Telekommunikációs Hálózatok

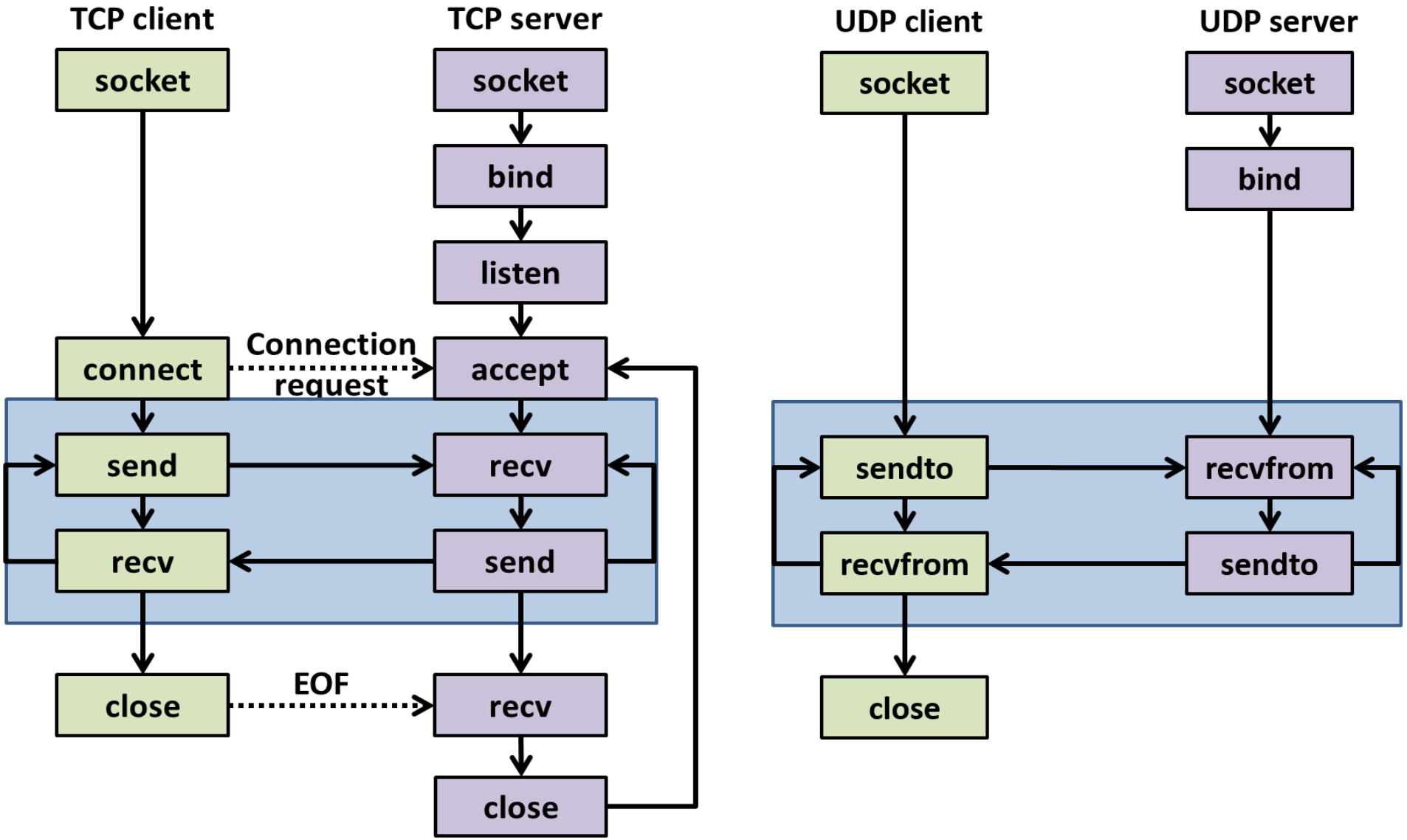
6. gyakorlat

PYTHON SOCKET - UDP

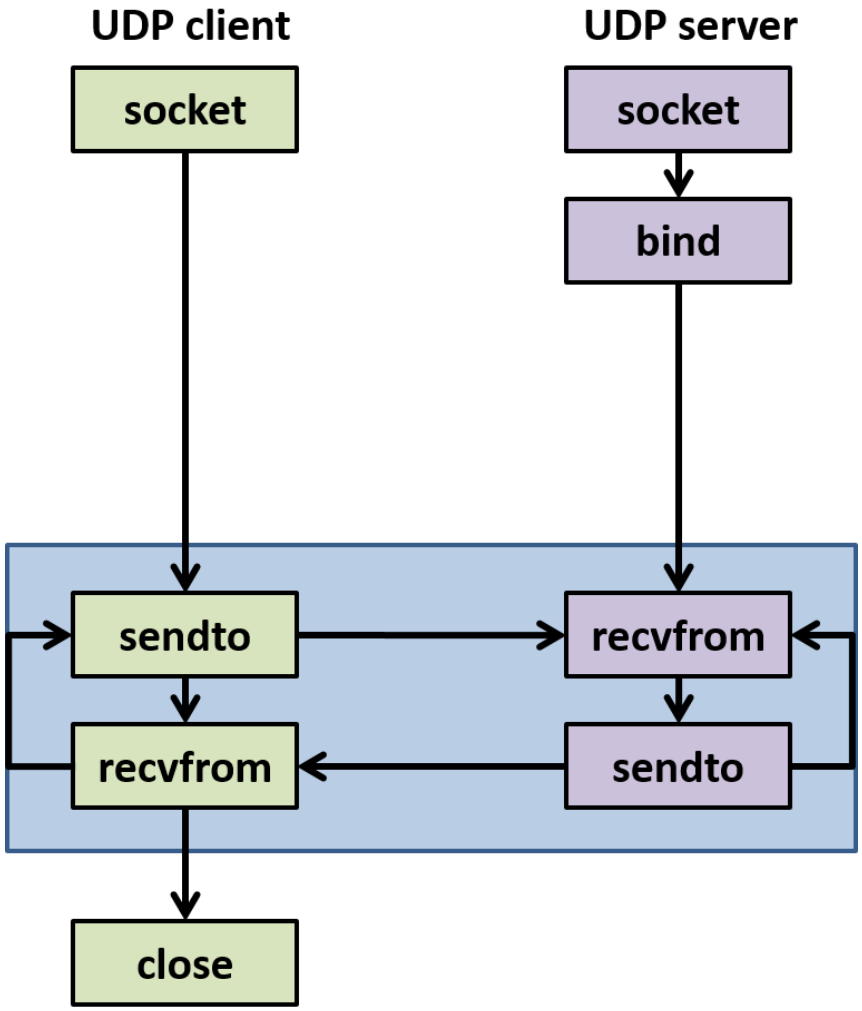
A kommunikációs csatorna kétféle típusa

- 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 VS UDP


TCP client
TCP server
socket
socket
bind
listen
connect
Connection request
accept
send
recv
recv
send
close
EOF
recv
close
UDP client
UDP server
socket
socket
bind
sendto
recvfrom
recvfrom
sendto
close

