****

**Universitatea „Politehnica” din Bucureşti**

**Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei**

Algoritmi de rutare cu implicații în protocoale de rutare

**Student: DOBRE Nicoleta**

**Grupa: 441A**

**Bucuresti, 2016**

**Cuprins**

**1.** Introducere

**2.**Clasificarea algoritmilor de rutare

**3.** Algoritmi de rutare (prezentarea algoritmilor)

**4.** Măsurare de performanțe și comparație între performanțele algoritmilor de rutare/dirijare

**5.** Bibliografie

**1.Introducere**

**Rutarea** este un termen folosit în rețele de calculatoare pentru a desemna procesul de alegere a căii pe care un pachet este transmis de la sursă la destinație sau destinații, chiar și între două rețele diferite. Rutarea este bazată pe o tabelă care are în principal următoarele câmpuri: *adresa rețelei* (net address), *masca de rețea* (netmask), *adresa următorului rute* (next hop) și/sau *adresa interfeței de ieșire*.

Protocoalele de rutare determina regulile prin care ruterele schimba informaţii despre accesibilitatea reţelelor. În funcţie de informaţiile furnizate de aceste protocoale se construieşte tabela de rutare, iar pe baza tabelei de rutare este determinat traseul pe care trebuie trimis fiecare pachet.

Protocoalele de rutare, uneori denumite și protocoale de rutare dinamică, au drept obiectiv schimbarea informațiilor despre rețelele cunoscute între routerele ce rulează același protocol de rutare. Pe baza acestor informații se construiesc rutele dinamice. [1]

Routerele au câte o tabelă de rutare pentru fiecare protocol rutat. Pentru un protocol rutat dat informaţiile oferite de toate protocoalele de rutare de găsesc într-o singură tabela. Prin urmare, aceeaşi tabelă de rrutare va conţine rutele directe conectate,rutele statice şi cele dinamice.

Un router poate rula una sau mai multe protocoale de rutare, numărul protocoalelor de rutare ce pot fi rulate fiind limitate în general de sistemul de operare sau de modelul routerului. Un router Cisco, de exemplu, rulează în general până la 30 instante de protocoale de rutare.

***Distanţa administrativă*** este un număr între 0 şi 255, asociat cu un tip de ruta sau cu un protocol de rutare, ce permite ierarhizarea protocoalelor de rutare.

***Metrică unei rute*** este un număr, rezultat din aprecierea calităţii unui drum spre o anumită destinaţie în raport cu un set de criterii. Metrică şi setul de criterii sunt relevane pentru un anumit protocol de rutare. Prin urmare, nu are sens compararea metricilor unor rute obţinute prin protocoale de rutare diferite.

Figura 1 - Funcționarea unui ruter

**Caracteristicile protocoalelor de rutare [2]**

Protocoalele de rutare pot fi comparate pe baza următoarelor caracteristici:

* *Timpul de convergență* - Timpul de convergență definește cât de repede routerele din topologia de rețea partajează informația de rutare și ajung la o stare de cunoștințe coerenta, despre topologia rețelei. Cu cât este mai rapidă convergența, cu atât este mai preferabil protocolul. Buclele de rutare pot apărea când tabelele de rutare inconsecvente nu sunt actualizate, datorită convergenței lente într-o rețea cu modificări de topologie.
* *Scalabilitatea* - Scalabilitatea definește mărimea rețelei bazată pe protocolul de rutare implementat. Cu cât mai mare este rețeaua, cu atât mai scalabil trebuie să fie protocolul de rutare.
* *Classless* (Utilizarea VLSM) și *Classfull* – protocoalele de rutare de tip **classless** includ masca de rețea în mesajele de actualizare. Această caracteristică suportă opțiunea**Variable Length Subnet Masking** (VLSM) și o sumarizare mai optimală a rutelor. Protocoalele de rutare **classfull** nu includ masca de subrețea și nu suportă VLSM.
* *Utilizarea resurselor* - utilizarea resurselor include cerințele unui protocol de rutare, cum ar fi spațiul de memorie, utilizarea procesorului, și link-ul de utilizare a lățimii de bandă. Cerințele mai înalte de resurse, necesită echipament mai puternic, pentru a suporta operarea protocolului de rutare în plus față de procesele de transmitere a pachetelor.
* *Implementarea și întreținerea* – Implementarea și întreținerea descrie nivelul de cunoștințe care sunt necesare pentru un administrator de rețea pentru a pune în aplicare și a întreține rețeaua bazată pe protocolul de rutare utilizat.

