eficient

4.3.2 Configurarea RIP v2

RIP v2 este un protocol de rutare dinamic, care se configurează tastând *RIP Version 2* şi asignând adresele IP de rețea, **fără a specifica valoarea măştilor de subrețea**. Figura următoare ilustrează configurarea unui ruter care interconectează trei rețele.

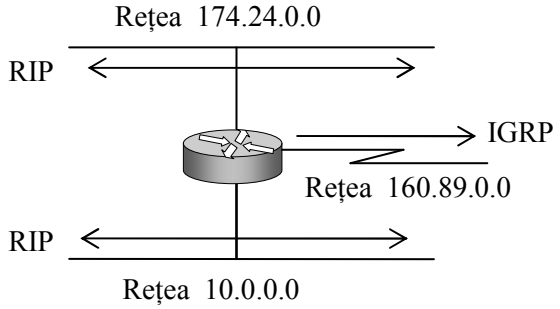


Fig.4.8 Reţele interconectate şi protocoalele de rutare folosite

Comanda ***router*** startează procesul de rutare.
Comanda ***network*** determină implementarea următoarelor funcţii:

- actualizările de rutare/difuzate pe interfeţele de ieşire
- procesarea actualizărilor intrate pe interfaţă
- subreţeaua care este direct conectată la interfaţă este anunţată.

Exemplu de configurare rutere

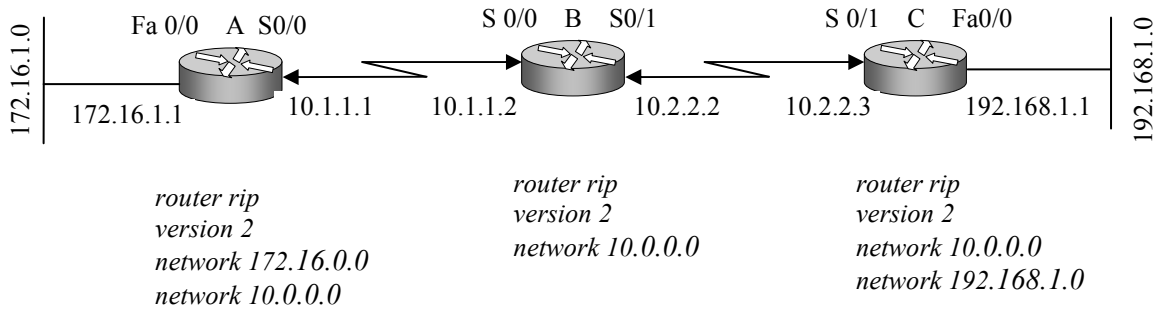


Fig. 4.9 Comenzi pentru configurarea ruterelor din reţea

4.3.3 Interior Gateway Routing Protocol (IGRP)

IGRP este un protocol de rutare dezvolatat la mijlocul anilor 1980 de Cisco Systems. Scopul principal a fost crearea unui protocol robust pentru rutarea în sisteme autonome. Protocoalele de acest fel se numesc **Interior Gateway Routing Protocols**. Înainte de apariţia IGRP, cel mai utilizat protocol era RIP. Deşi RIP era eficient pentru rutarea în reţele mici, relativ omogene, limitele sale deveneau evidente odată cu mărirea dimensiunii reţelelor. În particular, limita maximă de 16 salturi ale RIP-ului restricţiona mărimea reţelelor; metrica unică (numărarea salturilor) nu asigura destulă flexibilitate în medii complexe.

IGRP foloseşte **vectorii distanţă**. Ruterele trimit la intervale regulate mesaje update la toţi vecinii lor. Mesajele circulând prin toată reţeaua din vecin în vecin, se vor descoperi noi destinaţii

adăugate rețelei sau se vor identifica destinațiile ce nu pot fi atinse şi se vor calcula vectorii distanță către toate destinațiile cunoscute.

Metrica IGRP

IGRP foloseşte o metrică compozită calculată prin luarea în considerare a valorilor metricilor întârziere, lărgime de bandă, fiabilitate şi încărcare. Administratorii de rețea pot modifica valorile pentru fiecare metrică. Toate aceste metrici sunt ponderate de o serie de constante definite de administrator care pot influența alegerea unei rute. Acestea permit administratorului să regleze selecția automată a rutelor în IGRP.

