

Számítógépes Hálózatok

3. gyakorlat

NC-NetCat (SoCat), avagy hálózati svájcbicska

szerver imitálása

```
nc -l -p 1234
```

kliens imitálása

```
nc destination_host 1234
```

NetCat TUTORIAL:

<https://www.binarytides.com/netcat-tutorial-for-beginners>

SoCat TUTORIAL:

<https://blog.rootshell.be/2010/10/31/socat-another-network-swiss-army-knife>

Tcpdump – hálózati forgalomfigyelés

```
lakis@dppk-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
```

Protokol szűrés

```
lakis@dppk-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
```

Tcpdump – hálózati forgalomfigyelés

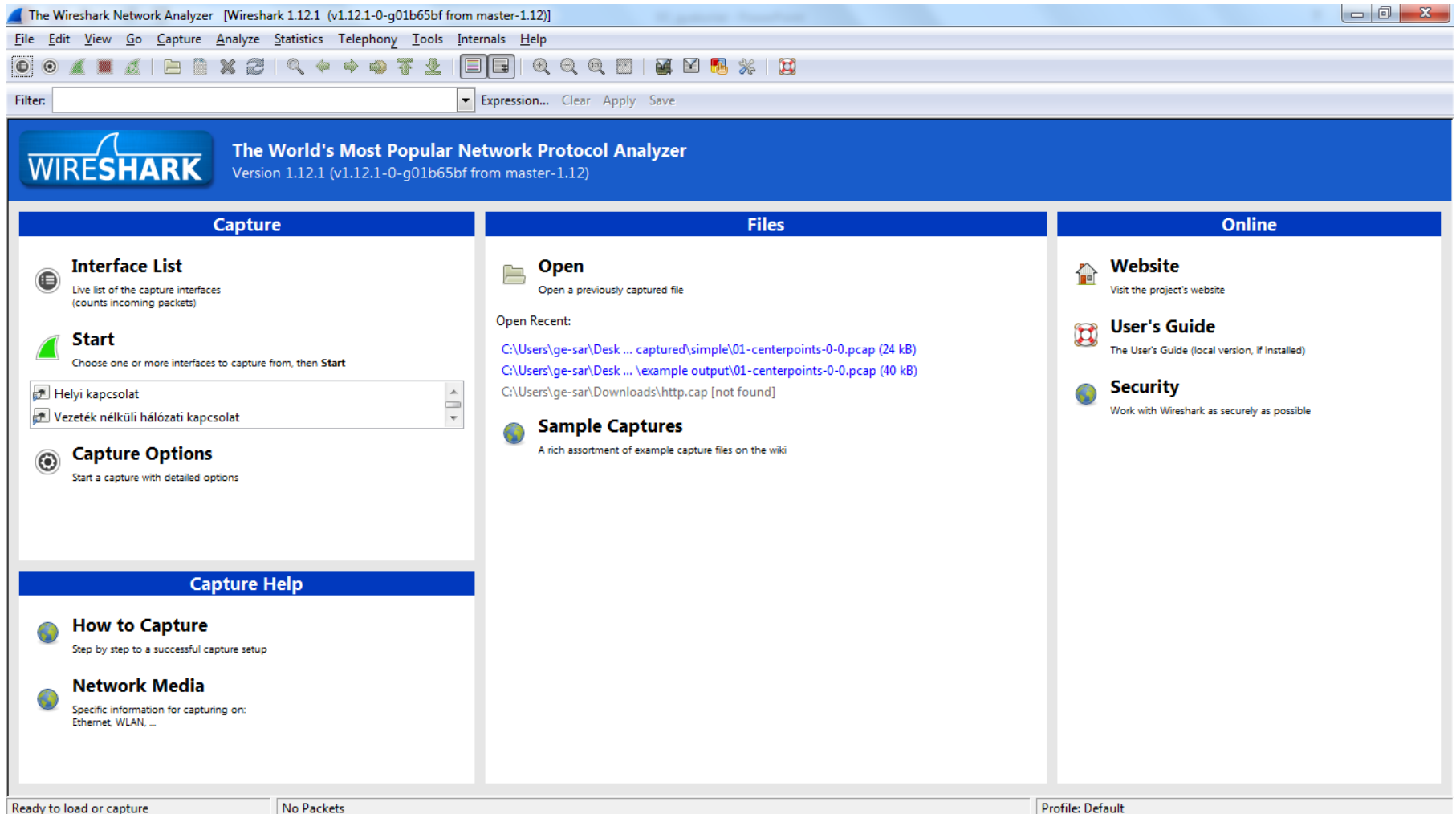
host és port szűrés

```
lakis@dppdk-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 v
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
```

Mentés pcap fileba

```
lakis@dppdk-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@dppdk-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
```

