Laborator 5

Adresarea IPv4 şi IPv6

Partea 1: Adresarea IPv4; soluţii de extindere

1.1 Introducere

Adresele IPv4, scrise sub forma a 4 octeţi A.B.C.D, au 32 biţi. Numărul total al adreselor disponibile este deci 2^{32} adică cca. 4 miliarde. Mai grav, contează mai puţin numărul adreselor, cît numărul reţelelor şi subreţelelor independente care se pot crea, întrucît un grup de adrese nefolosite din reţeaua unui client nu se pot aloca cu uşurinţă în reţeaua altui client (ci doar se poate reduce reţeaua respectivă prin creşterea lungimii măştii de reţea şi împărţirea în subneturi, urmînd ca subnetul liber să fie dat altcuiva – dar acest lucru este dificil şi nu e posibil decît în anumite cazuri). Pentru a amîna momentul epuizării adreselor s-au implementat mai multe soluţii: CIDR, DHCP (care are şi alte utilizări), adresele private din RFC1918 (10.0.0.0/8, 172.16.0.0-172.31.0.0/16, 192.168.0.0/24), NAT.

1.2 CIDR

CIDR (*Classless InterDomain Routing)* se referă la renunţarea la alocarea de adrese conform împărţirii pe clasele A,B,C. În primii ani ai Internetului, anumite organizaţii sau companii cum ar fi, IBM, HP, Apple, MIT şi altele [7] au primit clase A (16 milioane de adrese) pe care în mod evident nu le folosesc integral; este dificilă acum retragerea şi realocarea acestor adrese.

De aceea, se folosesc subneturi de lungime variabilă şi se alocă fiecărui client/ fiecărei organizaţii atîtea adrese cîte are nevoie. De exemplu, o organizaţie care are nevoie de 1000 de adrese primeşte 4 reţele de lungime /24 (rezultînd un supernet de lungime /22 dacă cele 4 reţele sînt reţele clasă C contingue); aceste reţele pot fi şi subreţele ale unei clase A de exemplu, în acest caz combinarea a 4 subneturi /24 într-un subnet formează tot un subnet /22, denumirea de supernet fiind de obicei rezervată "încălcării" unei limite de clasă.

CIDR este sinonim cu adresarea şi rutarea *classless*; implementarea sa a necesitat introducerea obligatorie a măştii de reţea asociată oricărei adrese, şi transmiterea acesteia şi de către protocoalele de rutare; de exemplu, RIP v1 a trebuit înlocuit cu RIP v2.

1.3 DHCP

DHCP este folosit pentru a asigna automat adresele IP hosturilor dintr-o reţea; astfel se uşurează munca de configurare a acestora. În afara de adresa IP, prin DHCP se mai pot asigna şi alti parametri: adresa de *default gateway*, numele de domeniu, adresa serverului de DNS, etc.

Funcţionarea este următoarea: în reţea se instaleaza un server de DHCP; hosturile care au nevoie de adresă IP (pot fi la limita şi calculatoare fără hard disc sau alt mediu pe care să fie stocata configuraţia) trimit la pornire un mesaj în care cer un IP; adresa sursă a mesajului este adresa MAC a hostului, în timp ce adresa destinaţie este adresa *broadcast* a reţelei, întrucît ele nu cunosc adresa serverului de DHCP. Serverul de DHCP raspunde, trimiţînd către adresa MAC de unde a venit mesajul, un mesaj în care îi comunica hostului doritor ce adresa IP (şi alti parametri) să folosească. Această adresă poate fi aleasa aleator sau configurată dupa o tabela de corespondenta adresa MAC-adresa IP, configurată manual de către administrator.

De notat că, în anii furnizării accesului internet către clienţii de tip "*home users*" prin reţeaua telefonică, DHCP a fost şi o soluţie de economisire a adreselor IP, întrucît clienţii care se conectau telefonic nu stăteau conectaţi tot timpul; prin urmare, un ISP putea avea un milion de clienţi, din care doar 1/10 să fie conectaţi simultan, fiind suficient numărul corespunzător de adrese.

Odată cu trecerea la accesul internet prin tehnologii de tipul "*always on*" (ISDN, cablu, ADSL, etc) nu se mai pot face economii de adrese pe această cale, iar DHCP este folosit doar pentru simplificarea administrării alocării de adrese IP.

1.4 NAT

NAT (*Network Address Translation*) este principala tehnologie care a permis Internetului să se dezvolte în ciuda epuizarii spaţiului de adrese IP. În interiorul unei organizaţii se pot folosi adrese IP private, nerutabile (din RFC1918), dar hosturile cu aceste adrese nu sînt vizibile în Internet. NAT rezolvă această problemă ca în figura 1:

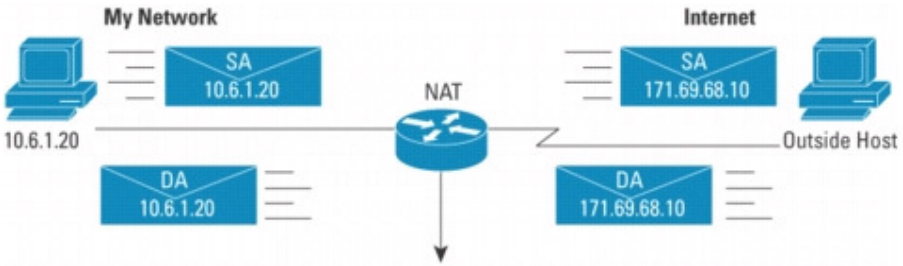

Fig. 1 sursa: [1]

O organizaţie cumpără un număr redus de adrese IP rutabile de la un ISP, dar nu le asignează hosturilor interne; acestea primesc adrese private. Un PC cu adresa privată (10.6.1.20) trimite un pachet cu această adresă sursă (**SA**) către un host din Internet. Ruterul de la graniţa organizaţiei schimbă această adresă cu o adresa rutabilă (171.69.68.10), dintre adresele cumpărate, şi trimite pachetul în Internet. De asemenea, el introduce această translaţie într-o tabelă locală. Hostul destinaţie trimite pachetul de răspuns cu adresa destinaţie (**DA**) rutabilă, şi acesta ajunge înapoi la ruter. Ruterul işi consultă tabela de translaţie şi schimbă la loc adresa cu cea din interiorul organizaţiei, astfel ca PC-ul initial nici nu va detecta că "pe drum" adresa i-a fost translatata.

