Telekommunikációs Hálózatok

10. gyakorlat

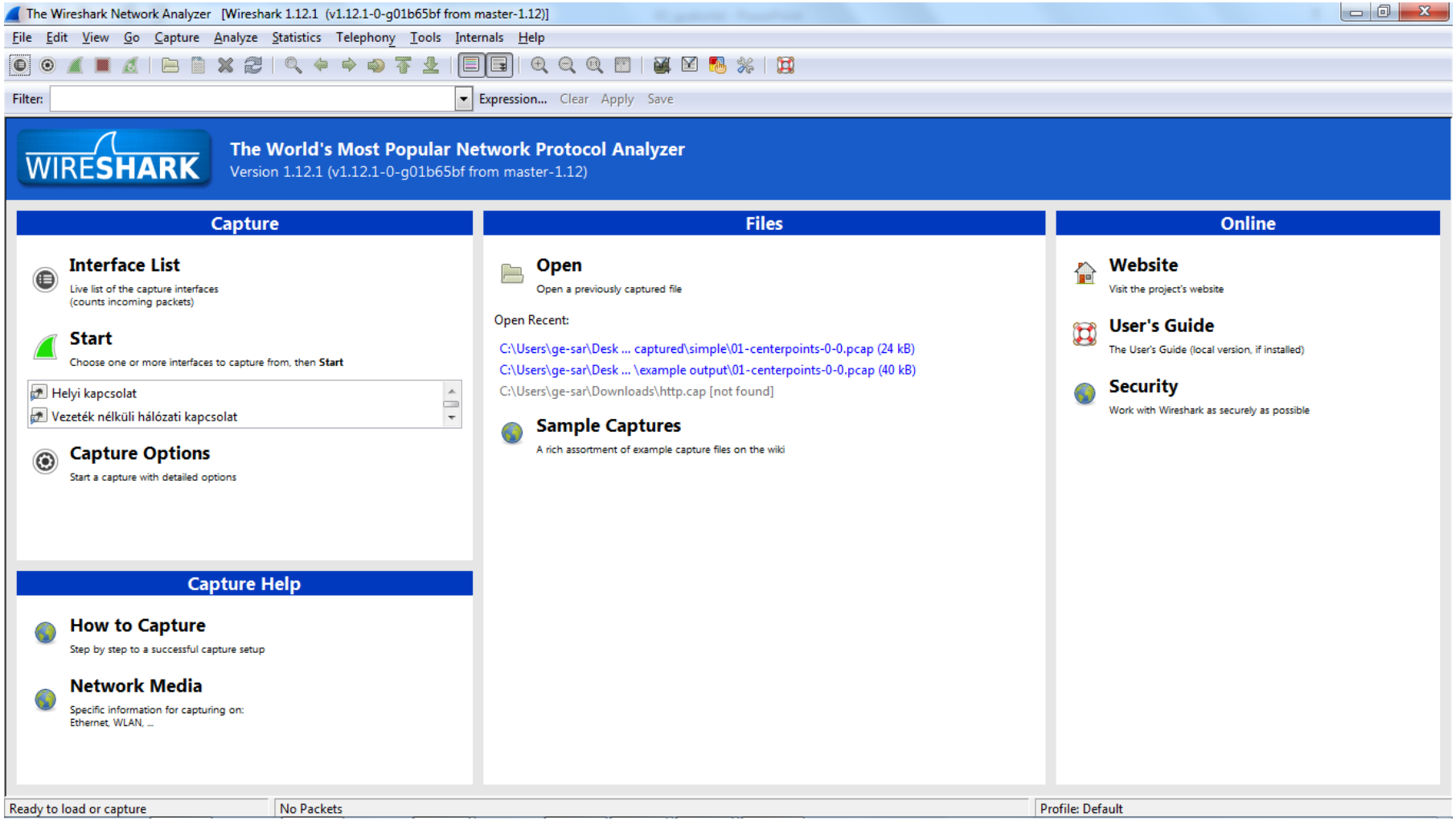
pótZH időpont!!!

