PROTOCOALE DE COMUNICAȚIE 1

Universitatea POLITEHNICA din București 27/03/2022

1 Prezentare generală

În cadrul temei vom implementa procesul de dirijare a pachetelor dintr-un router (“forwarding”). Un router are două părți:

- Data plane - partea care implementează procesul de dirijare, propriu-zis. Tema constă în implementarea acestei componente. În cele ce urmează, în lipsa altor precizări, toate referințele la router din textul temei se referă la data plane.
- Control plane - componenta care implementează algoritmii de rutare (e.g. RIP, OSPF, BGP); acești algoritmi distribuiți calculează rutele pentru fiecare rețea destinație și le inserează în data plane. **NU** va fi nevoie sa implementați acesti algoritmi pentru temă. Routerul va funcționa cu o tabelă de rutare statică, primită într-un fișier de intrare, și care nu se va schimba pe parcursul rulării.

Un router are mai multe interfețe și poate recepționa datagrame pe oricare dintre acestea. Routerul trebuie să transmită pachetul mai departe, către un calculator sau către alt router direct conectat, în funcție de regulile din tabela de rutare. Se cere implementarea procesului de dirijare al routerului.

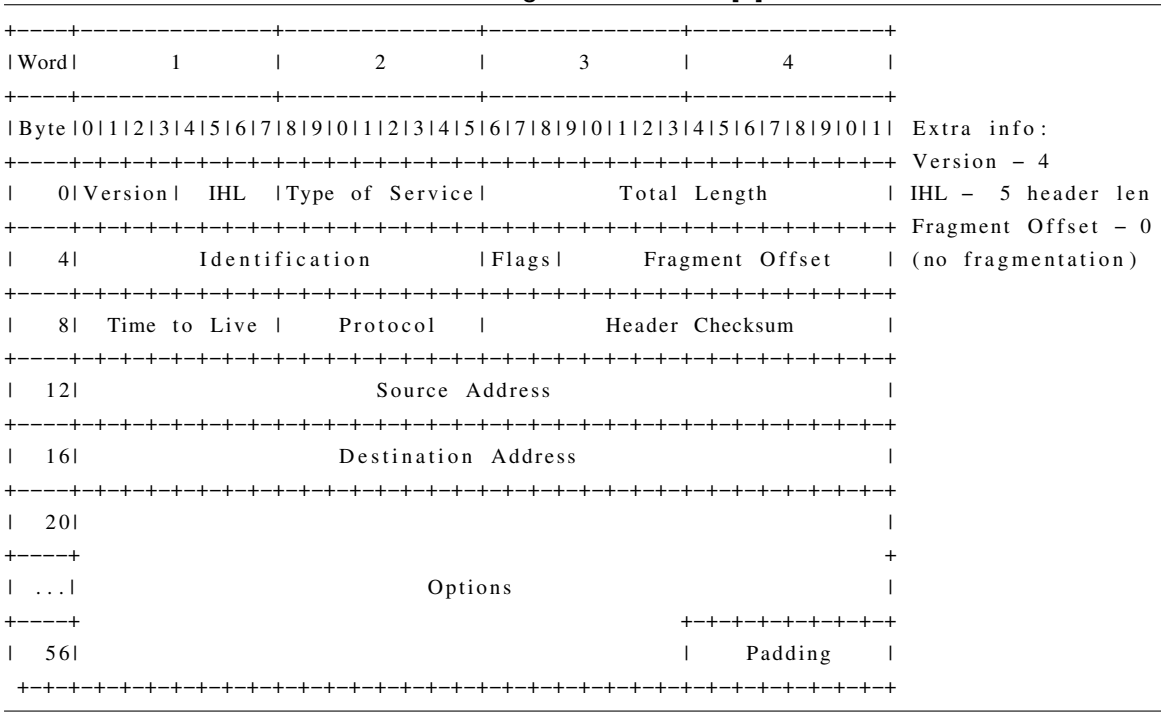
2 Procesul de dirijare (forwarding)

