Számítógépes Hálózatok Péntek

6. gyakorlat

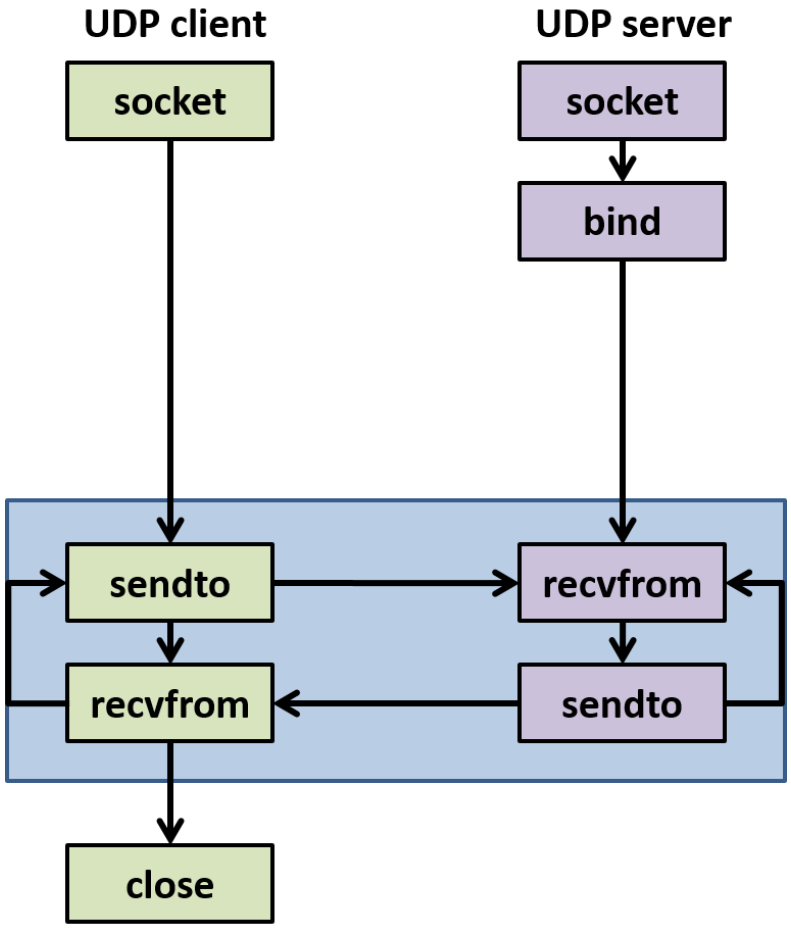
PYTHON SOCKET - UDP

Ismétlés

A kommunikációs csatorna kétféle típusa

- Kapcsolat-orientált modell (analógia: telefonbeszélgetés)
  - csomagok megérkeznek jó sorrendben
  - ilyen protokoll a TCP
  - kapcsolódó típus: stream socket
- **Kapcsolat-nélküli modell (analógia: postai levelezés)**
  - **csomagok nem biztos, hogy sorrend helyesen érkeznek, sőt el is veszhetnek**
  - **előnye a jobb teljesítmény**
  - **ilyen protokoll a UDP**
  - **kapcsolódó típus: datagram socket**

UDP

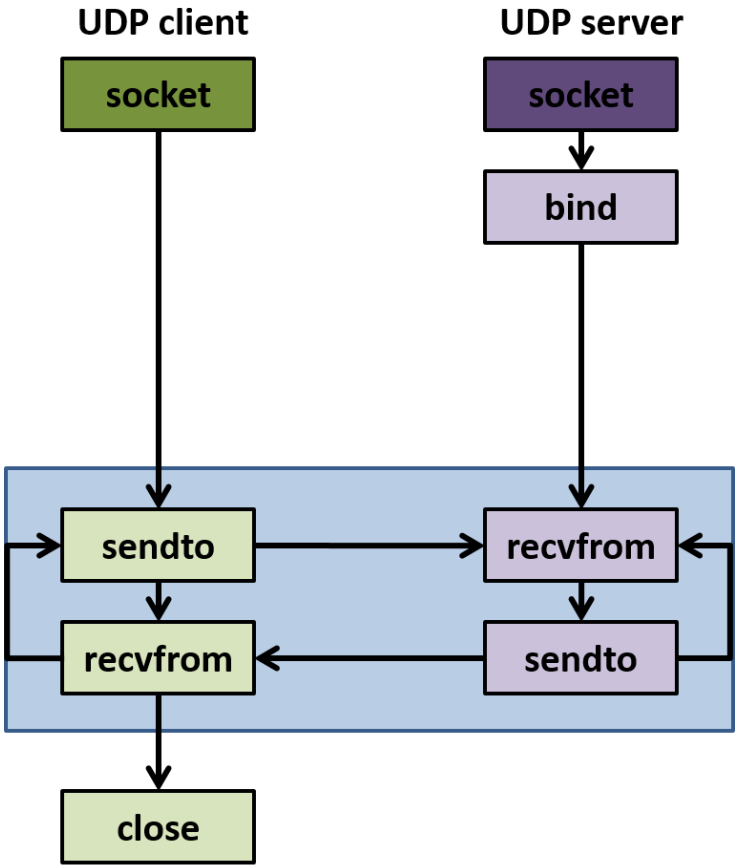

UDP client
UDP server
socket
socket
bind
sendto
recvfrom
recvfrom
sendto
close

Socket leíró beállítása

- socket.**socket**([*family*
  [, *type*
  [, *proto*]]])
- *family* : socket.AF_INET → IPv4 (AF_INET6 → IPv6)
- ***type* : socket.SOCK_DGRAM → UDP**
- *proto* : 0

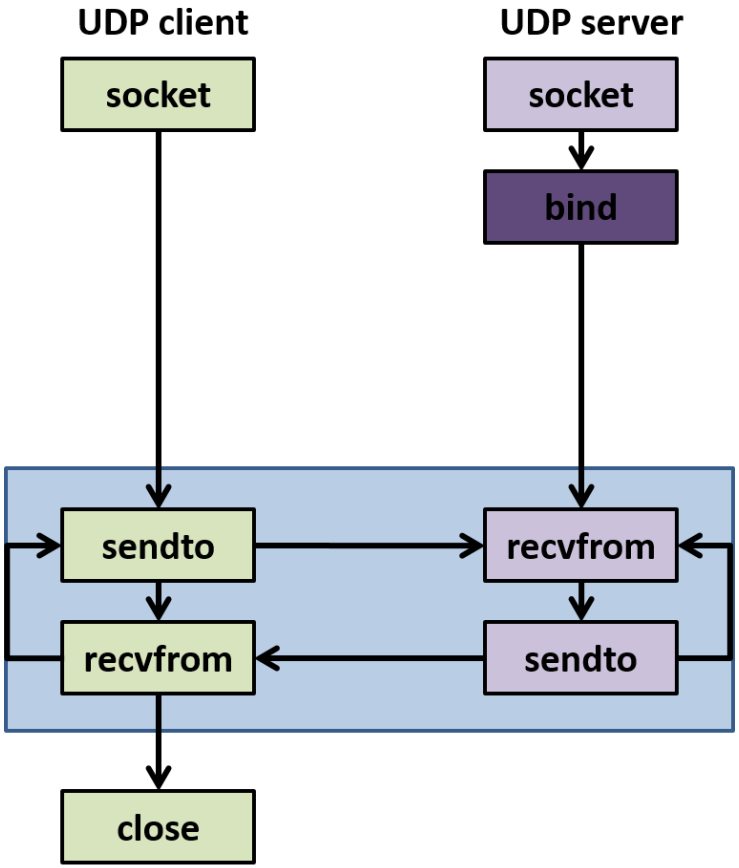
(alapértelmezett protokoll lesz)

- visszatérési érték: egy socket objektum, amelynek a metódusai a különböző socket rendszer hívásokat implementálják



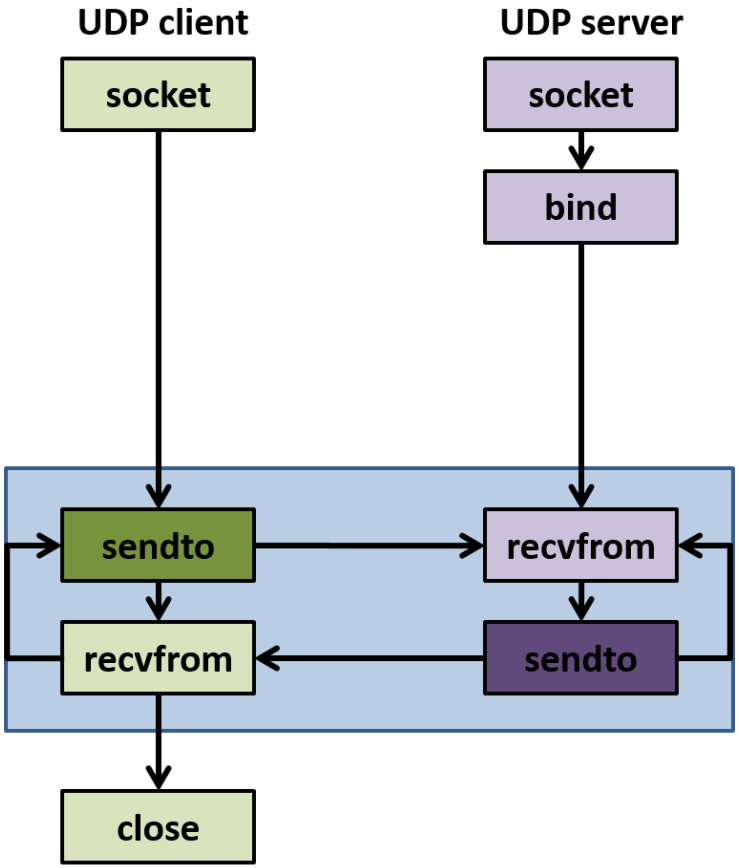
Bindolás – ismétlés

- socket.socket.**bind**(*address*)
- A socket objektum metódusa
- *address* : egy tuple, amelynek az első eleme egy hosztnév vagy IP cím (sztring reprezentációval), második eleme a portszám



