Telekommunikációs Hálózatok

7. gyakorlat

MapBank zh gyakorlás

- 3 script: naviClient,naviServer,mapBank
- A naviClient 4 számot kap a parancssorról (2 koordináta, óra, perc), ezeket + hard codeolva a Neptun kódodat becsomagolja és UDP-n keresztül elküldi a naviServernek. A naviServer kiírja a kapott infokat a konzolra majd egy TCP kapcsolaton keresztül továbbítja a mapBank-nak.
- A mapBank megnézi, hogy megvan-e neki eltárolva, hogy mennyi idő alatt lehet eljutni a megadott koordinátákra (a kiindulási koordinátáktól most eltekintünk) a megadott időpontban indulva.
- Amennyiben nem, random generál egy értéket amit elküld válaszként, ha igen, akkor a tárolt választ küldi. A naviServer kiírja a konzolra a kapott választ is és továbbítja azt a naviClientnek.

Eddigiek

- Mit tanulmányoztunk eddig?
- TCP
- UDP
- Proxy
- Input módok
- Hashek
- Bytestringek
- Struktúrák
- JSON formátum

Netmask

- Alhálózat címeinek leírása.


```
Address:   192.168.0.1          11000000.10101000 .00000000.00000001
Netmask:   255.255.0.0 = 16     11111111.11111111 .00000000.00000000
Wildcard:  0.0.255.255          00000000.00000000 .11111111.11111111
=>
Network:   192.168.0.0/16       11000000.10101000 .00000000.00000000 (Class C)
Broadcast: 192.168.255.255      11000000.10101000 .11111111.11111111
HostMin:   192.168.0.1          11000000.10101000 .00000000.00000001
HostMax:   192.168.255.254      11000000.10101000 .11111111.11111110
Hosts/Net: 65534                (Private Internet)
```

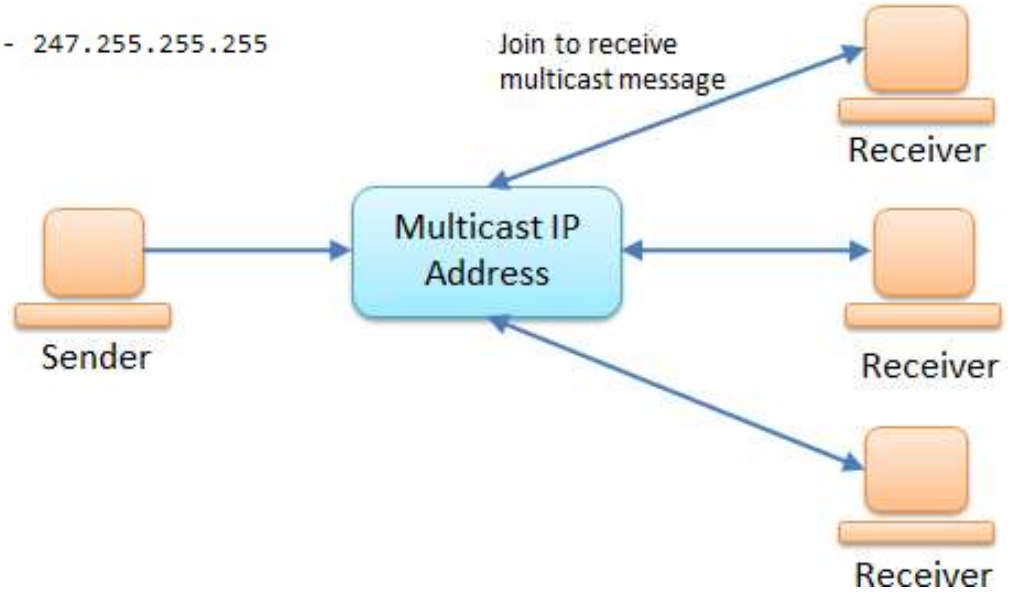
Feladat 3

- Hány cím elérhető a következő netmaskokkal és adjuk meg a minimális és maximális címet:
  - 188.100.22.12/32
  - 188.100.22.12/20
  - 188.100.22.12/10

http://jodies.de/ipcalc

Multicast

```
Bit -->  0                            31          Address Range:
        +-+----------------------------+
        |0|       Class A Address      |       0.0.0.0 - 127.255.255.255
        +-+----------------------------+
        +-+-+--------------------------+
        |1 0|     Class B Address      |     128.0.0.0 - 191.255.255.255
        +-+-+--------------------------+
        +-+-+-+------------------------+
        |1 1 0|   Class C Address      |     192.0.0.0 - 223.255.255.255
        +-+-+-+------------------------+
        +-+-+-+-+----------------------+
        |1 1 1 0|  MULTICAST Address   |     224.0.0.0 - 239.255.255.255
        +-+-+-+-+----------------------+
        +-+-+-+-+-+--------------------+
        |1 1 1 1 0|     Reserved       |     240.0.0.0 - 247.255.255.255
        +-+-+-+-+-+--------------------+
```



Multicast

- setsockopt() (sender)

```
ttl = struct.pack('b', 1)
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)
mreq = struct.pack('4sL', group, socket.INADDR_ANY)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
```

Gyakorló feladat

- Valósítsunk meg egy login szerver-kliens alkalmazást.
- A kliens elküld egy felhasználónév-jelszó párost a szervernek, a szerver összehasonlítja az általa tárolt felhasználónév-jelszó hash párokkal. Amennyiben van pontos egyezés, a szerver „OK” üzenetet küld vissza, amennyiben egyáltalán nem létezik a felhasználó, a szerver tárolja el a párost és „CREATED” üzenetet küldjön vissza. Amennyiben létezik a felhasználó de a jelszó nem egyezik a hash-vel, akkor „INCORRECT” üzenetet küld vissza.
- A szerver az indulásakor fájlból tölti be a felhasználó-hash párokat.
- A kliens a kipróbálandó párokat paranccsori argumentumként kapja.

Gyakorló feladat

- A kliens és a szerver TCP-n keresztül kommunikál.
- A kliens minden próbálkozás eredményét elküldi egy history szervernek UDP-n keresztül.

VÉGE
KÖSZÖNÖM A FIGYELMET!