Laborator APC - 1

Arhitecturi și echipamente de rețea

Obiective

- Această primă lucrare de laborator oferă o imagine de ansamblu asupra subiectelor studiate în cursul APC și ilustrează concepte fundamentale privind componentele unei rețele și interacțiunile dintre ele. Veți efectua experimente în care intervin protocoale de comunicație de pe toate nivelurile stivei TCP/IP și diverse tipuri de echipamente de rețea. Vom reveni asupra acestor subiecte în laboratoarele următoare, pentru un studiu mai aprofundat.
- De asemenea, veți face cunoștință cu emulatorul de rețele `netkit`, pe care îl vom folosi și în lucrările următoare. Rețeaua este emulată prin mașini virtuale Linux care sunt configurate astfel încât să opereze ca rutere, comutatoare Ethernet, servere sau calculatoare personale. Astfel, veți putea efectua experimente cu rețele relativ complexe, lucrând indepedent pe câte un calculator și având acces deplin la fiecare echipament și la comunicațiile dintre ele.
- În fine, vă veți familiariza cu câteva programe utilitare pe care le vom folosi și în celelalte lucrări, de exemplu analizoarele de protocoale `tcpdump` și `wireshark`, și o serie de comenzi Linux/Unix pentru configurarea și examinarea stării subsistemului de comunicație.

Precondiții

Pentru a putea efectua experimentele și a interpreta rezultatele trebuie să studiați în prealabil capitolele din materialul de curs (și eventual bibliografia suplimentară) care prezintă noțiunile de bază privind arhitecturi de rețele și echipamente.

Software și echipamente

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

Vom folosi implementări ale protocoalelor și echipamentelor disponibile în sistemul de operare Linux. Veți captura și analiza comunicațiile dintre echipamente folosind analizoarele de protocoale `tcpdump` și `wireshark`.

A. Comunicații la nivel legătură de date și nivel rețea

Studiu de caz

În prima parte a lucrării vom folosi rețeaua din Figura 1, alcătuită din 2 subrețele interconectate prin ruterul r1. În fiecare subrețea, echipamentele comunică prin intermediul unui comutator (punte) Ethernet (s1 și s2).

Figura 1 indică blocul de adrese IPv4 alocat fiecărei subrețele și adresa alocată fiecărei interfețe. Un bloc de adrese este o secvență de adrese IPv4 cu același prefix și este identificat prin prima adresă și lungimea prefixului. De exemplu, notația 10.0.1.0/24 reprezintă blocul de adrese care începe cu 10.0.1.0 și are un prefix de 24 biți, deci ultima adresă este 10.0.1.255. Interfețelor conectate la o subrețea le sunt alocate adrese IP din blocul alocat subrețelei.

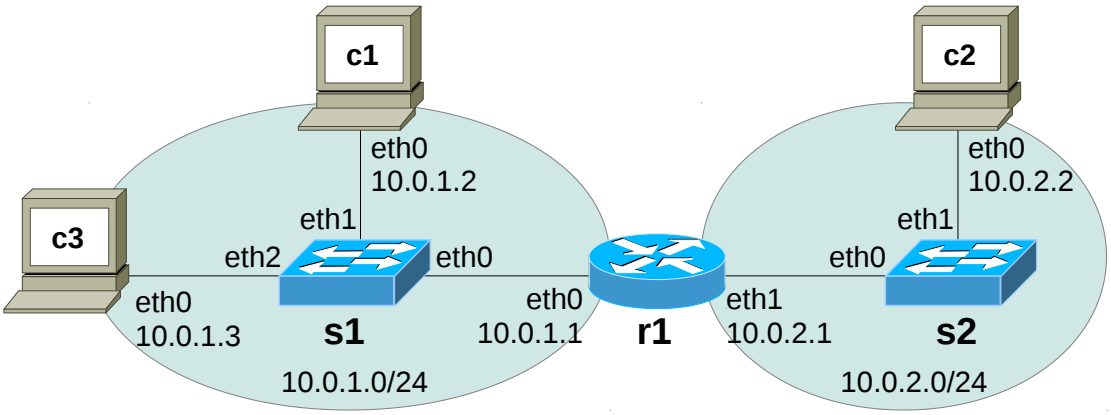

Figura 1: Rețeaua utilizată în prima parte a lucrării de laborator.

