Laborator APC - 5

Rețele IP - Partea a II-a

Vom continua în această lucrare de laborator studiul rețelelor bazate pe protocolul IPv4, început în lucrarea precedentă.

Obiective

În această lucrare vom considera o rețea IP privată conectată (conceptual) la Internet. Vom studia, în principal, următoarele tehnici și protocoale:

- Construirea tabelelor de rutare și dirijarea pachetelor pentu o rețea privată conectată la Internet. Configurarea interfețelor cu adrese alocate static și configurarea rutelor statice în Linux/Unix folosind comanda `ip` (în locul comenzilor `ifconfig` și `route`). Examinarea stării interfețelor, a tabelului ARP cache și a tabelului de rutare folosind acestei comenzi.
- Utilizarea tehnicii NAPT (Network Address and Port Translation) pentru conectarea la Internet a unei rețele care folosește spațiu privat de adrese.
- Rutarea dinamică folosind protocolul RIP (Routing Information Protocol).

Precondiţii

Pentru a putea efectua experimentele și a interpreta rezultatele trebuie să studiaţi în prealabil capitolele din materialul de curs (și eventual bibliografia suplimentară) care prezintă noțiunile de bază privind: alocarea adreselor și dirijarea pachetelor în rețele IPv4, protocoalele ARP și ICMP, adrese IPv4 private și tehnicile NAT/NAPT.

Software şi echipamente

Vom folosi implementări ale protocoalelor și serviciilor disponibile în sistemul de operare Linux. În particular, vom folosi pachetul `iptables` pentru a implementa NAPT și implementarea RIP din suita Quagga pentru rutare dinamică. Veți captura și analiza comunicațiile dintre echipamente folosind analizoarele de protocoale `tcpdump` și `wireshark`.

Fiecare student (sau echipă de 2 studenți) va lucra pe un calculator cu sistemul de operare Linux. Sistemul studiat este emulat pe fiecare calculator folosind platforma de emulare `netkit`. Fiecare componentă a sistemului este implementată ca o mașină virtuală Linux și este accesibilă prin intermediul unui terminal (pentru configurare, examinarea stării, executarea unor utilitare, etc.).

A. Rutare statică

Studiu de caz

Figura 1 prezintă sistemul pe care îl vom folosi pentru experimente în prima parte a lucrării: o rețea privată conectată conceptual la Internet. Rețeaua privată este alcătuită din 5 subrețele, notate SN1, SN2, ..., SN5. Acestei rețele i s-a alocat blocul de adrese 10.1.2.0/23. Tabelul 1 descrie modul în care au fost alocate adrese din acest bloc celor 5 subrețele (a doua coloană din tabel specifică dimensiunea fiecărui bloc, determinată de numărul de adrese necesare subrețelei respective). Figura 1 indică blocul de adrese alocat fiecărei subrețele și adresa fiecărei interfețe.

Pentru a reduce numărul de mașini virtuale, conectarea la Internet este simplificată: sistemul nostru

conține doar o subrețea din Internet, cu prefixul 8.1.1.0/24, conectată direct la ruterul de frontieră al rețelei private, r4. Este suficient pentru experimentele pe care le vom efectua.

În fiecare subrețea, echipamentele comunică între ele folosind protocolul Ethernet ca și când ar fi conectate la un hub. Prin urmare, puteți captura întregul trafic dintr-o subrețea executând `tcpdump` pentru oricare dintre echipamentele conectate la subrețeaua respectivă.

Proiectul `netkit` cu care începeți lucrarea conține topologia din Figura 1, dar echipamentele nu sunt configurate complet.