- **December 11.** pótzárthelyi, az utolsó gyakorlat helyén és idejében

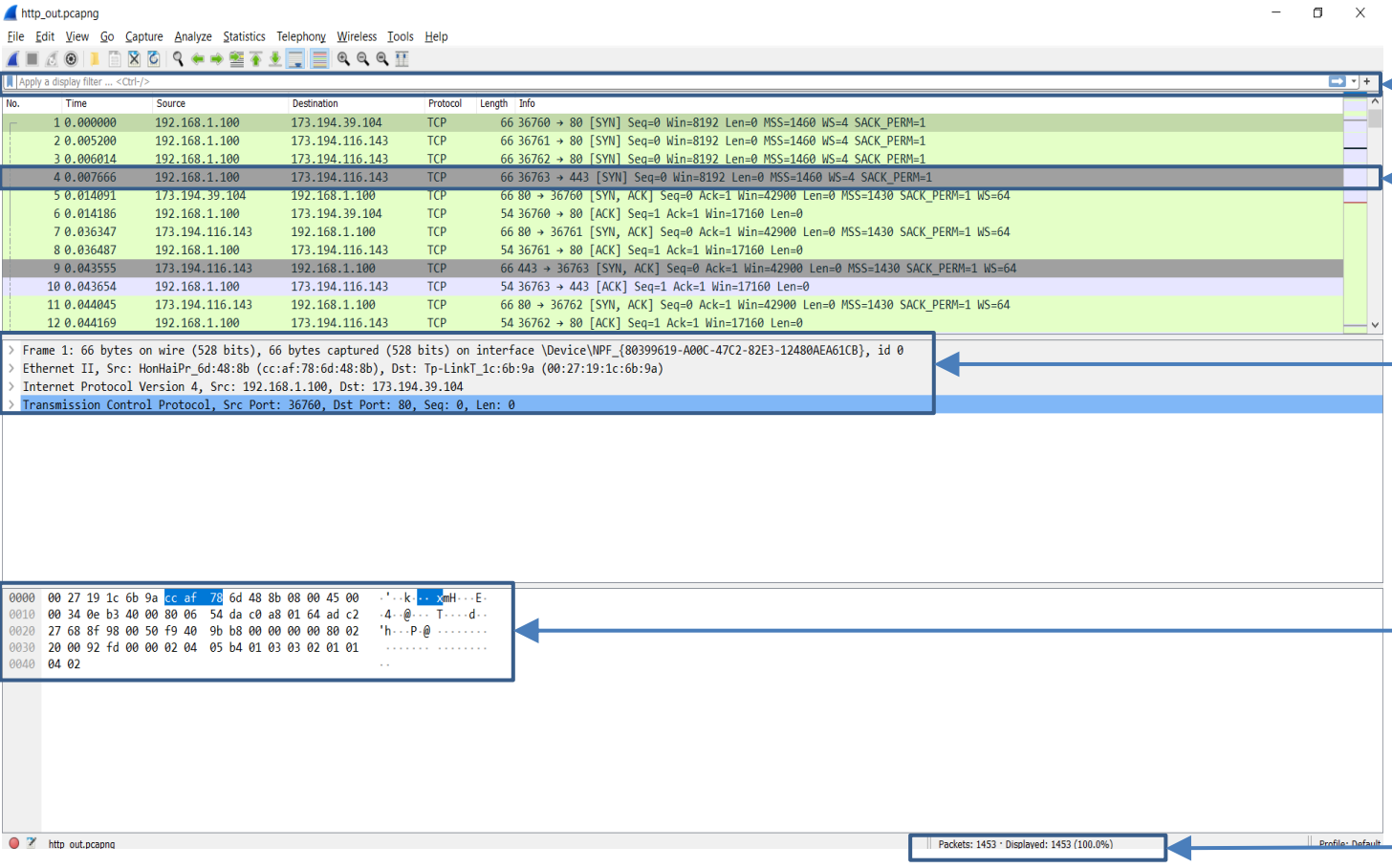
Wireshark

- Forgalomelemző eszköz: korábban rögzített adatok elemzésére szolgál
- Windows-on és Linux-on is elérhető
- www.wireshark.org

