

Számítógépes Hálózatok

5. gyakorlat

PYTHON ALAPOK V.

Socket programozás, UDP

Óra eleji kisZH

- Elérés:
 - <https://canvas.elte.hu>

☰ 2018/19/1 XK85DZ-IP-08abcSZHG - Számítógépes hálózatok GY. (BSc,08,A) > Kvízek

2018/19/1

Kezdőlap

Hirdetmények

Feladatok

Fórumok

Értékelések

Résztvevők

Oldalak

Fájlok

Tematika

Tanulási eredmények

Kvízek


Modulok

Beállítások


+ Kvíz

⚙️

▼ Gyakorló kvízek

 Demo kvíz

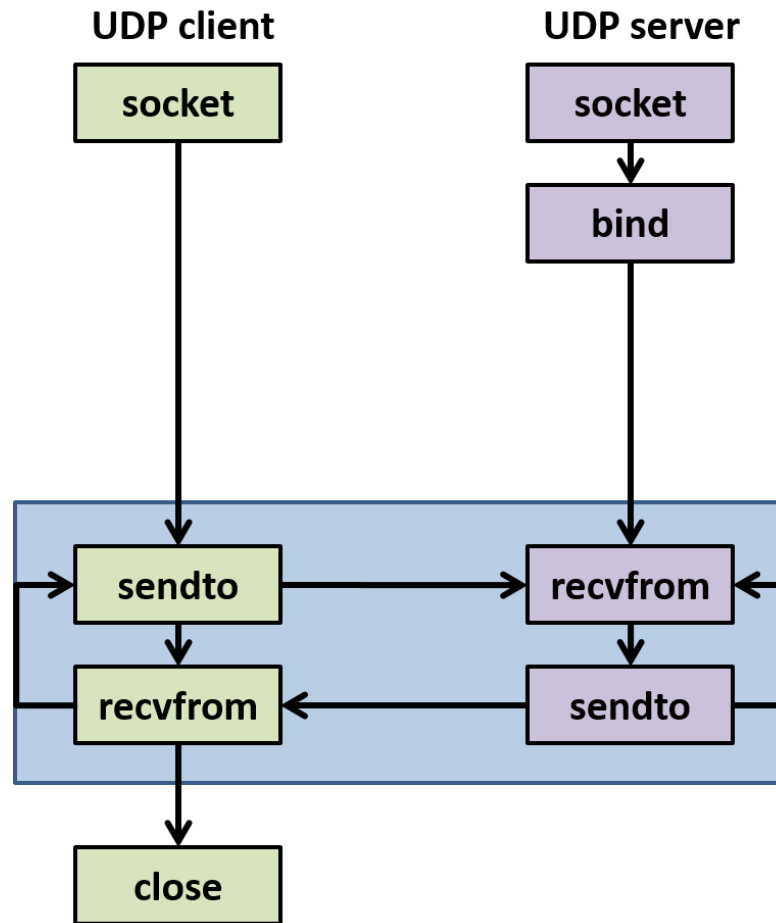
Elérhető Többes határidő | Határidő Többes határidő | 5 pont | 5 kérdés

