Laborator 2
Rutarea statică; rutarea dinamică folosind RIP; introducere în ACL

Partea 1. Breviar teoretic

1. Rutare statică

Un ruter deţine implicit informaţie despre reţelele direct conectate. Informaţiile despre reţelele care nu sînt direct conectate la un ruter sînt obţinute fie prin specificarea lor direct de către administrator (rutare statică), fie prin intermediul unui protocol de rutare (rutare dinamică).

A) Configurarea unei rute statice se face folosind una din următoarele variante:

```
Router(config)#ip route reţea_destinaţie masca_reţea ip_next_hop
Router(config)#ip route reţea_destinaţie masca_reţea interf_ieşire
```

Evident, *ip_next_hop* este o destinaţie cunoscută de ruter (aflată într-o reţea direct conectată)! A doua variantă este doar pentru cazul reţelelor punct-la-punct, cum sînt cele corespunzînd interfeţelor seriale.

De exemplu, pentru a configura o rută statică pe `Router0` pentru a ajunge la reţeaua 192.168.1.0/24:

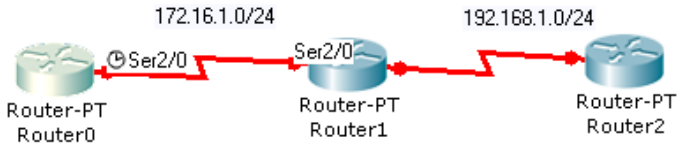

Figura 1

se va folosi una din comenzile:

```
Router0(config)#ip route 192.168.1.0 255.255.255.0 172.16.1.2
```

unde 172.16.1.2 este adresa IP a interfeţei s2/0 de pe Router1, sau:

```
Router0(config)#ip route 192.168.1.0 255.255.255.0 s2/0
```

unde s2/0 este interfaţa ruterului Router0.

B) Configurarea unei rute *default* (implicite)

Pentru toate destinaţiile care nu apar în tabela de rutare se poate defini o rută implicită:

```
Router2(config)# ip route 0.0.0.0 0.0.0.0 ip_next_hop|interf_out
```

Pentru *default route* se foloseşte notaţia `0.0.0.0 0.0.0.0` întrucît procesul de rutare funcţionează astfel: se compară adresa destinaţie cu fiecare intrare din tabela de rutare, folosind functia ŞI (AND). Comparaţia la care se potrivesc cei mai multi biţi va da rezultatul de valoarea cea mai mare şi aceasta va fi calea aleasa. Rezultatul unui AND între o adresă oarecare şi orice altă adresă este, în cazul cel mai defavorabil, 0.0.0.0 (atunci cînd nici un bit nu se potriveşte). Aşadar 0.0.0.0 semnifică “orice altceva”.

2. Rutarea dinamică folosind RIP

2.1 Generalităţi

Rutarea statică are avantajul că este simplă, nu necesită putere de calcul în ruter pentru determinarea rutelor (această muncă e făcută de administrator) şi nu generează trafic în reţea (ruterele nu schimbă între ele informaţii de rutare), dar prezintă următoarele dezavantaje:

- în cazul unor schimbări, administrarea este dificilă
- nu se adaptează automat în cazul apariţiei unor defecţiuni; o rută devenită inaccesibilă rămîne inaccesibilă, pînă la intervenţia administratorului.

Protocoalele de rutare dinamică – pentru care în continuare se va folosi termenul de „protocoale de rutare" – se clasifică, în functie de algoritmul de stabilire a informatiei de rutare, în: protocoale de tip „*link state*" şi protocoale de tip „*distance vector*"

O clasificare suplimentară, care ţine cont de modul de definire a ariei de acţiune a protocoalelor de rutare, constă în *protocoale de rutare interne* (IGP - „Interior Gateway Protocols"), acestea actionînd în interiorul unui domeniu de rutare compus dintr-un număr de rutere aflate sub administraţie unică şi care definesc un asa-numit sistem autonom (AS - „Autonomous System"), şi *protocoale de rutare externe* (EGP - „Exterior Gateway Protocol"), utilizate pentru interconectarea sistemelor autonome.

Indiferent de procedeul de rutare ales, rutarea IP este de tipul "next hop", adică fiecare nod specifică *numai* nodul (hopul) următor, spre deosebire de "rutarea sursă" în care se specifică întreaga cale de urmat între sursă şi destinatie. Rutarea sursă nu se foloseste în Internet întrucît ar presupune o cunoaştere centralizată a întregii reţele şi nu s-ar putea adapta usor schimbărilor neprevăzute.

2.2 Protocoale de rutare de tip „distance vector"

Prin intermediul protocoalelor de rutare „distance vector", informaţia de rutare este:

(1) determinată pe baza unui algoritm de tip „shortest path", semnificativ fiind algoritmul Bellman-Ford – determinarea căii optime de ajungere la un nod distant ţinînd cont de cîteva criterii de cost bine definite (număr de hop-uri (noduri intermediare), lărgime de bandă a conexiunii intre doua rutere, etc.);

(2) distribuită către celelalte noduri ale reţelei în vederea actualizării tabelelor de rutare ale acestora.

Exemple: RIP, RIP-v2, IGRP, EIGRP, etc (ultimele 2 sînt proprietare Cisco).

2.3 Protocoale de rutare de tip „link-state"

Prin intermediul protocoalelor de rutare „link-state" informaţia de rutare este determinată:

(1) prin intermediul unui procedeu de replicare în reţea (replicare realizată de către fiecare nod component al reţelei) a unei baze de date conţinînd informaţii despre starea nodului respectiv şi a conexiunilor adiacente – în acest mod fiecare nod al reţelei ajungînd sa cunoască datele referitoare la poziţionarea fiecărui alt nod component al reţelei;

(2) prin calculul – efectuat prin aplicarea unui algoritm SPF („shortest path first", exemplu fiind algoritmul Dijkstra) – unui arbore de căi minime către fiecare destinaţie.

Exemplu: OSPF – va fi studiat în detaliu într-o lucrare ulterioară

2.4 Caracteristici comparative ale protocoalelor de rutare RIP şi OSPF

RIP (RFC 1058, 2453, etc.)