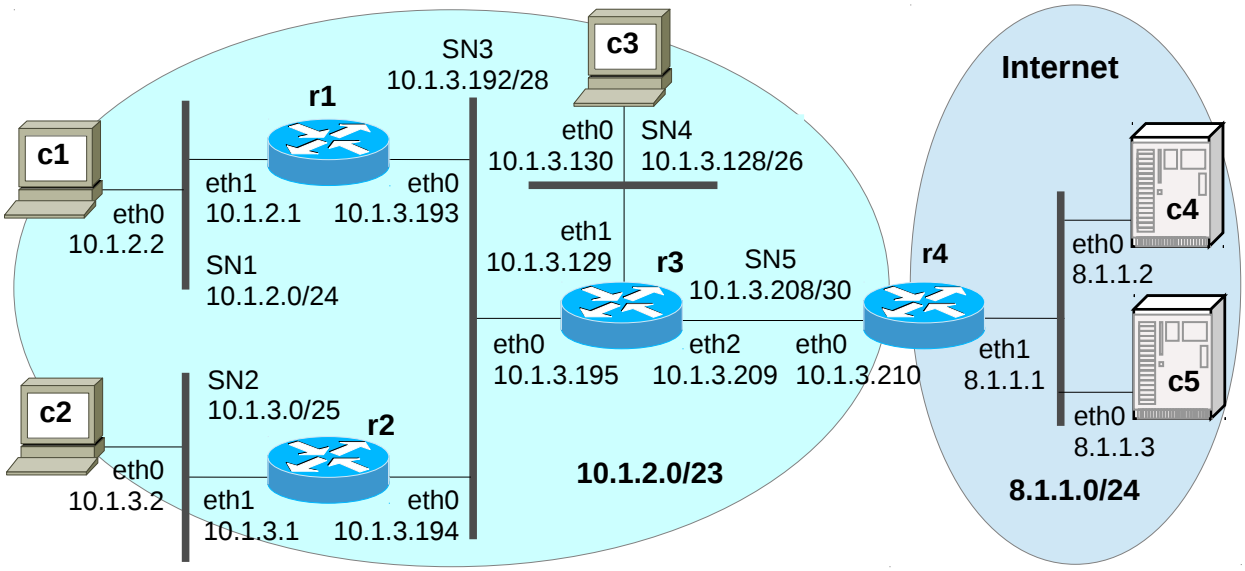

Figura 1: Sistemul folosit pentru experimente: rețea privată conectată la Internet.

Tabelul 1: Alocarea adreselor în rețeaua privată.

Nume subrețea	Număr adrese	Prefixul blocului	Intervalul de adrese	Masca de subrețea
SN1	256	10.1.2.0/24	10.1.2.0 - 10.1.2.255	255.255.255.0
SN2	128	10.1.3.0/25	10.1.3.0 - 10.1.3.127	255.255.255.128
SN3	16	10.1.3.192/28	10.1.3.192 - 10.1.3.208	255.255.255.240
SN4	64	10.1.3.128/26	10.1.3.128 - 10.1.3.191	255.255.255.192
SN5	4	10.1.3.208/30	10.1.3.208 - 10.1.3.211	255.255.255.252

A.1. Inițializarea sistemului și configurarea interfețelor

A.1.1. Porniți emularea rețelei din Figura 1, executând comanda `lstart` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei lucrări.

A.1.2. Configurați interfețele calculatoarelor și ruterelor, folosind comanzile `ip link` și `ip address` (în loc de de `ipconfig`) . De exemplu, pentru c1 folosiți comenzile următoare:

`c1:~# ip link set dev eth0 up`	activează interfața `eth0`
`c1:~# ip addr add 10.1.2.2/24 dev eth0`	atribuie adresa IP, specifică lungimea prefixului

Interfețele celorlalte echipamente sunt configurate similar.

De ce este necesară specificarea lungimii prefixului pentru fiecare interfață?

A.1.3. După ce ați teminat configurarea interfețelor, examinați starea calculatoarelor și ruterelor folosind comenzile următoare:

```
ip link show
ip addr show
ip route show
```

Sunt activate și configurate corect interfețele, conform Figurii 1? Există în tabelul de rutare al fiecărui echipament rute pentru toate subrețelele conectate direct? Cum au fost obținute aceste rute?