Adresa privată se numeste *inside address* iar cea publică, obţinută prin translatare, se numeşte *outside address*. De notat că în general NAT poate face translaţia între orice fel de adrese, nu numai între private şi publice, aceasta fiind doar varianta cea mai răspîndită (şi cea mai utilă).

Variante de NAT:

- NAT-ul simplu, "clasic", este atunci cînd există suficiente adrese reale atfel ca toate hosturile interne să poata accesa, simultan, prin translatare, destinaţii externe. Această formă de NAT nu este utilă pentru rezolvarea crizei de adrese reale, ci doar pentru "mascarea" adreselor interne din diferite motive (de exemplu, de securitate).
- NAT-ul cu *overload* este atunci cînd numărul de adrese reale este mai mic decît numarul hosturilor interne (caz tipic). În acest caz, ruterul care face NAT va modifica şi portul sursă al pachetului, nu numai adresa sursă; 2 pachete cu aceeasi adresa *outside* vor putea fi diferenţiate prin numărul de port, şi la venirea pachetului de răspuns se refac adresa şi portul iniţiale. Această metoda funcţionează pentru că în aproape toate cazurile, *portul sursă* al unui pachet este ales la întîmplare de către sistemul de operare cînd trimite pachetul (utilizatorul nici nu îl vede), deci nu contează că este translatat. Numai *portul destinaţie* este important şi este binecunoscut (de exemplu, 80 pt web). Există, totuşi, excepţii, care fac ca nu toate aplicaţiile să funcţioneze în spatele NAT.
- PAT (*Port Address Translation*) sau NAPT (*Network Address Port Translation*) este cazul precedent dus pînă la extrem, şi anume folosirea unei singure adrese *outside* rutabile şi diferenţierea între adresele *inside* numai dupa numarul de port. Această metodă este folosita pe

scara larga de către organizatiile mici sau home-users, care cumpara un singur IP real. Sub Linux această varianta se mai numeşte uneori şi *IP Masquerading*.

- *Port Forwarding* este un caz particular în care un port clar specificat de pe un host este translatat într-un alt port (cu acelaşi număr sau număr diferit) de pe alt host (cu adresa diferita). O aplicaţie directă este instalarea de servere pentru anumite servicii (web, mail, etc) în interiorul unei organizaţii care foloseşte adrese private. Porturile destinaţie ale serviciilor respective, de pe adresa/adresele reale ale ruterului care face NAT, accesate din exterior, sînt translatate către porturile corespunzătoare ale serverelor din interior. Astfel, serviciul respectiv pare dinspre Internet că ruleaza pe ruter, cînd de fapt rulează pe un host din interior.

  Din punct de vedere al alegerii adreselor, NAT poate fi *static* sau *dinamic*.

- NAT static se configurează specificînd explicit, printr-o singura comandă, adresa *inside* şi adresa *outside*. Pentru figura 1:

```
Router(config)# ip nat inside source static 10.6.1.20 171.69.68.14
```

- NAT dinamic este mai flexibil şi permite translatarea unui domeniu în alt domeniu. Se configureaza în următoarele etape:

  1. se defineşte un domeniu de adrese din care să se ia adresa *outside*; dacă este vorba de PAT, în locul domeniului avem o singură adresă, sau se poate specifica numele unei interfeţe şi se asignează automat adresa interfeţei respective. Pentru situaţia din figura 1, presupunînd că avem 14 adrese publice:

```
Router(config)# ip nat pool test1 171.69.68.1 171.69.68.14 netmask
255.255.255.0
```

  2. se defineşte un access-list care să specifice domeniul adreselor *inside* care vor fi translatate; întrucît un access-list se termină implicit cu un *deny any,* nu vor fi translatate alte adrese.

```
Router(config)# access-list 1 permit 10.6.1.0 0.0.0.255
```

  3. se asociază domeniul din (2) cu cel din (1).

```
Router(config)# ip nat inside source list 1 pool test1
```

- Indiferent de tipul de NAT, pentru interfaţa internă şi cea externă, se specifică acest lucru folosind cuvintele cheie *inside* şi *outside*:

```
Router(config)# int S0/0
Router(config-if)# ip nat inside
Router(config-if)# int S0/1
Router(config-if)# ip nat outside
```

Partea 2: Adresarea IPv6

2.1 Reprezentarea adresei

Soluţiile prezentate anterior nu fac decît să amîne momentul epuizării adreselor IP. Acest moment a fost iniţial pronosticat pentru 2008 şi între timp a fost împins înspre 2010-2013. Pentru a rezolva definitiv problema numărului insuficient de adrese IPv4, s-au creat adresele IPv6, care au 128 de biţi în locul celor 32 de biţi ai adreselor IPv4; astfel, sînt posibile în total circa $3.4 \cdot 10^{38}$ adrese.

Scrierea adreselor IPv6 se face pe 8 grupuri de cîte 16 biţi; fiecare grup se reprezintă ca 4 cifre *hexazecimale*, întrucît reprezentarea cu cifre zecimale ca în adresele IPv4 ar duce la adrese prea lungi. Separarea între grupuri se face cu semnul ":". De exemplu:

AAAA:BBBB:CCCC:DDDD:0000:1111:2222:3333

Pentru a simplifica scrierea se folosesc 2 reguli care se aplică adreselor care conţin zerouri în interior:

Regula 1: zerourile iniţiale dintr-un grup se pot omite. De exemplu, adresa:
00AB:0ABC:0000:0000:1234:0001:0001:0001 se poate scrie:
AB:ABC:0:0:1234:1:1:1

Regula 2: unul sau mai multe grupuri adiacente care conţin doar zerouri se pot omite complet, înlocuindu-se cu "::". Această regulă se poate aplica *o singură dată* într-o adresă, în caz contrar nu se mai ştie cîte grupuri de "0" au fost prescurtate. Exemple:

1:0:2:3:4:5:6:7	se poate scrie	1::2:3:4:5:6:7		
ABCD:0:0:0:0:0:0:1234	se poate scrie	ABCD::1234		
0:0:0:0:0:0:1:A	se poate scrie	::1:A		
1:0:0:0:0:0:0:0	se poate scrie	1::		
1:0:0:0:1:0:0:1	se poate scrie	1::1:0:0:1	sau	1:0:0:0:1::1
0:0:0:0:0:0:0:0	se poate scrie	::		

2.2 Împărţire; subnetting

Adresele IPv6 sînt de mai multe tipuri: *unicast*, *multicast* şi *anycast*. Spre deosebire de IPv4, în IPv6 nu există adresă *broadcast*. Adresele unicast sînt similare cu cele IPv4 şi identifică o singură interfaţă a unui host. Adresele multicast identifică un grup de hosturi; un pachet trimis către o adresă multicast este trimis tuturor hosturilor din grup. Adresele *anycast* sînt similare cu cele multicast, cu deoseburea că un pachet trimis către o astfel de adresă e trimis doar membrului cel mai apropiat al grupului. O aplicaţie posibilă a adreselor anycast este pentru site-uri *multi-homed,* adică avînd multe adrese IP distribuite geografic, cum este de exemplu www.google.com; un pachet către un astfel de site va fi rutat către cel mai apropiat membru al grupului unicast, în funcţie de cum este definit conceptul de "apropiat" ca metrică în protocolul de rutare folosit.