Dirijarea este un proces care are loc la nivelul 3 (“Rețea”) din stiva OSI. În momentul în care un pachet ajunge la router, acesta trebuie să efectueze următoarele acțiuni:

1. **Parsarea pachetului:** din fluxul de octeți ce reprezintă pachetul, routerul trebuie să afle ce antete sunt prezente și ce valori conțin câmpurile acestora, construind și inițializând structuri interne corespunzătoare. La acest pas, dacă routerul descoperă că un pachet primit e malformat (e.g. prea scurt), îl aruncă.
2. **Validarea L2:** pachetele conțin un antet de `Ethernet`; routerul nostru trebuie să considere doar pachetele trimise către el însuși (i.e. câmpul `MAC destination` este același cu adresa MAC a interfeței pe care a fost primit pachetul) sau către toată lumea (i.e. câmpul `MAC destination` este adresa de *broadcast*, `FF:FF:FF:FF:FF:FF`). Orice alt pachet trebuie aruncat.
3. **Inspectarea următorului header:** routerul inspectează câmpul `"ethertype"` pentru a descoperi următorul antet prezent. În cadrul temei, ne vor interesa doar două posibilități: IPv4 și ARP. Următoarele acțiuni ale routerului vor depinde de tipul acestui header. Orice alt fel de pachet trebuie ignorat.

2.1 Protocolul IPv4

Listing 1: Antet IPv4. [4]



Când un router primește un pachet de tip IPv4 trebuie să realizeze următoarele acțiuni:

1. **Verificarea pachete proprii:** deși un intermediar, routerul este de asemenea o entitate cu interfețe de rețea și adrese IP asociate acestora, deci poate fi destinatarul unui pachet. În acest caz, routerul nu trebuie să trimită mai departe pachetul, ci să îi înțeleagă conținutul pentru a putea acționa în consecință. În cadrul acestei teme, routerul nu trebuie să răspundă decât la mesaje de tip ICMP destinate lui.

2. **Verificare checksum:** routerul trebuie să recalculeze suma de control a pachetului, conform [1] și să o compare cu cea primită în antetul de IP; dacă sumele diferă, pachetul a fost corupt și trebuie aruncat.

3. **Verificare și actualizare** `TTL`**:** pachetele cu câmpul `TTL` având valoarea 1 sau 0 trebuiesc aruncate. Routerul va trimite înapoi, către emițătorul pachetului un mesaj ICMP de tip "Time exceeded" (mai multe detalii în secțiunea ICMP). Altfel, câmpul `TTL` e decrementat.

4. **Căutare în tabela de rutare:** routerul caută adresa IP destinație a pachetului în tabela de rutare pentru a determina adresa următorului hop, precum și interfața pe care va trebui scos pachetul. În caz că nu găsește nimic, pachetul este aruncat. Routerul va trimite înapoi, către emițătorul pachetului un mesaj ICMP de tip "Destination unreachable" (mai multe detalii în secțiunea ICMP).

5. **Actualizare checksum:** routerul recalculează suma de control a pachetului (această trebuie recalculată din cauza schimbării câmpului `TTL`) și o surpascrie în antetul IP al pachetului.

6. **Rescriere adrese L2:** pentru a forma un cadru corect care să fie transmis la următorul hop, routerul are nevoie să rescrie adresele de L2: adresa sursă va fi adresa interfeței routerului pe care pachetul

e trimis mai departe, iar adresa destinație va fi adresa MAC a următorului hop. Pentru a determina adresa următorului hop, routerul folosește protocolul ARP.

7. **Trimiterea noului pachet pe interfața corespunzătoare.**

2.2 Tabela de rutare

Fiecare router dispune de o "tabelă de rutare" – o structură pe baza căreia alege portul pe care să emită un pachet. În mod normal, aceste tabele sunt populate de către control plane, în urma rulării unui algoritm de rutare (e.g. OSPF); în cadrul temei, vom folosi o tabelă statică.