A.1. Inițializarea rețelei și examinarea stării inițiale

A.1.1. Pentru a porni emularea rețelei din Figura 1, executați comanda `lstart` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

A.1.2. Ruterul r1 și comutatoarele Ethernet s1 și s2 sunt deja configurate. Examinați starea interfețelor acestor echipamente, folosind comanda `ifconfig`, ca în exemplele de mai jos (numele interfețelor sunt indicate în Figura 1). Interpretați informația afișată.

```
s1:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 4e:8f:ad:b7:22:c8
          inet6 addr: fe80::4c8f:adff:feb7:22c8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
...
s1:~# ifconfig eth1
...
```

```
r1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0e:ab:f8:0c:10:4b
          inet addr:10.0.1.1  Bcast:10.0.1.255  Mask:255.255.255.0
          inet6 addr: fe80::cab:f8ff:fe0c:104b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
...
eth1      Link encap:Ethernet  HWaddr fa:de:dc:30:96:57
          inet addr:10.0.2.1  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::f8de:dcff:fe30:9657/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
...
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

Explicații (parțiale):

Interfețele listate mai sus au fost activate administrativ (UP) și funcționează corect. Toate interfețele fizice (HW) ale echipamentelor sunt interfețe Ethernet și fiecare dintre ele are alocată o adresă de nivel 2 distinctă (adresa MAC, numită aici `HWaddr`).

Comutatoarele (punțile) Ethernet operează pe nivelul 2 (legătură de date), deci au nevoie doar de adrese MAC. De fapt, aceste adrese MAC sunt folosite doar pentru comunicații între comutatoare, în planul de control (de exemplu, în protocolul STP, pe care îl vom studia în altă lucrare).

Ruterele operează pe nivelul 3 (rețea), folosind ambele versiuni ale protocolului IP, IPv4 și IPv6. Prin urmare, interfețelele lor au și adrese de nivel 3: o adresă IPv4 (`inet addr`) care a fost configurată manual și o adresă IPv6 (`inet6 addr`) care a fost configurată automat. Această adresă IPv6 poate fi utilizată doar local, în subrețeaua respectivă (`Scope:Link`). Valoarea adresei IP și lungimea prefixului său identifică univoc blocul de adrese IPv4 alocat subrețelei la care este conectată interfața. Pentru IPv4 lungimea prefixului este specificată aici prin masca de subrețea.

Există și interfețe logice (SW), de exemplu interfața loopback (notată `lo`), care este folosită exclusiv pentru comunicații între procese care rulează pe echipamentul respectiv și are alocată (în acest exemplu) o adresă rezervată special în acest scop.

A.1.3. Examinați starea interfețelor calculatoarelor (indicate în Figura 1), folosind comanda `ifconfig`, ca în exemplul de mai jos. Interpretați informația afișată.

```
c1:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr ee:40:19:05:85:e9
          BROADCAST MULTICAST  MTU:1500  Metric:1
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
...
```

Indicații: Interfețele Ethernet ale calculatoarelor nu au fost configurate: nu sunt activate și nu au adrese IP alocate; adresele MAC sunt configurate de fabricant.

A.1.4. Examinați tabelele de rutare ale echipamentelor folosind comanda `route` (sau comanda `netstat -r`), ca în exemplul de mai jos. Interpretați informația afișată.

