Retele de calculatoare - proiect

Aplicatie pentru descoperirea unei topologii de retea pe baza mecanismului de comunicație RIPv2

Nistor Paula-Alina

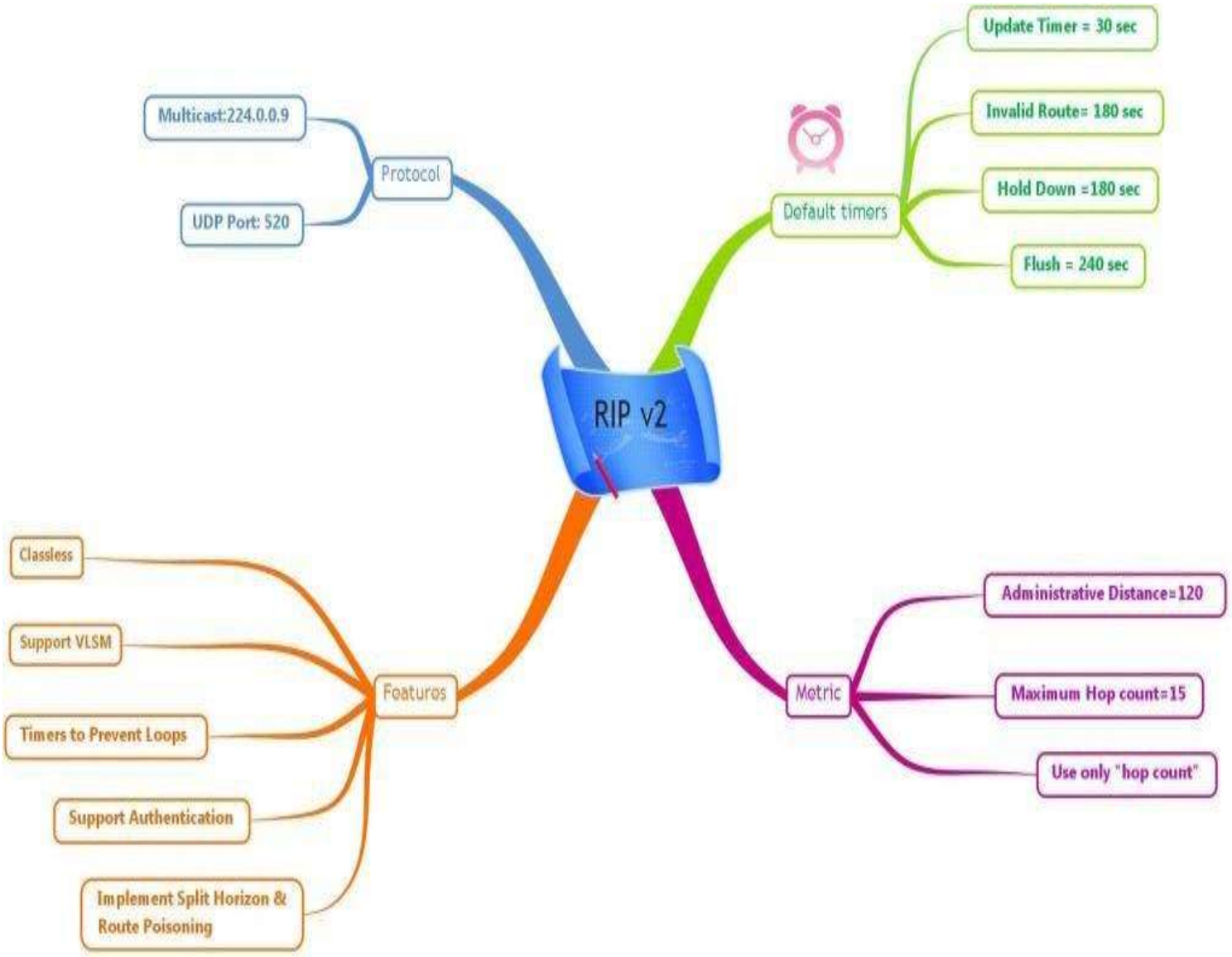
Sirghi Maria -

Simona

1. Notiuni teoretice

1.1. Introducere

Protocolul de Rutare a Informației (RIP, Routing Information Protocol) este un protocol de rutare bazat pe algoritmul Bellman-Ford (sau cu vectori de distanta) ce implică utilizarea ca metrică de rutare a numărului de pași de rutat.



