Capitolul 4: Protocoale de rutare Distance Vector

Protocoalele Distance Vector (1)

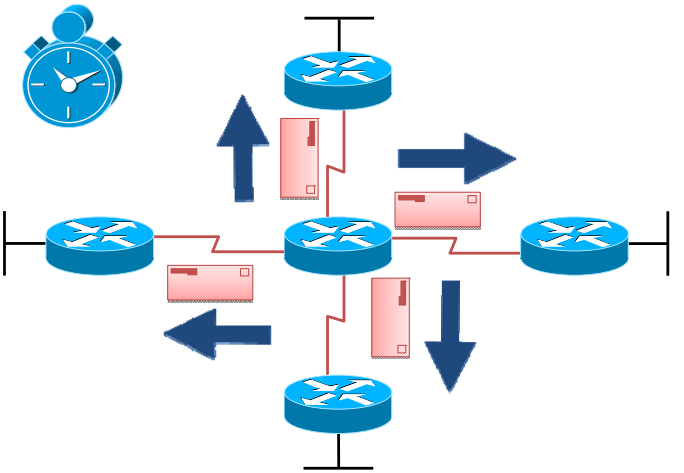
- Ce implică ideea de Distance Vector ?
  - rutele sunt reținute și propagate sub forma unui vector
  - fiecare intrare reține rețeaua destinație, direcția către ea și distanța până la ea
- Un ruter care folosește un protocol Distance Vector
  - nu cunoaște toată calea pe care va fi transmis un pachet
  - cunoaște următorul "hop"
  - cunoaște distanța până la rețea

Un protocol distance vector își reține rutele în forma unui „vector de distanță și de direcție". Distanța, în acest context, se definește cu ajutorul unei metrici. Această metrică poate fi reprezentată, de exemplu, de numărul de hop-uri (protocolul RIP) sau se poate referi la bandwidth, delay, relability, load, MTU (protocolul EIGRP). Direcția reprezintă următorul hop, sau interfața pe care ruterul trimite pachete către o anumită rețea.

Dacă un ruter folosește un protocol distance vector, acesta va cunoaște direcția pe care o va lua un pachet și distanța până la destinație. Totuși, ruterul nu va ști întreaga cale către rețeaua destinație.

Protocoalele Distance Vector (2)

- Efectuează update-uri periodice prin broadcast sau multicast



Protocoalele distance vector prezintă un set de caracteristici comune:

Sunt trimise update-uri periodice către vecini, la intervale fixe, chiar dacă topologia rețelei nu s-a schimbat pe parcursul unui anumit interval de timp.

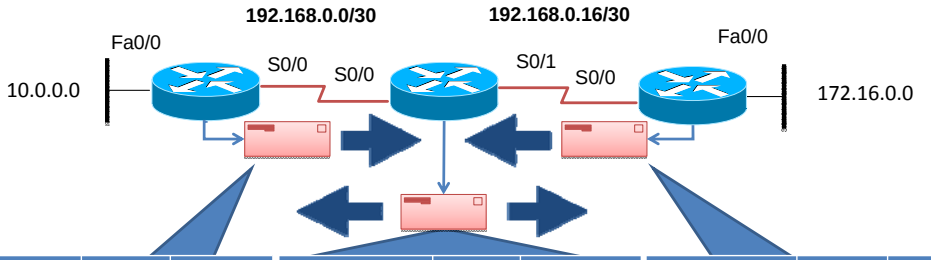
Vecinii unui ruter sunt acele rutere care sunt direct conectate la acesta și care folosesc același protocol de rutare.

Update-urile sunt trimise prin broadcast. Ruterele vecine vor procesa aceste update-uri. Toate celelalte echipamente din rețea vor decapsula aceste pachete până la nivelul 3, după care le vor arunca.

Update-uri cu tabela de rutare vor fi trimise de protocoalele distance vector, cu câteva excepții (EIGRP), către vecinii săi. Vecinii care primesc aceste update-uri vor procesa întregul pachet pentru a găsi informații pertinente și vor arunca restul datelor.

Protocoalele Distance Vector (3)

- Orice ruter își cunoaște doar vecinii direcți
- Se trimite întreaga tabelă de rutare vecinilor



Rețea	Interfață	Număr de hopuri
10.0.0.0	Fa0/0	0
192.168.0.0	S0/0	0

Rețea	Interfață	Număr de hopuri
192.168.0.0	S0/0	0
192.168.0.16	S0/1	0

Rețea	Interfață	Număr de hopuri
172.16.0.0	Fa0/0	0
192.168.0.16	S0/0	0

În acest exemplu, cele 3 rutere își trimit tabela de rutare numai către vecinii lor. Se observă că primul ruter trimite un pachet către al doilea ruter, dar nu și către al treilea ruter, în timp ce ruterul 2 trimite pachete ruterului 1 și 3.

