Számítógépes Hálózatok

3. gyakorlat

NetCat, Tcpdump, Wireshark

HÁLÓZATI FORGALOM

NC-NetCat (SoCat), avagy hálózati svájcibicska

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-armyknife

Tcpdump – hálózati forgalomfigyelés

```
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
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
```

Tcpdump – szűrések (pl. protokollra)

```
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
```

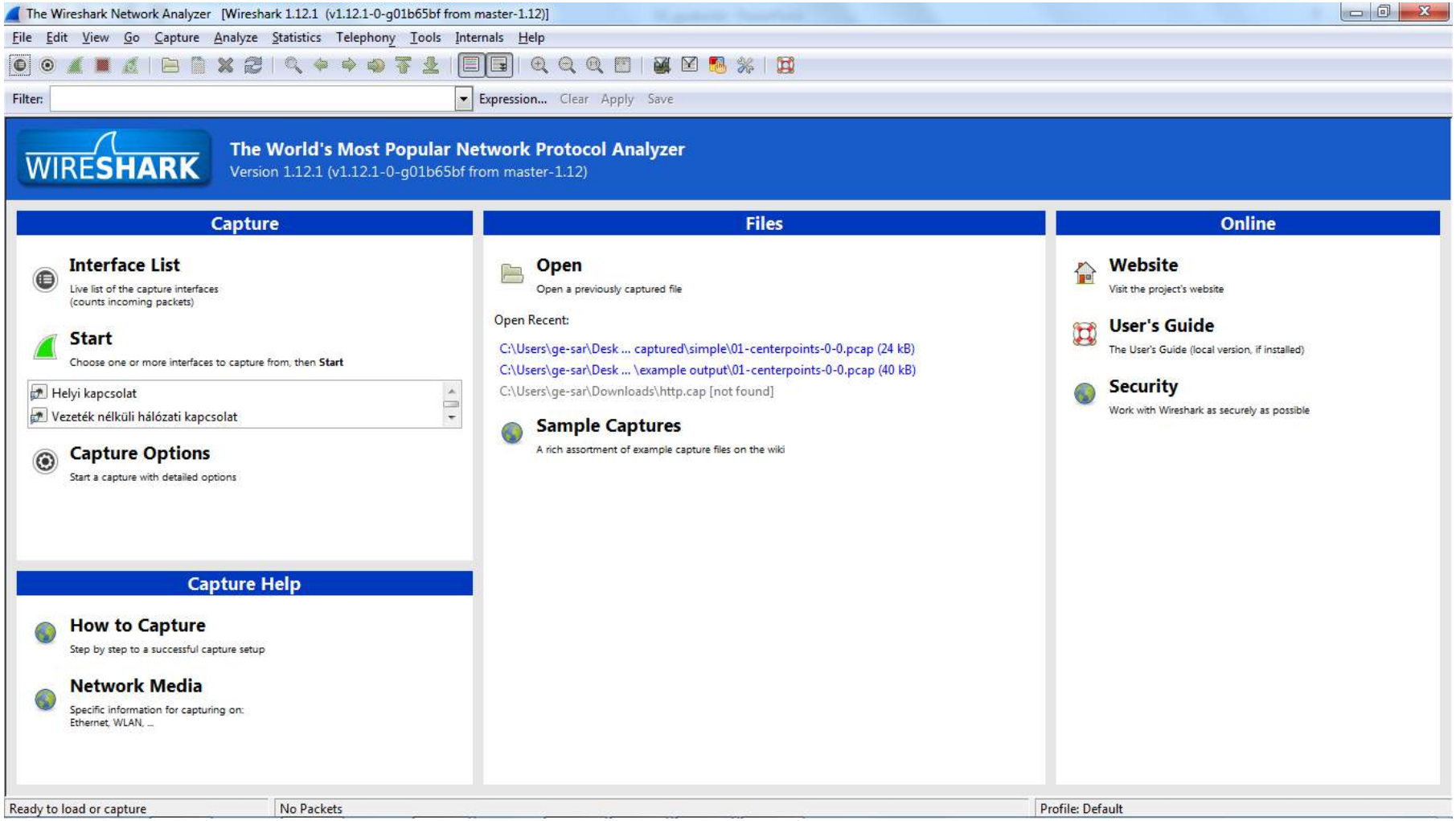
Tcpdump – host és portszűrés

```
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
```