Adresele unicast sînt la rîndul lor de mai multe tipuri, printre care categoriile mai importante sînt: globale, *link-local* şi *site-local.*

Adresele globale sînt adresele "obişnuite", publice, similare cu adresele IPv4. Adresele *link-local* sînt nişte adrese speciale, folosite doar intern, pe acele linkuri care au doar 2 capete (de exemplu linkuri seriale), şi nu sînt vizibile decît de către hostul curent şi hostul de la celălalt capăt al linkului. Ele nu sînt accesibile din restul reţelei. Adresele globale *site-local* sînt oarecum similare cu adresele IPv4 private definite de RFC1918; în 2003 însă, ele au fost propuse spre retragere

(RFC3879) din următoarele motive: spaţiul mare de adrese IPv6 nu mai justifică alocarea unor adrese private, fiecare organizaţie putînd obţine cu uşurinţă oricîte adrese globale; folosirea adreselor private în ultimii ani a creat probleme, atunci cînd aceste adrese au ajuns, involuntar, dar mai des decît s-a crezut iniţial, în internet, în adrese de site-uri sau de mail – aceste situaţii generează trafic inutil în internet şi sînt greu de rezolvat căci nu este evident cine e "sursa".

Adrese unicast globale: Aşa cum o adresă IPv4 unicast este împărţită în 2 secţiuni de lungimi *variabile* numite *network* (incluzînd şi eventualul *subnetwork*) şi *host*, o adresă IPv6 unicast *de tip global* avînd cîmpurile A:B:C:D:E:F:G:H este compusă din 3 secţiuni de lungimi *fixe*:

- A:B:C (48 biţi) se numeşte *site prefix*-ul şi este alocat de către un ISP. Este similar cu *network* din IPv4. Este împărţit în mai multe sub-cîmpuri, pentru a permite o ierarhizare între ISP mari (*top-level*) care la rîndul lor distribuie blocuri de adrese către ISP mai mici, etc.

  Primul cîmp (primii 3 biţi) este fixat, momentan, la valoarea binară 001. Aceasta înseamnă că adresele unicast globale sînt între 2000:x:x:…. şi 3FFF:x:x:… (1/8 din spaţiul total IPv6). Pentru prefix rămîn disponibili 48-3= 45 biţi care se pot împărţi pe niveluri de ierarhizare.

  Primii 13 biţi din prefixul rămas formează ceea se se cheamă TLA ID (*Top-Level Aggregation Identifier*) şi permite împărţirea spaţiului de adrese la 2^{13}=8192 ISP "mari" – adică ISP care menţin tabela de rutare globală în Internet. O tabelă de maxim 8192 rute, care trebuie cunoscută de către *toţi* aceşti ISP mari, a fost considerată un compromis tehnic, corespunzător puterii de calcul a ruterelor actuale.

  Restul de 45-13= 32 de biţi se împart între un cîmp momentan rezervat (numai biţi 0) şi cîmpul NLA ID (*Next-Level Aggregation ID*). În acest spaţiu se crează ierarhii pentru organizaţiile de tip ISP de ordinul 2 şi alte organizaţii care împart adresele IP. Existenţa unui spaţiu rezervat permite dezvoltări ulterioare.

- D (16 biţi) se numeşte *subnet ID* sau SLA ID (*Site-Level Aggregation ID*) şi este similar cu subnetul IPv4, doar că nu este opţional (în cazul IPv4, dacă se foloseşte adresarea classful, nu există subneturi).

  Acest cîmp este la dispoziţia organizaţiei finale (client) care primeşte blocul de adrese de la ultimul ISP din lanţ. Se pot crea local 2^{16} = 65536 combinaţii sau subneturi individuale şi se pot ierarhiza subneturile pe 16 nivele de subnetting.

- E:F:G:H (64 biţi) se numeşte *interface ID* şi este similar cu hostul IPv4. Se observă că numărul de hosturi posibil în fiecare organizaţie este $2^{64} = 1.8 \cdot 10^{19}$ hosturi.

Adrese unicast *link-local:* sînt împărţite tot în cele 3 secţiuni, dar de dimensiuni diferite:

- primii 10 biţi reprezintă *site prefix*; primul cîmp poate lua valori numai între FE80 şi FEBF (se asignează local, nu de către ISP)
- următorii biţi reprezintă *subnet ID* şi sînt toţi 0, întrucît subnetul nu este necesar
- ultimii 64 biţi reprezintă *interface ID;* de obicei este aceeaşi cu a adresei unicast globale.

Prin urmare o adresă de tip *link-local* are forma de genul:

FE80:0:0:0:0:0:AB12:CD34 sau FE80::AB12:CD34

Adrese unicast *site-local*: Primii 10 biţi pot lua valori astfel încît primul cîmp este între FEC0 şi FEFF. Nu vor fi tratate în continuare din motivele arătate mai sus.

Adrese unicast rezervate:

- **de *loopback***: termenul loopback se referă aici în mod specific la o adresă de test, care trimite înapoi către sursă toate pachetele primite; în IPv4 este rezervată adresa 127.0.0.1, iar în IPv6 adresa 0:0:0:0:0:0:0:1 care se mai poate scrie ::1
- **nespecificate**: atunci cînd un host nu-şi cunoaşte (încă) propria adresă, va folosi adresa 0:0:0:0:0:0:0:0 sau pe scurt ::
- **compatibile cu IPv4**: pentru a reprezenta o adresă IPv4 A.B.C.D, de 32 biţi, ca adresă IPv6, se completează cu 0 toţi biţii mai semnificativi (din stînga) pînă la 128, deci în total 96 biţi zero (sau 6 cîmpuri de 16 biţi). Adresa se poate scrie în notaţia specială 0:0:0:0:0:0:A.B.C.D sau prescurtat ::A.B.C.D
- **compatibile cu IPv4 în situaţii de folosire mixtă IPv6/IPv4 (*dual stack*)**: similar cu cazul de mai sus, dar definite ulterior; înaintea completării cu 0 a tuturor biţilor mai semnificativi, se adaugă grupul FFFF. Reprezentarea se poate scrie ::FFFF:A.B.C.D

Tipic, masca de reţea corespunzătoare adreselor IPv6 este /64 (ultimii 64 de biţi sînt de interfaţă, adică de host). Lungimea măştii de reţea se scrie *numai* sub această formă /n. Pentru a specifica prefixe de anumite lungimi se foloseşte n<64; de exemplu, toate adresele unicast globale au prefixul /3 (conform celor 3 biţi 001).

