

Network Working Group

C. Hedrick Request for Comments: 1058

Rutgers University

June 1988

Routing Information Protocol

Privire de ansamblu

Acest RFC descrie un protocol pentru schimbul de informatii de routare intre gatewaye si alte gazde(hosts). Este menit sa fie folosit ca o baza pentru dezvoltarea software-ului utilizat in configurarea gatewayelor, pentru a fi folosit in comunitatea internet. Distributia acestui memo este nelimitata.

Cuprins

1. Introducere	
1.1 Limitari ale protocolului.....	4
1.2 Organizarea documentului... ..	5
2. Algoritmi cu vectori de distanta	6
2.1. Tratarea schimbarilor in topologie	12
2.2. Prevenirea instabilitatii	13
2.2.1. Split horizon	16
2.2.2. Actualizari declansate.....	18
3. Specificatii de protocol	19
3.1. Formatul mesajelor	21
3.2. Consideratii de adresare.....	24
3.3. Timeri	26

3.4. Procesarea intrarilor	28
3.4.1. Cererea	29
3.4.2. Raspunsul	30
3.5. Procesarea iesirilor	33
3.6. Compatibilitate	36
4. Functii de control	36

Aceasta lucrare e gandita sa urmareasca urmatoarele lucruri:

- Documentarea unui protocol si algoritmi de rutare, care sunt in prezent intr-o larga folosire, dar care nu au fost formal documentati.

- Specificarea unor imbunatatiri a algoritmilor care vor imbunatati stabilitatea gateway-lor in retelele mari. Aceste imbunatatiri nu vor introduce vreun fel de incompatibilitati cu implementarile existente. Acestea vor fi incorporate in toate implementarile acestui protocol.

- Sugereaza niste trasaturi optionale pentru a permite un control si o configurare mult mai amanuntite. Aceste trasaturi au fost dezvoltate special sa rezolve probleme care au aparut in utilizarea de catre comunitatea NSFnet. Totusi, ar trebui sa aiba o utilitate mai generala.

The Routing Information Protocol (RIP) descris aici este bazat pe programul "routed", distribuit cu 4.3 Berkley Software Distribution. Totusi, sunt multe alte implementari a ceea ce se presupune a fi acelasi protocol. Din pacate, aceste implementari variate sunt in contradictie in mai multe detalii. Aceste specificatii reprezinta o combinatie de trasaturi luate din mai multe implementari. Credem ca un program construit conform acestui document va "interopera" cu root-ul si cu alte implementari ale RIP-ului de care avem cunostinta.

De remarcat ca aceasta descriere adopta un punct de vedere diferit fata de multe implementari existente referitoare la cazul in care metricile ar trebui incrementate. Facand o schimbare corespunzatoare in metrica folosita la o retea locala, am retinut compatibilitatea cu alte implementari existente. Vezi sectiunea 3.6 pentru detalii privind acest subiect.

1. Introducere

Acest capitol descrie un protocol dintr-o serie de protocoale de retea, bazate pe algoritmul Bellman-Ford (sau cu vector de distanta). Acest algoritm a fost folosit pentru rutari in retele de calculatoare de la inceputul ARPANET-ului. Pachetele si protocoalele descrise aici sunt bazate pe programul "routed", care este inclus cu distributia Berkeley din Unix. A devenit un "de facto" standard pt schimbarea informatiilor de rutare intre gateway-uri si host-uri. In acest scop este implementat de cei mai de succes comercianti de Ip-uri pt Gateway-uri. Totusi multi dintre acesti comercianti au propriile lor protocoale care sunt folosite printre gateway-urile lor.

Acest protocol este cel mai folosit ca un "Interior Gateway Protocol". Intr-o retea internationala cum este Internetul, este foarte putin probabil ca un singur protocol de rutare va fi folosit pentru intreaga retea. Mai degraba, retea va fi organizata ca o colectie de "sisteme autonome". Un sistem autonom va fi, in general, administrat de o singura entitate sau va avea, cel putin un grad rezonabil de control administrativ si tehnic. Fiecare sistem autonom va avea propria sa tehnologie de rutare. Aceasta poate fi diferita pentru sisteme diferite. Protocolul de rutare folosit in aceste sisteme autonome este referit ca un "Interior Gateway Protocol", sau IGP. Un protocol separat este folosit sa interfateze cu sistemele autonome. Primul astfel de protocol, inca folosit in Internet este "EGP" (Exterior Gateway Protocol). Asemenea protocoale sunt referite ca protocoale de rutare inter-AS. Rip a fost construit sa lucreze cu retele de marimi moderate si folosind tehnologii omogene. Astfel, este potrivit ca IGP pentru multe campusuri si retele regionale, care folosesc linii seriale a caror viteza nu variaza mult. Nu este gandit pentru folosirea in medii mai complexe. Pentru mai multe informatii despre in ce contexte se potriveste RIP consultati: Braden si Posttel [3].

Rip este unul dintre algoritmi cunoscut ca "algoritmi cu vector de distanta". Prima descriere a acestei clase de algoritmi cunoscuta autorului este in Ford si Fulkerson [6]. Din aceasta cauza, sunt cunoscuti ca algoritmi Ford-Fulkerson. De asemenea este folosit si termenul de Bellman-Ford. Vine din faptul ca formularea este bazata pe o ecuatie a lui Bellman, baza "programarii dinamice". (Pentru o introducere standard in aceasta zona, vedeti [1].) Prezentarea din acest document este usor bazata pe [2]. Textul contine o

introducere in matematica algoritmilor de rutare. Descrie si justifica mai multe variante ale algoritmului prezentat aici, precum si un numar de alti algoritmi asemanatori. Algoritmii de baza descrisi in acest protocol erau folositi in rutarea calculatoarelor inca din 1969 in ARPANET. Totusi, originea acestui protocol este in protocoalele retelei Xerox. Protocoalele PUP (vezi [4]) folosite in "Gateway Information Protocol" pentru schimbarea informatiilor de rutare. O asa zisa versiune innoita a acestui protocol a fost adoptata pentru arhitectura "Xerox Network Systems" (XNS), cu numele de "Routing Information Protocol". (vezi [7].). Rutarea Berkley este foarte asemanatoare ca "Routing Information Protocol", cu adresele XNS inlocuite de un format de adrese mai general, capabil sa manipuleze IP-uri si alte tipuri de adrese, si cu innoirea rutarii limitata la una la fiecare 30 de secunde. Din cauza acestei asemanari, termenul de "routing information protocol" (sau doar RIP) este folosit sa se refere la ambele protocoale: XNS si protocolul folosit de rutare.

RIP este menit pentru folosirea in Internetul bazat pe IP-uri. Internetul este organizat intr-un numar de retele conectate la gateway-uri. Retelele pot fi fie punct-la-punct (point-to-point), sau retele mai complexe cum ar fi Ethernet sau ARPANET-ul. Host-urile si gateway-urile sunt prezentate cu datagrame de ip-uri adresate unor host-uri. Rutarea este metoda prin care host-ul sau gateway-ul decide unde sa trimita datagrama. Poate sa trimita datagrama direct la destinatie, daca destinatia este o retea conectata la host sau la gateway. Totusi, cazul cel mai interesant este cand destinatia nu poate fi accesata direct. In acest caz, host-ul sau gateway-ul incearca sa trimita datagrama unui gateway mai apropiat de destinatie. Scopul protocolului de rutare este foarte simplu: este de a aproviziona informatie necesara rutarii.

1.1. Limitari ale protocolului

Acest protocol nu rezolva toate problemele rutarii. Cum am mentionat mai sus, el este gandit sa fie folosit ca un IGP, in retele omogene de marime moderata. In plus, trebuie mentionate urmatoarele limitari (specifice):

- Protocolul este limitat la retele in care lungimea maxima unui drum sa nu fie mai mare de 15 "salturi". Proiectantii cred ca designul protocolului de baza nu este

potrivit pentru rețele mai mari. “A se observa ca aceasta afirmatie a limitarii presupune ca un cost de 1 este folosit pentru fiecare retea”.

Asa este de fapt cum a fost configurat RIP. Daca administratorul de sistem alege sa foloseasca costuri mai mari, limita de sus de 15 poate usor sa devina o problema.

- Protocolul depinde de “numararea pana la infinit” pentru a rezolva anumite situatii deosebite. (Aceasta va fi explicat in sectiunea urmatoare.)

Daca sistemul de rețele are cateva sute de rețele, si s-a format o bucla de rutare care le implica pe toate, solutionarea buclei ar cere ori prea mult timp (daca frecventa rutarii ar avea limitari) sau largime de banda mai mare(daca actualizarile ar fi trimise de fiecare data cand sunt detectate). Astfel de bucla ar consuma o mare latime de banda inainte ca ea sa fie corectata. Credem ca in cazurile reale, aceasta nu ar fi o problema decat pentru rețelele cu transfer scazut. Chiar si atunci, in cele mai multe cazuri, problema ar fi ceva neobisnuit, intrucat sunt luate o serie de masuri pentru a preveni aparitia acestor probleme.

-Acest protocol foloseste metrici fixe pentru compararea rutelor alternative. Nu este potrivit pentru situatii in care rutele trebuie alese referitor la parametri de timp real ca: latentă măsurată, siguranță, sau încărcare. Extensiile evidente pentru a permite metrici de acest tip sunt pasibile de a introduce instabilitati de asa maniera incat protocolul nu este construit sa le combata.

1.2. Organizarea documentului

Partea principala a acestei lucrari este impartita in doua parti, care ocupa urmatoarele doua sectiuni:

2 O dezvoltare conceptuala si justificarea algoritmilor cu vector de distanta, in general.

3 Descrierea protocolului.

Fiecare din aceste doua sectiuni pot in mare masura sa fie individuale. Sectiunea 2 incearca sa dea o prezentare matematica simpla a bazelor algoritmului. De observat ca prezentarea urmeaza o metoda gen “spirala”. Este descris un algoritm initial, destul de

simplicu. Rafinamentele sunt adaugate in sectiuni succesive. Sectiunea 3 este descrierea propriu-zisa a protocolului. Exceptand cazurile in care se fac referiri la sectiunea 2, ar trebui sa fie posibil sa implementati RIP numai pe baza specificarilor date in sectiunea 3.

2. Algoritmi cu vectori de distanta

Rutarea este incercarea de a gasi o cale de la expeditor la o destinatie. In modelul IP "Catenet" aceasta se reduce, in primul rand, la problema de a gasi porti intre retele. Atata timp cat un mesaj ramane intr-o retea sau subretea, orice probleme de rutare sunt rezolvate de o tehnologie care e specifica retelei. De exemplu, deopotriiva Ethernet-ul si ARPANET-ul definesc o metoda in care expeditorul poate corespunda cu orice destinatie, care e specifica retelei. Rutarea cu IP-uri apare in primul rand, cand mesajele trebuie sa treaca de la un expeditor din cadrul unei astfel de retele spre o destinatie din alta retea. In acest caz, mesajul trebuie sa treaca prin gateway-uri (gatewaye) care conecteaza retelele. Daca retelele nu sunt adiacente, mesajul trebuie sa treaca prin mai multe retele intermediare si gateway-urile (portile) care le leaga.