```
r1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        *               255.255.255.0   U     0      0        0 eth0
10.0.2.0        *               255.255.255.0   U     0      0        0 eth1
```

```
s1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
```

```
c1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
```

Indicații (parțiale):

Tabelele de rutare ale echipamentelor s1 și s2 nu conțin nicio înregistrare pentru că sunt mașini virtuale Linux configurate să opereze ca punți (comutatoare) Ethernet și nu au adrese IP alocate. Nici calculatoarele nu au nicio înregistrare în tabelele de rutare, pentru că interfețele lor nu au fost configurate încă pentru comunicații IP. Tabelele de rutare ale ruterului r1 conține cele 2 rute corespunzătoare celor 2 subrețele conectate direct, create automat în urma configurării interfețelor.

Comanda listează tabelul de rutare folosit de IPv4. Fiecare rută specifică subrețeaua destinație, prin prefixul blocului de adrese IPv4 care i-a fost alocat (`Destination, GenMask`) și modul în care sunt dirijate pachetele a căror adresă destinație este în subrețeaua respectivă: interfața pe care trebuie transmise pachetele (`Iface`) și ruterul căruia trebuie să-i fie transmise (`Gateway`) pentru a ajunge la destinația respectivă. În acest exemplu, cele 2 subrețele sunt conectate direct la ruter, deci nu este nevoie de un ruter intermediar.

A.2. Configurarea calculatoarelor pentru comunicații folosind IPv4

A.2.1. Configurați interfețele calculatoarelor folosind comanda `ifconfig`, ca în exemplul de mai jos (adresele IP sunt indicate în Figura 1):

```
c1:~# ifconfig eth0 10.0.1.2 netmask 255.255.255.0 up
```

Indicații: Comanda activează interfața (`up`), îi alocă o adresă IPv4 din blocul de adrese alocat subrețelei la care este conectată și specifică masca de subrețea (lungimea prefixului).

A.2.2. Verificați rezultatul configurării interfețelor calculatoarelor folosind comanda `ifconfig`:

```
c1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr ee:40:19:05:85:e9
          inet addr:10.0.1.2  Bcast:10.0.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ec40:19ff:fe05:85e9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
...
```

Au fost activate toate interfețele? Au toate interfețele adrese IPv4 configurate corect (valoare mască), ca în Figura 1?

A.2.3. Examinați tabelele de rutare ale calculatoarelor folosind comanda `route` (sau comanda `netstat -r`), ca în exemplul de mai jos. Interpretați informația afișată.

```
c1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        *               255.255.255.0   U     0      0        0 eth0
```

Cum a apărut această rută?

A.2.4. Verificați conectivitatea IP între c1, c2, c3 și r1 folosind utilitarul `ping`, ca în exemplul de mai jos. Explicați rezultatul testelor.

```
c1:~# ping -c3 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=0.334 ms
64 bytes from 10.0.1.3: icmp_seq=2 ttl=64 time=0.681 ms
64 bytes from 10.0.1.3: icmp_seq=3 ttl=64 time=0.564 ms
--- 10.0.1.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.334/0.526/0.681/0.145 ms
c1:~# ping -c3 10.0.2.2
connect: Network is unreachable
```

Indicații: Programul `ping` testează conectivitatea IP între echipamentul pe care este executat și echipamentul cu adresa IP indicată (în primul exemplu, c1 și c3). Acest test folosește protocolul ICMP: `ping` transmite echipamentului indicat un pachet IP conținând un mesaj ICMP Echo

Request, iar acesta răspunde cu un pachet IP conținând mesajul ICMP Echo Reply. Opțiunea `-c3` solicită repetarea testului de 3 ori. Testul permite și măsurarea întârzierii de transfer dus-întors, notată `rtt` (round-trip time).

De ce echipamentele c1, c3 și r1 (sau c2 și r1) pot comunica între ele, iar c1 și c2 (sau c2 și c3) nu pot comunica?

A.2.1. Configurați pe fiecare calculator o rută statică implicită (default) folosind comanda `route` și apoi examinați tabelul de rutare, ca în exemplul de mai jos (adresele IP sunt indicate în Figura 1):