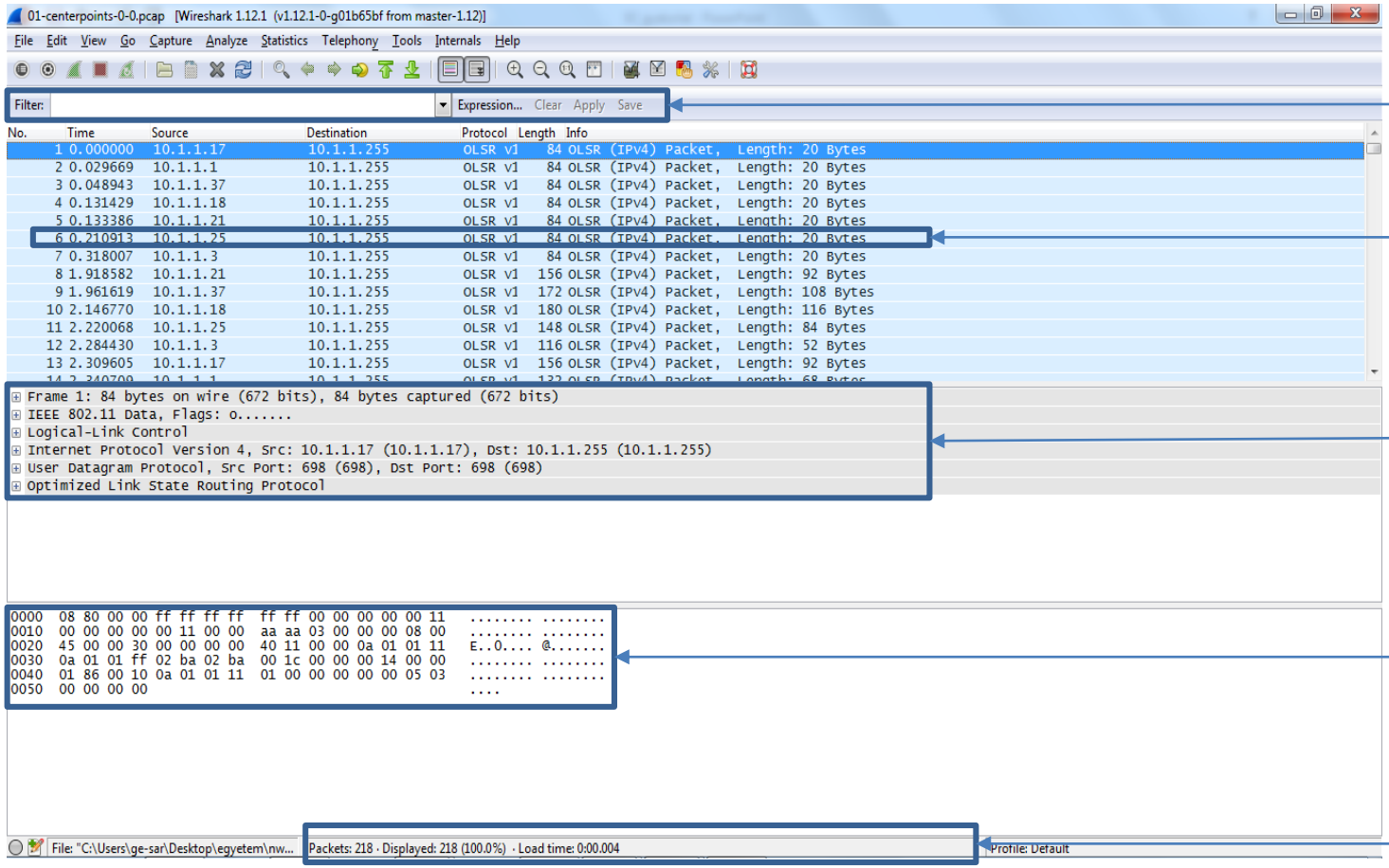
Tcpdump – pcap fájlba mentés

```
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
```

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



Szűrők definiálására alkalmas input eszközök

Csomag összefoglaló nézete

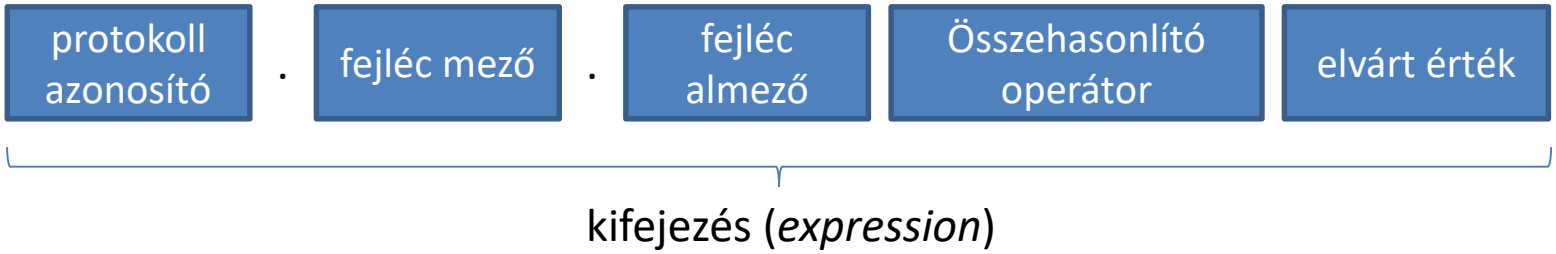
Kiválasztott csomag hierarchikus nézet

Kiválasztott csomag bájt-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 folyamát. 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 folyamát. 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!

portok, kódolás

KOMMUNIKÁCIÓS CSATORNA I.

Port	Service name	Transport protocol
20, 21	File Transfer Protocol (FTP)	TCP
22	Secure Shell (SSH)	TCP and UDP
23	Telnet	TCP
25	Simple Mail Transfer Protocol (SMTP)	TCP
50, 51	IPSec	
53	Domain Name System (DNS)	TCP and UDP
67, 68	Dynamic Host Configuration Protocol (DHCP)	UDP
69	Trivial File Transfer Protocol (TFTP)	UDP
80	HyperText Transfer Protocol (HTTP)	TCP
110	Post Office Protocol (POP3)	TCP
119	Network News Transport Protocol (NNTP)	TCP
123	Network Time Protocol (NTP)	UDP
135-139	NetBIOS	TCP and UDP
143	Internet Message Access Protocol (IMAP4)	TCP and UDP
161, 162	Simple Network Management Protocol (SNMP)	TCP and UDP
389	Lightweight Directory Access Protocol	TCP and UDP
443	HTTP with Secure Sockets Layer (SSL)	TCP and UDP
989, 990	FTP over SSL/TLS (implicit mode)	TCP
3389	Remote Desktop Protocol	TCP and UDP

Pár neves port

A port jelenti a kommunikáció végpontját, azaz bemeneti-kimeneti "kapuk", melyeken folyik az adatátvitel.