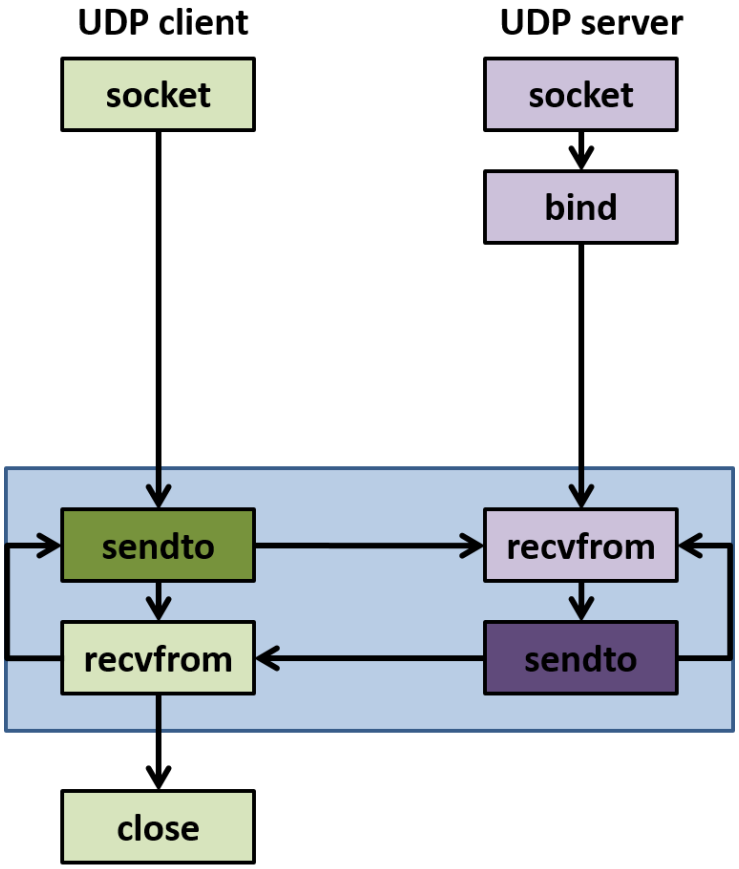
sendto

- socket.socket.**sendto**(*bytes* , ***address***)
- socket.socket.**sendto**(*bytes* , *flags* , ***address***)
- A socket objektum metódusai
- Adatküldés (*bytes*) a socketnek
- *flags* : 0 (nincs flag meghatározva)
- **A socketnek előtte nem kell csatlakozni a távoli sockethez, mivel azt az *address* meghatározza**



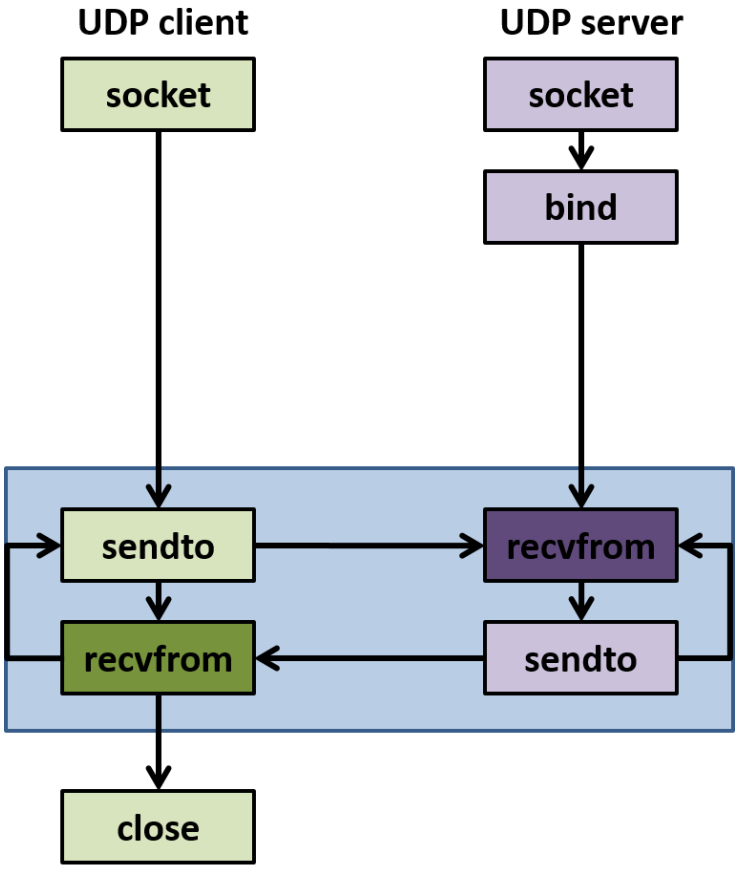
sendto

- **Fontos, hogy egy UDP üzenetnek bele kell férni egy egyszerű csomagba (ez IPv4 esetén kb. 65 KB-ot jelent)**
- visszatérési érték: az átküldött bájtok száma
  - az alkalmazásnak kell ellenőrizni, hogy minden adat átment-e
  - ha csak egy része ment át: újra kell küldeni a maradékot



recvfrom

- socket.socket.**recvfrom**(*bufsize [, flags]*)
- A socket objektum metódusa
- Üzenet fogadása
- *bufsize* : a max. adatmennyiség, amelyet egyszerre fogadni fog
- *flags* : 0 (nincs flag meghatározva)
- **visszatérési érték: egy (*bytes*, *address*) tuple, ahol a fogadott adat bytes reprezentációja és az adatküldő socket címe szerepel**



UDP

- socket

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

- recvfrom()

```
data, address = sock.recvfrom(4096)
```

- sendto()

```
sent = sock.sendto(data, address)
```

Feladat 1

Készítsünk egy kliens-szerver alkalmazást, amely UDP protokollt használ. A kliens küldje a ‚Hello Server’ üzenetet a szervernek, amely válaszolja a ‚Hello Kliens’ üzenetet.

Nézzük meg a megoldást!

Feladat 2 - Számológép UDP felett

Készítsünk egy szerver-kliens alkalmazást, ahol a kliens elküld 2 számot és egy operátort a szervernek, amely kiszámolja és visszaküldi az eredményt. A kliens üzenete legyen struktúra. Használjunk UDP protokollt!

Nézzük meg a megoldást!

Feladat 3 – fájlátvitel UDP felett

Fájlátvitel megvalósítása úgy, hogy a fájl letöltése UDP felett legyen megoldva. Készüljünk fel arra, hogy az átvitel során csomagvesztés, vagy sorrend csere is történhet! Az UDP szerver portját szabadon definiálhatjuk!

A hibakezeléshez egy javaslat:

Max. 1000 bájtonként UDP csomagokban elkezdjük átküldeni a fájl tartalmát. Minden csomag egy pár bájtos fejléccel indul, amiben jelezzük, hogy az utolsó darab-e, amit átküldtünk, továbbá egy másik mező jelzi a byteoffset-et a fájl elejétől. Működés:

- Ha a kliens kapott egy adatcsomagot, akkor egy nyugtacsomagot küld vissza.
- A nyugtacsomag fogadása után a szerver, küldi a következő adatcsomagot.
- Ha nem jön nyugta, akkor T idő után újraküldi a korábbi adatcsomagot. (pl. T=200ms)
- Ha nyugta veszik el, akkor a vevő az offset alapján el tudja dönteni, hogy egy új adatcsomag, vagy egy korábbi duplikátuma érkezett-e.
- Ha az utolsó csomag is megérkezett, akkor a kliens nyugtázza azt is és lezárja a fájlba írást. A szerver az utolsó nyugta után befejezi az átvitelt.

