


Laboratorul 04 - Forwarding

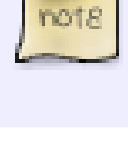
Inainte de laborator

Pentru a simula o retea virtuala vom folosi  Mininet. Vom avea nevoie de urmatoarele pachete: mininet si openvswitch-testcontroller.

```
sudo apt install mininet openvswitch-testcontroller xterm
sudo cp /usr/bin/ovs-testcontroller /usr/bin/ovs-controller
```



Mininet nu functioneaza pe Windows Subsystem for Linux. Este recomandat sa folositi Linux nativ sau intr-un mediu virtualizat. Recomandam Ubuntu 18.04.



La adresa  <https://ocw.cs.pub.ro/courses/pc/res/mv> puteti gasi o masina virtuala cu Ubuntu 18.04.

Prezentare generala

In cadrul laboratorului vom implementa un **router**. Un router are mai multe interfețe si poate receptiona datagrama pe oricare dintre acestea. Routerul trebuie sa transmita pachetul mai departe in functie de multimea de reguli din **tabela de rutare**. In general, tabela de rutare contine urmatoarele informatii:

- Interfața pe care trebuie sa trimita pachetul
- Adresa IP a routerului hop

Se cere implementarea procesului de rutare. (Nu trebuie sa implementati algoritmi de populare a tabeli de rutare e.g. RIP, OSPF, BGP)

Procesul de rutare

Procesul de rutare constă în primirea unui pachet, investigarea tabeli de rutare, descoprirea rutei corespunzătoare și rutarea pachetului, adică transmiterea pachetului mai departe conform rutei.

Rutele se găsesc în tabela de rutare și constau din două elemente:

```
partea de match: adresa de rețea destinație (adresă și mască de rețea)
partea de acțiune: următorul dispozitiv de rutare (next hop) (sau interfața de ieșire)
```

În momentul în care un dispozitiv care rotează (un router) primește un pachet, extrage adresa IP destinație, localizează adresa de rețea destinație în tabela de rutare și apoi rotează (dirijează, retransmite) pachetul către următorul ruter (next hop). Procesul se reia pe următorul ruter până când pachetul ajunge la destinație.

Rutarea este un proces care are loc la nivelul 3 (Rețea) din stiva OSI, lucrând cu adresa IP.

Tabela de rutare din cadrul laboratorului va fi structurata ca in exemplul de mai jos.

Prefix	Next hop	Mask	Interface
192.168.0.0	192.168.0.2	255.255.255.0	0
192.168.1.0	192.168.1.2	255.255.255.0	1
192.168.2.0	192.168.2.2	255.255.255.0	2
192.168.3.0	192.168.3.2	255.255.255.0	3

Protocoloale utilizate


Ethernet

In cadrul laboratorului puteti folosi urmatoarea structura pentru un pachet Ethernet.

Click to display 

IPv4

In cadrul laboratorului puteti folosi urmatoarea structura pentru un pachet IPv4.

Click to display 

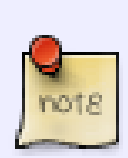
Implementare router

Implementarea routerului se va face in **router.c**. Datagramele primite/trimise de router vor avea urmatorul format:

```
typedef struct {
    int len;
    char payload[MAX_LEN];
    int interface;
} msg;
```

In scheletul routerului se gasesc doua functii: **get_packet** care primeste o datagrama si **send_packet** care trimite o datagrama pe o interfața specificata.

```
int get_packet(msg *m);
int send_packet(int interface, msg *m);
```



Functia get_packet este blocanta.


Pentru a calcula suma de control a unui pachet IP puteti utiliza urmatoarea functie pusa la dispozitie in schelet.

```
uint16_t ip_checksum(void* vdata, size_t length);
```

Routerul trebuie sa implementeze algoritmul de **longest prefix match**. Etapele algoritmului sunt urmatoarele:

- Routerul primește un pachet apeland functia **get_packet**.
- Routerul caută in tabela de rutare intrările care fac match pe adresa destinație a pachetului IP primit.
- Dintre toate rutele pe care s-a făcut match in etapa anterioara, este aleasa ruta cea mai specifica (adica ruta cu prefixul cel mai lung).
- In cazul in care nici o intrare din tabela nu face match, routerul arunca datagrama.
- Este verificat checksum-ul. Daca acesta este incorect, pachetul este aruncat.
- Routerul decrementeaza campul TTL din header-ul IP. In cazul in care TTL-ul e 0 sau mai mic, pachetul este aruncat. Se recalculeaza checksum-ul la acest pas.
- Routerul folosește functia **send_packet** pentru a trimite pachetul la next hop.

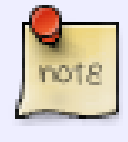
Cerinte laborator:

1. Descarcati  arhiva de laborator si creati rețeaua virtuala folosind comanda de mai jos. Aceasta va porni 6 terminale: 4 pentru hosts, 1 pentru router si 1 pentru controller(nu vom interactiona cu el). Pe hosts se pot rula comenzi precum **ping**, **wget**, **nc** etc. Pe router vom rula programul rezultat in urma rularii comenzii **make**. In cadrul laboratorului vom lucra in fisierul **router.c**.

```
sudo python topo.py
```



Pentru a compila folositi comanda **make** in cadrul terminalului deschis pentru router

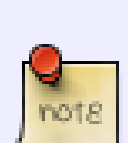


In cazul in care apar probleme, consultati sectiunea [FAQ](#).