Port számok

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

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

Feladat: port szolgáltatások

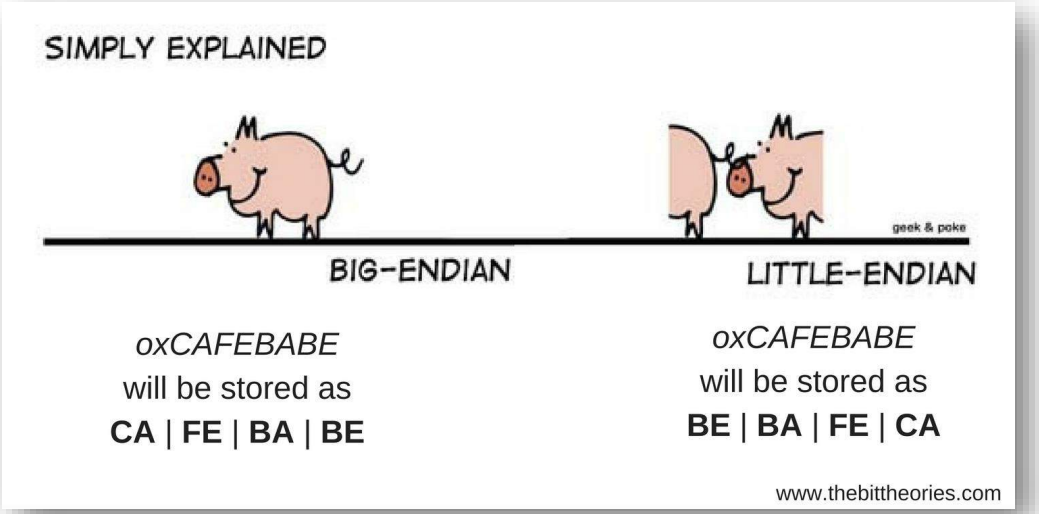
- Írassuk ki a 1..100-ig a portokat és a hozzájuk tartozó protokollokat!

- Ha nem tartozik hozzá, írajuk ki saját üzenetet, küldönben pedig a port nevét.

Bájtsorrendek

00000000 00000000 00000100 00000001

Address	Big-Endian representation of 1025	Little-Endian representation of 1025
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000



- Nézzük meg, hogy a saját gépünk milyen kódolást használ! sys.byteorder
- A hálózati bájt sorrend big-endian (a magasabb helyi értéket tartalmazó bájt van elől)
- 16 és 32 bites pozitív számok kódolása
  - htons(), htonl() – host to network short / long
  - ntohs(), ntohl() – network to host short / long

python socket, TCP

KOMMUNIKÁCIÓS CSATORNA II.