```
c1:~# route add default gw 10.0.1.1 dev eth0
c1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        *               255.255.255.0   U     0      0        0 eth0
default         10.0.1.1        0.0.0.0         UG    0      0        0 eth0
```

Indicații: Ruta implicită este folosită atunci când adresa destinație a pachetului nu se potrivește cu prefixul niciunei rute explicite.

A.2.4. Testați din nou conectivitatea IP între c1 și c2 (sau c3 și c2) folosind utilitarul `ping`. Explicați rezultatul testului pe baza modificărilor efectuate în tabelele de rutare.

A.3. Transferul pachetelor IP

În experimentele următoare vom folosi utilitarul `tcpdump` pentru a captura și analiza pachetele transferate între echipamentele din rețea în timpul unor teste efectuate cu `ping`.

A.3.1. Capturați traficul pe interfața `eth1` a punții s1:

```
s1:~# tcpdump -ten -i eth1
...
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
```

Consultați pagina de manual a utilitarului (`man tcpdump`) și explicați opțiunile folosite.

A.3.2. Generați trafic IP între c1 și c3 folosind `ping` și analizați traficul capturat de `tcpdump`, ca în exemplul următor. Explicați secvența de pachete transferate.

```
c1:~# ping -c2 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=10.1 ms
64 bytes from 10.0.1.3: icmp_seq=2 ttl=64 time=0.734 ms
...
```

```
s1:~# tcpdump -ten -i eth1
...
ee:40:19:05:85:e9 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: arp
who-has 10.0.1.3 tell 10.0.1.2
f6:7d:1f:b9:80:77 > ee:40:19:05:85:e9, ethertype ARP (0x0806), length 42: arp
reply 10.0.1.3 is-at f6:7d:1f:b9:80:77
ee:40:19:05:85:e9 > f6:7d:1f:b9:80:77, ethertype IPv4 (0x0800), length 98:
10.0.1.2 > 10.0.1.3: ICMP echo request, id 16130, seq 1, length 64
f6:7d:1f:b9:80:77 > ee:40:19:05:85:e9, ethertype IPv4 (0x0800), length 98:
10.0.1.3 > 10.0.1.2: ICMP echo reply, id 16130, seq 1, length 64
```

```
ee:40:19:05:85:e9 > f6:7d:1f:b9:80:77, ethertype IPv4 (0x0800), length 98:
10.0.1.2 > 10.0.1.3: ICMP echo request, id 16130, seq 2, length 64
f6:7d:1f:b9:80:77 > ee:40:19:05:85:e9, ethertype IPv4 (0x0800), length 98:
10.0.1.3 > 10.0.1.2: ICMP echo reply, id 16130, seq 2, length 64
...
```

Indicații:

Identificați interfețele corespunzătoare adreselor MAC folosind `ifconfig`.

Se observă că între c1 și c3 au avut loc 2 tipuri de interacțiuni: un dialog cu protocolul ARP, în urma căruia c1 află adresa MAC a lui c3, urmat de un dialog cu protocolul ICMP, care reprezintă testul de conectivitate propriu-zis, repetat de 2 ori.

Pe calculatorul c1, programul `ping` solicită modulului IP să transmită un prim mesaj ICMP Echo Request către adresa destinație 10.0.1.3 (c3). Consultând tabelul său de rutare, c1 află că pachetul trebuie dirijat pe ruta către subrețeaua 10.0.1.0/24 (același prefix), cu livrare directă, folosind protocolul de nivel 2 Ethernet, prin interfața `eth0`. Nivelul 3 (IP) trebuie să solicite nivelului 2 (Ethernet) să transmită pachetul IP echipamentului destinație, specificând adresa MAC a destinației (nu adresa IP, care nu are sens pentru nivelul 2). Pentru a afla adresa MAC, c1 interoghează echipamentele din subrețeaua respectivă printr-un pachet ARP Request care specifică adresa IP a destinației (într-un cadru Ethernet cu adresa destinație MAC broadcast). Calculatorul c3 răspunde cu ARP Reply, furnizând adresa MAC solicitată. Odată obținută adresa MAC, c1 poate să-i transmită lui c3, într-un cadru Ethernet, pachetul IP conținând ICMP Echo request.

Observați că dialogul ARP Request/Reply nu mai este repetat atunci când c3 transmite răspunsul ICMP Echo Reply: c3 află adresa MAC a lui c1 din ARP Request. Mai mult, dialogul nu mai este necesar nici atunci când c1 efectuează al doilea test, transmițând încă un mesaj ICMP Echo Request, deoarece ARP stochează adresele pe care le-a aflat în memoria sa cache.

A.3.3. Repetați testul `ping` pentru c1 și c3 și apoi examinați conținutul ARP cache folosind comanda `arp`. Exemplu:

```
c1:~# ping -c1 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=7.29 ms
...
c1:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.1.3                 ether   f6:7d:1f:b9:80:77   C                     eth0
```

```
c3:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.1.2                 ether   ee:40:19:05:85:e9   C                     eth0
```

Indicație: Înregistrările din ARP cache sunt eliminate dacă nu mai sunt folosite un anumit interval de timp. De ce credeți că este necesar acest lucru? Dați câteva example.

A.3.4. Repetați experimentul precedent pentru o comunicație între c1 și c2. Capturați traficul pe interfața `eth0` a lui s1 și pe interfața `eth0` a lui s2. Explicați secvența de pachete capturate pe cele 2 interfețe și operațiile efectuate de c1, r1 și c3 pentru livrarea pachetelor.

A.4. Transferul cadrelor Ethernet într-o rețea cu punți

Experimentele precedente au demonstrat că punțile s1 și s2 livrează corect cadrele Ethernet, pe baza

adreselor MAC. De asemenea, transferul este transparent pentru echipamentele interconectate prin punți (ca și când ar exista o conexiune directă între ele). Vom explora acum (sumar) funcționarea punților transparente (definite în standardul IEEE 802.1D).

Algoritmul de dirijare a cadrelor folosește un tabel cu înregistrări care conțin adresa MAC a unui echipament din rețea și portul pe care va fi transmis un cadru pentru a fi livrat la adresa respectivă. Când primește un cadru, puntea caută în tabel înregistrarea care conține adresa destinație a cadrului și îl retransmite pe portul indicat. Dacă adresa căutată lipsește din tabel, cadrul este retransmis pe toate porturile, cu excepția celui pe care a fost recepționat (ceea ce este ineficient și trebuie evitat). Înregistrările sunt create și actualizate pe baza cadrelor recepționate de punte: o înregistrare din tabel conține de fapt adresa sursă a unui cadru recepționat și numărul portului pe care a sosit.

A.4.1. Examinați tabelul de adrese al fiecărei punți folosind comanda `brctl showmacs`. Exemplu:

```
s1:~# brctl showmacs br0
port no mac addr                is local?       ageing timer
  3     2e:62:02:56:5d:fd       yes                0.00
  1     4e:8f:ad:b7:22:c8       yes                0.00
  2     b6:ab:81:26:f7:ed       yes                0.00
```

```
s2:~# brctl showmacs br0
port no mac addr                is local?       ageing timer
  2     62:6d:e3:1e:d1:8d       yes                0.00
  1     ee:c3:5f:ff:7b:6d       yes                0.00
```

Indicații: Inițial, puntea nu cunoaște decît adresele propriilor porturi. O înregistrare este eliminată dacă nu mai este folosită un anumit interval de timp (nu a fost primit niciun cadru cu adresa sursă respectivă). De ce credeți că este necesar acest lucru?

A.4.2. Generați trafic între c1 și c3 folosind `ping`, apoi examinați efectul acestui trafic asupra tabelelor de adrese ale punților. Exemplu:

```
c1:~# ping -c1 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=0.825 ms
```

```
s1:~# brctl showmacs br0
port no mac addr                is local?       ageing timer
  3     2e:62:02:56:5d:fd       yes                0.00
  1     4e:8f:ad:b7:22:c8       yes                0.00
  2     b6:ab:81:26:f7:ed       yes                0.00
  2     ee:40:19:05:85:e9       no                24.40
  3     f6:7d:1f:b9:80:77       no                24.40
```