Un protocol distance vector va folosi un algoritm specific pentru instalarea rutelor în tabela de rutare, pentru trimiterea de update-uri vecinilor și pentru luarea deciziilor de rutare. Acesta va avea definite următoarele mecanisme:

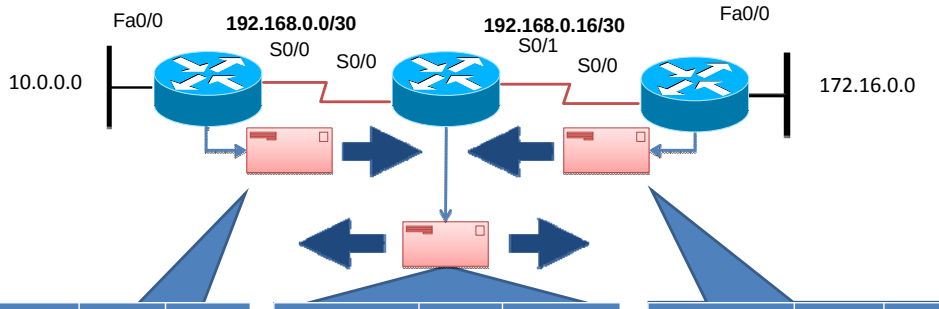
•Mecanism pentru primirea și trimiterea informației de rutare

•Mecanism pentru calcularea celor mai eficiente rute și instalarea lor în tabela de rutare

•Mecanism pentru detectarea și adaptarea la schimbări în topologia rețelei

Protocoalele Distance Vector (4)

- Tabela de rutare după primul update



Rețea	Interfață	Număr de hopuri
10.0.0.0	Fa0/0	0
192.168.0.0	S0/0	0
192.168.0.16	S0/0	1

Rețea	Interfață	Număr de hopuri
192.168.0.0	S0/0	0
192.168.0.16	S0/1	0
10.0.0.0	S0/0	1
172.16.0.0	S0/1	1

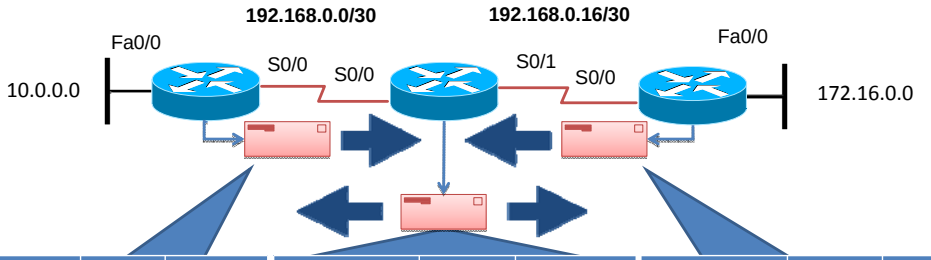
Rețea	Interfață	Număr de hopuri
172.16.0.0	Fa0/0	0
192.168.0.16	S0/0	0
192.168.0.0	S0/0	1

În cazul de față, cele trei rutere primesc informații despre cel puțin o rețea anterior necunoscută. Fiecare ruter va analiza independent update-ul primit, va calcula cea mai scurtă cale către noile adrese de rețea, iar apoi acestea vor fi adăugate sau actualizate în tabela de rutare.

Atunci când un ruter primește un update care conține întreaga tabelă de rutare a unui vecin, va păstra numai partea din pachet care îi aduce informații noi, iar pe cealaltă o va arunca. Astfel, ruterul 2 trimite către ruterul 1 ambele adrese aflate în tabela sa de rutare (192.168.0.0, 192.168.0.16), dar ruterul 1 instalează numai ruta pe care nu o cunoștea (192.168.0.16). La primirea unei rute deja existente în tabela de rutare, dar cu o metrică mai bună, vechea rută va fi actualizată conform noilor informații primite.

Protocoalele Distance Vector (5)

- Tabela de rutare după al doilea update
- Putem observa că rețeaua a convers



Rețea	Interfață	Număr de hopuri
10.0.0.0	Fa0/0	0
192.168.0.0	S0/0	0
192.168.0.16	S0/0	1
172.16.0.0	S0/0	2

Rețea	Interfață	Număr de hopuri
192.168.0.0	S0/0	0
192.168.0.16	S0/1	0
10.0.0.0	S0/0	1
172.16.0.0	S0/1	1

Rețea	Interfață	Număr de hopuri
172.16.0.0	Fa0/0	0
192.168.0.16	S0/0	0
192.168.0.0	S0/0	1
10.0.0.0	S0/0	2

După al treilea update, fiecare ruter din exemplu va avea informații complete despre topologie. Convergența este realizată atunci când fiecare ruter are o imagine completă și corectă asupra întregii topologii. Practic informațiile tuturor ruterelor despre rețele oferă posibilitatea existenței unei conexiuni cu orice destinație din cadrul domeniului de rutare.