Pyton 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)
```

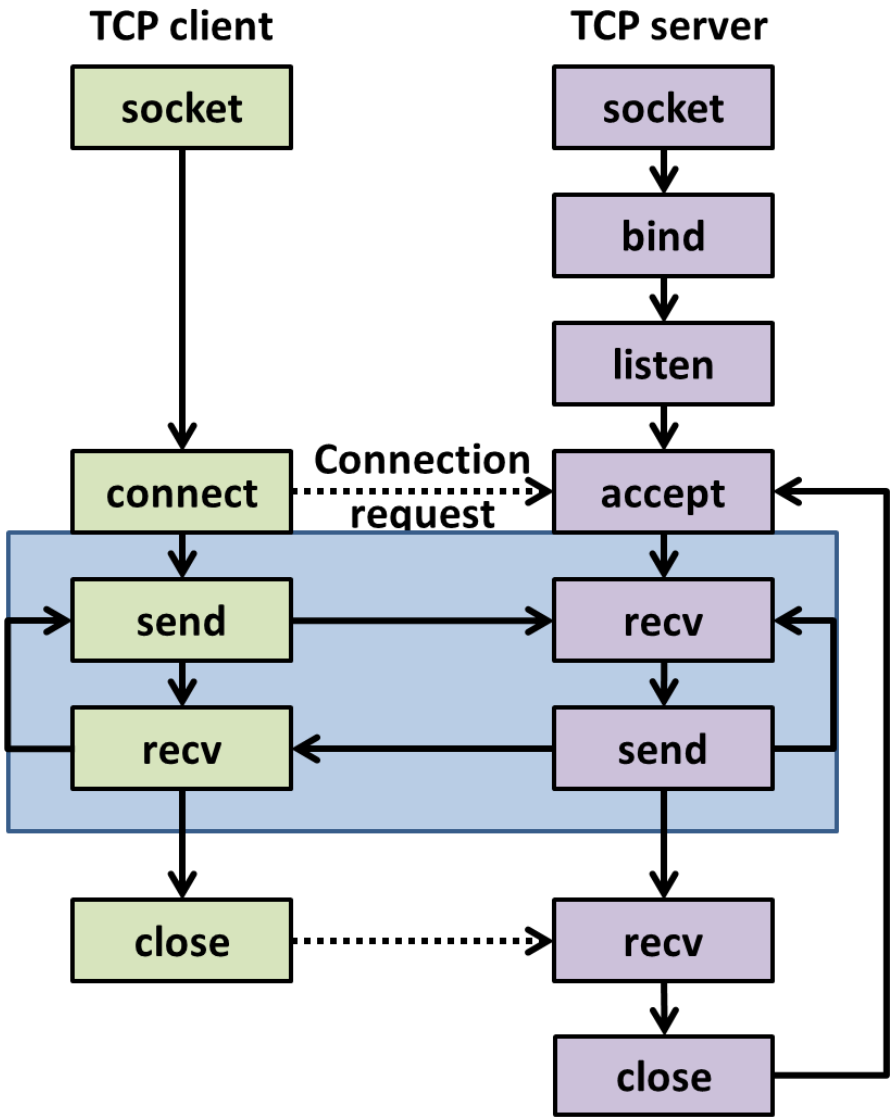
- gethostbyaddr()

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

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

TCP

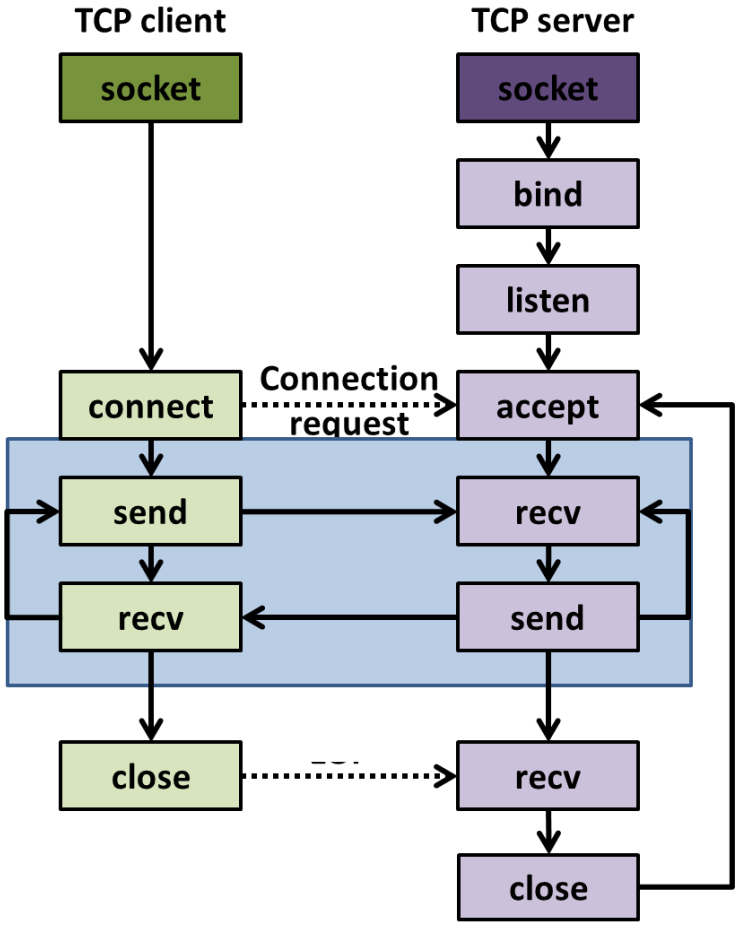

TCP client
TCP server
socket
socket
bind
listen
connect
Connection request
accept
send
recv
recv
send
close
recv
close

Socket leíró beállítása

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

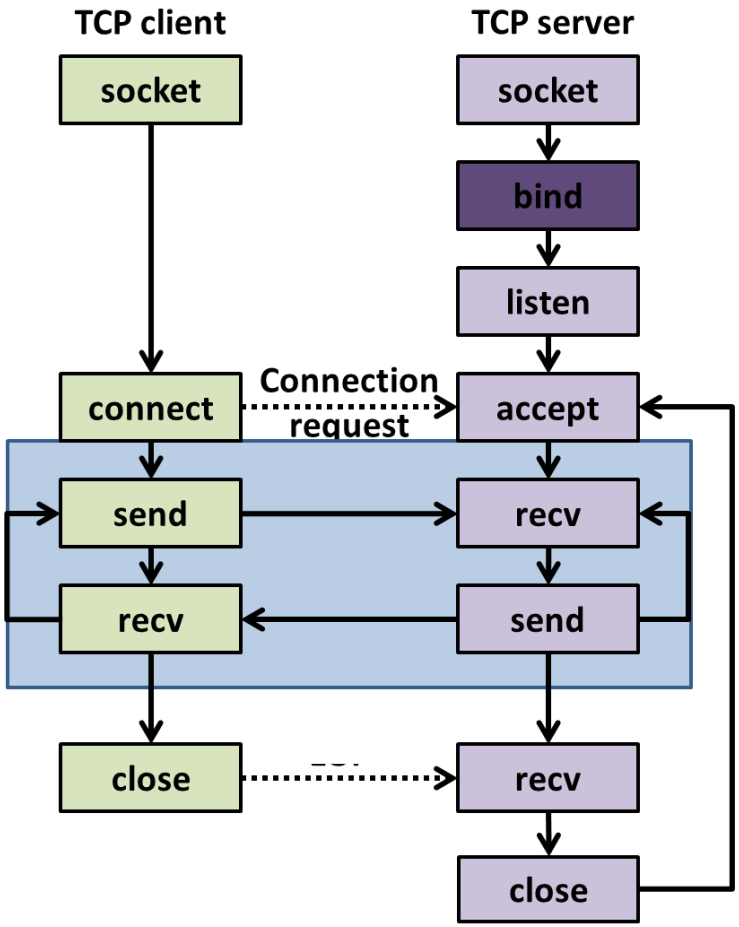
(alapértelmezett protokoll lesz)

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



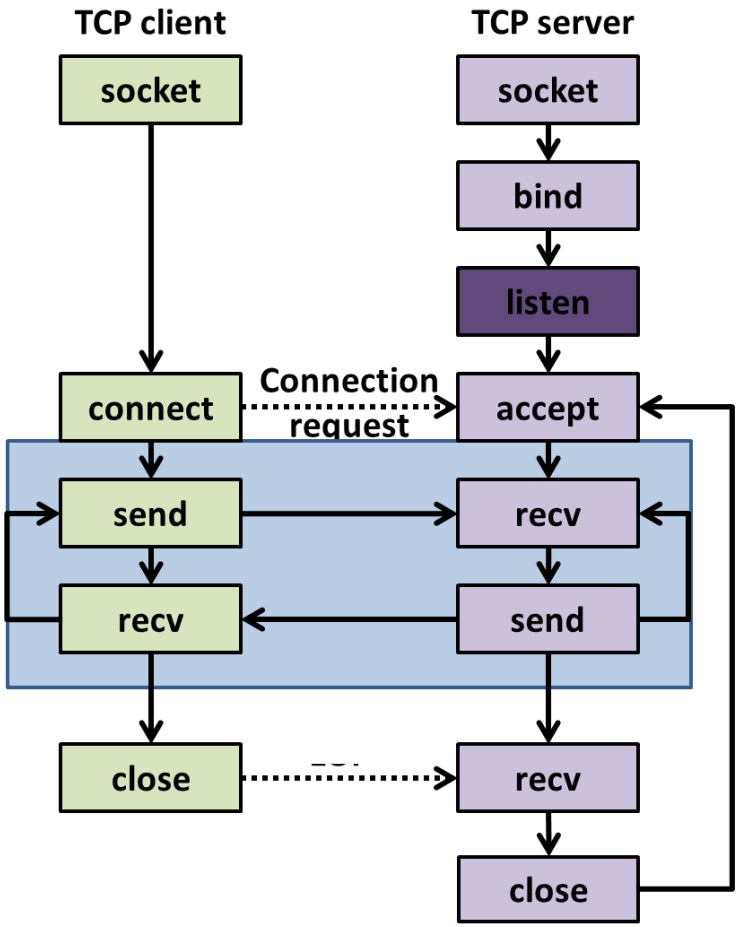
Bindolás

- 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



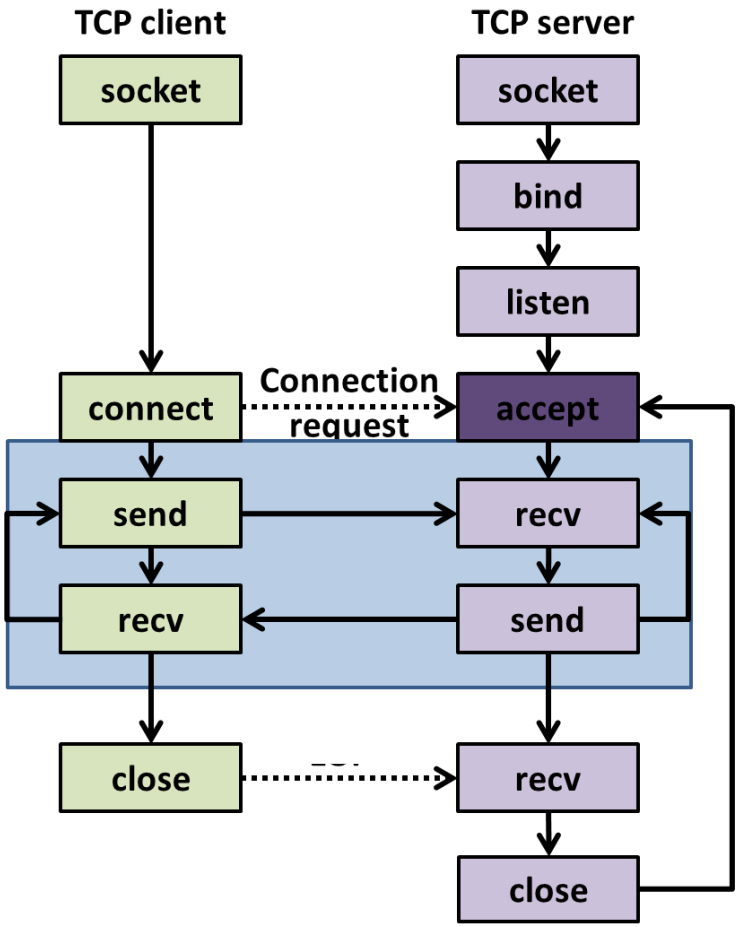
Listen

- socket.socket.**listen**(*backlog*)
- A socket objektum metódusa