- metrica utilizată în selectarea (stabilirea) căii („path selection") este numărul de hopuri;
- max. 15 hop-uri
- update-uri transmise, în mod predefinit („by default"), la fiecare 30 de secunde.
- update-urile sînt trimise către adresa de broadcast 255.255.255.255

RIP v2

- transmite în pachetele de update către vecini şi masca de reţea, pentru a suporta reţele classless şi VLSM
- suportă autentificare
- update-urile sînt trimise către adresa multicast 224.0.0.9 (adresele cu primul octet mai mare decit 223 nu sînt în clasele A,B,C ci au destinaţii speciale; clasa multicast se numeste clasa D).

OSPF

- lărgimea de bandă („bandwidth") şi întîrzierea („delay") – avute în vedere în calculul caii optime;
- convergenţa rapidă.

2.5 Detalii privind RIP

Cîteva comenzi care permit studiul tabelelor de rutare şi a altor aspecte asociate rutarii:

comanda	*descriere*
`show ip route`	arată tabela de rutare; rutele sînt clasificate dupa tip: cele direct conectate, cele introduse manual (static) şi cele învăţate prin intermediul protocoalelor de rutare
`show ip route` *`destination_network`*	arată informaţii despre ruta către un net
`show ip protocols`	oferă informaţii despre protocoalele de rutare care rulează
`show ip rip database`	informaţii despre tabela RIP
`debug ip rip` eliminat cu `undebug all`	activează tipărirea mesajelor de debug la consolă: de fiecare data cînd se produce un eveniment asociat RIP (de exemplu este trimis sau receptionat un update), sînt afisate informatii despre acesta. Atentie! aceste mesaje sînt tiparite doar la consola, nu şi prin telnet !

Rutele învăţate de un ruter se pot şterge cu comanda:

```
Router# clear ip route *
```

unde * este un "wildcard" şi specifică "toate rutele". Alternativ, se poate specifica o singură rută în loc de *.

Fiecare rută are asociată o *distanţă administrativă;* dacă există mai multe rute disponibile, ruterul va alege ruta cu distanţa administrativă *cea mai mică*. Distanţele administrative se văd în tabela de rutare *generică* următoare (valoarea cu bold din paranteze; cealalta valoare, după "/", este **metrica** rutei, care la RIP este numărul de hopuri pînă la reţeaua respectivă; în cazul altor protocoale, metrica este diferită).

```
Router# sh ip route

     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
R       10.0.0.0/8 [120/2] via 172.16.1.2, 00:00:05, FastEthernet1/0
C       10.1.1.0/24 is directly connected, FastEthernet0/0
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
R       172.16.0.0/16 [120/3] via 172.16.1.2, 00:00:05, FastEthernet1/0
C       172.16.1.0/24 is directly connected, FastEthernet1/0
R       172.16.2.0/24 [120/1] via 172.16.1.2, 00:00:05, FastEthernet1/0
S       172.16.3.0/24 [1/0] via 10.1.1.2
R    192.168.1.0/24 [120/2] via 172.16.1.2, 00:00:05, FastEthernet1/0
R    193.1.1.0/24 [120/2] via 172.16.1.2, 00:00:05, FastEthernet1/0
```

Rutele statice au o distanţă 1 în timp ce cele învăţate prin RIP au 120. De aceea, ruterul va prefera rutele statice, şi va folosi RIP doar dacă acestea devin indisponibile. Pentru a pune o ruta statică de rezervă, care sa nu fie preferată faţă de RIP, i se va pune o distanţă mai mare, de exemplu 130:

```
Router0(config)#ip route 192.168.1.0 255.255.255.0 172.16.1.2 130
```

2.6 Sumarizare

Prin *sumarizare* se înţelege concentrarea mai multor rute/destinaţii într-una singură (*n* rute din interiorul unei anumite zone sînt percepute în afara acelei zone ca o singură rută). Acest lucru presupune o adresare continguă în acea zonă – dacă subneturile aceluiaşi net sînt distribuite aleator pe mai multe rutere, nu se poate face sumarizarea.

În cazul sumarizării mai multor reţele într-una singură, această reţea este uneori numită *super-reţea (supernet)* prin analogie cu denumirile *net – subnet*.

Observaţie: termenul *supernet* este folosit mai ales pentru a combina mai multe reţele într-o reţea superioară limitei implicite de clasă, aşa cum termenul *subnet* se foloseşte atunci cînd se fac reţele mai mici decît cele specificate de clasă. Două reţele de clasă C (/24) combinate într-o reţea /23 formează un supernet; 2 subneturi /30 combinate într-o reţea /29 nu se consideră de obicei un supernet, ci tot un subnet.

Avantajele sumarizării:

- reducerea dimensiunii tabelelor de rutare
- dacă o reţea din "spatele" unui ruter, anunţată de acesta în cadrul unei super-reţele, "cade" din cauza unei probleme tehnice (şi eventual îşi revine cu intermitenţă), acest lucru nu este propagat în tabela de rutare, deci celelate rutere nu trebuie să-şi modifice de fiecare dată tabelele de rutare. Problema rămîne o problemă "internă" ruterului respectiv.

Exemplul 1:
- se dau 4 reţele: 192.168.0.0/24, 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24
- se doreşte combinarea lor într-un *supernet* de mască /22 (4 reţele ocupă 2 biţi)
- supernetul scris în binar (bold) va fi 192.168.*000000**00.00000000* adică 192.168.0.0/22
- aşadar 4 reţele independente de cîte 256 adrese se pot scrie ca un supernet de 1024 adrese.

Exemplul 2:
reţelele 10.10.1.0/24, 10.10.2.0/24, …, 10.10.255.0/24 sînt percepute ca fiind reţeaua mai "mare" 10.10.0.0/16. Supernetul 10.10.0.0/16 s-a obţinut prin realizarea operaţiei "AND" logic între cei 4 octeti ai reţelelor initiale. Observaţi că nu este un supernet tipic, pentru că limita de clasă era /8.