O intrare în tabel are patru coloane:

- `prefix` și `mask`: împreună aceste două câmpuri formează o adresă de rețea (e.g. `192.168.1.0/24`)
- `next hop`: adresa IP a mașinii către care va fi trimis pachetul; daca routerul nu e conectat direct la destinatar, aceasta va fi adresa unui router intermediar.
- `interface`: identificatorul interfeței pe care routerul va trebui să trimită pachetul respectiv către `next hop`.

Listing 2: Exemplu de Tabelă 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

În contextul temei, este suficient să considerați ca numarul maxim de intrări din tabela de rutare este de 100000 de intrări.

2.3 Longest Prefix Match

Pentru adresa IPv4 destinație din pachetul primit, routerul trebuie să caute intrările din tabela de routare care descriu o rețea ce cuprinde adresa respectivă. Este posibil ca mai multe intrări să se potrivească; în acest caz, routerul trebuie s-o aleagă pe cea mai specifică, i.e. cea cu masca cea mai mare. Acest criteriu de căutare se numește "Longest Prefix Match" (LPM).

Programatic, routerul poate verifica apartenența unei adrese la o rețea, făcând operația de `AND` pe biți și verificând că este egală cu prefixul:

```
ip.destination & entry.mask == entry.prefix
```

Odata ce routerul gaseste toate prefixele care se potrivesc adresei destinatie din pachet, va alege intrarea din tabela de rutare corespunzatoare prefixului cei mai lung (masca cea mai mare).

2.4 Protocolul ARP

Listing 3: Antet ARP [2]

```
0         7         15        23        31
+--------+--------+--------+--------+
|        HTYPE     |        PTYPE     |  HTYPE - Format of hardware address (1 for Ethernet)
+--------+--------+--------+--------+  PTYPE - Format of protocol address (2048 for IPV4)
|  HLEN  |  PLEN  |         OP        |  HLEN - Length of hardware address (6 for MAC)
+--------+--------+--------+--------+  PLEN - Length of protocol address (4 for IP)
|          SHA (bytes 0-3)           |  OP - ARP opcode (command, request or reply)
+--------+--------+--------+--------+  SHA - Sender hardware address
| SHA (bytes 4-5) | SPA (bytes 0-1) |  SPA - Sender IP address
+--------+--------+--------+--------+  THA - Target hardware address
| SPA (bytes 2-3) | THA (bytes 0-1) |  TPA - Target IP address
+--------+--------+--------+--------+
| THA (bytes 2-3) | THA (bytes 4-5) |
+--------+--------+--------+--------+
|          TPA (bytes 0-3)           |
+--------+--------+--------+--------+
```

După ce un router a determinat următorul hop pentru un pachet, trebuie să-l trimită mai departe, actualizând pachetul astfel încât adresa MAC destinație să fie cea a următorului hop. Cum știe routerul această adresă? O opțiune ar fi ca această să fie reținută static, într-un tabel.

În realitate, însă, din mai multe motive (e.g. flexibilitate), routerul *nu știe* aceste adrese, ci trebuie să le determine folosind protocolul ARP [2]. Având adresa IP a următorului hop precum și identificatorul interfeței care duce la acesta, routerul generează un mesaj de broadcast în rețeua respectivă, întrebând care e adresa MAC asociată acelei adrese IP. Mașina respectivă observă că este vorba de propriul său IP și îi trimite routerului un mesaj cu propria adresă MAC. Acum routerul poate forma cadrul corespunzător; deasemena, păstrează într-un cache adresa MAC primită, pentru un interval limitat de timp.

Protocolul ARP este deci relevant în două puncte: atunci când primim un pachet cu antet ARP și atunci când trebuie să rescriem pentru forwardare un pachet IPv4.