Odata ce un mesaj ajunge la un gateway (o poarta) care este in aceeasi retea cu destinatia, este folosita tehnologia retelei pentru a ajunge la destinatie.

De-a lungul acestei sectiuni, termenul de "retea" este folosit generic, ca sa acopere o singura retea de dimensiuni mari (e.g., Ethernet), conexiune punct la punct, sau ARPANET-ul. Punctul critic este faptul ca o retea este tratata ca o singura entitate de catre IP. Fie nu este necesara rutarea (cu o conexiune punct la punct), sau rutarea este facuta intr-o maniera care este transparenta pentru IP, permitandu-i acestuia sa trateze toata retea ca un singur sistem conectat in totalitate (ca Ethernet-ul sau ARPANET-ul). De notat ca aici folosim termenul de "retea" pentru a referi subretelele in cazul in care e folosita adresarea retelei.

Mai multe abordari diferite sunt posibile pentru a gasi cai intre retele. O metoda folositoare de a categorisi aceste abordari este pe baza informatiei de care gateway-urile (portile) au nevoie sa o schimbe ca sa poata gasi caile. Algoritmii cu vector de distanta se bazeaza pe schimbul unei cantitati mici de informatie. Fiecare entitate (gateway or host) care participa in protocolul de rutare este presupus ca pastreaza informatii despre toate

destinatiile acelei retele. Aceasta rezumare este posibila deoarece, in ceea ce priveste IP-ul, rutarea in interiorul retelei este invizibila. Ficare intrare in baza de date a rutarii include urmatoarea gateway (poarta) pentru entitatea la care ar trebui trimise datagramele. In plus, include o "metrica" masurand in total distanta pana la entitate. Distanta este, cumva, un termen generalizat, care poate acoperi timpul de intarziere care e necesar trimiterii mesajului la entitate, costul trimiterii mesajului...etc. Algoritmii cu vector de distanta isi iau numele din faptul ca este posibila calcularea rutelor optime, cand singura informatie schimbata este schimbul distantei. In plus, informatia este schimbata numai intre entitati care sunt adiacente, adica entitatile care impart o retea comuna.

Desi rutarea este bazata pe informatia despre retele, este uneori necesara monitorizarea rutelor la host-uri individuale. Protocolul RIP nu face deosebire intre retele si host-uri. El descrie doar schimbul de informatie in jurul destinatiilor, care ar putea sa fie retele sau host-uri. (De retinut totusi, ca este posibil pentru un "implementor" sa aleaga sa nu suporte rute la host-uri. Vezi sectiunea 3.2.) De fapt, dezvoltarile matematice sunt cel mai convenabil gandite ca rute intre un host sau gateway (poarta) si alta. Cand discutam algoritmul in termeni abstracti, este bine sa ne gandim la o intrare de rutare pentru o retea ca o abreviere pentru intrarile de rutare ale tuturor entitatilor conectate la acea retea. Asemenea reducere are sens datorita faptului ca noi gandim o retea ca neavand o structura interna care sa fie vizibila la nivelul IP-ului. Astfel, in general vom atribui aceeasi distanta fiecărei entitati dintr-o retea data.

Am spus mai sus ca fiecare entitate retine o baza de date de rutare cu o singura intrare pentru fiecare destinatie din sistem posibila. O implementare reala, probabil va avea nevoie sa retina urmatoarea informatie despre fiecare destinatie:

- adresa: in implementarile IP ale acestor algoritmi, aceasta ar fi adresa IP a host-urilor sau retelei.

- gateway (poarta) : prima gateway (poarta) de-a lungul rutei catre destinatie.

- interfata: reseaua (fizica) care trebuie folosita sa ajunga la prima gateway (poarta).

- metrica: un numar, care indica distanta pana la destinatie.

- timer-ul : timpul de la ultima actualizare a intrarii.

In plus, vor fi probabil incluse, mai multe flag-uri si alte informatii interne. Aceasta baza de date este initializata cu o descriere a entitatilor care sunt conectate direct la sistem. Este actualizata conform informatiilor primite in mesaje de la gateway-uri (porti) vecine.

Cea mai importanta informatie schimbata de host-uri si gateway-uri (porti) este cea purtata in mesajele de actualizare. Fiecare entitate care participa in schema de rutare, trimite mesaje de actualizare care descriu baza de date a rutarii care exista in entitatea respectiva. Este posibil de mentinut rute optime pentru intregul sistem folosind doar informatia obtinuta de la entitati vecine. Algoritmul folosit pentru aceasta va fi descris in urmatoarea sectiune.

Cum am mentionat mai sus, scopul rutarii este de a gasi o cale de trimitere datagramelor la destinatia lor. Algoritmii cu vector de distanta sunt bazati pe un tabel care arata cea mai buna cale spre fiecare destinatie din sistem. Bineinteles, ca sa definim fiecare cale, este mai bine sa avem o metoda sa masuram ce e mai bun. Acestea sunt metricile.

In retele simple, este obisnuit sa folosim o metrica care calculeaza prin cate gateway-uri (porti) trebuie sa treaca un mesaj. In retele mai complexe, o metrica reprezinta suma totala de intarzieri pe care le sufera mesajul, costul trimiterii lui, sau alta cantitate care poate fi minimizata. Cererea principala este de a fi posibila reprezentarea metricii ca o suma de "costuri" pentru hopurile individuale.

Formal, este posibil sa ajungem direct de la entitatea i la entitatea j (i.e., fara a trece printr-o alta gateway (poarta), atunci un cost, $d(i,j)$, este asociat cu un salt intre i si j . In cazul normal, in care toate entitatile dintr-o retea data sunt considerate a fi la fel, $d(i,j)$ sunt aceleasi pentru toate destinatiile dintr-o retea data si reprezinta costul folosirii acelei retele. Pentru a obtine metrica unei rute, se adauga costul hopurilor individuale care formeaza ruta. Pentru scopul acestui memo, preupunem costurile a fi numere pozitive.

Fie $d(i,j)$ care reprezinta metrica celei mai bune cai de la entitatea i la entitatea j care ar trebui definit pentru fiecare pereche de entitati. $d(i,j)$ reprezinta costurile pasilor individuali. Formal, fie $d(i,j)$ reprezentand costurile drumului direct de la entitatea i la entitatea j . Este infinit daca i si j nu sunt vecini intermediari. (notati ca $d(i,j)$ este infinit.

Adica, nu consideram ca este o conexiune directa de la un nod la el insusi). De vreme ce costurile sunt adunate, este usor de aratat ca cea mai buna metrica trebuie descrisa de:

$$D(i,i)=0, \quad \text{oricare } i$$

$$D(i,j)= \min_k [d(i,k) + D(k,j)], \quad \text{altfel}$$

si ca cele mai bune cai incep de la i la acei vecini k pentru care $d(i,k) + D(k,j)$ are valoarea minima. (Acesta lucruri pot fi aratate prin inductie dupa numarul de pasi din rute.) Observam ca, putem limita a doua ecuatie la k , care e vecin intermediar cu i . Pentru ceilalti, $d(i,k)$ este infinit, asa ca termenul care ii include nu poate fi niciodata minim.

Rezulta ca se poate calcula metrica printr-un simplu algoritm bazat pe aceasta. Entitatea i isi face ca vecinii k sa-si trimita estimarile distantei la destinatia j . Cand i primeste estimarile de la k , adauga $d(i,k)$ la fiecare numar. Acesta este costul traversarii retelei intre nodurile i si k . Astfel i compara valorile de la toti vecinii si o alege pe cea mai mica.

O dovada data in [2], este ca acest algoritm converge la estimarile corecte a lui $D(i,j)$ in timp infinit, in absenta schimbarilor din topologie.

Autorii fac foarte putine presupuneri privind ordinea in care entitatile isi trimit informatii, sau cand este recalculat minimul.

Deci, entitatile nu se pot opri din a trimite actualizari sau sa recalculeze metrica, iar retelele nu pot intarzia la nesfarsit mesajele. (Prabusirea unei entitati de rutare este o schimbare in topologie.) De asemenea, dovada lor nu face presupuneri privind estimarile initiale ale lui $D(i,j)$, exceptie facand ca ele trebuie sa fie nenegative. Faptul ca aceste presupuneri sunt destul de bune este foarte important. Deoarece nu trebuie sa facem presupuneri despre momentul cand sunt trimise actualizarile, este sigur pentru a rula algoritmul asincronic. Adica, fiecare entitate poate trimite actualizari conform timpului propriu. Actualizarile pot fi lasate de retea, atat timp cat nu sunt lasate toate. Deoarece nu trebuie sa facem presupuneri privind conditia de start, algoritmul poate suporta schimbari. Cand se schimba sistemul, algoritmul de rutare isi schimba echilibrul, folosindu-l pe cel vechi ca punct de inceput. Este important ca algoritmul sa converga la timp infinit oricare ar fi punctul de plecare. Altfel, anumite schimbari pot duce la o comportare non-convergenta.

Expunerea de mai sus a algoritmului (si dovada) presupune ca fiecare entitate retine copii ale estimarilor care vin de la fiecare din vecinii ei si alege minimul dintre toti vecinii. In realitate, implementarile nu fac asta. Pur si simplu retin cea mai buna metrica vazuta pana la un moment dat si identitatea vecinului care a trimis-o. Inlocuiesc aceasta informatie de fiecare data cand intalnesc alta mai buna(mica). Asta le permite sa calculeze cresterea minima, fara a trebui sa depoziteze date de la toti vecinii.

Nu exista alta diferenta intre algoritmul descris in document si cei folositi in protocoalele reale cum ar fi RIP: in descrierea de mai sus, fiecare entitate include o intrare pentru ea, aratand o distanta fata de 0. De fapt, aceasta nu este facuta in general. Amintiti-va ca toate entitatile dintr-o retea sunt adunate de o singura intrare in retea. Fie situatia unui host sau Gateway(porti) G care e conectata la retea A. C reprezinta costul folosirii retelei A (de obicei o metrica de 1). (De amintit presupunerea ca structura interna a unei retele nu este vizibila pentru IP, si de aceea costul drumului de la o entitate la alta este acelasi.) In principiu, G ar trebui sa primeasca un mesaj de la fiecare entitate H din retea A, aratand un cost de 0 de a merge de la o entitate la ea insasi. G ar trebui sa adune $C+0$ ca distanta pana la H. Decat sa se uite la toate mesajele identice, pur si simplu incepe prin a face o intrare pentru retea A in tabelul sau, asignandu-i o metrica C. Acesta metrica pentru retea A ar trebui gandita ca sumand intrarile pentru toate intrarile in retea A. Singura entitate din A care nu poate fi adunata la acea intrare comuna este G, de vreme ce costul drumului de la G la G este 0, si nu C. Dar din moment ce nu ne-au trebuit intrarile 0, putem sa ne descurcam usor numai cu o singura intrare pentru A. O alta implicatie a acestei strategii este: deoarece nu trebuie sa folosim intrarile 0 la nimic, Host-urile care nu functioneaza ca Gateway-uri(porti) nu trebuie sa trimita mesaje actualizate. Host-urile care nu functioneaza ca gateway-uri (i.e., host-uri care sunt conectate la o singura retea) pot sa nu aiba informatie utila care sa contribuie la alta intrare in afara de a lor $D(i,j)=0$. Cum au o singura interfata, este usor de vazut ca o cale la oricare alta retea prin ele, va trece prin interfata si se va intoarce in afara. Desi costul unei astfel de rute va fi mai mare decat cel mai bun cost cel putin cu C. De vreme ce nu avem nevoie de intrarile 0, non-gateway-urile nu trebuie sa participe in protocolul de rutare.