Wireshark


The Wireshark Network Analyzer [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Version 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)
Capture
Interface List
Live list of the capture interfaces
(counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
Helyi kapcsolat
Vezeték nélküli hálózati kapcsolat
Capture Options
Start a capture with detailed options
Capture Help
How to Capture
Step by step to a successful capture setup
Network Media
Specific information for capturing on:
Ethernet, WLAN, ...
Files
Open
Open a previously captured file
Open Recent:
C:\Users\ge-sar\Desk ... captured\simple\01-centerpoints-0-0.pcap (24 kB)
C:\Users\ge-sar\Desk ... \example output\01-centerpoints-0-0.pcap (40 kB)
C:\Users\ge-sar\Downloads\http.cap [not found]
Sample Captures
A rich assortment of example capture files on the wiki
Online
Website
Visit the project's website
User's Guide
The User's Guide (local version, if installed)
Security
Work with Wireshark as securely as possible
Ready to load or capture
No Packets
Profile: Default

Wireshark ablakok



Szűrők definiálására alkalmas input eszközök

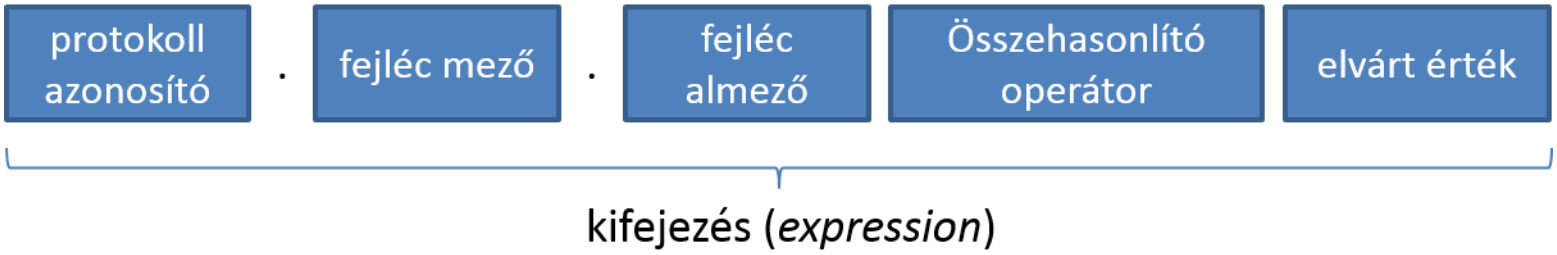
Csomag összefoglaló nézete

Kiválasztott csomag hierarchikus nézete

Kiválasztott csomag bájt-alapú nézete

Szűrés statisztikái

Wireshark szűrők



- Operátorok: or, and, xor, not
- protokollok: ip, tcp, http… (teljes listát lásd → Analyze→Display filter expression…)
- Példa: tcp.flags.ack==1 and tcp.dstport==80 (tcp nyugta flag és fogadó port beállítva)

Wireshark példa

- A **http_out.pcapng** állomány felhasználásával válaszoljuk meg az alábbi kérdéseket:

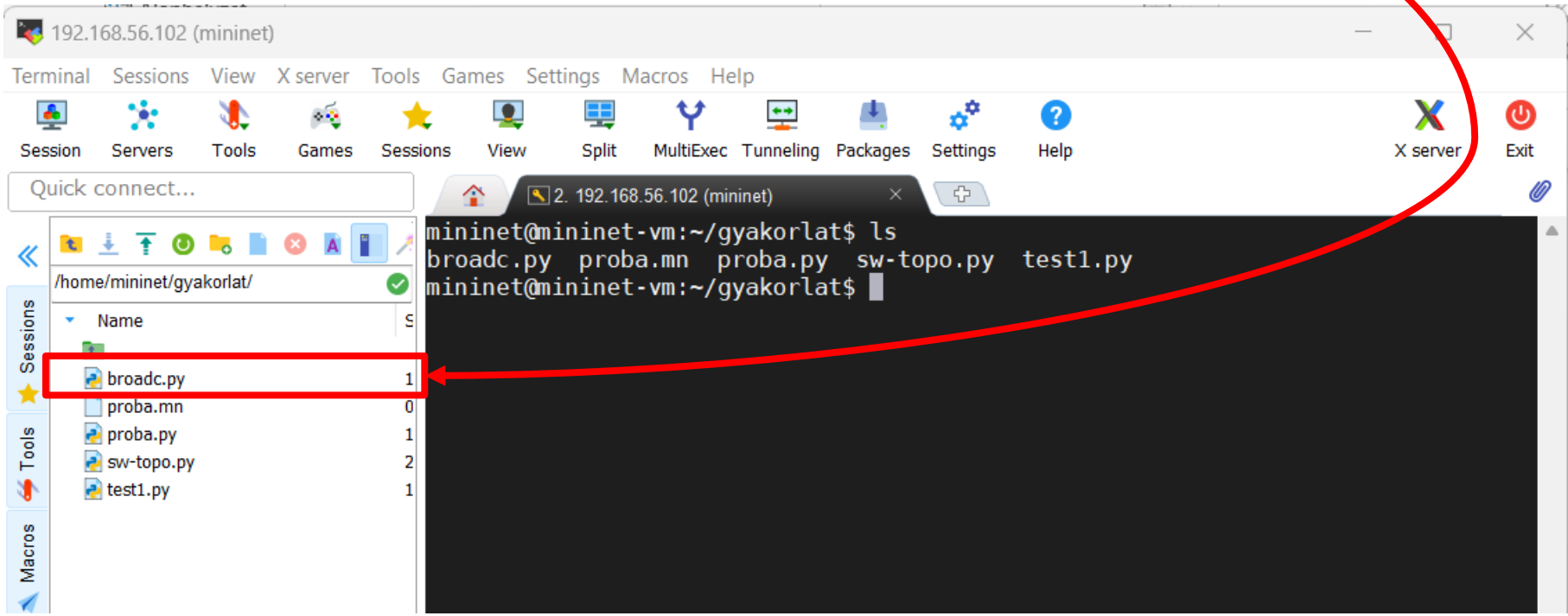
1. Milyen oldalakat kértek le a szűrés alapján HTTP GET metódussal? Milyen böngészőt használtak hozzá?
2. Hány darab képet érintett a böngészés? (Segítség: *webp.*)
3. Volt-e olyan kérés, amely titkosított kommunikációt takar? (Segítség: *SSL/TLS.*)

MININET

Előfeltétel: *Mininet beállítás.pdf* diasoron végigmenni!