1.2. Descrierea algoritmului – logica de comunicatie

Rutarea este incercarea de a gasi o cale de la expeditor la o destinatie. 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.

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 IPul, 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.

Formal, este posibil sa ajungem direct de la entitatea i la entitatea j (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. 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.

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 deasemenea.

- Se retine 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, se trimite o actualizare a informatiilor de rutare la fiecare vecin. Actualizarea este un set de mesaje care contine toate informatiile de la tabelul de rutare. Contine o intrare pentru fiecare destinatie, cu distanta aratata acelei destinatii.
- Cand o actualizare a informatilor de rutare ajunge la vecinul G’, se aduna costul asociat cu reteaua care este impartita cu G’. Fie distanta rezultata D’. Se compara distantele rezultate cu intrarile curente din retea. Daca noua distanta D’ pentru N este mai mica decat valoarea existenta D, se adopta noua ruta. Adica, se schimba intrarea din tabela pentru N pentru a rezulta metrica D’ si gateway-ul (poarta) G’. Daca G’ este gateway-ul (poarta) de unde a pornit ruta, G’=G, atunci se foloseste noua metrica chiar daca este mai mare decat cea initiala.

1.3. Specificatii de protocol

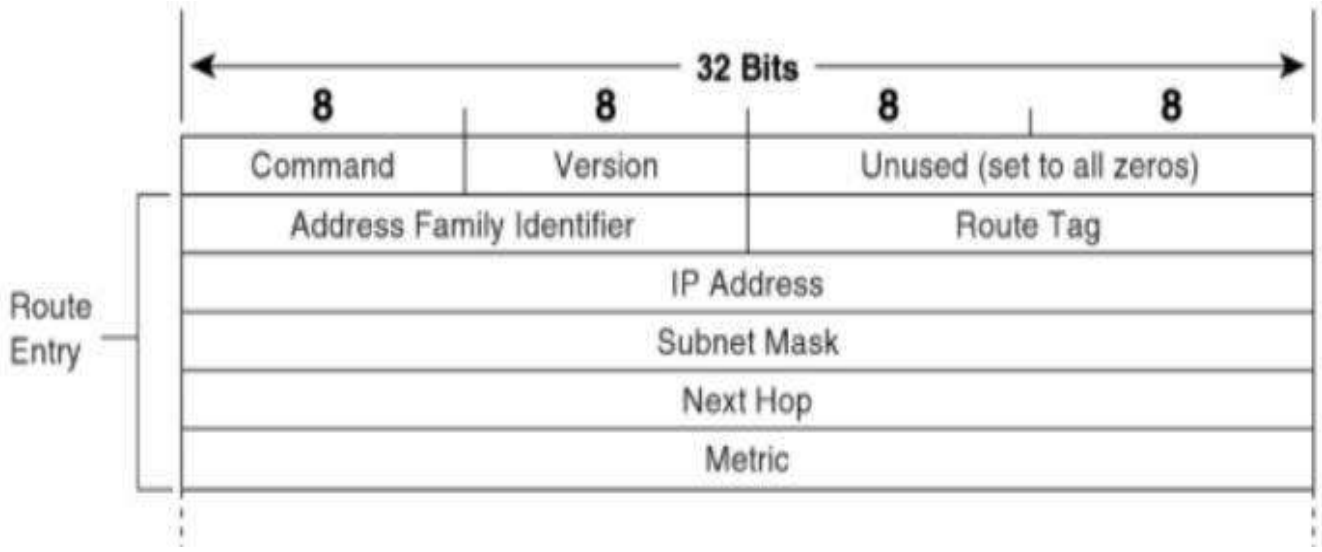
Fiecare gazda care implementeaza RIP isi atribuie dreptul de a avea o tabela de rutare. Aceasta tabela are o singura intrare pentru fiecare destinatie care este disponibila prin sistemul descris de RIP. Fiecare intrare contine cel putin informatiile urmatoare:

- Adresa IP a destinatiei.
- 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 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.

