Reţele de calculatoare

#8 Cerinţele de proiectare ale nivelului reţea
Algoritmi de dirijare în rețea

FI-AIA-3-Retele de calculatoare-2022/2023

Adrian Runceanu

https://www.runceanu.ro/adrian/

Curs 8

1. Cerinţele de proiectare ale nivelului reţea
2. Algoritmi de dirijare în rețea

Nivelul reţea

1. Cerinţele de proiectare ale nivelului reţea

1.1 Comutare de pachete de tip Memorează-şi-Retransmite(Store-and-Forward)

1.2 Servicii furnizate nivelului transport

1.3 Implementarea serviciului neorientat pe conexiune

1.4 Implementarea serviciilor orientate pe conexiune

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

1. Cerinţele de proiectare ale nivelului reţea

1.1 Comutare de pachete de tip Memorează-şi-Retransmite (Store-and-Forward)

Contextul în care operează protocoalele de la nivelul reţea:

- Componentele majore ale sistemului sunt:
  - *echipamentul companiei de telecomunicaţii* (routere conectate prin linii de transmisie), prezentat în interiorul ovalului umbrit
  - *echipamentul clientului*, prezentat în afara ovalului
- Gazda H1 este conectată direct la unul dintre routerele companiei de telecomunicaţii A, printr-o linie închiriată.
- În contrast, gazda H2 este într-o reţea LAN cu un router, F, deţinut şi operat de către client.
- Acest context este prezentat în fig. 1.

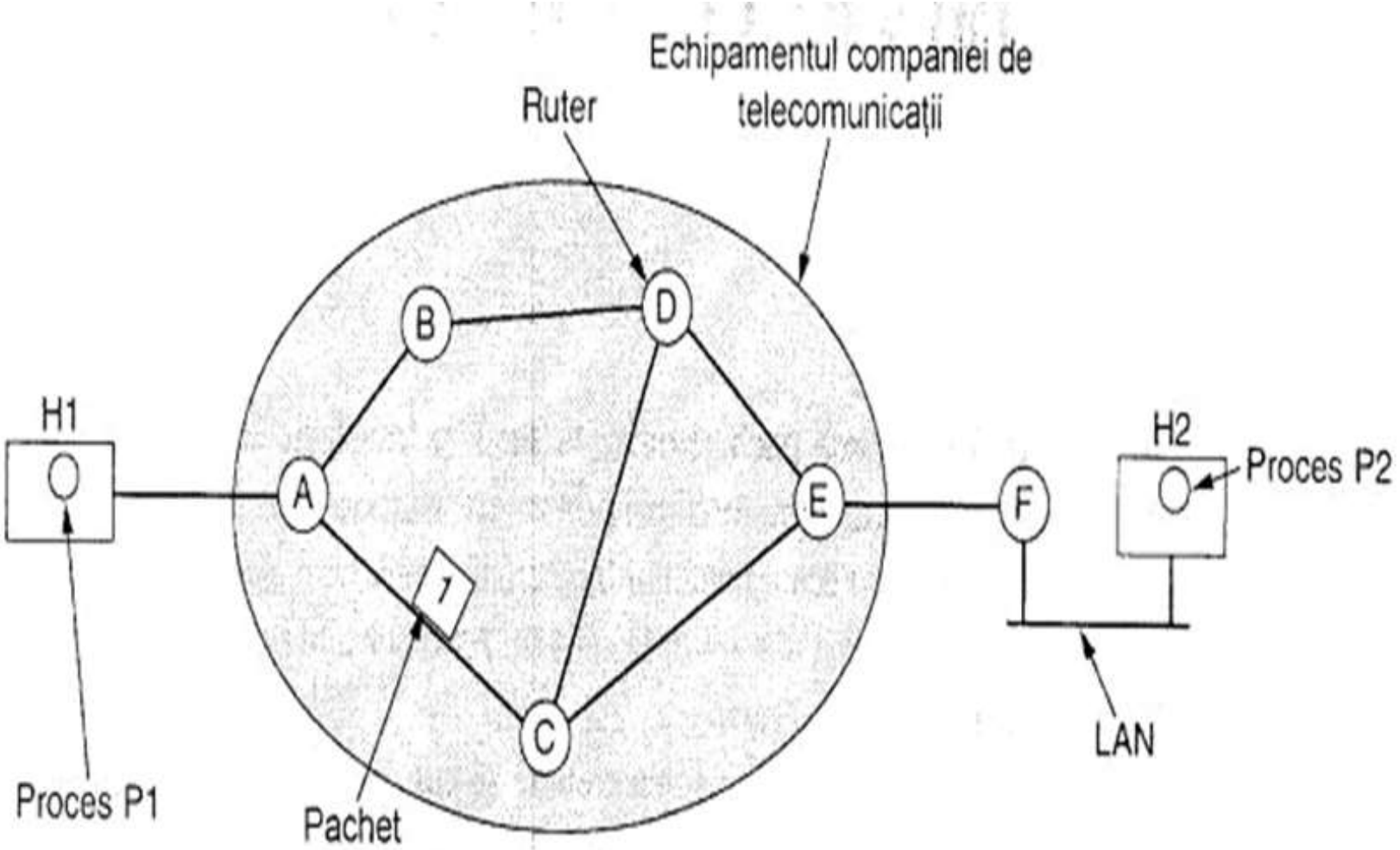


Fig. 1. Cadrul protocoalelor nivelului rețea

1.1 Comutare de pachete de tip Memorează-şi-Retransmite(Store-and-Forward)

Acest echipament este folosit după cum urmează:

- O gazdă care are de transmis un pachet îl transmite celui mai apropiat router, fie în aceeaşi reţea LAN, fie printr-o legătură punct la punct cu compania de telecomunicaţii.
- Pachetul este memorat acolo până ajunge integral, astfel încât să poată fi verificată suma de control.
- Apoi este trimis mai departe către următorul router de pe traseu, până ajunge la destinaţie, unde este livrat.
- Acest mecanism reprezintă comutarea de pachete de tip memorează-şi-retransmite.

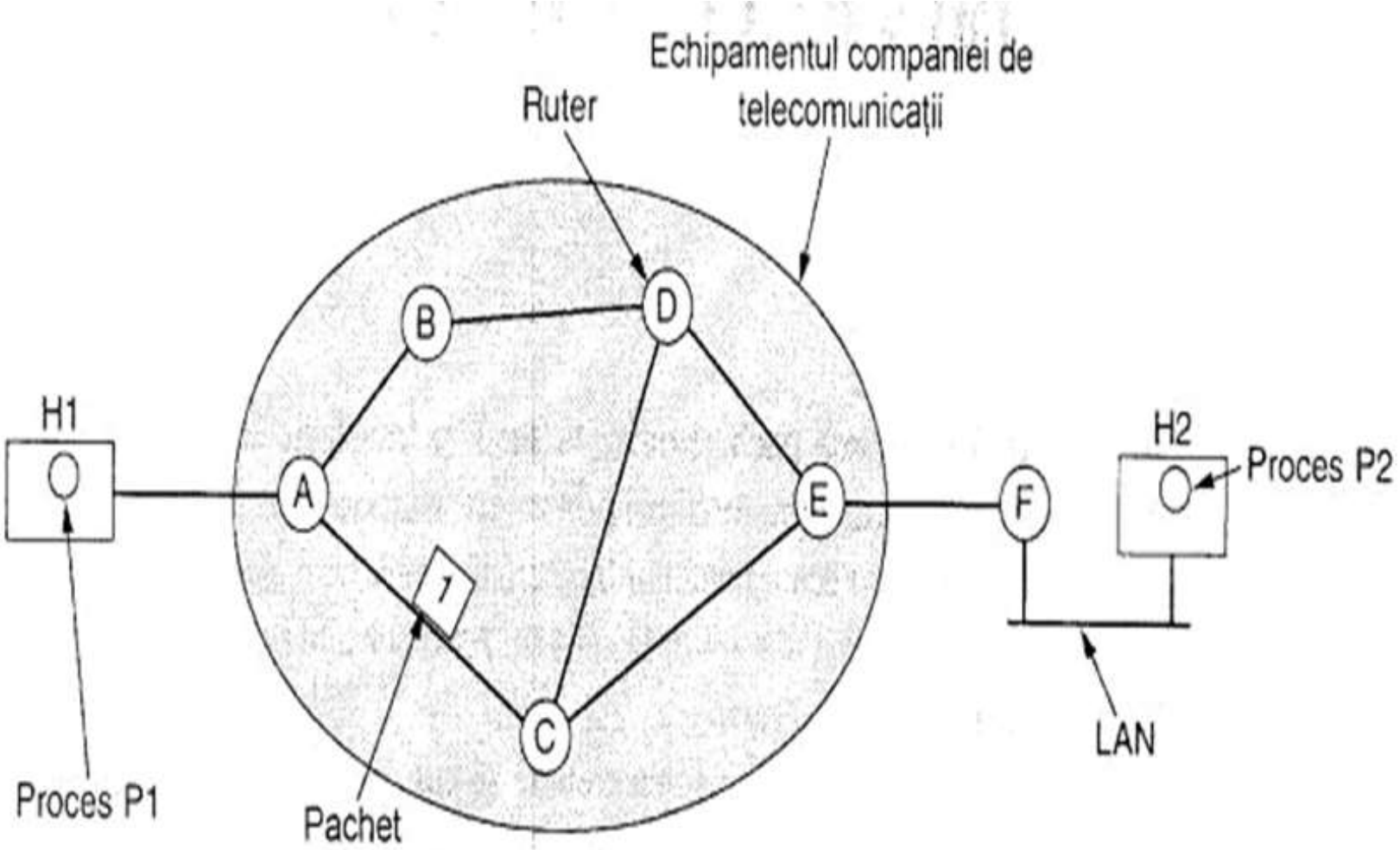


Fig. 1. Cadrul protocoalelor nivelului rețea

Nivelul reţea

1. Cerinţele de proiectare ale nivelului reţea

1.1 Comutare de pachete de tip Memorează-şi-Retransmite(Store-and-Forward)

1.2 Servicii furnizate nivelului transport

1.3 Implementarea serviciului neorientat pe conexiune

1.4 Implementarea serviciilor orientate pe conexiune

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

1.2 Servicii furnizate nivelului transport

Serviciile nivelului reţea au fost proiectate având în vedere următoarele scopuri:

1. Serviciile trebuie să fie *independente de tehnologia routerului*.
2. *Nivelul transport* trebuie să fie *independent de numărul, tipul şi topologia routerelor existente*.
3. *Adresele de reţea* disponibile la nivelul transport *trebuie să folosească o schemă de numerotare uniformă*, chiar în cadrul reţelelor **LAN** şi **WAN**.

1.2 Servicii furnizate nivelului transport

- Obiectivele fiind stabilite, proiectantul nivelului reţea are o mare libertate în a scrie specificaţiile detaliate ale serviciilor oferite nivelului transport.
- Această libertate degenerează adesea pareri diferite între două variante diferite.