Comanda `ip` acceptă diverse prescurtări, care sunt foarte utile atunci când o folosiți în mod interactiv. Încercați, de exemplu, următoarele comenzi: `ip l`, `ip a`, `ip r`.

Comanda `ip address` permite și ștergerea adresei IP (`ip addr del` sau `ip addr flush`).

A.1.4. Verificați conectivitatea între echipamente folosind comanda `ping`. Este posibilă comunicația între c1 și r1? Dar între c1 și c2? De ce?

A.2. Configurarea rutelor statice

Pentru început, veți construi tabelele de rutare folosind rute statice, pe care le veți identifica pe baza topologiei rețelei și le veți configura manual. Spre deosebire de lucrarea precedentă, veți folosi în acest scop comanda `ip route`.

A.2.1. Configurați mai întâi ruterele. Un ruter trebuie să poată dirija pachetele IP către orice destinație din rețeaua privată. În plus, pentru dirijarea pachetelor către destinații aflate în afara rețelei private, adăugați pe r1, r2 și r3 o rută implicită către r4.

Pentru a putea examina și modifica ulterior comenzile, deschideți un editor de text pe calculatorul gazdă, editați comenzile pentru toate ruterele conform exemplului de mai jos și salvați-le într-un fișier. Configurați apoi fiecare ruter, copiind comenzile din editor în terminalul ruterului.

De exemplu, pentru ruterul r1, folosiți comenzile următoare:

```
r1:~# ip route add 10.1.3.0/25 via 10.1.3.194 dev eth0
r1:~# ip route add 10.1.3.128/26 via 10.1.3.195 dev eth0
r1:~# ip route add 10.1.3.208/30 via 10.1.3.195 dev eth0
r1:~# ip route add default via 10.1.3.195 dev eth0
```

Celelalte rutere sunt configurate similar.

Explicați cum este folosită ruta implicită (`default`) în acest caz.

A.2.2. După ce ați terminat configurarea ruterelor, examinați și verificați conținutul tabelelor de rutare folosind comanda `ip route show`. Copiați informația afișată din terminalul fiecărui ruter în fișierul care conține comenzile de configurare.

Există rute corecte către toate subrețelele? Cum ați putea reduce numărul de rute prin agregare?

A.2.3. Configuraţi acum calculatoarele. Este suficient să adăugăm pe fiecare calculator o rută implicită către ruterul care îl conectează la restul reţelei. De exemplu, pentru c1, această rută se poate configura cu comanda următoare:

```
c1:~# ip route add default via 10.1.2.1 dev eth0
```

Celelalte calculatoare sunt configurate similar.

A.2.4. După ce aţi terminat configurarea calculatoarelor, examinaţi şi verificaţi conţinutul tabelelor lor de rutare folosind comanda `ip route show`.

A.2.5. Verificaţi conectivitatea între echipamente folosind comanda `ping`.

Este posibilă acum comunicaţia între oricare dintre echipamentele din Figura 1, inclusiv c4 şi c5?

A.2.6. Puteţi examina şi modifica înregistrările din ARP cache folosind comanda `ip neighbor show` (abreviată `ip n`), în locul comenzii `arp`. Generaţi trafic între c1 şi c2 folosind `ping` şi apoi examinaţi conţinutul ARP cache din c1, r1 şi c2 folosind această comandă.

Puteţi folosi comanda `ip neigh` pentru a efectua modificări în tabelul ARP: se pot adăuga (`ip neigh add`) sau şterge înregistrări (`ip neigh del` sau `ip neigh flush`).

B. Interconectarea reţelelor folosind NAT / NAPT

Noţiunea de adresă IP privată a fost introdusă în a doua jumătate a anilor '90, în condiţiile creşterii rapide a Internetului şi a riscului ca spaţiul de adrese oferit de IPv4 să fie epuizat înainte de finalizarea tranziţiei de la IPv4 la IPv6.