Sa rezumam ce face un host sau o gateway G. Pentru fiecare destinatie din sistem, G va retine o estimare curenta a metricii pentru aceea destinatie (i.e., costul total de a ajunge la ea) si identitatea gateway-urilor vecine pe a caror date este bazata metrica. Daca destinatia este intr-o retea conectata direct cu G, atunci G foloseste o intrare care arata costul folosirii retelei, si faptul ca nu este folosita nici o gateway (poarta) pentru a se ajunge la destinatie.

Este usor de aratat ca odata ce adunarea converge la metricile corecte, vecinul care e inregistrat prin aceasta metoda este de fapt prima gateway din calea destinatiei. (daca exista mai multe cai la fel de bune, este prima poarta de pe oricare din ele.)

Aceasta combinatie de destinatii, metrici, si gateway-uri, se refera la o ruta spre destinatie cu metrica, folosind acea poarta.

Pana acum metoda are o singura metoda de a mica metrica, pt ca metrica existenta este pastrata pana apare una mai mica. Este posibil ca estimarea initiala sa fie mica. Astfel, trebuie sa existe o metoda pentru a creste metrica. Se dovedeste a fi suficient sa folosim urmatoarea regula: presupunem ca ruta curenta spre o destinatie are metrica D si foloseste poarta G. Daca a ajuns un nou set de informatii de la alta sursa in afara de G, se actualizeaza numai ruta daca metrica e mai buna ca D. Dar, daca ajunge un nou set de informatii de la G, se actualizeaza intotdeauna D dupa noua valoare. Este usor de aratat cu aceasta regula, ca procesul de actualizare crescut (incrementat) produce aceeași ruta ca un calcul care retine ultima informatie despre toti vecinii si face un minim explicit. (De remarcat ca aceste lucruri presupun ca reseaua este configurata static. Nu admite posibilitatea ca un sistem sa cedeze).

In concluzie, asa arata algoritmul vectorului de distanta asa cum a fost dezvoltat pana acum. (Acesta nu este o afirmatie a protocolului RIP. Mai sunt cateva finisari de adaugat.) Urmatoarea procedura este purtata de fiecare entitate care participa la protocolul de rutare. Acesta trebuie sa includa toate portile din sistem. Host-urile care nu sunt gateway-uri pot participa de asemenea.

- Retineti o tabela cu o intrare pentru fiecare destinatie posibila din sistem. Intrarea contine distanta D la destinatie, si prima gateway G din ruta la acea retea. Conceptual, ar trebui sa existe o intrare pentru entitate cu metrica 0, dar aceasta nu este in realitate inclusa.

- Periodic, trimiteți o actualizare a informațiilor de rutare la fiecare vecin. Actualizarea este un set de mesaje care conține toate informațiile de la tabelul de rutare. Conține o intrare pentru fiecare destinație, cu distanța aratăta acelei destinații.

-Când o actualizare a informațiilor de rutare ajunge la vecinul G' , se adună costul asociat cu rețeaua care este împartită cu G' . (Aceasta ar trebui să fie rețeaua în care au ajuns actualizările.) Fie distanța rezultată D' . Comparați distanțele rezultate cu intrările curente din rețea. Dacă noua distanță D' pentru N este mai mică decât valoarea existentă D , se adoptă noua rută. Adică, se schimbă intrarea din tabelă pentru N pentru a rezulta metrica D' și gateway-ul (poarta) G' . Dacă G' este gateway-ul (poarta) de unde a pornit rută, i.e., $G'=G$, atunci se folosește noua metrică chiar dacă este mai mare decât cea inițială.

2.1. Tratarea schimbărilor în topologie

În discuția de mai sus se presupune că topologia rețelei este fixă. În practică, gateway-urile (portile) și liniile, de multe ori, cad și își revin iar. Pentru a trata această posibilitate, trebuie să modificăm un pic algoritmul. Versiunea teoretică a algoritmului presupune un minim dintre toți vecinii cei mai apropiați. Dacă topologia se schimbă, se schimbă și setul de vecini. De aceea, următoarea dată când este făcut calculul, schimbarea se va vedea. Totuși, cum s-a spus mai sus, implementările actuale folosesc o versiune incrementată a minimizării. Este reținută numai cea mai bună rută pentru orice destinație dată. Dacă gateway-ul (poarta) implicat în rută cade, sau conexiunea cu rețeaua cade, e posibil ca rezultatul să nu reflecte schimbarea. Algoritmul, așa cum a fost prezentat până acum, depinde de o gateway (poarta) care își anunță vecinii dacă se schimbă metricile. Dacă cade gateway-ul (poarta), atunci ea nu are cum să-și anunțe vecinii de schimbări.

Pentru a trata o astfel de problemă, protocoalele vectorului de distanță trebuie să facă anumite provizii pentru a cronometra rutele. Detaliile depind de un protocol specific. De exemplu, în RIP fiecare gateway (poarta) care participă în rutare trimite un mesaj actualizat la toți vecinii la fiecare 30 de secunde. Să presupunem că rută curentă pentru rețeaua N folosește gateway-ul G .

Daca nu se aude nimic de la G timp de 180 de secunde, putem presupune fie ca a cazut gateway-ul (poarta) sau ca conexiunea cu ea a devenit nefolosibila. Astfel, marcam ruta ca fiind invalida. Cand captam un mesaj de la alt vecin care are o ruta valida spre N, ruta valida o va inlocui pe cea invalida. Observati ca asteptam 180 de secunde inainte sa inchidem o ruta chiar daca asteptam sa primim raspuns de la fiecare vecin la fiecare 30 de secunde. Din nefericire, uneori mesajele sunt pierdute in retea. Astfel, probabil nu este o idee buna sa invalidezi o ruta bazata pe un singur mesaj pierdut.

Asa cum vom vedea mai jos, este util sa avem o metoda de a anunta vecinii prin alte protocoale ale acestei clase. Putem face asta printr-un mesaj actualizat, marcand reseaua ca inaccesibila. Este aleasa o valoare a unei anumite metrici pentru a indica o destinatie inaccesibila; valoarea acelei metrici este mai mare decat cea mai mare valoare valida pe care ne asteptam sa o vedem. In aceasta implementare a RIP-ului , este folosita valoarea 16. Aceasta valoare este referita ca “infinita”, din moment ce este mai mare decat cea mai mare metrica valida. Valoarea 16 pare a fi un numar extrem de mic. Este ales asa de mic din motive pe care le vom vedea in scurt timp. In majoritatea implementarilor, este folosita aceeasi conventie pentru a marca o ruta invalida.

2.2. Prevenirea instabilitatii

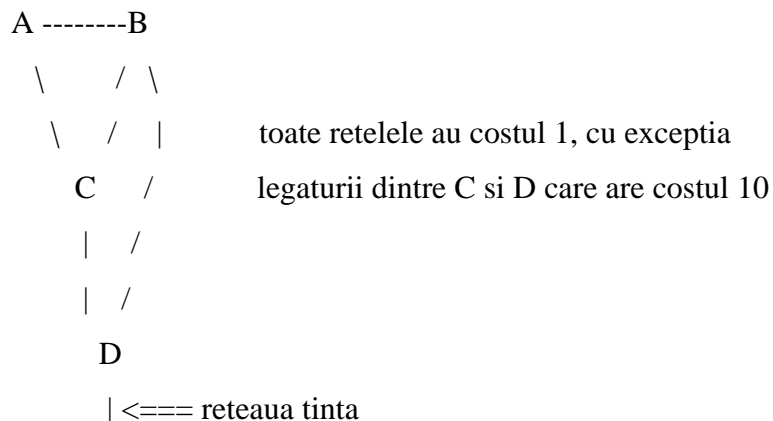
Algoritmul prezentat mai sus va aloca intotdeauna o gazda sau un gateway pentru calcularea tabelii corecte de routare. Oricum, aceasta nu este inca suficient pentru a-l face folositor in practica. Dovezile de mai sus arata doar ca tabelele de routare vor ajunge la valori corecte in timp finit. Ele nu garanteaza ca acest timp va fi suficient de mic pentru a fi folositor, nici nu vor spune ce se va intampla cu metricile retelelor care vor deveni inaccesibile.

Este destul de usor sa extindem calculele pentru a trata rutele devenite inaccesibile. Conventia sugerata mai sus va realiza acest lucru. Alegem o metrica mare pentru a reprezenta “infinitul”. Aceasta valoare trebuie sa fie destul de mare pentru ca nici o metrica reala sa nu devina niciodata atat de mare. Pentru acest exemplu vom utiliza valoarea 16. Presupunem ca o retea devine inaccesibila. Toate gateway-urile din imediata apropiere vor seta metrica pentru acea retea la 16.

Pentru simplitate, vom presupune ca toate gateway-urile vecine au o noua componenta hardware care le conecteaza direct la retea disparuta. Deoarece aceasta este singura conexiune cu aceasta retea, toate celelalte gateway din sistem vor apela la noi cai care conduc la unul din aceste gatewaye. Este usor de inteles ca odata realizata convergenta, toate gateway vor avea metrica macar 16 pentru retea cazuta. Gateway-urile la un hop distanta de vecinii originali vor urca metricele la macar 17; gateway-urile la 2 hopuri distanta vor urca la minim 18, etc. Cum aceste metrice sunt mai mari decat valoarea maxima pentru metrica, toate sunt setate la 16. Este evident ca sistemul va converge acum la o metrica de 16 pentru retea cazuta la toate gateway-urile.

Din nefericire, la intrebarea cat de mult va lua conversia nu este obligat sa gaseasca un raspuns atat de simplu. Inainte de a merge mai departe, este bine sa vedem un exemplu (luat din [2]). Observam, de asemenea, ca ceea ce vom arata nu se va intampla cu o implementare corecta a RIP. Incercam sa aratam de ce anumite caracteristici sunt necesare.

Literele corespund gatewayelor, iar liniile retelelor.



Fiecare gateway va avea o tabela care arata ruta la fiecare retea.

Oricum, pentru scopul acestei ilustratii, vom arata doar rutele pentru fiecare gateway la retea marcata la sfarsitul diagramei.