- Problema centrală a discuţiei este dacă nivelul reţea trebuie să furnizeze:
  1. *servicii orientate pe conexiune*
  2. *servicii neorientate pe conexiune*

1.2 Servicii furnizate nivelului transport

1. O primă variantă (reprezentată de comunitatea Internet) este aceea prin care se afirmă că *scopul routerului este de a transfera pachete şi nimic mai mult*.

- În aceasta variantă subreţeaua este inerent nesigură, indiferent cum ar fi proiectată.

- De aceea *calculatoarele gazdă trebuie să accepte faptul că reţeaua este nesigură şi să facă controlul erorilor (adica, detecţia şi corecţia erorii) şi controlul fluxului ele însele.*

1.2 Servicii furnizate nivelului transport

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

1.2 Servicii furnizate nivelului transport

2. Cea de a doua variantă (reprezentată de companiile de telefoane) afirmă că *subreţeaua trebuie să asigure un serviciu orientat pe conexiune sigur.*

- Ei susţin că peste 100 de ani de experienţă cu sistemul telefonic mondial reprezintă un ghid excelent.
- În această perspectivă, calitatea serviciului este elementul dominant, şi într-o subreţea fără conexiuni, calitatea serviciului este dificil de obţinut, în special pentru trafic în timp real cum ar fi voce şi imagine.

1.2 Servicii furnizate nivelului transport

Aceste două variante sunt cel mai bine exemplificate de reţeaua Internet şi reţelele ATM.

1. Reţeaua Internet oferă un *serviciu la nivelul reţea neorientat pe conexiune*
2. Reţelele ATM oferă un *serviciu la nivelul reţea orientat pe conexiune*

Totuşi, este interesant de notat că, cu cât garantarea calităţii serviciului devine din ce în ce mai importantă, Internet-ul evoluează.

Nivelul reţea

1. Cerinţele de proiectare ale nivelului reţea

1.1 Comutare de pachete de tip Memorează-şi-Retransmite(Store-and-Forward)

1.2 Servicii furnizate nivelului transport

1.3 Implementarea serviciului neorientat pe conexiune

1.4 Implementarea serviciilor orientate pe conexiune

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

1.3 Implementarea serviciului neorientat pe conexiune

- După ce am văzut cele două clase de servicii pe care nivelul reţea le furnizează utilizatorilor săi, este momentul să vedem funcţionarea internă a acestui nivel.

- Sunt posibile două organizări diferite, în funcţie de tipul serviciului oferit:

1. serviciul neorientat pe conexiune
2. serviciul orientat conexiune

1.3 Implementarea serviciului neorientat pe conexiune

1. Dacă este oferit un serviciu neorientat pe conexiune, atunci *pachetele sunt trimise în subreţea individual şi dirijate independent de celelalte*.

- Nu este necesară nici o iniţializare prealabilă.
- În acest context, pachetele sunt numite frecvent **datagrame** (datagrams) (prin analogie cu telegramele), iar subreţeaua este numită **subreţea datagramă** (datagram subnet).

1.3 Implementarea serviciului neorientat pe conexiune

2. Dacă este folosit serviciul orientat conexiune, atunci, *înainte de a trimite pachete de date, trebuie stabilită o cale de la routerul sursă la routerul destinaţie*.

- Această conexiune este numită **VC - circuit virtual (virtual circuit)**, prin analogie cu circuitele fizice care se stabilesc în sistemul telefonic, iar subreţeaua este numită **subreţea cu circuite virtuale (virtual-circuit subnet).**

1.3 Implementarea serviciului neorientat pe conexiune

Să vedem cum funcţionează o **subreţea datagramă**:

- Să presupunem că procesul P1 din fig. 2 are un mesaj lung pentru procesul P2.
- El transmite mesajul nivelului transport, cu instrucţiunile 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* şi transferă rezultatul nivelului reţea, probabil o altă procedură din cadrul sistemului de operare.

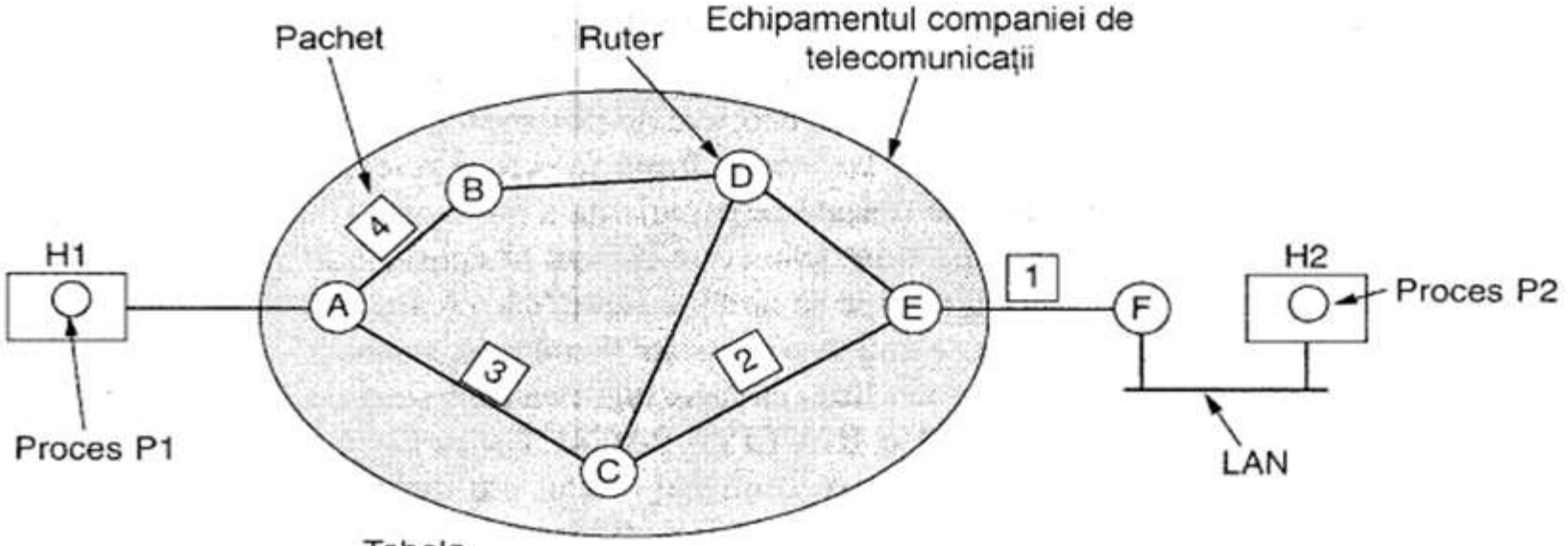


Tabela ruterului A

iniţial		ulterior	
A	–	A	–
B	B	B	B
C	C	C	C
D	B	D	B
E	C	E	B
F	C	F	B

Dest. Rută

Tabela ruterului C

A	A
B	A
C	–
D	D
E	E
F	E

Tabela ruterului E

A	C
B	D
C	C
D	D
E	–
F	F

Fig. 2. Dirijarea într-o subreţea datagramă

1.3 Implementarea serviciului neorientat pe conexiune

- Să presupunem că mesajul este de patru ori mai lung decât dimensiunea maximă a unui pachet, aşa că nivelul reţea trebuie să îl spargă în patru pachete, 1, 2, 3, şi 4 şi să le trimită pe fiecare în parte routerului F, folosind un **protocol punct-la-punct**, de exemplu, **PPP (point-to-point protocol)**.
- Din acest punct controlul este preluat de compania de telecomunicaţii.
- Fiecare router are o tabelă internă care îi spune unde să trimită pachete pentru fiecare destinaţie posibilă.

1.3 Implementarea serviciului neorientat pe conexiune

- *Fiecare intrare în tabelă* este *o pereche compusă din destinaţie şi linia de ieşire folosită pentru acea destinaţie.*
- Pot fi folosite doar linii conectate direct.
- De exemplu, în fig. 2, A are doar două linii de ieşire - către B şi C - astfel că fiecare pachet ce vine trebuie trimis către unul dintre aceste routere, chiar dacă ultima destinaţie este alt router.
- Tabela de rutare iniţială a lui A este prezentată în figură sub eticheta „iniţial".

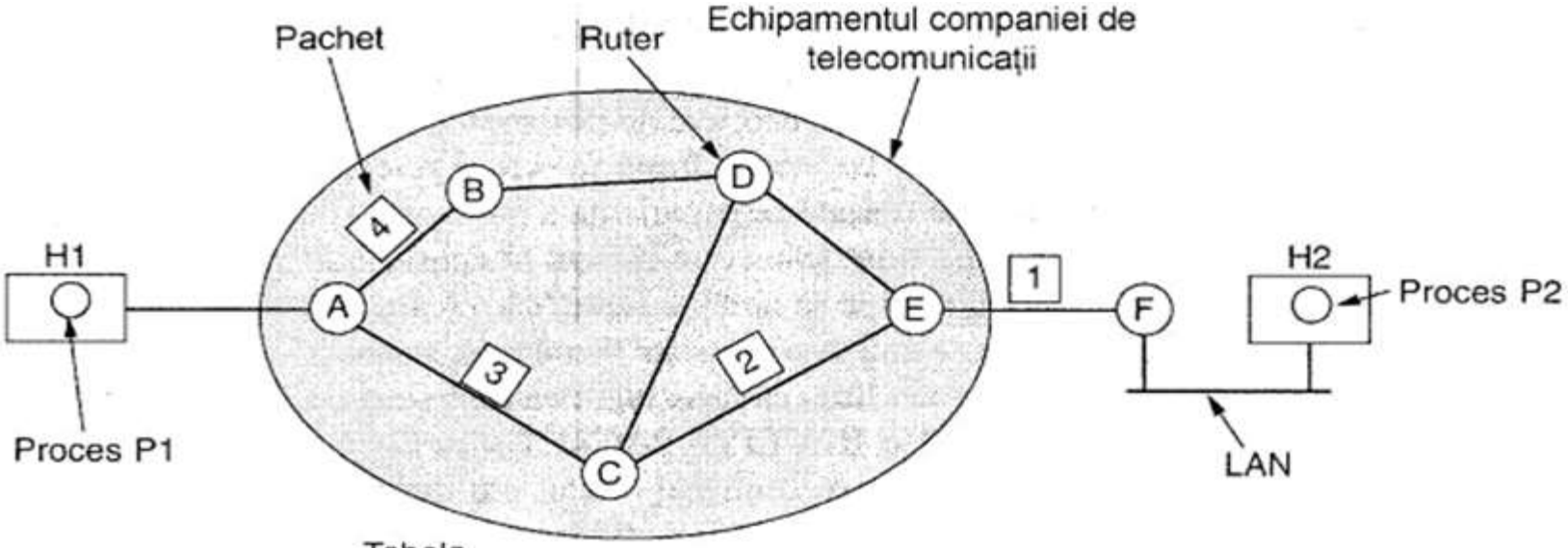


Tabela ruterului A

iniţial	ulterior
A –	A –
B B	B B
C C	C C
D B	D B
E C	E B
F C	F B

Dest. Rută

Tabela ruterului C

A	A
B	A
C	–
D	D
E	E
F	E

Tabela ruterului E

A	C
B	D
C	C
D	D
E	–
F	F

Fig. 2. Dirijarea într-o subreţea datagramă

1.3 Implementarea serviciului neorientat pe conexiune

- Cum au ajuns la A, pachetele 1, 2 şi 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 şi apoi către F.
- Când a ajuns la F, a fost încapsulat într-un cadru al nivelului legătură de date şi trimis către calculatorul gazdă H2 prin reţeaua LAN.