3. Access Control Lists - ACL

3.1 Generalităţi

Un ACL permite stabilirea de reguli prin care să fie permis sau interzis accesul pachetelor cu anumite adrese IP, fie integral, fie numai către anumite destinaţii, porturi sau protocoale. Practic, folosind ACL se poate crea un *firewall* folosind un ruter Cisco. Exemple de reguli:

- hosturile 192.168.1.1-192.168.1.63 nu vor avea acces la siteurile de web dintr-o listă
- calculatoarele pe care lucrează studenţii vor avea acces numai la anumite site-uri web
- nici un host din exteriorul domeniului nu va avea acces Telnet la hosturile din interiorul domeniului, dar reciproca va fi permisă

Specificarea hosturilor afectate de o regulă se face cu perechea *adresa_IP, wildcard_mask.*

3.2. *Wildcard masks* în ACL

Un *wildcard mask* prezintă similitudini cu un *netmask*, adică permite specificarea căror biţi din adresa IP li se aplică o regulă şi cărora nu li se aplică. Mai precis, se selectează/verifică acei biţi din adresa IP care au "0" în mască şi ignoră pe cei care au "1" în mască, oarecum invers faţă de *netmask*. Dar, ***wildcard masks*** **sînt mai generale şi nu sînt neapărat egale cu inversul *netmask*.**

Cazul particular în care ele sînt egale este atunci cînd se doreşte specificarea unei întregi reţele sau subreţele, de exemplu să fie interzis sau permis traficul de la *toate* hosturile unei reţele. De exemplu, pentru a specifica "întreaga reţea 192.168.1.0/24", se scrie:

192.168.1.0 0.0.0.255

adică se verifică (0=*check*) primii 3 octeţi, care au 0 în mască, cu semnificaţia ca se verifică octeţii care desemneaza reţeaua, şi se ignoră (1=*ignore*) octetul care desemnează hosturile. Faptul că nu se verifică biţii de host este echivalent cu a spune că regula se aplică tuturor hosturilor din net.

Cazurile în care ele nu sînt egale apar atunci cînd se doreşte realizarea unor selecţii mai complexe, şi anume doar o parte din hosturile dintr-o reţea. Aceasta înseamnă că un *wildcard mask* permite să selectăm anumite hosturi *fără* să trebuiască să creem subneturi doar pentru aceasta.
De exemplu, fie reţeaua 192.168.1.0/24, care nu este subnetată. Asignăm hosturile cu numere mari (de la 193 în sus, care ar fi echivalente cu ultimul subnet de lungime /26) pentru calculatoarele celor din management şi hosturile 1-192 pentru angajaţi. Din punct de vedere al rutării, toate hosturile trebuie să acceseze internetul, deci nu are sens să subnetăm reţeaua, dar dorim să dăm acces la un anumit site de web din intranet numai celor din management. Specificăm aceste hosturi cu sintaxa:

adresa 192.168.1.***11000000***		wildcard mask	0.0.0.***00111111***	adică:
192.168.1.192			0.0.0.63	în zecimal

Aşadar ACL-ul va verifica ca hosturile să înceapă cu **11** în binar, chiar dacă netmaskul reţelei rămîne 255.255.255.0 (netmaskul nu apare în ACL). Mai mult, am putea adăuga încă o regulă, împărţind în continuare hosturile > 192 astfel: primele pentru "lower management" şi ultimele pentru "upper management", diferenţiind în continuare în ACL-uri restricţii suplimentare (nu toţi să aibă acelaşi acces, sau aceeaşi viteză, etc).

Fiecare regulă poate avea propriul *wildcard mask*. Putem crea o cu totul altă regulă, care să funcţioneze în paralel cu prima. De exemplu, stabilim că hosturile cu număr impar să fie plasate la etajul 1 şi hosturile cu număr par la etajul 2 al clădirii (par/impar în binar înseamnă că se termină în

binar cu 0, respectiv 1). În acest fel, dacă dorim să dăm acces numai celor cu hosturi impare la imprimanta de pe etajul 1, specificăm aceste hosturi cu combinaţia:

adresa 192.168.1.***11111111*** wildcard mask 0.0.0.***11111110*** adică:
192.168.1.255 0.0.0.254 în zecimal

unde se observă că pe poziţia unde avem "1" în *wildcard mask* nu contează ce avem în adresa IP, ACL-ul va verifica doar ultimul bit din host! (am fi putut scrie la fel de bine adresa ca 192.168.0.1 sau 192.168.0.3 sau orice care să se termine cu "1" în binar!)
În concluzie, netmaskul rămîne unic pentru procesul de rutare, în timp ce folosind diferite *wildcard masks* în diferite reguli, putem controla oricît de "fin" traficul de la anumite hosturi ale reţelei.

Anumite combinaţii mai răspîndite au următoarele prescurtări:

expresia (*adresa_IP*, masca)	prescurtarea	explicatie
A.B.C.D 0.0.0.0	**host** *A.B.C.D*	întrucît 0.0.0.0 înseamnă "verifică toţi biţii", este clar că se referă la adresa unui singur host
0.0.0.0 255.255.255.255	**any**	întrucît 255.255.255.255 înseamnă "ignoră toţi biţii", rezultă că semnificaţia este "orice adresă"

3.3 Configurarea ACL

Există 2 tipuri de ACL pentru adrese IP pe ruterele Cisco:

- **Standard ACLs** au numere între 1-99, şi nu permit specificarea decît a *adresei sursă* a pachetului IP. De aceea, ele se plasează în reţea cît mai aproape de *destinaţia* afectată de ACL, în ideea că pachetele pot ajunge la acea destinaţie pe mai multe căi.

configurarea se face astfel:

```
Router(config)# access-list număr deny|permit [remark comentariu]
adr_sursa wildcard_mask [log]
```

Se va specifica numai una din actiunile **deny** (interzice) sau **permit** (permite). Opţiunea **remark** permite scrierea unui comentariu despre listă; opţiunea **log** generează o înregistrare în fisierul log al ruterului atunci cînd un eveniment declanşează aplicarea ACL-ului.

Exemple:

```
access-list 1 deny host 192.168.1.1
```
nu permite traficul de la hostul specificat, filtrînd efectiv pacherele de la acel host

```
access-list 2 permit 192.168.1.0 0.0.0.255
```
permite traficul de la toate hosturile din netul 192.168.1.0/24

- **Extended ACLs** au numere între 100-199 sau 2000-2699. Ele permit specificarea *adresei sursă, adresei destinaţie, protocolului* şi *portului* ceea ce le face mult mai versatile. Vor fi studiate într-o lucrare viitoare.