D: conectata direct, metrica 1

B: ruta prin D, metrica 2

C: ruta prin B, metrica 3

A: ruta prin B, metrica 3

Acum sa presupunem ca legatura dintre B si D cade. Rutele ar trebui sa foloseasca acum legatura dintre C si D. Din nefericire, va dura o vreme pentru ca aceasta sa se intample. Rutele schimba startul atunci cand B anunta ca ruta spre D nu mai este utilizabila. Pentru simplitate, in graficul de mai jos presupunem ca toate gateway-urile trimit notificari in acelasi timp. Graficul arata metrica pentru retea tinta, asa cum apare in tebla de routare la fiecare gateway.

Timpul----->

D: dir, 1 dir, 1 dir, 1 dir, 1 ... dir, 1 dir, 1

B: unreach C, 4 C, 5 C, 6 C, 11 C, 12

C: B, 3 A, 4 A, 5 A, 6 A, 11 D, 11

A: B, 3 C, 4 C, 5 C, 6 C, 11 C, 12

dir = conectata direct

(unreach)care nu poate fi ajunsa = indisponibila

Problema este: B este capabil sa scape de rutele sale esuate folosind mecanismul timeout. Dar urmele acestei rute persista in sistem pentru un timp indelungat. Inital, A si C cred ca mai pot merge spre D prin B. Deci trimit inca notificari inregistrand metrica 3. In urmatoarea iteratie, B va pretinde ca poate merge spre D prin A sau C. Desigur, nu poate. Rutele pretinse de A si C sunt acum cedate, dar ele nu au cum sa stie asta. Si chiar cand descopera ca rutele prin B sunt cazute, ele inca cred ca este o ruta disponibila prin alta parte. Eventual sistemul converge matematic, trebuie sa converga. Dar poate lua ceva timp pentru a face aceste lucru. Cel mai nefavorabil caz este cand retea devine complet inaccesibila din orice parte a sistemului. In

acest caz, metricele pot creste incet ca in exemplul de mai sus pana cand devin infinite. Din acest motiv, problema se numeste "calcul la infinit".

Acum intelegem de ce "infinitul" este ales sa fie cat mai mic posibil. Daca o retea devine complet inaccesibila dorim ca acest calcul la infinit sa fie oprit cat mai curand. "Infinitul" trebuie sa fie destul de mare astfel incat nici o ruta reala nu este atat de mare. Dar nu trebuie sa fie mai mare decat este nevoie. Astfel alegerea "infinitului" este o problema de alegere intre marimea retelei si viteza de convergenta cand are loc calculul la infinit. Designerii RIP-ului cred ca acest protocol este improbabil sa fie practicat pentru retele cu diametrul mai mare de 15.

Sunt cateva lucruri care pot fi facute pentru a preveni problemele de acest gen. Cele folosite de RIP se numesc "split horizon with poisoned reverse", si "actualizari declansate".

2.2.1. Split horizon

Observam ca problema de mai sus este cauzata de faptul ca A si C sunt angajate intr-un exemplu de directionare gresita reciproca. Fiecare pretinde sa o ia spre D prin cealalta. Aceasta poate fi prevenita prin existenta unui bit mai atent la destinatia informatiei. In particular, nu este niciodata folositor sa cerem accesibilitate la o retea, vecinilor de la care ruta a fost invatata. "Split horizon" este o schema pentru evitarea problemelor cauzate de includerea rutelor in notificari trimise gatewayului de la care le-au invatate. Schema "simple split horizon" omite rutele invatate de la un vecin din notificari trimise acestuia. Schema "Split horizon with poisoned reverse" include astfel de rute in notificari, dar isi seteaza metricele la infinit.

Daca A crede ca poate merge spre D prin C, mesajul sau catre C ar trebui sa indice ca D este inaccesibil. Daca ruta prin C este reala, atunci C ori are o conexiune directa cu D, ori o conexiune prin alt gateway. Ruta lui C nu se poate intoarce la A deoarece formeaza o bucla. Spunandu-i lui C ca D nu este accesibil, A reduce posibilitatea ca C sa

creada ca exista o ruta spre D prin A. Acest lucru e evident pentru o linie punct-la punct. Dar, sa consideram posibilitatea ca A si C sunt conectate printr-o retea de tip broadcast ca Ethernet-ul, si exista alte gatewaye pe acea retea. Daca A are o ruta catre C, ar trebui sa indice ca D este inaccesibil cand vorbeste cu oricare alt gateway din acea retea. Celelalte gatewaye din retea o pot lua ele in seama spre C. Nu vor avea niciodata nevoie sa o ia spre C prin A. Daca intr-adevar cea mai buna ruta a lui A este prin C, nici un alt gateway din retea nu are nevoie sa stie ca A poate ajunge la D. Acesta e un caz fericit, deoarece inseamna ca acelasi mesaj de notificare care este folosit pentru C poate fi folosit pentru toate celelalte gatewaye din aceeași retea. Astfel, mesajul de notificare poate fi trimis prin broadcast.

In general, Split horizon with poisoned reverse este mai sigur decat simple split horizon. Daca doua gatewaye au drumuri una prin cealalta, anuntarea rutelor inverse cu o metrica de 16 va intrerupe bucla imediat. Daca rutele de intoarcere pur si simplu nu sunt anuntate, rutele eronate vor fi eliminate prin asteptarea unui timeout. Oricum, poisoned reverse are un dezavantaj: creste marimea mesajului de rutare. Sa consideram cazul unei coloane vertebrale de campus care conecteaza un numar de cladiri diferite. In fiecare cladire, este un gateway care conecteaza magistrala la o retea locala. De remarcat ce actualizari de rutare ar trebui sa trimita gateway-urile pe retea backbone (coloana vertebrala). Tot restul retelei trebuie neaparat sa stie despre fiecare gateway la care retea locala este conectat. Folosind simple split horizon doar acele rute vor aparea in mesajele de notificare trimise de gateway retelei coloana vertebrala. Daca este folosit split horizon with the poisoned reverse gatewayul trebuie sa mentioneze toate rutele pe care le invata de la coloana vertebrala, cu metrica 16. Daca sistemul este mare, aceasta poate conduce la un mesaj de notificare mare, altfel toate acele intrari indica retele inaccesibile.

Intr-un sens statistic, anuntarea rutelor de intoarcere cu metrica 16, furnizeaza informatie neaditionala. Daca sunt multe gatewaye pe retea broadcast, aceste intrari suplimentare pot folosi largime de banda semnificativa. Motivul pentru care sunt acolo este de a imbunatati comportamentul dinamic. Cand topologia se schimba, mentionarea rutelor care nu ar trebui sa treaca prin gateway precum si acelora care ar putea mari convergenta. Oricum, in astfel de situatii, managerii retelelor ar putea prefera sa accepte oarecum convergenta in ceata pentru a minimiza (peste nivelul) [overhead](#)-ul rutarii. Astfel

programatorii pot in acest fel sa implementeze split horizon simple decat cu poisoned reverse, sau pot oferi o optiune de configuratie care permite administratorului de retea sa aleaga ce comportament sa foloseasca . De asemenea este permis sa implementeze scheme hibride care sa promoveze ruterelor de intoarcere cu o metrica de 16 si sa omita altele. Un exemplu de astfel de scheme ar fi sa foloseasca o metrica de 16 pentru rute de intoarcere pentru o anumita perioada de timp dupa ce se schimba rutarea care le implica, in consecinta omitandu-le din actualizari.

2.2.2. Actualizari declansate (Triggered updates)

Split horizon with poisoned reverse va preveni orice bucla de rutare care implica doar doua gateways. Oricum, este inca posibil sa se ajunga la exemplul in care trei gatewaye sunt angajate intr-o pacalire reciproca. De exemplu, A poate crede ca are un drum catre D, D catre C, si C catre A. Split horizon nu poate opri o astfel de ciclare. Aceasta ciclare poate fi rezolvata doar cand metrica ajunge infinita si reseaua implicata este declarata inaccesibila. Actualizarile declansate sunt o incercare de a mari aceasta convergenta. Pentru a obtine actualizari declansate, pur si simplu adaugam o regula care de fiecare data cand un gateway isi schimba metrica pentru o ruta, trebuie sa trimita un mesaj de actualizare aproape imediat, chiar daca nu este inca timpul pentru unul dintre mesajele de actualizare obisuite. (Detaliile de timp vor diferi de la un protocol la altul. Unele protocoale cu vectori de distanta, inclusiv RIP, specifica un timp scurt de intarziere, pentru a evita generarea actualizarilor declansate in exces.) Observam cum aceasta se combina cu regulile pentru calcularea noilor metrici. Presupunem ca ruta unui gateway spre destinatia N trece prin gatewayul G. Daca o actualizare ajunge de la insusi G, gatewayul care a primit-o este nevoit sa creada noua informatie, ca noua metrica este mai mare sau mai mica decat cea veche. Daca rezultatul este o schimbare in metrica, atunci gatewayul destinatar va trimite actualizari declansate tuturor gazdelor (hosts) si gatewayelor conectate direct la el. Acestia in transformare pot trimite fiecare actualizari vecinilor lor. Rezultatul este o cascada de actualizari declansate. Este usor de aratat care gateway si gazda este implicat in cascada. Presupunem ca un ruter G opreste o ruta spre destinatia N. G va trimite actualizari declansate catre toti vecinii sai. Cu toate acestea,

singurii vecini care vor crede noua informatie sunt aceia ale caror rute spre N sunt prin G. Celelalte gatewaye si gazde vor vedea aceasta ca pe o informatie despre o noua ruta care este mai rea decat cea pe care o au deja in folosinta, si o vor ignora. Vecinii a caror rute trec prin G isi vor schimba metricele si vor trimite actualizari declansate tuturor vecinilor lor. Din nou, doar acei vecini a caror rute trec prin ei vor fi atenti. Astfel, actualizarile declansate se vor propaga inapoi de-a lungul tuturor cailor din directia gateway-ului G, modificand metricele la infinit. Aceasta propagare se va opri atunci cand se va ajunge la o portiune din retea a carei ruta spre destinatia G o ia prin alta cale.

Daca sistemul ar putea fi facut sa stea linistit cat timp cascada de actualizari declansate are loc, ar fi posibil sa demonstram ca acest calcul la infinit nu se va intampla niciodata. Rutele proaste vor fi intotdeauna indepartate imediat, si astfel nici o bucla de rutare nu s-ar forma.

Din nefericire, lucrurile nu sunt atat de simple. Atata timp cat actualizarile declansate sunt trimise, actualizarile normale pot fi trimise in acelasi timp. Gateway-urile care nu au primit actualizarea declasata inca vor trimite inca informatii bazate pe rute care deja nu mai exista. Este posibil ca dupa ce un actualizarea declasata a trecut de un gateway, acesta ar putea primi o notificare normala de la unul din acele gateway-uri (gatewaye) care nu au primit mesajul. Aceasta ar putea reconstrui o ramasita a rutei pierdute. Daca actualizarile declansate sunt destul de rapide, aceasta este foarte improbabil. Oricum, calculul la infinit este inca posibil.