1.3 Implementarea serviciului neorientat pe conexiune

- Totuşi, ceva diferit s-a întâmplat cu pachetul 4.
- Când a ajuns la A, a fost trimis către routerul B, chiar dacă şi 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 şi şi-a actualizat tabela de rutare, aşa cum apare sub eticheta „ulterior".
- *Algoritmul ce administrează tabelele şi ia deciziile de rutare* se numeşte **algoritm de rutare** (routing algorithm).

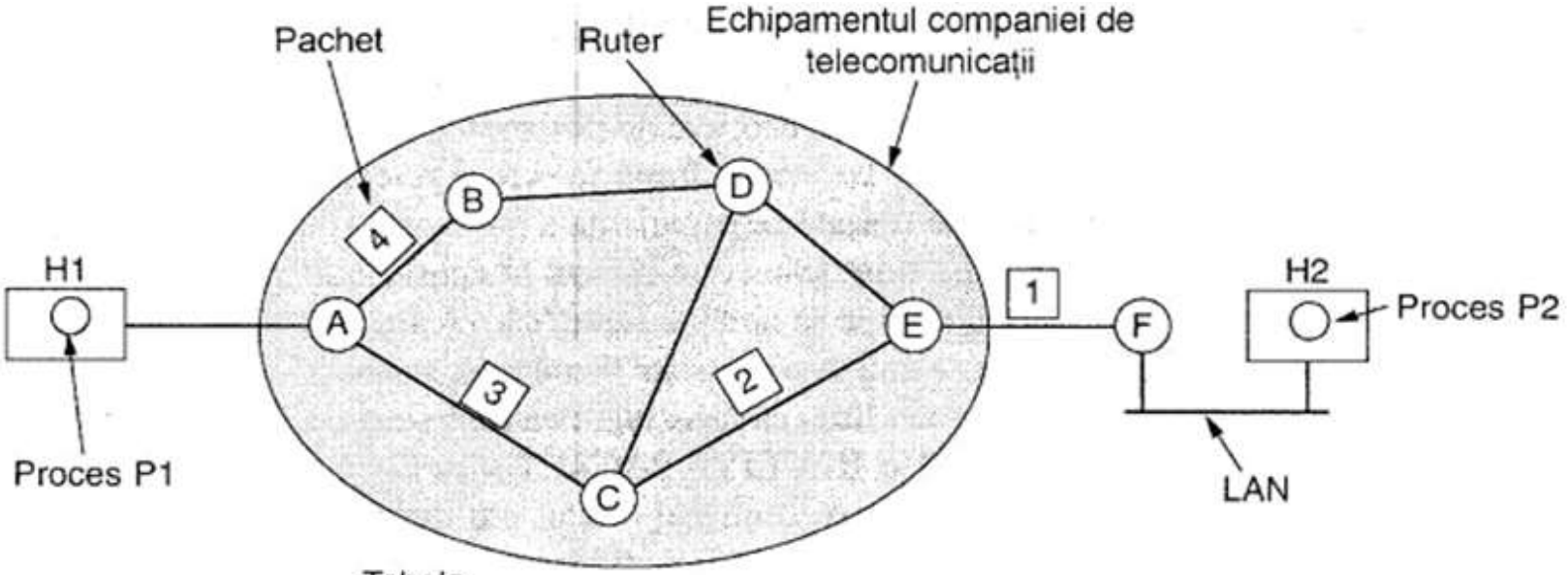


Tabela ruterului A

inițial		ulterior	
A	–	A	–
B	B	B	B
C	C	C	C
D	B	D	B
E	C	E	B
F	C	F	B

Dest. Rută

Tabela ruterului C

A	A
B	A
C	–
D	D
E	E
F	E

Tabela ruterului E

A	C
B	D
C	C
D	D
E	–
F	F

Fig. 2. Dirijarea într-o subrețea datagramă

Nivelul reţea

1. Cerinţele de proiectare ale nivelului reţea

1.1 Comutare de pachete de tip Memorează-şi-Retransmite(Store-and-Forward)

1.2 Servicii furnizate nivelului transport

1.3 Implementarea serviciului neorientat pe conexiune

1.4 Implementarea serviciilor orientate pe conexiune

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

1.4 Implementarea serviciilor orientate pe conexiune

Pentru serviciile orientate conexiune, avem nevoie de o subreţea cu circuite virtuale

- Ideea care se stă la baza circuitelor virtuale este evitarea alegerii unei noi căi (rute) pentru fiecare pachet trimis, ca în fig. 2.
- În schimb, atunci când se stabileşte o conexiune, se alege o cale între maşina sursă şi maşina destinaţie, ca parte componentă a iniţializării conexiunii şi aceasta este memorată în tabelele routerelor.
- Acea cale este folosită pentru tot traficul de pe conexiune, exact în acelaşi mod în care funcţionează sistemul telefonic.
- Atunci când conexiunea este eliberată, este închis şi circuitul virtual.
- În cazul serviciilor orientate conexiune, *fiecare pachet poartă un identificator care spune cărui circuit virtual îi aparţine*.

1.4 Implementarea serviciilor orientate pe conexiune

- De exemplu, să considerăm situaţia din fig. 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 routerul 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 iniţializează conexiunea şi aceasta este singura conexiune) şi indică subreţelei să stabilească circuitul virtual.
- Aceasta conduce la a doua linie din tabele.
- Observaţi că apare un conflict deoarece deşi A poate distinge uşor 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 ieşire al celei de a doua conexiuni.
- Pentru *evitarea conflictelor de acest gen routerele trebuie să poată înlocui identificatorii de conexiune în pachetele care pleacă.*
- În unele contexte, aceasta se numeşte **comutarea etichetelor** (label switching).

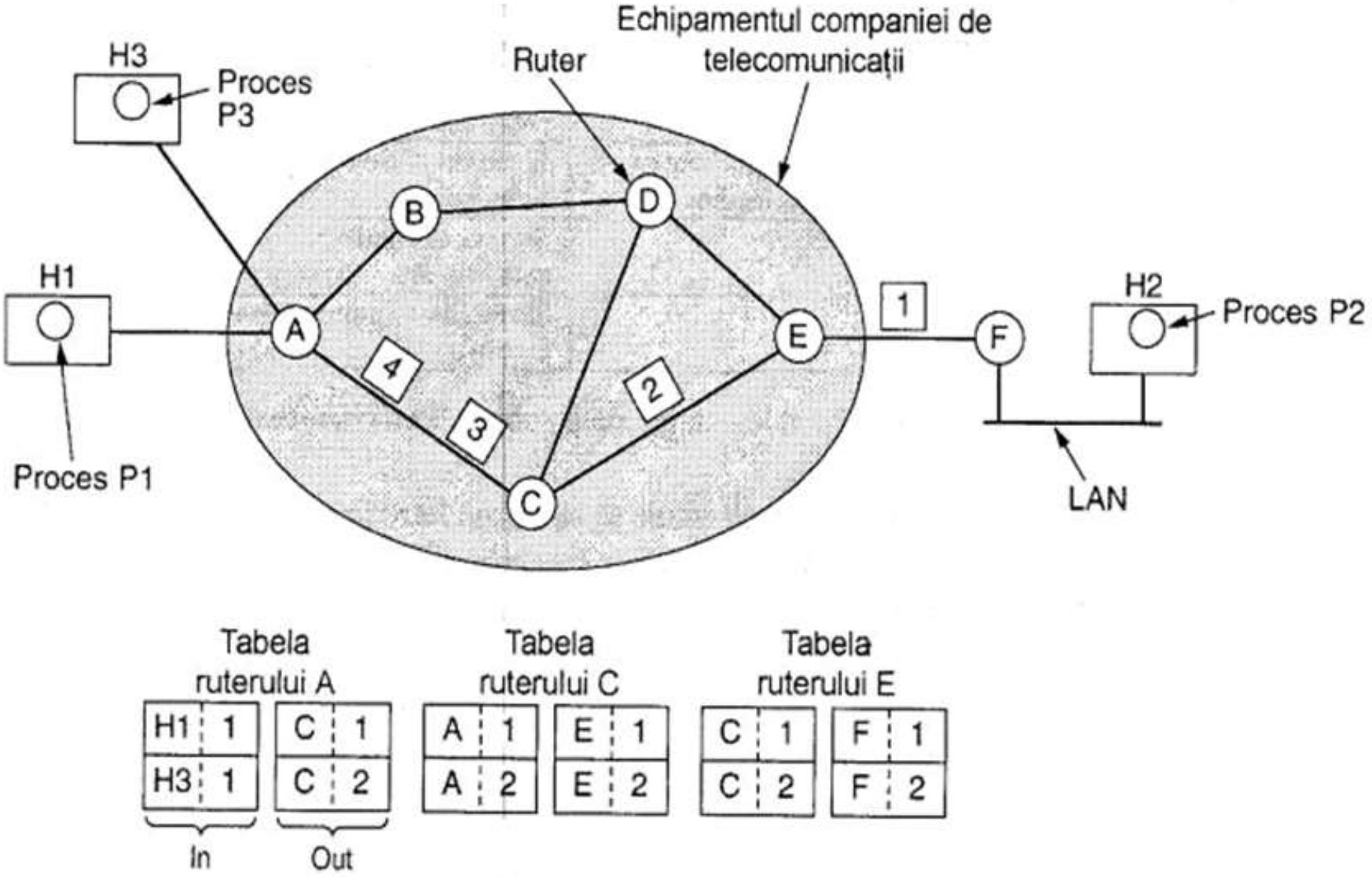


Fig. 3. Dirijare în cadrul unei subrețele cu circuite virtuale

Nivelul reţea

1. Cerinţele de proiectare ale nivelului reţea

1.1 Comutare de pachete de tip Memorează-şi-Retransmite(Store-and-Forward)

1.2 Servicii furnizate nivelului transport

1.3 Implementarea serviciului neorientat pe conexiune

1.4 Implementarea serviciilor orientate pe conexiune

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

- Atât circuitele virtuale cât şi datagramele au suporteri şi oponenţi.
- Vom încerca acum să rezumăm argumentele ambelor variante.
- Principalele aspecte sunt prezentate în figura urmatoare:

Problemă	Subreţea datagramă	Subreţea cu circuite virtuale (CV)
Stabilirea circuitului	Nu este necesară	Obligatorie
Adresare	Fiecare pachet conţine adresa completă pentru sursă şi destinaţie	Fiecare pachet conţine un număr mic de CV
Informaţii de stare	Routerele nu păstrează informaţii despre conexiuni	Fiecare CV necesită spaţiu pentru tabela routerului per conexiune
Dirijare	Fiecare pachet este dirijat independent	Calea este stabilită la iniţierea CV; toate pachetele o urmează
Efectul defectării routerului	Nici unul, cu excepţia pachetelor pierdute în timpul defectării	Toate circuitele virtuale care trec prin routerul 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 in avans suficiente resurse

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

- În interiorul subreţelei există situaţii în care trebuie să se aleagă între facilităţi antagoniste specifice fie *circuitelor virtuale*, fie *datagramelor*.