Să presupunem că la un moment dat, din diferite motive, rețeaua 172.16.0.0 devine inaccesibilă. În această situație ruterul 3 va trimite un update provocat (triggered update) către vecinul său, ruterul 2, care va scoate rețeaua din tabela sa de rutare. La rândul său, al doilea ruter va trimite un update către ruterul 1, care va șterge și el adresa respectivă din tabela sa proprie de rutare.

Protocoalele Distance Vector (6)

- Avantaje
  - configurare și întreținere simplă
  - consumul de resurse al ruterului este redus
- Dezavantaje
  - timp de convergență ridicat
  - scalabilitate limitată
  - pot apărea bucle de rutare

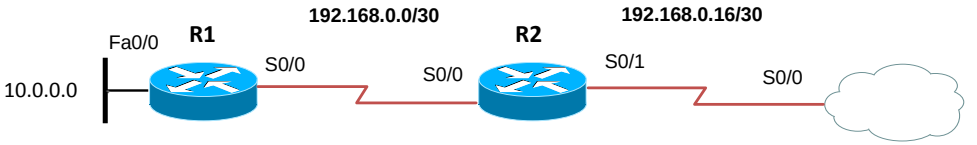
Datorită simplității configurării unui protocol de rutare distance vector, nivelul de cunoștințe al unui administrator necesar pentru implementarea și întreținerea ulterioară a protocolului de rutare distance vector într-o rețea este redus.

Complexitatea redusă a algoritmilor utilizați de protocoalele de rutare distance vector, dar și informațiile nu foarte detaliate despre rutele schimbate între vecini nu necesită o cantitate mare de memorie și nici o putere de procesare sporită. În cazul folosirii unor echipamente cu resurse reduse, protocoalele distance vector reprezintă alegerea ideală pentru dirijarea pachetelor în mod dinamic în rețea.

Timpul de convergență al protocolelor de rutare distance vector este ridicat datorită update-urilor de rutare trimise la un anumit interval de timp definit, ceea ce implică o scalabilitate redusă, un număr mare de echipamente crescând posibilitatea apariției bulelor de rutare.

Bucle de rutare (1)

- Exemplu: Problema “count-to-infinity”



Rețea	Interfață	Număr de hopuri
10.0.0.0	Fa0/0	0
192.168.0.0	S0/0	0
192.168.0.16	S0/0	1

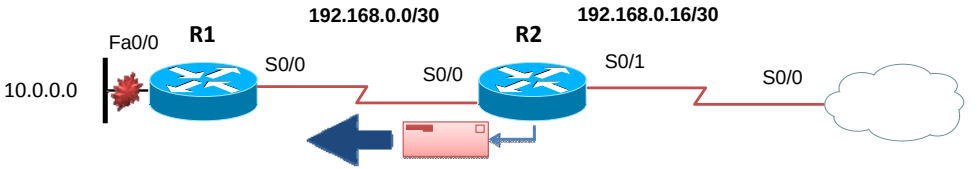
Rețea	Interfață	Număr de hopuri
192.168.0.0	S0/0	0
192.168.0.16	S0/1	0
10.0.0.0	S0/0	1

Problema „count to infinty” apare în momentul în care între rutere circulă update-uri eronate despre o rețea indisponibilă.

Să presupunem, în topologia de mai sus în care s-a realizat convergența, că rețeaua 10.0.0.0 devine indisponibilă. În același timp, R2 îi trimite un update lui R1 cu informația că are o rută către 10.0.0.0 cu metrica 1, cunoscând faptul că are o rută în tabela de rutare către 10.0.0.0 prin ruterul vecin R1. În această situație, R1 introduce în tabela sa proprie de rutare o rută către 10.0.0.0 cu metrica 2. După un timp, R1 trimite update la R2 conținând toate rutele sale. R2 vede că ruta către 10.0.0.0 nu mai are metrica 1, ci 2, așa că va mări metrica la 3. Când R2 va trimite update la R1, acesta va vedea că metrica rutei către 10.0.0.0 a crescut, așa că va incrementa metrica în tabela sa de rutare. Acest proces se poate repeta la infinit, generând problema „count to infinty”.

Bucle de rutare (2)

- Rețeaua 10.0.0.0 devine inaccesibilă, simultan R2 trimite un update lui R1



Rețea	Interfață	Număr de hopuri
~~10.0.0.0~~	~~Fa0/0~~	~~0~~
192.168.0.0	S0/0	0
192.168.0.16	S0/0	1

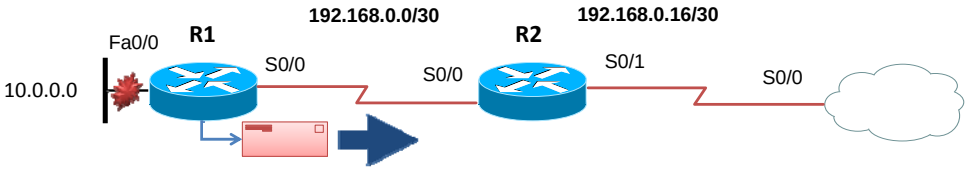
Rețea	Interfață	Număr de hopuri
192.168.0.0	S0/0	0
192.168.0.16	S0/1	0
10.0.0.0	S0/0	1

O buclă de rutare este o situație în care un pachet este transmis în mod continuu între o serie de rutere, fără a ajunge la destinația dorită. Acest lucru se întâmplă atunci când anumite rutere din rețea au informații incorecte despre o cale care apare ca fiind validă, către o destinație care, în mod real, nu există.