Routerul este pornit automat la rularea comenzii python topo.py si output-ul este afisat in /tmp/debug.txt. In cazul in care vreti sa il porniti manual, inchideti procesul actual si porniti ./router

2.1. Implementati procesul de rutare. Tabela de rutare este reprezentata printr-un array si poate fi accesata la un index prin variabila rtable[i]. rtable_size este dimensiunea tabeli. Nu trebuie implementata parsarea fisierului **rtable.txt**. Structura de date utilizate pentru o intrare din tabela este **route_table_entry** din **parser.h**.

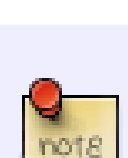


Vom rula executabilul **router** pe terminalul cu numele router

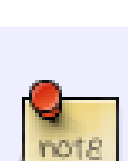
2.2. Completati header-ul Ethernet al pachetului receptat. Astfel, in header-ul Ethernet va trebui completat MAC-ul destinatiei in functie de adresa destinatiei din header-ul IP. De asemenea, trebuie completat si campul sursei cu adresa MAC a interfeței pe care urmeaza sa trimitem pachetul. Nu trebuie implementata parsarea fisierului arp_table.txt. Structura de date utilizate pentru o intrare din tabela este **arp_entry** din skel.h.

Pentru evita utilizarea protocolului ARP, se presupun cunoscute urmatoarele valori:

IP	MAC
192.168.0.2	de:ad:be:ef:00:00
192.168.1.2	de:ad:be:ef:00:01
192.168.2.2	de:ad:be:ef:00:02
192.168.3.2	de:ad:be:ef:00:03



Puteti folosi inet_addr a face conversia adresei IP in binar. Pentru a face conversia MAC-ului in binar puteti folosi functia hwallr_aton pusa la dispozitie in **skel.h**.



Functia int get_interface_mac(int interface, uint8_t *mac) intoarce adresa MAC a interfeței router-ului.

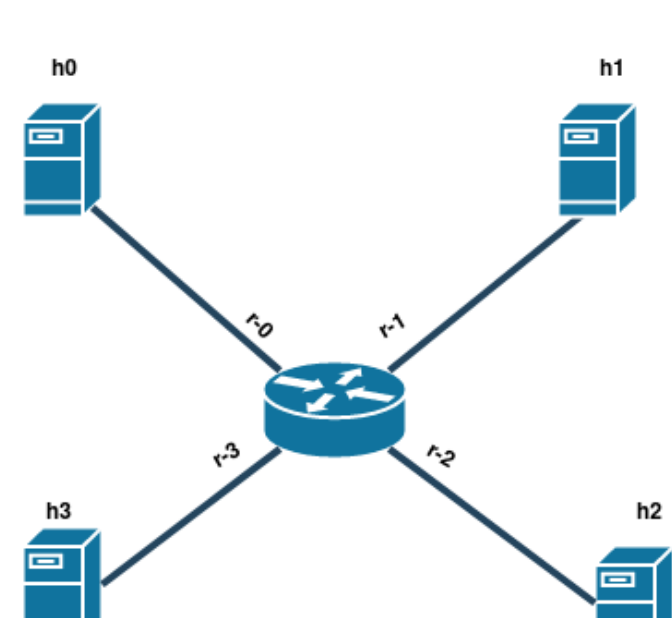
Pentru testare puteti folosi comanda **ping** sau **nc** din terminalele pentru orice host, generand trafic catre alt host. De exemplu, in terminalul host 1 puteti executa:

```
ping h2
```

- Pentru fiecare pachet IP verificati checksum-ul; daca este gresit, pachetul va fi distrus.
- Pentru fiecare pachet scadeti TTL-ul; daca TTL-ul este pozitiv, recalculati checksum-ul.
- BONUS: Implementati parsarea tabeli de rutare.
- BONUS: Implementati un algoritm cu o complexitate mai buna de O(N) pentru longest prefix match.

Testare

Pentru testare vom folosi utilitarele puse la dispozitie de linux pe cei 4 hosts. Reteaua simulata are urmatoarea structura:



FAQ

- Daca aveti eroarea de mai jos trebuie sa instalati **openvswitch-testcontroller** si creat fisierul **/usr/bin/ovs-controller**

```
raise Exception('Could not find a default OpenFlow controller')
```

- Nu imi apar terminalele cand rulez mininet.

```
sudo apt install xterm
```

- In cazul in care portul este ocupat rulati **sudo fuser -k 6653/tcp**

```
Exception: Please shut down the controller which is running on port 6653
```

- Cum pot reprezenta o adresa IP in memorie?

```
uint32_t
```

- Valorile primite pe retea nu coincid cu valorile la care ma asteptam.



Atentie la network order vs host order

- Cum fac trecerea intre network si host byte order?

```
#include <arpa/inet.h>
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

- Cum extrag header-ul IP dintr-un pachet de tip msg?

```
struct iphdr *ip_hdr = (struct iphdr *) (packet.payload + sizeof(struct ether_header));
```

- Cum pot vedea traficul de pe o interfața?

```
tcpdump
```

- Cum afisez interfețele unui host?

```
ip a s
```

- Ce face daca se trimit multe pachete duplicate? **make distclean** din Linux si **make distclean && make** din cadrul terminalului deschis pentru router.
- Cum folosesc ip_checksum?

```
ip_checksum(ip_hdr, sizeof(struct iphdr)); /* initial header checksum = 0 */
```



Checksum-ul unui pachet se calculeaza cu valoare initiala a checksumului 0.

Resurse Utile

-  Introducere in Mininet
-  Packet Switching
-  Router Design and Packet Lookup
-  RFC 791 - Internet Protocol