Laborator 6. Adresare IP și rutare în Linux

Cunoștințe și abilități ce vor fi dobândite

- Configurare de adrese IP în Linux
- Configurarea rutării în Linux
- Depanarea problemelor de configurare a rețelei în Linux

Cheat sheet

- Cheat Sheet

Pregătire infrastructură de laborator

- Avem nevoie de o mașină virtuală a laboratorului. Vă rugăm urmăriți pagina aceasta pentru instrucțiuni, apoi reveniți.
- **Atenție:** puteți accesa mașina virtuală (din spațiul `10.9.0.0/16`) doar prin intermediul serverului fep!
- **Sfat util pentru toate laboratoarele:** pentru a vă autentifica pe `fep.grid.pub.ro` fără să vă ceară autentificare web și two-factor, se recomandă generatul și instalatul unei perechi de chei publice/private după pașii de mai jos (pe un sistem Linux):

```
# pe PC/laptop, dați o listare în folderul .ssh:
ls -l ~/.ssh/
# DACĂ AVEȚI DEJA VREUN id_rsa existent, NU mai este nevoie de dat ssh-keygen!
# altfel, generați o cheie, recomandăm să dați ENTER la tot ce vă întreabă:
ssh-keygen # ENTER pentru a folosi setările implicite la toate câmpurile
# ne-a generat fișierele ~/.ssh/id_rsa și ~/.ssh/id_rsa.pub
cat ~/.ssh/id_rsa.pub
# copiem tot (inclusiv id-rsa ... până la final) în clipboard
# ne autentificăm la fep, clasic (încă nu merge fără parolă că n-am terminat configurarea):
ssh username@fep.grid.pub.ro
# odată conectați pe fep:
mkdir -p ~/.ssh
vim ~/.ssh/authorized_keys  # sau nano, dacă nu vă place vim :(
# și dați paste la cheia voastră publică aici (dacă sunteți prin vim, nu uitați să intrați în INSERT)
# ieșiți și apoi vă reconectați la fep, nu ar trebui să vă mai ceară parolă :D
ssh username@fep.grid.pub.ro  # doamne-ajută!
```

Pentru utilizatorii de Windows, ar trebui să existe utilitarul ssh-keygen în consola clasică sau PowerShell. Dacă folosiți Putty, din păcate, pașii sunt mult mai laborioși (folosește alt format pentru chei publice și va trebui să faceți conversia), deci nu vom lista pașii necesari aici (hint: use Google sau folosiți ssh-ul nativ de pe Windows 10, e mai OK).

- Pentru a pregăti configurația de laborator, pe mașina virtuală (stația `host`) folosiți comenzile următoare din contul utilizatorului `root` de pe stația `host` (puteți da copy/paste la comenzi în terminal):

```
# ATENȚIE: update_lab nu funcționează de pe root, folosiți student inițial
student@host:~# update_lab --force
student@host:~# start_lab ip
```

- Deschideți trei noi tab-uri în terminal (folosiți combinația de taste `Ctrl+Shift+t`), și conectați-vă, din nou, la mașina virtuală folosind comanda `ssh` de mai sus.
  - De pe cele trei noi tab-uri, conectați-vă la cele trei containere (`red`, `green` și `blue`).
  - Pentru o conectare mai ușoară puteți folosi aliasul `go` (ex. `go red`)