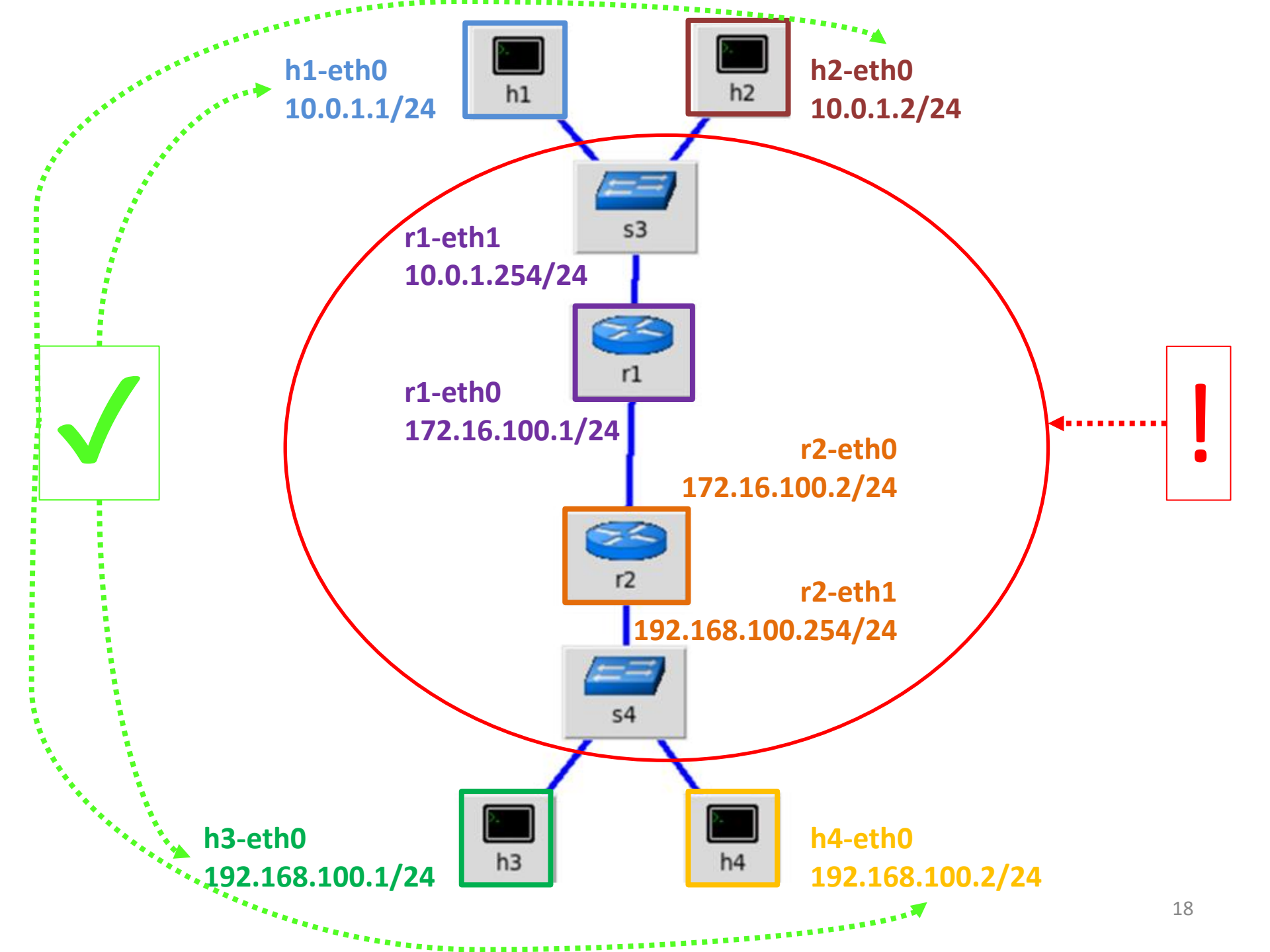
Feladat 1

- Belépés után másoljuk be a MobaXTerm-be a ***broadc.py*** szkriptet (oda kell húzni a fájlt):