Anumite adrese de host cum ar fi ::1 (loopback) se pot scrie ::1/128, prin similitudine cu adresele de host IPv4 /32.

2.3 Configurarea adreselor IPv6 pe interfeţe

Configurarea se poate face similar ca la IPv4 (manual sau prin DHCP) şi se introduce o metodă nouă, autoconfigurarea pe baza adresei MAC.

- Configurarea manuală se face alocînd manual atît cei 64 biţi de prefix+subnet, cît şi cei 64 biţi de *interface ID*. Se foloseşte comanda `ipv6 address` în loc de `ip address`; masca se specifică in forma /n:

```
Router(config)# int F0
Router(config-if)# ipv6 address 2003:2:3:4:5:6:7:8/64
```

- Configurarea prin DHCP se face folosind o versiune numită DHCPv6; există şi o variantă numită *stateless autoconfiguration* rezervată numai pentru PC-uri şi alte echipamente de tip *end-device* (nu rutere), care foloseşte mesaje ale protocolului ICMPv6.
- Autoconfigurarea porţiunii de interfaţă a adresei pe baza adresei MAC: această metodă de configurare se numeşte EUI-64 (*Extended Universal Identifier*-64). În acest caz, se configurează manual doar cei 64 de biţi ai prefixului. Ultimii 64 biţi se alocă automat pe baza adresei MAC. Cum aceasta are doar 48 biţi, cei 16 biţi rămaşi (practic, biţii din "mijlocul" cîmpului de 64 de biţi) se alocă automat la valoarea FFFE.
  De exemplu, adresa MAC (scrisă ca 6 numere hexazecimale de 8 biţi) 11:22:33:44:55:66 produce *Interface ID* cu valoarea 1122:33FF:FE44:5566.
  Configurarea se face adăugînd cuvîntul-cheie **`eui-64`** după specificarea prefixului:

```
Router(config)# int F0
Router(config-if)# ipv6 address 2003:2:3:4::/64 eui-64
```

Avantajul EUI-64 constă în configurarea simplă (prefixul este obţinut de la ISP şi e acelaşi în toată organizaţia, urmînd ca doar SLA să fie configurat manual). Dezavantajul este că adresa MAC este oarecum aleatoare (şi cam lungă), unii administratori preferînd un control mai strict asupra adreselor alocate.

În momentul configurării adresei IPv6 a unei interfeţe, ruterul alocă automat şi o adresă de tip *link-local*. Implicit, se alocă prefixul FE80:: şi se folosesc aceiaşi 64 biţi pentru porţiunea de host ca în adresa globală.

Dacă se doreşte configurarea manuală a adresei *link-local*, se face la fel ca pentru adresa globală, doar că nu se specifică masca şi se adaugă cuvîntul-cheie **`link-local`**:

```
Router(config)# int F0
Router(config-if)# ipv6 address FE80::1:1:1:1 link-local
```

De menţionat că dacă prefixul nu este între FE80 şi FEBF se va primi un mesaj de eroare.

2.4 Soluţii de interoperare IPv6/IPv4

Trebuie reţinut că adresele IPv4 şi IPv6 fac parte din familii complet diferite şi sînt incompatibile. Pentru a face posibilă interoperarea echipamentelor care folosesc cele 2 tipuri de adrese se folosesc următoarele soluţii:

- *Dual Stacking*: denumirea provine din faptul că protocoalele IPv4/IPv6 formează 2 "stive" (*stack*) de protocoale. Soluţia presupune configurarea dual: pe fiecare echipament se configurează atît adrese IPv4 cît şi IPv6. Astfel, un server web cu *dual stack* va fi accesat de către clienţii IPv4 la adresa sa IPv4 şi de către clienţii IPv6 la adresa sa IPv6. Nu se face nici un fel de translaţie. Această metodă este cea mai simplă şi cea mai recomandată.
- Tunelare: un *tunel* există şi în IPv4 şi reprezintă încapsularea unui pachet într-un alt pachet, cu scopul de a ascunde adresa pachetului iniţial.
  O aplicaţie tipică IPv4 a unui tunel este următoarea: avem 2 filiale ale unei companii, în 2 locaţii diferite; în interiorul companiei se foloseşte o schemă de adresare privată, de exemplu reţelele 192.168.1.0 şi 192.168.2.0 în cele 2 locaţii. Cele 2 reţele sînt conectate la Internet prin cîte un ruter de margine. Pentru ca *fiecare* host din cele 2 reţele să aibă conectivitate cu oricare alt host, se crează un tunel între cele 2 rutere. Un pachet din reţeaua 1 cu destinaţia reţeaua 2 este încapsulat, trimis prin Internet, şi decapsulat de către ruterul de margine al reţelei 2. Prin încapsulare este posibil ca adresele private să fie păstrate, dar să nu fie vizibile în Internet.
  În cazul IPv6, pachetele sînt încapsulate în interiorul unor pachete IPv4, circulă prin Internetul IPv4, şi sînt decapsulate la destinaţie. Prin această tunelare este posibilă conectarea mai multor "insule" IPv6 izolate în Internetul IPv4.
  Un dezavantaj al tunelării în general este scăderea dimensiunii maxime a pachetului: un pachet transportat prin încapsulare într-un alt pachet va avea, implicit, spaţiul pentru încărcătura utilă (*payload*) redus cu dimensiunea headerului pachetului care-l transportă.
- NAT: soluţii de translaţie, similare cu NAT IPv4, există, dar sînt mai complexe decît acesta din urmă. Cisco suportă NAT-PT (NAT-Protocol Translation) începînd cu IOS 12.3(2)T. Datorită complexităţii şi problemelor inerente translaţiilor, aceste soluţii nu sînt recomandate.