1. Un astfel de compromis este acela între spaţiul de memorie al routerului şi lăţimea de bandă.

- *Circuitele virtuale permit pachetelor să conţină numere de circuite în locul unor adrese complete*.

- Dacă pachetul tinde să fie foarte mic, atunci existenţa unei adrese complete în fiecare pachet poate reprezenta o supraîncărcare (overhead) importantă şi deci o irosire a lăţimii de bandă.
- Preţul plătit pentru folosirea internă a circuitelor virtuale este spaţiul necesar păstrării tabelei în router.

- Soluţia mai ieftină este determinată de *raportul între costul circuitelor de comunicaţie şi cel al memoriei routerului*.

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

2. Alt compromis este cel între ***timpul necesar stabilirii circuitului şi timpul de analiză a adresei***.

- Folosirea circuitelor virtuale presupune existenţa unei faze iniţiale de stabilire a căii, care cere timp şi consumă resurse.
- Oricum, este uşor să ne imaginăm ce se întâmplă cu un pachet de date într-o subreţea bazată pe circuite virtuale:
  - *routerul foloseşte numărul circuitului ca un index într-o tabelă pentru a afla unde merge pachetul*
- Intr-o reţea bazată pe datagrame, pentru a găsi intrarea corespunzătoare destinaţiei *se foloseşte o procedură de căutare mult mai complicată*.

3. O altă problemă este cea a ***dimensiunii spaţiului necesar pentru tabela din memoria router-ului***.

- O subreţea datagramă necesită o intrare pentru fiecare destinaţie posibilă, în timp ce o subreţea cu circuite virtuale necesită o intrare pentru fiecare circuit virtual.

- Totuşi, acest avantaj este relativ iluzoriu deoarece şi pachetele de iniţializare a conexiunii trebuie rutate, iar ele folosesc adresele destinaţie, la fel ca şi datagramele.

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

- Circuitele virtuale au unele **avantaje** în *garantarea calităţii serviciului şi evitarea congestionării subreţelei*, deoarece resursele (de exemplu zone tampon, lărgime de bandă şi cicluri CPU) pot fi rezervate în avans, atunci când se stabileşte conexiunea.
- La sosirea pachetelor, *lăţimea de bandă necesară şi capacitatea routerului vor fi deja pregătite*.

- Pentru o subreţea bazată pe datagrame, evitarea congestionării este mult mai dificilă.

- Pentru sistemele de prelucrare a tranzacţiilor (de exemplu *apelurile magazinelor pentru a verifica cumpărături realizate cu cărţi de credit*) supraincarcarea datorata stabilirii şi eliberarii unui circuit virtual poate reduce cu uşurinţă utilitatea circuitului.
- *Dacă majoritatea traficului este de acest tip, folosirea internă a circuitelor virtuale în cadrul subreţelei nu prea are sens.*
- Pe de altă parte, ar putea fi de folos circuite virtuale permanente, stabilite manual şi care să dureze luni sau chiar ani.

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

Circuitele virtuale au o problemă de vulnerabilitate:

- Dacă un router se defectează şi îşi pierde conţinutul memoriei, *atunci toate circuitele virtuale care treceau prin el sunt suprimate*, chiar dacă acesta îşi revine după o secundă.
- Prin contrast, *dacă se defectează un router bazat pe datagrame* vor fi afectaţi doar acei utilizatori care aveau pachete memorate temporar în cozile de aşteptare ale routerului şi este posibil ca numărul lor să fie şi mai mic, în funcţie de câte pachete au fost deja confirmate.

1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă

- *Pierderea liniei de comunicaţie este fatală pentru circuitele virtuale care o folosesc, însă poate fi uşor compensată dacă se folosesc datagrame.*

- De asemenea, datagramele permit routerului să echilibreze traficul prin subreţea, deoarece căile pot fi modificate parţial în cursul unei secvenţe lungi de pachete transmise.

Curs 8 (partea II)

2. Algoritmi de dirijare în rețea

Curs 8

Algoritmi de dirijare

1. **Algoritmul Dijkstra**

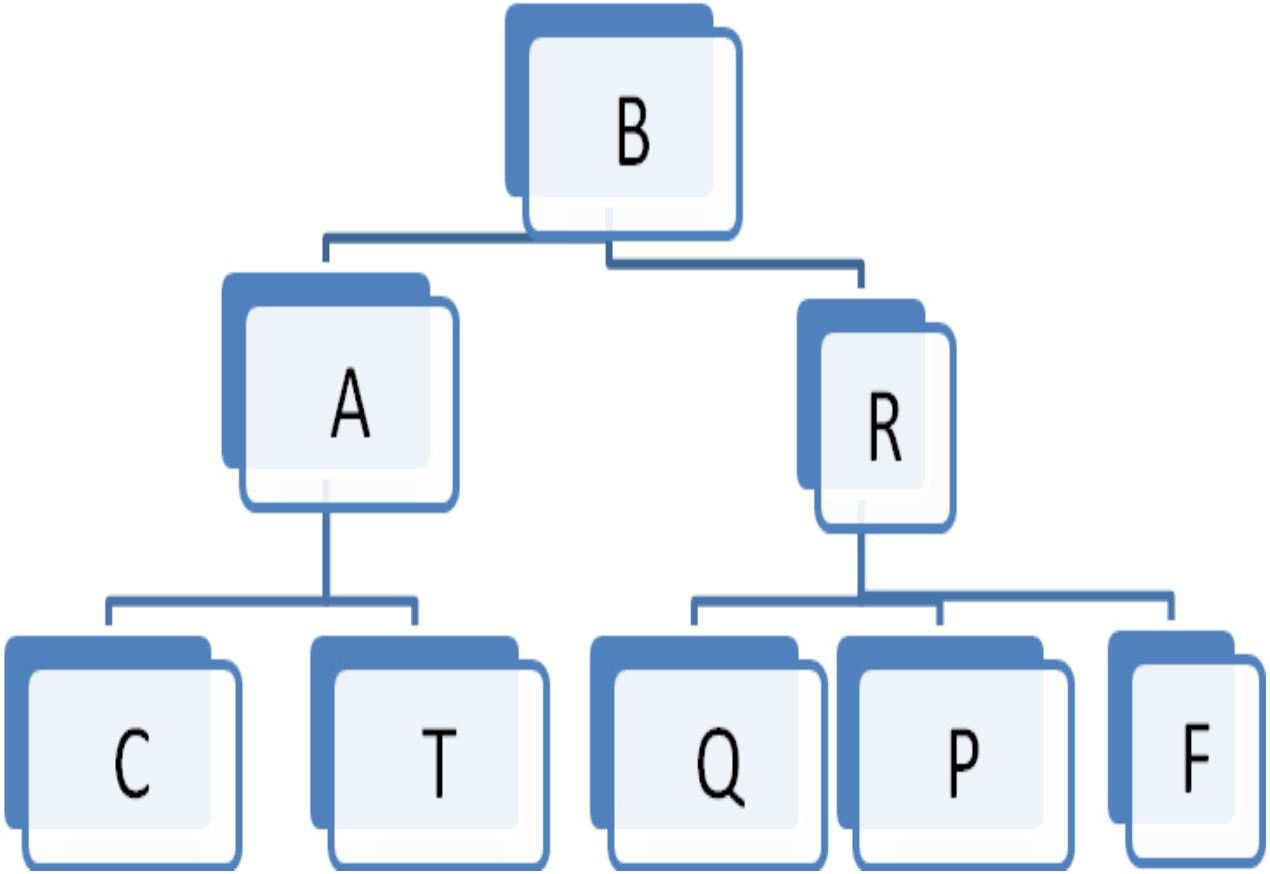
2. **Algoritmul Belmann-Ford**

Principiul optimalităţii

- *Principiul optimalităţii presupune că dacă o rută optimă intre nodul A şi nodul B trece prin nodul C atunci ruta optimă intre B şi C este inclusă în ruta A-B.*

- O consecinţă a principiului optimalităţii este formarea arborelui de scufundare care semnifică *mulţimea rutelor optime către un nod destinaţie*, reprezentat de rădăcina arborelui.

- Scopul algoritmilor de dirijare constă în *descoperirea arborilor de scufundare pentru fiecare rută*.



Arbore de scufundare pentru nodul B

Algoritmi de dirijare

- *Algoritmul de dirijare este acea parte a software-lui nivelului reţea care răspunde de alegerea liniei de ieşire pe care un pachet recepţionat trebuie trimis mai departe*.
- În cazul datagramelor această decizie trebuie luată din nou pentru fiecare pachet recepţionat, deoarece e posibil ca cea mai bună rută să se fi modificat între timp.
- În cazul circuitelor virtuale, decizia de dirijare se ia doar la stabilirea unui nou circuit virtual.
- După aceea pachetele vor urma doar calea stabilită anterior.
- Acest ultim caz este numit uneori sesiune de dirijare deoarece calea rămâne în funcţiune pentru o întreagă sesiune utilizator (ex. transfer de fişiere, login prin terminal).

Clasificare

Algoritmii pot fi:

1. neadaptivi - *alegerea căii se calculează în avans (of line) şi parvine routerului la iniţializarea reţelei*

2. adaptivi - *îşi modifică deciziile de dirijare pentru a reflecta modificările de topologie şi pe cele de trafic*

Algoritmii adaptivi diferă prin:

1. *locul de unde îşi iau informaţia*:
   a) local
   b) de la un vecin
   c) de la toate routerele
2. *momentul în care schimbă rutele*:
   a) când se schimbă încărcarea
   b) când se schimbă topologia
   c) la ΔT secunde
3. *metrica folosită pentru optimizare*:
   a) distanţa
   b) timpul de tranzit
   c) numărul de salturi

1. Algoritmi statici de dirijare

1.1. Dirijarea pe calea cea mai scurtă (shortest path)

- *Calea cea mai scurtă este dictată de metrica de măsurare a distanţei.*
- Aceasta poate fi:
  - distanţa în km
  - numărul de salturi
  - rata de transfer
  - traficul mediu
  - costul comunicaţiei
  - lungimea minimă a cozilor de aşteptare
  - întârzieri măsurate

1.2. Algoritmul de inundare - generează foarte multe pachete.

Pentru a limita numărul acestora se procedează astfel:

1. contor de salturi în antetul fiecărui pachet decrementat la fiecare salt şi care face ca pachetul să fie distrus atunci când contorul atinge valoarea 0.

