Számítógépes Hálózatok GY – 9.hét

Laki Sándor
ELTE-Ericsson Kommunikációs Hálózatok Laboratórium
ELTE IK - Információs Rendszerek Tanszék
lakis@elte.hu
http://lakis.web.elte.hu

Teszt – 10 kérdés 10 perc

canvas.elte.hu

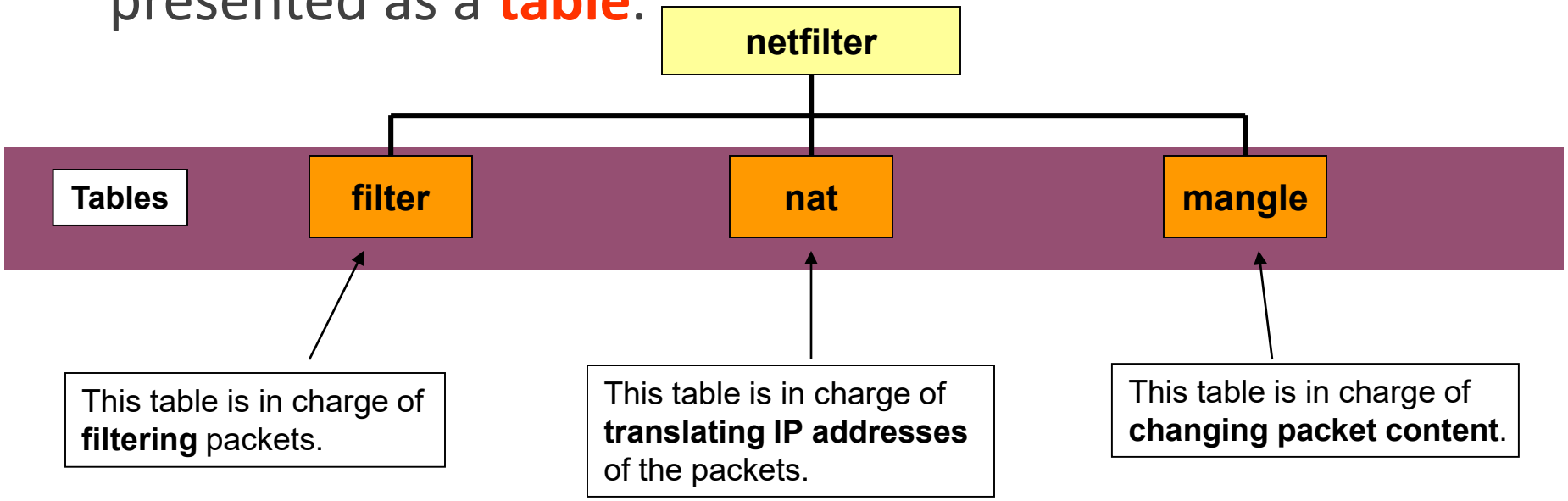
Kód:

iptables - topológia

- http://lakis.web.elte.hu/szh201819II/mininet/minitopo.mn
- http://lakis.web.elte.hu/szh201819II/mininet/minitopo.py