Feladat 4

- Készítsünk egy ***kliens***-***proxy***-***szerver*** alkalmazást, ahol:
  - a ***szerver*** egy UDP szerver,
  - a ***proxy*** a ***szerver*** irányába egy UDP kliens, a ***kliens*** irányába egy TCP szerver,
  - a ***kliens*** egy TCP kliens a ***proxy*** irányába
- Folyamat:
  - a ***kliens*** küldje a ‚Hello Server' üzenetet a ***proxy***nak,
  - amely küldje tovább azt a ***szerver***nek,
  - amely válaszolja vissza a ‚Hello Kliens' üzenetet a ***proxy***nak,
  - amely küldje tovább azt a ***kliens***nek

ALAPVETŐ ESZKÖZÖK I.

tcpdump, wireshark

Hálózati forgalom felvétele/elemzése

tcpdump (Linux):

forgalom figyelő eszköz, a hálózati interfészről jövő csomagokat tudja olvasni

```
lakis@dpdk-switch:~$ sudo tcpdump -i enp8s0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp8s0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:15:26.376139 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 4154664816:4154665024, ack 289117644, win 384, length 208
09:15:26.376403 IP 192.168.0.102.43549 > 192.168.0.192.domain: 52681+ PTR? 35.167.181.157.in-addr.arpa. (45)
09:15:26.376994 IP 192.168.0.192.domain > 192.168.0.102.43549: 52681* 1/0/0 PTR oktnb35.inf.elte.hu. (78)
09:15:26.377100 IP 192.168.0.102.57511 > 192.168.0.192.domain: 64457+ PTR? 102.0.168.192.in-addr.arpa. (44)
09:15:26.377645 IP 192.168.0.192.domain > 192.168.0.102.57511: 64457 NXDomain 0/1/0 (79)
09:15:26.377723 IP 192.168.0.102.49012 > 192.168.0.192.domain: 6981+ PTR? 192.0.168.192.in-addr.arpa. (44)
09:15:26.377851 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 208:400, ack 1, win 384, length 192
09:15:26.378180 IP 192.168.0.192.domain > 192.168.0.102.49012: 6981 NXDomain 0/1/0 (79)
09:15:26.378267 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 400:976, ack 1, win 384, length 576
09:15:26.378291 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 976:1248, ack 1, win 384, length 272
09:15:26.378340 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 1248:1600, ack 1, win 384, length 352
09:15:26.378387 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 1600:1776, ack 1, win 384, length 176
09:15:26.378440 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 1776:1952, ack 1, win 384, length 176
09:15:26.378489 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 1952:2128, ack 1, win 384, length 176
09:15:26.378538 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 2128:2304, ack 1, win 384, length 176
09:15:26.378587 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 2304:2480, ack 1, win 384, length 176
09:15:26.378636 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 2480:2656, ack 1, win 384, length 176
09:15:26.378685 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 2656:2832, ack 1, win 384, length 176
09:15:26.378734 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 2832:3008, ack 1, win 384, length 176
09:15:26.378783 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 3008:3184, ack 1, win 384, length 176
09:15:26.378832 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 3184:3360, ack 1, win 384, length 176
```

Hálózati forgalom felvétele/elemzése

tcpdump – protokoll filter

```
lakis@dpdk-switch:~$ sudo tcpdump -i enp8s0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp8s0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:16:49.470737 IP dpdk-pktgen > 192.168.0.102: ICMP echo request, id 5668, seq 1, length 64
09:16:49.470766 IP 192.168.0.102 > dpdk-pktgen: ICMP echo reply, id 5668, seq 1, length 64
09:16:50.471818 IP dpdk-pktgen > 192.168.0.102: ICMP echo request, id 5668, seq 2, length 64
09:16:50.471834 IP 192.168.0.102 > dpdk-pktgen: ICMP echo reply, id 5668, seq 2, length 64
09:16:51.471716 IP dpdk-pktgen > 192.168.0.102: ICMP echo request, id 5668, seq 3, length 64
09:16:51.471732 IP 192.168.0.102 > dpdk-pktgen: ICMP echo reply, id 5668, seq 3, length 64
09:16:52.471713 IP dpdk-pktgen > 192.168.0.102: ICMP echo request, id 5668, seq 4, length 64
09:16:52.471729 IP 192.168.0.102 > dpdk-pktgen: ICMP echo reply, id 5668, seq 4, length 64
09:16:53.471720 IP dpdk-pktgen > 192.168.0.102: ICMP echo request, id 5668, seq 5, length 64
09:16:53.471736 IP 192.168.0.102 > dpdk-pktgen: ICMP echo reply, id 5668, seq 5, length 64
```

Hálózati forgalom felvétele/elemzése

tcpdump – további filterek

```
lakis@dpdk-switch:~$ sudo tcpdump -i enp8s0 host 192.168.0.101 and port 1111
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp8s0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:20:23.289035 IP dpdk-pktgen.48524 > 192.168.0.102.1111: Flags [S], seq 1544265047, win 29200, options [mss 1460,sackOK,TS val 409718781 ecr 0,nop,wscale 7],
length 0
09:20:23.289067 IP 192.168.0.102.1111 > dpdk-pktgen.48524: Flags [R.], seq 0, ack 1544265048, win 0, length 0
```

Hálózati forgalom felvétele/elemzése

tcpdump – további filterek

```
lakis@dpdk-switch:~$ sudo tcpdump -vvv -A -i enp8s0 host 192.168.0.101 and port 1111
tcpdump: listening on enp8s0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:27:26.361105 IP (tos 0x10, ttl 64, id 14532, offset 0, flags [DF], proto TCP (6), length 60)
    dpdk-pktgen.48546 > 192.168.0.102.1111: Flags [S], cksum 0xelle (correct), seq 3578222049, win 29200, options [mss 1460,sackOK,TS val 409824549 ecr 0,nop,ws
cale 7], length 0
E..<8.@.@......e...f...W.GU.......r............
.mm%........
09:27:26.361137 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.0.102.1111 > dpdk-pktgen.48546: Flags [S.], cksum 0x824a (incorrect -> 0xdda8), seq 1341274724, ack 3578222050, win 28960, options [mss 1460,sackOK,
TS val 835209270 ecr 409824549,nop,wscale 7], length 0
E..<..@.@......f...e.W..O.:d.GU...q .J.........
1.H6.mm%....
09:27:26.361250 IP (tos 0x10, ttl 64, id 14533, offset 0, flags [DF], proto TCP (6), length 52)
    dpdk-pktgen.48546 > 192.168.0.102.1111: Flags [.], cksum 0x7cb0 (correct), seq 1, ack 1, win 229, options [nop,nop,TS val 409824549 ecr 835209270], length 0
E..48.@.@......e...f...W.GU.O.:e....|......
.mm%1.H6
09:27:31.152091 IP (tos 0x10, ttl 64, id 14534, offset 0, flags [DF], proto TCP (6), length 59)
    dpdk-pktgen.48546 > 192.168.0.102.1111: Flags [P.], cksum 0x4a14 (correct), seq 1:8, ack 1, win 229, options [nop,nop,TS val 409825747 ecr 835209270], lengt
h 7
E..;8.@.@......e...f...W.GU.O.:e....J......
.mq.1.H6Hello

09:27:31.152109 IP (tos 0x0, ttl 64, id 29267, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.0.102.1111 > dpdk-pktgen.48546: Flags [.], cksum 0x8242 (incorrect -> 0x734f), seq 1, ack 8, win 227, options [nop,nop,TS val 835210468 ecr 40982574
7], length 0
E..4rS@.@.FU...f...e.W..O.:e.GU......B.....
1.L..mq.
09:27:42.531278 IP (tos 0x0, ttl 64, id 29268, offset 0, flags [DF], proto TCP (6), length 55)
    192.168.0.102.1111 > dpdk-pktgen.48546: Flags [P.], cksum 0x8245 (incorrect -> 0x15be), seq 1:4, ack 8, win 227, options [nop,nop,TS val 835213313 ecr 40982
5747], length 3
E..7rT@.@.FQ...f...e.W..O.:e.GU......E.....
1.X..mq.Hi

09:27:42.531425 IP (tos 0x10, ttl 64, id 14535, offset 0, flags [DF], proto TCP (6), length 52)
    dpdk-pktgen.48546 > 192.168.0.102.1111: Flags [.], cksum 0x5d10 (correct), seq 8, ack 4, win 229, options [nop,nop,TS val 409828592 ecr 835213313], length 0
E..48.@.@......e...f...W.GU.O.:h....]......
.m|.1.X.
09:27:50.984854 IP (tos 0x10, ttl 64, id 14536, offset 0, flags [DF], proto TCP (6), length 67)
    dpdk-pktgen.48546 > 192.168.0.102.1111: Flags [P.], cksum 0xf203 (correct), seq 8:23, ack 4, win 229, options [nop,nop,TS val 409830705 ecr 835213313], leng
th 15
E..C8.@.@......e...f...W.GU.O.:h...........
.m.11.X.Hogy vagyunk?

09:27:50.984872 IP (tos 0x0, ttl 64, id 29269, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.0.102.1111 > dpdk-pktgen.48546: Flags [.], cksum 0x8242 (incorrect -> 0x4c81), seq 4, ack 23, win 227, options [nop,nop,TS val 835215426 ecr 4098307
05], length 0
```