|  |  |
| --- | --- |
| **Avantaje** | **Dezavantaje** |
| ***Implementare și întreținere simplă*** - Nivelul de cunoștințe necesar pentru a implementa și a menține o rețea, cu un protocol bazat pe vector de distanță nu este mare. | ***Convergență lentă*** – utilizarea actualizărilor periodice pot provoca convergență lentă. Chiar dacă sunt folosite tehnici avansate cum ar fi actualizările cu declanșator, convergența globală este mai lentă în comparație cu protocoalele link state. |
| ***Cerințe de consum reduse*** - protocoalele de rutare bazate pe vectori de distanță nu au nevoie de resurse mari de memorie pentru a stoca informații. Nici nu au nevoie de un procesor puternic. În funcție de dimensiunea rețelei | ***Scalabilitate limitată*** – convergența lentă poate limita mărimea rețelei, fiindcă rețele mari necesită mai mult timp pentru a propaga informația de rutare. |
|  | ***Bucle de rutare*** – buclele de rutare pot apărea când tabelele de rutare, în mod inconsecvent, nu sunt actualizate, datorită convergenței lente dintr-o rețea. |

**2.Clasificarea algoritmilor de rutare**

Algoritmii de rutare se pot clasifica după cum urmează:

1. Neadaptivi (statici) - alegerea căii se calculează în avans (of line) şi parvine ruterului la iniţializarea reţelei

* + Shortest Path Algorithm (Algoritmul de dirijare cu cea mai scurtă cale)
	+ Flooding (Inundare)

2. Adaptivi (dinamici) - îşi modifică deciziile de dirijare pentru a reflecta modificările de topologie şi pe cele de trafic

* + Distance Vector Routing (Algoritm de dirijare cu vector distanță)
	+ Link State Routing (Rutarea folosind starea legăturii)

Algoritmii adaptivi diferă prin:

a. locul de unde îşi iau informaţia: local, de la un vecin, de la toate ruterele

b. momentul în care schimbă rutele: când se schimbă încărcarea, când se schimbă topologia, la ΔT secunde

c. metrica folosită pentru optimizare: distanţa, timpul de tranzit, numărul de salturi

**3. Algoritmi de rutare (prezentarea algoritmilor)**

***3.1* Shortest Path Algorithm (Algoritmul de dirijare cu cea mai scurtă cale)**

Algoritmul de dirijare cu vectori distanţă (distance vector routing) presupune că fiecare ruter menţine o tabelă (de exemplu un vector) care păstrează cea mai bună distanţă cunoscută spre fiecare destinaţie şi linia care trebuie urmată pentru a ajunge acolo.

Ideea este de a construi un graf al subreţelei, fiecare nod al grafului fiind un router, iar fiecare arc al grafului fiind o linie de comunicaţie (numită adesea legătură). Pentru a alege o cale între o pereche dată de rutere, algoritmul trebuie să găsească în graf calea cea mai scurtă dintre ele. Conceptul de cea mai scurtă cale (shortest path routing) necesită unele explicaţii. O modalitate de a măsura lungimea căii este numărul de salturi. Folosind această metrică, căile ABC şi ABE din Figura 2 sunt la fel de lungi. O altă metrică este distanţa geografică în kilometri, caz în care ABC este clar mult mai mare decât ABE. Oricum, sunt posibile multe alte metrici în afară de salturi şi distanţa geografică. De exemplu, fiecare arc poate fi etichetat cu valorile medii ale aşteptării în coadă şi întârzierii de transmisie pentru anumite pachete standard de test, aşa cum sunt determinate de măsurători care se fac din oră în oră. Cu această etichetare, cea mai scurtă cale este cea mai rapidă, nu neapărat cea cu mai puţine arce sau kilometri. În cazul cel mai general, etichetele de pe arce ar putea fi calculate ca funcţii de distanţă, lărgime de bandă, trafic mediu, cost al comunicaţiei, lungime medie a cozilor de aşteptare, întârzieri măsurate şi alţi factori. Prin modificarea ponderilor, algoritmul ar putea calcula cea mai „scurtă” cale, în conformitate cu oricare dintre aceste criterii sau cu combinaţii ale acestor criterii.