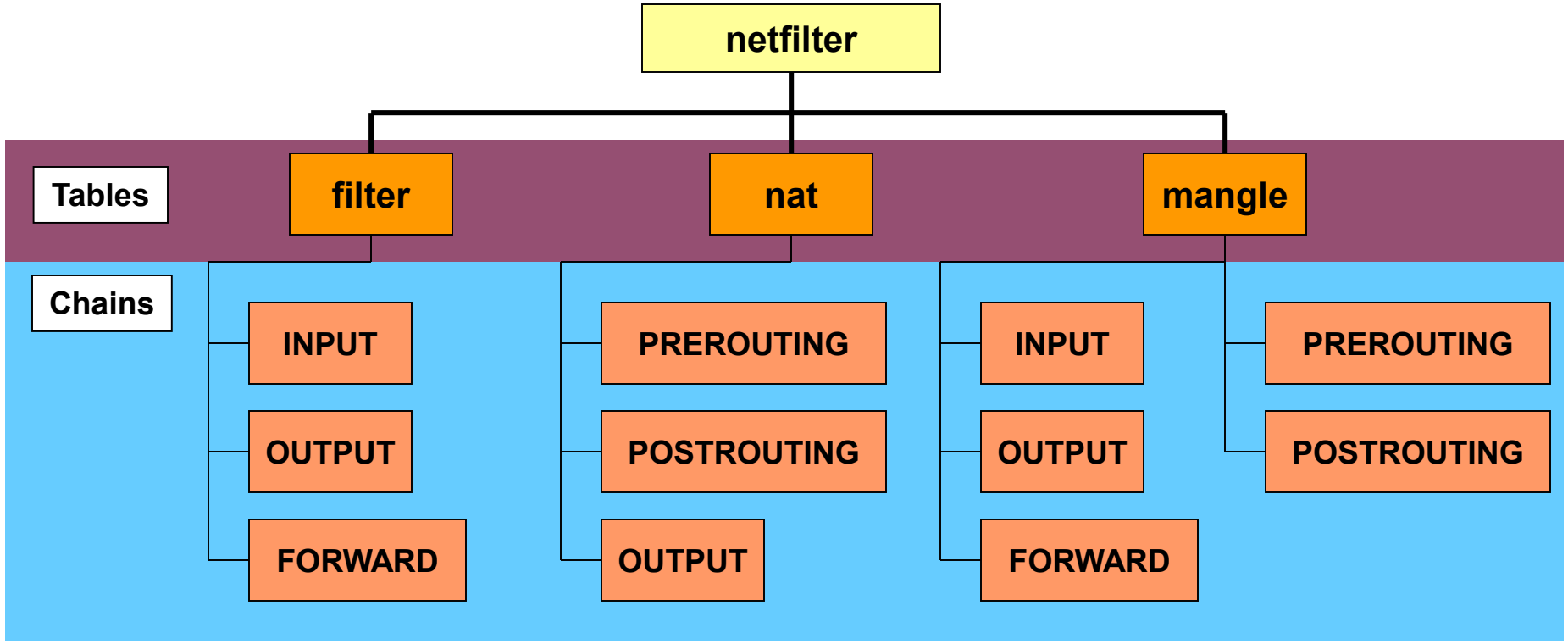
iptables – Tables and Chains

- Each function provided by the netfilter architecture is presented as a **table**.

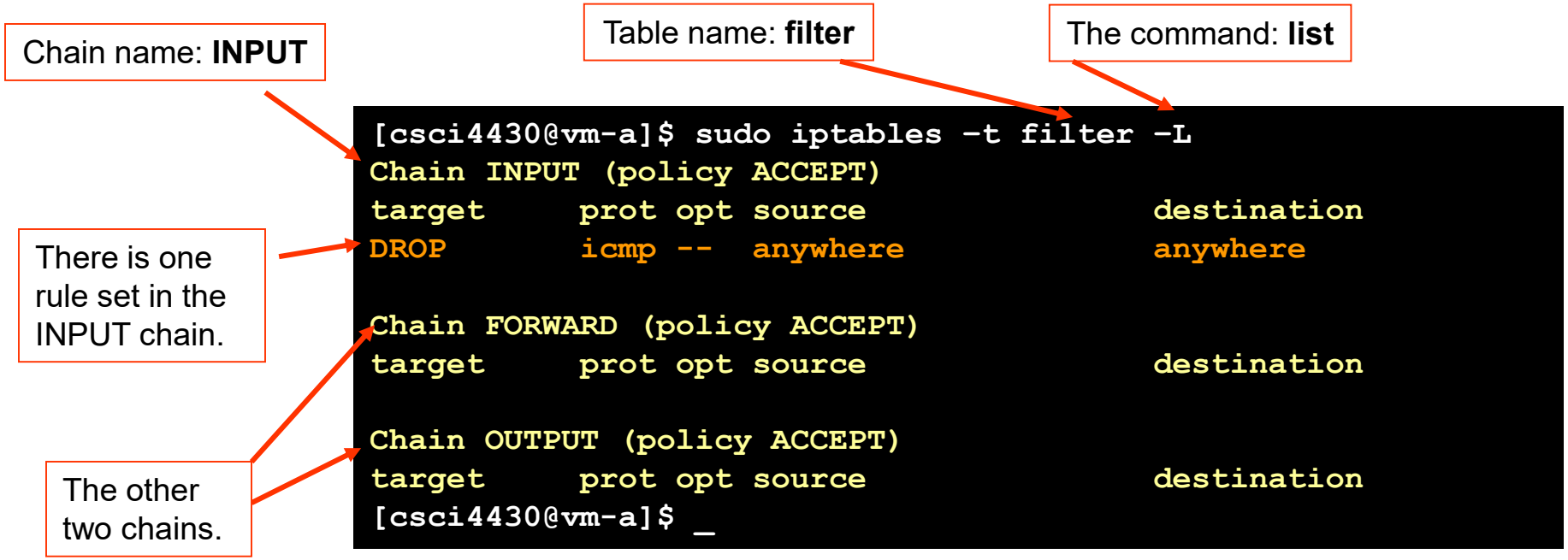


iptables – Tables and Chains

- Under each table, there are a set of **chains**.
  - Under each chain, you can assign a set of **rules**.