1.4. 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 mesajele de notificare ale routarii sunt trimise spre portul 520. Mesajele de notificare si routare 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.

Mesajul are urmatoarea structura:



- Command – este setat fie “1” pentru mesaje de cerere, fie “2” pentru mesajele de raspuns.
- Version – se va seta “2” pentru protocolul RIPv2. Daca este setat “0” sau “1” dar mesajul nu este un mesaj RIPv1 valid acesta va fi sters. Totusi RIPv2 va procesa mesajele RIPv1 valide.
- Address Family Identifier – este setat “2” pentru adresele IPv4. Singura exceptie este atunci cand se trimite un mesaj de cerere pentru tabela de routare a unui router (sau gazda), caz in care campul va fi setat pe “0”.
- Route Tag – ofera un camp pentru etichetarea routerelor externe sau a routerelor care au fost redistribuite in procesul RIPv2. Deși RIP in sine nu folosește acest camp, protocoalele externe de navigație conectate la un domeniu RIP în mai multe locatii pot utiliza campul route tag pentru a face schimb de informatii intre domeniul RIP.
- IP Address – este adresa IPv4 a destinației rutei. Poate fi o adresa de rețea majora, o subretea sau o ruta gazda.
- Subnet Mask – este o masca de 32 de biti care identifica portiunea de retea si subretea a adresei IPv4.
- Next Hop – identifica adresa urmatorului router spre destinatie.
- Metric – este un numar cuprins intre 1 si 16.

1.5. Detalii implementare pachete

Mesajele sunt reprezentate prin intermediul clasei RipPack a carei implementare este alcatuita din cele doua clase auxiliare Header si RouteEntry.

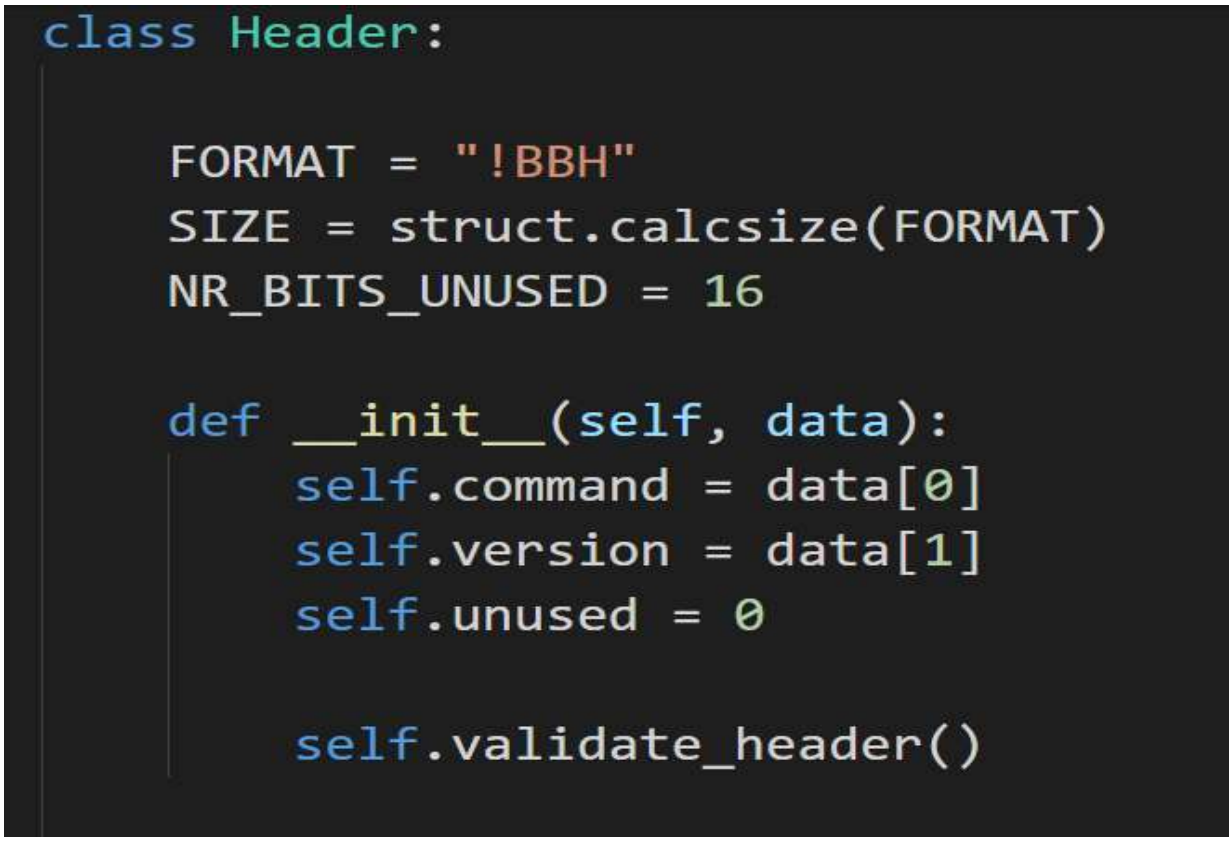
```
5  class RipPack:
6      def __init__(self):
7          self.header = None
8          self.route_entries = []
9
```