Stabilitatea IGRP

IGRP are mecanisme folosite pentru a-i mări stabilitatea. Printre acestea sunt *menținerea rutelor căzute* (*holddowns)*, *despicarea orizontului* şi *actualizarea inversă*.

Holddowns sunt folosite pentru a preveni mesajele update regulate să reinstaleze o rută care a căzut. Când un ruter pică, ruterele vecine detectează acest lucru prin lipsa mesajelor update regulate. Aceste rutere calculează noile rute şi îşi informează vecinii despre acest lucru. Toate aceste mesaje update se răspândesc în rețea, dar durează până ajung la toate ruterele. Deci este posibil ca un ruter care nu a fost încă informat de noile schimbări, să trimită mesaje update în care să specifice tocmai ruterul căzut. Holddowns specifică ruterelor să mențină orice schimbare care ar putea afecta rutele o perioadă de timp. Perioada holddown este calculată, de obicei, să fie mai mare decât perioada de timp necesară propagării mesajelor update în toată rețeaua.

Despicarea orizontului porneşte de la premisa că nu este niciodată necesar să trimiți informații despre o rută înapoi în direcția din care a venit. Split horizon previne apariția buclelor de rutare. Split horizons ar trebui să prevină buclele de rutare între routere adiacente, dar actualizări inverse sunt necesare pentru înlăturarea unor bucle de rutare mai mari. Măririle valorilor metricilor de rutare, de obicei, indică existența buclelor. Atunci se trimit actualizări inverse pentru a îndepărta ruta şi a o plasa în holddown.

Timere

IGRP foloseşte **timp de actualizare** (update-timer), **timp de invalidare** (invalid-timer), **interval de menținere** (hold-time period) şi **timp de eliminare** (flush timer). Update timer specifică cât de des se trimit mesajele update (de obicei are valoarea 90 secunde). Invalid timer conține cât se aşteaptă, în absența mesajelor update până să declare ruta invalidă (aproximativ de trei ori update-timer). Variabila hold-time specifică perioada menținere (valoarea standard este de trei ori update-timer plus 10 secunde). Flush timer indică cât timp să treacă până când o rută ar trebui să fie eliminată din tabela de rutare (standard este de 7 ori update-timer).

IGRP are funcționalitatea similară cu RIP. Are aceleaşi calități ca RIP şi în plus nu are limită maximul de 16 hopuri, are metrica mai complexă, se poate folosi în rețele mai mari decât RIP. IGRP nu are suport pentru măşti de rețea de lungime variabilă.

4.3.4 Enhanced Interior Gateway Routing Protocol (EIGRP) Suplimentar

Protocolul Enhanced IGRP *reprezintă o evoluție față de predecesorul său IGRP. Apariția lui a fost necesară datorită evoluției şi heterogenității rețelelor. EIGRP integrează atributele protocoalelor link-state în protocoalele distance-vector. În plus, EIGRP înglobează încă câteva protocoale importante, care măresc considerabil eficiența sa operațională. Unul dintre aceste protocoale este* **Diffusing update algorithm (DUAL),** *dezvoltat de către Dr. J.J. Garcia-Luna-Aceves. DUAL oferă posibilitatea ca un ruter EIGRP să poată să determine dacă o rută primită de la un vecin este în circuit (*looped*) sau nu (*loop-free*) şi permite ruterului să găsească rute alternative fără a aştepta mesaje update de la alte rutere.*

Pachete EIGRP

EIGRP foloseşte pachete hello, acknowledgment, update, query şi reply.