2. routerul sursă să plaseze un număr de secvenţă în fiecare pachet pe care îl primeşte de la calculatorul gazdă asociat.

1.2. Algoritmul de inundare

- *Fiecare router necesită menţinerea unei liste pentru fiecare router sursă, cu numere de secvenţă iniţiate de acel router sursă şi care au fost deja trimis mai departe.*
- Dacă soseşte un pachet care deja se află în listă el nu se mai trimite.
- Se poate însoţi lista de un contor *k* care semnifică faptul că toate numerele de secvenţă până la *k* au fost deja tratate.
- Inundarea se utilizează:
  - în aplicaţii militare
  - în aplicaţii cu baze de date distribuite în care este necesar să fie actualizate concurent

1.3. Dirijarea bazată pe flux

- În condiţiile în care traficul mediu de la i la j este cunoscut în avans şi într-o aproximare rezonabilă, constant în timp, este posibilă analiza matematică a fluxurilor pentru a optimiza dirijarea.
- Ideea care stă la baza analizei este aceea că pentru o anumită linie dacă se cunosc capacitatea şi fluxul mediu, este posibil să se calculeze întârzierea medie a unui pachet pe linia respectivă, folosind teoria cozilor.

- Pe baza întârzierilor medii ale tuturor liniilor se poate calcula imediat, ca medie ponderată după flux, întârzierea medie a unui pachet pentru întreaga subreţea.

- *Problema dirijării se reduce apoi la găsirea algoritmului de dirijare care produce întârzierea medie minimă pentru subreţea.*

- Folosirea acestei tehnici presupune:
  - cunoaşterea topologiei subreţelei
  - matricea traficului
  - matricea capacităţilor liniilor, C_i (măsurată în Kbps)

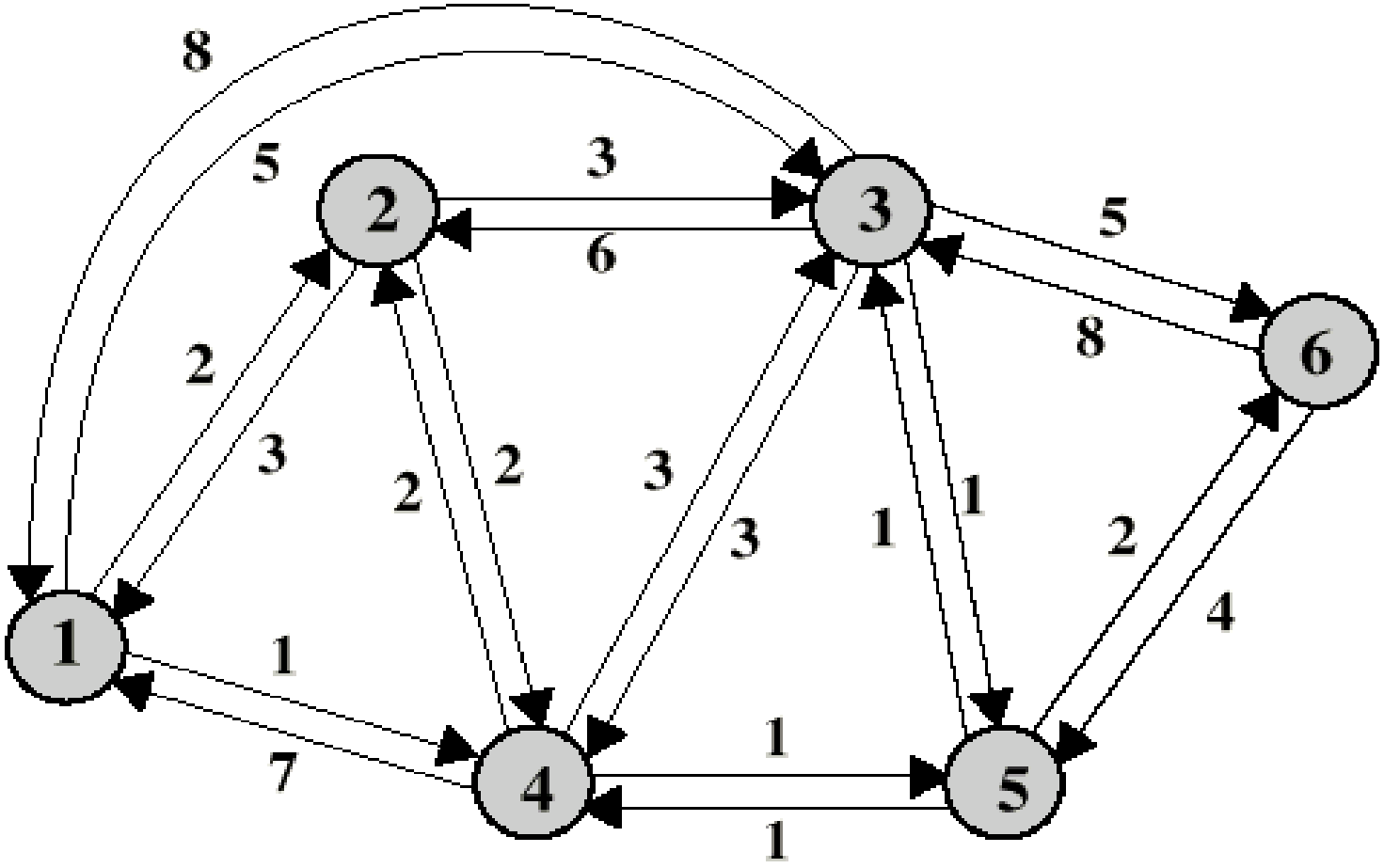
2. Algoritmi dinamici de dirijare

Cei mai folosiţi sunt:

1. Algoritmul de dirijare cu vectori distanţă
2. Algoritmul de dirijare bazat pe starea legăturilor

Algoritmul de dirijare cu vectori distanţă (Bellman - Ford - Fulkerson)

Costurile rutelor



Strategii de Rutare

1. Rutare fixă
2. Rutare bazată pe inundare
3. Rutare aleatoare
4. Rutare adaptivă

1. Rutare Fixă

- O singură cale pentru fiecare pereche sursă-destinaţie
- Rutele sunt determinate printr-un *algoritm de cost minim*
- Rutele rămân fixe, până la schimbarea topologiei reţelei

Tabele de Rutare Fixe

CENTRAL ROUTING DIRECTORY

To Node \ From Node	1	2	3	4	5	6
1	—	1	5	2	4	5
2	2	—	5	2	4	5
3	4	3	—	5	3	5
4	4	4	5	—	4	5
5	4	4	5	5	—	5
6	4	4	5	5	6	—

Node 1 Directory

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Directory

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Directory

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Node 4 Directory

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Node 5 Directory

Destination	Next Node
1	4
2	4
3	3
4	4
6	6

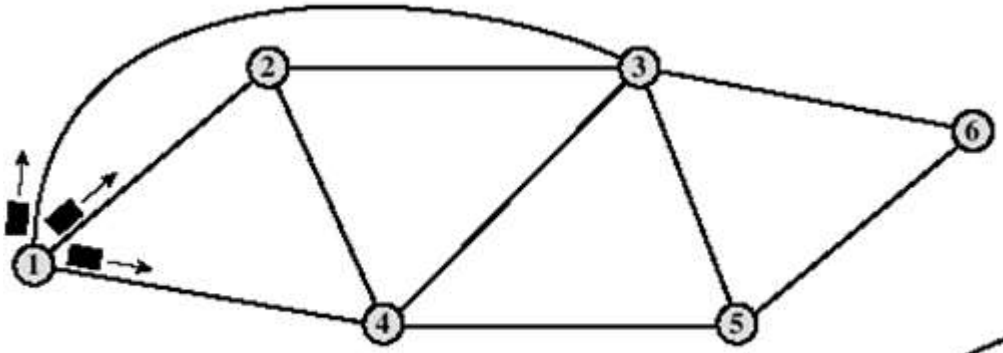
Node 6 Directory

Destination	Next Node
1	5
2	5
3	5
4	5
5	5

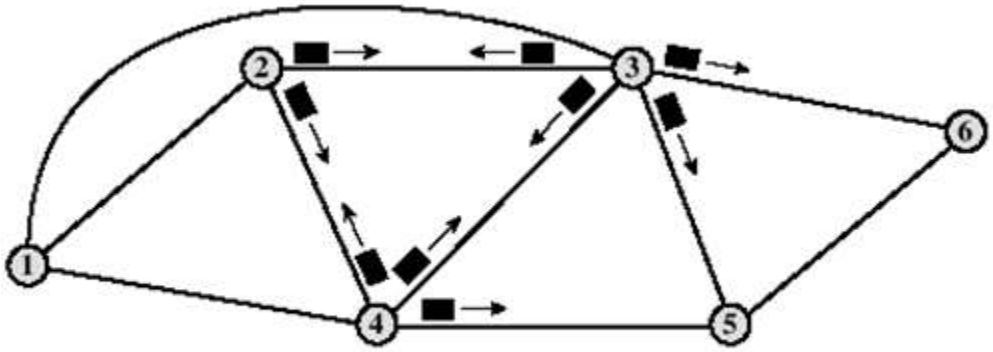
2. Rutare bazata pe inundare

- Nu sunt necesare informatii despre reţea
- *Pachetul este trimis la toţi vecinii*
- *Sau la toţi în afară de unde a venit*
- Un număr de copii ajung după un timp la destinaţie
- Fiecare pachet are un număr unic, duplicatele se ignoră
- Nodurile pot reţine identitatea pachetelor pentru a nu le ruta din nou
- Se poate defini un timp de viaţă a pachetelor

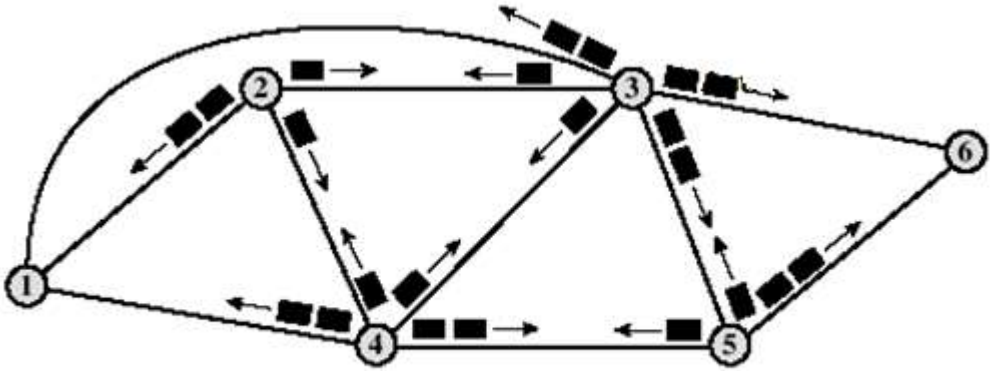
Inundare Exemplu



(a) Primul salt



(b) Al doilea salt



(c) Al treilea salt

Proprietăţi ale Inundării

- TOATE rutele posibile sunt încercate
  - foarte robust
- Cel puţin un pachet va ajunge pe calea de cost minim
  - Se poate folosi pt. stabilirea unui circuit virtual
- Toate nodurile sunt atinse
  - Util pentru distribuirea de informaţii (ex. rutare)