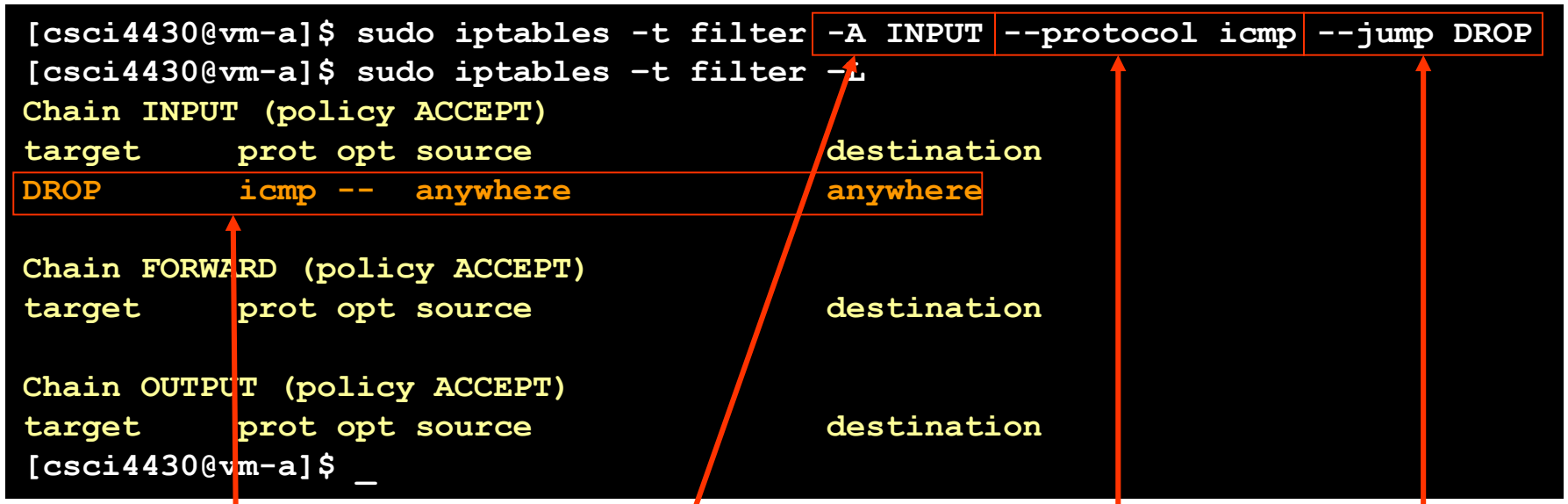
iptables – Tables and Chains



The rule in the INPUT chain means:

When a packet with ICMP payload passes through the **INPUT hook**, **DROP** that packets, no matter it is **from anywhere** and **to anywhere**.

iptables – Tables and Chains



This entry shows that a new rule is added to the INPUT chain of the filter table successfully.

Add a new rule to the INPUT chain.

The **protocol** of the packets in which this rule is interested is **ICMP**.

If a packet
(1) passes through the INPUT hook, and
(2) is an ICMP packet,

then the packet **jumps to the target DROP** – **to discard the packet**.

NAT Rules Set Up

- Your private network can "access" CUHK network and itself only.

```
[csci4430@vm-a]$ sudo iptables -t nat -A POSTROUTING \
                 -s 10.0.<vm_group_id>.0/24 -d 137.189.0.0/16 \
                 -j MASQUERADE
```

- Your private network can only **use SSH** to reach the outside world!

```
[csci4430@vm-a]$ sudo iptables -t nat -A POSTROUTING \
                 -p tcp -d ! 10.0.<vm_group_id>.0/24 --dport 22 \
                 -j MASQUERADE
```

iptables – More rules

- Clear all existing rules
  - Flush all entries in the filter table
    - ❖ `iptables -t filter -F`
  - Flush all entries in the nat table
    - ❖ `iptables -t nat -F`
  - Flush all entries in the mangle table
    - ❖ `iptables -t mangle -F`
- List all entries in the nat table
  - ❖ `iptables -t nat -L`
- Always take the manual for reference.