UDP

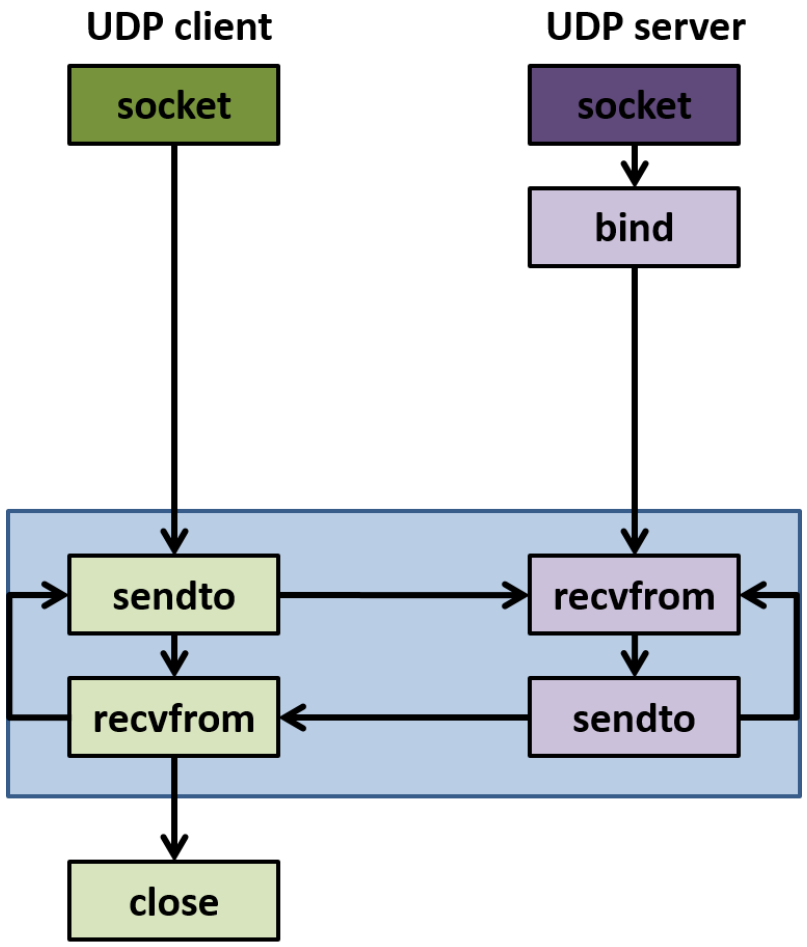

UDP client
UDP server
socket
socket
bind
sendto
recvfrom
recvfrom
sendto
close

Socket leíró beállítása

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

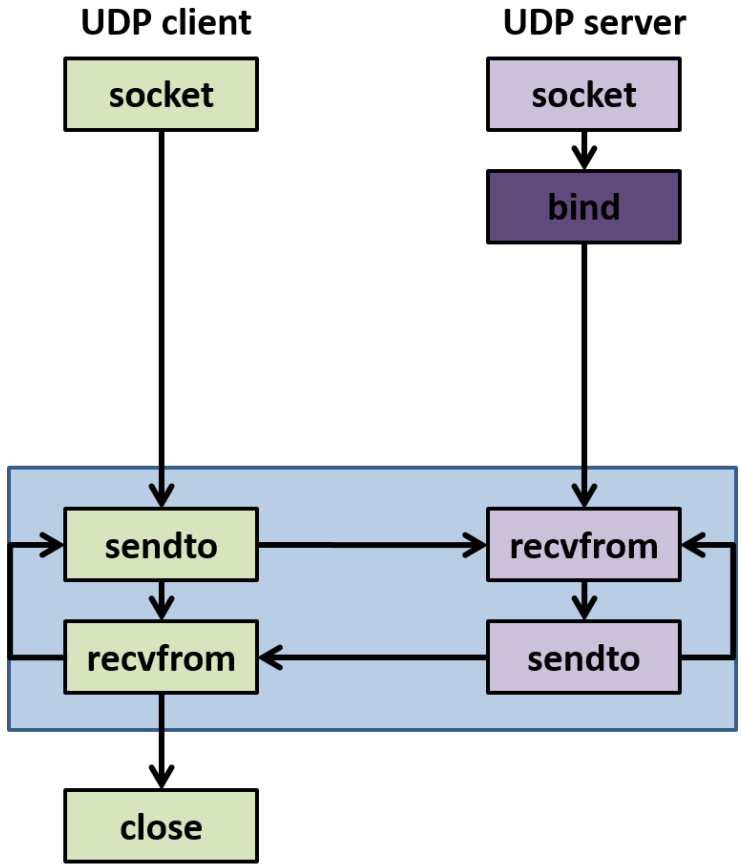
(alapértelmezett protokoll lesz)