Soluţia propusă prevedea rezervarea unor bocuri de adrese IPv4, numite adrese private, care să poată fi folosite de orice reţea pentru comunicaţii interne (private). Reutilizând aceste blocuri de adrese într-un mare număr de reţele s-a putut realiza o economie masivă de adrese IPv4. Acest lucru a permis creşterea rapidă a Internetului bazat pe IPv4 timp de încă două decenii.

Adresele private nu sunt rutabile global, pentru că sunt utilizate în mai multe reţele, în acelaşi timp. Ca să comunice cu restul Internetului, o reţea care foloseşte intern adrese private are nevoie şi de una sau mai multe adrese care să fie unice la nivel global. Conectarea reţelei la Internet se face printr-un ruter care înlocuieşte adresele private din antetul fiecărui pachet cu una dintre adresele globale alocate reţelei. Această tehnică se numeşte NAT (Network Address Translation).

Numărul de adrese globale alocate reţelei este mult mai mic decât numărul de adrese private utilizate intern. În felul acesta se pot economisi adrese. Pe de altă parte, dispozitivul NAT trebuie să permită realizarea tuturor comunicaţiilor externe solicitate de echipamentele din reţeaua privată, folosind doar adresele globale alocate reţelei, de o manieră (pe cât posibil) transparentă pentru aplicaţiile care comunică.

Una dintre metodele folosit în acest scop este NAPT (Network Address and Port Translation), care înlocuieşte atât adresa IP privată, cât şi numărul de port asociat. De exemplu, există 2^{16} porturi TCP, deci pentru fiecare adresă globală care este alocată reţelei, un dispozitiv NAPT ar putea distinge 2^{16} conexiuni TCP (similar pentru UDP şi alte protocoale). Astfel, un număr mare de echipamente din reţeaua privată pot comunica în acelaşi timp cu echipamente externe.

Reţeaua privată din exemplul nostru foloseşte o parte din blocul de adrese private 10.0.0.0/8. Prin urmare, pentru a o putea conecta la Internet, ruterul r4 trebuie să asigure şi funcţia NAT. Mai mult decât atât, reţeaua noastră privată dispune de o singură adresă globală, 8.1.1.1 (Figura 2). Cu toate acestea, folosind NAPT, ruterul r4 va putea asigura mai multe comunicaţii externe în acelaşi timp,

între oricare dintre echipamentele din rețeaua privată și echipamente conectate la Internet. Pentru activarea acestei funcții, este suficient să executați un script care a fost deja pregătit în acest scop.

B.1. Activarea funcției NAPT

B.1.1. Activați NAPT pe ruterul r4 executând comanda următoare:

```
r4:~# /root/setnat.sh
```

În urma executării acestui script, ruterul va fi traversat doar de pachete aparținând unor comunicații inițiate din interiorul rețelei private.