3. Rutare Aleatoare

- *Nodul selectează o cale de ieşire pentru transmiterea unui pachet primit*

- Selecţia poate fi aleatoare sau round-robin (fiecare nod este analizat)
- Se pot utiliza şi probabilităţi
- *Nu sunt necesare informaţii despre reţea*
- Ruta nu este în general optimă
- Trafic inutil mai mic ca la inundare

4. Rutare Adaptivă - caracteristici

- **Rutarea adaptiva** este cel mai des utilizată
- Decizia de rutare se adaptează condiţiilor din reţea:
  - Defecte de linie sau noduri
  - Congestie
- *Necesită informaţii despre reţea*
- Decizia este mai complexă
- Compromis între calitatea reţelei şi overhead (supraîncărcare)
- Reacţie prea rapidă produce oscilaţii
- Prea încet pentru a fi relevant

Rutare Adaptivă – Avantaje

1. Creşterea performanţei

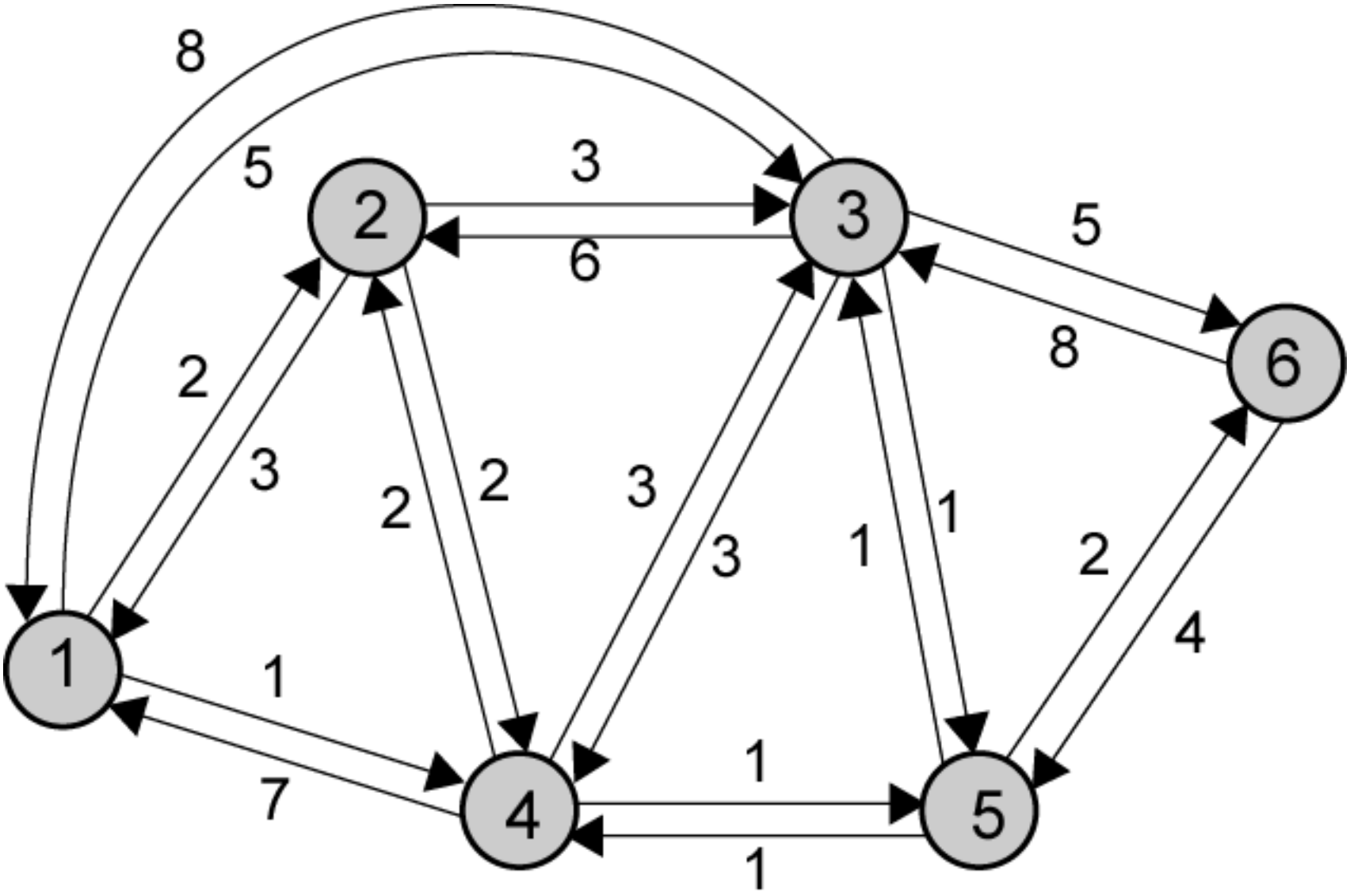
2. Ajută la controlul congestiei

3. Sistem Complex
   – Poate să nu ajungă la beneficiile teoretice

Criterii de selectie a rutelor

- Minimum hop count (numărul de salturi)
- Minimum cost

Retea



Algoritm de cost minim

- Decizia de rutare este data de:
  - Numar de hopuri(salturi)
  - Cost link invers proportional cu capacitatea
- Retea cu link-uri bidirectionale
- Cost asociat in ambele directii
- Costul caii de rutare = suma costurilor pe link
- Se cauta *calea de cost minim pentru toate perechile sursa si destinatie*
- Costurile pot fi diferite pe cele doua sensuri

Curs 8

Algoritmi de dirijare

1. **Algoritmul Dijkstra**

2. **Algoritmul Belmann-Ford**

Algoritm Dijkstra - definiţii

- *Caută calea cea mai scurtă de la un nod sursă la toate nodurile prin dezvoltarea de căi de lungime din ce în ce mai mare*

Notatii:

- N = multimea de noduri în reţea
- s = nod sursă
- T = multimea de noduri încorporate în algoritm
- w(i, j) = costul legaturii de la nod i la nod j
  - w(i, i) = 0
  - w(i, j) = ∞ dacă nu sunt direct conectate
  - w(i, j) $\geq$ 0 dacă sunt direct conectate
- L(n) = costul căii cunoscute cea mai ieftina de la nod s la nodul n

  - la terminare, L(n) este costul căii celei mai ieftine de la s la n

Algoritm Dijkstra - metodă

Pas 1 [Iniţializare]

- T = {s} Nod sursă
- L(n) = w(s, n) pentru n ≠ s
- Initial doar costuri link

Pas 2 [Următorul Nod]

- Caută nod vecin necuprins în T cu cost minim faţă de s
- Adaugare nod în T

Pas 3 [Actualizare cale cea mai ieftină]

- L(n) = min[L(n), L(x) + w(x, n)] pentru toţi n $\notin$ T
- Dacă al doilea termen este minimul, actualizare cale prin concatenare

[Terminare] Algoritmul se termină dupa adaugarea tuturor nodurilor

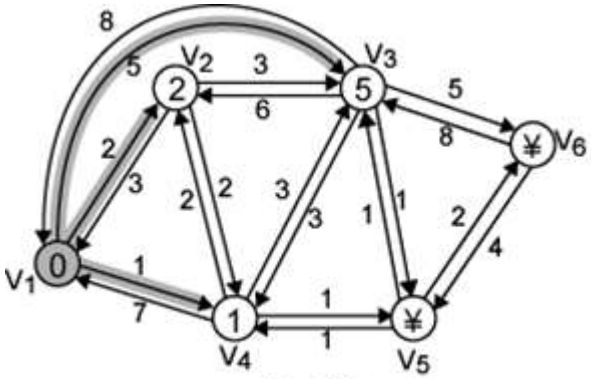
Algoritm Dijkstra - observaţii

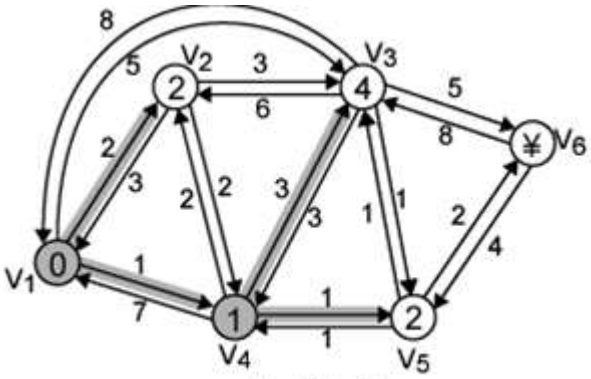
- La terminare, *valoarea L(x) asociată fiecărui nod x este costul (lungimea) căii de cost minim de la s la x.*

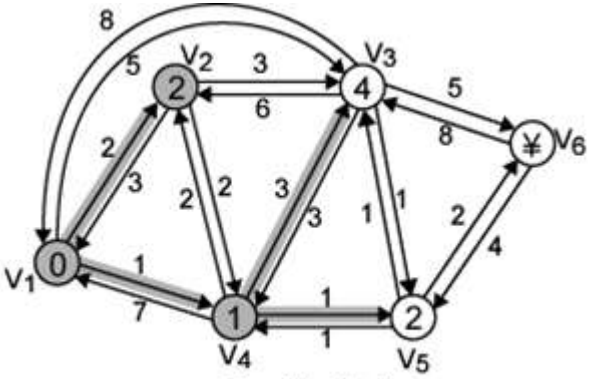
- *Multimea T defineşte calea cea mai ieftină de la s la fiecare alt nod*

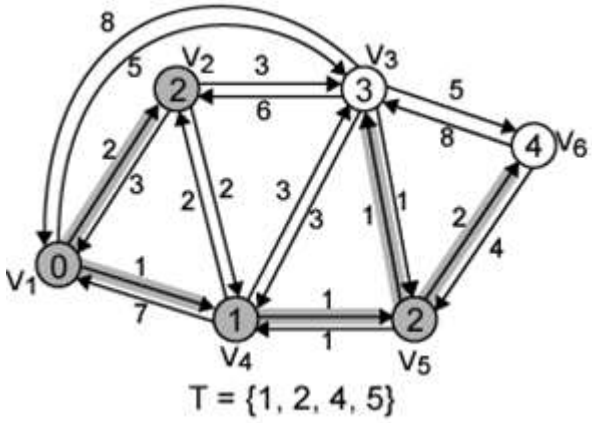
- O iteraţie a paşilor 2 şi 3 adaugă un nou nod în T

