

# Számítógépes Hálózatok

## 7. gyakorlat

**MULTICAST, PROXY**

# 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

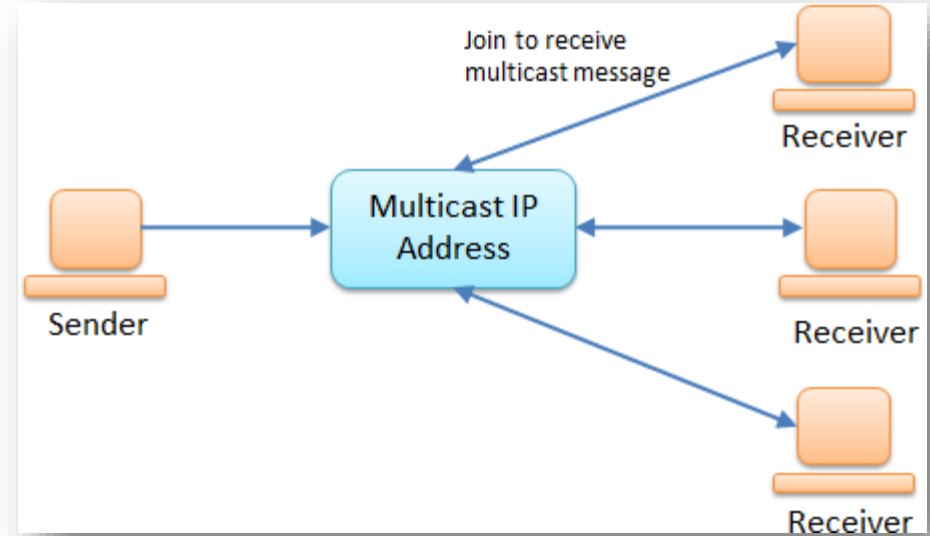
 ⚙️

# Socket beállítása – emlékeztető

- `socket.setsockopt(level, optname, value)`: az adott socket opciót állítja be
- Általunk használt *level* értékek az alábbiak lesznek:
  - `socket.IPPROTO_IP`: jelzi, hogy IP szintű beállítás
  - `socket.SOL_SOCKET`: jelzi, hogy socket API szintű beállítás
- Az *optname* a beállítandó paraméter neve, pl.:
  - `socket.SO_REUSEADDR`: a kapcsolat bontása után a port újrahasznosítása
- A *value* lehet sztring vagy egész szám:
  - Az előbbi esetén biztosítani kell a hívónak, hogy a megfelelő biteket tartalmazza (a struct segítségével)
  - A `socket.SO_REUSEADDR` esetén ha 0, akkor lesz hamis a „tulajdonság”, egyébként igaz

# Multicast

- A pont-pont összeköttetés sokféle kommunikációs igényt ki tud szolgálni



- Ugyanazt az infót külön-külön elküldeni a társaknak nem optimális az erőforrás kihasználtság szempontjából
- A multicast egy időben több végpontnak is tudja szállítani az üzenetet → jobb hatékonyság

# Multicast

- A multicast üzenetek küldésénél **UDP-t** használunk
  - (a TCP végpontok közötti kommunikációs csatornát igényel)
- Egy IPv4 címtartomány van lefoglalva a multicast forgalomra
  - (224.0.0.0-230.255.255.255)
- Ezeket a címeket speciálisan kezelik a routerek és switchek

# Multicast üzenet küldése

- Ha a multicast üzenet küldője választ is vár, nem fogja tudni, hogy hány db. válasz lesz
- → időtúllépési értéket állítunk be, hogy elkerüljük a blokkolást a válaszra történő határozatlan idejű várakozás miatt:

```
sock.settimeout(0.2) # 0.2 sec.
```

# Multicast üzenet küldése

- Továbbá élettídő (Time To Live (TTL)) értéket is szükséges beállítani a csomagon:
  - A TTL kontrollálja, hogy hány db. hálózat kaphatja meg a csomagot
    - „Hop count”: a routerek csökkentik az értékét, ha 0 lesz  
→ eldobják a csomagot
  - A **setsockopt** függvény segítségével majd a **socket.IP\_MULTICAST\_TTL**-t kell beállítani



# Multicast üzenet fogadása

- A fogadó oldalon szükség van arra, hogy a socket-et hozzáadjuk a multicast csoporthoz:
  - A **setsockopt** segítségével az **IP\_ADD\_MEMBERSHIP** opciót kell beállítani
  - A `socket.inet_aton(ip_string)`: az IPv4 cím sztring reprezentációjából készít 32-bitbe csomagolt bináris formátumot
  - Meg lehet adni azt is, hogy a fogadó milyen hálózati interfészen figyeljen, esetünkben most az összesen figyelni fog: `socket.INADDR_ANY`