Simpla *definire* a unui ACL ca mai sus nu are nici un efect asupra ruterului. Mai departe, el trebuie *folosit*. Cea mai simplă situaţie e cînd ACL-ul e ***plasat pe o interfaţă a ruterului*** (2 alte situaţii de utilizare a unui ACL vor fi prezentate în 3.6 şi 3.7). Pentru aceasta se foloseşte comanda (în modul de configurare al interfeţei care se doreşte):

```
Router(config-if)# ip access-group număr_ACL in|out
```

unde sensul de intrare (in) sau ieşire (out) se consideră privind "din interiorul ruterului". De exemplu, pentru a plasa lista cu numărul 1 pe interfaţa S0/1 astfel încît să afecteze pachetele care intră în ruter prin acea interfaţă (a cărei adresă IP a fost în prealabil configurată):

```
Router(config)# int S0/1
Router(config-if)# ip access-group 1 in
Router(config-if)# CTRL-Z
```

Folosind prefixul "no" se poate scoate un ACL de pe interfaţa (no ip access-group…); astfel, el va fi dezactivat fără a fi necesara ştergerea ACL-ului. De asemenea, se poate folosi "no" pentru a şterge complet un ACL: no access-list 101 …

3.4 Observaţii importante privind ACL

1) Pe o interfaţă şi un sens se aplică *doar un singur ACL* (definit cu `access-group` număr), dar un ACL poate avea mai multe reguli. De exemplu sintaxa:

```
access-list 10 permit host 192.168.1.1
access-list 10 deny host 192.168.1.2
```

crează ACL-ul 10 cu 2 reguli. În continuare se pot adăuga şi alte reguli.

2) La definirea unei reguli intr-un ACL (indiferent că e cu nume sau cu număr), regula se adaugă ***la sfîrşitul celor deja existente***. Acest comportament este diferit de, spre exemplu, comenzi ca "`ip addr x.x.x.x`" sau "`shutdown | no shutdown`" pe o interfaţă, la care *ultima* comandă care se dă este reţinută şi şterge efectul eventualelor comenzi date înainte. La ACL, folosirea prefixului "no" şterge toate regulile (nu se pot şterge sau modifica selectiv reguli).

3) Este important de înţeles că, atunci cînd ruterul primeşte un pachet pe o interfaţă, *el verifică fiecare regulă faţă de acel pachet, **în ordinea** în care au fost definite regulile, şi efectueaza **prima** acţiune de tip permit sau deny care se potriveşte cu acel pachet, după care verificarea incetează.*
Consecinţă: ***este esenţială ordinea** definirii regulilor intr-un ACL.*

De exemplu, să presupunem că dorim să permitem traficul de la 192.168.1.1 dar să interzicem traficul de la celelalte hosturi din 192.168.1.0. Următoarea definiţie este corectă:

```
access-list 1 permit host 192.168.1.1
access-list 1 deny 192.168.1.0 0.0.0.255
```

iar următoarea definiţie este greşită:

```
access-list 1 deny 192.168.1.0 0.0.0.255
access-list 1 permit host 192.168.1.1
```

întrucît pachetul de la 192.168.1.1 a fost deja respins de prima regulă, cu `deny 192.168.1.0`, *care este mai generală* (192.168.1.1 face parte din netul 192.168.1.0), după care ruterul nu mai verifică şi regulile următoare pentru acel pachet. Rezultă că trebuie să definim, în ordine, *mai intîi regulile mai puţin generale şi apoi regulile mai generale.*

4) Dacă pe un ruter nu este definit nici un ACL, implicit tot traficul este permis. Dar, dacă este definit chiar şi un singur ACL, tot traficul care NU este specificat de acel ACL este implicit interzis. Prin urmare dacă pe un ruter sînt definite N reguli, este ca şi cînd ar exista o a N+1-a regula care spune **`deny any`**

De aceea, dacă se doreşte implicit comportamentul permisiv, ca tot ce nu a fost specificat în ACL să fie permis, trebuie să fie adăugată ***la sfîrşit*** regula **`permit any`**

Aceasta regulă va avea ca rezultat permiterea pachetului respectiv (orice pachet corespunde descrierii *any*) şi ruterul va ieşi din verificare. Regula N+1, cea cu *deny*, există în continuare, dar ruterul nu mai "ajunge" la ea.

Atenţie! în lipsa unui `permit any`, este posibil să blocăm inclusiv *routing updates* şi rutarea să înceteze să mai funcţioneze !

Faptul că ordinea este foarte strictă "complică" un pic scrierea de ACL-uri, întrucît dacă s-a greşit, nu se mai poate corecta greşeala decît ştergînd tot ACL-ul cu "no" şi reluînd procesul de la capăt (sau editînd fişierul de configurare al ruterului în mod offline).

3.5 Testarea ACLs

Pentru a testa funcţionarea ACL, trebuie generat trafic de tipul specificat în ACL şi verificat dacă comportamentul este cel dorit. Întrucît un ruter poate avea mai multe interfeţe, uneori trebuie specificat de pe ce interfaţa "pleacă" traficul.

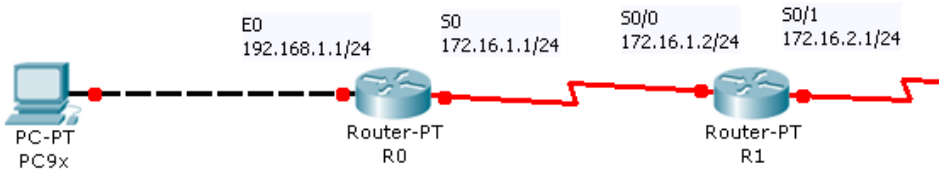

Figura 2

De exemplu, în figura 2, presupunem că am creat un ACL pe R1 care să impiedice traficul de la adresele 192.168.1.0/24 către 172.16.2.1/24. Dacă sîntem conectaţi pe R0 şi dăm comanda "ping 172.16.2.1" putem constata că adresa sursă a pachetelor IP nu este 192.168.1.1, ci 172.16.1.1, întrucît ambele sînt adrese valide ale R0, iar cea de-a doua adresă este mai aproape de destinaţie. De aceea, trebuie să putem selecta manual interfaţa sursă a pachetelor pentru ping şi telnet.