Figura 2 - Primii cinci pasi folositi in calcularea celei mai scurte cai de la A la D. Sagetile indica nodul curent

Se cunosc mai mulţi algoritmi pentru calculul celei mai scurte căi între două noduri dintr-un graf. Cel mai cunoscut este cel propus de Dijkstra (1959). Fiecare nod este etichetat (în paranteze) cu distanţa de la nodul sursă până la el, de-a lungul celei mai bune căi cunoscute. Iniţial nu se cunoaşte nici o cale, aşa că toate nodurile vor fi etichetate cu infinit. Pe măsură ce se execută algoritmul şi se găsesc noi căi, etichetele se pot schimba, reflectând căi mai bune. O etichetă poate fi fie temporară, fie permanentă. Iniţial toate etichetele sunt temporare. Atunci când se descoperă că o etichetă reprezintă cea mai scurtă cale posibilă de la sursă către acel nod, ea devine permanentă şi nu se mai schimbă ulterior. Pentru a ilustra cum funcţionează algoritmul de etichetare, se urmăreşte graful neorientat, etichetat din Figura 2(a), unde etichetele reprezintă, de exemplu, distanţa. Se doreşte să se afle cea mai scurtă cale de la A la D. Se începe prin a marca nodul A ca permanent, indicând aceasta printr-un cerc colorat. Apoi se examinează fiecare nod adiacent cu A (care este acum nodul curent), reetichetând fiecare nod cu distanţa până la nodul A. De fiecare dată când un nod este reetichetat, se va eticheta şi cu nodul de la care s-a făcut încercarea, pentru a putea reface calea ulterior. După ce s-a examinat toate nodurile adiacente ale lui A, se examinează toate nodurile cu etichetă temporară din întregul graf şi se face permanent pe cel cu eticheta minimă, aşa cum se observă din Figura 2(b). Acest nod devine noul nod curent. Acum se începe din B şi se examinează toate nodurile sale adiacente. Dacă suma între eticheta lui B şi distanţa de la B la nodul considerat este mai mică decât eticheta acelui nod, înseamnă că am găsit o cale mai scurtă şi va trebui făcută reetichetarea nodului. După ce toate nodurile adiacente nodului curent au fost inspectate şi au fost schimbate toate etichetele temporare posibile, se reia căutarea în întregul graf pentru a identifica nodul cu eticheta temporară minimă. Acest nod este făcut permanent şi devine nodul curent al etapei următoare. Figura 2 prezintă primii cinci paşi ai algoritmului. Pentru a vedea de ce merge algoritmul, trebuie urmărită Figura 2 (c). La momentul respectiv de abia s-a făcut permanent nodul E. Se presupune că ar exista o cale mai scurtă decât ABE, de exemplu AXYZE. Există două posibilităţi: fie nodul Z a fost deja făcut permanent, fie încă nu a fost. Dacă a fost, atunci E a fost deja examinat (la pasul imediat următor celui la care Z a fost făcut permanent), astfel încât calea AXYZE nu a fost ignorată şi deci nu poate fi cea mai scurtă cale. Se consideră acum cazul în care Z este încă etichetă temporară. Atunci fie eticheta lui Z este mai mare sau egală cu cea a lui E, caz în care ABE nu poate fi o cale mai scurtă decât AXYZE, fie este mai mică decât cea a lui E, caz în care Z şi nu E va deveni permanent mai întâi, permiţând lui E să fie examinat din Z.

***3.2* Flooding (Inundare)**

„Flooding” este un algoritm de rutare simplu în care fiecare pachet recepţionat de un ruter este trimis prin fiecare legătură mai puţin cea de la care a fost recepţionat. Astfel, fiecare mesaj este livrat către toate nodurile reţelei. Există mai multe variante ale acestui algoritm.

Fiecare nod are rol de transmiţător şi receptor. Fiecare nod încearcă să trimită mai departe fiecare mesaj primit către fiecare vecin.

Deşi nu constituie o variantă eficientă de transmitere a pachetelor în reţea, este un algoritm destul de utilizat ce facilitează descoperirea de rute şi topologii.

Avantajul este dat de faptul că pachetele vor fi, cu siguranţă, livrate, din moment ce se vor utiliza toate căile disponibile în reţea.

Dezavantajul este acela că se iroseşte multă bandă şi poate periclita fiabilitatea unei reţele, în cazul, spre exemplu, a unui ping flood sau atac de tip DoS(Denial of Service).