Un generator tipic pentru o astfel de buclă este exemplul de mai sus. După cum se poate observa, rețeaua 10.0.0.0 devine inaccesibilă, dar înainte ca R1 sa trimită un mesaj cu această informație, primește un update de la R2 care are o rută către 10.0.0.0. Acesta este o altă situație când apare problema „count to infinity”.

Bucle de rutare (3)

- R1 reintroduce rețeaua în tabela de rutare și trimite și el update



Rețea	Interfață	Număr de hopuri
10.0.0.0	S0/0	2
192.168.0.0	S0/0	0
192.168.0.16	S0/0	1

Rețea	Interfață	Număr de hopuri
192.168.0.0	S0/0	0
192.168.0.16	S0/1	0
10.0.0.0	S0/0	1

Deoarece ruterul nu cunoaște topologia rețelei, va presupune că informația primită de la R2 este corectă și va adăuga în tabela de rutare adresa rețelei 10.0.0.0, cu hop-ul 2. În acest moment, un pachet trimis de pe R1 cu destinația din rețeaua 10.0.0.0 va ajunge la R2. Mai departe, R2 cunoaște ca next-hop pentru rețeaua 10.0.0.0 ruterul R1. Astfel s-a format o buclă de rutare, pachetul circulând între ruterele R1 și R2 până va fi aruncat când valoarea TTL va ajunge 0.

Buclele de rutare pot apărea în diferite circumstanțe:

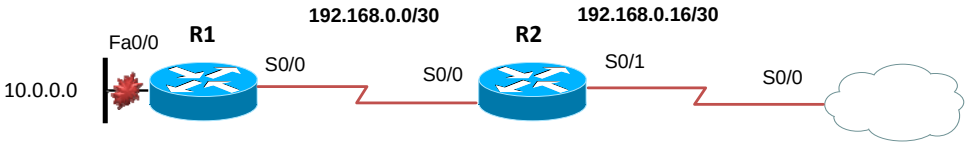
•Rute statice configurate incorect

•Redistribuire de rute configurate incorect (redistribuția este un proces prin care pot fi transmise informații introduse manual sau învățate prin intermediul altui protocol de rutare)

•Tabele de rutare inconsistente, datorită unui timp de convergență crescut

Bucle de rutare (4)

- Bucla se repetă la infinit



Rețea	Interfață	Număr de hopuri
10.0.0.0	S0/0	2
192.168.0.0	S0/0	0
192.168.0.16	S0/0	1

Rețea	Interfață	Număr de hopuri
192.168.0.0	S0/0	0
192.168.0.16	S0/1	0
10.0.0.0	S0/0	3

O buclă de rutare se poate dovedi extrem de dăunătoare pentru o rețea, ducând la performanțe scăzute sau chiar la blocarea rețelei. Printre efectele unei bucle de rutare se numără:

•Utilizarea excesivă a bandwith-ului care poate determina congestionarea rețelei datorită fluxului de pachete care ciclează într-o buclă de rutare

•Procesorul ruter-ului va fi suprasolicitat datorită procesării numărului mare de operații într-un interval scăzut de timp

•Poate fi afectată convergența rețelei, ducând la rutare suboptimală sau la pierderi de pachete

•Update-urile periodice se pot pierde sau pot fi întârziate, generând astfel mai multe bucle de rutare

Bucle de rutare (5)

- Apar datorită cunoașterii reduse a rețelei și a convergenței lente
- Cum le prevenim ?
  - setarea unei valori maxime a metricii unei rute
  - hold-down timer (ruterele sunt instruite să ignore update-uri despre o anumită rută un interval de timp)

Protocoalele distance vector sunt foarte simple în ceea ce priveste operațiile efectuate. Totuși, această simplicitate prezintă dezavantaje precum buclele de rutare. O buclă de rutare poate fi foarte dăunătoare unei rețele, ocupând excesiv bandwitdh și resurse. Există o serie de mecanisme pentru prevenirea buclelor de rutare:

•Setarea unei valori maxime a metricii unei rute, prevenind astfel problema „cout to infinity” (ex.: RIP poate avea metrica maxim 15)

•Definirea unui hold-down timer; în momentul când un ruter este informat că o rețea este inaccesibilă, acesta pornește un contor de timp pe durata căruia ruterul nu va accepta nici un update despre rețeaua respectivă

Bucle de rutare (6)