Hálózati forgalom felvétele/elemzése

tcpdump – mentés pcap fájlba és fájlból elemzés

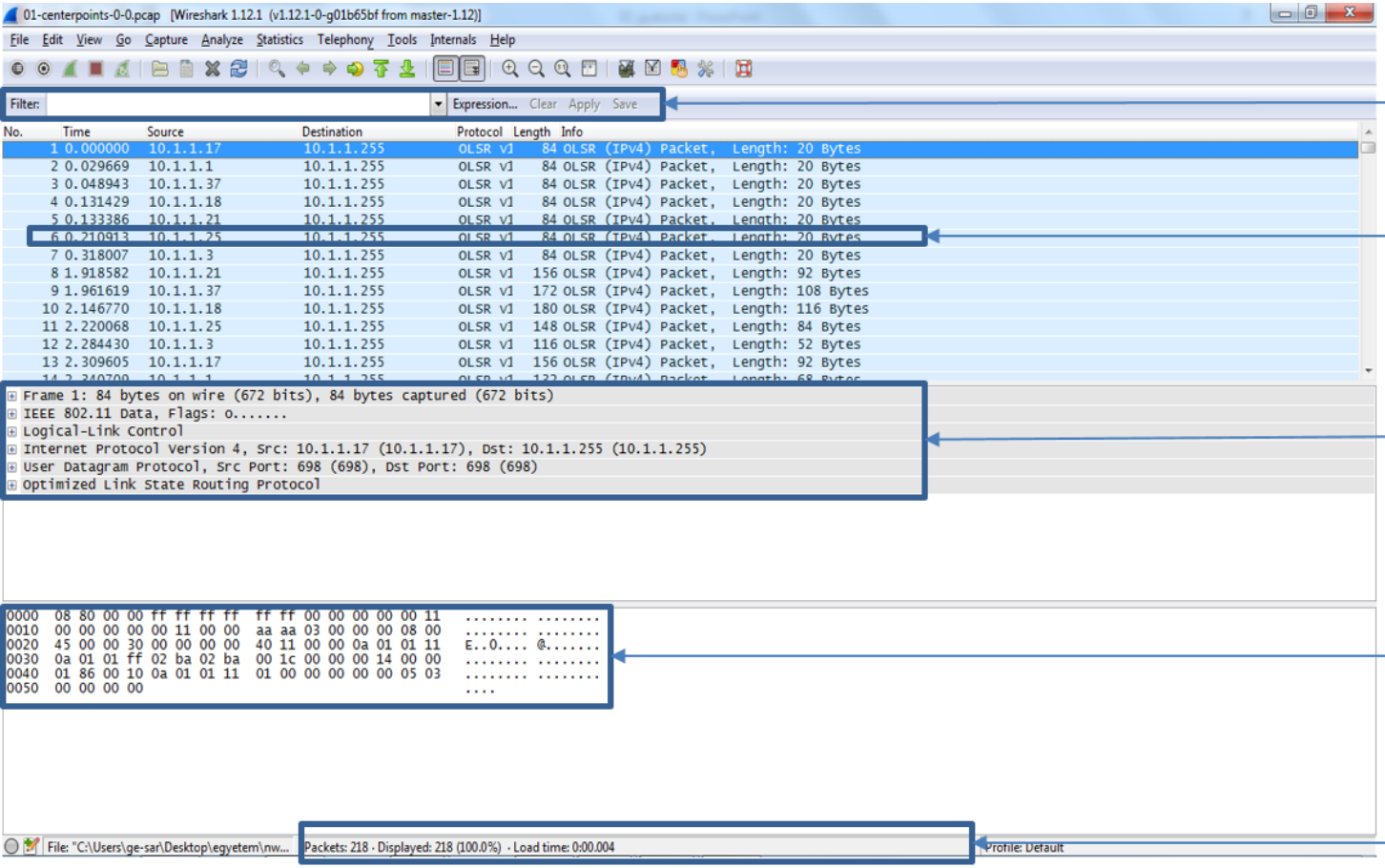
```
lakis@dpdk-switch:~$ sudo tcpdump -w test.pcap -i enp8s0
tcpdump: listening on enp8s0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C4 packets captured
6 packets received by filter
0 packets dropped by kernel
lakis@dpdk-switch:~$ tcpdump -r test.pcap
reading from file test.pcap, link-type EN10MB (Ethernet)
09:31:32.000164 IP 192.168.0.102.ssh > oktnb35.inf.elte.hu.55015: Flags [P.], seq 4154857792:4154857936, ack 289145644, win 384, length 144
09:31:32.060031 IP oktnb35.inf.elte.hu.55015 > 192.168.0.102.ssh: Flags [.], ack 144, win 3542, length 0
09:31:34.354029 IP 192.168.0.192.48309 > 255.255.255.255.7437: UDP, length 173
09:31:37.377992 IP 192.168.0.192.48309 > 255.255.255.255.7437: UDP, length 173
```

Pcap fájl visszajátszására is van lehetőség: tcpreplay

Wireshark

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

Wireshark ablakok



Szűrők definiálására alkalmas input eszközök

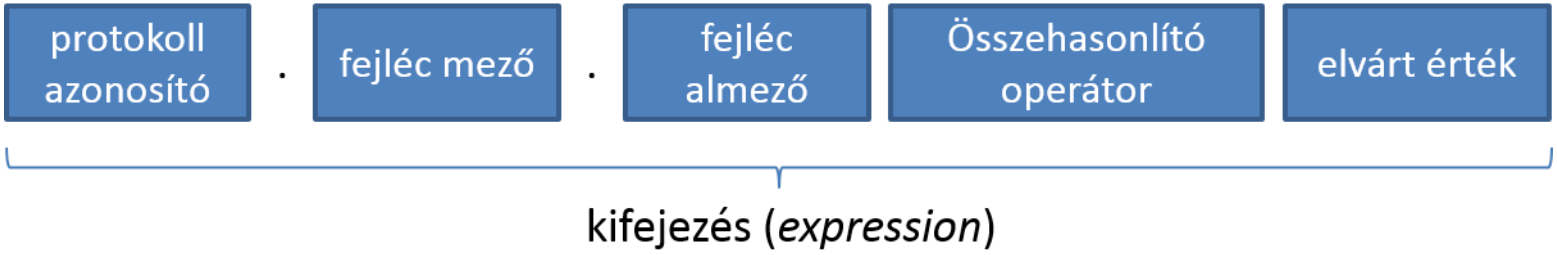
Csomag összefoglaló nézete

Kiválasztott csomag hierarchikus nézet

Kiválasztott csomag bájt-alapú nézet

Szűrés statisztikái

Wireshark szűrők



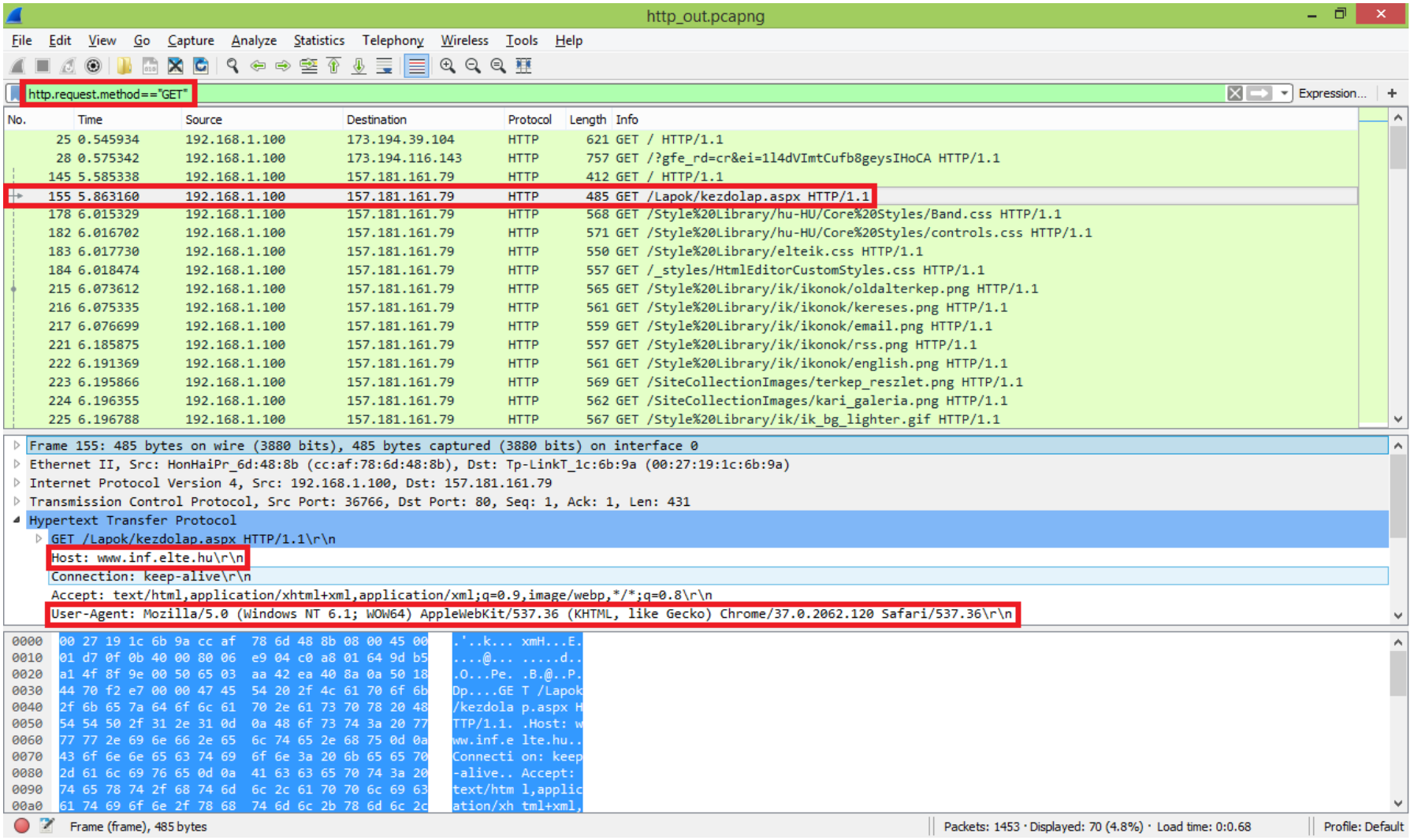
- Operátorok: or, and, xor, not
- protokollok: ip, tcp, http… (teljes listát lásd → Expression gomb)
- Példa: tcp.flags.ack==1 and tcp.dstport==80 (tcp nyugta flag és fogadó port beállítva)