- *backlog* : egy egész szám, ennyi kapcsolódási igény várakozhat a sorban



Accept

- socket.socket.**accept**()
- A socket objektum metódusa
- A szerver elfogadhatja a kezdeményezett kapcsolatokat
- visszatérési érték: egy tuple,
  - amelynek az első eleme egy **új socket objektum** a kapcsolaton keresztüli adatküldésre és fogadásra
  - második eleme a kapcsolat túlsó végén lévő cím



TCP

- socket()

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

- bind()

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

- listen()

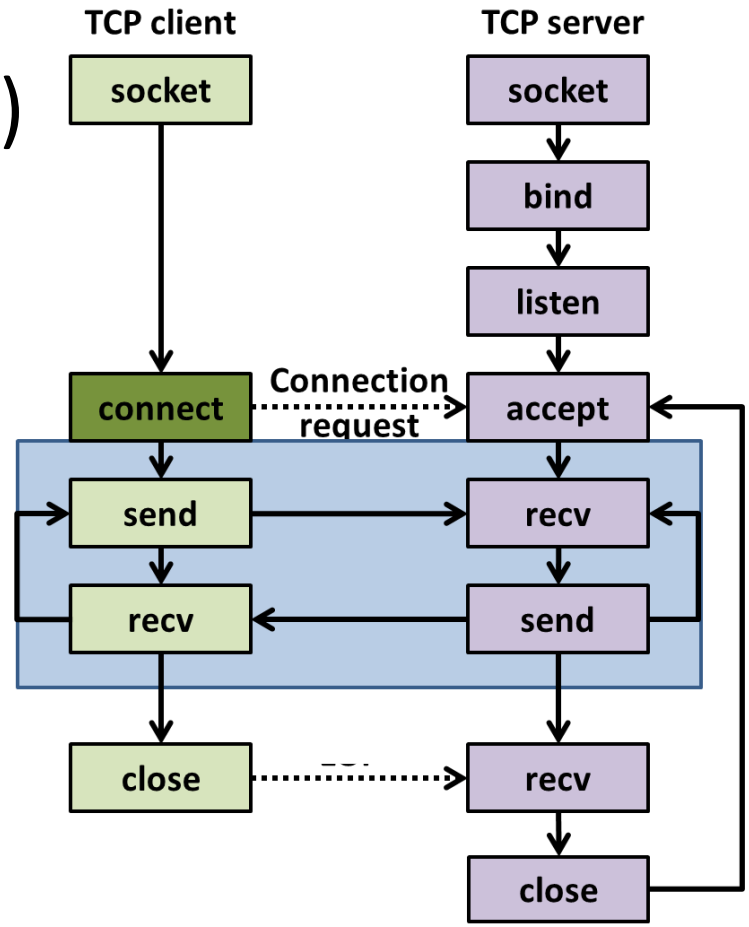
```
sock.listen(1)
```

- accept()

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

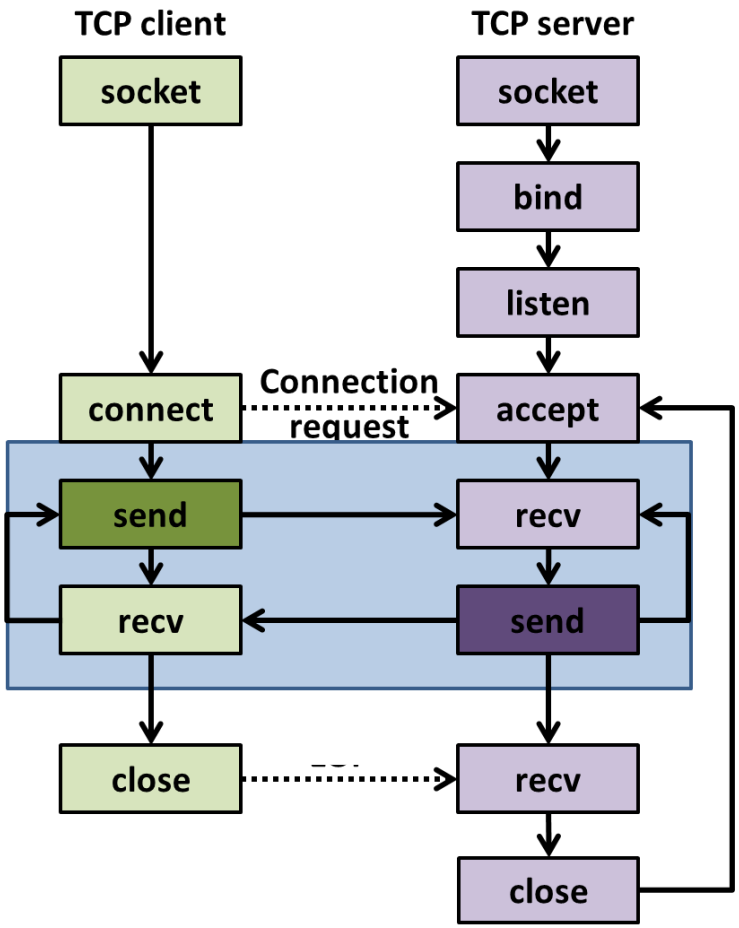
Connect

- socket.socket.**connect**(*address*)
- A socket objektum metódusa
- Kapcsolódás megkezdése egy távoli sockethez az *address* címen (ezt például kezdeményezheti egy kliens)