Header: Orice mesaj contine un header alcatuit dintr-o comanda si numarul versiunii. Comanda este utilizata in specificarea scopului mesajului. In acest sens s-au definit doua tipuri de comenzi:

“1” - request - o cerere emisa de catre sistemul care a receptionat un mesaj de a trimite partial sau complet routing table-ul

“2” - response - un mesaj compus tot sau anumite segmente din routing table. Acest mesaj poate fi trimis ca raspuns pentru o cerere sau poate fi o simpla actualizare nesolicitata a routing table-ului generata de cel care a emis mesajul

In implementare, cele doua campuri mentionate anterior sunt reprezentate de proprietatile command si version. In continuare este definita proprietatea unused oferind backwards compatibility cu versiunea precedenta.



```
class Header:

    FORMAT = "!BBH"
    SIZE = struct.calcsize(FORMAT)
    NR_BITS_UNUSED = 16

    def __init__(self, data):
        self.command = data[0]
        self.version = data[1]
        self.unused = 0

        self.validate_header()
```

RouteEntry: RIPv2 codifica adresa de next-hop in fiecare route entry. In implementarea acestuia sunt definite urmatoarele proprietati:

- address_family_identifier: contine tipul adresei
- route_tag: atribut asociat unei rute care trebuie sa fie pastrat si republicat
- ip_address: adresa IP asociata host-ului sau network-ului
- subnet_mask: ofera routerlui posibilitatea sa identifice diversele subretele care se afla intr-o singura retea sau subretelele aflate in retele la distanta
- next_hop: adresa IPv4 a urmatorului router din parcursul mesajului catre destinatie
- metric: o valoare numerica care indica distanta pana la destinatie

```
class RouteEntry:

    FORMAT = "!HHIIII"
    SIZE = struct.calcsize(FORMAT)

    def __init__(self, entry):
        self.address_family_identifier = entry[0]
        self.route_tag = entry[1]
        self.ip_address = entry[2]
        self.subnet_mask = entry[3]
        self.next_hop = entry[4]
        self.metric = entry[5]

        self.timeout = None
        self.garbage_time = None
        self.changed_flag = False
```