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?
3. Volt-e olyan kérés, amely titkosított kommunikációt takar?

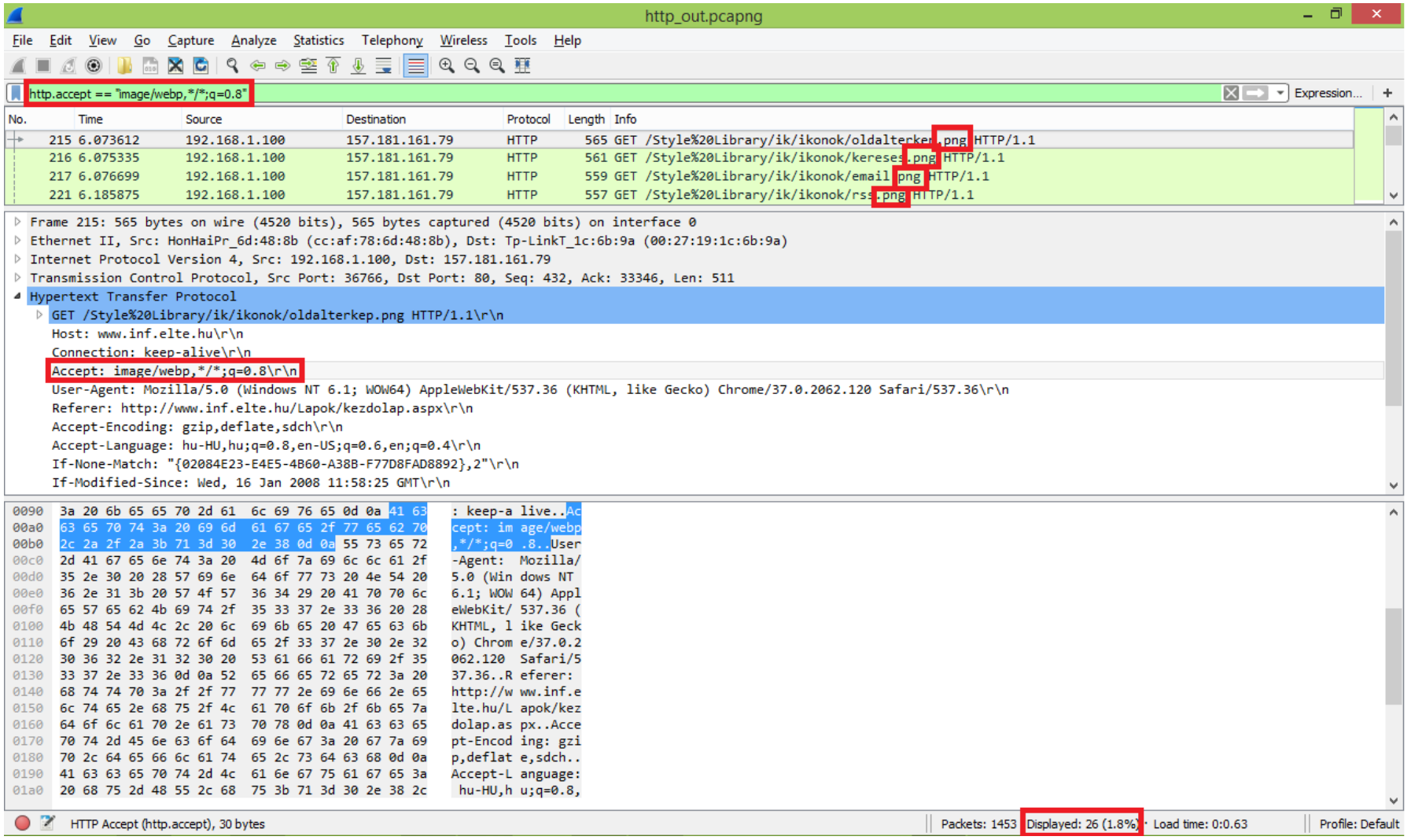
Wireshark példa megoldás I.



Wireshark példa megoldás I.

- Milyen oldalakat kértek le a szűrés alapján HTTP GET metódussal? Milyen böngészőt használtak hozzá?
- Szűrés: http.request.method=="GET"
- User-agent header-ből lehet következtetni a böngésző típusára: User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36
- Ehhez segítségünkre lehet ez a link: http://www.zytrax.com/tech/web/browser_ids.htm

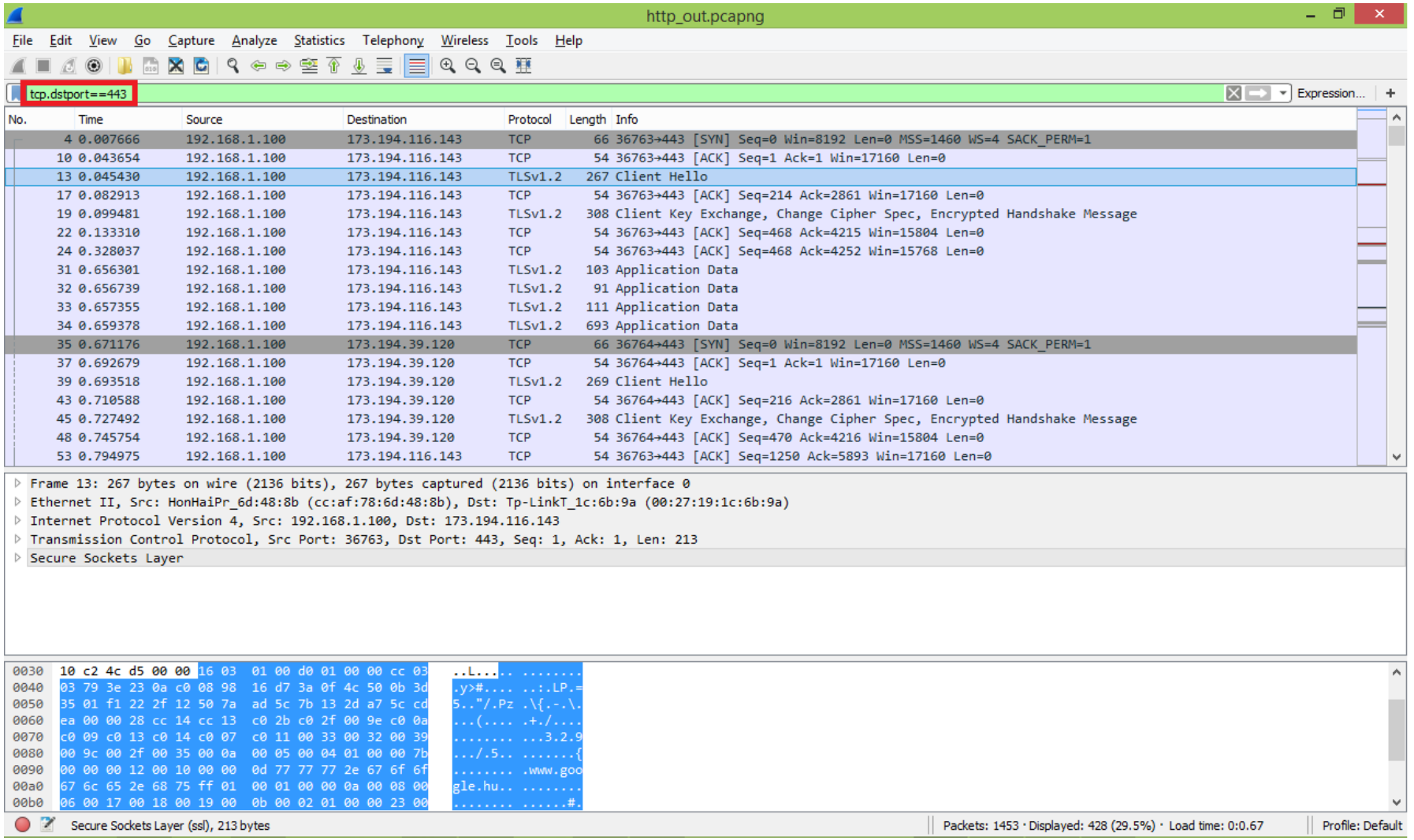
Wireshark példa megoldás II.



