

## 1 Prezentare generala

In cadrul temei vom implementa procesul de dirijare a pachetelor dintr-un router (forwarding in engleza). Un router are doua parti, astfel:

- Data-plane - care implementeaza procesul de dirijare. Tema voastra este implementarea acestei componente. In cele ce urmeaza, in lipsa altor precizari, toate referintele la router din textul temei se refera la partea de dirijare a pachetelor.
- Control plane - componenta care implementeaza algoritmi de rutare e.g. RIP, OSPF, BGP; acesti algoritmi distribuie calculeaza routele pentru fiecare retea destinatie si le insereaza in dataplane. NU este nevoie sa implementati acesti algoritmi pentru aceasta tema. Routerul vostru va functiona cu o tabela de rutare statica, data intr-un fisier de intrare, si care nu se schimba in timpul rularii routerului.

Un router are mai multe interfete si poate receptiona datagrame pe oricare dintre acestea. Routerul trebuie sa transmita pachetul mai departe, catre un calculator sau alt router direct conectat, in functie de regulile din tabela de rutare. Se cere implementarea procesului de dirijare a routerului.

## 2 Procesul de dirijare

Procesul de dirijare constă în primirea unui pachet, investigarea tabelii de rutare, descoperirea rutei corespunzătoare și dirijarea pachetului. Dirijarea este un proces care are loc la nivelul 3 (Rețea) din stiva OSI, lucrând cu adresa IP destinatie din pachetele dirijate.

Rutele (in numar arbitrar de mare) 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)

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

Listing 1: Tabela de rutare

| 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
```

---

**Nota:** tabela de mai sus este data cu scop informativ. Tabelele folosite pentru testarea temelor pot contine oricate intrari, oricate interfete, si adrese next hop si masti arbitrare.

În momentul în care un dispozitiv care rutează primește un pachet, extrage adresa IP destinație, localizează cea mai specifica intrare din tabela de rutare care se potrivește cu pachetul si dirijează (retransmite) pachetul către următorul ruter (next hop).

Functionalitatile routerului, executate într-o bucla infinita, sunt:

1. Primește un pachet de la oricare din interfetele adiacente.
2. Daca este un pachet IP destinat routerului, raspunde doar in cazul in care acesta este un pachet ICMP ECHO request. Arunca pachetul original.
3. Dace este un pachet ARP Request catre un IP al routerului, raspunde cu ARP Reply cu adresa MAC potrivita.
4. Daca este un pachet ARP Reply, updateaza tabela ARP; daca exista pachete ce trebuie dirijate catre acel router, transmite-le acum.
5. Daca este un pachet cu TTL  $\leq 1$ , sau un pachet catre o adresa inexistentă in tabela de rutare, trimite un mesaj ICMP corect sursei (vezi mai jos); arunca pachetul.
6. Daca este un pachet cu checksum gresit, arunca pachetul.
7. Decrementeaza TTL, updateaza checksum.
8. Cauta intrarea cea mai specifica din tabela de rutare (numita  $f$ ) astfel incat ( $iph -> daddr \& f.mask == f.prefix$ ). Odata identificata, aceasta specifica next hop pentru pachet. In cazul in care nu se gaseste o ruta, se trimite un mesaj ICMP sursei; arunca pachetul
9. Modifica adresele source si destination MAC. Daca adresa MAC nu este cunoscuta local, genereaza un ARP request si transmite pe interfata destinatie. Salveaza pachetul in coada pentru transmitere. atunci cand adresa MAC este cunoscuta (pasul 4).
10. Trimite pachetul mai departe folosind functia `send_packet(...)`.

Procesul se reia pe următorul router până când pachetul ajunge la destinație. In cadrul temei, va exista un singur router care este conectat direct la un numar de calculatoare gazda emulate (host-uri).

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

---

Listing 2: Functionalitati router.

---

```
typedef struct {
```

```
    int len;
    char payload[MAX_LEN];
    int interface;
} packet;
```

---

În scheletul routerului se găsesc două funcții `get_packet` care primește o datagramă și `send_packet` care trimite o datagramă pe o interfață specificată.

---

Listing 3: Funcționalități router.

---