 ⚙️

A kommunikációs csatorna kétféle típusa

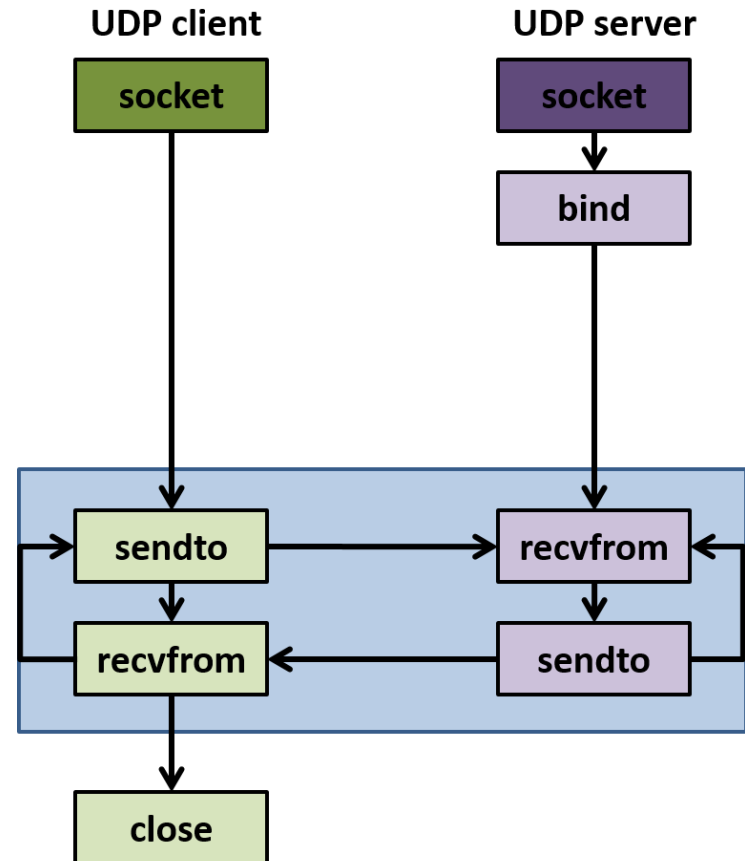
- Kapcsolat-orientált modell (analógia: telefonbeszélgetés)
 - csomagok megérkeznek jó sorrendben
 - ilyen protokoll a TCP
 - kapcsolódó típus: stream socket
- **Kapcsolat-nélküli modell (analógia: postai levelezés)**
 - csomagok nem biztos, hogy sorrend helyesen érkeznek, sőt el is veszhetnek
 - előnye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