Figura 3 si 4 – Algoritmul de inundare

***3.3* Distance Vector Routing (Algoritm de dirijare cu vector distanță)**

Algoritmul de dirijare cu vectori distanţă (distance vector routing) presupune că fiecare ruter menţine o tabelă (de exemplu un vector) care păstrează cea mai bună distanţă cunoscută spre fiecare destinaţie şi linia care trebuie urmată pentru a ajunge acolo. Aceste tabele sunt actualizate prin schimbul de informaţii între nodurile vecine. Algoritmul de dirijare cu vectori distanţă este cunoscut şi sub alte nume, cel mai des algoritmul distribuit de dirijare Bellman-Ford şi algoritmul Ford-Fulkerson, după numele cercetătorilor care l-au propus (Bellman, 1957; şi Ford şi Fulkerson, 1962). A fost algoritmul de dirijare folosit iniţial în reţeaua ARPANET, a fost folosit de asemenea în Internet sub numele de RIP.

Sunt o categorie de algoritmi care presupun construirea si actualizarea automata a unei tabele locale de rutare de catre echipamentele din retea.

Astfel, oricare nod poseda o tabela cu distantele(costurile) catre toate celelalte noduri si distribuie acest „vector” tuturor vecinilor sai.

Se fac urmatoarele presupuneri:

* Fiecare nod cunoaste costul leagaturilor cu nodurile adiacente
* O legatura cazuta are un cost infinit



Figura 5 – (a) o subretea. (b) Intrări de la A, I, H şi K şi noua tabelă de dirijare pentru J



Figura 6 - Exemplu

|  |  |
| --- | --- |
| **Informatia stocata in nod** | **Distanta catre fiecare nod** |
| **A** | **B** | **C** | **D** | **E** | **F** | **G** |
| **A** | 0 | 1 | 1 | 2 | 1 | 1 | 2 |
| **B** | 1 | 0 | 1 | 2 | 2 | 2 | 3 |
| **C** | 1 | 1 | 0 | 1 | 2 | 2 | 2 |
| **D** |  | 2 | 1 | 0 | 3 | 2 | 1 |
| **E** | 1 | 2 | 2 | 3 | 0 | 2 | 3 |
| **F** | 1 | 2 | 2 | 2 | 2 | 0 | 1 |
| **G** |  | 3 | 2 | 1 | 3 | 1 | 0 |

Figura 7 - Distanțele finale stocate in fiecare nod (imagine de ansamblu)

*3.4* **Link State Routing (Rutarea folosind starea legaturii)**

Variante de dirijare folosind starea legăturilor sunt actualmente foarte răspândite. Ideea algoritmului bazat pe starea legăturilor este simplă şi poate fi formulată în 5 puncte. Fiecare ruter trebuie să facă următoarele:

1. Să descopere care sunt vecinii săi şi afle adresele de reţea ale acestora.

2. Să măsoare întârzierea sau costul până la fiecare din vecinii săi.

3. Să pregătească un pachet prin care anunţă pe toată lumea că tocmai a terminat de cules datele despre vecini.

4. Să trimită acest pachet către toate celelalte rutere.

5. Să calculeze cea mai scurtă cale spre fiecare ruter.

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

**4. Masurare de performanțe și comparație între performanțele algoritmilor de rutare/dirijare**

• **DV (Distance Vector Routing)**

– Transmit întreaga tabela de rutare

– Actualizări periodice

– Convergentă greoaie

– Putin scalabile

+ Folosesc mai puţine resurse

+ Transmit informaţii la vecini

+ Sunt mai uşor de configurat

• **LS (Link State Routing)**

– Cerinţe mai mari de hardware

– Transmit informaţii în întreaga reţea (porţiuni din tabela de rutare)

+ Imagine de ansamblu a reţelei

+ Actualizări determinate de schimbări în topologie

+ Mai puţin predispuse la bucle

+ Convergenta rapidă

~ În privinţa consumului de bandă, protocoalele LS vor trimite o copie a întregi tabele în toată reţeaua în etapa de stabilire a adiacentei

~ LS va folosi mai multă bandă decât DV în etapa iniţială dar pentru un interval de timp semnificativ de funcţionare DV va consuma mai multă bandă

~ pentru o reţea foarte instablila performanţele unui protocol DV pot fi superioare unui protocol LS

**5. Bibliografie**

[1] Ravan Rughinis, Razvan Deaconescu, Andrei Cioaba, Bogran Doinea – "Retele locale"

[2]https://ro.wikipedia.org/wiki/Rutare\_dinamic%C4%83\_(adaptiv%C4%83)

[3] Curs "Retele de calculatoare" - Universitatea “Constatin Brâncuşi” din Târgu-Jiu

[4]-Andrew Tanenbaum – "Retele de calculatoare" editia IV, Ed. Byblos 2003

[5]-Andrew Tanenbaum – "Computer Networks" editia V, Ed. Prentice Hall 2011