2.5 Rutarea IPv6

Rutarea pachetelor IPv6 se face similar cu IPv4: se compară prefixul destinaţie cu prefixele din tabela de rutare disponibilă, folosind funcţia AND, şi se alege rezultatul cu cei mai mulţi "1" (*longest prefix match*).

Protocoalele de rutare IPv4: RIP, OSPF, IS-IS, BGP etc. au fost adaptate (sau sînt în curs de adaptare) pentru IPv6. În continuare se va prezenta RIPv6, numit şi RIPng (*next generation*). Acesta este bazat pe RIP v2 (din IPv4). Pentru configurare se procedează astfel:

- implicit pe ruterele Cisco este activată rutarea IPv4 între interfeţe, dar nu şi cea IPv6. Indiferent dacă se foloseşte RIPng sau alt protocol (sau chiar rutare statică), ea se activează folosind:

```
Router(config)# ipv6 unicast-routing
```

- ca şi OSPF, RIPng cere specificarea unui nume (sau număr) pentru un proces de rutare; astfel, se pot porni mai multe procese de rutare separate.
- o diferenţă între RIP v1/v2 şi RIPng este că nu se anunţă reţelele folosind cuvîntul-cheie *network*, ci se activează cu cuvîntul-cheie *enable* interfeţele pe care va rula protocolul. De fapt, nu este o diferenţă de fond, pentru că şi în RIP/OSPF, am văzut că cuvîntul-cheie *network* are semnificaţia de a preciza interfeţele prin care se trimit *routing updates*, spre deosebire de BGP. De exemplu, dacă un ruter are intefeţele E0 şi S0 care vor participa la RIPng, şi procesul de rutare se numeşte cisco1:

```
Router(config)# ipv6 router rip cisco1
Router(config)# int E0
Router(config-if)# ipv6 address ....
Router(config-if)# ipv6 rip cisco1 enable
Router(config-if)# int S0
Router(config-if)# ipv6 address ....
Router(config-if)# ipv6 rip cisco1 enable
```

Observaţie: pentru a examina funcţionarea IPv6, se folosesc comenzi *show* similare cu cele pentru IPv4, la care în general se înlocuieşte *ip* cu *ipv6*. De exemplu:

```
show ipv6 interface
show ipv6 interface brief
show ipv6 protocols
show ipv6 rip
show ipv6 route
```

2.6 Access-lists IPv6

ACL în IPv6 sînt o extensie a ACL IPv4; ca particularităţi, menţionăm:

1) se pot defini doar cu nume; ACL IPv4 se pot defini atît cu nume, cît şi cu număr (implicit).
Definirea se face astfel:

```
ipv6 access-list nume
regula 1...
regula 2...
exit
```

iar fiecare din reguli are sintaxa de bază (care permite opţiuni suplimentare, vezi help):

```
permit | deny protocol adr_sursa [port] adr_destinatie [port]
[numar-secventa]
```

unde:
protocolul poate fi ipv6 (care le cuprinde pe toate) sau tcp, udp, icmp etc;
adresele sursa/destinaţie pot fi de una din următoarele forme:

- any (similar cu IPv4)
- host N:N:N:N:N:N:N:N (similar cu IPv4)
- *adresa/lungime_prefix,* de exemplu 2001:1::/64 – diferenţa faţă de IPv4 e că se foloseşte acelaşi tip de prefix ca la definirea interfeţelor, rutelor etc, şi nu prefixul inversat ca în *wildcard mask*

Ca şi în IPv4, se poate prefixa cu *no* orice comandă;

2) la sfîrşitul unui ACL există implicit următoarele reguli:

```
permit icmp any any nd-na
permit icmp any any nd-ns
deny ipv6 any any
```

faţă de IPv4, primele 2 reguli au rolul de a permite pachetele de *network discovery*.

Vizualizarea ACL se face cu:

```
sh ipv6 access-list nume
```

2.7 Suportul Cisco IOS pentru IPv6

Imaginile IOS pentru ruterele Cisco au nume de forma *cnnnn-xxx-...-xxx-mmm-pp.bin*, unde:

- *nnnn* este numele platformei, de exemplu 1700, 2600, 3600 etc
- grupurile notate *xxx* pot fi în număr variabil; sînt foarte multe variante şi reprezintă *feature set*-ul, adică setul de comenzi, protocoale, facilităţi şi funcţii suportate. Pentru a afla exact ce *features* sînt într-un *feature set* se foloseşte Cisco Feature Navigator [2].
- *mmm* este numărul versiunii; 123 înseamnă versiunea 12.3
- *pp* este numărul *release*-ului, care este o sub-versiune în cadrul versiunii. De la un release la altul se corectează bug-uri şi se introduc unele funcţii suplimentare.
- mai poate exista o literă ca sufix care specifică categoria de IOS, cum ar fi T pentru *Technology*, E pentru *Entreprise*, etc. Fără [2] nu se poate determina exact ce facilităţi aduce această literă.

Cisco a introdus suportul pentru IPv6 în versiunea de IOS 12.2(1)T. În plus, IPv6 trebuie să fie parte a *feature set*-ului imaginii respective; aceasta înseamnă că o imagine numită (de exemplu) c1700-xxx-xxx-121-10.bin *sigur* nu suportă IPv6, întrucît este vorba de versiunea 12.1(10), în timp ce o imagine numită c1700-xxx-xxx-123-15.bin *ar putea* suporta IPv6, fiind vorba de versiunea 12.3(15), dar trebuie examinat *feature set*-ul pentru a determina în mod sigur acest lucru.

Partea 3. Desfăşurare; configurare; exerciţii

OBSERVAŢIE 1: reamintim că pentru a întrerupe o comandă *ping, traceroute* etc care nu funcţionează se va folosi secvenţa **CTRL-SHIFT-6**
OBSERVAŢIE 2: pentru a forţa ruterul să nu ne mai scoată automat din modul "enabled" după un timp de inactivitate, se dau comenzile:

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

Faza 1: Asignarea adreselor prin DHCP

În mod normal, DHCP este folosit aproape exclusiv pentru a asigna adrese în mod dinamic PC-urilor, atunci cînd acestea sînt numeroase (pe interfaţa PC se configurează opţiunea "*assign IP address automatically*"). Totuşi, este posibil ca şi un ruter să primească adresa IP prin DHCP.

Realizaţi topologia din figura 2. Se urmăreşte ca R1 să primească adresa IP şi ceilalţi parametri de configurare de la R0.