Wireshark



Wireshark

01-centerpoints-0-0.pcap [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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.17	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
2	0.029669	10.1.1.1	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
3	0.048943	10.1.1.37	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
4	0.131429	10.1.1.18	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
5	0.133386	10.1.1.21	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
6	0.210913	10.1.1.25	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
7	0.318007	10.1.1.3	10.1.1.255	OLSR v1	84	OLSR (IPv4) Packet, Length: 20 Bytes
8	1.918582	10.1.1.21	10.1.1.255	OLSR v1	156	OLSR (IPv4) Packet, Length: 92 Bytes
9	1.961619	10.1.1.37	10.1.1.255	OLSR v1	172	OLSR (IPv4) Packet, Length: 108 Bytes
10	2.146770	10.1.1.18	10.1.1.255	OLSR v1	180	OLSR (IPv4) Packet, Length: 116 Bytes
11	2.220068	10.1.1.25	10.1.1.255	OLSR v1	148	OLSR (IPv4) Packet, Length: 84 Bytes
12	2.284430	10.1.1.3	10.1.1.255	OLSR v1	116	OLSR (IPv4) Packet, Length: 52 Bytes
13	2.309605	10.1.1.17	10.1.1.255	OLSR v1	156	OLSR (IPv4) Packet, Length: 92 Bytes
14	2.240700	10.1.1.1	10.1.1.255	OLSR v1	120	OLSR (IPv4) Packet, Length: 68 Bytes

Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)

IEEE 802.11 Data, Flags: 0.....

Logical-Link Control

Internet Protocol Version 4, Src: 10.1.1.17 (10.1.1.17), Dst: 10.1.1.255 (10.1.1.255)

User Datagram Protocol, Src Port: 698 (698), Dst Port: 698 (698)

Optimized Link State Routing Protocol

0000 08 80 00 00 ff ff ff ff ff ff 00 00 00 00 11
0010 00 00 00 00 00 11 00 00 aa aa 03 00 00 00 08 00
0020 45 00 00 30 00 00 00 00 40 11 00 00 0a 01 01 11 E..0...@.....
0030 0a 01 01 ff 02 ba 02 ba 00 1c 00 00 00 14 00 00
0040 01 86 00 10 0a 01 01 11 01 00 00 00 00 05 03
0050 00 00 00 00

Packets: 218 · Displayed: 218 (100.0%) · Load time: 0:00.004

Profile: Default

Szűrők definiálására
alkalmas input
eszközök

Csomag összefoglaló
nézete

Kiválasztott csomag
hierarchikus nézet

Kiválasztott csomag bájttól
alapú nézet

Szűrés statisztikái

Wireshark

- Korábban rögzített adatok elemzésére szolgál.
- Szűrés felépítése:



- Operátorok: or, and, xor, not
- Példa: `tcp.flags.ack==1 and tcp.dstport==80`

Szűrési feladatok 1 - HTTP

A [http_out.pcapng](#) felhasználásával állomány felhasználásával válaszolja meg az alábbi kérdéseket:

1. Milyen oldalakat kértek le a szűrés alapján? Milyen böngészőt használtak hozzá?
2. Hány darab képet érintett a böngészés? (Segítség: *webp*.)
3. Hány olyan erőforrás volt, amelyet nem kellett újra töltenie a böngészőnek? Mely oldalakat érintette ez?
4. Volt-e olyan kérés, amely titkosított kommunikációt takar? (Segítség: *SSL/TLS*.) Kövesse végig az első TCP folyamatát. Mit tud kideríteni a kommunikációról?

Szűrési feladatok 1 - HTTP

A [http_out.pcapng](#) felhasználásával állomány felhasználásával válaszolja meg az alábbi kérdéseket:

1. Milyen oldalakat kértek le a szűrés alapján? Milyen böngészőt használtak hozzá?
 - `http.request.method=="GET"`
2. Hány darab képet érintett a böngészés?
 - `http.accept == "image/webp,*/*;q=0.8"`
3. Hány olyan erőforrás volt, amelyet nem kellett újra töltenie a böngészőnek? Mely oldalakat érintette ez?
 - `http.response.code == 304` (to.ttk.elte.hu és www.inf.elte.hu)
4. Volt-e olyan kérés, amely titkosított kommunikációt takar? (Segítség: *SSL/TLS*.) Kövesse végig az első TCP folyamatát. Mit tud kideríteni a kommunikációról?
 - `tcp.dstport==443`

Szűrési feladatok 2 - DNS

- A `dns_out.pcapng` felhasználásával állomány felhasználásával válaszolja meg az alábbi kérdéseket:
 1. Hány domén név feloldást kezdeményeztek a szűrés alapján? Mely domén nevek voltak ezek?
 2. Válaszon ki 3 darab különböző domén nevet, és keresse meg a válasz csomagokat hozzájuk? Hány darab válasz van az egyes kérésekre? (Segítség: *ID.*)
 3. Hány olyan névfeloldás volt, amelyre több válasz is érkezett?
 4. Volt-e iteratív lekérdezés a szűrésben? Ha igen, akkor mennyi? Ha nem, akkor mi lehet a magyarázat?