- Pentru a forţa pachetele de tip "ping" să plece de pe o anumită adresă sursă se foloseşte **extended ping**, adică comanda ping fără parametri. Atunci ruterul ne va întreba protocolul dorit, adresa sursă, numărul de pachete, etc. La fiecare întrebare raspunsul implicit este afişat între paranteze drepte şi se poate da apasînd ENTER. Remarcaţi că la "extended commands" trebuie raspuns *y*.
- Dacă testăm cu Telnet în loc de ping, putem forţa pachetele de tip "Telnet" să plece de pe o anumita adresă sursă folosind comanda:

```
ip telnet source-interface adresa_ip
```

- *Atenţie !* ACLs dintr-un ruter nu verifică *traficul generat local*, pe ruter. Ele verifică traficul provenind de la alte echipamente conectate la acel ruter.

Vizualizarea ACLurilor se poate face cu comenzile:

comanda	*explicaţie*
`show ip interface`	arată detalii despre interfaţă, inclusiv ACL-urile aplicate
`show access-lists [number]`	arată toate ACLurile definite, sau numai cea specificată
`show running-config`	arată toată configuraţia, inclusiv ACLurile

3.6 Utilizarea ACL pentru *telnet*

Un caz particular de plasare a unui ACL nu pe o interfaţă, ci pe sesiunile *Telnet*, se face folosind comanda `access-class,` astfel:

1) se defineşte un *access-list* în modul obişnuit, de exemplu:

```
access-list 99 permit 192.168.1.0 0.0.0.255
```

2) se plasează această listă pe liniile de tip *vty* (virtual terminal) adică *Telnet*. În exemplul care urmează, se permite doar o singură sesiune de telnet (din cele 5 posibile pe 2500) şi doar de la adresele specificate:

```
line vty 0
 password class
 access-class 99 in
 login
line vty 1 4
 no login
```

3.7 Utilizarea ACL pentru filtrarea rutelor

După cum se ştie, un protocol de rutare (RIP, OSPF, etc) funcţionează prin transmiterea de actualizări ale tabelei de rutare (*routing updates)* către toate ruterele direct conectate. Uneori însă acest lucru nu este dorit. De exemplu, pe figura 1, dacă există protocol de rutare între cele 2 rutere, nu ar trebui trimise *updates* către PC, din următoarele motive:

- PC-ul are o singură cale de ieşire, prin ruterul său direct conectat (R0), aşadar este suficient ca acest ruter să fie definit ca *default gateway* în configuraţia PC-ului. *Routing updates* trimise către PC doar consumă inutil banda şi puterea de procesare a ruterului

- *routing updates* care ajung pe PC oferă informaţii despre tabela de rutare, care pot fi folosite în scop maliţios (risc de securitate)

Pentru a opri transmiterea acestor *updates* se pot folosi 2 metode:

1) interfaţa dintre ruter şi PC poate fi declarată *passive-interface*. O astfel de interfaţă nu va transmite nici un fel de *update* şi se declară astfel:

```
R0(config)# router rip
R0(config-router)# passive-interface E0
```

2) o metodă mai versatilă presupune folosirea unui ACL în care să se specifice care reţele vor fi permise şi care vor fi interzise în *routing updates*. În acest fel se poate controla foarte precis rutarea; de exemplu, se poate dori ca un ruter de la marginea reţelei (*Border Router*) să nu trimită în exterior *updates* despre unele reţele interne, fie din considerente de securitate, fie pentru a reduce dimensiunea tabelei de rutare.

De exemplu, în figura 1, reţeaua 192.168.1.0 dintre ruter şi PC trebuie ascunsă în *updates* trimise de R1 în exterior; se defineşte un ACL care să interzică *doar* acea reţea:

```
R1(config)# access-list 50 deny 192.168.1.0 0.0.0.255
R1(config)# access-list 50 permit any
```

Se aplică ACL-ul folosind cuvîntul cheie **`distribute-list`**, care are efectul unui filtru:

```
R1(config)# router rip
R1(config-router)# network ....        (se definesc toate reţelele necesare)
R1(config-router)# distribute-list 50 out
```

Observaţii:

- filtrul se poate aplica şi în cazul altor protocoale de rutare, nu doar RIP
- cuvîntul-cheie `out` semnifică faptul că nu vor fi *trimise* reţelele respective în *updates*. Este posibilă specificarea folosind cuvîntul-cheie `in` pentru a nu permite *primirea* de *updates* despre anumite reţele;
- în acest exemplu, plasarea filtrului interzice trimiterea de *updates* pe toate interfeţele, ceea ce nu afectează funcţionarea rutării în domeniul PC-R0-R1 (R1 a aflat de această reţea de la R0, deci oricum nu o va trimite înapoi). În anumite cazuri însă, restricţia se poate aplica doar pe o interfaţă, adăugînd cuvîntul-cheie *interface*:

```
R1(config-router)# distribute-list 50 out interface S0/1
```

sau cu sintaxa simplificată:

```
R1(config-router)# distribute-list 50 out S0/1
```

Partea 2. Desfăşurare; exerciţii

Faza 1. Pregătirea topologiei de test; rutare statică

1) Se va realiza topologia din figura 3, configurînd pe fiecare ruter:

- *hostname*-ul corespunzător
- adresele IP ale interfeţelor (pentru PC-uri adresa este scrisă pe etichetă, iar interfaţa corespunzătoare de pe ruter este .1)
- *Observaţie: nu contează la care dintre cele 3 rutere sînt legate cele 2 PC-uri.* Alegeţi atunci cînd realizaţi topologia.
- se vor configura cele 2 interfeţe *loopback* numite *lo1*
- clock-ul (64000) în cazul interfeţelor seriale DCE
- `no shutdown` pe interfeţe

- parola pe telnet va fi *class*; nu se va pune parolă de enable.