- Cum le prevenim ?
  - split-horizon (o rută nu este trimisă înapoi pe calea pe unde a fost învățată)
  - poison reverse (se face un update cu o metrică “infinită” pentru rutele care devin inaccesibile)
  - câmpul TTL din antetul IP (asigură că pachetele nu vor circula la infinit)

- „Split horizon” împiedică un ruter să trimită update despre o anumită rută pe aceeași interfață pe care a primit inițial informații despre existența ei
- „Route poisoning” sau „poison reverse” se referă la cazul în care se trimite explicit informația că o rețea este inaccesibilă (este trimis un update în care rețeaua respectivă are metrica maximă, astfel că ruterele care o primesc să o considere inaccesibilă)
- TTL-ul din antetul IP asigură faptul că un pachet va putea traversa un număr finit de hop-uri, după care va fi aruncat; la fiecare traversare a unui hop valoarea TTL-ului este decrementată, astfel că atunci când ajunge la valoarea 0, pachetele vor fi ignorate
- Update-urile provocate (triggered updates) sunt folosite de unele protocoale în cazul în care o rețea devine inaccesibilă, pentru a informa rapid celelalte rutere explicit despre acest eveniment

RIP

- Protocol open standard
- Metrica folosită: numărul de hop-uri
  - metrica maximă pentru o rețea este de 15 – nu poate fi folosit în rețele de diametru mai mare de 15
- Trimite update-uri la fiecare 30 de secunde (implicit)
- Trimite și update-uri provocate de modificări ale topologiei (triggered updates)
- Equal cost load balancing

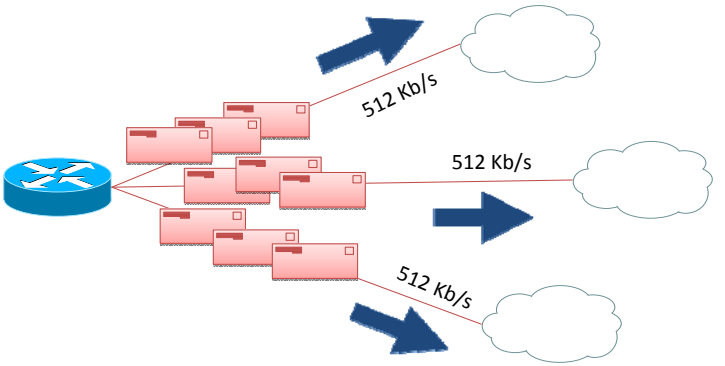
Protocolul RIP (Routing Information Protocol) este primul protocol dinamic folosit. Acesta a evoluat, în timp, de la un protocol classful (RIPv1) la un protocol classless (RIPv2). Deoarece este „open standard”, RIP poate fi implementat de către orice producator de rutere.

Metrica RIP se bazează numai pe numărul de hop-uri. Metrica maximă este 15. Aceast lucru înseamnă că nu poate fi folosit în rețele cu diametru (hop-count) mai mare de 15 hopuri. Totuși, este ușor de configurat și de întreținut, ceea ce îl face o alegere bună pentru rețele de dimensiuni mici.

RIP trimite update-uri periodice la intervale de 30 de secunde, dar poate trimite și update-uri provocate (triggered updates), dacă are loc o schimbare a topologiei. O altă funcționalitate a protocolului RIP este faptul că poate face „equal cost load balancing”.

Equal cost load-balancing

- Având mai multe legături cu aceeași metrică se împarte traficul în mod egal pe ele



În situația în care un ruter cunoaște mai multe căi către o singură destinație, iar metrica lor are aceeași valoare (în funcție de protocol: același număr de hop-uri, aceeași lățime de bandă, etc.) spunem că avem o situație de „equal cost metric”.

În tabela de rutare, „equal cost load balancing” se traduce prin existența unei singure intrări cu adresa rețelei, care are asociate mai multe interfețe de ieșire sau adrese IP next-hop. Un dezavantaj în cazul RIP-ului care ia în considerare doar hop-count-ul este că, dacă există, spre exemplu, două rute cu același număr de hop-uri către aceeași destinație, faptul că una este de 2Mb iar cealaltă de 512Kb nu va reprezenta un criteriu de diferențiere a celor două rute. Ele vor fi tratate egal, RIP realizând procesul de „equal cost load balancing”.

RIPv1 (1)

- Update timer – durata până la trimiterea următorului update de rutare (implicit 30 secunde)
- Invalid timer – durata până când o rută este marcată nevalidă (setarea metricii 16) dacă lipsește din update-urile vecinilor (implicit 180 secunde)
- Hold-down timer – folosit pentru prevenirea buclelor de rutare (implicit 180 secunde)
- Flush timer – durata până când rutele nevalide sunt scoase din tabela de rutare (implicit 240 secunde)

Protocolul RIPv1 prezintă o serie de caracteristici specifice protocoalelor distance vector:

•Update-uri periodice, trimise la un interval de timp fix, setat la valoarea de 30 de secunde