3. Specificatii de protocol

RIP intentioneaza sa permita gazdelor si gateway-urilor sa schimbe informatii pentru calcularea rutelor printr-un IP de baza al retelelor. RIP este un protocol cu vectori distanta. Astfel, avem trasaturile generale descrise in sectiunea 2. RIP poate fi implementata de ambele: gazde si gateway-uri(porti). Ca in toate documentatiile IP, termenul „gazda” (host) este folosit pentru a exprima informatii despre rute la „destinatii”, care pot fi gazde individuale, retele, sau destinatii speciale folosite pentru a denumi o ruta care nu mai este valabila. Orice gazda care foloseste RIP se presupune ca are o interfata cu una sau mai multe retele. Aceasta se refera la „retelele direct conectate”.

Protocolul faciliteaza accesul la anumite informatii despre fiecare retea. Cea mai importanta este metrica sau „costul”. Metrica unei retele este un intreg intre 1 si 15 inclusiv. Este setata intr-o maniera nespecificata in acest protocol. Cele mai multe dintre implementarile existente folosesc intotdeauna metrica 1. Noile implementari ar trebui sa permita administratorului sa seteze metrica fiecarei retele. In plus, pe langa cost fiecare retea va avea o adresa IP si o masca subnet asociata cu ea. Acestea trebuie setate de administratorul de sistem intr-o maniera nespecificata in acest protocol.

Observam ca regulile specificate in sectiunea 3.2. presupun ca este o singura masca subnet aplicata fiecarei adrese IP, si ca doar mastile subnet ale retelelor direct conectate sunt cunoscute. Pot fi sisteme care folosesc masti subnet diferite pentru subretele diferite ale unei singure retele. Pot fi deasemenea instante unde este de dorit ca un sistem sa cunoasca mastile subnet ale retelelor departate. Oricum, astfel de situatii vor necesita modificari ale regulilor care guverneaza domeniul informatiei subretelelor. Astfel de modificari cauzeaza consecintele interoperabilitatii, si astfel trebuie vazuta ca o modificare a protocolului.

Fiecare gazda care implementeaza RIP isi atribuie dreptul de a avea o tabela de rutare. Aceasta tabela are o singura intrare pentru fiecare destinatie care este disponibila prin sistemul descris de RIP. Fiecare intrare contine cel putin informatiile urmatoare:

- adresa IP a destinatiei.
- o metrica, care reprezinta costul total a trecerii unei datagrame de la gazda la destinatie. Aceasta metrica este suma tuturor costurilor asociate cu retelele care vor fi traversate spre destinatie.
- Adresa IP a gateway-ului(portii sau gatewayului) urmator de-a lungul drumului spre destinatie. Daca destinatia este una dintre retelele direct conectate, acest item nu este necesar.
- Un flag pentru a indica acele informatii pe care ruta le-a schimbat recent. Acesta va fi folosit ca „flag de schimbare a rutei”.
- Timpi diferiti asociati rutei. Vezi sectiunea 3.3 pentru mai multe detalii despre aceasta.

Intrarile pentru retelele direct conectate sunt setate de gazda, folosind informatii stranse prin mijloace nespecificate in acest protocol. Metrica pentru retelele direct conectate este setata la costul acelei retele. In implementarile RIP existente, 1 este intotdeauna folosit pentru cost. In acest caz, metrica RIP se reduce la o simpla numarare a hopurilor. Metricele mai complexe pot fi folosite cand este de preferat sa aratam preferinte pentru anumite retele, de exemplu pentru diferentele de largime a benzii sau siguranta.

Programatorii pot deasemenea alege sa permita administratorului de sistem sa introduca rute aditionale. Acestea mai degraba vor fi routate catre gazde sau retele dinafara razei de actiune a sistemului de rutare.

Intrarile pentru alte destinatii decat cele initiale sunt adaugate si actualizate prin algoritmi descriși in sectiunile urmatoare.

Pentru ca protocolul sa furnizeze informatii complete despre rutare, fiecare gateway din sistem trebuie sa participe. Gazdele care nu sunt gateway nu trebuie sa participe, dar multe implementari iau masuri de prevenire pentru ca ele sa asculte informatiile de rutare cu scopul de a le permite sa-si mentina tabelele de rutare.

3.1. Formatul mesajelor

RIP este un protocol UDP. Fiecare gazda care foloseste RIP are un proces de rutare care trimite si primeste datagrame prin portul UDP numarul 520. Toate comunicari indreptate spre prosoarele RIP ale altor gazde sunt trimise spre portul 520. Toate mesajele de notificare ale routarii sunt trimise spre portul 520. Mesajele de notificare si rutare nesolicitate au portul sursa si destinatie setate 520. Acelea trimise ca raspuns unei cereri sunt trimise la portul de la care vine cererea. Intrebarile si cererile de debugging specifice pot fi trimise spre alte porturi decat 520, dar ele sunt indreptate spre portul 520 la masina tinta.

Sunt masuri in protocol pentru permiterea proceselor RIP „mute”. Un proces mut este unul care in mod normal nu trimite nici un fel de mesaje. Cu toate acestea, asculta mesajele trimise de altii. Un proces RIP mut poate fi folosit de gazdele care nu se comporta ca niste gateway-uri, dar doresc sa asculte mesajele de notificare in scopul de a

monitoriza gateway-urile locale si de a pastra tabelele de rutare interne la zi. Un gateway care a pierdut legatura cu toti dar una dintre retelele sale poate alege sa devina muta, din momentul in care mai este gateway.

Oricum, aceasta nu ar trebui facuta daca exista orice sansa ca gateway-urile vecine sa depinda mai departe de mesajele sale pentru a detecta care dintre retelele cazute a revenit in actiune. (Programul de rutare 4BSD foloseste pachetele de rutare pentru a monitoriza operatiunile legaturilor punct cu punct).

Formatul pachetului este aratat in figura 1.

Formatul datagramelor care contin informatii despre retele. Marimea campurilor este data in octeti. Daca nu altfel specificate, campurile contin intregi binari, in conventia normala pentru Internet cu cel mai semnificativ octet primul. Fiecare marca reprezinta un bit.

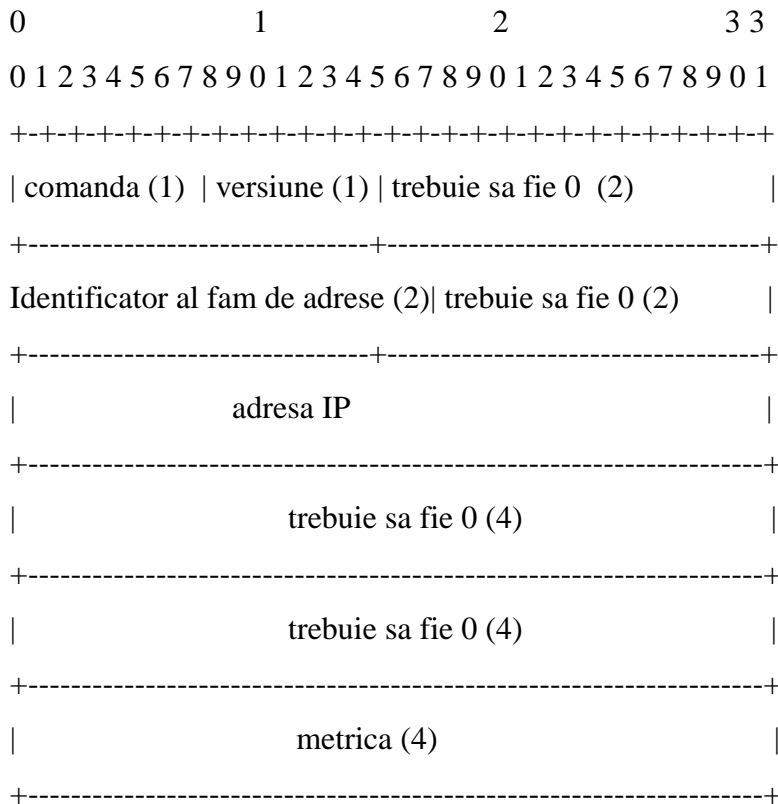


Figura 1. Formatul pachetelor.

Portiunea de datagrama dintre identificatorul familiei de adrese si metrica poate aparea de pana la 25 de ori. Adresa IP este adresa Internet obisnuita de 4 octeti, in conventia retelelor.

Fiecare datagrama contine o comanda, un numar de versiune, si argumentele posibile. Acest document descrie versiunea 1 a protocolului. Detalii ale versiunilor de procesare sunt descrise in sectiunea 3.4. Campul comanda este folosit pentru a specifica scopul acestei datagrame. Aici este un sumar al comenzilor implementate in versiunea 1:

1 – cerere O cerere pentru sistemul raspunzator de a trimite toata sau o parte din tabela de rutare.

2 – raspuns Un mesaj continand toata sau o parte a tabelii de rutare a trimitatorului. Acest mesaj poate fi trimis ca raspuns la o cerere, sau poate fi un mesaj de notificare generat de expeditor.

3 – traceon, invechit. Mesajele continand aceasta comanda sunt ignorate.

4 – traceoff, invechit. Mesajele continand aceasta comanda sunt ignorate.

5 – rezervat. Aceasta comanda este folosita de Sun Microsystems pentru scopuri proprii. Daca sunt aduse comenzi noi in versiunile urmatoare, trebuie sa inceapa cu 6. Mesajele continand aceasta comanda ar trebui ignorate de implementarile care nu aleg sa raspunda la ele.

Pentru cerere si raspuns restul datagramei contine o lista de destinatari, cu informatii despre fiecare. Fiecare intrare din aceasta lista contine o retea destinatie sau o gazda, si metrica pentru ea. Formatul pachetului intentioneaza sa permita RIP-ului sa transporte informatiile de rutare pentru cateva protocole diferite. Astfel, fiecare intrare are un identificator al familiei de adrese pentru a indica ce tip de adresa este specificata in acea intrare. Acest document descrie rutarea doar pentru retelele de Internet. Identificatorul familiei de adrese pentru IP este 2. Nici una din implementarile RIP disponibile nu permite programatorului sa foloseasca un alt tip de adrese. Oricum, pentru a permite dezvoltari viitoare, implementarile sunt realizate sa omita intrarile care specifica familii de adrese care nu sunt suportate de implementari. (Marimea acestor intrari va fi aceeaasi cu marimea intrarilor specificate in adresele IP.) Procesarea mesajului continua normal dupa ce orice intrare nesuportata este sarita. Adresa IP este adresa normala Internet, memorata pe 4 octeti in ordinea retelei. Campul metrica trebuie sa contina o valoare intre 1 si 15 inclusiv, specificand metrica curenta pentru destinatie, sau 16 cand indica faptul

ca destinatia nu este accesibila. Fiecare ruta trimisa de un gateway suprima oricare ruta anterioara spre aceeasi destinatie de la acelasi gateway.

Marimea maxima a unei datagrame este de 512 octeti. Aceasta include doar portiunea de datagrama descrisa mai sus. Nu conteaza antentele IP sau UDP. Comenzile care implica informatii retea permit informatiei sa fie impartita intre mai multe datagrame. Nu sunt necesare masuri de precautie speciale, odata ce se produc rezultate corecte daca datagramele sunt procesate individual.