3. Feladat - tűzfalak

Először: iptables.pdf

A) ICMP tiltás

```
Iptables –t filter -I INPUT –p icmp –j DROP
```

B) TCP port forwarding

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to 172.31.0.23:80
```

```
iptables -A FORWARD -i eth0 -p tcp --dport 80 -d 172.31.0.23 -j ACCEPT
```

C) NAT – simple eth1 is connected to the Internet

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Mininet

- Nézzük meg a `test1.py`-t:

```
root@networks:/home/networks/ComputerNetworks/L2-switching# vim test1.py
```

  - Egy `LinuxBridge`-et definiálunk, amellyel futtatni tudjuk a feszítőfa algoritmust (Spanning Tree Protocol, STP) hurkok kezelésére
  - Hozzáadunk hosztokat is, privát IP címekkel
  - Végül összekötjük ezeket a topológia alapján
  - A `h1` és `s1` kapcsolat sávszélessége: 10 Mbps (alapból elvileg nem limitált, a TCLink osztály azért kell, hogy limitálni tudjuk)
- Indítsuk el:

```
root@networks:/home/networks/ComputerNetworks/L2-switching# python test1.py
mininet>
```

Mininet

- Elérhető csomópontok:

```
mininet> nodes
```

- Az s1 switchről infót kaphatunk
  - (brctl: ethernet bridge adminisztráció)

```
mininet> sh brctl show
```

  - Látszik, hogy nincs engedélyezve az STP
- A h1 h2 hostokon elindíthatunk egy-egy terminált:

```
mininet> xterm h1 h2
```

Mininet

- Itt lekérhetők az interface adatok, érdemes a mac címet megnézni!

```
# ifconfig
```

- Írassuk ki az ARP tábla aktuális tartalmát:

```
# arp
```

- Az s1 switch forwarding tábláját lekérdezhetjük a mininet konzolban:

```
mininet> sh brctl showmacs s1
```

Mininet

- Figyeljük a forgalmat minden interfészen! mininet konzolba írva:

```
mininet> s1 tcpdump -n -i any
```

- Pingetés xterm ablakból - pl. h1 termináljából: (a h1 h2 nevek itt nem használhatók!)

```
# ping 10.0.0.2
```

- Írassuk ki az ARP tábla aktuális tartalmát:

```
# arp
```

Mininet

- Közben látjuk a mininet konzolban, hogy mentek ARP üzenetek
- Pingetés mininet konzolból, pl.:

```
mininet> h1 ping h2
```

- Kilépés:

```
mininet> exit
```

Mininet

- Indítsuk el a miniedit-et:

```
root@networks:/home/networks# mininet/examples/miniedit&
```

  - Nyissuk meg a `sw-topo.mn` fájlt
  - Hurkot tartalmaz!
- Indítsuk el:

```
root@networks:/home/networks/ComputerNetworks/L2-switching# python sw-topo.py
mininet>
```

Mininet

- Nézzük meg a switcheket a mininet konzolban:

```
mininet> sh brctl show
```

- STP mindenhol ki van kapcsolva!
- h1 és h2 szomszédok

```
mininet> h1 ping h2
```

- Azt tapasztaljuk, hogy nagy a késés és csak néhány csomag megy át
- h1 és h4 távol vannak egymástól

```
mininet> h1 ping h4
```

- Csak sikertelen próbálkozás lesz, semmi se megy át

Mininet

- `tcpdump`-pal érdekes jelenség látható:

```
mininet> sh tcpdump -n -i any
```

- Multicast üzenetek próbálják a hálózatot felderíteni
- Konklúzió: hurok van a hálózatban, nem igazán működik semmi
- Kilépés:

```
mininet> exit
```

Mininet

- Indítsuk el újra `--stp` kapcsolóval:

```
root@networks:/home/networks/ComputerNetworks/L2-switching# python sw-topo.py --stp
mininet>
```

- bridge állapot:

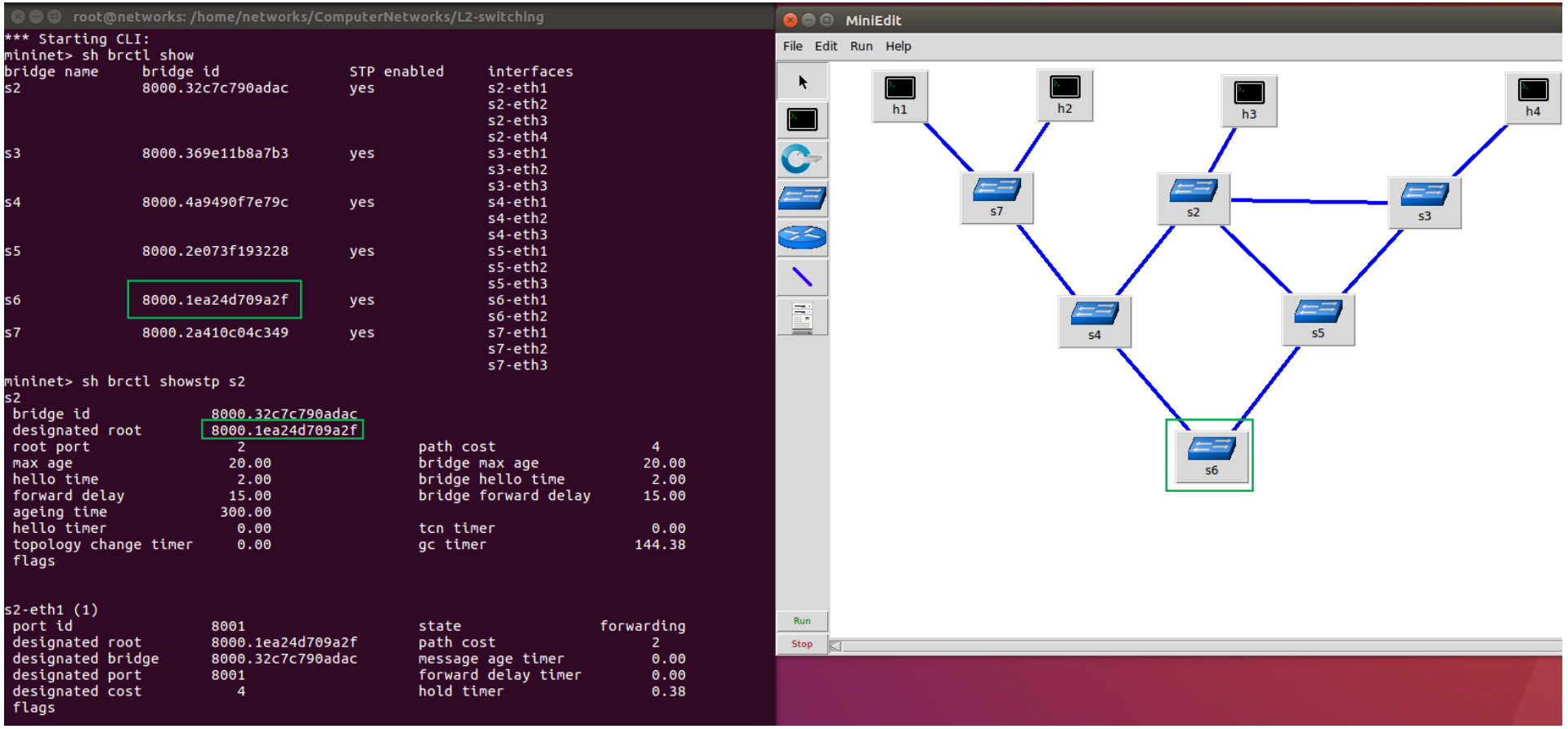
```
mininet> sh brctl show
```

- STP információ az `s2` switchhez:

```
mininet> sh brctl showstp s2
```

  - Nézzük meg mit ír ki: ki a designated root, ki a designated bridge, mely portok blokkoltak (a körök kiszűrésére)?

Mininet


root@networks: /home/networks/ComputerNetworks/L2-switching
*** Starting CLI:
mininet> sh brctl show
bridge name bridge id STP enabled interfaces
s2 8000.32c7c790adac yes s2-eth1
s2-eth2
s2-eth3
s2-eth4
s3 8000.369e11b8a7b3 yes s3-eth1
s3-eth2
s3-eth3
s4 8000.4a9490f7e79c yes s4-eth1
s4-eth2
s4-eth3
s5 8000.2e073f193228 yes s5-eth1
s5-eth2
s5-eth3
s6 8000.1ea24d709a2f yes s6-eth1
s6-eth2
s7 8000.2a410c04c349 yes s7-eth1
s7-eth2
s7-eth3
mininet> sh brctl showstp s2
s2
bridge id 8000.32c7c790adac
designated root 8000.1ea24d709a2f
root port 2 path cost 4
max age 20.00 bridge max age 20.00
hello time 2.00 bridge hello time 2.00
forward delay 15.00 bridge forward delay 15.00
ageing time 300.00
hello timer 0.00 tcn timer 0.00
topology change timer 0.00 gc timer 144.38
flags
s2-eth1 (1)
port id 8001 state forwarding
designated root 8000.1ea24d709a2f path cost 2
designated bridge 8000.32c7c790adac message age timer 0.00
designated port 8001 forward delay timer 0.00
designated cost 4 hold timer 0.38
flags
MiniEdit
File Edit Run Help
h1
h2
h3
h4
s7
s2
s3
s4
s5
s6
Run
Stop

Vége
Köszönöm a figyelmet!