Wireshark példa megoldás II.

- Hány darab képet érintett a böngészés?
- Szűrés: http.accept == "image/webp,*/*;q=0.8"
- Accept header: a kérésre adott válasz tartalmának elfogadható típusa
- WebP: új, veszteséges tömörítést alkalmazó képformátum, amelyet a Google fejlesztett ki a web hálózati forgalmának csökkentésére

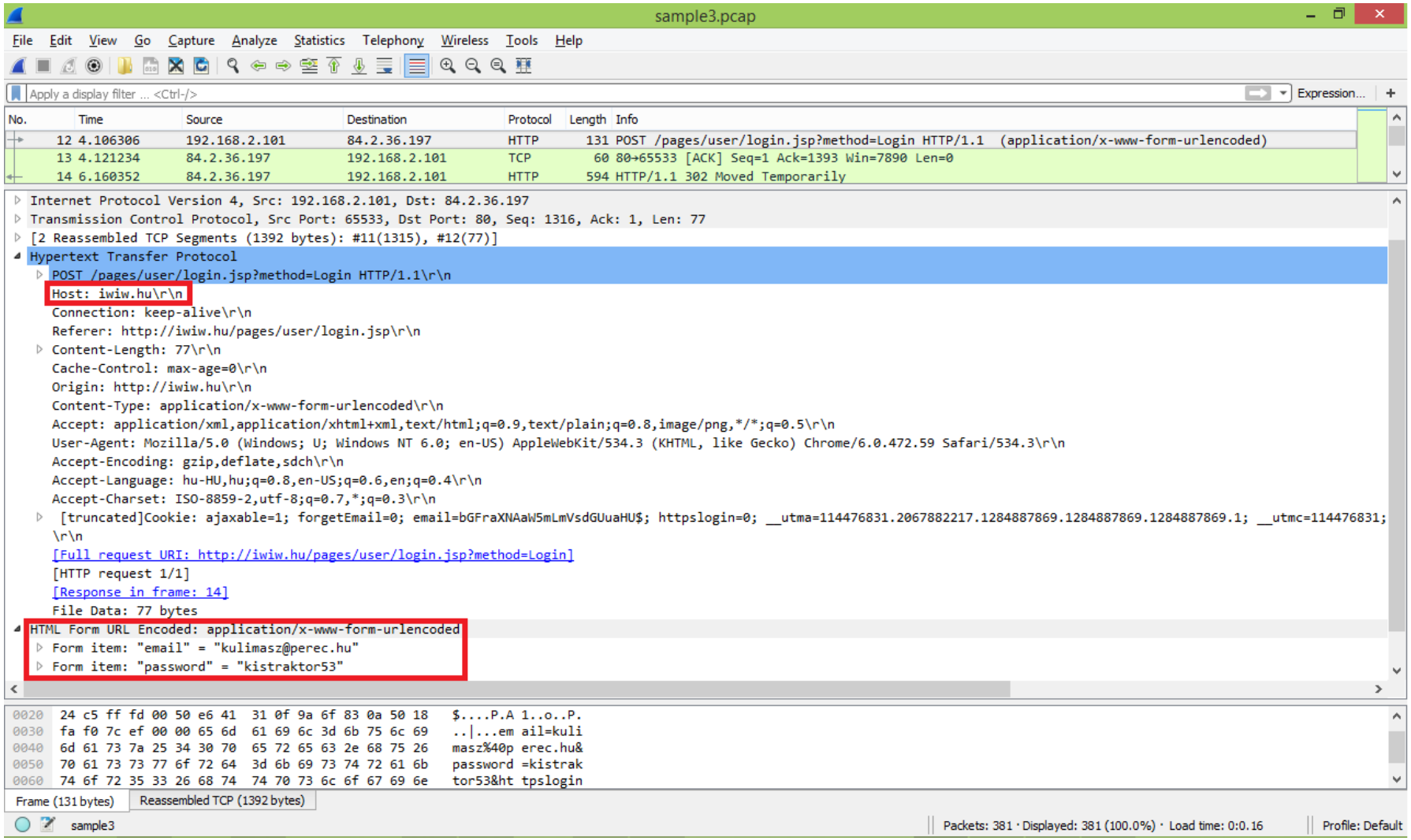
Wireshark példa megoldás III.



Wireshark példa megoldás III.

- Volt-e olyan kérés, amely titkosított kommunikációt takar?
- Szűrés: tcp.dstport==443
- Transport Layer Security (TLS) titkosító protokoll feletti HTTP kommunikációra utal a 443-as port
- Elektronikus levelezéshez, banki szolgáltatásokhoz stb. elengedhetetlen
- Nélküle le lehet hallgatni a kommunikációt, lásd a következő diát (sample3.pcapng felhasználásával)

Wireshark – „leleplezés”



Alhálózati maszk

- Az alhálózat egy logikai felosztása egy IP hálózatnak. Az IP cím ezért két részből áll: hálózatszámból és hoszt azonosítóból.
- A szétválasztás a 32 bites alhálózati maszk segítségével történik, amellyel bitenkénti ÉS-t alkalmazva az IP címre megkapjuk a hálózat-, komplementerével pedig a hoszt azonosítót.
- Ez arra jó, hogy meg tudjuk határozni, hogy a címzett állomás a helyi alhálózaton van-e, vagy sem.
- Az utóbbi esetben az alapértelmezett router felé továbbítják a csomagot (default gateway).

Alhálózati maszk

- CIDR jelölés: kompakt reprezentációja egy IP címnek és a hozzátartozó hálózatszámnak
- → IP cím + ’/’ + decimális szám.
- Pl.: 135.46.57.14/24 esetben 135.46.57.14 az IP cím,
- 255.255.255.0 a hálózati maszk (24 db. 1-es bit az elejétől),
- így 135.46.57.0 a hálózat azonosító.

Alhálózati maszk – példa

	10000111	00101110	00111001	00001110	**135.46.57.14**
AND	11111111	11111111	11111111	00000000	**255.255.255.0**
	10000111	00101110	00111001	00000000	**135.46.57.0**

135.46.57.14/24 → 135.46.57.0

Számolós feladat 1

- Hány cím érhető el a következő alhálózati maszkokkal? Adjuk meg a minimális és maximális címet is!
- 188.100.22.12/32
- 188.100.22.12/20
- 188.100.22.12/10

Számolós feladat 1 megoldása

188.100.22.12/32

	10111100	01100100	00010110	00001100	**188.100.22.12**
AND	11111111	11111111	11111111	11111111	**255.255.255.255**
	10111100	01100100	00010110	00001100	**188.100.22.12**

egy darab a 188.100.22.12

Számolós feladat 1 megoldása

188.100.22.12/20

	10111100	01100100	00010110	00001100	**188.100.22.12**
AND	11111111	11111111	11110000	00000000	**255.255.240.0**
	10111100	01100100	00010000	00000000	**188.100.16.0**

$2^{32-20} = 2^{12} = 4096$ darab lenne, de valójában ebből még kettőt le kell vonni, mert speciális jelentéssel bírnak:

- csupa 0: az alhálózat hálózati címe (magára az alhálózatra vonatkozik)
- csupa 1-es: broadcast a helyi hálózaton

Min.	10111100	01100100	00010000	00000001	**188.100.16.1**
Max.	10111100	01100100	00011111	11111110	**188.100.31.254**

Számolós feladat 1 megoldása

188.100.22.12/10

	10111100	01100100	00010110	00001100	**188.100.22.12**
AND	11111111	11000000	00000000	00000000	**255.255.240.0**
	10111100	01000000	00000000	00000000	**188.64.0.0**

$2^{32-10} = 2^{22} - 2 = 4194302$ darab

Min.	10111100	01000000	00000000	00000001	**188.64.0.1**
Max.	10111100	01111111	11111111	11111110	**188.127.255.254**

Hálózati címfordítás (NAT)