3.2. Consideratii de adresare

Cum este indicat in sectiunea 2, rutarea cu vectori de distanta poate fi folosita pentru a descrie rutele catre gazde individuale sau retele. Protocolul RIP permite oricare dintre aceste posibilitati. Destinatiile aparute in mesajele de cerere si raspuns pot fi retele, gazde sau codificari speciale folosite pentru a indica o adresa inaccesibila. In general, tipurile de rute folosite acum vor depinde de strategia de rutare folosita pentru retelele particulare. Multe retele sunt setate astfel incat informatia de rutare nu este necesara pentru gazde individuale. Daca fiecare gazda dintr-o retea sau subretea data este accesibila prin aceleasi gateway, atunci nu avem motive sa mentionam gazde individuale in tabela de routare. Oricum, retelele care includ linii punct cu punct cer cateodata gateway-urilor sa pastreze urma rutei catre o anumita gazda. Daca acest lucru este cerut sau nu depinde de abordarea adresarii sau rutarii folosita in sistem. De aceea unele implementari pot alege sa nu suporte rutele gazdelor. Daca rutele gazdelor nu sunt suportate, ele vor fi ignorate cand sunt receptionate in mesajele de raspuns. (Vezi sectiunea 3.4.2.).

Formatul pachetelor RIP nu distinge intre tipuri variate de adrese. Campurile care sunt etichetate „adresa” pot contine oricare dintre tipurile de mai jos:

Adresa gazdei

Numar subretea

Numar retea

0, indicand o ruta inaccesibila

Entitatile care folosesc RIP accepta sa foloseasca informatia cea mai specifica disponibila cand ruteaza o datagrama. Aceasta inseamna ca atunci cand ruteaza o

datagrama adresa sa de destinatie trebuie mai intai comparata cu lista de adrese ale gazdelor. Trebuie verificat pentru a vedea daca se potriveste cu vreun numar de retea sau subretea cunoscut. In final, daca niciunul nu se potriveste este considerata inaccesibila.

Cand o gazda evalueaza informatiile primite prin RIP, interpretarea sa asupra unei adrese depinde daca ea cunoaste masca subretea aplicabila retelei. Daca da, atunci este posibil sa determine intelesul adresei. De exemplu, sa consideram reteaua 128.6. Are o masca subretea de 255.255.255.0. Astfel 128.6.0.0 este un numar de retea, 128.6.4.0 este un numar de subretea, si 128.6.4.1 este o adresa de gazda. Cu toate acestea, daca gazda nu cunoaste masca subretea, evaluarea unei adrese poate fi ambigua. Daca este o parte de numar de gazda diferita de zero, nu este clar de ce sa determinam daca adresa reprezinta un numar subretea sau o adresa de gazda. Asa cum un numar retea este nefolositor fara masca subretea, adresele se presupune ca reprezinta gazde in aceasta situatie. In incercarea de a evita acest tip de ambiguitati, gazdele nu trebuie sa trimita rute subretea la gazde care nu este de asteptat sa cunoasca masti subretea apropiate. In mod normal gazdele stiu doar masti subretea pentru retelele direct conectate. Pentru aceasta, daca nu au fost facute previziuni speciale, nu trebuie trimise rute catre o subretea decat daca aceasta face parte din acea retea.

Aceasta filtrare este realizata de gateway-uri (porti) la „marginea” retelei cu subretele (subnetted network). Acestea sunt gateway-urile care conecteaza acea retea cu oricare alta retea. In reteaua cu subretele, fiecare subretea este tratata ca o retea individuala. Intrarile de routare pentru fiecare subretea sunt indrumate de RIP. Oricum, poarta de margine trimite doar o singura intrare pentru retea ca un intreg pentru gazdele din alta retea. Aceasta inseamna ca o poarta de margine va trimite informatii diferite la vecini diferiti. Pentru vecinii conectati la reteaua cu subretele, se genereaza o lista cu toate subretelele care sunt conectate direct, folosind numarul subretea. Pentru vecinii conectati la alte retele, realizeaza o singura intrare pentru retea ca un intreg, aratind metrica asociata cu acea retea. (Aceasta metrica va fi in mod normal cea mai mica metrica pentru subretelele la care gateway-ul este atasat.)

Similar, portile de margine nu trebuie sa mentioneze rutele gazdelor pentru gazdele uneia dintre retelele direct conectate in mesajele catre alte retele. Aceste rute vor fi subsumate de o singura intrare pentru retea ca un intreg. Nu specificam ce trebuie facut

cu rutele gazdelor pentru gazde indepartate (i.e. gazde care nu fac parte dintr-o retea direct conectata). In general aceste rute indica niste gazde care sunt printr-o ruta care nu suporta alte gazde in retea din care gazda face parte.

Adresa speciala 0.0.0.0 este folosita pentru a descrie o ruta necunoscuta. O ruta necunoscuta este folosita atunci cand nu este convenabil sa insiruiem fiecare retea posibila in notificările RIP, si cand una sau mai multe gateway-uri conectate in apropiere din sistem sunt pregatite sa dirijeze traficul spre retelele care nu sunt inregistrate explicit. Aceste gateway-uri ar trebui sa creeze intrari RIP pentru adresa 0.0.0.0, doar daca exista o retea cu care ele sunt conectate. Decizia cum gateway-urile creeaza intrari pentru 0.0.0.0 este lasata implementarii. Cel mai comun, administratorul sistemului va fi inzestrat cu o cale de a specifica ce gateway ar trebui sa creeze intrari pentru 0.0.0.0. Oricum, alte mecanisme sunt posibile. De exemplu, un programator ar putea decide ca orice gateway care vorbeste EGP ar trebui declarat ca fiind un astfel de gateway. Ar putea fi folositor sa permitem administratorului de retea sa aleaga metrica folosita pentru astfel de intrari. Daca este mai mult de un default gateway, aceasta ar trebui sa faca posibila exprimarea unei preferinte pentru unul sau altul. Intrarile 0.0.0.0 sunt tratate de RIP in exact aceeasi maniera ca si cum ar fi o retea actuala cu aceasta adresa. Cu toate astea, intrarea este folosita pentru a dirija orice datagrama a carei adresa de destinatie nu se potriveste nici unei alte retele din tabela. Implementarile nu sunt obligate sa suporte aceasta conventie. Cu toate astea, este foarte recomandat. Implementarile care nu suporta 0.0.0.0 trebuie sa ignore intrarile cu aceasta adresa. In acest caz, nu trebuie sa le treaca mai departe in notificările RIP proprii. Astfel, rutele care implica 0.0.0.0 nu ar trebui in general sa paraseasca granita unui sistem autonom. Mecanismele pentru aplicarea acestora nu sunt specificate in acest document.

3.3. Timerii

Aceasta sectiune descrie toate evenimentele care sunt tratate de timeri.

La fiecare 30 de secunde, procesul de iesire este instruit sa genereze un raspuns complet pentru fiecare gateway vecin. Cand sunt multe gateway-uri la o singura retea, exista o tendinta de a se sincroniza intre ele astfel incat toate trimit actualizari in acelasi

moment. Aceasta se poate intampla de fiecare data cand timerul de 30 de secunde este afectat de incarcarea procesului in sistem. Nu este de dorit ca mesajele de actualizare sa devina sincronizate, deoarece pot conduce la coliziuni nedorite in retelele de tip broadcast. Astfel, implementarile trebuie sa ia una din precautiile:

- Actualizarile de 30 de secunde sunt tratate de un ceas a carui rata nu este afectata de incarcarea sistemului sau de cererea de timp pentru a servi timerul de actualizare anterior.

- Timerul de 30 de secunde este inchis prin aderarea la un mic timp aleator de fiecare data cand este setat.

Sunt doi timeri asociati cu fiecare ruta, un „timeout” si un „timp colector de gunoi”. Dupa expirarea timpului „timeout”, ruta nu mai este valida. Cu toate acestea, este retinuta in tabela pentru scurt timp, astfel incat vecinii sa poata fi anuntati ca ruta a fost stearsa. Dupa expirarea timpului in care ruta este retinuta, timpul colector de gunoi, ruta este stearsa cu adevarat din tabela.

Timpul timeout este initializat cand o ruta este stabilita, si oricand un mesaj de notificare este primit din partea rutei. Daca trec 180 secunde de cand timeout a fost initializat ultima data, ruta este considerat ca a expirat, si procesul de stergere pe care il vom descrie este inceput.

Stergerea poate fi facuta din doua motive: (1) timeout expira, sau (2) metrica este setata la 16 pentru ca a fost primit un mesaj de notificare de la gateway-ul curent. (Vezi sectiunea 3.4.2 despre discutia procesarii notificarilor de la alte gateway-uri.) Altfel, unul dintre evenimentele de mai jos s-a intamplat:

- Timerul colector de gunoi este setat la 120 secunde.
- Metrica pentru ruta este setata la 16 (infinit). Aceasta cauzeaza stergerea rutei din tabela.
- Este setat un flag anuntand ca aceasta intrare a fost schimbata, si procesul de iesire este semnalat sa trimita un raspuns.

Pana cand timerul colector de gunoi expira, ruta este inclusa in toate notificarile trimisa de aceasta gazda, cu metrica 16 (infinit). Cand timerul colector de gunoi expira, ruta este stearsa din tabela.

Ar trebui ca o noua ruta la aceasta retea sa fie stabilita atata timp cat timerul colector de gunoi ruleaza, noua ruta o va inlocui pe cea care este pe cale de a fi stearsa. In acest caz timerul colector de gunoi trebuie sa fie eliberat.

Vezi sectiunea 3.5 cu privire la intarzierea care este ceruta pentru procesarea notificarilor. Desi implementarea acestei intarzieri va necesita un timer, este mai natural sa o discutam in sectiunea 3.5 decat aici.

3.4. Procesarea intrarilor

Aceasta sectiune va descrie manipularea datagramelor primite pe un port 520. Inaintea procesarii datagramii in detaliu, trebuie facute cateva verificari generale asupra formatului. Aceasta depinde de campul cu numarul versiunii din datagrama, ca mai jos:

0 Datagramele ale caror numar de versiune este 0 vor fi ignorate. Acestea sunt de la o versiune anterioara a protocolului, a carui format de pachet era specific-masina.

1 Datagramele ale caror numar de versiune este 1 vor fi procesate si sunt descrise in urmatoarea parte a acestor specificatii. Toate campurile care sunt descrise mai sus ca „trebuie sa fie zero” vor fi verificate. Daca vreunul din aceste campuri este o valoare diferita de zero, intregul mesaj va fi ignorat.

>1 Datagramele ale caror numar de versiune este mai mare ca 1 vor fi procesate si sunt descrise in urmatoarea parte a acestor specificatii. Toate campurile care sunt descrise mai sus ca „trebuie sa fie zero” vor fi ignorate. Versiunile viitoare ale acestui protocol ar putea pune date in aceste campuri. Implementarile versiunea 1 vor ignora aceste date in plus si vor procesa doar campurile specificate in acest document.

Dupa verificarea numarului versiunii si realizarea celorlalte verificari preliminare, procesarea va depinde de valoarea din campul comanda.