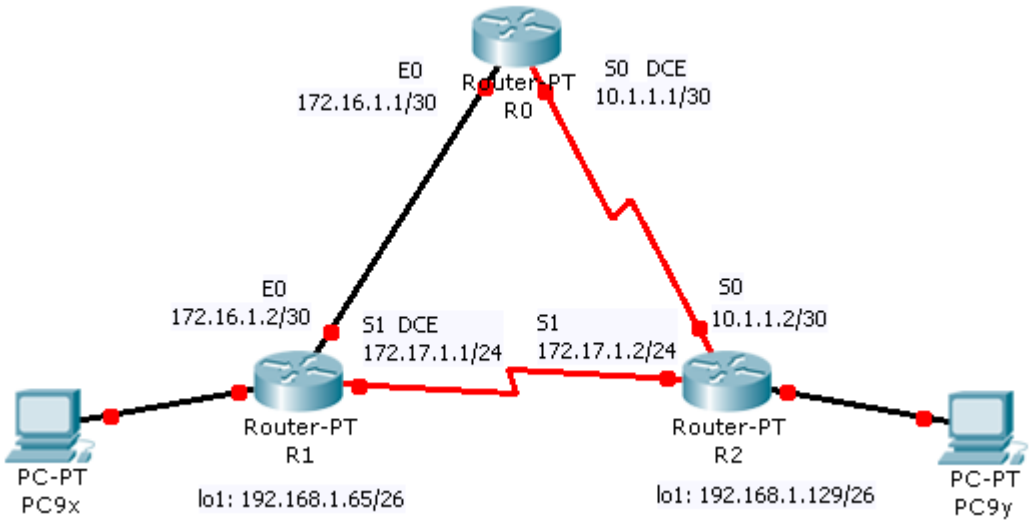

Figura 3

Q1. Desenaţi topologia corespunzătoare echipei, notînd numele de pe etichetele ruterelor efectiv folosite (alături de R0,R1,R2 pentru a uşura identificarea pe parcursul lucrării), numele interfeţelor şi toate adresele IP, inclusiv legăturile cu PC-urile.

Observaţie: pentru ca ruterul sa nu ne mai scoată automat din modul "enabled" după un timp de inactivitate, se dau comenzile:

```
line con 0
no exec-timeout
```

Se verifică folosind ping conectivitatea între vecini (inclusiv PC-urile).

Se urmăreşte acum ca R0, R1 să-şi poată da reciproc ping la orice interfaţă serială sau Ethernet. Pe R0 se defineşte o rută statică către R1/S1:

```
ip route 172.17.1.0  255.255.255.0  172.16.1.2
```

Se încearcă ping de pe R0 pe 172.17.1.1

Q2: Funcţionează ping ?
Q3: Instalaţi o rută statică pe R1 pentru a putea da ping către R0/S0; ce comandă aţi folosit ? testaţi.

Se observă că la ping se specifică doar adresa destinaţie. Nu putem verifica, de exemplu, dacă merge ping de pe R1/lo1 pe R0/E0. Pentru a specifica la ping şi adresa sursă, nu doar cea destinaţie, trebuie folosit *ping extins*, dînd comanda ping fără parametri şi răspunzînd apoi la întrebări ca mai jos (cu bold, ENTER în rest); de exemplu, pe R1:

```
R1# ping
Protocol [ip]:
Target IP Address: 172.16.1.1
Repeat count [5]:
Datagram size [100]:
Timeout în seconds [2]:
Extended commands [no]: y
```

```
Source address or interface: 192.168.1.65
Type of service [0]:
Set DF bit în IP header [no]:
Data pattern [0xABCD]:
Loose, strict, record, timestamp, verbose [None]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.2.1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5)
```

Q4: Are succes ping ca mai sus ? de ce sau de ce nu ?

Observaţi că, pentru ca un ping (şi în general, rutarea IP) să funcţioneze, nu e suficient să fie cunoscută calea de la A la B, dacă nu e cunoscută şi calea de la B la A.

Pentru a vedea tabela de rutare de pe un ruter se foloseşte `show ip route`

Q5: Vizualizaţi tabela de pe R0; ce reţele cunoaşte acest ruter ? care este diferenţa dintre rutele marcate cu "S" şi cele marcate cu "C" ?

Q6: Adăugaţi o rută statică pe R0 către reţeaua lui lo1 de pe R1; scrieţi comanda folosită, apoi repetaţi ping extins de mai sus; funcţionează sau nu ? explicaţi.

Faza 2. Instalarea unor rute *default*

În acest scenariu se urmăreşte ca R0 să aibă acces la reţeaua 172.17.1.0 în mod redundant, fie prin R1 fie prin R2. Este o situaţie similară cu un client care are acces la internet prin 2 ISP, unul de bază şi unul de rezervă. Se va prefera conexiunea prin R1, iar cea prin R2 va fi de rezervă.

R0 ştie deja o rută către 172.17.1.0; se va şterge această rută, prefixînd comanda cu no:

```
no ip route 172.17.1.0 255.255.255.0 172.16.1.2
```

Se verifică cu show ip route pe R0 că ruta a dispărut din tabela de rutare!

Se vor configura 2 rute statice de tip default, cu distanţe administrative diferite (130 şi 140). Amintim că ruterele Cisco vor prefera distanţele administrative *mai mici* (0=cea mai buna rută, 255=rută care nu va fi folosita niciodată).
Pe ruterul R0 se vor configura:

```
ip route 0.0.0.0  0.0.0.0  172.16.1.2  130
ip route 0.0.0.0  0.0.0.0  10.1.1.2    140
```

Q7: Ce semnifică prescurtarea 0.0.0.0 0.0.0.0 ?

Testarea conexiunii se va face dînd ping la oricare din interfeţele R1/S1 sau R2/S1. Se va verifica tabela de rutare de pe R0 cu `show ip route`.

Q8. Cine apare în tabela de rutare a R0 ca "gateway of last resort" ?

Q9. În tabelă nu apare reţeaua 172.17.1.0/24. De ce merge ping ? Cum "ştie" R0 să ajungă la această reţea ?

Se va întrerupe legatura din stînga, dînd `shutdown` la interfaţa E0 a lui R0; se va verifica noua tabelă de rutare şi faptul că ping încă funcţionează către 172.17.1.2.

Q10. Cine apare acum în tabela de rutare a R0 ca "gateway of last resort" ?

Se reporneşte legatura din stînga (`no shutdown` la interfaţa E0).

Faza 3. Configurarea RIP

Se instalează un protocol de rutare de tip classful – RIP versiunea 1 (implicit, dacă nu se specifică versiunea, este v1).