Pe lângă materialele de curs, va recomandăm următorul video (10 min) și capitolul 5.4.1 (Link-Layer Addressing and ARP) din Computer Networking: A Top-Down Approach (6th Edition)

Exemplu de pachet de tip ARP request capturat cu Wireshark:

```
Ethernet II, Src: Micro-St_57:ff:f6 (d8:bb:c1:57:ff:f6), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
   Hardware type: Ethernet (1)
   Protocol type: IPv4 (0x0800)
   Hardware size: 6
   Protocol size: 4
   Opcode: request (1)
   Sender MAC address: Micro-St_57:ff:f6 (d8:bb:c1:57:ff:f6)
   Sender IP address: 192.168.0.150
   Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
   Target IP address: 169.254.255.255
```

Pașii necesari pentru dirijarea unui pachet IPv4 sunt următorii:

1. **Căutare în cache:** routerul se uită în cache-ul său ARP pentru a vedea dacă există o intrare curentă pentru adresa IPv4 a următorului hop. Dacă aceasta există, routerul o ia din cache, rescrie antetul L2 al pachetului și îl trimite mai departe.

2. **Salvare pachet pentru mai târziu:** dacă adresa necesară nu se găsește în cache-ul ARP, routerul va trebui să facă o interogare generând un pachet ARP și așteptând un răspuns. Pachetul original

care trebuia dirijat este adăugat într-o coadă, pentru a putea fi trimis mai târziu, după sosirea răspunsului ARP.

3. **Generare ARP request:** routerul generează un pachet de tip ARP pentru a interoga despre adresa MAC a mașinii cu adresa IPv4 a următorului hop. Pentru asta are nevoie să genereze un pachet cu un antet Ethernet, urmat de un antet ARP.

   - **Antetul Ethernet:** trebuie să conțină un `ethertype` care să identifice un pachet de tip ARP (`0x806`). Adresa MAC sursă va fi adresa interfeței routerului către next hop; adresa MAC destinație va fi cea de broadcast (`FF:FF:FF:FF:FF:FF`).
   - **Antetul ARP:** trebuie să conțină tipul de adresă folosit în căutare (IPv4) împreună cu adresa în sine, precum și tipul de adresă căutată (MAC) împreună cu adresa în sine. Pentru mai multe detalii, consultați [2]

4. **Parseaza ARP reply.** Atunci cand routerul primeste un pachet de tip ARP reply, il va adauga in cache-ul ARP local. In plus, routerul va parcurge lista de pachete care asteapta raspunsuri ARP si le va trimite pe cele pentru care adresa urmatorului hop este cunoscuta.

2.5 Protocolul ICMP

Listing 4: Antet ICMP. [3]

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Depends on type and code                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Protocolul ICMP [3] este folosit de routere în procesul de dirijare pentru a transmite informatii legate de conectivitatea la nivel IP si transport gazdelor care au trimis datagrama in cauza. Protocolul ICMP se situeaza deasupra IP in stiva de protocoale, dar nu este un protocol de transport, ci un protocol de control (i.e. de debugging) pentru IP.

De exemplu, in anumite situații în care un pachet este aruncat de router, un mesaj ICMP este generat de router și trimis către expeditorul pachetului. În cadrul temei, ne vor interesa doar următoarele situații și mesaje:

- **Destination unreachable** (type 3, code 0) - Trimis în cazul în care nu există rută până la destinație, atunci când pachetul nu este destinat routerului.
- **Time exceeded** (type 11, code 0) - Trimis dacă pachetul este aruncat din cauza expirării câmpului `TTL`.

Pentru ambele tipuri de mesaj de eroare, pachetul emis de router trebuie să conțină, deasupra headerului ICMP, primii 64 de octeți din **payload-ul** pachetului original (adică doar ce se află deasupra antetului IPv4).