•Invalid timer – o rută este setată ca fiind invalidă dacă timp de 180 secunde lipsește din update-urile vecinilor, fiind totuși păstrată în tabela de rutare până la expirarea flush timer-ului

•Hold-down timer – sunt oprite orice schimbări în tabela de rutare pentru un interval de timp (180 secunde), pentru a preveni instalarea incorectă a unei rute inaccesibile, semnalate printr-un update de rutare primit de la un vecin care poate să nu fi aflat despre modificarea survenită

•Flush timer – previne ștergerea unei rute pentru un timp fixat chiar dacă aceasta a devenit inaccesibilă (240 secunde)

RIPv1 (2)

- Aflarea timpului trecut de la ultimul update

```
Router#show ip route
Codes: I - IGRP derived, R - RIP derived, O - OSPF derived,
C - connected, S - static, E - EGP derived, B - BGP derived,
* - candidate default route, IA - OSPF inter area route,
i - IS-IS derived, ia - IS-IS, U - per-user static route,
o - on-demand routing, M - mobile, P - periodic downloaded static route,
D - EIGRP, EX - EIGRP external, E1 - OSPF external type 1 route,
E2 - OSPF external type 2 route, N1 - OSPF NSSA external type 1 route,
N2 - OSPF NSSA external type 2 route


Gateway of last resort is 10.119.254.240 to network 10.140.0.0


R 172.150.0.0 [120/5] via 10.119.254.6, 0:01:00, Ethernet2
R 172.17.10.0 [120/8] via 10.119.254.244, 0:02:22, Ethernet2
R 172.70.132.0 [120/5] via 10.119.254.6, 0:00:59, Ethernet2
R 10.130.0.0 [120/3] via 10.119.254.6, 0:00:59, Ethernet2
```


Rutele care conțin caracterul „R” în fața adresei de rețea au fost adăugate la tabela de rutare prin protocolul RIP.

La apelarea comenzii **`show ip route`** se va afișa, în dreptul fiecărei adrese, timpul care a trecut de la ultimul update, în secunde. În caz că rutele au fost învățate prin RIP în acest fel se poate observa dacă există posibilitatea ca ruterul să efectueze load balancing. În acest caz se va afișa o rețea instalată cu mai multe interfețe de ieșire sau adrese IP next-hop.

Tot în tabela de rutare sunt afișate în dreptul fiecărei rute două valori numerice între paranteze drepte, sparate prin caracterul „/”. Aceste valori reprezintă distanța administrativă împreună cu metrica, specifice protocolului de rutare implementat. În acest caz, se observă că RIP are AD egală cu 120, iar metrica, în funcție de fiecare rută, numărul de echipamente de nivel 3 prin care trece un pachet până la destinație.

RIPv1 (3)

- Afișarea timerelor protocolului

```
Router#show ip protocols


Routing Protocol is "rip"
Sending updates every 30 seconds, next due in 2 seconds
Invalid after 180 seconds, hold down 180, flushed after 240

<output omis>

Routing for Networks:
172.19.0.0
2.0.0.0
10.3.0.0
Routing Information Sources:
Gateway Distance Last Update
Distance: (default is 120)
```


Comanda **`show ip protocols`** furnizează informații despre protocoalele de rutare active. Timpul trecut de la ultimul update apare sub eticheta „Last Update”. Pe lângă acesta, comanda va afișa când va fi trimis următorul update.

Se pot vizualiza, de asemenea, valorile pentru timer-ele invalid, hold-down și flush. Timer-ul invalid reprezintă timpul după care o rută este marcată invalidă dacă nu se mai primesc informații despre aceasta, timer-ul flush pornește când o rută devine invalidă și reprezintă timpul după care va fi ștersă din tabela de rutare dacă nu își revine, iar timer-ul hold-down este folosit pentru a evita buclele de rutare când o rută devine invalidă.

În output-ul comenzii se pot vedea și rețelele classful despre care ruterul respectiv va trimite update-uri.

RIPv2

- Folosește Split Horizon și Poison Reverse
- Funcționează classless
- Are metode de autentificare
- Suportă VLSM
- Suportă sumarizarea manuală a rutelor

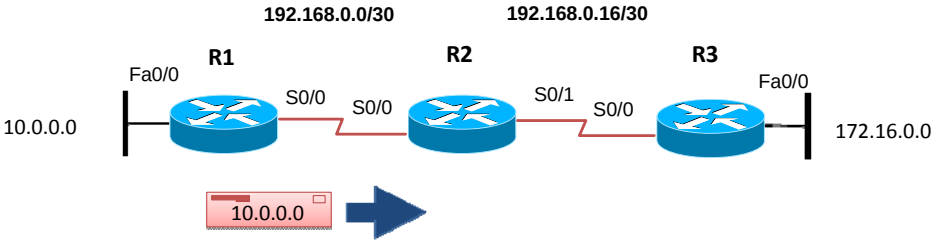
RIPv2 este un protocol de rutare classless, ceea ce înseamnă că include masca atașată unei adrese de rețea în update-urile trimise către celelalte rutere. VLSM reprezintă prescurtarea de la „Variable Length Subnet Mask”, adică există posibilitatea subnetării rețelelor classful, folosind diferite măști de rețea.