- Az *address* típusát ld. a **bind** függvénynél



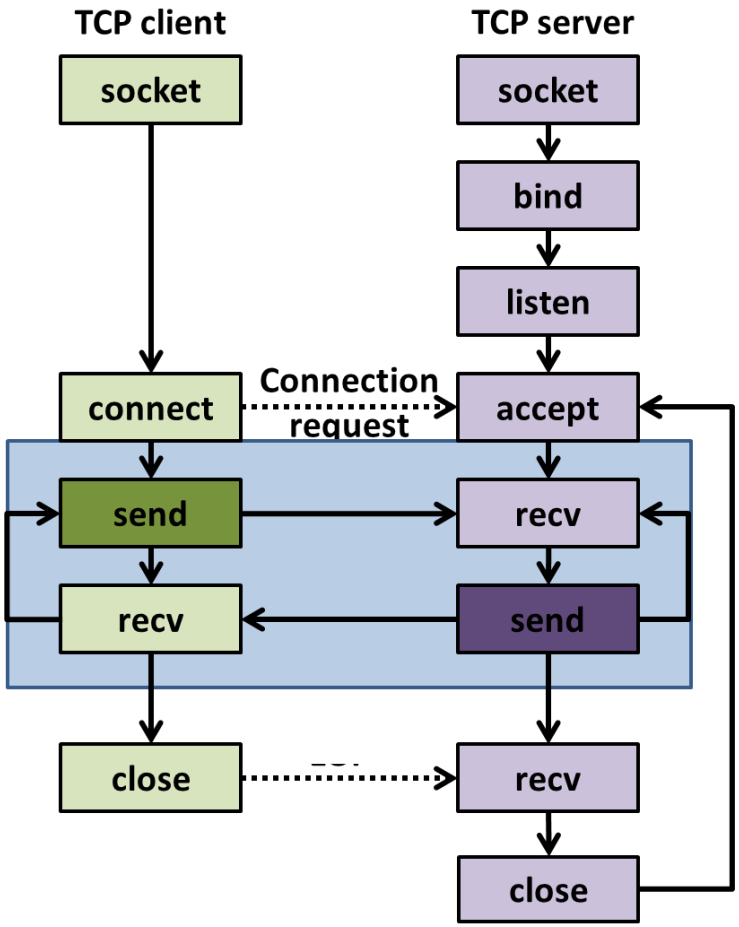
Send

- socket.socket.**send**(*string* *[, flags]*)
- A socket objektum metódusa
- Adatküldés (*string*) a socketnek
- *flags* : 0 (nincs flag meghatározva)
- A socketnek előtte már csatlakozni kellett a távoli sockethez!
- 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



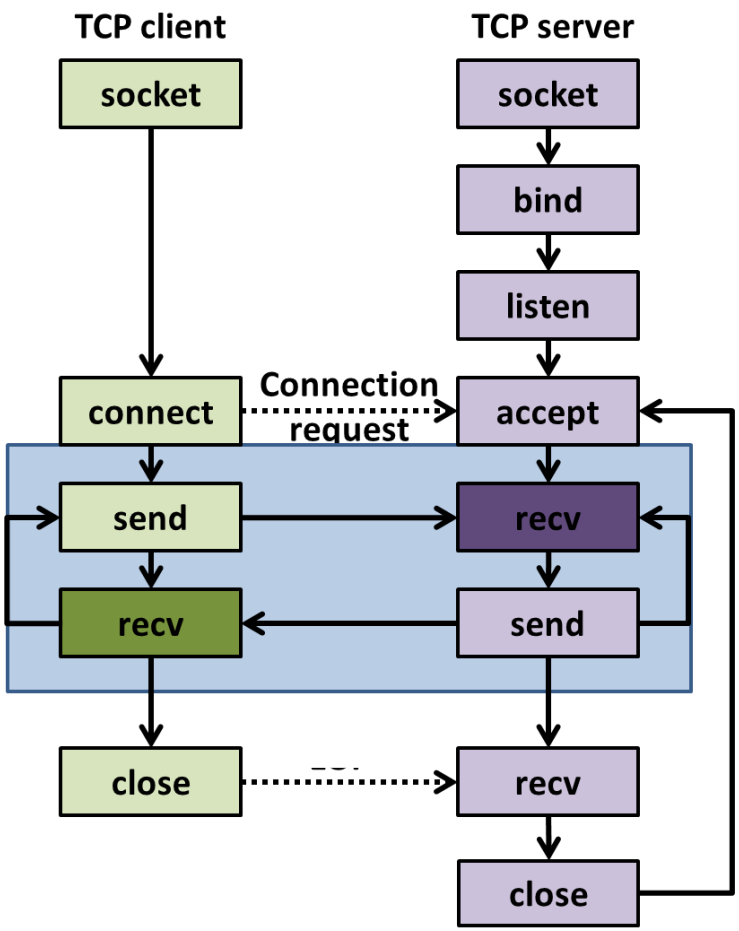
Sendall

- socket.socket.**sendall**(
  string
  [, flags])
- A socket objektum metódusa
- Az előzőhöz hasonló
- A különbség: addig küldi az adatot a *string*-ből, ameddig az összes át nem ment, vagy hiba nem történt (ebben az esetben már nem lehet kideríteni, hogy mennyi adat ment át)
- visszatérési érték: None, ha sikeres volt



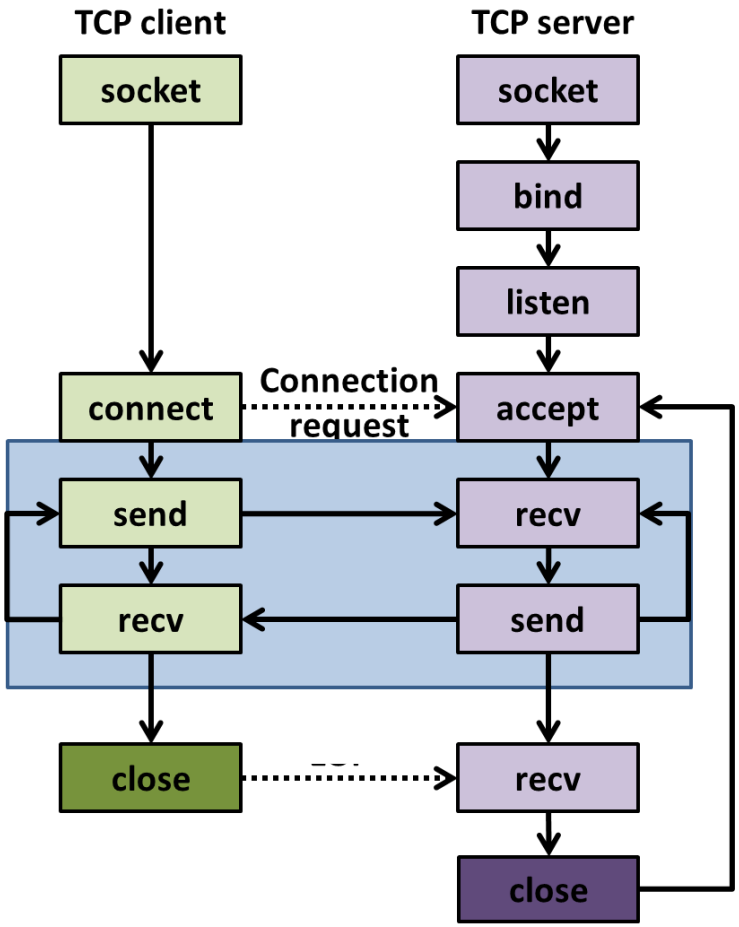
Recv

- socket.socket.**recv**(*bufsize* *[, flags]*)
- A socket objektum metódusa
- Üzenet fogadása
- *bufsize* : a max. adatmennyiség, amelyet egyszerre fogadni fog
- *flags* : 0 (nincs flag meghatározva)
- visszatérési érték: a fogadott adat sztring reprezentációja



Close

- socket.socket.**close**()
- A socket objektum metódusa
- A socket lezárása:
  - az összes további művelet a socket objektumon el fog bukni
  - a túlsó végpont nem fog több adatot kapni
  - ez el fogja engedni a kapcsolathoz tartozó erőforrásokat, **de** nem feltétlen zárja le azonnal (ha erre szükség van, akkor érdemes **shutdown** hívást a **close** elé tenni)



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: Hello

- 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 tetszőlegeset kapjunk az oprendszertől! (sys.argv[1])

Fájl átvitel

- fájl bináris megnyitása