# Multicast üzenet fogadása

- `socket.INADDR_ANY` a `bind` hívásnál is lehet használni
  - Ott az `"` (üres) sztring reprezentálja → a socket az összes lokális interfészhez kötve lesz
- Nem mindenhol tudunk kötni egy multicast címre
  - Nem minden platform támogatja, a Windows az egyik ilyen
  - Ilyenkor: „`socket.error: [Errno 10049] The requested address is not valid in its context`” hiba jön
  - Kénytelenek vagyunk ebben az esetben az **`INADDR_ANY`**-t használni, viszont az fontos, hogy a portnak **a szerver által használt portot adjuk meg**
  - (localhost-tal nem működne, mert akkor a multicast hálózatot nem tudjuk elérni)

# Példa hívások multicast-nál

- `setsockopt()` (sender)

```
ttl = struct.pack('b', 1) # '\x01'  
sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, ttl)
```

- `socket` hozzávétele a multicast grouphoz (recv)

```
multicast_group = '224.3.29.71'  
group = socket.inet_aton(multicast_group) # '\xe0\x03\x1dG'  
mreq = struct.pack('4sL', group, socket.INADDR_ANY) # '\xe0\x03\x1dG\x00\x00\x00\x00'  
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
```

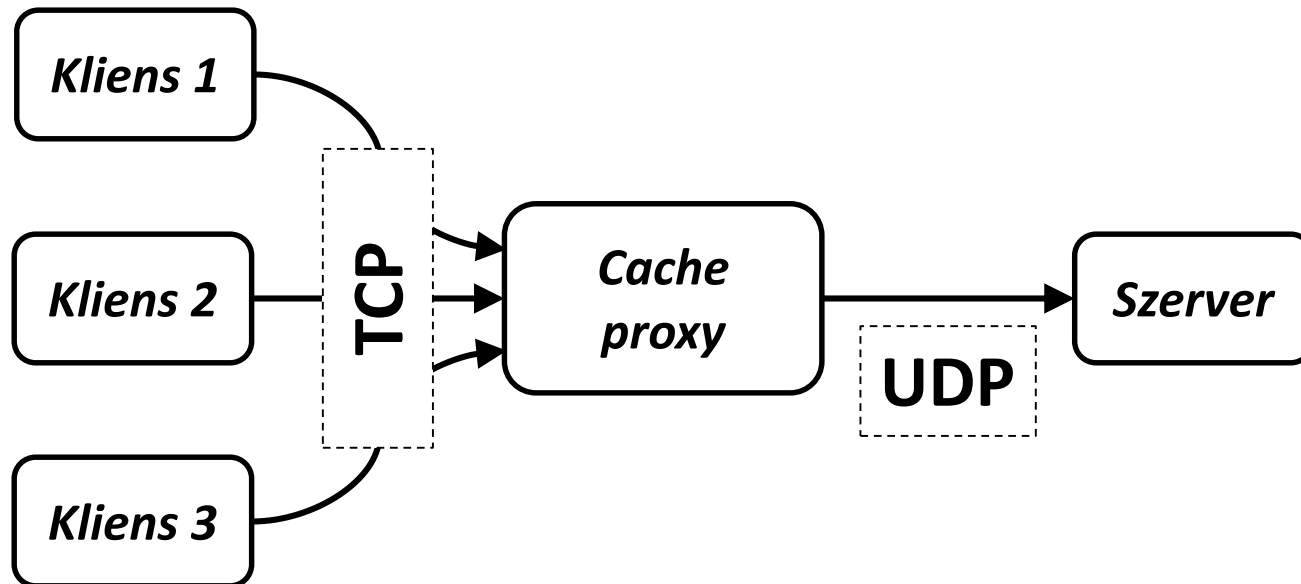
# Feladat 1

- Készítsünk egy multicast fogadó és küldő alkalmazást!
- Először csak a saját gépünkön fusson a küldő és a fogadó is (TTL értéke 1)
- Majd a tanári gépen fogjuk futtatni a küldő, és mindenki feliratkozhat rá (TTL értéke 2)

# Órai feladat (4 pont)

- Készítsünk egy cache proxy alkalmazást, ami a számítógép alkalmazást egészíti ki azzal, hogy ha a cacheben megtalálható az eredmény, akkor nem kérdezi le a szervert.
- A kliensek a proxyra csatlakoznak TCP kapcsolaton keresztül (a select függvény használatával).
  - Elküldik a két számot és az operátort.
  - A cache proxy megnézi, hogy az eredmény megtalálható-e a cache-ben, valamint hogy a tárolt érték aktuális-e. Az érték akkor nem aktuális, ha 1 percnél régebben található meg a cache-ben.
  - Ha nincs eredmény, vagy nem aktuális, akkor megkérdezi a szervert UDP-n keresztül, aki kiszámolja, majd visszaküldi az értéket, amelyet a proxy letárol a cacheben és visszaküldi a kliensnek.

# Órai feladat ábra



**VÉGE**  
**KÖSZÖNÖM A FIGYELMET!**