RIPv2 folosește mecanisme de autentificare pentru a securiza update-urile. Acest lucru are o importanță majoră când accesul în rețea nu poate fi strict controlat, existând potențiale încercări de interceptare și modificare a update-urilor.

RIPv2 preia de la RIPv1 tehnicile split horizon și poison reverse utilizate pentru prevenirea apariției buclelor de rutare. În loc de adrese broadcast, protocolul RIPv2 folosește adrese multicast pentru transmiterea update-urilor. De asemenea, spre deosebire de RIPv1, acest protocol suportă sumarizarea manuală a rutelor.

Split Horizon (1)

- R1 propagă ruta 10.0.0.0 către R2

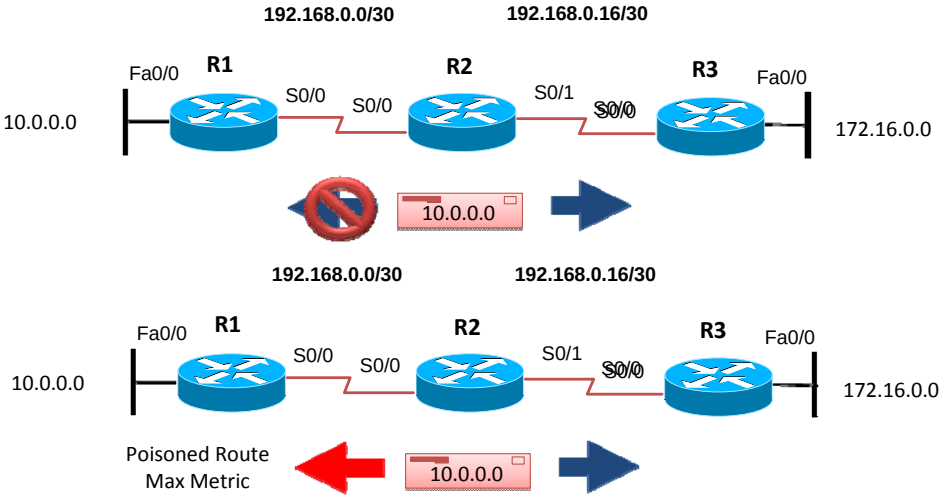


Split horizon este folosit pentru a preveni buclele de rutare cauzate de timpul de convergență crescut. Această regulă spune că un ruter nu poate transmite un update cu o anumită rețea pe aceeași interfață pe care a primit inițial informații despre adresa respectivă. Când un ruter află pentru prima oară despre o rută de la unul dintre vecini, se consideră că vecinul respectiv este mai aproape de destinație. În consecință, primul ruter nu îi va trimite update-uri vecinului conținând aceași rută pentru a evita suprascrierea tabelei de rutare a acestuia cu informații neactualizate.

În exemplul de mai sus, R1 trimite lui R2 un update cu adresa 10.0.0.0. R2 va adăuga această adresă în tabela sa de rutare.

Split Horizon (2)

- R2 transmite ruta 10.0.0.0 doar către R3, nu și înapoi spre R2



Respectând regula split horizon, ruta 10.0.0.0, pe care R2 a învațat-o de la R1, nu va fi inclusă în update-ul trimis care R1, dar va fi transmisă către R3.

O metodă alternativă de a preveni buclele de rutare este folosirea tehnicii de „route poisoning”. Aceasta implică faptul că se va transmite un update explicit despre rută care va fi marcată ca inaccesibilă, în loc ca aceasta să nu fie inclusă în viitoarele update-uri și să fie marcată ca invalidă la expirarea timer-ului invalid. Această metodă micșorează timpul de convergență. Update-urile vor conține o valoare a metricii egală cu 16, ceea ce indică în RIP o metrică infinită. Informația despre rețeaua inaccesibilă este astfel propagată în întreaga rețea de către ruterele vecine, nemaifiind nevoie să se aștepte până la expirarea anumitor timere, procesul de convergență fiind accelerat semnificativ.

Interior Gateway Routing Protocol

- Protocol proprietar Cisco
- Protocol Distance Vector
- A fost înlocuit de protocolul EIGRP
- Folosea o formulă în funcție de bandwidth, reliability și delay pentru a calcula metrica
- Update-urile de rutare erau trimise implicit la 90 de secunde

Protocolul IGRP (Interior Gateway Routing Protocol) este un protocol distance vector, care, spre deosebire de RIP, este proprietar Cisco, ceea ce înseamnă că va funcționa numai pe echipamente Cisco. Acest protocol nu mai este folosit în prezent, el fiind înlocuit de EIGRP.