De asemenea, routerul fiind și el o entitate în rețea, poate primi mesaje ICMP de tip `"Echo request"` (type 8, code 0) destinate lui însuși. Acesta trebuie să răspundă cu un mesaj ICMP de tip `"Echo reply"` (type 0, code 0). În cazul acestor mesaje, octeții 4-5 din header capată semnificația unui "număr de identificare", iar octeții 6-7 ai unui "număr de secvență". Interpretarea acestora este de datoria hostului care a emis pachetele ICMP, routerul trebuie doar să se asigure că păstreze aceleași valori în pachetul de tip "Echo reply". Routerul trebuie să trimită înapoi și orice date care se aflau deasupra antetului ICMP în pachetul original.

Va recomandăm următorul video, Looking at ARP and ping packets, și capitolul 4.4.3 (Internet Control Message Protocol (ICMP)) din Computer Networking: A Top-Down Approach (6th Edition).

O captura de pachet ICMP request este urmatoarea:

```
▸ Ethernet II, Src: IntelCor_bb:1a:3e (f0:b6:1e:bb:1a:3e), Dst: Tp-LinkT_03:e1:a4 (68:ff:7b:03:e1:a4)
▸ Internet Protocol Version 4, Src: 192.168.0.106, Dst: 8.8.8.8
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xedd5 [correct]
    [Checksum Status: Good]
    Identifier (BE): 4 (0x0004)
    Identifier (LE): 1024 (0x0400)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 405190]
    Timestamp from icmp data: Mar 27, 2022 16:27:10.000000000 EEST
    [Timestamp from icmp data (relative): 0.821762257 seconds]
  ▸ Data (48 bytes)
```

3 Cerințe temă

Pentru rezolvarea temei, trebuie să implementați funcționalitățile routerului. Punctajul este împărțit în mai multe componente, după cum urmează:

- **Protocolul ARP (30p).** Tema se poate preda și cu o tabelă de ARP statică, cu pierderea punctajului aferent. În arhivă aveți inclusă o astfel de tabelă. **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, nu adăugați fișierul** `arp_table.txt`.

  Va trebui să implementați și funcționalitatea de caching; după ce primiți un răspuns ARP, rețineți adresa MAC a hostului interogat. În realitate, intrările dintr-un cache sunt temporare, fiind șterse după un anumit interval de timp. Pentru implementarea temei, este suficient să folosiți **intrări permanente**.

- **Procesul de dirijare (30p).** Va trebui să implementați pașii prezentați în secțiunea IPv4.

- **Longest Prefix Match eficient (15p).** La laborator, am implementat LPM folosind o căutare liniară, i.e. trecând de fiecare dată prin toate intrările din tabel. În practică, o tabelă de routare poate conține foarte multe intrări [1] – astfel de căutare este ineficientă.

  O abordare mai bună este să folosim fie un trie sau o căutare binară (într-o tabelă sortată după valoarea prefixului, apoi după lungimea măștii).

  Orice implementare mai eficientă decât căutarea liniară valorează 15 puncte.

- **Protocolul ICMP (20p).**

[1] https://blog.apnic.net/2022/01/06/bgp-in-2021-the-bgp-table/

Implementați funcționalitatea descrisă în secțiunea ICMP.

- **BONUS: actualizarea sumei de control incrementale (10p).**

  Singurul motiv pentru care routerul trebuie să recalculeze suma de control, este decrementarea câmpului `TTL`: astfel, pentru o mică schimbare apare un overhead pe care ne-am dori să-l evităm.

  RFC 1624 [5] descrie o metodă de actualizare rapidă a sumei de control IPv4, fără a o recalcula de la zero.

Restul de 5 puncte vor fi acordate, în urma corectării manuale, în funcție de claritatea codului și a fișierului README.

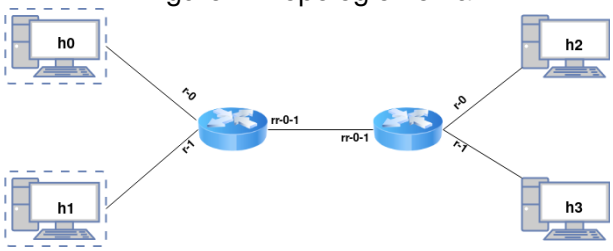
Atenție: Trebuie menționat în README ce subcerințe ați rezolvat.