Pentru a avea o topologie "curată" se şterg prin prefixare cu "no" toate rutele statice care au fost definite pînă acum:

```
Router(config)# no ip route ...
Router(config)# no ip route ...
Router(config)# exit
```

se şterge tabela de rutare:

```
Router# clear ip route *
```

Pentru configurarea RIP, trebuie listate cu cuvîntul-cheie `network` toate ***reţelele direct conectate*** la ruter (inclusiv ale loopback-urilor şi ale PC-urilor, şi se specifică doar adresa de reţea, nu şi masca de reţea; ***nu se listează*** reţele care nu sînt direct conectate! Reţelele listate vor fi anunţate către ruterele vecine.

```
Router(config)# router rip
Router(config-router)# network ...
Router(config-router)# network ...
```

Q11. Ce reţele aţi configurat pe fiecare ruter?

Se examinează tabela de rutare cu `sh ip route`; se verifică funcţionarea RIP cu `sh ip protocols`.

Q12. Ce literă apare în faţa unei reţele învăţate prin RIP, listată în tabela de rutare ?
Q13. Pe R0, Apar reţelele 192.168.1.64/26 şi 192.168.1.128/26 în tabela de rutare? dacă nu, ce apare în locul lor?
Q14. Se verifică cu `sh running-config` pe R1 cum arată linia *network* pentru reţeaua 192.168.1.64. Ce s-a întîmplat?

Se încearcă ping între oricare 2 noduri ale reţelei. Se observă că nu funcţionează în toate cazurile, ceea ce înseamnă că RIP nu e configurat corect.

Q15. Funcţionează ping de pe R1 pe 192.168.1.129 ? de ce sau de ce nu ?

În acest caz, prin folosirea unui protocol classful, se cheamă că RIP *sumarizează rutele*, adică combină toate rutele către subneturi într-o singură rută către netul corespunzator (adică un *summary* – un net care este un "rezumat" al subneturilor sale). De exemplu, ruta către 192.168.1.64/26 este înlocuită cu 192.168.1.0/24. Aceasta constituie o problemă majoră în cazul în care există 2 subneturi ale aceluiaşi net, cum sînt cele 2 subneturi reprezentate de *lo1* de pe R1 şi R2, *şi aceste subneturi sînt ne-contingue* (în cazul nostru, sînt separate prin 2 alte neturi ataşate lui R0).

Se vizualizează pe R1 trimiterea de *routing updates* cu comanda `debug ip rip`.

Q16. Ce reţele trimite R1 intr-un *update* ? sînt subneturile corecte? Către ce adresă se trimite tabela de rutare? (vezi mesajele *sending v1 update via...*)

Se opreşte debugging-ul cu `undebug all`

Dacă cele 2 subneturi erau amîndouă ataşate de exemplu lui R1, nu era nici o problemă, pentru că prin sumarizare R1 anunţa că ştie o rută către 192.168.1.0/24, orice pachet către .33 sau .65 ajungea pe R1, şi mai departe R1 distribuia pachetul către subnetul corect. În acest caz, *sumarizarea ar fi constituit un avantaj*, pentru ca în loc sa transmita *updates* despre 2 rute, R1 transmite despre o singură rută, reducînd tabela de rutare transmisă vecinilor. *Fără sumarizare, Internetul de azi nu ar putea funcţiona, tabelele de rutare devenind uriaşe!* În cazul subneturilor necontingue însă, sumarizarea la limita de clasă oferă informaţia falsă că cele 2 subneturi sînt disponibile simultan pe R1 şi R2.

Se trece de la RIP v1 la RIP v2. Este posibil să se adauge pur şi simplu liniile "`version 2`" şi "`no auto-summary`" în modul de configurare `router rip`; uneori însă, pot apare probleme în funcţionare, de aceea calea cea mai sigură este: mai întîi pe fiecare ruter se opreşte protocolul RIP v1:

```
Router(config)# no router rip
Router(config)# CTRL-Z
```

Se configurează RIP v2 la fel ca v1, adăugînd 2 linii în plus:

```
Router(config)# router rip
Router(config-router)# version 2
Router(config-router)# no auto-summary
Router(config-router)# network ....
```

Observaţi comanda suplimentară `no auto-summary`. Aceasta nu este legată de diferenţa dintre v1 şi v2 (după cum spuneam, ruterele din Internet folosesc sumarizarea din plin), dar pe *această* topologie evident că nu dorim sumarizarea, datorită existenţei subneturilor ne-contingue.

Se vizualizează din nou cu `sh running-config` cum arată configurarea RIP. Se observă că, deşi am dat comanda suplimentară, rutele sînt arătate tot sumarizate la limita de clasă; aceasta este o restricţie a lui RIP: chiar dacă se va vedea că în tabela de rutare el le afişeaza corect, la `sh run` comportamentul este neschimbat.

Se vizualizează tabela de rutare pe R1 cu `sh ip route`.

Q17. Ce linie de pe ecran indică efectul comenzii `no auto-summary`?

Se verifică că acum e posibil de pe R1 să dăm ping către toate destinaţiile.
Se vizualizează mesajele *routing updates* dintre rutere folosind `debug ip rip`.

Q18. Scrieţi rutele trimise de R1 în *routing updates* observînd dacă se respectă măştile subneturilor. Ce semnifică valoarea *metric* raportată pentru fiecare reţea? Către ce adresă trimite R1 tabela de rutare? (vezi mesajele *sending v2 update via ....*)

Deşi, în urma învăţării, fiecare ruter cunoaşte rutele către toate reţelele, observaţi că ruterul R2 nu trimite către R0 reţelele primite de la acesta din urmă, ci numai propriile reţele.

Q19. Care e cauza acestui comportament?

Q20. În cazul lui R1, trimite RIP *updates* prin toate interfeţele? sînt toate aceste *updates* necesare? care nu sînt necesare?

Se opreşte debugging-ul cu `undebug all`.

Q21. Verificaţi ping şi între cele 2 PC-uri conectate la topologie. Funcţionează? de ce?

Se opreşte trimiterea de *updates* pe acele interfeţe pe care nu sînt conectate alte rutere ci doar PC-uri (care nu rulează RIP deci oricum ignoră *updates*) astfel:

```
Router(config)# router rip
Router(config-router)# passive-interface nume_interfaţă
```

Se verifică cu `debug ip rip` că nu se mai trimit updates pe acea interfaţă. Apoi se opreşte din nou debugging-ul.