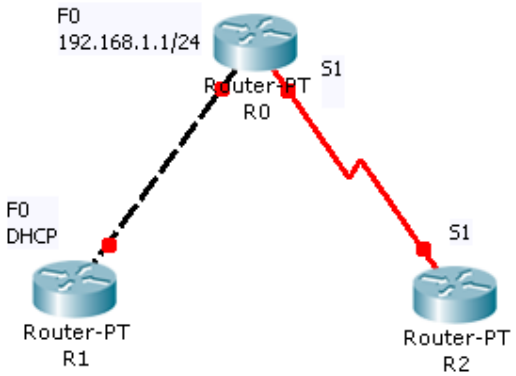

Fig. 2 Topologia pentru DHCP

Q1. Desenaţi topologia, precizînd interfeţele folosite. Este recomandabil să folosiţi interfeţe cu nume asemănătoare cu fig. 2 pentru a urmări uşor lucrarea.

a) se configurează interfaţa F0 a R0 cu adresa 192.168.1.1/24

b) pe R0 se configurează un *pool* (domeniu) de adrese pentru clienţi, cu numele "**test**", conţinînd toate adresesele din reţeaua 192.168.1.0/24

```
R0(config)# ip dhcp pool test
R0(dhcp-config)# network 192.168.1.0 255.255.255.0
R0(dhcp-config)# default-router 192.168.1.1
R0(dhcp-config)# dns-server 192.168.1.2
R0(dhcp-config)# exit
```

(Observaţie: ultimele 2 comenzi sînt utile unui PC, nu unui ruter).

Din acest domeniu se exclud primele 10 adrese:

```
R0(config)# ip dhcp excluded-address 192.168.1.1 192.168.1.10
```

Pentru a observa procesul de alocare se porneste *debugging*-ul:

```
R0# debug ip dhcp server events
R0# debug ip dhcp server packets
```

c) se configurează R1 drept client dhcp:

```
R1(config)# int F0
R1(config-if)# no shutdown
R1(config-if)# ip address dhcp
CTRL-Z
```

Pe R0 se vizualizează mesajele de debug.

Q2. Cum se numesc cele 4 pachetele DHCP schimbate (cele care încep cu DHCP) ?Ce adresă IP a primit R1 ? de ce ?
Q3. Este posibil să se suprapună adresa clientului cu a serverului ? de ce sau de ce nu ?

Verificaţi că e posibil să daţi ping de pe R1 pe R0 (interfaţa Fast Ethernet).

Q4. Verificaţi tabela de rutare de pe R1. Există vreo rută *default*? Cine este *Gateway of last resort?*

Pe R0 se examinează IP-urile alocate clienţilor:

```
R0# show ip dhcp binding
```

d) se opreşte *debugging*-ul folosind `undebug all`

Faza 2: Configurarea NAT

Se configurează ruterele conform figurii de mai jos. Pe R1 se va şterge automat adresa DHCP atunci cînd se va configura adresa statică 192.168.1.2 (deci nu trebuie s-o ştergeţi manual). Nu se instalează rute statice sau vreun protocol de rutare.

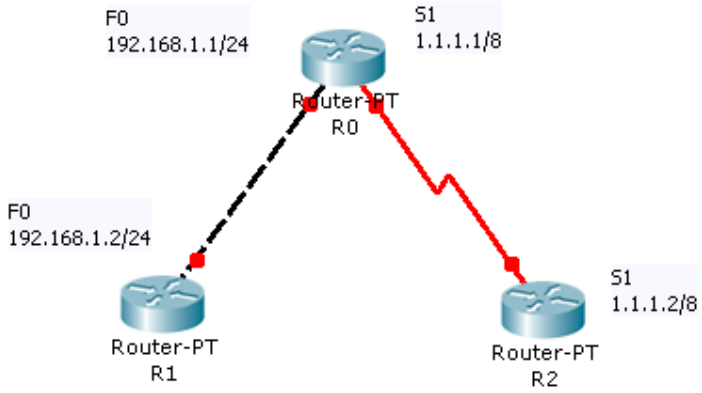

Fig. 3 Topologia pentru NAT

Considerăm că reţeaua R0-R1 este reţeaua internă, care foloseşte adrese private (RFC1918: 192.168.x.x), iar R0/S1 şi R2 fac parte din "Internet" şi au adrese reale, rutabile.

Q5. Verificaţi dacă este posibil ping între R1 şi R2. Daţi cel puţin 2 motive pentru care nu este posibil, ţinînd seama şi că R2 este în "Internet".

a) R0 va fi configurat să translateze folosind NAT adresele "locale" din 192.168.1.0 într-una din adresele "globale" disponibile în reţeaua dinspre "Internet". Decidem să folosim doar 2 adrese pt NAT, într-un *pool* cu numele "test":

```
R0(config)# ip nat pool test 1.1.1.10 1.1.1.11 netmask 255.0.0.0
```

b) Se configurează, folosind un access-list, lista adreselor private care vor fi translatate:

```
R0(config)# access-list 1 permit 192.168.1.0 0.0.0.255
```

c) se va face translaţia propriu-zisă: toate adresele din lista 1 se translateaza în *pool*-ul "test"

```
R0(config)# ip nat inside source list 1 pool test
```

d) pe R0 se configurează (ca în 1.4 după pct. 3), pentru fiecare interfaţă, de ce "parte" este, respectiv *inside* (spaţiul privat) sau *outside* (spaţiul public): S1 este *outside* şi F0 este *inside*.

e) pentru ca R1 să aibă o rută către "Internet" se pune o rută *default* (această rută fusese dată şi de DHCP dar acum a dispărut şi trebuie setată manual)

```
R1(config)# ip route 0.0.0.0 0.0.0.0 192.168.1.1
```

f) se testează funcţionarea NAT-ului. Se porneşte *debugging*-ul pe R0:

```
R0# debug ip nat
```

Se dă ping de pe R1 către R2. Apoi se verifică translaţiile folosind:

```
R0# show ip nat translations
```

Q6. Ce se observă după ce se dă această comandă ? ce adresă sursă au pachetele de pe R1?

f) se verifică dacă se poate da ping din Internet către hosturile private. Se va da pe R2 un ping către adresa destinaţie 192.168.1.1 sau 192.168.1.2

Q7. Are succes acest ping ? de ce sau de ce nu ?

Se opreşte *debugging*-ul pe R0.