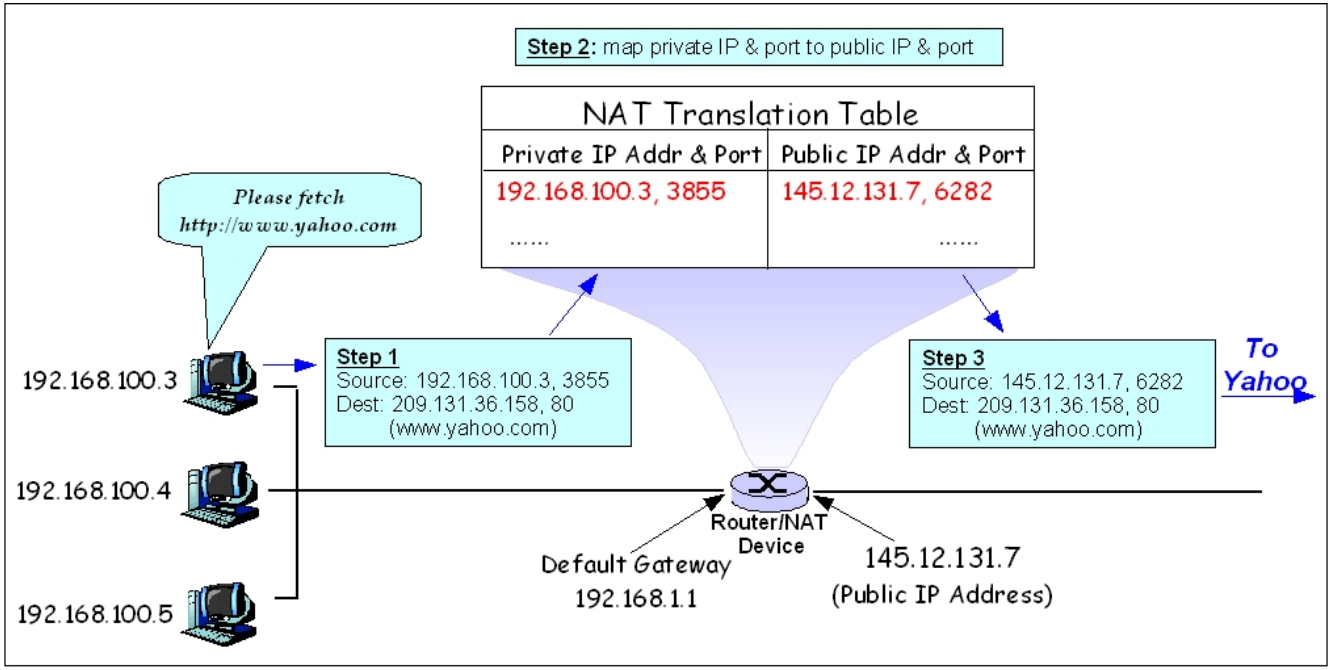
- Gyors javítás az IP címek elfogyásának problémájára.
- Az internet forgalomhoz minden cégnek egy vagy legalábbis kevés IP címet adnak (publikus IP cím(ek))
- A publikus IP cím hozzá van rendelve egy router-hez, a helyi hálózaton (LAN) belül, - amely mögötte van, - minden eszközhöz egy privát IP cím van rendelve
- A privát IP címek csak a LAN-on belül érvényesek (vannak IP cím tartományok erre a célra foglalva)

Hálózati címfordítás (NAT)

- Ha a helyi hálózaton lévő másik géppel akarunk kapcsolatot létesíteni → közvetlenül el tudjuk érni
- Amikor helyi eszközről akarunk egy külső eszközt elérni, mi történik?
- Szükségünk van port mezők használatára, ami TCP-nél vagy UDP-nél van

Hálózati címfordítás (NAT)

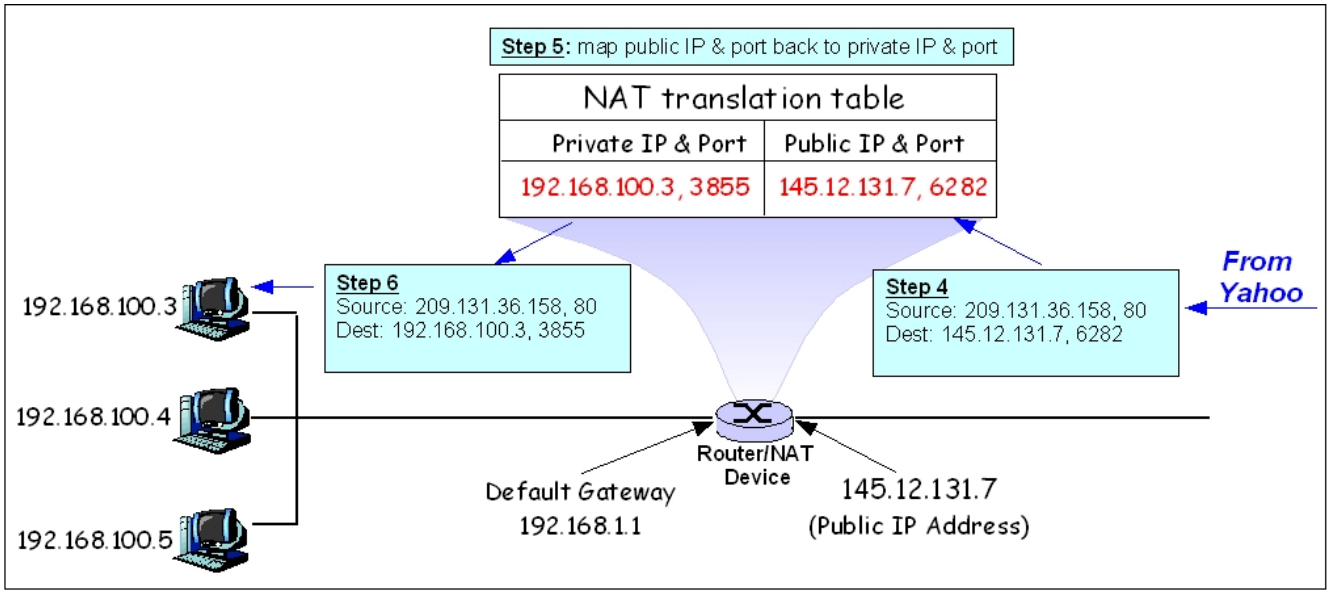
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- 192.168.100.3 privát IP című gépről HTTP kérés, 3855 porton → Default gateway (192.168.1.1): megnézi a transzlációs tábláját:
  - Ha létezik már a (192.168.100.3, 3855) párhoz (publikus IP cím, port) bejegyzés → lecseréli a küldő forrását arra
  - Ha nincs létrehoz egy új bejegyzést (egyedi lesz!), és azt használja fel a cseréhez

Hálózati címfordítás (NAT)

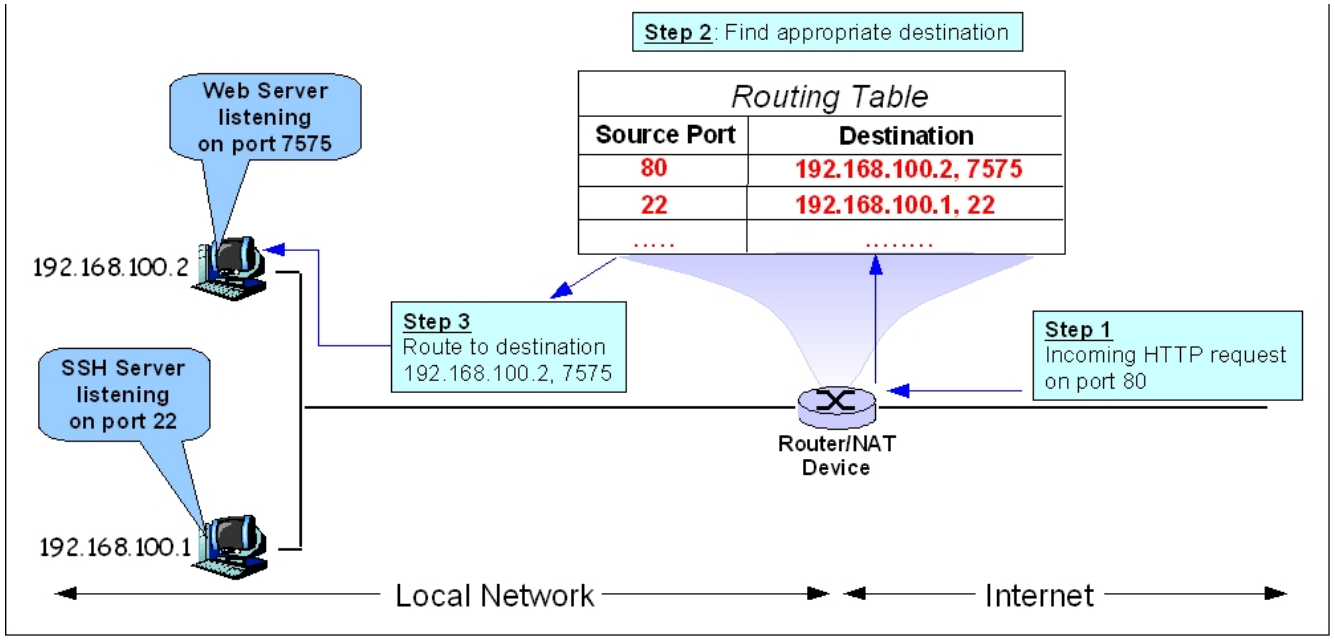
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- A HTTP válasz a yahoo-tól ugyanúgy a router transzlációs tábláján keresztül megy végbe, csak fordított irányban
- Egy különbség: hiányzó bejegyzés esetén a csomagot eldobja a router

Porttovábbítás (port forwarding)