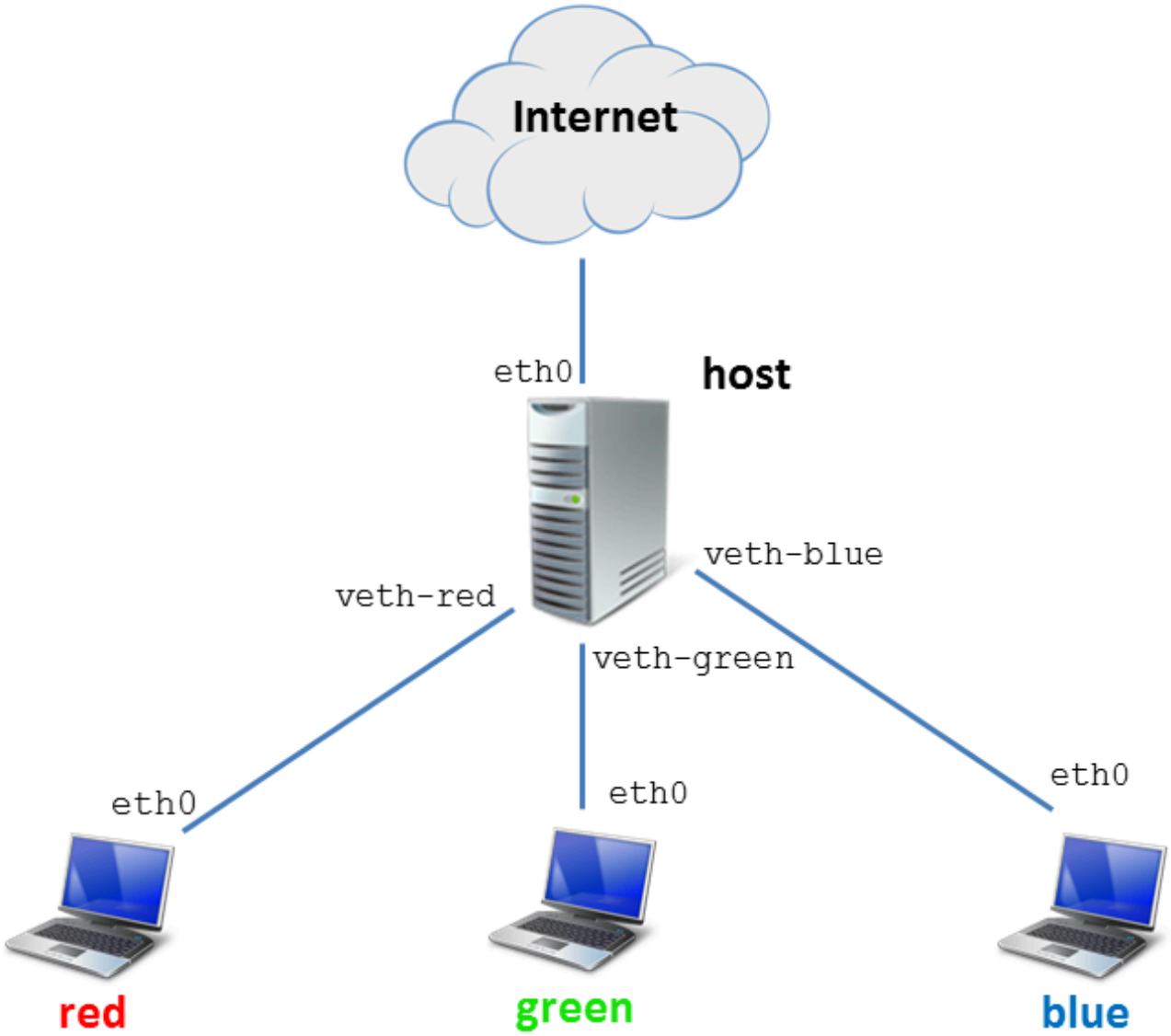
Pentru a vedea cum accesați stațiile `red`, `green` și `blue` (containere Docker configurate peste mașina virtuală VMware - stația `host`) urmăriți indicațiile din pagina cu instrucțiuni de utilizare a mașinii virtuale.

Conturile de acces la mașina virtuală (stația `host`) sunt (`username:parola`):

- `root:student`
- `student:student`

În mod implicit folosiți contul `root` pentru conectare pe toate stațiile. Aveți nevoie de drepturi privilegiate pentru configurare. Folosiți contul `student` doar unde vi se cere explicit.