Faza 3: Port forwarding

Dorim să permitem accesul dinspre Internet înspre un server (care oferă un serviciu oarecare: web, telnet, etc) din interiorul domeniului cu adrese private, ceea ce nu se poate în mod direct. Vom folosi adresa publică, de Internet, a lui R0 (1.1.1.1); accesul Telnet (port 23) de pe această adresă va fi redirectat către serverul de Telnet de pe R1 (192.168.1.2).

b) pe R0 se configureaza port forwarding:

```
R0(config)# ip nat inside source static tcp 192.168.1.2 23 1.1.1.1
23
```

cuvîntul *static* în loc de *list* semnifică faptul că este un caz particular de translaţie (statică - ambele adrese sînt precizate explicit); în acest caz, această comanda este suficientă; nu trebuie definit nici un *pool* de adrese externe, nici un access-list pentru adresele interne.

c) se permite accesul Telnet pe R1

```
R1(config)# line vty 0 4
R1(config-line)# password cisco
```

d) se face Telnet de pe R2 pe adresa 1.1.1.1

Q8. La ce destinaţie aţi ajuns ? (verificaţi numele ruterului)
Q9. Ce translaţii apar pe R0 cu comanda `sh ip nat translations` ?

Ştergeţi cu *no* comenzile de translaţie NAT (statică şi dinamică; nu e nevoie să ştergeţi *pool*-ul, ACL-ul, definiţiile *inside* şi *outside*). Dacă nu vă permite, daţi înainte comanda `clear ip nat translations *` şi încercaţi din nou (înseamnă că translaţiile sînt în uz).

Faza 4. Asignarea adreselor IPv6

a) Asignăm pe R1 adrese IPv6 de tip EUI-64. Mai întîi, se examinează adresa MAC a interfeţei Ethernet, folosind `show int F0`. Adresa MAC este numită de către Cisco BIA (*Burned-In Address*), cu sensul că nu se poate modifica.

Q10. Care este adresa MAC a lui F0 ?

apoi, se configurează adresa IPv6 2001::/64 de tipul EUI-64. Se verifică adresele interfeţei folosind `show ipv6 int F0`. Observaţi similitudinile, dar şi diferenţele dintre ultimii 64 biţi şi adresa MAC.

Q11. Care sînt adresele IPv6 ? (globală şi link-local; se ignoră adresele "*joined group*" care reprezintă 3 grupuri multicast care se configurează automat)
Q12. Care sînt cele 4 cifre hexazecimale care au apărut la mijlocul porţiunii de *interface ID* (ultimii 64 biţi), între primii 3 şi ultimii 3 octeţi ai adresei MAC ?

În afară de cele 4 cifre hexazecimale în plus, mai există o diferenţă.

Q13. Care este această diferenţă? scrieţi valoarea "aşteptată" şi valoarea afişată de ruter, pentru grupul de 4 cifre hexazecimale unde apare diferenţa.

Pentru a "rezolva" diferenţa se procedează astfel:
- se ia primul grup de 4 cifre hexazecimale din cadrul celor 64 biţi de interfaţă
- se transformă în binar
- se inversează al 7-lea bit (numit bitul *universal/local*)
- se transformă din nou în hexazecimal;

Q14. Ce valoare aţi obţinut ?

Similar, se configurează pe R2, interfaţa serială S1, adresa 2002::/64 de tipul EUI-64.

Q15. Există o adresă MAC a lui S1, ca în cazul F0 ?
Q16. Care sînt adresele IPv6 (globală şi *link-local*) ?
Q17. De unde credeţi că ia ruterul adresa MAC ? verificaţi !

b) se observă că adresele EUI-64 sînt greu de memorat. Vom asigna adresele manual, conform figurii 4 (pe care sînt figurate doar adresele IPv6; adresele IPv4 *se lasă aşa cum sînt*).

Q18. Cîte hosturi se pot forma în subnetul /96 de pe reţeaua R0-R1 ? cît ar fi fost subnetul maxim necesar pentru a nu "irosi" hosturi ?

Atenţie! O diferenţă între adresele IPv4 şi IPv6 este că, implicit, o interfaţă poate avea mai multe adrese IPv6. Prin urmare, configurarea unei noi adrese *nu o şterge automat* pe cea veche, ca în IPv4 (şi în IPv4 se pot pune adrese multiple pe interfeţe, dar nu implicit).

Se scot, folosind prefixul *no,* vechile adrese cu EUI-64 de pe R1/F0 şi R2/S1. Se configurează noile adrese din fig. 4.

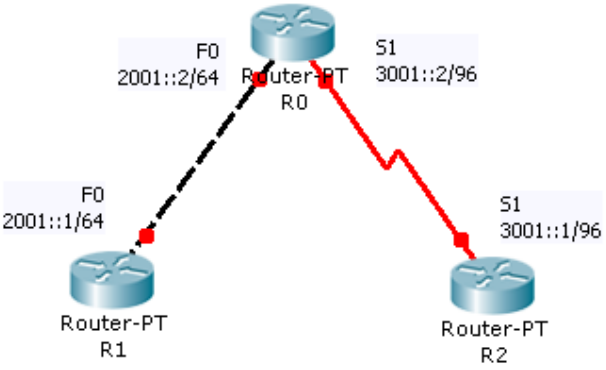

Fig. 4 Topologia pentru adresarea IPv6.

Pe R1 înlocuiţi şi adresa link-local folosind cuvîntul-cheie `link-local` (şi fără a preciza masca):

```
R1(config)# int F0
R1(config-if)# ipv6 addr 2001::1 link-local
```

Q19. Ce mesaj de eroare primiţi?
Q20. Cum trebuie modificată această adresă ? verificaţi.

Verificaţi cu ping de pe R1 propria interfaţă (adresa globală şi cea link-local).
Observaţi că dacă încercaţi ping la interfaţa *link-local* primiţi întrebarea "Output Interface:". Răspunsul trebuie să conţină un spaţiu, de exemplu "F 0", nu "F0".

Q21. Verificaţi cu ping conectivitatea între R0-R1 şi R0-R2. Avem conectivitate între R1 şi R2? De ce ?

Faza 5. Rutarea IPv6

a) Se activează mai întîi rutarea IPv6 pe toate ruterele cu `ipv6 unicast-routing`:

```
Router(config)# ipv6 unicast-routing.
```

Vom defini o rută statică *default* pe R1 pentru a putea da ping către S1 a lui R0:

```
R1(config)# ipv6 route ::/0 2001::2
```

Q22. Explicaţi semnificaţia lui ::/0 în acest context, comparînd cu modul de specificare al rutei statice default în IPv4.