Acest protocol a fost dezvoltat pentru a depăși neajunsurile protocolului RIP. Deoarece metrica RIP se poate dovedi ineficientă, IGRP suportă metrici multiple pe rute, cum ar fi: bandwidth, load, MTU, reliability și delay. Pentru a compara două rute, aceste metrici vor fi combinate într-o singură metrică cu ajutorul unei formule.

IGRP face update-uri periodice la un interval de 90 de secunde. Acestea sunt transmise prin broadcast.

EIGRP (1)

- Protocol proprietar Cisco
- Protocol Distance Vector avansat
- Poate face load balancing pe rute de cost inegal (“unequal cost load-balancing”)
- Folosește algoritmul DUAL pentru a calcula calea cea mai scurtă de la sursă la destinație
- Update-urile de rutare nu se trimit periodic, ci doar în cazul unor modificări în rețea (“triggered updates”)

Enhanced IGRP (EIGRP) a fost dezvoltat pe baza protocolului IGRP. Acesta este un protocol distance vector classless, cu unele caracteristici similare protocoalelor link-state. Caracteristicile EIGRP includ:

•Nu are update-uri periodice, ci folosește update-uri provocate atunci când există schimbări în topologia rețelei

•Folosește o tabelă de topologie pentru a reține toate rutele primite de la vecini, nu numai pe cele mai eficiente

•Suporta VSML și sumarizare manuală; acestea permit crearea de topologii mari, structurate ierarhic

•Poate face load balancing pe rute de cost inegal ținând cont și de alte caracteristici în afară de hop-count

•Metrica sa este bazată pe bandwidth-ul și pe delay-ul unei rute

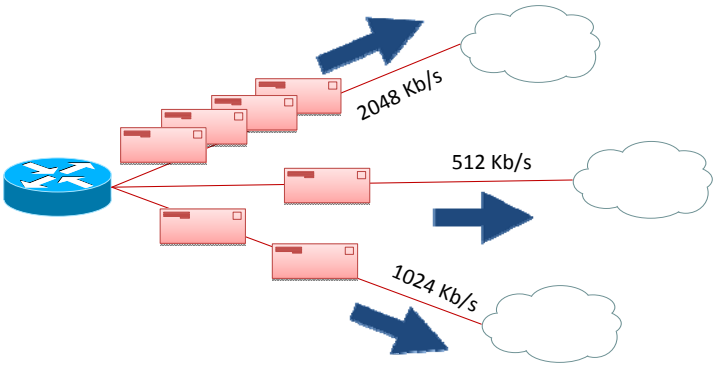
EIGRP (2)

- Protocol proprietar Cisco
- Protocol Distance Vector avansat
- Poate face load balancing pe rute de cost inegal (“unequal cost load-balancing”)
- Folosește algoritmul DUAL pentru a calcula calea cea mai scurtă de la sursă la destinație
- Update-urile de rutare nu se trimit periodic, ci doar în cazul unor modificări în rețea (“triggered updates”)

- Folosește algoritmul DUAL (Diffusion Update Algorithm) pentru a calcula cea mai scurtă rută până la o anumită destinație. Acesta permite inserarea rutelor de back-up în tabela de topologie EIGRP, care sunt folosite dacă ruta primară devine inaccesibilă. Astfel, trecerea la ruta de back-up în cazul unei probleme este aproape imediată și nu presupune nici o acțiune din partea altor rutere. În cazul inexistenței rutei de back-up, algorimtul DUAL este reinițiat pentru descoperirea unor posibile rute alternative.
- EIGRP suportă diferite protocoale de nivel 3, cum ar fi IP, IPX sau AppleTalk. Astfel, indiferent peste ce protocol rulează EIGRP-ul sau ce fel de rute transportă, el va aplica același algoritm.

Unqual cost load-balancing

- EIGRP poate distribui traficul pe legături în funcție de metrica lor



Protocolul EIGRP va distribui traficul pe mai multe legături, în funcție de metrica care la rândul ei depinde de delay si bandwidth. Faptul că ține cont de bandwidth și că știe să facă unequal load-balancing este un avantaj major în comparație cu RIP.

EIGRP folosește bounded updates, adică numai ruterele care au nevoie de o anumită informație primesc pachetele de update, minimizând congestionarea legăturilor. O asemănare cu protocoalele de rutare link-state este faptul că EIGRP transmite informații numai atunci când există o schimbare în topologia rețelei incluzând informații numai despre modificările care au avut loc.

Rezumat

- Avantaje și dezavantaje ale protocalelor Distance Vector
- Bucle de rutare
- Split-horizon
- Poison-reverse
- RIPv1
- RIPv2
- EIGRP

1. Pe ce adresă destinație sunt transmise update-urile protocolului de rutare RIPv1?

2. Ce reprezintă problema „count to infinity" și care sunt cauzele apariției acesteia?

3. Care sunt metodele de prevenire a apariției buclelor de rutare?

4. Care sunt îmbunătățirile principale aduse de RIPv2 în comparație cu RIPv1?

5. Care este algoritmul folosit de EIGRP pentru determinarea rutelor optime către destinație?