```
/*
    send_packet nu va elibera memoria pachetului. întoarce zero în caz
    de eroare.
*/
int send_packet(int interface, packet *m);

/*
    receive_packet scrie un pachet la adresa primită
    întoarce 0 în caz de eroare.
*/
int get_packet(packet *m);

/* get the IP of one of the router's interfaces */
char *get_interface_ip(int interface);

/* get the MAC of one of the router's interface */
int get_interface_mac(int interface, uint8_t *mac);
```

---

## 3 Protocoale utilizate

În schelet apar următoarele structuri pentru header-urile protocoalelor utilizate. Puteți să le folosiți sau să definiți propria voastră structură.

### 3.1 Ethernet

În cadrul temei o să lucrați cu frame-uri de Ethernet. Trebuie să înțelegeți ideea de MAC destinație și sursă și faptul că atunci când un pachet este trimis către next hop schimbăm adresa sursă și adresa destinație a pachetului.

În headerul de sistem `netinet/if_ether.h`, puteți găsi o descriere C a antetului de Ethernet sub forma structurii `struct ether_header`.

## 3.2 IP

Un pachet IP poate avea ca destinație una din adresele IP ale routerului sau poate fi destinat în alta rețea. Pachetele cu destinația diferită de interfețele routerului urmăresc procesul normal de forwarding.

În headerul de sistem `netinet/ip.h`, puteți găsi o descriere C a antetului de Ethernet sub forma structurii `struct iphdr`.

## 3.3 ARP

Protocolul ARP este utilizat pentru a determina adresa MAC a next hop-ului. Astfel, routerul va trebui să genereze mesaje de tip ARP REQUEST și ARP RESPONSE. Pentru a evita poluarea rețelei cu cereri ARP va trebui să stocați datele într-o tabelă ARP. Structura tabelii este la latitudinea voastră.

Cererile ARP vor fi trimise ca broadcast(ff:ff:ff:ff). Raspunsurile ARP sunt trimise către adresa MAC a host-ului ce a făcut cererea.

În headerul de sistem `net/if_arp.h`, puteți găsi o descriere C a antetului de ARP sub forma structurii `struct arphdr`.

În headerul de sistem `netinet/if_ether.h`, puteți găsi o descriere C a antetului necesar pentru un reply/request, sub forma structurii `struct ether_arp` (care conține un `struct arphdr` menționat mai sus).

## 3.4 ICMP

Un pachet IP poate conține un mesaj ICMP. Routerul va trebui să genereze următoarele mesaje ICMP ca răspuns:

- **Echo reply** (type 0) - Trimis ca răspuns la un echo request(ping). Trebuie să răspundeți doar mesajelor ICMP care sunt adresate routerului.
- **Destination unreachable** (type 3, code 0) - Trimis în cazul în care nu există ruta până la destinație, atunci când pachetul nu este destinat routerului.
- **Time exceeded** - Trimis dacă TTL-ul este 0.

În headerul de sistem `netinet/ip_icmp.h`, puteți găsi o descriere C a antetului de Ethernet sub forma structurii `struct icmphdr`.

## 4 API suplimentar

Vă punem la dispoziție în schelet un API pentru folosirea unei cozi. Urmăriți comentariile din fișierul `include/queue.h`.

## 5 Cerinte tema

Descarcati arhiva temei si creati reseaua 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 temei vom lucra in fisierul router.c.

- Scrieti cod care parseaza tabela de rutare (10p). Daca folositi functia de parsare din [laboratorul 4 \(parser.o + parser.h\)](#), pierdeti cele 10p pentru aceasta cerinta.
- Implementati protocolul ARP (30p). Tema se poate preda si cu o tabela de ARP statica, cu pierderea punctajului aferent.

În arhivă aveți inclusă o tabelă ARP statică. Puteți folosi funcția de parsare din laboratorul 4.

**Prezența acestui fișier în arhiva va opri checkerul din a rula teste de ARP, deci nu veți primi puncte pe ele. Dacă ați implementat protocolul ARP, ștergeți fișierul arp\_table.txt.**

- Implementati procesul de dirijare (25p).

Algoritmul de cautare in tabela de rutare trebuie sa aiba o complexitate  $O(\log(n))$  sau  $O(1)$ , unde  $n$  este dimensiunea tablei de rutare (15p). Implementarea unei cautari liniare duce la pierderea celor 15p.

**Atentie:** In cazul in care un pachet trebuie trimis mai departe dar inainte de dirijare se face un ARP Request, atunci pachetul va fi pus intr-o coada si va fi dirijat dupa ce s-a primit ARP Reply-ul.

- Implementati suport pentru protocolul ICMP (20p).
- BONUS: Implementati modificarea checksumului dupa decrementarea TTL folosind algoritmul incremental din RFC 1624. (10p)

**Notă:** Puteți scrie implementarea în C sau C++. Pe mașina virtuală de pe vmchecker, există gcc 7.5.0

## 6 Testare

Vom folosi mininet pentru a simula o rețea cu următoarea topologie:

Aveți la dispoziție un script de Python3, `topo.py`, pe care îl puteți rula pentru a realiza setupul de testare. Acesta trebuie rulat ca root:

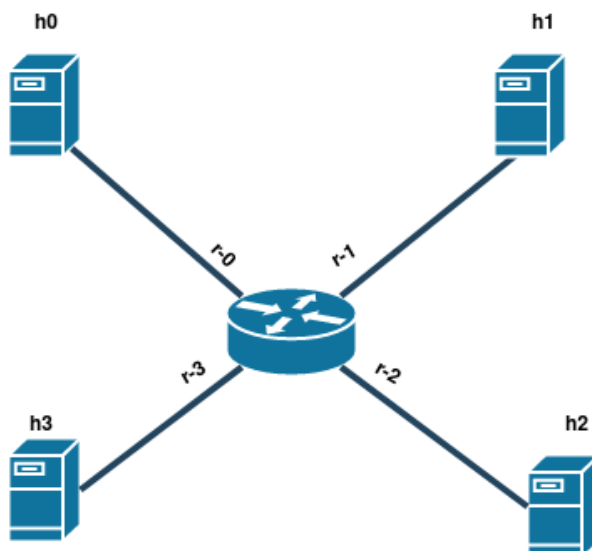
---

