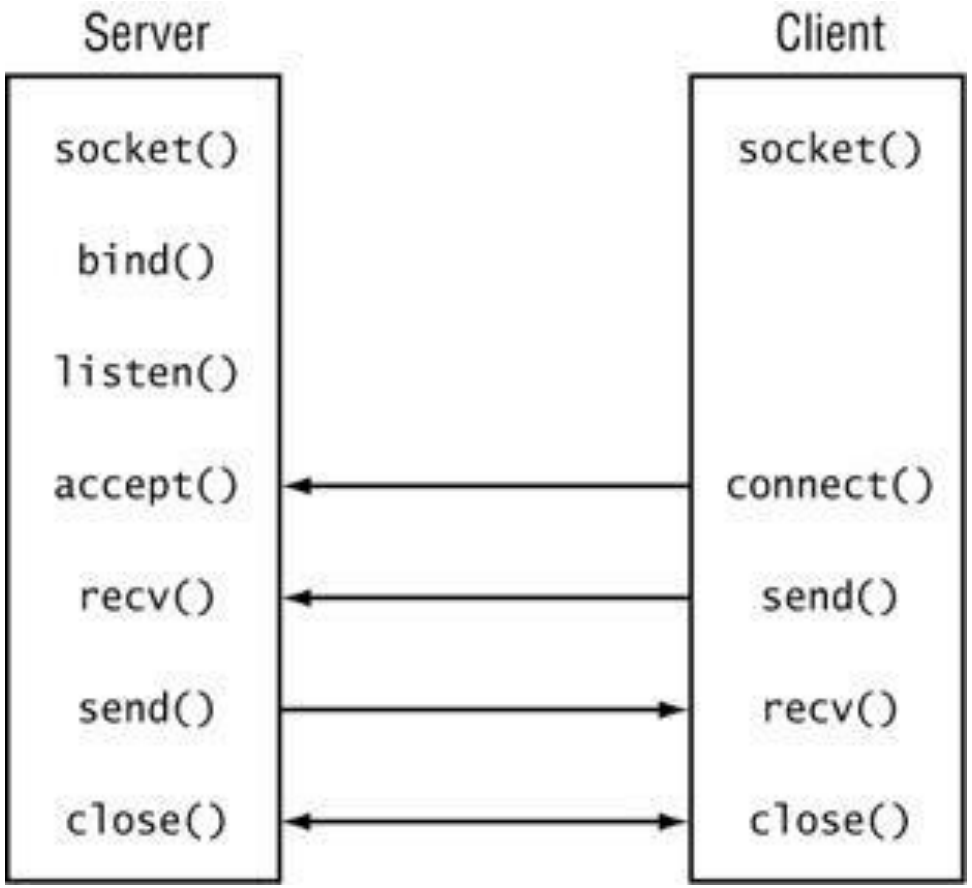
Számítógépes Hálózatok

4. gyakorlat

TCP


Server
Client
socket()
bind()
listen()
accept()
recv()
send()
close()
socket()
connect()
send()
recv()
close()

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)
```

Struktúraküldése

- Binárissá alakítjuk az adatot

```
import struct
values = (1, ”ab”.encode(), 2.7)          #python2: nem kell encode()
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')
unpacked_data = unpacker.unpack(data)
```

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

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.

Select

- setblocking() or settimeout()

```
connection.setblocking(0)      # or connection.settimeout(1.0)
```

- select()

```
inputs = [ server ]
outputs = [ ]
timeout=1
readable, writable, exceptional = select.select(inputs, outputs, inputs,timeout)
…
for s in readable:
      if s is server:    #new client connect
            client, client_addr = s.accpet()
            inputs.append(client)
      else:
            ….          #handle client
```

Feladat – Számológép II.

- Alakítsuk át úgy a számológép szervert, hogy egyszerre több klienssel is képes legyen kommunikálni! Ezt a select függvény segítségével tegye!

- Alakítsuk át a kliens működését úgy, hogy ne csak egy kérést küldjön a szervernek, hanem csatalakozás után 5 kérés-válasz üzenetváltás történjen, minden kérés előtt 2 mp várakozással (time.sleep(2))! A kapcsolatot csak a legvégén bontsa a kliens!

Feladat - Chat

- Készítsünk egy chat alkalmazást, amelyen a chat szerverhez csatlakozott kliensek képes beszélni egymással!
- A szerver szerepe, hogy a kliensektől jövő üzenetet minden más kliensnek továbbítja névvel együtt: [<név>] <üzenet> ; pl. [Józsi] Kék az ég!
- A kliensek a szervertől jövő üzeneteket kiírják a képernyőre.

Házi feladat – 2 pont

- Készítsünk egy barkóba alkalmazást. A szerver legyen képes kiszolgálni több klienst. A szerver válasszon egy egész számot 1..100 között véletlenszerűen. A kliensek próbálják kitalálni a számot.
- A kliens üzenete egy összehasonlító operátor: <, >, = és egy egész szám, melyek jelentése: kisebb-e, nagyobb-e, mint az egész szám, illetve rákérdez a számra. A kérdésekre a szerver Igen/Nem/Nyertél/Kiestél/Vége üzenetekkel tud válaszolni. A Nyertél és Kiestél válaszok csak a rákérdezés (=) esetén lehetségesek.
- Ha egy kliens kitalálta a számot, akkor a szerver minden újabb kliens üzenetre az „Vége” üzenetet küldi, amire a kliensek kilépnek. A szerver addig nem választ új számot, amíg minden kliens ki nem lépett.
- • Nyertél, Kiestél és Vége üzenet fogadása esetén a kliens bontja a kapcsolatot és terminál. Igen/Nem esetén folytatja a kérdezgetést.
- A kommunikációhoz TCP-t használjunk!
- Folytatás a következő oldalon!

Házi feladat – 2 pont

- A kliens logaritmikus keresés segítségével találja ki a gondolt számot. A kliens tudja, hogy milyen intervallumból választott a szerver.
- AZAZ a kliens NE a standard inputról dolgozzon.
- Minden kérdés küldése előtt véletlenszerűen várjon 1-5 mp-et. Ezzel több kliens tesztelése is lehetséges lesz.
- Folytatás a következő oldalon!

Házi feladat – 2 pont

- Üzenet formátum:
  - Klienstől: bináris formában **egy db karakter, 32 bites egész szám**
    A karakter lehet: <: kisebb-e, >: nagyobb-e, =: egyenlő-e
  - Szervertől: ugyanaz a bináris formátum, de a számnak nincs szerepe (bármi lehet)
    A karakter lehet: I: Igen, N: Nem, K: Kiestél, Y: Nyertél, V: Vége
- Fájlnevek és parancssori argumentumok:
- Szerver: **server.py** <bind_address> <bind_port> # A bindolás során használt pár
- Kliens: **client.py** <server_address> <server_port> # A szerver elérhetősége

- Beadási határidő: **BEAD rendszerben**

VÉGE