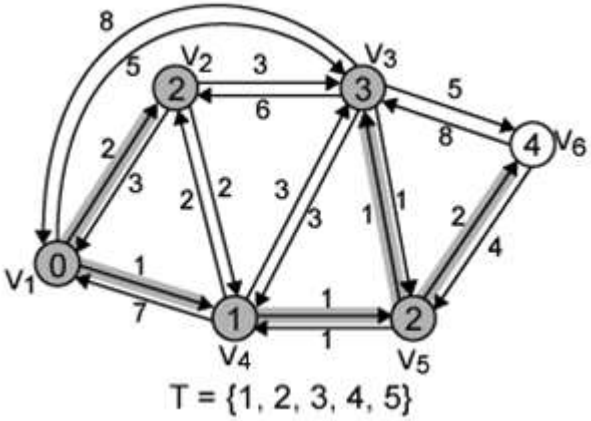
Algoritm Dijkstra - exemplu

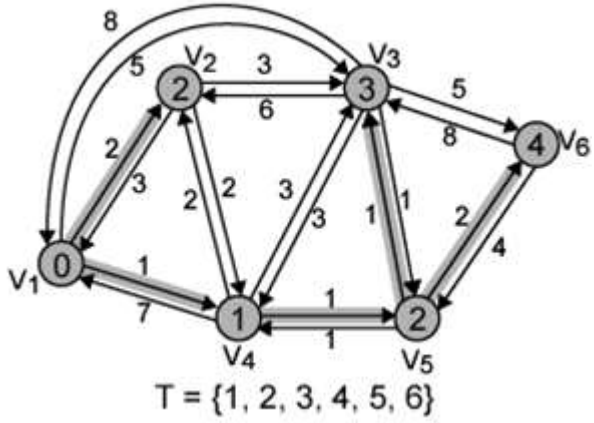


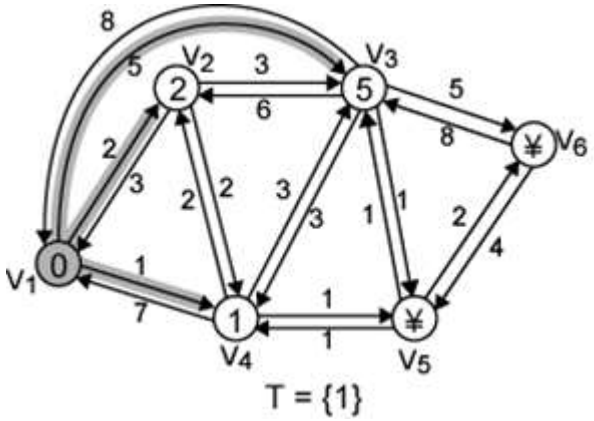


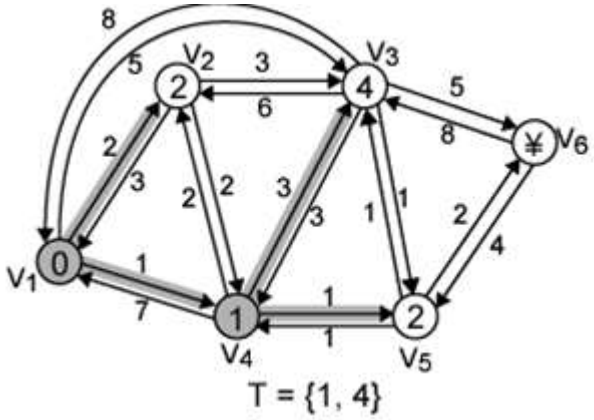






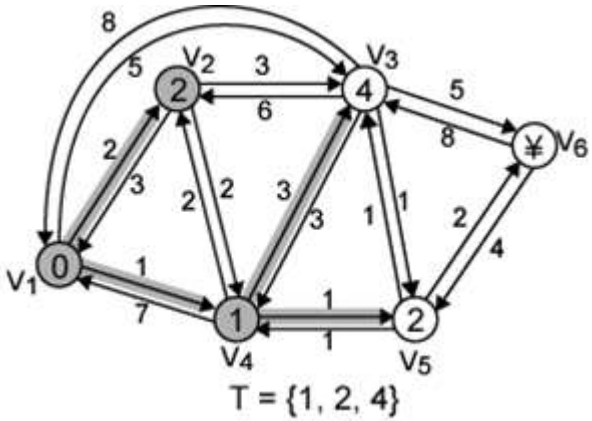


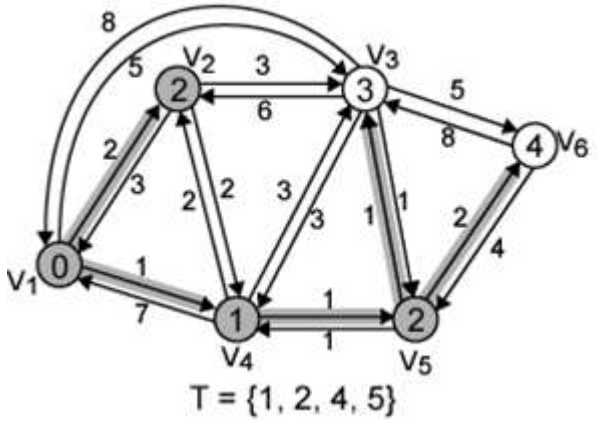




Iteratii	T	L(2)	Cale	L(3)	Cale	L(4)	Cale	L(5)	Cale	L(6)	Cale
1	{1}	2	1–2	5	1-3	1	1–4	∞	-	∞	-

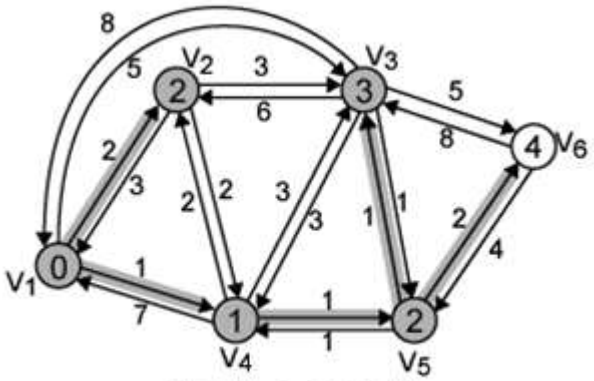
Iteratii	T	L(2)	Cale	L(3)	Cale	L(4)	Cale	L(5)	Cale	L(6)	Cale
2	{1,4}	2	1–2	4	1-4-3	1	1–4	2	1-4–5	∞	-

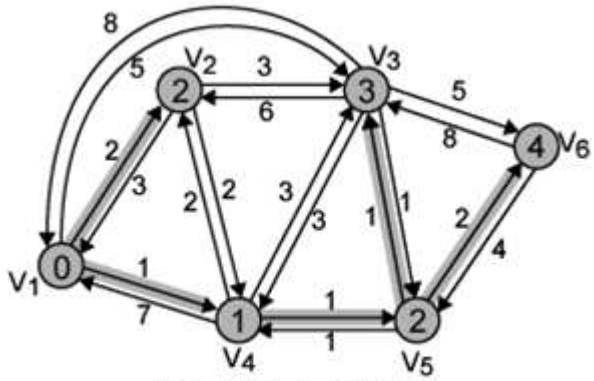




Iteratii	T	L(2)	Cale	L(3)	Cale	L(4)	Cale	L(5)	Cale	L(6)	Cale
3	{1, 2, 4}	2	1–2	4	1-4-3	1	1–4	2	1-4–5	∞	-

Iteratii	T	L(2)	Cale	L(3)	Cale	L(4)	Cale	L(5)	Cale	L(6)	Cale
4	{1, 2, 4, 5}	2	1–2	3	1-4-5–3	1	1–4	2	1-4–5	4	1-4-5–6





Iteratii	T	L(2)	Cale	L(3)	Cale	L(4)	Cale	L(5)	Cale	L(6)	Cale
5	{1, 2, 3, 4, 5}	2	1–2	3	1-4-5–3	1	1–4	2	1-4–5	4	1-4-5–6

Iteratii	T	L(2)	Cale	L(3)	Cale	L(4)	Cale	L(5)	Cale	L(6)	Cale
6	{1, 2, 3, 4, 5, 6}	2	1-2	3	1-4-5-3	1	1-4	2	1-4–5	4	1-4-5-6

Rezultate la Exemplu
Algoritm Dijkstra

Iteratii	T	L(2)	Cale	L(3)	Cale	L(4)	Cale	L(5)	Cale	L(6)	Cale
1	{1}	2	1–2	5	1-3	1	1–4	∞	-	∞	-
2	{1,4}	2	1–2	4	1-4-3	1	1–4	2	1-4–5	∞	-
3	{1, 2, 4}	2	1–2	4	1-4-3	1	1–4	2	1-4–5	∞	-
4	{1, 2, 4, 5}	2	1–2	3	1-4-5–3	1	1–4	2	1-4–5	4	1-4-5–6
5	{1, 2, 3, 4, 5}	2	1–2	3	1-4-5–3	1	1–4	2	1-4–5	4	1-4-5–6
6	{1, 2, 3, 4, 5, 6}	2	1-2	3	1-4-5-3	1	1-4	2	1-4–5	4	1-4-5-6

Curs 8

Algoritmi de dirijare

1. Algoritmul Dijkstra

2. Algoritmul Belmann-Ford

Algoritm Bellman-Ford. Definiţii

- Caută calea cea mai ieftină formată din cel mult un link
- Caută calea cea mai ieftină formată din cel mult două link-uri
- ș.a.m.d.