Pachetele ***hello*** *sunt pachete multicast pentru neighbor discovery/recovery şi nu necesită confirmare. Un pachet* ***acknowledgment*** *este un pachet hello care nu conține date. Pachetele care nu necesită confirmare conțin un număr nenul şi sunt trimise folosind o adresă unicast.*

Pachetele ***update*** *sunt folosite pentru a publica accesibilitatea destinațiilor. Când un nou vecin este descoperit, se trimit pachete unicast către acel vecin pentru a putea să-şi construiască o tabelă topologică. În alte cazuri (de exemplu schimbarea costului unei legături), mesajele update sunt transmise multicast. Pachetele update sunt trimise totdeauna sigur (cu confirmare).*

Pachetele ***query*** *şi* ***reply*** *sunt trimise când o destinație nu are feasible successors. Pachetele query sunt totdeauna multicast. Pachetele reply sunt trimise ca răspuns la pachetele query pentru a-l înştiința pe cel care a expediat pachetul query să nu recalculeze ruta pentru că există succesori realizabili. Pachetele reply sunt unicast. Atât pachetele query, cât şi reply sunt transmise sigur.*

Protocolul EIGRP dezvoltat de Cisco este robust, îmbină atributele protocoalelor vector distanță cu atributele protocoalelor bazate pe starea liniei, rezultând un protocol hibrid, este uşor de configurat, eficient, sigur (RTP), rapid convergent.

4.3.5 Open Shortest Path First (OSPF)

OSPF este un protocol de rutare dezvoltat pentru rețele Internet Protocol (IP) de grupul *Interior Gateway Protocol (IGP) working group*, care face parte din Internet Engineering Task Force (IETF). Grupul s-a format în 1988 pentru a dezvolta un IGP folosit în Internet bazat pe algoritmul Shortest Path First (SPF). OSPF a fost creat din aceleaşi motive ca şi IGRP, în anii 1980, când protocolul RIP începea să fie incapabil să servească rețele din ce în ce mai largi şi mai eterogene.

OSPF are două caracteristici primare. Protocolul este deschis, adică specificațiile sunt în domeniul public. Specificațiile OSPF sunt publicate în Request For Comments (RFC) 1247. A doua caracteristică este că OSPF este bazat pe algoritmul SPF (algoritmul lui Dijkstra).

OSPF este un protocol de rutare bazat pe starea legăturilor, deci publică starea rutelor către toate ruterele din aceiaşi arie ierarhică. După ce au acumulat informaţii despre aceste stări, ruterele OSPF folosesc algoritmul lui Dijkstra pentru a calcula cea mai scurtă cale către fiecare nod.

Ierarhia de rutare

Spre deosebire de RIP, OSPF poate opera într-o ierarhie. Cea mai mare entitate în ierarhie este **sistemul autonom**, care este *o colecţie de reţele sub o administrare cu aceleaşi reguli peste tot*. OSPF este un protocol de rutare intradomeniu, dar este capabil să primească şi să trimită rute la alt sistem autonom.

Un sistem autonom poate fi divizat într-un număr de arii, care sunt grupuri de reţele contigue. Routere cu interfeţe multiple pot participa în arii multiple. Aceste routere se numesc *Area Border Routers*, ele menţin date separate despre topologia fiecărei arii. Topologia unei arii este invizibilă entităţilor din afara ariei. Ţinând topologiile separate, OSPF are un trafic mai mic decât dacă sistemul autonom nu ar fi partiţionat.

Partiţionarea în arii generează două tipuri diferite de rutare, după cum sursa şi destinaţia sunt în aceiaşi arie sau sunt în arii diferite. Routarea intra-arie se face în aceiaşi arie, iar rutarea inter-arie se face în arii diferite. Routerele responsabile de rutarea informaţiilor între arii formează o coloană vertebrală a sistemului autonom OSPF (*OSPF backbone*). OSPF backbone este formată din toate Area Border Routers, reţele care nu sunt conţinute în totalitate în nici o arie, şi rutele ataşate lor.

Backbone este ea însăşi o arie OSPF, deci toate ruterele backbone folosesc aceleaşi proceduri şi algoritmi pentru gestionarea informaţiilor de rutare ca şi orice alt ruter dintr-o arie oarecare. Topologia backbone este invizibilă pentru toate ruterele intra-arie, după cum sunt şi topologiile ariilor individuale, invizibile pentru backbone. Ariile pot fi definite astfel încât backbone să nu fie contiguă. În acest caz, conectivitatea backbone trebuie refăcută prin legături virtuale. Legăturile virtuale sunt configurate între orice rutere backbone care împart o legătură cu o arie şi funcţionează ca şi cum ar fi legături directe.