Topologie



Navigare

Laboratorul 6

- 01. [10p] Configurare și ștergere adrese IP

- 02. [20p] Configurare adrese IP
- 03. [10p] Adresare IP și rutare
- 04. [20p] Configurare conectivitate completă
- 05. [10p] Tabela ARP
- 06. [10p] Depanare problemă de configurare adresă IP
- 07. [10p] Depanare problemă de conectivitate
- 08. [10p] IPv6
- 09. [BONUS - 10p] Configurare persistentă

Exerciții

În cadrul exercițiilor din laboratoarele de Linux vom folosi topologia de mai sus.

01. [10p] Configurare și ștergere adrese IP

Dacă nu ați următit pașii de mai sus, **rulați acum**:

```
# ATENȚIE: update_lab nu funcționează de pe root, folosiți student inițial
student@host:~# update_lab --force
student@host:~# start_lab ip
```

Dorim, pentru început, să asigurăm conectivitate între stațiile `host` și `red`. În acest tutorial vom folosi suita `iproute` de pe Linux pentru a realiza configurările frecvente de nivel 3 (adresare IP).

Vom configura câte o adresă IP din clasa `192.168.0.0/24` pe interfețele de legătură dintre stația `host` și stația `red`. Adică între `host(veth-red)` (interfața `veth-red` de pe stația `host`) și `red(red-eth0)` (interfața `red-eth0` de pe stația `red`).

Pe interfața `veth-red` de pe stația `host` vom configura adresa IP `192.168.0.1` cu masca `255.255.255.0` (`/24` în forma prefixată):

```
root@host:~# ip address add 192.168.0.1/24 dev veth-red
```

Observați că suita `iproute2` (adică utilitarul `ip`) folosește masca în format prefixat: `/24`.

Imediat după o configurare de rețea rulați o comandă pentru validarea configurării. În cazul nostru este comanda de afișare a configurării de nivel 3 (Rețea), adică a adresei IP:

```
root@host:~# ip address show dev veth-red
47: veth-red: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 4e:1b:b8:d9:14:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.1/24 scope global veth-red
```

Pe interfața `red-eth0` de pe stația `red` vom configura adresa IP `192.168.0.2` cu masca `255.255.255.0` (`/24` în forma prefixată):

```
root@host:~# go red
[...]
root@red:~# ip address add 192.168.0.2/24 dev red-eth0
root@red:~# ip address show dev red-eth0
46: red-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 00:16:3e:8e:84:21 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.2/24 scope global eth0
    inet6 fe80::216:3eff:fe8e:8421/64 scope link
       valid_lft forever preferred_lft forever
```

La fel, după o configurare de rețea, am rulat comanda de validare, în cazul acesta `ip address`.

Pentru a vă putea întoarce la consola stației `host`, puteți da `exit` (sau deschideți / folosiți alt terminal).

Pentru a testa conectivitatea între stațiile `host` și `red` folosim comanda `ping`:

```
root@host:~# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
^C
--- 192.168.0.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1007ms
```

După câteva secunde opriți comanda `ping` folosind combinația de taste `Ctrl+c`.

Observați că nu există conectivitatea între cele două stații: pachetele sunt pierdute în întregime (`100% packet loss`). Motivul este că nu am activat interfețele, ci doar am realizat configurații de nivel 3.

Urmăriți configurația de nivel 2 a interfețelor folosind comanda `ip link`:

```
root@host:~# ip link show dev veth-red
10: veth-red: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 3e:03:f0:76:76:ab brd ff:ff:ff:ff:ff:ff
```

Observați că interfața nu este activă la nivelul 2 (Legătură de date). Pentru a activa interfața folosiți comanda:

```
root@host:~# ip link set dev veth-red up
```

Urmăriți din nou configurația de nivel 2 (Legătură de date) a interfeței `veth-red` și observați că acum este parțial `UP` (apare și `UP` și `DOWN` în output-ul comenzii):

```
root@host:~# ip link show dev veth-red
10: veth-red: <NO_CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 3e:03:f0:76:76:ab brd ff:ff:ff:ff:ff:ff
```

Testați din nou conectivitatea folosiți comanda `ping`. În continuare nu există conectivitate. Acest lucru și faptul că apărea și `DOWN` în output-ul comenzii anterioare se datorează faptului că nu am activat interfața `red-eth0` de pe stația `red`. Interfața `red-eth0` de pe stația `red` este cea conectată la interfața `veth-red` de pe stația `host`; ambele trebuie să fie activate pentru a avea o conexiune activă.