Szűrési feladatok 3 - NEPTUN

- A [neptun_out.pcapng](#) felhasználásával állomány felhasználásával válaszolja meg az alábbi kérdéseket:
 1. Milyen oldalakat kértek le a szűrés alapján? Milyen böngészőt használtak hozzá?
 2. Hány darab `SSL/TLS` protokollt használó csomag van? Az elsőn kövesse végig a kommunikációt. Minden működési elvnek megfelelően lezajlott?
 3. Kezdeményezett-e megszakítást a szerver a kommunikáció során?
 4. Kideríthető-e, hogy milyen kommunikáció folyt a szerver és a kliens között? Esetleg megtippelhető-e a használt böngésző típusa?

HTTP vs HTTPS

- Nyissuk meg a sample3.pcapng fájlt!
- Keressük olyan POST kérést, amely login oldalra vezet!
- Nézzük meg a POST hívás paramétereit!

Python socket, host név feloldás

- Socket csomag használata

```
import socket
```

- `gethostname()`

```
hostname = socket.gethostname()
```

- `gethostbyname()`

```
hostname = socket.gethostbyname('www.example.org')
```

- `gethostbyname_ex()`

```
hostname, aliases, addresses = socket.gethostbyname_ex(host)
```

- `gethostbyaddr()`

```
hostname, aliases, addrs = socket.gethostbyaddr('157.181.161.79')
```

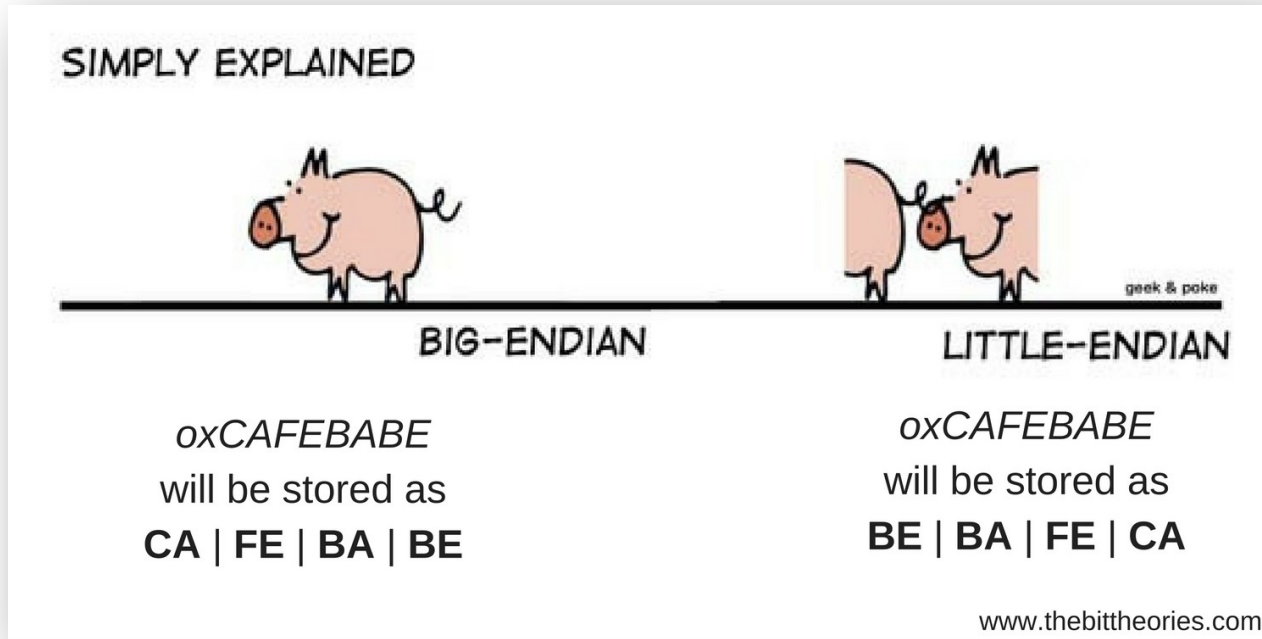
Port számok

- Bizonyos protokollokhoz tartoznak fix portszámok, konstansok (szállítási protokollok)!
- `getservbyport()`

```
socket.getservbyport(22)
```