UDP



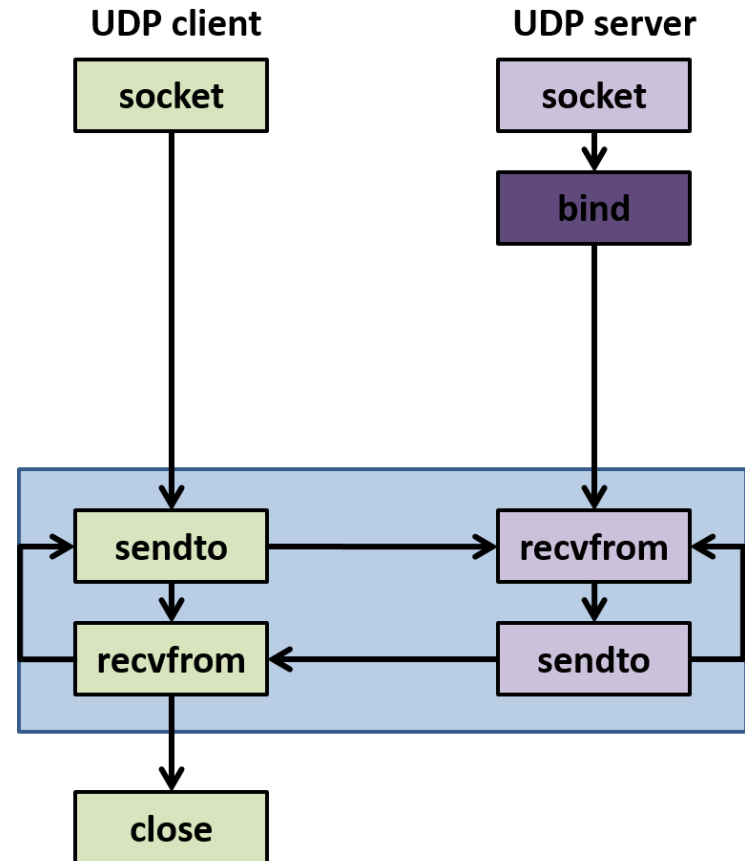
Socket leíró beállítása

- `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



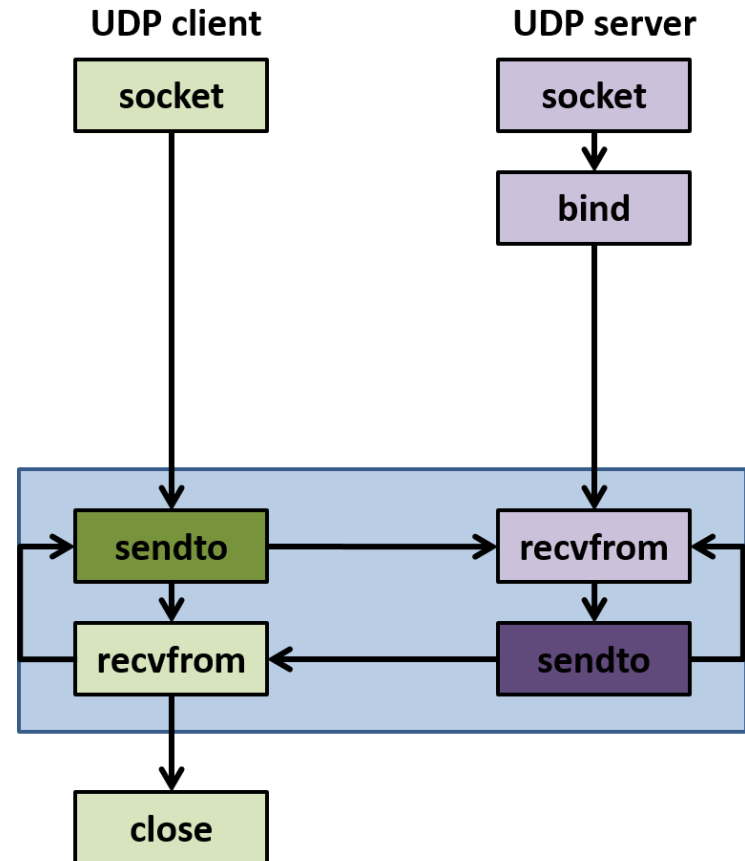
Bindolás – ismételés

- `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



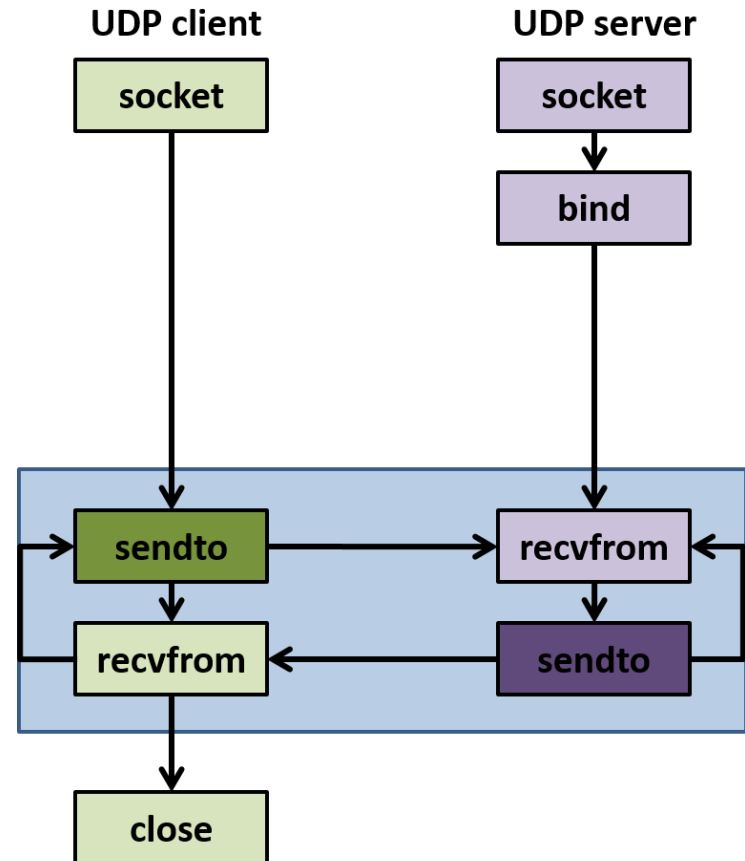
sendto

- `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**



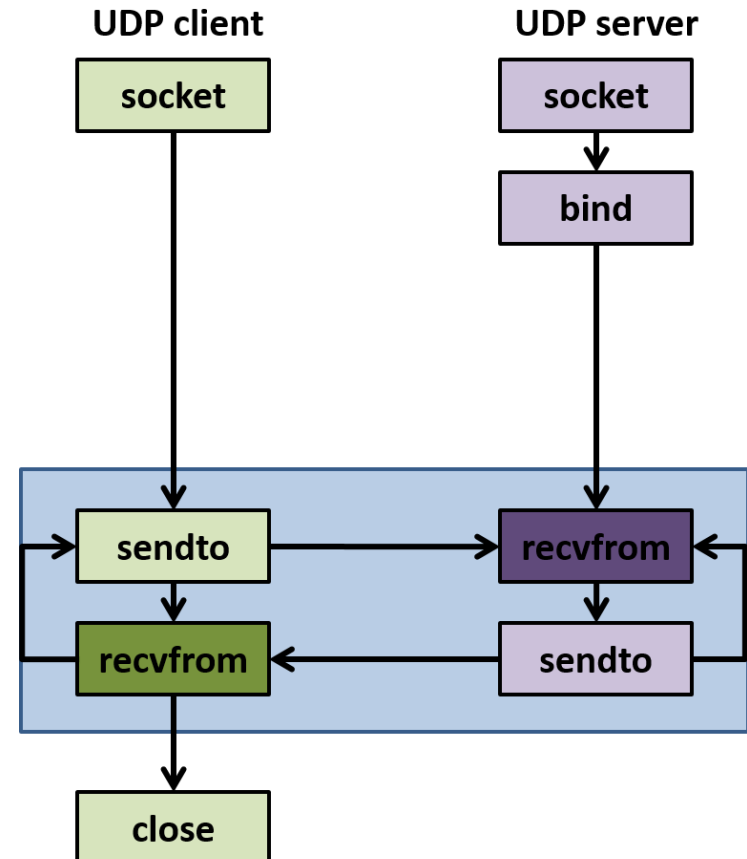
sendto

- **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 (*string*, *address*) tuple, ahol a fogadott adat sztring reprezentációja és az adatküldő socket címe szerepel**



UDP

- `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.

Queue – szálbiztos FIFO konténer

- A Queue python modul egy FIFO implementációt tartalmaz: `Queue.Queue` osztályt, ami megfelelő a többszálúsághoz
- A sor végére a **`put()`** függvénnnyel helyezzük az elemeket
- Az elejéről a **`get()`** függvénnnyel szedjük le,
- vagy a **`get_nowait()`**-tel,
 - amely nem blokkol, azaz nem vár elérhető elemre,
 - és kivételt jön, ha üres a sor

Órai feladat (4 pont)

- Készítsünk egy posta alkalmazást!
- A kliensek elsőre küldenek egy bejelentkező üzenetet a szervernek, amiben az első sztring: "LOGIN", a második sztring a nevük (pl. `sys.argv[1]`). A két sztring legyen "+" jellel összekonkatenálva.
- A szerver mindenkit folyamatosan felvesz a „névjegyzékbe” (= a névhez egy (IP-cím, portszám) azonosítót rendel).