Figura 4.10 prezintă un sistem autonom compus din câteva arii şi backbone-uri. Ruterele 4, 5, 6, 10, 11 şi 12 constituie backbone. Dacă hostul H1 din aria 3 doreşte să trimită un pachet către hostul H2 din aria 2, pachetul este trimis routerului 13, care îl direcţionează către ruterul 12, apoi către 11. Apoi ruterul 11 trimite pachetul prin backbone la Area Border Router 10, care trimite pachetul prin cele 2 rutere intra-arie 9 şi 7 pentru a ajunge la hostul H2.

Algoritmul SPF

Algoritmul de rutare SPF este fundamentul operaţiilor OSPF. Când un ruter SPF este pornit, se iniţializează structurile de date ale protocolului de rutare şi se aşteaptă confirmarea din partea protocoalelor de pe nivelele de mai de jos, că interfeţele sale funcţionează.

După ce un ruter se asigură că interfeţele sale funcţionează, foloseşte protocolul OSPF **hello** pentru a-şi descoperi vecinii. Pachetele hello mai sunt folosite şi ca *pachete keepalive* (se determină dacă mai este funcţională legătura). În reţele multiacces (*multiaccess networks* - care suportă mai mult de două rutere), protocolul hello alege un ruter pentru transmisie (***designated router, DR***) şi un ruter pentru transmisii în cazul excepţiilor (***backup designated router, BDR*****).** Un designated router este responsabil cu generarea mesajelor update pentru întreaga reţea multiacces. Designated routers permit reducerea traficului în reţea şi reducerea mărimii bazelor de date despre topologie.

Când bazele de date link-state ale două rutere vecine sunt sincronizate, se spune că ruterele sunt adiacente (*adjacent*). În rețelele multiacces ruterul desemnat determină care rutere ar trebui să devină adiacente. Adiacența controlează distribuția de pachete ale protocolului de rutare, fiind trimise şi primite pachete doar pe baza adiacenței.

Fiecare ruter trimite periodic mesaje update, pentru a furniza informații despre adiacențele unui ruter oarecare, sau pentru a-i informa pe ceilalți când starea unui ruter se schimbă. Comparând adiacențele stabilite cu starea legăturilor, ruterele căzute pot fi detectate rapid şi topologia rețelei poate fi schimbată să reflecte acest lucru. Din bazele de date despre topologie generate de mesajele update, fiecare ruter calculează un arbore cu rutele cele mai scurte (*shortest-path tree*) în care el însuşi este rădăcină. Apoi, arborele shortest-path devine tabelă de rutare.