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

4 Testare

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

Figure 1: Topologie Tema



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, câte un terminal pentru fiecare router și unul pentru controller (cu care nu vom interacționa); terminalele pot fi identificate după titlu.

Fiecare host e o simplă mașină Linux, din al cărei terminal puteți rula comenzi care generează trafic IP pentru a testa funcționalitatea routerului implementat. Vă recomandăm arping, ping și netcat. Mai mult, din terminal putem rula **Wireshark** sau tcpdump pentru a face inspectia de pachete.

Pentru a porni routerele manual folosim următoarele comenzi, prima pe router 0 și a doua pe router 1:

```
./router rtable0.txt rr-0-1 r-0 r-1
./router rtable1.txt rr-0-1 r-0 r-1
```

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`).

5 API și schelet

Pentru rezolvarea temei vă punem la dispoziție un schelet de cod care implementeze unele funcționalități esențiale pentru rezolvarea cerințelor, precum și unele funcții ajutătoare a căror utilizare este opțională. In **include/skel.h** veti gasi mai multi functii auxiliare utile, printre care:

- **Pachete:** un pachet e modelat de următoarea structură de date:

```
// "include/skel.h"
typedef struct {
    int len;
    char data[MAX_LEN];
    int interface;
} packet;
```

  Câmpul `data` conține toți octeții pachetului, inclusiv antetele, începând de la cel de L2 (i.e. Ethernet).

- **Structuri pentru antete:** pentru a inspecta și manipula valori din antetele pachetelor într-un mod convenabil, puteți folosi următoarele structuri de date pentru a modela antete:

```
// "include/skel.h"
struct arp_header;

// <net/ethernet.h>
struct ether_header;

// <netinet/ip.h>
struct iphdr;

// <netinet/ip_icmp.h>
struct icmphdr;
```

- **Recepționare/trimitere pachete:** aveți la dispoziție următoarele două funcții pentru a primi un pachet (interfața pe care a ajuns, sau pe care trebuie trimis pachetul este reținută în `m.interface`):

```
// "include/skel.h"
/* get_packet populates the structure m received as a pointer.
 * The function is blocking.
 */
int get_packet(packet *m);

/* Sends a packet, returns -1 if there was an error */
int send_packet(packet *m);
```

- **Intrări în tabela de routare:** puteți modela o intrare în tabela de routare folosind următoarea structură:

```
// "include/skel.h"
struct route_table_entry;
```

- **Parsare tabela de routare:** pentru a parsa tabela de routare, puteți folosi funcția:

```
// "include/skel.h"
int parse_rtable(const char *filepath, struct route_table_entry *rtable);
```

- **Intrări în tabela ARP:** puteți modela o intrare în tabela ARP folosind următoarea structură:

```
// "include/skel.h"
struct arp_table_entry;
```

- **Parsare tabela statică ARP:** în cazul în care doriți să folosiți tabela statică de ARP, puteți să o parsați folosind funcția:

```
// "include/skel.h"
void parse_arp_table();
```

- **Calcul sume de control:** pentru a realiza calcularea/verificarea sumelor de control din IPv4, respectiv ICMP, puteți folosi următoarele două funcții:

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

uint16_t icmp_checksum(uint16_t *buffer, size_t size);
```

  Urmăriți comentariile aferente din `include/skel.h`.

  Notă: funcțiile nu implementează checksum incremental. Pentru bonus va trebui să scrieți propria implementare.

- **Coadă de pachete:** după cum este menționat în descrierea protocolului ARP, veți avea nevoie

să folosiți o coadă pentru pachete. Vă punem la dispoziție implementarea unei cozi cu elemente generice; urmăriți comentariile din `include/queue.h`.

- **Determinarea MAC interfață proprie:** pentru a determina adresa MAC a unei interfețe a routerului, folosiți funcția: Argumentul `mac` trebuie să indice către o zonă de memorie cu cel puțin șase octeți alocați.