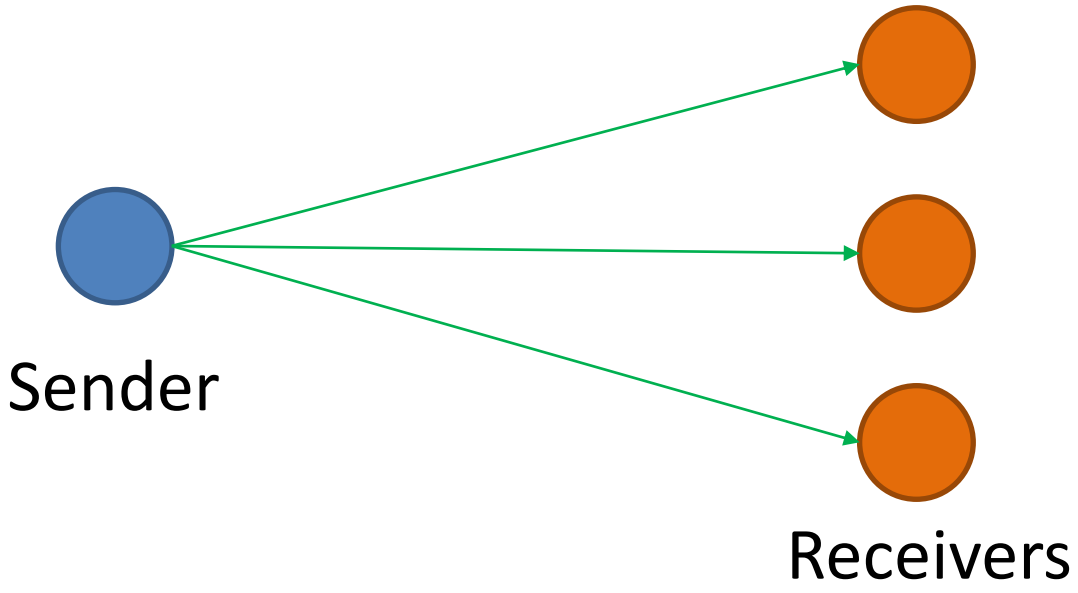
Feladat 1

- A következő dián látható egy ábra a szkript alapjául szolgáló topológiáról
- A h1-eth0, h2-eth0, h3-eth0 és h4-eth0 interfészekhez már beállítottuk az IP címeket
- Állítsuk be az r1-eth0, r1- eth1, r2-eth0 és r2-eth1 interfészekhez kapcsolódó IP címeket az ábra alapján!
- Állítsuk be az alapértelmezett útvonalakat a hosztoknál és router-eknél:
  - h1 és h2 hosztok esetében a 10.0.1.254 lokális átjárón keresztül,
  - h3 és h4 hosztok esetében a 192.168.100.254 lokális átjárón keresztül,
  - r1 router esetében a 172.16.100.2 lokális átjárón keresztül, amelyet az r1-eth0 eszközön lehet elérni,
  - r2 router esetében a 172.16.100.1 lokális átjárón keresztül, amelyet az r2-eth0 eszközön lehet elérni!
- Ellenőrizzük le, hogy a ping működik-e h1 és h4 között!


h1-eth0
10.0.1.1/24
h1
h2
h2-eth0
10.0.1.2/24
s3
r1-eth1
10.0.1.254/24
r1
r1-eth0
172.16.100.1/24
r2-eth0
172.16.100.2/24
r2
r2-eth1
192.168.100.254/24
s4
h3
h4
h3-eth0
192.168.100.1/24
h4-eth0
192.168.100.2/24
!

BROADCAST

Broadcast



- A broadcast egy időben több végpontnak is tudja szállítani az üzenetet → jobb hatékonyság

Broadcast

- A broadcast üzenetek küldésénél is **UDP-t** használunk
  - (a TCP végpontok közötti kommunikációs csatornát igényel)
- Egy IPv4 cím van lefoglalva a broadcast-ra:
  - egy adott alhálózatnál a csupa 1-es bitből álló hoszt azonosító
  - Pl. broadcast IP cím a h1, h2 és r1 alhálózatnál: 10.0.1.255

Broadcast – python3 socket

- A broadcast üzenetekhez a **setsockopt** függvény segítségével be kell állítani az alábbit (a küldő oldalon mindenképpen):

```
sock = socket.socket(type=socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
```

- A küldő az alhálózatnak megfelelő broadcast IP címre fogja küldeni a csomagját egy olyan port-ra (pl. 5005), amelyhez a fogadók előzetesen hozzákötötték (**bind** fv.) magukat
- Tehát fontos, hogy a fogadók socket-jénél az alábbi legyen (feltéve, hogy a port 5005):

```
sock.bind('10.0.1.255', 5005)
```

Feladat 2

- Készítsünk egy broadcast küldő és fogadó alkalmazást és próbáljuk is ki az előző mininet-es hálózaton, pontosabban annak h1, h2 és r1 alhálózatán!
- Egyszerű „Hello world!” legyen az üzenetszórásnál
- Lehet úgy is, hogy az elkészített szkripteket egy szövegszerkesztőben vagy fejlesztői környezetben hozzuk létre a „gazda” gépünkön, majd bemásoljuk a MobaXTerm-be
- A mininet-es hálózatot elindítva és „xterm” paranccsal a h1, h2 és r1-en terminálokat nyitva a „python3 feladat2_sender.py” és „python3 feladat2_receiver.py” parancsokkal tudjuk futtatni


"Node: r1"@mininet-vm
root@mininet-vm:~/gyakorlat# ls
broadc.py feladat2_receiver.py proba.mn sw-topo.py
feladat1.txt feladat2_sender.py proba.py test1.py
root@mininet-vm:~/gyakorlat# python3 feladat2_sender.py
root@mininet-vm:~/gyakorlat#
"Node: h1"@mininet-vm
root@mininet-vm:~/gyakorlat# python3 feladat2_receiver.py
b'Hello world' ('10.0.1.254', 50450)
root@mininet-vm:~/gyakorlat#
"Node: h2"@mininet-vm
root@mininet-vm:~/gyakorlat# python3 feladat2_receiver.py
b'Hello world' ('10.0.1.254', 50450)
root@mininet-vm:~/gyakorlat#

VÉGE
KÖSZÖNÖM A FIGYELMET!