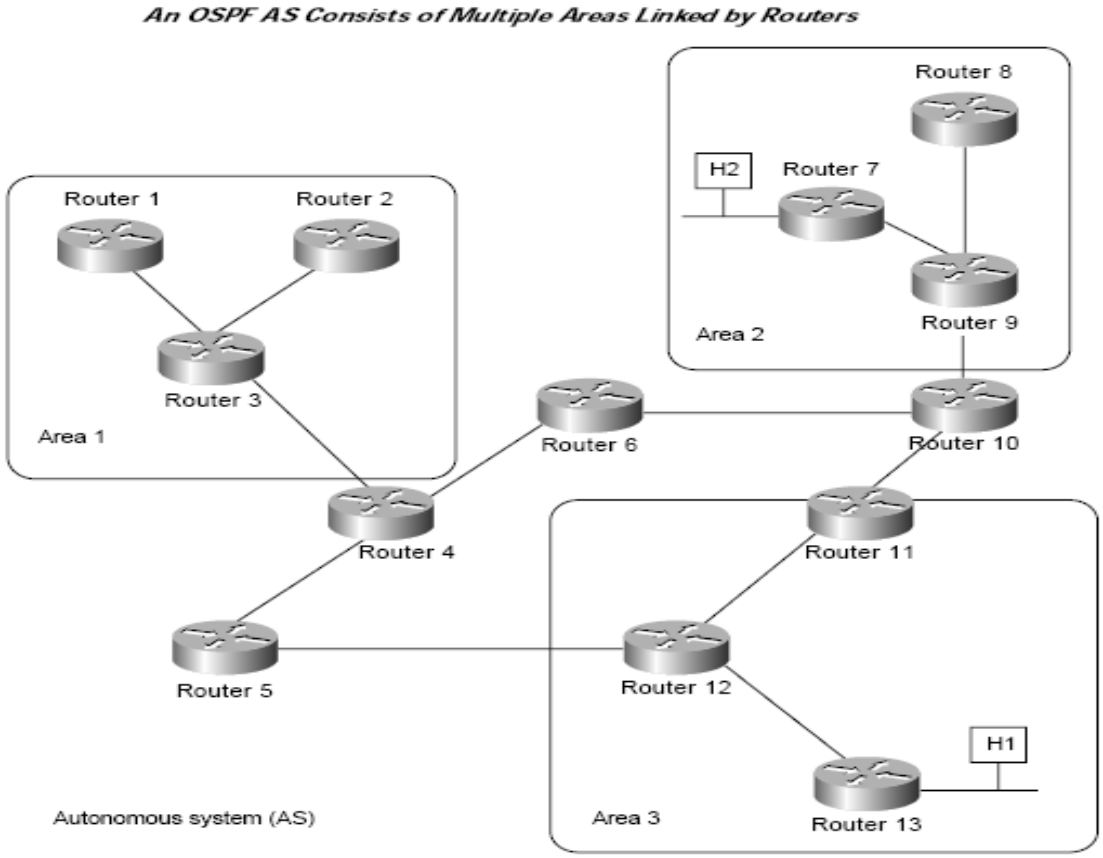


Fig. 4.10 Sistem autonom cu mai multe arii

Formatul pachetelor

Toate pachetele OSPF încep cu un header de 24 bytes, cum e ilustrat mai jos.

1-byte versiune	1-byte tip	2-bytes lungimea pachetului	4-bytes ID-ul ruterului	4-bytes ID-ul ariei	2-bytes checksum	2-bytes tipul autentificării	8-bytes autentificare	n-bytes date

tip	• **hello** Stabileşte şi menţine legătura cu vecinii. • **Database description** Descrie conţinutul bazelor de date despre topologie. Aceste mesaje sunt schimbate când este iniţializată o adiacenţă. • **link-state request** Cererile părţilor de baze de date despre topologie de la rutere vecine. Aceste mesaje sunt schimbate după ce un ruter descoperă (examinând pachetele de descriere a bazei de date) ca anumite părţi din baza de date despre topologie sunt prea vechi. • **link-state update** Răspunsurile la o cerere link-state request. Aceste mesaje sunt folosite pentru trimiterea regulată de pachete update. • **link-state acknowledgment** Confirmă pachetele link-state update.
lungimea pachetului	- specifică lungimea pachetului în bytes, fiind inclus şi header-ul OSPF;
id-ul routerului	- identifică sursa pachetului;
id-ul ariei	- identifică aria căreia îi aparţine pachetul; toate pachetele OSPF sunt asociate cu o arie unică;
checksum	- verifică dacă conţinutul întregului pachet a suferit erori în tranzit;
tipul autentificării	- conţine tipul autentificării; toate transmisiunile în protocolul OSPF sunt autentificate; tipul autentificarii este configurabil în funcţie de arie;
autentificarea date	- conţine informaţii despre autentificare;

Rezumat

Nivelul reţea se ocupă cu dirijarea pachetelor sau a fluxurilor de date prin nodurile intermediare de la sursă la destinaţie. Pentru stabilirea rutelor optime de la o sursă la o destinaţie se folosesc algoritmi de rutare. Pentru dirijarea pechetelor pe o anumită rută, echipamentele de rutare rulează algoritmi rutabili. Pentru controlul dirijării pachetelor şi cunoaşterea în orice moment a stării procesului de rutare se folosesc protocoale de control. Toate aceste protocoale rulează pe echipamente de reţea numite generic rutere. Acestea sunt veritabile sisteme de calcul capabile se proceseze în timp real o mulţime de informaţii de rutare.

În scopul facilizării procesului de rutare şi de adresare în reţele mari şi foarte mari, inclusiv în Internet, reţelele sunt organizate ierarhic şi ca sisteme autonome. În interiorul unui sistem autonom se folosesc protocoale de porţi interioare, iar între sistemele autonome se folosesc protocoale de porţi exterioare.