- A kliens oldalon a felhasználó „levelet adhat fel” (= a szabványos bemeneten keresztül üzenhet) valamilyen címzettnek. A címzett és az üzenet legyen "+" jellel összekonkatenálva.
- A szerver Queue-kban tárolja a nevekhez tartozó üzeneteket.

Órai feladat (4 pont)

- Ha a felhasználó "QUERY" ad meg a címzettnél (az üzenet szövege ilyenkor automatikusan töltődjön ki "EMPTY"-vel), akkor a szerver úgy veszi, hogy csak a „postaládáját” akarja megnézni (= megkérdezi a szervert, hogy van-e neki szóló üzenet)
- Ekkor a szerver a kliensnek küldött üzeneteket továbbítja egy hosszú üzenetté összefűzve, üzeneteket "|" jellel elválasztva (a feladó az egyszerűség kedvéért most nem számít), és kiüríti az adott névhez tartozó tárolót. Ha nincs üzenete, akkor csak "EMPTY" üzenetet küld vissza.
- Ha olyan címzett érkezik a szerverhez, amely nem szerepel a névjegyzékben, az üzenetet egyszerűen eldobja.

Órai feladat (4 pont)

```
\Gyak5>python client_gyak5_orai_posta.py A
Addressee: B
Message: valami
Addressee: C
Message: semmi
Addressee: B
Message: meg valami
Addressee: _
```

```
\Gyak5>python client_gyak5_orai_posta.py B
Addressee: A
Message: adat
Addressee: QUERY
Messages: valami|meg valami|
Addressee:
```

```
\Gyak5>python client_gyak5_orai_posta.py C
Addressee: QUERY
Messages: semmi|
Addressee: _
```


VÉGE
KÖSZÖNÖM A FIGYELMET!