```
void get_interface_mac(int interface, uint8_t *mac);
```

6 Trimitere

Tema trebuie trimisă pe vmchecker v2, la această adresă. Pentru trimitere este necesară o arhivă zip cu numele de forma `Grupa_Nume_Prenume_Tema1.zip` care să conțină, în rădăcină, următoarele fișiere:

- `README`: în care să explicați pe larg abordarea implementării și problemele întâmpinate. În README va trebui să menționați ce puncte din cerință ați rezolvat. Deasemenea, fișierul trebuie să conțină, pe prima linie, numele și grupa.
- 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 temă! 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: Supraîncărcarea Vmchecker prin submisii succesive ale aceleiași versiuni ale temei, poate duce la depunctare.

Notă: Orice probleme de punctare a temelor de către Vmchecker, dacă există, vor fi soluționate de către echipa de corectare. Nu vor fi acordate extensii ale termenului dacă unele teste din Vmchecker sunt greșite; dacă găsiți astfel de teste, vă rugăm să le semnalați.

7 Deadline

Tema trebuie trimisă până pe data de 17 aprilie 2022, ora 23:55, pentru obținerea punctajului total de 100p (110p cu bonus).

Vă reamintim părțile relevante din regulamentul cursului de PCom:

- După expirarea acestui termen limita se mai pot trimite teme un interval de maxim 3 zile, cu următoarele depunctări: 10p în prima zi, 20p în a doua zi și 30p în a treia zi.
- După cele 3 zile tema nu se mai poate trimite.
- Oferim posibilitatea fiecărui student de a avea un număr de maxim **5 zile** numite “sleep days”.

- Aceste zile pot fi folosite pentru a amâna termenul de predare al temei de casă (fără penalizări).
- Nu se pot folosi mai mult de **două** sleep days pentru o temă de casă.
- Pentru a utiliza aceste zile, trimiteți un mesaj asistentului vostru.
- Temele de casă sunt *individuale*.

8 Dependințe temă

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` si `openvswitch-testcontroller`. Tema va fi rulată pe Linux (recomandam Ubuntu 18.04). Aici puteți găsi o mașină virtuală cu Ubuntu 18.04.

Listing 5: 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
```

9 FAQ & probleme

- **Q:** Nu îmi apar terminalele când rulez mininet.

  A: probabil nu aveți `xterm` instalat.

```
sudo apt install xterm
```

- **Q:** Cum pornesc mai mult de un terminal?

  A: rulați în background.

```
xterm &
```

- **Q:** Primesc următoarea eroare:

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

  A: în cazul in care portul este ocupat rulați sudo fuser -k 6653/tcp

- **Q:** 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' )
```

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

  A:

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

- **Q:** Cum pot vedea traficul de pe interfata X?

  A: Puteți folosi `tcpdump` din linia de comandă (adăugați `-XX` pentru a vedea un hexdump pentru fiecare pachet).

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

- **Q:** Cum pot reprezenta o adresa IP in memorie?

  A: `uint32_t`

- **Q:** Cum afisez interfetele unui host?

  A:

```
ip a s
```

- **Q:** Valorile observate într-o captură de rețea nu coincid cu cele emise.

  A: Cel mai probabil este o problemă de endianness. Câmpurile unor pachete în tranzit trebuie să fie în forma "Network Byte Order" (care este de fapt big endian), pe când mașinile voastre sunt foarte probabil little endian. Folosiți următoarele funcții pentru a obține endiannessul corect.