Q23. Verificaţi cu `sh ipv6 route` tabela de rutare. Apare un *gateway of last resort* ?.

Verificaţi cu ping de pe R1 destinaţia 3001::2. Apoi verificaţi destinaţia 3001::1

Q24. A mers sau nu ping către 3001::1 ? explicaţi rezultatul.

b) pentru a permite conectivitatea totală vom configura protocolul de rutare RIPv6.
Pe fiecare ruter:

- porniţi protocolul, de exemplu cu numele *c6*, folosind comanda

```
Router(config)# ipv6 router rip c6
```

- pe fiecare interfaţă, configuraţi-o să participe la rip *c6* folosind *enable*:

```
Router(config-if)#ipv6 rip c6 enable
```

Verificaţi tabela de rutare folosind `sh ipv6 route` şi procesul rip folosind `sh ipv6 rip`

Q25. Care e distanţa administrativă a lui RIPv6?

În momentul acesta se observă că pe fiecare ruter există atît adrese IPv4 (rămase ca în figura 3) cît şi IPv6. Pe lîngă RIPv6, *se va porni şi RIP v2* (pentru IPv4) pe fiecare ruter. Se face abstracţie, în acest moment, că o parte din adresele IPv4 sînt private şi altele nu (nu mai folosim NAT).

Verificaţi cu ping că avem conectivitate totală şi pe IPv4 (R0-R1-R2). Dacă nu funcţionează, verificaţi dacă aţi şters translaţiile NAT la sfîrşitul fazei 3.

Q26. Cum se numeşte existenţa simultană a adreselor şi protocoalelor de rutare v4 şi v6 ?

Faza 6. IPv6 ACL

a) Configurăm pe R0 un access-list cu numele *interzis1* care să interzică traficul ping de la R1 la R2 (orice interfeţe). Tot restul traficului ipv6 va fi permis.

Q27. Scrieţi comanda folosită

Aplicăm lista pe interfaţa F0/0, sensul *in*; cuvîntul-cheie diferă faţă de IPv4, este *traffic-filter* în loc de *access-group*:

```
int f0/0
ipv6 traffic-filter interzis1 in
```

b) configurăm pe R0 un access-list cu numele *interzis2* care să interzică traficul telnet de pe orice host pe interfaţa 3001::2 a lui R0; tot restul traficului ipv6 va fi permis.

Q28. Scrieţi comanda folosită

aplicăm lista pe telnet, cu cuvîntul-cheie *access-class* (similar cu IPv4):

```
line vty 0 15
```

```
ipv6 access-class interzis2 in
```

Faza 7. Tunelare IPv6 în IPv4

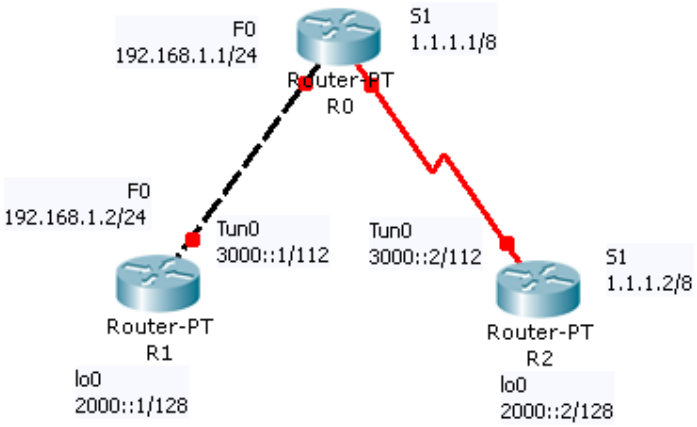

Fig. 5 Topologia pentru tunel IPv6-IPv4

Vom ilustra tunelarea pachetelor IPv6 prin „Internetul IPv4". În acest scop, vom avea doar adrese IPv4 pe interfeţele seriale şi Ethernet ale R0,R1,R2. Vom avea 2 „insulele" IPv6 izolate, care se simulează prin cîte o interfaţă loopback pe R1 şi R2. Comunicarea între insule se face printr-un tunel; capetele tunelului sînt R1/F0 şi R2/S1. Se observă că R0 (ruterul de la mijloc) nu are configurat nimic de IPv6 – el este „transparent" pentru tunel.

Etape:

a) se elimină toate adresele IPv6 de pe rutere şi se opreşte RIPv6 – revenim la „Internetul IPv4"; adresele sînt cele rămase din figura 3, dar fără NAT.

b) se adaugă cele 2 interfeţe lo0 cu adresele IPv6 2000::1/128, respectiv 2000::2/128

Q29. Ce fel de mască este /128 ?

c) se porneşte pe R1, R2 un proces de rutare RIPv6 cu numele *c64*, la care vor participa momentan doar interfeţele loopback

```
R1(config)# ipv6 router rip c64
R1(config)# int lo0
R1(config-if)# ipv6 rip c64 enable
```

d) se crează cele 2 capete ale tunelului; se activează participarea lor la RIPv6:

```
R1(config)# interface Tunnel0
R1(config-if)# ipv6 address 3000::1/112
R1(config-if)# ipv6 rip c64 enable
R1(config-if)# tunnel source F0
R1(config-if)# tunnel destination 1.1.1.2
R1(config-if)# tunnel mode ipv6ip
```

Se procedează similar pentru R2.

Q30. Cine este sursa, respectiv destinaţia tunelului de pe R2?

Observaţie: folosind comanda incompletă (de tip help) `tunnel mode ?` putem vedea ce alte tipuri de tunele sînt suportate.

În acest moment, folosind `sh ipv6 route`, ar trebui să vedem reţelele RIP de la celălalt capăt al tunelului (nu şi lumina ! :-)

Q31. Ce reţele, aparţinînd lui R2, se văd pe R1 ?
Q32. Verificaţi cu ping de pe R1 dacă răspund adresele 2000::2 şi 3000::2 de pe R2, şi viceversa.

Bibliografie

[1] Cisco IOS Network Address Translation Overview (document online Cisco)
[2] Cisco Feature Navigator, *http://tools.cisco.com/ITDIT/CFN/jsp/index.jsp*
[3] Cisco CCNA Exploration 4.0, modulul 4, cap. 7
[4] RFC3587, RFC2574 *IPv6 global unicast address format*
[5] RFC3879, *Deprecating Site-local addresses*
[6] Tunneling IPv6 through an IPv4 network (document online Cisco)
[7] Map of the Internet, 2006 (webcomic): http://xkcd.com/195