Identificarea reţelelor se face prin adrese de reţea. Formatul acestor adrese depinde de tipul de reţea. În Internet adresele au lungimea de 4 octeţi în cazul protocolului IPv4 şi 16 în cazul Ipv6. Tradiţional, adresele de Internet Ipv4 erau bazate pe clase de adrese şi utilizate ca atare. Epuizarea rapidă şi neaşteptată a spaţiului de adresare în varianta tradiţională, ca şi nevoia de simplificare a procesului de rutare în reţele foarte mari, au determinat o nouă concepţie a modului de folosire a

adreselor. Au apărut conceptele de rutare fără clasă, de mască de rețea cu lungime variabilă, de subdivizare a rețelelor şi de agregare a rutelor. Aceste artificii au permis temporar continuarea extinderii Internetului până la introducerea şi generalizarea protocolului IPv6.

Protocolul IP cu cele două versiuni ale sale nu este singurul protocol şi sistem de adresare de nivel 3. Diferite alte rețele, de exemplu rețelele SDH, ATM, ISDN etc. folosesc alte sisteme de adrese, alte protocoale de rutare şi, desigur, alte echipamente de rutare/comutare.

Întrebări de control

1. Care sunt formatele de adresă Ipv4? Cum se pot realiza rețele şi subrețele IP? Care este rolul măştii de rețea? Dați exemple.
2. Ce se înțelege prin congestie în rețele şi cum poate apărea. Care sunt mecanismele de control ale acesteia?
3. Care sunt asemănările şi deosebirile dintre rutarea de pachete şi rutarea de circuite?
4. Ce semnificație au adresele următoare: 173.56.0.0 şi 173. 56.255.255? Din ce clasă fac parte adresele 12.10.254.17, 168.128.13.29, 175.175.175.175?
5. Între ce limite se află primul octet al unei adrese clasă D (multicast)?
6. Ce rol au protocoale de control în Internet? Descrieți câteva protocoale (ARP, ICMP, RARP etc.)
7. Ce versiuni de protocoale Internet cunoaşteți? Descrieți IPv6: caracteristici, forma antetului, deosebiri față de IPv4.
8. Dați exemple argumentate de aplicații pentru care este adecvat serviciul orientat pe conexiune şi exemple pentru care este mai adecvat serviciul neorientat pe conexiune.
9. Dați exemple de parametrii ai protocolului care pot fi negociați în timpul stabilirii conexiunii.
10. Petru rețeaua din fig. 1 se foloseşte algoritmul de inundare pentru a trimite pachete din A în D. Care este numărul maxim de salturi la care trebuie setat contorul de salt pentru a nu inunda inutil rețeaua? Enumerați toate rutele pe care le va parcurge un pachet emis de A cu numărul maxim de salturi egal cu 3. Câte noduri consumă inutil timp de procesare?

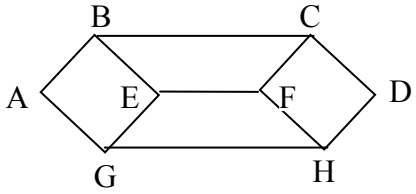
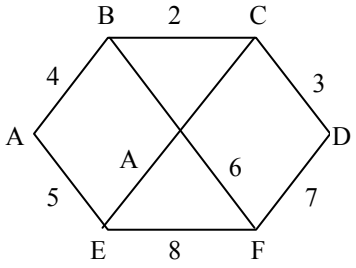


Fig 1

11. Se consideră rețeaua din figura 2 care foloseşte dirijarea după *Algoritmul vectorilor distanță*. La ruterul C tocmai a su sosit următorii vectori: de la B (5,0,8,12,6,2); de la D (16,12,6,0,9,10) şi de la E (7,6,3,9,0,4). Întârzierile măsurate către B, D şi E sunt 6, 3 şi respectiv 5. Care este noua tabelă de dirijare a lui C? Precizați atât linia de ieşire folosită cât şi întârzierea presupusă.



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

Fig. 2

12. Ce algoritmi se pot folosi pentru a calcula distanța minimă dintre noduri din rețea?