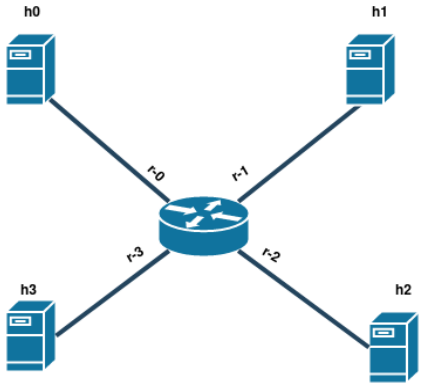
Lab 4 - Forwarding

Topologie

- Un router cu 4 interfete
  - r-0 → 192.168.0.1
  - r-1 → 192.168.1.1
  - r-2 → 192.168.2.1
  - r-3 → 192.168.3.1
- 4 gazde cu cate o interfata, conectate la router:
  - **h0** → 192.168.0.2 / fe80::0:2 / de:ad:be:ef:00:00
  - **h1** → 192.168.1.2 / fe80::1:2 / de:ad:be:ef:00:01
  - **h2** → 192.168.2.2 / fe80::2:2 / de:ad:be:ef:00:02
  - **h3** → 192.168.3.2 / fe80::3:2 / de:ad:be:ef:00:03



Scop

- Router-ul trebuie sa poata dirija pachete (venite pe oricare din cele patru interfete) catre destinatiile lor → *de exemplu, vreau sa trimit un pachet de la h0 la h2*
- Se foloseste de o tabela de rutare de forma:

Prefix	Masca	Next hop	Metrica	Interfata
192.168.2.0	255.255.255.0 (sau /24)	192.168.2.2	100	2
192.168.0.0	255.255.0.0 (sau /16)	…	…	…

- Router-ul are si o tabela de vecini (asociere adresa nivel 2 cu adresa nivel 3 MAC - IP):

Adresa IP	Adresa MAC
192.168.2.2	de:ad:be:ef:00:02
...	...

Pachete si cadre

- Pachete → unitati de date la nivelul retea (IP)
- Cadre → unitati de date la nivelul legatura de date (Ethernet)

ETH HEADER (2)	IP HEADER (3)	Date

- Header Ethernet (*net/ethernet.h*):

```
struct ether_header {
    u_char  ether_dhost[6]; // adresa MAC destinatie
    u_char  ether_shost[6]; // adresa MAC sursa
    u_short ether_type;     // protocolul de nivel 3 (ETHERTYPE_IP/ETHERTYPE_IPV6)
};
```

- Header IP (*linux/ip.h sau netinet/ip.h*)

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
    __u8  ihl:4,
          version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
    __u8  version:4,
          ihl:4;
#else
#error "Please fix <asm/byteorder.h>"
#endif
    __u8  tos;
    __u16 tot_len;
    __u16 id;
    __u16 frag_off;
    __u8  ttl;   // time to live
    __u8  protocol;
    __u16 check; // checksum-ul header-ului
    __u32 saddr; // adresa IP sursa
    __u32 daddr; // adresa IP destinatie
    /*The options start here. */
};
```

- Reprezentarea unei adrese IPv4 (*netinet/in.h*)

```
struct in_addr {
    unsigned long s_addr;
};
```

Simulator

- Mininet → scriem codul pentru router, apoi compilam si rulam:

```
$ make
$ sudo python topo.py
```

- La rulare, se pot da comenzi/porni terminale pentru cele 4 gazde si pentru router:

```
mininet> host0 ping 192.168.2.2
mininet> host0 xterm&
mininet> router xterm&
```

- Testare:

```
host0 ping -c1 h2
router cat /tmp/debug.txt
```

Logica router

- Primeste un pachet → `get_packet(...)`
- Verifica daca pachetul este IPv4 → face drop la pachet daca nu este, extrage header-ul IP daca este - **TODO #3**
    - `struct iphdr *ip_hdr = (struct iphdr *)(m.payload + sizeof(struct ether_header));`
- Verifica integritatea pachetului (nivelul 3) → `ip_checksum(...) == 0` - **TODO #4**
- Verifica TTL >= 1 (nivelul 3) - **TODO #5**
- Interogheaza tabela de rutare pentru a sti unde trimite pachetul - **TODO #1, #6**
    - longest prefix match

```
for each entry in routing_table
    if (entry.netmask & dest_IP) == entry.prefix
        && entry.mask is max
            && entry.metric is max
```

- Interogheaza tabela de vecini pentru a afla MAC-ul urmatorului hop - **TODO #2, #7**
    - cauta o intrare cu aceeasi adresa IP ca a urmatorului hop
- Actualizeaza header-ul de nivel 3 (IP) - **TODO #8**
    - decrementeaza TTL-ul
    - recalculeaza checksum pe pachet:
        - `ip_hdr->check = 0;`
        - `ip_hdr->check = ip_checksum(...);`
- Actualizeaza header-ul de nivel 2 (Ethernet) - **TODO #9**
    - copiaza adresa MAC a destinatiei in header-ul Ethernet
    - copieaza adresa MAC a sursei in header-ul Ethernet → `get_interface_mac(...)`
- Trimite pachetul mai departe → `send_packet(...)` - **TODO #10**.

Cheatsheet

- https://gitlab.cs.pub.ro/-/snippets/42

Link-uri

https://ocw.cs.pub.ro/courses/pc/laboratoare/04
https://forms.gle/XgUNG2S2oaWDnDNb6