```
$ sudo python3 topo.py
```

---

Astfel, se va inițializa topologia virtuală și se va deschide câte un terminal pentru fiecare host, cât și un terminal pentru router și unul pentru controller (cu care nu vom interacționa);

Figure 1: Topologie Tema



terminalele pot fi identificate după titlu. Din terminalele cu nume de forma "h-X", puteți folosi comenzi care generează trafic în IP pentru a testa funcționalitatea routerului implementat. Vă recomandăm [arping](#), [ping](#) și [netcat](#).

Deasemenea, vă punem la dispoziție și o suită de teste, pe care o puteți rula cu argumentul "tests":

---

```
$ sudo python3 topo.py tests
```

---

În urma rulării testelor, va fi generat un folder `host_outputs` care conține, pentru fiecare test, un folder cu outputul tuturor hoștilor (ce au scris la `stdout` și `stderr`). În cazul unui test picat, s-ar putea să găsiți utilă informația de aici, mai ales cea din fișierele de `stderr`.

**Notă:** Scopul testelor este de a ajuta cu dezvoltarea și evaluarea temei. Mai presus de rezultatul acestora, important este să implementați *cerința*. Astfel, punctajul final poate diferi de cel tentativ acordat de teste, în situațiile în care cerința temei nu este respectată (un caz extrem ar fi hardcodarea outputului pentru fiecare test în parte). Vă încurajăm să utilizați modul interactiv al topologiei pentru a explora și alte moduri de testare a temei (e.g. `ping`, `arping`).

## 7 Trimitere

Tema trebuie trimisă pe `vmchecker v2`, la [această adresă](#). Pentru trimitere este necesară o arhivă zip cu numele `Nume_Prenume_Grupa.zip` care să conțină, în radacina următoarele fișiere:

- `README`: în care să explicați pe larg abordarea implementării și problemele întâmpinate.
- Codul sursă al routerului
- `Makefile`: checkerul va rula `make` fără niciun argument

Nu este nevoie să includeți fișierele din checker sau tabela de rutare.

**Notă:** Nu folosiți Vmchecker pentru a face debugging la tema. Tema trebuie dezvoltată și testată temeinic pe calculatorul vostru, cu trafic serios (e.g. iperf), cu multe intrări în tabela de rutare (de exemplu 100K), și încărcată doar atunci când sunteți convinși că este corectă.

**Notă:** Suprîncărcarea Vmchecker prin submisii succesive ale aceeși versiuni ale temei, poate duce la depunere.

**Notă:** Orice probleme de punctare a temelor de către Vmchecker, dacă există, vor fi soluționate de către echipa de la PC până la punctajele finale vor fi anunțate. Nu vor fi acordate extensii ale termenului dacă unele teste din Vmchecker sunt gresite; dacă găsiți astfel de teste, vă rugăm să le semnați. Ele vor fi rezolvate până la anunțarea notelor finale pe tema.

## Deadline

Tema trebuie trimisă până pe data de 5 aprilie 2020, ora 23:59, pentru obținerea punctajului total de 100p. Pentru fiecare zi întârziere după acest termen, punctajul maxim pentru tema scade cu 10p. Numărul maxim de zile de întârziere este 7.

Termenul este hard: nu vor fi acordate nici un fel de extensii.

## Dependințe tema

Pentru a simula o rețea virtuală în care să testăm ruterul nostru, vom folosi emulatorul de rețea Mininet. Vom avea nevoie de următoarele pachete: mininet și openvswitch-testcontroller. Tema va fi rulată pe Linux (recomandăm Ubuntu 18.04). [Aici](#) puteți găsi o mașină virtuală cu Ubuntu 18.04.

---

### Listing 4: Instalare software necesar pentru tema

```
sudo apt install mininet openvswitch-testcontroller xterm python3-pip
sudo cp /usr/bin/ovs-testcontroller /usr/bin/ovs-controller
sudo pip3 install scapy
sudo pip3 install pathlib
sudo pip3 install git+https://github.com/mininet/mininet.git
```

---

## 8 FAQ

- Nu îmi apar terminalele când rulez mininet.

---

```
sudo apt install xterm
```

---

- În cazul în care portul este ocupat ruleți `sudo fuser -k 6653/tcp`

---

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

---

- 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' )
```

---

- 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 interfata X?

---

```
tcpdump -n -i X
```

---

- Cum pot reprezenta o adresa IP in memorie?

---

```
uint32_t
```

---

- Cum afisez interfetele unui host?

---

```
ip a s
```

---

- Valorile primite pe retea nu coincid cu valorile la care ma asteptam, ce fac?

Pachetele primite sunt in format big endian, folositi urmatoarele functii pentru a face trecerea intre Little Endian si Big Endian.

---

```
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 initializez o coada?

---

```
queue q;  
q = queue_create();
```

---



- Cum adaug un element in coada?

---

```
queue_enq(q, packet);
```

---

- Am implementat totul si nu imi trec testele.

In cazul in care aplicati diferite operatii asupra tabeli de rutare la etapa de preprocesare, aveti grija ca acestea sa aiba o complexitate  $O(n \log n)$  deoarece testarea temei incepe dupa 2 sec de initializare.

- Cum determin adresa IP si MAC-ul unei interfete a routerului?

---

```
/* Intoarce adresa in format zecimal, e.g. "192.168.2.1" */
char *get_interface_ip(int interface);
/* Adresa e intoarsa prin parametrul de iesire mac; sunt intorsi ↔
   octetii. */
void get_interface_mac(int interface, uint8_t *mac);
```

---

- Nu văd output de la router (router\_output.txt este gol)

---

```
/* first instruction in main from router.c */
setvbuf(stdout, NULL, _IONBF, 0);
```

---

- Când rulez make primesc o eroare ce conține "cannot open output file router: No such file or directory".

---

```
$ make distclean
$ make
```

---

## References

- [1] [RFC 791 - Internet Protocol](#)
- [2] [RFC 826 - An Ethernet Address Resolution Protocol](#)
- [3] [RFC 894 - A Standard for the Transmission of IP Datagrams over Ethernet Networks](#)
- [4] [RFC 792 - Internet Control Message Protocol](#)
- [5] [RFC 1624 - Computation of the Internet Checksum via Incremental Update](#)