Q22. Nu era mai simplu să oprim updates pe acea interfaţă, eliminînd network-ul corespunzător PC-ului din RIP? ce problemă ar fi apărut în acest caz?

Faza 4. Legarea topologiei la "internet"

Se va simula "internetul" ca fiind o nouă interfaţă loopback *lo2,* conectată la R2. Se configurează această interfaţă cu adresa 172.30.1.1/24.

Se configurează reţeaua interfeţei respective ca *default-network*, adică reţeaua în care se trimit toate pachetele pentru care nu există destinaţie explicită:

```
R2(config)#ip default-network 172.30.1.0
```

Q23. Se verifică pe R0 dacă se poate da ping către 172.30.1.1. De ce sau de ce nu ?

Q24. Se examinează tabela de rutare de pe R0 folosind `show ip route`. Există *gateway of last resort* (echivalent cu *default gateway* în Windows/Linux) ?

Se configurează R2 să anunţe ruta *default* către toate celelate rutere, astfel:

```
R2(config)# router rip
R2(config-router)# default-information originate
```

Q25. Se aşteaptă propagarea rutelor. Care sînt cele *două* linii afişate de `sh ip route` pe R0 care transmit informaţii despre ruta *default* şi ruterul respectiv? (*gateway of last resort).* Cu ce simbol este afişată (ca prefix) intrarea din tabela de rutare pentru ruta *default*?

Q26. Incercaţi ping de pe R1 către "internet" (172.30.1.1). Funcţionează? Explicaţi.

Q27. Pe care dintre cele 3 rutere apare setat *gateway of last resort*, şi pe care nu ? explicaţi !

Faza 5. Redistribuirea rutelor

Pentru ca RIP să anunţe o rută, nu este nevoie neapărat ca aceasta să fie o rută definită cu "network" în configurarea sa, sau ruta default. În general, RIP poate anunţa către vecinii săi şi rute învăţate pe alte căi. Se numeşte *redistribuire* faptul că RIP distribuie "mai departe" o rută pe care el-însuşi a învăţat-o din altă sursă (şi nu prin configurare directă).

Definim pe R2 o rută statică oarecare:

```
R2(config)# ip route 192.168.100.0 255.255.255.0 null0
```

Interfaţa **`null0`** nu există şi de aceea ruta nu va funcţiona, dar am creat-o doar pt. că dorim să vedem că se propagă; instruim RIP să redistribuie rutele statice:

```
R2(config)# router rip
R2(config-router)# redistribute static
```

Verificăm acum pe R1 şi R0 tabela de rutare.

Q28: Ce nouă rută a apărut ?

Faza 6. Introducerea de restricţii folosind ACL

Pentru fazele care urmează, pentru a elimina ambiguitatea căii de urmat între 2 destinaţii, în scopul unor configurări mai uşoare ale ACL, ***se va elimina legătura directă între ruterele la care sînt legate cele 2 PC-uri*** (seriala de jos între R1 şi R2 pe fig. 3; modificaţi în funcţie de topologia voastră). Opriţi interfeţele seriale respective cu `shutdown`, nu scoateţi fizic cablul.

Urmăriţi reactualizarea tabelei de rutare dintre R1 şi R2 (după timpul de convergenţă). Verificaţi că este în continuare posibil să daţi ping între diferite destinaţii care nu sînt direct conectate. Este important să meargă ping acum, pentru ca atunci cînd testaţi ACL-urile, dacă nu merge, ştiţi sigur că e de la ACL, nu de la o configurare defectuoasă.

Q29: Definiţi şi aplicaţi un ACL a.î. să nu fie permis traficul de pe PC9x pe R2, tot restul fiind permis. Explicaţi cum aţi făcut. Testaţi cu ping.

Q30. Definiţi şi aplicaţi un ACL a.î. să nu fie permis traficul de pe R2/lo1 către R0, tot restul fiind permis. Explicaţi cum aţi făcut. Testaţi cu ping. Ce fel de ping trebuie folosit?

Q31. Folosind un ACL pus pe Telnet (cu *access-class*), permiteţi Telnet pe R2 doar de pe R1/lo1. Explicaţi cum aţi făcut. Testaţi cu telnet. Ce comandă prealabilă trebuie dată pe R1 înainte de a testa?

Faza 7. Controlul anunţării reţelelor în RIP prin ACL

Considerăm că PC9x (din stînga pe fig. 3) este "privat" şi dorim să nu apară în "internet" ruta către reţeaua în care este conectat PC9x, simulînd "internetul" ca fiind loopback1 de pe R2. Vom restricţiona anunţarea acestei reţele de către R0 în *routing updates* către R2, şi vom face teste cu ping de pe R2.

(În prealabil *eliminaţi cu "no" eventualele ACL* care nu permit ping de pe R2 pe PC9x şi verificaţi că este accesibil).

Vom folosi un ACL standard care să restricţioneze (*deny*) reţeaua dintre PC9x şi R1 în updates; vom boteza acest ACL 50.

Folosind cuvîntul-cheie `distribute-list`, vom introduce acest ACL în procesul RIP de pe R0, cu efect pe interfaţa către R2 (cuvîntul-cheie `interface`; ***Atenţie: în funcţie de versiunea de IOS, cuvîntul-cheie poate lipsi,*** specificînd direct numele interfeţei)

Q32. Scrieţi comenzile cu care aţi definit acest ACL şi l-aţi aplicat în RIP.

Ştergeţi rutele de pe R2 folosind `clear ip route *` pentru a elimina rutele care există deja;

Q33. Aşteptaţi actualizarea tabelei de rutare de pe R2; verificaţi în tabelă dacă mai există ruta către PC9x. Mai funcţionează ping?

Verificaţi că mai funcţionează ping de pe R0 pe PC9x, deci filtrarea nu a afectat procesul RIP între R0 şi R1, ci doar între R0 şi R2.

Q34. Verificaţi şi scrieţi ce updates trimite R0 către R2, folosind `debug ip rip`.

Pentru a lăsa configuraţia curată, se şterge de pe fiecare ruter fişierul de configurare şi se resetează ruterul:

```
Router# erase start
Router# reload
```

şi se răspunde "no" dacă sîntem întrebaţi dacă dorim să salvăm configuraţia.