3.4.1. Cererea

Cererea este folosita pentru a solicita un raspuns care sa contina toata sau o parte din tabela de routare a gazdei. [Observam ca termenul gazda este folosit pentru oricare gazda sau gateway, in cele mai multe cazuri ar fi neobisnuit ca o gazda care nu este gateway sa trimita mesaje RIP.] In mod normal, cererea este trimisa la toate destinatiile (broadcast), de la un port UDP sursa 520. In acest caz, procesele tacute nu raspund la cerere. Procesele tacute sunt prin definitie procese care in mod normal nu vrem sa vada informatiile de routare. Cu toate astea, pot fi situatii care implica monitorizarea gateway-ului unde este de dorit sa fie vazuta tabela de rutare chiar si de un proces tacut. In acest caz, cererea ar trebui trimisa de la un port UDP diferit de 520. Daca o cerere vine de la portul 520, procesele tacute nu raspund. Daca cererea vine de la oricare alt port, procesele trebuie sa raspunda chiar daca sunt tacute.

Cererea este procesata intrare cu intrare. Daca nu sunt intrari, nu este dat nici un raspuns. Aici este un caz special. Daca este exact o intrare in cerere, cu un identificator de familie de adrese 0 (insemnand nespecificat), si o metrica de infinit (i.e. 16 pentru implementarile curente), aceasta este o cerere pentru trimiterea intregii tabele de rutare. In acest caz, este facuta o cerere la procesul de iesire sa trimita tabela de rutare la portul cerut.

Exceptand acest caz special, procesarea este simpla. Mergem in jos in lista intrarilor din cerere una cate una. Pentru fiecare intrare, vedem destinatia din datagrama de rutare a gazdei. Daca este vreo ruta, punem metrica acestei rute in campul metrica din datagrama. Daca nu este nici o ruta la destinatia specificata, punem infinit (i.e. 16) in campul metrica din datagrama. Odata ce toate intrarile au fost completate, setam comanda pe raspuns si trimitem datagrama inapoi la portul de la care a venit.

Observam ca exista o diferenta in interpretare depinzand daca cererea este pentru un set specificat de destinatii, sau pentru intreaga tabela de rutare. Daca cererea este pentru intreaga tabela de rutare, este data iesirea normala a procesarii. Aceasta include split horizon (vezi sect. 2.2.1) si ascunderea subretelelor (sect. 3.2), deci aceste intrari oarecare din tabela de rutare nu vor fi vazute. Daca cererea este pentru intrari specificate, va fi verificat in tabela gazdei si informatia va fi trimisa daca este ceruta. Noi anticipam ca

aceste cereri vor fi probabil folosite pentru diferite Toate campurile care sunt descrise mai sus ca „trebuie sa fie zero” vor fi verificate. Cand o gazda apare pentru prima data, va trimite cereri la fiecare retea conectata cerand intrega tabela de rutare. In general, presupunem ca intreaga tabela de rutare este ceruta pentru a fi folosita pentru innoirea tabelii de rutare a altei gazde. Pentru acest motiv, split horizon si toate celelalte filtrari trebuie folosite. Cererile pentru retele specifice sunt facute doar prin diagnosticarea software-lui, si nu sunt folosite pentru rutare. In acest caz, cel care face cererea va dori sa stie exact continutul datagramei de rutare, si nu va dori ca nici o informatie sa fie ascunsa.

3.4.2. Raspunsul

Raspunsuri pot fi primite din mai multe motive:

- raspuns la o cerere specifica
- actualizare normala
- actualizare generata de modificarea metricii

Procesarea este aceeaasi indiferent cum a fost generat raspunsul.

Deoarece procesarea raspunsului poate sa schimbe tabela de rutare a gazdei, raspunsul trebuie sa fie verificat atent pentru validitate. Raspunsul trebuie sa fie ignorat daca nu este de la portul 520. Adresa IP a sursei trebuie verificata pentru a vedea daca datagrama este de la un vecin valid. Sursa datagramei trebuie sa fie de la o retea direct conectata. Este deasemenea bine de vazut daca raspunsul este de la una din adresele proprii ale gazdelor. Interfetele pe retele broadcast pot primi copii de la broadcast-urile lor imediat. Daca o gazda proceseaza propria sa iesire ca o noua intrare, probabil este o confuzie, si astfel de datagrame trebuie ignorate (cu exceptia celor discutate in paragraful urmator).

Inainte de a procesa cu adevarat un raspuns, ar putea fi folositor sa utilizam prezenta sa ca o intrare a unui proces pentru pastrarea urmei starii interfetei. Cum am mentionat mai sus, punem o ruta pe timeout cand nu am auzit de gazda sa de un anumit timp. Aceasta merge bine pentru rute care vin de la alte gazde. Este de asemenea bine de stiut cand una din retelele direct conectate a cazut. Acest document nu specifica nici o metoda

particulara pentru a realiza aceasta, metode ca acestea depind de caracteristicile rețelei și de interfața hardware a acestora. Cu toate acestea, astfel de metode implică adesea ascultarea datagramelor sosite la interfața. Sosirea datagramelor poate fi folosită ca o indicație că interfața funcționează. Cu toate acestea, trebuie luate câteva precauții, cum este posibil ca interfețele să esueze în acest mod în care datagramele de intrare sunt primite, datagramele de ieșire nu sunt niciodată trimise cu succes.

Acum că datagrama a fost validată în întregime, procesăm intrările din ea una câte una. Din nou, pornim prin realizarea validării. Dacă metrica este mai mare decât infinit (16), ignorăm acea intrare. (Aceasta ar trebui să fie imposibil dacă cealaltă gazdă funcționează corect. Metrici incorecte sau alte erori de format ar trebui să genereze alerte.) Apoi ne uităm la adresa de destinație. Verificăm identificatorul familiei de adrese. Dacă nu este o valoare așteptată (e.g. 2 pentru adresele Internet), ignorăm acea intrare. Acum verificăm adresa pentru categorii diferite de adrese nepotrivite. Ignorăm intrările dacă adresa este clasa D sau E, dacă este la net 0 (cu excepția 0.0.0.0, dacă acceptăm rute necunoscute) sau dacă este la net 127 (rețeaua loopback). De asemenea verificăm adresa de broadcast, i.e., oricare adresă pentru care partea referitoare la host este formată numai din 1 într-o rețea ce suportă broadcast, și ignorăm astfel de intrări. Dacă programatorul a ales să nu suporte rutele gazdelor (vezi secțiunea 3.2), verificăm să vedem dacă porțiunea pentru gazdă din adresă este diferită de zero; dacă da ignorăm intrarea.

Anulam acele câmpuri de adresă care conțin un număr de octeți nefolositi. Dacă numărul de versiune al datagramii este 1, trebuie de asemenea verificate. Dacă vreuna dintre ele este diferită de zero, intrarea trebuie ignorată. (Multe din aceste cazuri indică faptul că gazda de la care vine mesajul nu funcționează corect. Astfel unele forme de erori de logare sau alerte ar trebui oprite.)

Modificăm metrica prin adăugarea rețelei la care ajunge mesajul. Dacă rezultatul este mai mare decât 16, folosim 16. Aceasta este:

$$\text{metrica} = \text{MIN}(\text{metrica} + \text{costul}, 16)$$

Acum ne uităm la adresa pentru a vedea dacă este deja o rută pentru aceasta. În general, dacă nu, vrem să adăugăm una. Oricum, sunt diferite excepții. Dacă metrica este infinită, nu adăugăm o intrare. (Am putea modifica una existentă, dar nu adăugăm o nouă intrare cu metrică infinită.) Vrem să evităm adăugarea de rute pentru gazde sau subrețele

pentru care avem deja o ruta cel puțin la fel de bună. Dacă nu apare nici una din aceste excepții adăugăm o nouă intrare în datagrama de rutare. Aceasta presupune următoarele acțiuni:

- Setarea destinațiilor și metricelor pentru aceasta în datagrama.
- Setăm gateway-ul ca fiind gazda de la care vine datagrama.
- Inițializăm timpul de expirare (timeout) pentru ruta. Dacă timerul colector de gunoi merge pentru această ruta, o oprim. (Vezi sect. 3.3 pentru discuția despre timeri.)
- Setăm flagul de schimbare al rutei, și semnalizăm procesului de ieșire să genereze un mesaj de actualizare (vezi 3.5).

Dacă există deja o ruta, mai întâi comparăm gateway-urile. Dacă datagrama este de la același gateway ca și ruta existentă, reinițializăm timpul de expirare. Apoi comparăm metricele. Dacă datagrama este de la același gateway ca și ruta existentă și metrica nouă este diferită de cea veche, sau dacă metrica nouă este mai mică decât cea veche, realizăm următoarele acțiuni:

- adoptăm ruta din datagrama. Deci, punem noua metrică aici, și setăm gateway-ul ca fiind gazda de la care vine datagrama.
- Inițializăm timpul de expirare pentru ruta.
- Setăm flagul de schimbare al rutei, și semnalizăm procesului de ieșire să genereze un semnal (vezi 3.5).
- Dacă noua metrică este 16 (infinit), este început procesul de ștergere.

Dacă noua metrică este 16, aceasta începe procesul pentru ștergerea rutei. Ruta nu mai este folosită pentru pachetele de rutare, și este pornit timerul de ștergere (vezi sect. 3.3). Observăm că procesul de ștergere este pornit doar atunci când metrica este setată de la început 16. Dacă metrica era deja 16, atunci nu este pornită o nouă ștergere. (Pornirea ștergerii setează un timer. Problema este că nu vrem să resetăm timerul la fiecare 30 de secunde, cum sosesc noile mesaje cu metrică infinită.)

Dacă noua metrică este la fel cu cea veche, este simplu să nu facem nimic suplimentar (înafară de reinițializarea timpului de expirare, cum am specificat mai sus). Cu toate acestea, rutările 4BSD folosesc o heuristică suplimentară aici. În mod normal, este fără sens să schimbăm ruta existentă cu una cu aceeași metrică dar printr-un gateway

diferit. Dacă ruta existentă arată semne de expirare, totuși, ar fi mai bine să schimbăm cu o ruta alternativă egală dar mai bună imediat, decât să așteptăm ca aceasta să expire. (Vezi sect. 3.3 pentru discuția despre timpii de expirare.) Așadar, dacă noua metrică este aceeași cu cea veche, rutările se uita la timpii de expirare pentru ruta existentă. Dacă este măcar la jumătatea timpului de expirare, ruta este schimbată cu cea nouă. Aceasta reprezintă schimbarea gateway-ului cu sursa mesajului curent. Aceasta euristica este opțională.

Orice intrare care esuează la aceste teste este ignorată, considerând că nu este mai bună decât vechea ruta.

3.5. Procesarea ieșirilor

Această secțiune descrie procesarea folosită în crearea mesajelor de răspuns care conțin tot sau o parte din tabelul de rutare. Această procesare poate fi declanșată de oricare din următoarele cazuri:

- de procesul de intrare, când este întâlnită o cerere. În acest caz, mesajul rezultat este trimis la o singură destinație

- de actualizarea regulată a rutării. La fiecare 30 de secunde, este trimis un răspuns la toți vecinii, care conține toate tabelele de rutare. (Vezi secțiunea 3.3.)