Notatii:

- s = nod sursă
- w(i, j) = cost link de la nod i la nod j
  - w(i, i) = 0
  - w(i, j) = ∞ dacă cele două noduri nu sunt conectate direct
  - w(i, j) $\geq$ 0 dacă cele două noduri sunt conectate direct

- h = număr maxim link-uri în cale în pasul curent al algoritmului
- $L_h(n)$ = costul căii cea mai ieftine de la s la n cu mai mult de h link-uri

Algoritm Bellman-Ford. Metodă

Pas 1 [Iniţializare]

- $L_0(n) = \infty$, pentru toţi $n \neq s$
- $L_h(s) = 0$, pentru toţi h

Pas 2 [Actualizare]

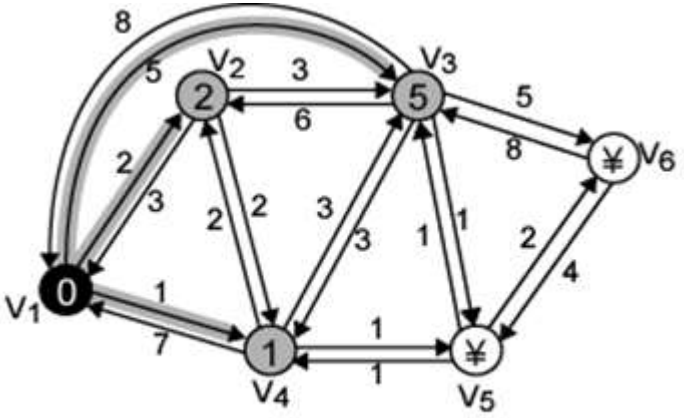
- Pentru fiecare $h \geq 0$ succesiv
- Pentru fiecare nod $n \neq s$, calculează
  - $L_{h+1}(n) = \min_j [L_h(j) + w(j,n)]$
- Conectează n cu nod predecesor j care atinge minimum
- *Elimină orice conexiune a unui nod n cu un nod predecesor diferit format în iteraţie anterioară*
- Calea de la s la n se termină cu link de la j la n

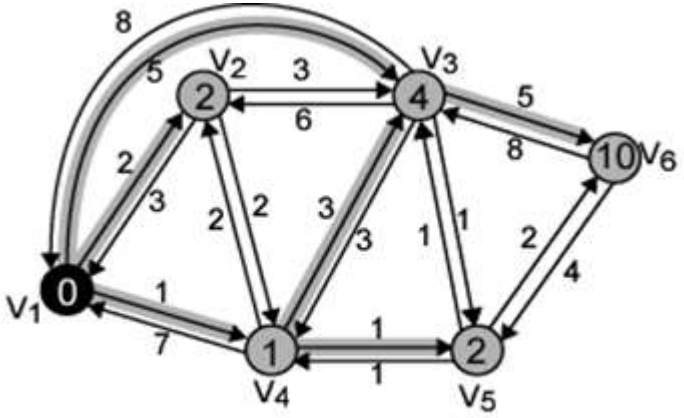
Observaţii Algoritm Bellman-Ford

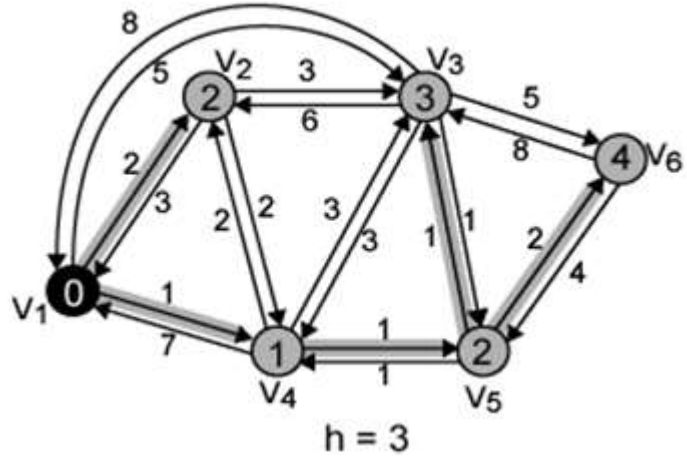
- *La fiecare iteraţie a pasului 2 pentru h=K şi fiecare nod destinaţie n, algoritmul compară căile de la s la n de lungime K=1 cu calea de la iteraţia anterioară*

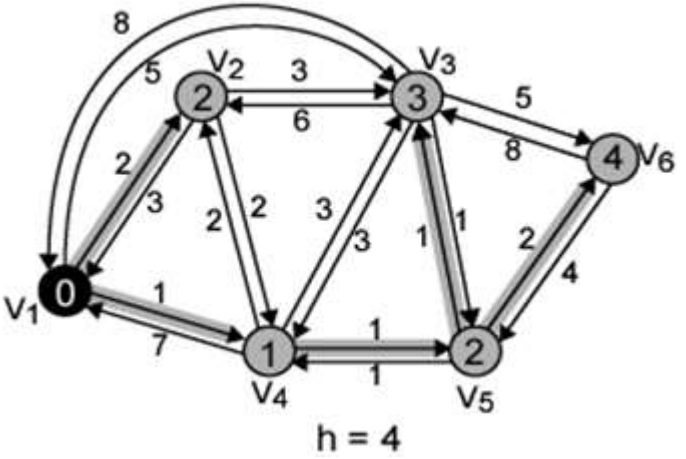
- În funcţie de cost se menţine calea anterioară sau se actualizează

Exemplu Algoritm Bellman-Ford

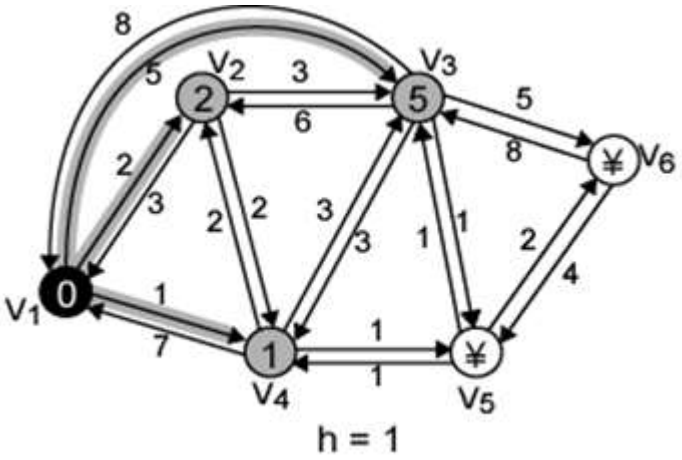






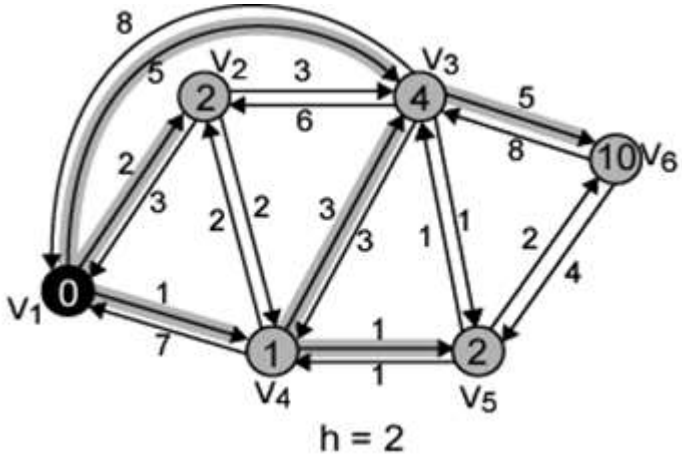


Exemplu Algoritm Bellman-Ford



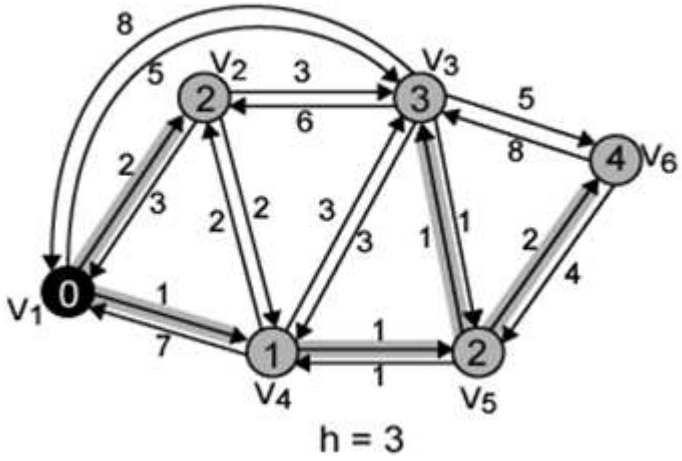
h	$L_h(2)$	Cale	$L_h(3)$	Cale	$L_h(4)$	Cale	$L_h(5)$	Cale	$L_h(6)$	Cale
0	∞	-	∞	-	∞	-	∞	-	∞	-

Exemplu Algoritm Bellman-Ford



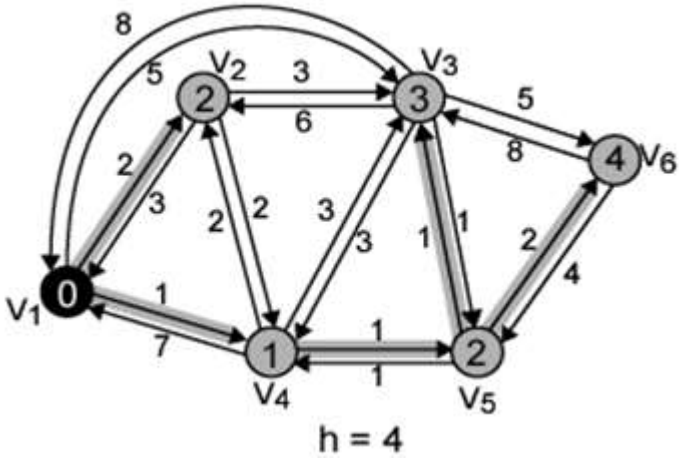
h	$L_h(2)$	Cale	$L_h(3)$	Cale	$L_h(4)$	Cale	$L_h(5)$	Cale	$L_h(6)$	Cale
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-

Exemplu Algoritm Bellman-Ford



h	$L_h(2)$	Cale	$L_h(3)$	Cale	$L_h(4)$	Cale	$L_h(5)$	Cale	$L_h(6)$	Cale
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6

Exemplu Algoritm Bellman-Ford



h	$L_h(2)$	Cale	$L_h(3)$	Cale	$L_h(4)$	Cale	$L_h(5)$	Cale	$L_h(6)$	Cale
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

Rezultate Exemplu Bellman-Ford

h	$L_h(2)$	Cale	$L_h(3)$	Cale	$L_h(4)$	Cale	$L_h(5)$	Cale	$L_h(6)$	Cale
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

Comparaţie Belmann-Ford cu Dijkstra

- Rezultatele celor doi algoritmi coincid
- Informaţii colectate:
  - Bellman-Ford
    - Calculul în nodul n necesită cunoaşterea costului link-ului la toţi vecinii plus totalul costului de la fiecare vecin la s
    - Fiecare nod poate menţine un set de costuri şi căi pentru fiecare alt nod
    - Face schimb de informaţii cu vecinii direcţi
    - Poate actualiza costurile şi căile pe baza informaţiilor de la vecini şi costurile linkurilor locale
  - Dijkstra
    - Fiecare nod are nevoie de topologia completă
    - Trebuie să cunoască costul tuturor linkurilor din reţea
    - Schimbă informaţii cu toate celelalte noduri

Bibliografie

1. Reţele de calculatoare, ANDREW S. TANENBAUM, ediţia a treia, 1998, Editura Computer Press Agora
2. Reţele de calculatoare, ANDREW S. TANENBAUM, ediţia a patra, 2000, Editura Computer Press Agora – *varianta electronică*
3. Reţele de calculatoare, Valentin Cristea, Nicolae Tapus, Trandafir Moisa, Valeriu Damian, 1992, Editura Teora.

Întrebări?