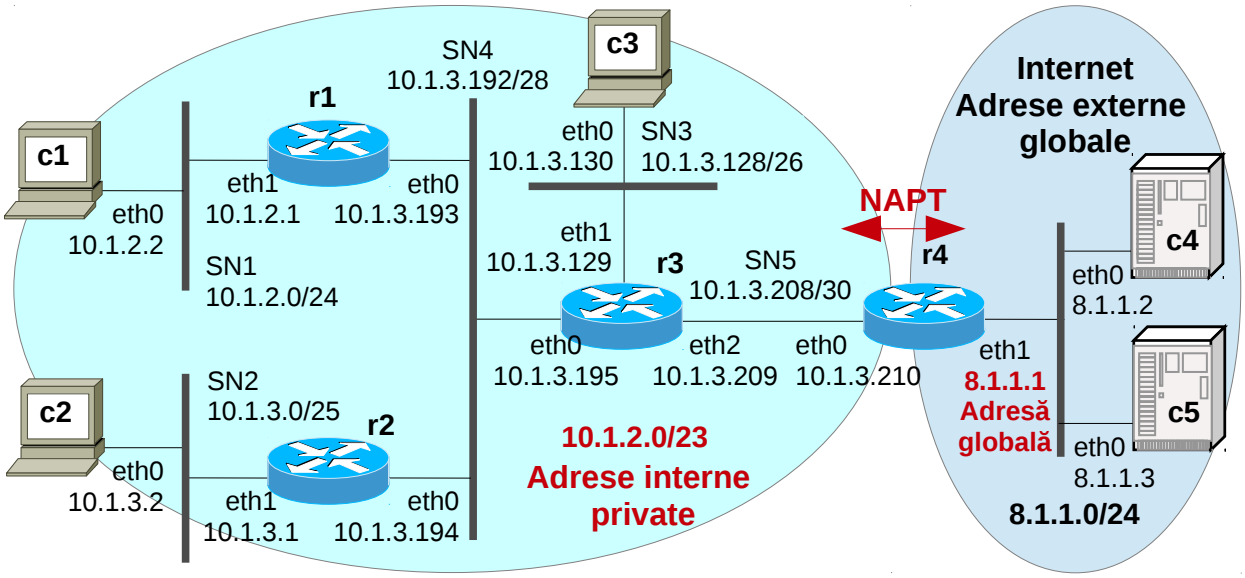

Figura 2: Conectarea la Internet folosind NAPT.

B.2. Trafic ICMP Echo Request/Reply

B.2.1. Porniți captura traficului pe interfața `eth2` a ruterului r3 și interfața `eth1` a ruterului r4:

```
r3:~# tcpdump -s0 -tnv -i eth2 icmp
```

```
r4:~# tcpdump -s0 -tnv -i eth1 icmp
```

B.2.2. Executați comanda `ping` pe echipamente din rețeaua privată pentru a testa dacă acestea pot să comunice cu echipamentele din rețeaua externă (c4 și c5).

Care este rezultatul afișat de `ping`? Comparați traficul capturat de `tcpdump` pe cele 2 interfețe ale ruterului r4 (în special adresele IP și conținutul mesajelor ICMP). Ce diferențe observați?

B.2.3. Executați simultan testele următoare:

```
c1:~# ping 8.1.1.2
```

```
c2:~# ping 8.1.1.2
```

Care este rezultatul afișat de `ping`? Cum puteți distinge cărui test îi aparțin pachetele capturate pe interfața `eth1` a ruterului r4? Cum credeți că distinge r4 pachetele conținând ICMP Echo Reply pe care le primește din rețeaua externă pentru a substitui corect adresele IP?

B.2.4. Efectuați testul invers, executând comanda `ping` pe c4:

```
c4:~# ping 10.1.3.2
```

Care este rezultatul afișat de `ping`? Analizați traficul capturat de `tcpdump` în timpul acestui test și explicați ce s-a întâmplat.

B.3. Trafic TCP

B.3.1. Vom testa mai întâi o conexiune TCP între c1 și c4 folosind programul `nc` (`netcat`).

Porniți captura traficului pe interfața `eth2` a ruterului r3 și interfața `eth1` a ruterului r4:

```
r3:~# tcpdump -s0 -i eth2 -w /hostlab/natr3e2web1.cap
```

```
r4:~# tcpdump -s0 -i eth1 -w /hostlab/natr4e1web1.cap
```

Porniți programul `nc` pe c4, cerându-i să aștepte cereri de conexiune TCP pe portul 100:

```
c4:~# nc -nv -l -p 100
listening on [any] 100 ...
```

Porniți apoi `nc` pe calculatorul c1, cerându-i să deschidă o conexiuneTCP cu c4, pe portul 100:

```
c1:~# nc -nv 8.1.1.2 100
(UNKNOWN) [8.1.1.2] 100 (?) open
```

Pentru a vă convinge că aveți o conexiune TCP funcțională, editați pe c1 un scurt mesaj și apăsați Enter pentru a fi transmis către c4. Apoi închideți conexiunea cu Ctrl-C.