```
uint32_t htonl(uint32_t);     /* host to network long */
uint16_t htons(uint16_t);     /* host to network short */
uint32_t ntohl(uint32_t);     /* network to host long */
uint16_t ntohs(uint16_t);     /* network to host short */
```

- **Q:** Cum inițializez o coadă?

  A:

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

- **Q:** Cum adaug un element în coadă?

  A:

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

- **Q:** Am implementat tot și nu îmi trec testele.

  A:

  În cazul în care aplicați diferite operații asupra tabelei de rutare la etapa de preprocesare, aveți grijă ca acestea să aiba o complexitate $O(nlogn)$ deoarece testarea temei începe după două de inițializare.

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

  A:

```
/* 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);
```

- **Q:** Nu văd output de la router (`router_output.txt` este gol).

  A: Probabil routerul crapă înainte să dea flush la output; setați `stdout` să fie unbuffered.

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

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

  A:

```
$ make distclean
$ make
```

- **Q:** În WireShark văd “Internet Protocol, bogus version”.

  A: Completarea antetului IP/ICMP nu este corectă.

- **Q:** Putem folosi C++?

  A: Da.

- **Q:** Ce biblioteci sunt permise?

  A: Este permisă folosirea oricărei funcții din biblioteca standard de C sau C++. Mai mult, puteți folosi orice header standard de Linux, cu precizarea că trebuie în continuare să rezolvați manual cerința. De exemplu, Linux știe să răspundă singur la ICMP; pe router această funcționalitate este dezactivată de framework-ul de testare. Reactivarea ei cu apeluri din C nu reprezintă o soluție validă și nu va fi punctată.

- **Q:** Primesc eroarea “[14]: ioctl SIOCGIFINDEX No such device” când încerc să rulez pe mașina virtuală de Linux.

A: Routerul caută anumite interfețe după nume; acestea probabil nu există pe sistemul vostru, sunt create de mininet. Asigurați-vă că rulați binarul de router dintr-un terminal de router după pornirea topologiei.

- **Q:** Când încerc să accesez informația din antetul ICMP, în timp ce dau ping de la un host la altul, toate informațiile din header sunt 0.

  A: Spre deosebire de laborator, în temă trebuie implementat și protocolul ARP. În cazul de față, se încearcă extragerea antetului ICMP dintr-un pachet ce conține doar Ethernet + ARP.

- **Q:** Cum pot determina din cod adresa MAC a unei interfețe?

  A: Aveți o funcție ajutătoare `get_interface_mac`, care primește o interfață și-i întoarce adresa MAC. Interfața pe care a venit pachetul trebuie extrasă din structura `msg`.

- **Q:** La testul "timeout is unreachable" primesc urmatoarea eroare:

```
File "./checker.py", line 38, in passive
status = fn(testname, packets)
File "/media/sf_Shared_Folder/PC/tema1/tests.py", line 351, in icmp_timeout_p
assert ICMP in packets[1], "no ICMP packet from router"
```

  A: Nu trimiți înapoi un pachet de tip ICMP (probabil nu setați corect câmpul protocol din headerul IP).

- **Q:** Pe local am mai multe puncte decat pe vmchecker.

  A: Problema poate apărea atunci când implementarea voastră are o performanță scăzută (e.g. faceți sortare la fiecare apel de LPM). Pentru a rezolva problema, asigurați-vă că aveți o implementare bună din punct de vedere al performanței.

- **Q:** Îmi pică ultimele 2 teste.

  A: Ultimele două teste o să pice, în general, dacă realizați LPM ineficient.

References

[1] Network Working Group et al. *RFC 1071—Computing the Internet Checksum, Sep. 1988*.

[2] David C. Plummer. "An Ethernet Address Resolution Protocol". In: *IRTF, RFC 826 (E)* (1982).

[3] Jon Postel. "Internet Control Message Protocol darpa internet program protocol specification". In: *RFC 792* (1981).

[4] Jon Postel. "Internet protocol—DARPA internet program protocol specification, rfc 791". In: (1981).

[5] Anil Rijsinghani et al. *Computation of the internet checksum via incremental update*. Tech. rep. RFC 1624, May, 1994.