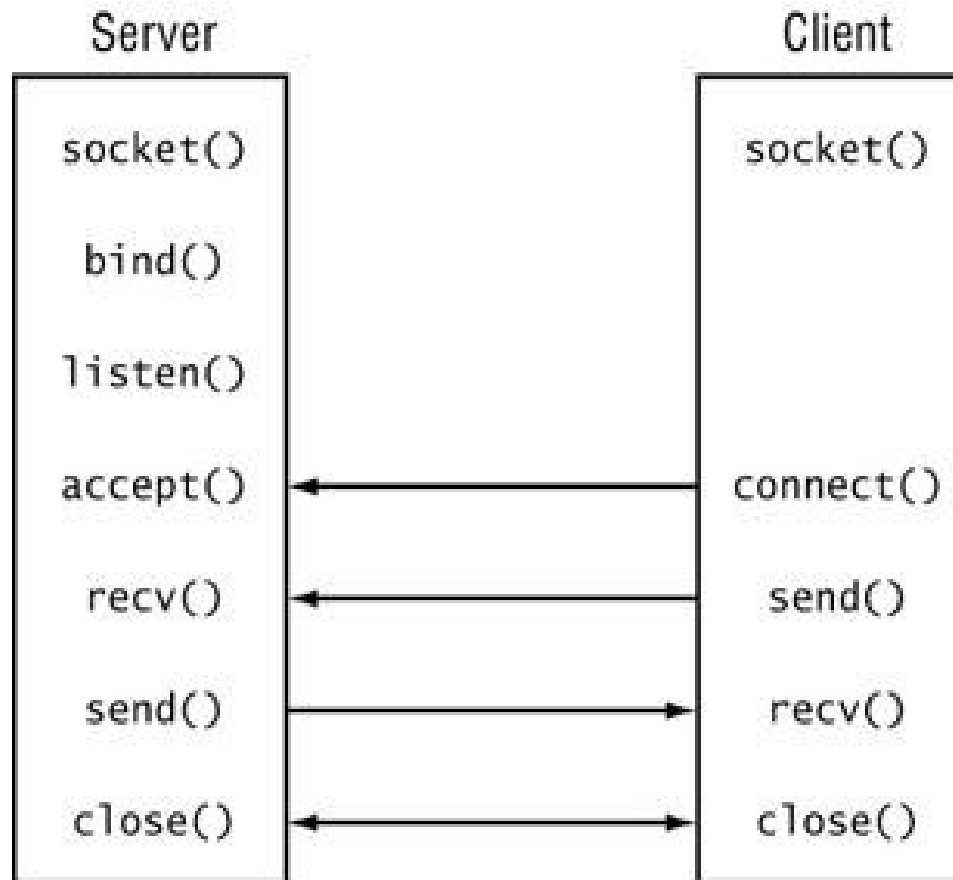
- Írassuk ki a 1..100-ig a portokat és a hozzájuk tartozó protokollokat!

Little endian, big endian



- 16 és 32 bites pozitív számok kódolása
 - htons(), htonl() – host to network short / long
 - ntohs(), ntohl() – network to host short / long

TCP



TCP

- `socket()`

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

- `bind()`

```
server_address = ('localhost', 10000)  
sock.bind(server_address)
```

- `listen()`

```
sock.listen(1)
```

- `accept()`

```
connection, client_address = sock.accept()
```

TCP

- `send()`, `sendall()`

```
connection.sendall(data) #python 2.x
```

```
connection.sendall(data.encode()) #python 3.x
```

- `recv()`

```
data = connection.recv(16) #python 2.x
```

```
data = connection.recv(16).decode() #python 3.x
```

- `close()`

```
connection.close()
```

- `connect()`

```
server_address = ('localhost', 10000)  
sock.connect(server_address)
```

Feladat

- Készítsünk egy egyszerű kliens-server alkalmazást, ahol a kliens elküld egy 'Hello server' üzenetet, és a szerver pedig válaszol neki egy 'Hello kliens' üzenettel!
- Változtassuk meg hogy ne az előre megadott portot adjuk, hanem egy tetszőlegesen kapjunk az oprendszer-től! (`sys.argv[1]`)

Struktúraküldése

- Binárisra alakítjuk az adatot

```
import struct
values = (1, 'ab'.decode(), 2.7)
packer = struct.Struct('l 2s f')      #Int, char[2], float
packed_data = packer.pack(*values)
```

- Visszalakítjuk a kapott üzenetet

```
import struct
unpacker = struct.Struct('l 2s f')
unpacked_data = unpacker.unpack(data)
```

- megj.: integer 1 – 4 byte, stringként 1 byte, azaz hatékonyabb stringként átküldeni.

Feladat

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.

VÉGE