```
with open („input.txt”, „rb”) as f:
        …
```

- read(x)
  - x byte beolvasása (ha binárisra van megnyitva)
  - x karakter beolvasása (ha file olvasásra van megnyitva)

```
…
f.read(128)    #128 byte-ot fog beolvasni
```

„When size is omitted or negative, the entire contents of the file will be read and returned; it’s your problem if the file is twice as large as your machine’s memory. „ - python.org

Struktúraküldése

- Binárissá alakítjuk az adatot

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

- Visszalakítjuk a kapott üzenetet

```
import struct
unpacker = struct.Struct('i 2s f’)          #struct.calcsize(unpacker)
unpacked_data = unpacker.unpack(data)
```

- megj.: integer 1 – 4 byte, stringként 1 byte, azaz hatékonyabb stringként átküldeni.
- https://docs.python.org/3/library/struct.html

Struktúra jellemzői

- Mire kell figyelni?
  - A Struct formátumnál az „Xs” (pl. „2s”) X db. bájtból álló **bájtliterált** jelent (pl. **b'abc'**)

```
import struct
values = (1, 'ab', 2.7)
packer = struct.Struct(,i 2s f')
packed_data = packer.pack(*values)
# HIBA: struct.error: argument for 's' must be a bytes object
# JÓ megoldás:
values = (1, b'ab', 2.7) # vagy values = (1, 'ab'.encode(), 2.7)
…
```

Struktúra jellemzői

- Mire kell figyelni?
  - A struktúra mérete byte-ban:

```
import struct
packer = struct.Struct(,i 2s f')
print(struct.calcsize(,i 2s f '))
print(packer.size)
# 12
```

    - i: int size = 4, 2s: 2 bytes, f: float size = 4, 4+2+4 ≠ **12** ???
  - Az int/float-ot úgy igazítja, hogy a kezdő pozíciója 4-gyel osztható legyen

Struktúra

Character	Byte order
`@`	native
`=`	native
`<`	little-endian
`>`	big-endian
`!`	network (= big-endian)

Format	C Type	Python type	Standard size
`x`	pad byte	no value	
`c`	char	bytes of length 1	1
`b`	signed char	integer	1
`B`	unsigned char	integer	1
`?`	_Bool	bool	1
`h`	short	integer	2
`H`	unsigned short	integer	2
`i`	int	integer	4
`I`	unsigned int	integer	4
`l`	long	integer	4
`L`	unsigned long	integer	4
`q`	long long	integer	8
`Q`	unsigned long long	integer	8
`n`	`ssize_t`	integer	
`N`	`size_t`	integer	
`e`	(6)	float	2
`f`	float	float	4
`d`	double	float	8
`s`	char[]	bytes	
`p`	char[]	bytes	
`P`	void*	integer	

Fájlkezelés

- SEEK: a fájlobjektumnak az aktuális pozíciója:

```
with open('alma.txt', 'r') as f:
  sor = f.readline()
  print('jelenlegi sor', sor)     # jelenlegi sor 1. sor

  sor = f.readline()
  print('jelenlegi sor', sor)     # jelenlegi sor 2. sor

  f.seek(0, 0)                    # f.seek(offset, whence)

  sor = f.readline()
  print('jelenlegi sor', sor)     # jelenlegi sor 1. sor
```

- offset: olvasás/írás mutató pozíciója a fájlban
- (whence: alapért. 0: abszolút poz., 1: relatív aktuális poz.-hoz, 2: rel. a fájl végéhez)

Fájlkezelés

- Bináris fájl és a SEEK

```
import struct

packer = struct.Struct('i3si')

with open('dates.bin', 'wb') as f:
  for i in range(5):
    values = (2020+i, b'jan', 10+i)
    packed_data = packer.pack(*values)
    f.write(packed_data)

with open('dates.bin', 'rb') as f:
  f.seek(packer.size*4)
  data = f.read(packer.size)
  print(packer.unpack(data))

### output: (2024, b'jan’, 14)
```

Bytesorozat vs string

- String → Byte sorozat

```
import struct
str = „hello”
str.encode()          # b’hello’

Struct.pack(’8s’,str)        #b’hello\x00\x00\x00’
```

- Byte sorozta → String

```
import struct
d = Struct.pack(’8s’,str)    #b’hello\x00\x00\x00’

d.encode().strip(’\x00’)     #’hello’
```

Feladat: Számológép

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