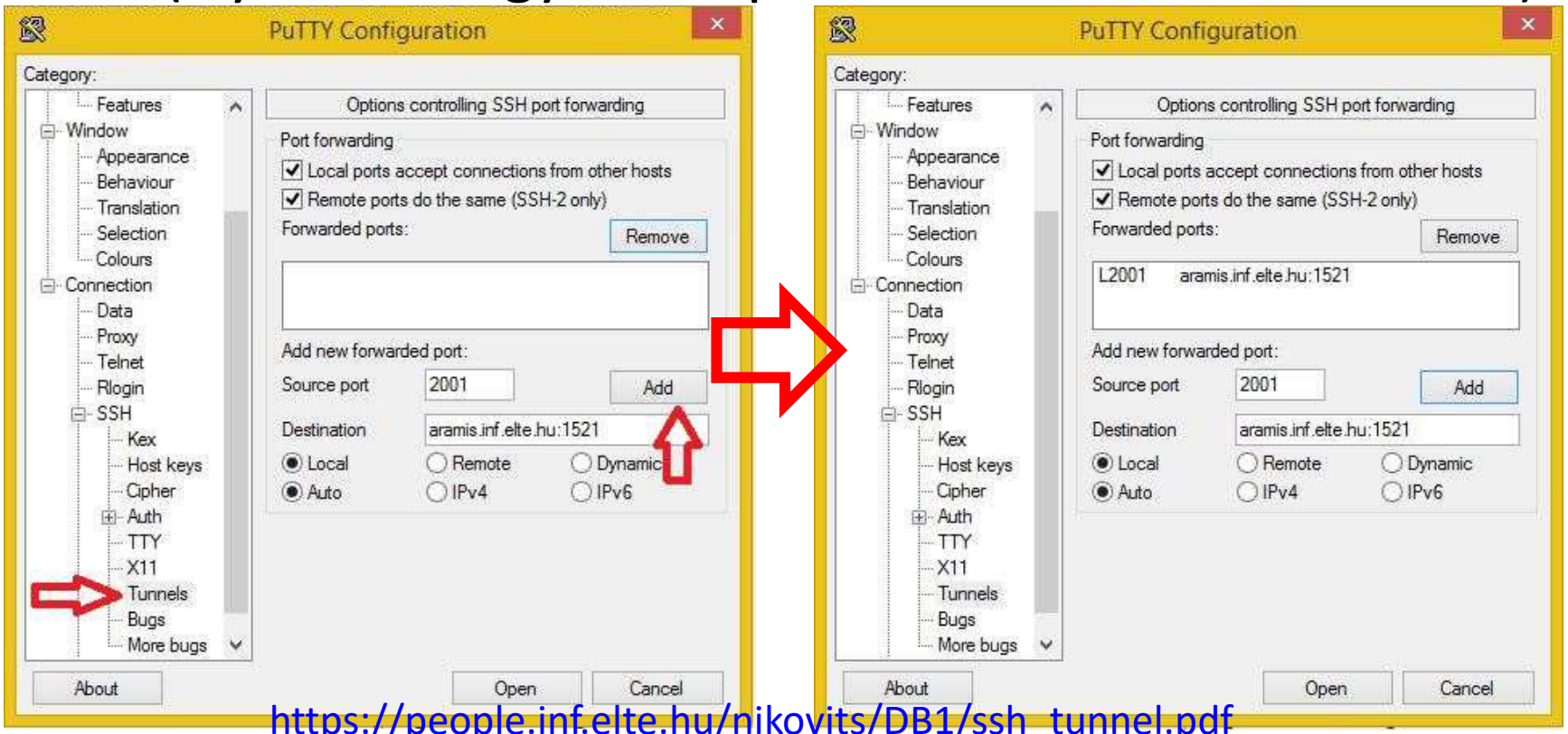
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- Az előző példánál a címfordítás transzparens volt (csak a router tudott arról, hogy IP konverzió zajlik). Mit lehet tenni, ha pl. egy belső hálózaton lévő HTTP szervert akarunk elérni kívülről?
- **Porttovábbítás** lehetővé teszi adott lokális hálózaton (LAN) lévő privát IP címek külső elérését egy megadott porton keresztül
- Gyakorlatilag ez a *statikus* NAT alkalmazása

SSH Tunnel

- A porttovábbítás egyik tipikus alkalmazása
- Windows (putty) beállítások
  - (Nyitni kell egy ssh kapcsolatot a caesar.elte.hu-ra)



https://people.inf.elte.hu/nikovits/DB1/ssh_tunnel.pdf

SSH Tunnel

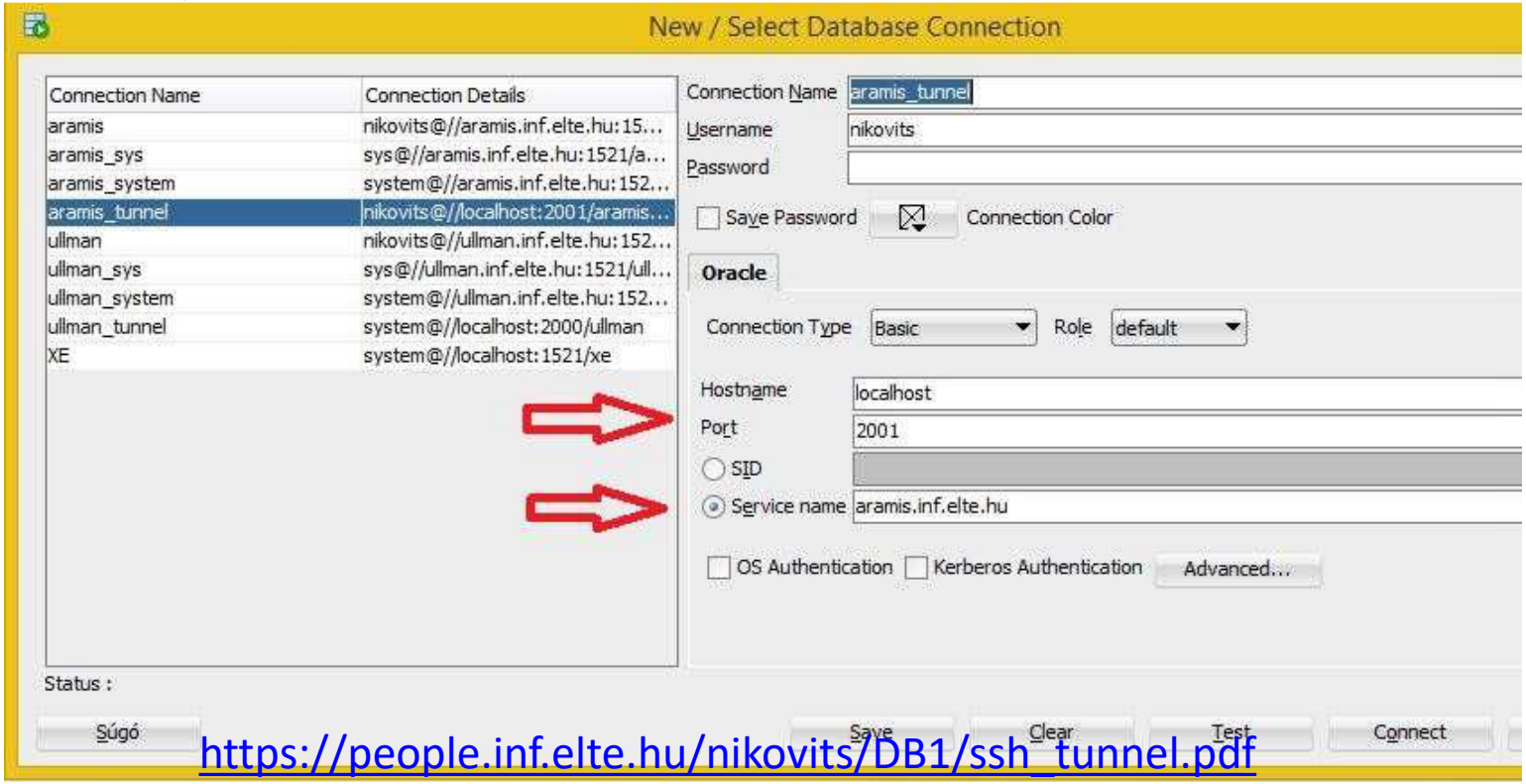
- Linux

```
ssh –L 2001:aramis.inf.elte.hu:1521 user@hostname
```

- ssh –L <localport>:<remote host>:<remote port> <gateway you can ssh in>
  - localport: a localhost ezen porján lesz elérhető a távoli szerver/szolgáltatás
  - remote host:remote port: ide csatlakozik a tunnel végpont, minden, amit a localportra küldünk ide fog továbbítódni és vissza. A gateway-ről elérhetőnek kell lennie!
  - gateway: a gép, amire be tudunk sshval lépni!

SSH Tunnel

- Használat SqlDeveloper-nél:
  - (ssh kapcsolatnak fenn kell állnia végig az adatbázis kapcsolat ideje alatt)



https://people.inf.elte.hu/nikovits/DB1/ssh_tunnel.pdf

VÉGE
KÖSZÖNÖM A FIGYELMET!