- de actualizarea declanșată. De fiecare dată când este schimbată metrică pentru o ruta, este declanșată o actualizare. (Actualizarea poate fi întârziată; vezi mai jos.)

Înainte de a descrie modul în care sunt generate mesajele pentru fiecare rețea direct conectată, vom comenta cum sunt alese destinațiile pentru cele două cazuri.

În mod normal, când este trimis un răspuns la toate destinațiile (adică, ori este pregătită o actualizare normală ori una declanșată), este trimis un răspuns la gazda de la polul opus la fiecare legătură conectată punct la punct., este difuzat un răspuns la toate rețelele conectate care suportă transmiterea. Astfel, este pregătit un răspuns pentru fiecare rețea direct conectată și trimis la adresa corespunzătoare (destinație sau broadcast). În majoritatea cazurilor, aceasta ajunge la toate gateway-urile învecinate. Totuși, sunt unele cazuri în care aceasta nu e de ajuns. Aceasta poate implica o rețea care nu suportă broadcast (e.g., ARPANET-ul), sau o situație care implică gateway-uri invalide. În astfel

de cazuri, este posibil sa fie necesar sa se specifice o lista a gazdelor si portilor vecine, si sa trimita o datagrama le fiecare in parte. Este lasat programatorului sa determine daca este nevoie de un astfel de mecanism, se sa defineasca cum este specificata lista.

Actualizarile declansate necesita o tratare speciala din doua motive. Primul, experienta arata ca actualizarile declansate pot cauza incarcaturi excesive in retele cu capacitate redusa. Astfel protocolul necesita ca programatorii sa includa provizii pentru limitarea frecventei „actualizarilor declansate” (triggered updates). Dupa ce este trimisa o actualizare declansata, ar trebui setat un contor la un interval de la 1 la 5 secunde. Daca au loc alte schimbari care ar declansa refresh-ul, inainte expirarea contorului, este declansat un singur refresh cand expira contorul, si acesta este setat pe alta valoare aleatoare intre 1 si 5. Actualizarile declansate ar putea fi suprimate daca o actualizare normala este pregatita exact pentru timpul in care este trimisa o actualizare declansata.

Al doilea, actualizarile declansate nu necesita includerea intregii tabele de rutare. In principiu, numai acele rute care au fost schimbate trebuie incluse. Astfel mesajele generate ca o parte a actualizarilor declansate, trebuie sa includa macar acele rute care au flagul de schimbare al rutei setat.

Pot contine si rute aditionale, sau toate rutele, la discretie programatorului; totusi, cand actualizarea completa a rutarii necesita pachete multiple, nu este recomandata trimiterea tuturor rutelor. Cand este procesata o actualizare declansata, mesajele ar trebuie generate pentru fiecare retea conectata direct. Procesarea split horizon este realizata cand sunt generate atat actualizari declansate cat si actualizarile normale. (vezi mai jos).

Daca dupa procesarea split horizon, va aparea o ruta identica schimbata intr-o retea cum s-a intamplat mai devreme, ruta va trebui trimisa; daca, ca rezultat, nu trebuie trimisa nici o ruta, actualizarile pot fi omise in acea retea. (Daca o ruta a avut doar o metrica schimbata, sau foloseste o poarta noua care este in aceeasi retea cu poarta veche, ruta va fi trimisa la retea portii vechi cu o metrica infinita, inainte si dupa schimbare.)

Odata ce au fost generate toate actualizarile declansate, flagurile pt schimbarea rutelor trebuie inlaturate.

Daca procesarea datelor de intrare este permisa in timp ce este generata iesirea, trebuie realizat un interblocaj adecvat. Flagurile de schimbare a rutei nu ar trebui

schimbate ca un rezultat a procesarii intrarii in timp ce este generat un mesaj de actualizare declansata.

Singura diferenta intre o actualizare declansata si alt mesaj actualizat este posibila omitere a rutelor care nu au fost schimbate. Restul mecanismelor, care vor fi descrise, trebuie aplicate la toate actualizarile declansate.

Aici este cum o datagrama de raspuns este generata pentru o retea direct conectata, mai speciala:

Sursa adresei de IP trebuie sa fie adresa gazdei de pe acea retea care transmite datele.

Acest lucru este important deoarece sursa adresei este pusa in tabela de rutare la alte gazde. Daca este folosita o sursa de adresa incorecta, alte gazde nu vor putea ruta datagrame. Cateodata gateway-urilor le sunt atribuite mai multe IP-uri pe o singura interfata fizica. In mod normal, aceasta inseamna ca mai multe din cele cateva retele logice IP sunt scoase inafara mediului fizic. In astfel de cazuri, trebuie trimis un mesaj de actualizare separat, cu acea adresa ca sursa de adresa IP.

Setati numarul versiunii curente a Rip-ului. (Versiunea descrisa aici este 1.) Setati comanda sa raspunda. Setati biti marcati cu „trebuie sa fie 0” cu 0. Acum incepeti sa umpleti intrarile.

Pentru a umple intrarile, verificati toate rutele in tabela tabelelor de intare. Reamintiti-va ca marimea maxima a datagramei este 512 biti. Atunci cand nu mai este loc in datagrama, trimiteti mesajul curent si incepeti unul nou. Daca este generata o actualizare declansata, trebuie incluse doar intrarile a caror flag de schimbare a rutei este setat.

Vezi descrierea din sectiunea 3.2 pentru o discutie asupra problemelor ridicate de subretele si alte rute de gazde. Rutele in subretele vor fi fara sens inafara retelei, si trebuie omise daca destinatia nu este in aceeasi subrete; ar trebui inlocuite cu o singura ruta spre retea din care fac parte si subretelele. Similar, rutele spre gazde trebuie eliminate daca sunt subsumate de o ruta din retea, asa cum am descris in sectiunea 3.2.

Daca ruta trece de aceste teste, atunci destinatia si metrica sunt puse in intrarea din datagrama de iesire. Rutele trebuie incluse in datagrama chiar si daca metricile lor sunt infinite. Daca poarta pentru ruta este in retea pentru care este pregatita datagrama, metrica din intrare este setata la 16, sau este omisa toata intrarea. Omiterea intrarii este

split horizon pur. Includerea unei intrari cu o metrica de 16 este split horizon with poisoned reverse . Vezi sectiunea 2.2 pentru o discutie mai amanuntita a acestor alternative.

3.6. Compatibilitate

Protocolul descris in acest document este gandit sa interopereze cu rutarile si alte implementari ale Rip-ului. Totusi, un punct de vedere diferit este adoptat cand trebuie incrementata metrica care a fost folosita in implementari anterioare. Folosind perspectiva anterioara, tabela interna de intrare are o metrica de 0 pentru toate retelele complet conectate. Costul (care este intotdeauna 1) este adaugat la metrica cand ii este trimisa rutei un mesaj de actualizare(refresh). In contrast, in acest document, retelele;direct conectate apar in tablea de intrare cu metrica egala cu costul; metricanu trebui sa fie in mod necesar 1. In acest document, costul este adaugat la metrica cand rutele primesc mesaje de actualizare. Metricile din tabela de rutare sunt trimise in mesajel de actualizare fara a fi schimbate (doar daca sunt modificate prin split horizon).

Cele doua puncte de vedere rezulta din cand sunt trimise mesajele identice de actualizare.etricile din tabela de rutare difera de o constanta din cele doua descrieri.

Astfel, de fapt nu este nici o diferenta. Schimbarea a fost facuta deoarece noua descriere face mai usoara tratarea situatiilor cand sunt folosite metricile diferite in retele direct-atasate.

Implementarile care suporta costuri de 1 ale retelei nu trebuie schimbata ca sa corespunda cu noul stil de prezentare. Totusi, trebuiesc urmareasca descrierea data in acest document in toate celelalte;te privinte.

4. Functii de control

Aceasta sectiune descrie contraolele administrative. Acestea nu fac parte din „prezentarea” protocolului. Totusi, experienta capatata cu retelele existente sugereaza ca sunt importante. Deoarece nu sunt o parte necesara din protcol, sunt considerate

optionale. Totusi, recomandam ca macar unele din ele sa fie incluse in fiecare implementare.

Aceste controale sunt gandite, in primul rand, sa permita RIP-ului sa fie conectat la retele a caror rutare poate fi instabila sau supusa la erori. Aici sunt cateva exemple:

Cateodata este de dorit sa fie limitate gazdele si gateway-urile de la care sunt acceptate informatii. IN unele imprejurari(circumstante), gazdele au fost configurate gresit in asa fel incat au inceput sa trimita informatie nepotrivita (gresita).

Anumite site-uri limiteaza numarul de retele pe care le admit in mesajele de actualizare. Organizatia A are o conexiune cu organizatia B, pe care o folosesc pentru comunicare directa. Din motive de securitate sau performanta, organizatia A poate sa nu ofere acces la conexiune altor organizatii. In astfel de cazuri, A nu ar trebui sa includa retelele organizatiei B, in actualizarile pe care A le trimite altor parteneri.

Aici sunt cateva controale tipice. De observat, totusi, ca protocolul RIP nu necesita astfel sau de controale sau alte tipuri.

- o retea vecina – administratorul de retea ar trebui sa poata defini o lista de vecini pentru fiecare gazda. O gazda ar trebui sa accepte mesaje de raspuns numai la gazde din lista sa de vecini.

- permiterea sau nepermiterea unor destinatii specifice - administratorul de retea ar trebui sa poata specifica o lista de adrese de destinatii permise sau nepermise. Aceasta lista ar fi asociata cu o interfata specifica pentru directia de intrare sau de iesire. Numai retelele permise vor fi mentionate in mesajele de raspuns care ies, sau procesate in mesajele de raspuns care intra. Daca o lista de adrese permiseeste specificata atunci toate celelalte adresa nu sunt permise. Daca este specificata o lista de adrese nepermise, atunci toate celelalte adrese sunt permise.

REFERINTE si BIBLIOGRAFIE

[1] Bellman, R. E., "Dynamic Programming", Princeton University Press, Princeton, N.J., 1957.

[2] Bertsekas, D. P., and Gallaher, R. G., "Data Networks",

Prentice-Hall, Englewood Cliffs, N.J., 1987.

- [3] Braden, R., and Postel, J., "Requirements for Internet Gateways",
USC/Information Sciences Institute, RFC-1009, June 1987.

- [4] Boggs, D. R., Shoch, J. F., Taft, E. A., and Metcalfe, R. M.,
"Pup: An Internetwork Architecture", IEEE Transactions on
Communications, April 1980.

- [5] Clark, D. D., "Fault Isolation and Recovery," MIT-LCS, RFC-816,
July 1982.

- [6] Ford, L. R. Jr., and Fulkerson, D. R., "Flows in Networks",
Princeton University Press, Princeton, N.J., 1962.

- [7] Xerox Corp., "Internet Transport Protocols", Xerox System
Integration Standard XSIS 028112, December 1981.