Terminați captura și vizualizați pachetele cu Wireshark. Comparați traficul capturat pe cele 2 interfețe ale ruterului r4 (în special adresele IP și antetul segmentelor TCP). Ce diferențe observați?

B.3.2. Vom testa acum mai multe comunicații TCP care au loc simultan, stabilite de c1 și c2 cu un server HTTP care rulează pe c4.

Porniți din nou captura traficului pe interfața `eth2` a ruterului r3 și interfața `eth1` a ruterului r4.
Porniți serverul HTTP pe calculatorul c4 (serverul așteaptă conexiuni pe portul 80):

```
c4:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Porniți programul `nc` pe c1 și pe c2, solicitând deschiderea unei conexiuni TCP cu serverul HTTP de pe c4 (portul destinație 80) și portul sursă 12345:

```
c1:~# nc -nv 8.1.1.2 80 -p 12345
(UNKNOWN) [8.1.1.2] 80 (www) open
```

```
c2:~# nc -nv 8.1.1.2 80 -p 12345
(UNKNOWN) [8.1.1.2] 80 (www) open
```

Pe conexiunile TCP pe care le-ați deschis, transmiteți serverului HTTP comanda "`GET /`" (apăsați de două ori Enter, pentru a termina mesajul HTTP). Exemplu:

```
c1:~# nc -nv 8.1.1.2 80 -p 12345
(UNKNOWN) [8.1.1.2] 80 (www) open
GET /

```

```
<title>Directory listing for /</title>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href=".bashrc">.bashrc</a>
<li><a href=".profile">.profile</a>
</ul>
<hr>
```

Terminați captura și vizualizați pachetele cu Wireshark. Comparați traficul capturat pe cele 2 interfețe ale ruterului r4 (adresele IP și antetul segmentelor TCP). Ce diferențe observați?

În acest test, am forțat clienții HTTP să folosească același port sursă. Cum reușește r4 să distingă pachetele aparținând celor două conexiuni? (Indicație: Examinați porturile TCP pentru conexiunea stabilită de c2, înainte și după r4.)

C. Rutare dinamică

În ultima parte a lucrării, vom reconfigura ruterele pentru a înlocui rutarea bazată pe rute statice din rețeaua privată cu rutare dinamică, folosind protocolul RIP (Routing Information Protocol).

RIP construiește tabelele de rutare folosind un algoritm distribuit care permite fiecărui ruter să descopere rutele cele mai scurte către toate destinațiile din rețea folosind vectori de distanțe (distance vector routing). Fiecare ruter anunță vecinilor (rutere conectate direct) lista destinațiilor către care poate livra pachetele, specificând pentru fiecare destinație prefixul și lungimea celei mai scurte rute pe care o cunoaște de la el însuși până la acea destinație (distance vector).

Metrica folosită de RIP pentru a evalua lungimea unei rute este numărul de rutere intermediare ("hop count"). Fiecare ruter cunoaște inițial doar rute către subrețelele conectate direct (în urma configurării interfețelor). Celelalte destinații și rutele cele mai scurte către ele sunt descoperite pe baza mesajelor primite de la ruterele vecine. Pentru fiecare destinație care nu este conectată direct, un ruter alege ruta cea mai scurtă dintre cele anunțate de vecinii săi.

Ruterele nu cunosc topologia întregii rețele și se bazează pe anunțurile primit de la vecini pentru a descoperi rute către destinațiilw care nu sunt conectate direct. În aceste condiții, pentru a evita apariția unor bucle de rutare, vectorul de distanțe transmis pe o interfață nu conține destinațiile către care pachetele sunt rutate prin acea interfață (regula '"split-horizon"). În plus, pentru a asigura convergența algoritmului atunci când o destinație nu mai este accesibilă (de pildă, în urma unei defecțiuni), lungimea maximă a unei rute este 15. Distanța 16 este folosită pentru a anunța (un timp limitat) că ruterul nu mai dispune de o rută funcțională către destinația respectivă.

C.1. Eliminarea rutelor statice

C.1.1. Prima fază a reconfigurării ruterelor constă în ștergerea rutelor statice configurate anterior. Pentru a simplifica această fază, deschideți fișierul în care ați salvat comenzile de configurare a a acestor rute, modificați comenzle ca în exemplul de mai jos, și copiați noile comenzi în terminalul fiecărui ruter. De exemplu, pentru ruterul r1, folosiți comenzile următoare:

```
r1:~# ip route del 10.1.3.0/25 via 10.1.3.194 dev eth0
```

```
r1:~# ip route del 10.1.3.128/26 via 10.1.3.195 dev eth0
r1:~# ip route del 10.1.3.208/30 via 10.1.3.195 dev eth0
r1:~# ip route del default via 10.1.3.195 dev eth0
```

Celelalte rutere sunt configurate similar.

C.1.2. Verificaţi rezultatul folosind comanda `ip route show`. De exemplu:

```
r1:~# ip r
10.1.3.192/28 dev eth0  proto kernel  scope link  src 10.1.3.193
10.1.2.0/24 dev eth1  proto kernel  scope link  src 10.1.2.1
```

C.1. Activarea şi configurarea protocolului RIP

Vom folosi o implementare a protocolului RIP pentru Linux/Unix, disponibilă în suita Quagga [4]. Quagga include implementări ale protocoalelor de rutare RIP, OSPF, IS-IS și BGP, precum și un program manager de rutare, numit Zebra.

În arhitectura Quagga, managerul de rutare Zebra și fiecare protocol de rutare activat sunt executate ca procese separate. Se poate interacţiona cu fiecare dintre aceste procese, pentru configurare și pentru examinarea stării, prin intermediul protocolului `telnet`. Quagga oferă și o interfață unificată, prin programul `vtysh`, care este mult mai convenabilă. Comenzile folosite pentru a interacţiona cu un ruter Quagga sunt similare celor disponbile în Cisco IOS. În cele ce urmează vom folosi doar `vtysh`.

C.1.1. Ruterele din rețeaua noastră au fost configurate parțial pentru a rula Zebra și RIP. Examinați fișierele de configurare inițiale ale protocoalelor de rutare. De exemplu:

```
r1:~# cat /etc/zebra/daemons
```

```
r1:~# cat /etc/zebra/zebra.conf
```

```
r1:~# cat /etc/zebra/ripd.conf
```

```
r1:~# cat /etc/zebra/vtysh.conf
```

C.1.2. Porniți procesul Zebra pe toate cele patru rutere. De exemplu:

```
r1:~# /etc/init.d/zebra start
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra ripd.
```

În urma executării acestei comenzi ar trebui să pornească două procese, Zebra și RIP (`ripd`). Verificaţi acest lucru cu comanda `ps ax`.

C.1.3. După activarea proceselor de rutare, porniți programul `vtysh` pe fiecare ruter și examinați starea inițială a ruterului. De exemplu:

```
r1:~# vtysh
Hello, this is Quagga (version 0.99.10).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
r1-quagga# show interface
...
r1-quagga# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
```

```
       I - ISIS, B - BGP, > - selected route, * - FIB route
C>* 10.1.2.0/24 is directly connected, eth1
C>* 10.1.3.192/28 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
```

Observați că, deși ați pornit procesul RIP pe fiecare ruter, tabelele de rutare conțin în continuare doar rute către subrețelele conectate direct. De fapt, protocolul RIP nu funcționează încă, pentru că mai are nevoie de câteva comenzi de configurare.

Pentru început, vom activa RIP pe r1 și r2 și vom observa mesajele RIP transmise de aceste rutere capturând traficul pe interfața `eth0` a ruterului r3.

C.1.4. Terminați pentru moment `vtysh` pe ruterul r3 și porniți captura de pachete pe interfața `eth0`:

```
r3-quagga# exit
r3:~# tcpdump -i eth0 -s0 -vn udp port 520
```

Mesajele RIP sunt transmise folosind UDP, cu portul sursă și destinație 520, care este rezervat pentru RIP. Prin urmare, comanda de mai sus va captura mesajele RIP transmise în subrețeaua SN4.

C.1.5. Configurați inițial protocolul RIP versiunea 2 (RIPv2) pe ruterele r1 și r2. De exemplu:

```
r1-quagga# configure terminal
r1-quagga(config)# router rip
r1-quagga(config-router)# version 2
r1-quagga(config-router)# network 10.1.0.0/16
r1-quagga(config-router)# exit
r1-quagga(config)# exit
r1-quagga#
```

Comanda `network 10.1.0.0/16` activează RIP pe toate interfețele ale căror adrese au prefixul 10.1.0.0/16. Ruterul r2 este configurat similar.

C.1.6. Examinați tabelele de rutare ale ruterelor r1 și r2. Exemplu:

```
r1-quagga# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

C>* 10.1.2.0/24 is directly connected, eth1
R>* 10.1.3.0/25 [120/2] via 10.1.3.194, eth0, 01:09:34
C>* 10.1.3.192/28 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
```

Ce modificări apar în tabelele de rutare?

Examinați pachetele capturate de `tcpdump`, identificați vectorii de distanțe din mesajele transmise de RIP și legătura dintre tabelele de rutare și acești vectori. Adresa IP destinație a mesajelor este adresa multicast rezervată pentru protocolul RIPv2, 224.0.0.9 ("all RIP2-aware routers").

Opriți captura pe r3 și porniți din nou `vtysh`.

C.1.7. Configurați RIP și pe ruterele r3 și r4, ca în exemplul de la punctul C.1.5.

Examinați tabelele de rutare ale celor 4 rutere și explicați modificările. Există rute către toate cele 5 subrețele?

Observație: În final, după executarea comenzile listate mai sus, RIP este activat pe toate interfețele celor 4 rutere, mai puțin interfața `eth1` a ruterului r4.

C.1.8. Porniți o captură pe interfața `eth0` a calculatorului c1.

Examinați și explicați traficul capturat. Examinați și tabelele de rutare ale calculatoarelor. Există vreo modificare față starea inițială? Explicați.

Observație: Pentru a minimiza consumul de resurse, RIP ar trebui activat doar pe interfețele care sunt conectate la subrețele unde mai există și alte rutere RIP (de exemplu, pentru r1, r2 și r3 nu ar trebui activat pe interfața `eth1`). Pentru comenzile de configurare din exemplul de mai sus, este suficient să modificați prefixul comenzii `network` (pentru r3 sunt necesare 2 comenzi). O variantă mai convenabilă este să folosiți comanda `passive interface`.

C.1.8. Verificați funcționarea corectă a rețelei folosind `ping` și `traceroute`.

În particular, sunt posibile comunicații între c1 și c4? De ce?

C.1.9. Modificați configurarea ruterului r4 astfel încât RIP să anunțe și o rută implicită:

```
r4-quagga# configure terminal
r4-quagga(config)# router rip
r4-quagga(config-router)# default-information originate
r4-quagga(config-router)# exit
r4-quagga(config)# exit
r4-quagga#
```

Examinați tabelele de rutare ale celor 4 rutere. Identificați și explicați modificările.

Verificați folosind `ping` dacă sunt posibile acum și comunicații cu calculatoarele externe c4 și c5.

Terminați `vtysh` cu comanda `exit`.

Comparați tabelele de rutare construite de RIP cu tabelele de rutare construite inițial cu rute statice.

Terminarea lucrării

Executați comanda `lcrash` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei lucrări.

Bibliografie

1. O. Catrina. APC - Note de curs.
2. The Linux Foundation. iproute2. https://wiki.linuxfoundation.org/networking/iproute2
3. Linux and Unix ip command. http://www.computerhope.com/unix/ip.htm
4. Quagga Routing Suite. http://www.nongnu.org/quagga/