```
s2:~# brctl showmacs br0
port no mac addr                is local?       ageing timer
  2     62:6d:e3:1e:d1:8d       yes                0.00
  1     ee:c3:5f:ff:7b:6d       yes                0.00
```

Observați că s1 a adăugat 2 înregistrări, pentru c1 și c3. Explicați cum au fost create și folosite aceste înregistrări în timpul testului, pornind de la traficul capturat în experimentul A.3.2. De ce tabelul comutatorului s2 a rămas nemodificat?

A.5. Terminarea primei părți a lucrării

Terminați emularea rețelei din Figura 1 executând comanda `lcrash` într-un terminal al

calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

B. Comunicații la nivel transport și nivel aplicație

În a doua parte a lucrării vom efectua experimente cu protocoale de nivel 4 (transport) și nivel aplicație. Un prim aspect care trebuie clarificat este faptul că sunt necesare alte metode de adresare.

Aplicațiile distribuite folosesc scheme de adresare specifice, adaptate propriilor cerințe. În Internet, acestea se bazează de obicei pe nume de domeniu, o schemă simplă și scalabilă, gestionată la nivel global de sistemul DNS (Domain Name System). De pildă, URL-ul (Unique Resource Locator) `http://www.oda.org/index.html`, identifică fișierul `index.html`, disponibil pe serverul cu numele de domeniu `www.oda.org` și accesibil folosind protocolul de nivel aplicație HTTP.

Spațiul numelor de domeniu are o structură de arbore. Fiecare nod din arbore, cu excepția nodului rădăcină, are o etichetă. Fiecărui nod îi este asociat un nume de domeniu, obținut prin concatenarea etichetelor nodurilor aflate pe calea către nodul rădăcină. Un subarbore reprezintă un domeniu și este identificat prin numele nodului său rădăcină. De exemplu, vom presupune că am obținut pentru rețeaua noastră domeniul `oda.org`, astfel încât putem numi echipamentele noastre `c1.oda.org`, `www.oda.org`, și așa mai departe (noduri în subarborele cu rădăcina `oda.org`).

Am văzut deja, în experimentele cu comunicații IP pe o legatură de date Ethernet, că modulul IP folosește protocolul ARP pentru a afla adresa MAC corespunzătoare adresei IP a echipamentului căruia trebuie sa îi trimită pachetul. Din motive similare, sistemul DNS permite unui protocol de nivel aplicație să afle adresa IP corespunzătoare unui nume de domeniu, folosind protocolul DNS.

Rolul nivelului transport este să completeze serviciul oferit de nivelul rețea cu funcții necesare comunicațiilor capăt la capăt, între procese de nivel aplicație. O primă funcție este adresarea capăt la capăt. Pe un echipament pot rula mai multe procese în paralel și oricare dintre ele poate fi implicat în mai multe comunicații în același timp. Schema de adresare oferită de nivelul transport combină o adresă IP, care identifică un echipament în rețea, cu un identificator local, numit număr de port, care identifică un capăt de comunicație în echipamentul respectiv.

Studiu de caz

În a doua parte a lucrării vom folosi rețeaua din Figura 2. Rețeaua este structurată în 3 subrețele interconectate prin 2 rutere și include un server DNS și un server web (HTTP).

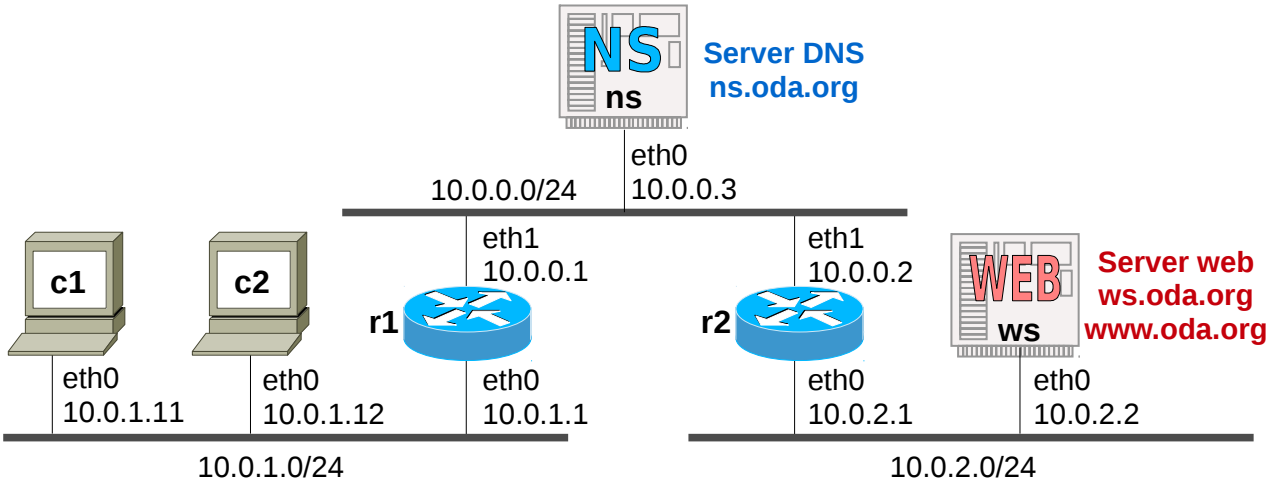

Figura 2: Rețeaua folosită în a doua parte a lucrării de laborator.

Pentru a facilita capturarea traficului şi analiza comunicaţiilor, echipamentele comunică direct, ca şi când ar fi conectate la un hub Ethernet. Prin urmare, puteţi captura întregul trafic dintr-o subreţea executând `tcpdump` pe oricare echipament conectat la subreţeaua respectivă. De asemenea, această facilitate oferită de `netkit` permite reducerea numărului de maşini virtuale.

B.1. Iniţializarea reţelei şi examinarea stării iniţiale

B.1.1. Pentru a porni emularea reţelei din Figura 2, executaţi comanda `lstart` într-un terminal al calculatorului gazdă, în directorul în care se află fişierele de configurare `netkit` ale acestei reţele.

B.1.2. Echipamentele din Figura 2 sunt parţial configurate. Examinaţi starea iniţială a interfeţelor, cu comanda `ifconfig`, şi conţinutul iniţial al tabelele de rutare, folosind comanda `route -n`.

Comparaţi informaţia afişată de aceste comenzi cu Figura 2:

- Sunt activate şi configurate corect toate interfeţele?
- Ce rute sunt necesare pentru succesul unui test folosind ping între 10.0.1.11 şi 10.0.2.2? Dar pentru un test între 10.0.1.12 şi 10.0.0.3?
- Permite configuraţia iniţială comunicaţii IP între toate echipamentele din reţea?

B.1.3. Adăugaţi ruta care lipseşte din tabelul de rutare al ruterului r1 folosind comanda `route`, ca în exemplul de mai jos. Verificaţi apoi efectul executării acestei comenzi asupra tabelului de rutare.