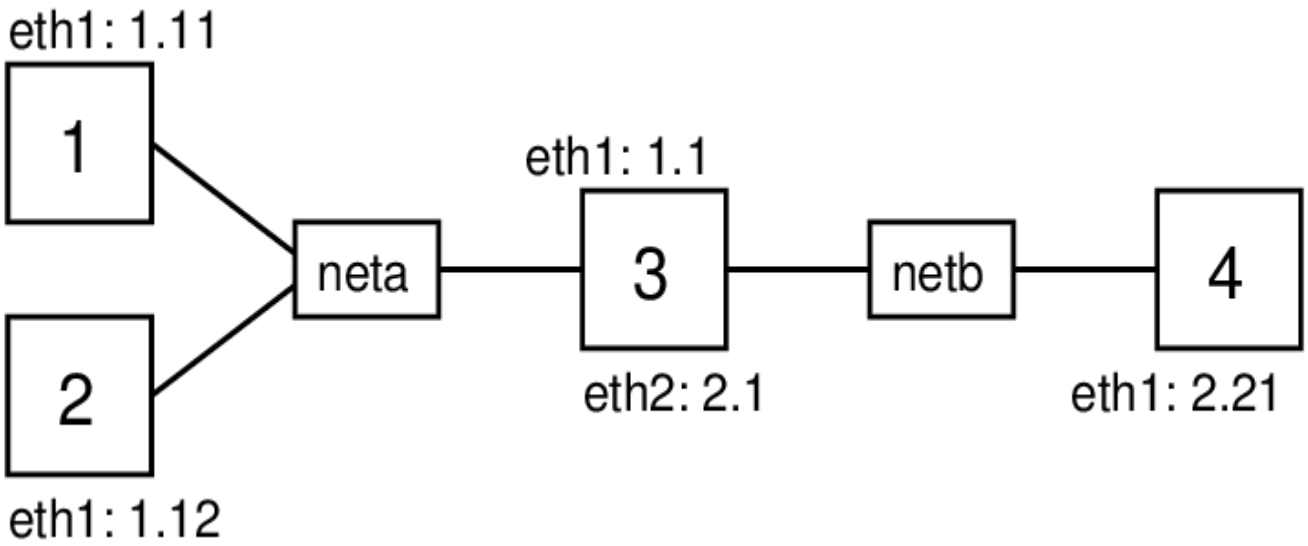
In aditia functionalitatilor deja enumerate, Header se ocupa si de tratarea eventualelor exceptii care pot aparea in realtie cu corectitudinea datelor ce compun pachetul.

```
def validate_header(self):
    if self.command not in [1, 2]:
        raise Exception("(RIP HEADER) Command must be 1 or 2")

    if self.unused != 0:
        raise Exception("(RIP HEADER) Unused bits are not set to zeros")

    if self.version not in [1, 2]:
        raise Exception("(RIP HEADER) Inavlid protocol version")
```

2. Implementare

2.1. Configurarea topologiei

Topologia de test a fost creata in VirtualBox, utilizand virtual machine-uri linux Debian 10 buster.



3. Librarii auxiliare

3.1. Comunicare in retea (socket)

Programarea prin socket este o modalitate de conectare a două noduri dintr-o rețea pentru a comunica între ele. Un socket (nod) ascultă pe un anumit port al unui IP, în timp ce alt socket ajunge la celălalt pentru a forma o conexiune.

3.2. Interpretare adrese IP (ipaddress)

Creaza o abstratie a adreselor IP sub forma de obiecte python pentru a le vizualiza si manipula.

3.3. Acces la interfete de retea din Python (netifaces)

Pentru a obține adresa (adresele) de rețea in mod direct a mașinii pe care rulează scripturile Python.

3.4. ipcalc

Permite calculul adresei IP a retelelor.

4. Interfata grafica

Interfata permite vizualizarea tuturor rutelor din tabela de routare pentru fiecare router in parte, putand fi ales timpul de actualizare a rutelor, de timeout – timpul dupa care o ruta devine invalida si flush time, cand o ruta invalida este stearsa din tabela de routare.

Pentru realizarea interfetei s-a utilizat libraria PyQt.