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



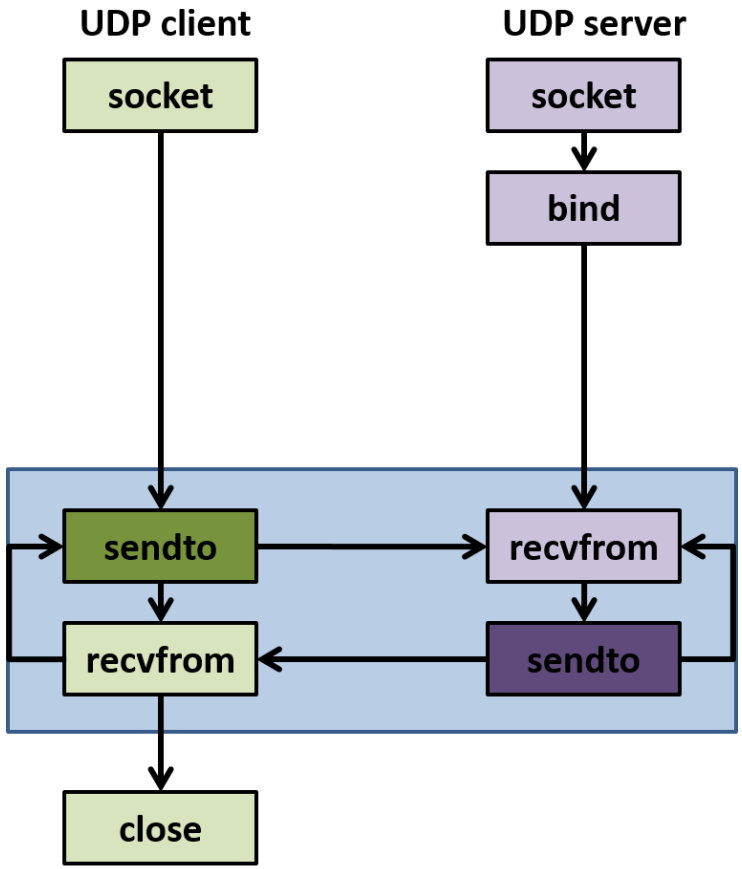
Bindolás – ismétlés

- 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



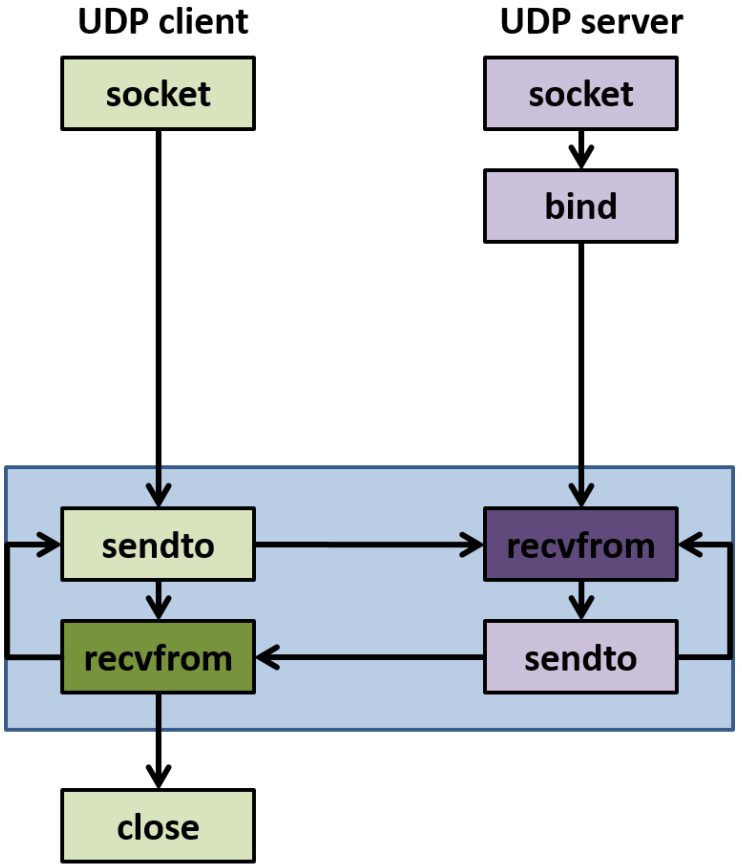
sendto

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



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 (*string*, *address*) tuple, ahol a fogadott adat sztring 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: Hello UDP felett

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

Feladat - 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!

Szorgalmi feladat otthonra – 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.

VÉGE
KÖSZÖNÖM A FIGYELMET!