```
r1:~# route add -net 10.0.2.0/24 gw 10.0.0.2 dev eth1
```

```
r1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.0   U     0      0        0 eth1
10.0.1.0        *               255.255.255.0   U     0      0        0 eth0
10.0.2.0        10.0.0.2        255.255.255.0   UG    0      0        0 eth1
```

B.1.4. Adăuga rutele care lipsesc pe r2 şi ns ca în exemplul precedent.

Indicaţie: Nu este recomandat să folosiţi o rută implicită pe calculatorul ns.

B.1.5. Verificaţi folosind `ping` faptul că, după efectuarea acestor modificări, toate echipamentele din reţea pot să comunice între ele.

B.2. Nume de domeniu şi serviciul DNS

Vom efectua acum câteva experimente simple cu comunicaţii între aplicaţii care folosesc nume de domeniu şi (în consecinţă) au nevoie de serviciul DNS.

Protocolul de nivel aplicaţie DNS permite clienţilor să obţină de la server informaţiile dorite printr-un dialog cerere-răspuns (de pildă adresa IP asociată unui nume de domeniu). Aceste 2 mesaje DNS sunt transmise de obicei folosind protocolul de nivel transport UDP. Serverul DNS aşteaptă cereri pe portul UDP cu numărul 53, rezervat pentru acest serviciu.

Serverul DNS rulează pe calculatorul ns (10.0.0.3, `ns.oda.org`). Toate echipamentele au fost configurate astfel încât clienţii DNS să apeleze la acest server.

Pentru o analiză mai detaliată a comunicaţiilor, vom folosi `tcpdump` pentru a captura traficul şi a-l salva într-un fişier, apoi vom vizualiza traficul capturat folosind analizorul de protocoale Wireshark (`netkit` nu permite rularea unor programe cu interfaţa grafică din terminalul unei maşini virtuale).

B.2.1. Porniți captura traficului pe interfața `eth0` a ruterului r1:

```
r1:~# tcpdump -s0 -i eth0 -w /hostlab/dnsping.cap
```

Indicații: Traficul capturat va fi salvat în fișierul `dnsping.cap` din directorul conținând fișierele de configurare ale lucrării de laborator.

B.2.2. Executați pe c1 comanda `ping` pentru destinația `www.oda.org`:

```
c1:~# ping -n -c3 www.oda.org
```

B.2.3. Opriți captura pe r1 (Ctrl-C) și vizualizați traficul capturat folosind `wireshark` (executați comanda într-un terminal al calculatorului gazdă):

```
wireshark -r dnsping.cap
```

Explicați comunicațiile care au avut loc, în special interogarea DNS efectuată de `ping`. Ce solicită clientul DNS? Ce răspuns primește? De ce este preferat protocolul UDP, în locul protocoulului TCP, pentru a transporta aceste mesaje DNS?

Indicații: În acest test, am specificat destinația printr-un nume de domeniu, în locul adresei IP. Prin urmare, pentru a putea transmite mesajele ICMP, utilitarul `ping` trebuie să afle adresa IP corespunzătoare numelui de domeniu, folosind serviciul DNS.

B.3. Comunicații folosind HTTP

În final, vom examina comunicațiile care au loc atunci când un program client solicită un fișier aflat pe un server folosind protocolul de nivel aplicație HTTP (Hypertext Transfer Protocol).

HTTP este un protocol cu interacțiuni de tip cerere-răspuns. Mesajele HTTP sunt transmise pe o conexiune TCP. În exemplul nostru, clientul transmite un mesaj HTTP cerere care conține comanda GET și numele fișierului. Serverul răspunde cu un mesaj HTTP care comunică rezultatul executării comenzii (succes sau eșec) și fișierul solicitat (în caz de succes).

Serverul HTTP va rula pe calculatorul ws (10.0.2.2) și va aștepta cereri pe portul TCP cu numărul 80, rezervat pentru acest serviciu.

B.3.1. Porniți captura traficului pe interfața `eth0` a ruterului r1:

```
r1:~# tcpdump -s0 -i eth0 -w /hostlab/httpdns.cap
```

B.3.2. Accesați serverul web folosind programul client (web browser) `links`:

```
c1:~# links http://www.oda.org/~guest/index.html
```

Fișierul conține un scurt mesaj ("Hello! Welcome ..."). Terminați programul `links` apăsând tasta q și apoi y.

B.3.2. Opriți captura pe r1 (Ctrl-C) și vizualizați traficul capturat folosind `wireshark` (executați comanda într-un terminal al calculatorului gazdă):

```
wireshark -r httpdns.cap
```

Explicați comunicațiile care au avut loc (DNS, HTTP, TCP). De ce este preferat protocolul TCP, în locul protocoulului UDP, pentru a transporta mesajele HTTP?

B.4. Terminarea lucrării

Executați comanda `lcrash` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale rețelei din Figura 2.

Bibliografie

1. O. Catrina. APC - Note de curs.
2. TCPDUMP/LIBPCAP Public Repository. http://www.tcpdump.org/
3. Wireshark User's Guide. https://www.wireshark.org/docs/wsug_html_chunked/