Pe stația `red` verificați configuratia de nivel 2 a interfației `red-eth0` de pe `red`. Observați că este `DOWN` și activați interfața dacă este cazul, folosind comanda

```
root@red:~# ip link set dev red-eth0 up
```

Verificați că acum interfața este activă folosind comanda

```
root@red:~# ip link show dev red-eth0
```

Folosiți comanda `ping` ca să retestați conectivitatea între stațiile `host` și `red`.

Dorim să revenim la configurația inițială. Pentru acesta rulați o comandă de forma

```
# ip address flush dev INTERFACE
```

unde `INTERFACE` este interfața interfața `veth-red` pe stația `host`, respectiv `red-eth0` pe stația `red`. Asigurați-vă că nu mai este configurată nicio adresă IP pe interfețe folosind o comandă de forma

```
# ip address show dev INTERFACE
```

unde `INTERFACE` este interfața interfața `veth-red` pe stația `host`, respectiv `red-eth0` pe stația `red`.

02. [20p] Configurare adrese IP

Dorim să avem conectivitate între stația `red` și stația `host`, respectiv între stația `green` și stația `host`. Pentru aceasta, vom configura adrese IP pe fiecare.

Configurați câte o adresă IP din clasa 10.10.10.0/24 pe legătura dintre stația `red` și stația `host` (adică legătura `red(red-eth0)` ↔ `host(veth-red)`) și testați conectivitatea.

Configurați câte o adresă IP din clasa 10.10.20.0/24 pe legătura dintre stația `green` și stația `host` (adică legătura `green(green-eth0) ↔ host(veth-green)`) și testați conectivitatea.

Aveți în vedere să verificați nivelul *Legătură de date* folosind comanda `ip link` și să activați, la nevoie, interfețele.

03. [10p] Adresare IP și rutare

Dorim să realizăm conectivitate și între stațiile `red` și `green`. Întrucât cele două stații sunt în rețele locale diferite, va trebui să configurăm stația `host` ca *default gateway* pe fiecare stație.

Pentru a adăuga *default gateway* pe stația `red` folosiți comenzile:

```
root@host:~# go red
[...]
root@red:~# ip route add default via 10.10.10.1
```

După configurare (adăugarea rutei), validăm configurația cu o comandă specifică. În acest caz urmărim tabela de rutare folosind comanda:

```
root@red:~# ip route show
default via 10.10.10.1 dev red-eth0
10.10.10.0/24 dev red-eth0  proto kernel  scope link  src 10.10.10.2
```

Adresa IP `10.10.10.1` reprezintă adresa IP a interfeței `veth-red` de pe stația `host`.

Intrați pe stația `green` și executați comenzile similar.

Testați conectivitatea, folosind `ping` între stațiile `green` și `red`. Observați că nu funcționează. Motivul pentru care nu există conectivitate este reprezentat de faptul că stația `host` nu are activată rutarea (nu trimite pachetele ce vin de pe o interfață pe altă interfață). Pentru a activa rutarea pe stația `host`, rulați comanda:

```
root@host:~# sysctl -w net.ipv4.ip_forward=1
```

Pentru a valida configurarea de activare a rutării, rulăm comanda:

```
root@host:~# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

Testați din nou conectivitatea între `red` și `green` și observați că funcționează.

Porniți comanda `ping` de pe stația `red` către stația `green`. Deschideți un nou terminal și executați pe stația `host` comanda:

```
root@host:~# tcpdump -n -i veth-red
listening on veth-red, link-type EN10MB (Ethernet), capture size 65535 bytes
18:46:48.783576 IP red.local > 10.10.20.2: ICMP echo request, id 434, seq 163, length 64
18:46:48.783622 IP 10.10.20.2 > red.local: ICMP echo reply, id 434, seq 163, length 64
```

Observați pachetele de tip `ICMP echo request/reply` care trec prin stația `host` (sau altfel zis stația `host` le rutează).

04. [20p] Configurare conectivitate completă

Dorim să asigurăm conectivitate completă între toate stațiile din topologie. Trebuie configurată corespunzător stația `blue`.

Configurați adrese IP din clasa 10.10.30.0/24 pe legătura dintre stația `host` și stația `blue` (adică între `host(veth-blue)` și `blue(blue-eth0)`).

Țineți cont să verificați legătura de nivel 2 folosind comanda `ip link`.

Testați conectivitatea între stația `host` și stația `blue`.

Pe stația `blue` configurați ca *default gateway* stația `host`, pentru a permite conectivitatea la celelalte stații.

Pe stația `blue` folosiți ca *default gateway* adresa IP de pe interfața `veth-blue` a stației `host`.

Testați conectivitatea între oricare două stații.

05. [10p] Tabela ARP

ARP (*Address Resolution Protocol*) este protocol care face intern fiecărui sistem de operare asocierea între adresele IP și adresele MAC ale stațiilor cu care comunică. De multe ori stațiile cunosc adresele IP ale vecinilor dar nu știu adresele MAC; protocolul ARP completează o tabelă ARP locală sistemului cu intrările necesare. Protocolul ARP este rulat implicit de sistemul de operare atunci când se comunică cu o stație a cărei adresă MAC nu este cunoscută.

Ne propunem să urmărim tabela ARP a unui sistem Linux.

Pe stația `host` urmăriți tabela ARP folosind comanda:

```
root@host:~# ip neighbor show
[...]
```

Este posibil ca tabela să fie goală (neexistând comunicație recentă) sau să aibă unele intrări (cele mai recente comunicații) sau intrările să fie marcate `STALE` (intrări nesigure).

Pentru a popula tabela ARP inițiați comunicație cu celelalte stații folosind comanda `ping`:

```
root@host:~# ping -c 1 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_req=1 ttl=64 time=0.033 ms
[...]
root@host:~# ping -c 1 10.10.20.2
PING 10.10.20.2 (10.10.20.2) 56(84) bytes of data.
64 bytes from 10.10.20.2: icmp_req=1 ttl=64 time=0.036 ms
[...]
root@host:~# ping -c 1 10.10.30.2
PING 10.10.30.2 (10.10.30.2) 56(84) bytes of data.
64 bytes from 10.10.30.2: icmp_req=1 ttl=64 time=0.080 ms
[...]
```

Urmăriți din nou tabela ARP:

```
root@host:~# ip neighbor show
10.10.10.2 dev veth-red lladdr 00:16:3e:8e:84:21 REACHABLE
10.10.20.2 dev veth-green lladdr 00:16:3e:d1:b2:95 REACHABLE
10.10.30.2 dev veth-blue lladdr 00:16:3e:32:0f:ae REACHABLE
10.8.0.1 dev eth0 lladdr 0a:00:27:00:00:00 REACHABLE
```

Observați că fiecare stație (`red`, `green` și `blue`) are câte o intrare aferentă în tabela ARP marcată cu `REACHABLE` (intrare validă). Intrarea suplimentară este comunicarea mașinii virtuale (stației `host`) cu sistemul `fep.grid.pub.ro`.

Realizați pașii de mai sus pentru fiecare dintre stațiile `red`, `green` și `blue`:

1. Urmăriți tabela ARP.
2. Inițiați comunicație cu celelalte stații pentru a popula tabela ARP.
3. Urmăriți din nou tabela ARP.

Observați că în tabela ARP a fiecăreia dintre stațiile `red`, `green` și `blue` se găsește câte o intrare ARP, corespunzătoare stației `host`. Aceasta se întâmplă întrucât comunicațiile trec prin *default gateway* (adică prin stația `host`) iar fiecare stație trebuie să cunoască doar adresa MAC a gateway-ului.

06. [10p] Depanare problemă de configurare adresă IP

Rulați scriptul de pregătire cu noul argument `ex6`:

```
root@host:~# start_lab ip ex6
```

În urma rulării scriptului a fost repornită stația `red` și au fost refăcute configurațiile. Va trebui să vă reconectați pe stația `red` folosind comanda:

```
root@host:~# go red
```

Scriptul configurează adresa IP `7.7.7.1` pe interfața `veth-red` a stației `host` și adresa IP `7.7.7.2` pe interfața `red-eth0` a stației `red`. Pentru a afișa configurația IP pe cele două interfețe folosiți comenzile:

```
root@host:~# ip address show veth-red
47: veth-red: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 4e:1b:b8:d9:14:bb brd ff:ff:ff:ff:ff:ff
    inet 7.7.7.1/32 scope global veth-red
```

```
root@red:~# ip address show red-eth0
46: red-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:16:3e:8e:84:21 brd ff:ff:ff:ff:ff:ff
    inet 7.7.7.2/24 scope global red-eth0
    inet6 fe80::216:3eff:fe8e:8421/64 scope link
       valid_lft forever preferred_lft forever
```

Folosiți comanda `ping` pentru a testa conectivitatea între cele două adrese IP (`7.7.7.1` și `7.7.7.2`) pe cele două stații. Observați ca nu există conectivitate.

Pentru a depana această problemă, urmărim tabela de rutare a fiecărei stații:

```
root@red:~# ip r s
default via 7.7.7.1 dev red-eth0
7.7.7.0/24 dev red-eth0 proto kernel scope link src 7.7.7.2
```

```
root@host:~# ip r s
default via 10.9.0.1 dev eth0
10.9.0.0/16 dev eth0 proto kernel scope link src 10.9.3.210
169.254.169.254 via 10.9.0.100 dev eth0
192.168.2.0/24 dev veth-green proto kernel scope link src 192.168.2.1
192.168.3.0/24 dev veth-blue proto kernel scope link src 192.168.3.1
```

Observați că pe `host` nu apare ruta relevantă (7.7.7.0/24) în tabela de rutare. Fie interfața este dezactivată, fie configurația este greșită. Uitați-vă cu atenție la informațiile de nivel 3:

```
root@host:~# ip address show veth-red
47: veth-red: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 4e:1b:b8:d9:14:bb brd ff:ff:ff:ff:ff:ff
    inet 7.7.7.1/32 scope global veth-red
```

```
root@red:~# ip address show red-eth0
46: red-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:16:3e:8e:84:21 brd ff:ff:ff:ff:ff:ff
    inet 7.7.7.2/24 scope global red-eth0
    inet6 fe80::216:3eff:fe8e:8421/64 scope link
       valid_lft forever preferred_lft forever
```

Putem observa că interfețele sunt active (UP). Însă una dintre aceaste adrese (7.7.7.1) are masca `/32`. Acest lucru înseamnă că nu pot comunica în rețea unele cu altele și explică și absența rutei relevante din tabela de rutare.

Repararea acestei greșeli se face prin adăugarea adresei IP cu masca corectă `7.7.7.1/24` pe interfața `veth-red` de pe `host`. Verificați că aveți conectivitate între `host` și `red`.

Nu uitați să ștergeți adresa greșită folosind comanda `ip address delete 7.7.7.1/32 dev veth-red`. Dacă nu ștergeți adresa greșită veți avea 2 adrese IP pe interfață, una cu masca `/24` și alta cu mască `/32`

O greșeală relativ frecventă în configurarea adresei IP în Linux este omiterea măștii de rețea. Aveți în vedere să nu omiteți masca în momentul în care configurați adrese IP pe interfețe în Linux.

Listarea tabelei de rutare a unei stații este printre primii pași care trebuie urmați pentru depanarea unei probleme de conectivitate.

07. [10p] Depanare problemă de conectivitate

Ne propunem să depanăm o problemă de conectivitate. Pentru a “genera” problema rulați scriptul de pregătire cu noul argument `ex7`:

```
root@host:~# start_lab ip ex7
```

Pentru depanare, primul pas recomandat este afișarea tabelei de rutare. Tabela de rutare vă va ajuta pentru depanare în cazul în care anumite intrări sunt absente sau configurate greșit.

Verificați conectivitatea între toate stațiile din topologie. Observați că nu există conectivitate între nici o stație și stația `blue`. Rezolvați problemele de conectivitate la stația `blue` de pe stația `host`.

Identificați și soluționați problemele.

08. [10p] IPv6

Dorim să asigurăm conectivitate IPv6 între stația host și red. În acest tutorial vom folosi suita `iproute` din Linux pentru a realiza configurările necesare. Folosiți parametrul `-6` pentru a face setările aferente IPv6.

Vom configura câte o adresă IP din clasa `2201::/64` pe interfețele de legătură dintre stația `host` și stația `red`. Adică între `host(veth-red)` (interfața `veth-red` de pe stația `host`) și `red(red-eth0)` (interfața `red-eth0` de pe stația `red`).

Pe interfața `veth-red` de pe stația `host` vom configura adresa IP `2201::1/64`

```
root@host:~# ip -6 address add 2201::1/64 dev veth-red
```

Imediat după o configurare de rețea rulați o comandă pentru validarea configurării. În cazul nostru este comanda de afișare a adresei IPv6:

```
root@host:~# ip -6 address show dev veth-red
47: veth-red: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2201::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe5b:a38e/64 scope link
       valid_lft forever preferred_lft forever
```

Pe interfața `red-eth0` de pe stația `red` vom configura adresa IP `2201::2/64`:

Configurați pe legătura host-blue adrese IPv6 din rețeaua 2202::/64 și pe legătura host-green adrese IPv6 din rețeaua 2203::/64.

Activați rutarea pentru IPv6 pe stația `host`:

```
root@host:~# sysctl -w net.ipv6.conf.all.forwarding=1
```

De asemenea, trebuie să adăugați rute default pe `red`, `green` și `blue`, către `host`.

Verificați conectivitatea între containere folosind comanda **ping**

09. [BONUS - 10p] Configurare persistentă

Dorim ca la o reponire a unei stații configurațiile de nivel 3 (adresele IP) să se păstreze. Configurările pe care le-am făcut până acum sunt temporare și sunt pierdute la repornirea stației. În Linux, persistența configurațiilor se realizează prin plasarea acestora în fișiere text specifice, fiecare distribuție (ex.: Debian, RedHat) având propriul mod de configurare.

Pentru a pregăti exercițiul, rulați scriptul de pregătire:

```
root@host:~# start_lab ip ex9
root@host:~# ip address flush dev veth-red
```

Realizați **persistent** configurația de la exercițiul 08. [10p] IPv6 (adrese IP și *default gateway*) pentru `host`. Distribuția de Linux folosită în laborator este Debian-based.

Pentru detalii despre cum puteți face configurații persistent pe sisteme Debian, consultați această pagină [https://wiki.debian.org/NetworkConfiguration#Setting_up_an_Ethernet_Interface]. Veți realiza o configurație statică.

După ce ați realizat configurațiile necesare pentru red, executați pe host:

```
ifdown veth-red
ifup veth-red
```

Observați că la executarea comenzii `ifup` interfața a preluat configurările din fișier.

Configurați sistemul astfel încât rutarea să fie activată la pornirea sistemului.

Pentru informații legate de activarea rutării, consultați această pagină [http://linuxpoison.blogspot.ro/2008/01/how-to-enable-ip-forwarding.html].

Reporniți mașina virtuală (stația `host`), folosind comanda:

```
root@host:~# reboot
```

După repornire ar trebui să aveți configurațiile activate și conectivitate completă la nivelul topologiei.

Navigare

Laboratorul 6

- 01. [10p] Configurare și ștergere adrese IP
- 02. [20p] Configurare adrese IP
- 03. [10p] Adresare IP și rutare
- 04. [20p] Configurare conectivitate completă
- 05. [10p] Tabela ARP
- 06. [10p] Depanare problemă de configurare adresă IP
- 07. [10p] Depanare problemă de conectivitate
- 08. [10p] IPv6
- 09. [BONUS - 10p] Configurare persistentă