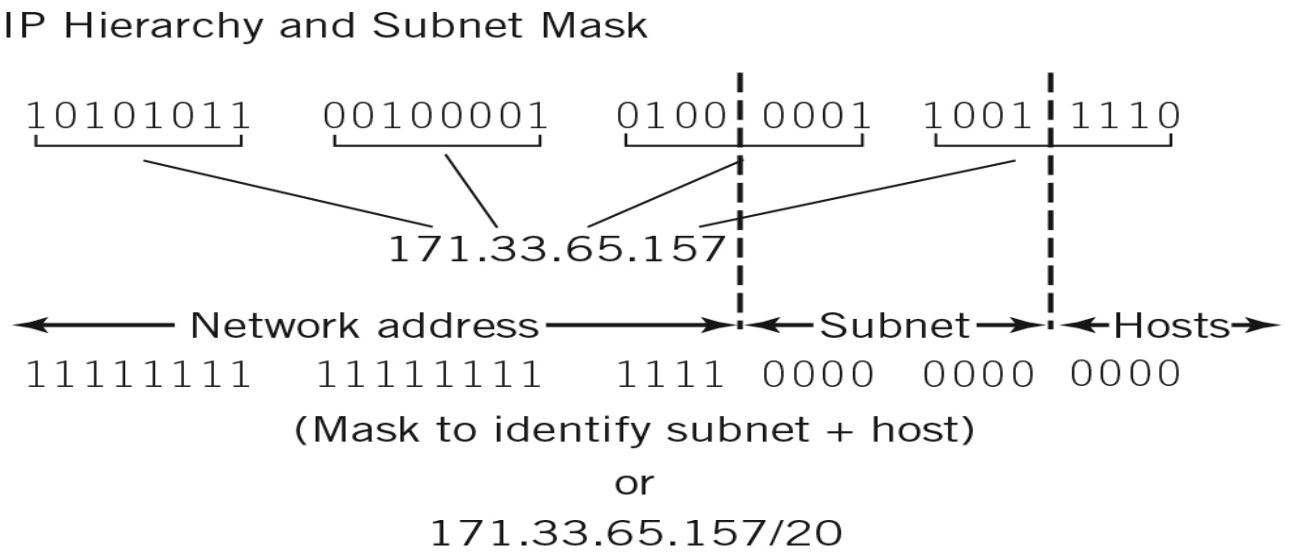
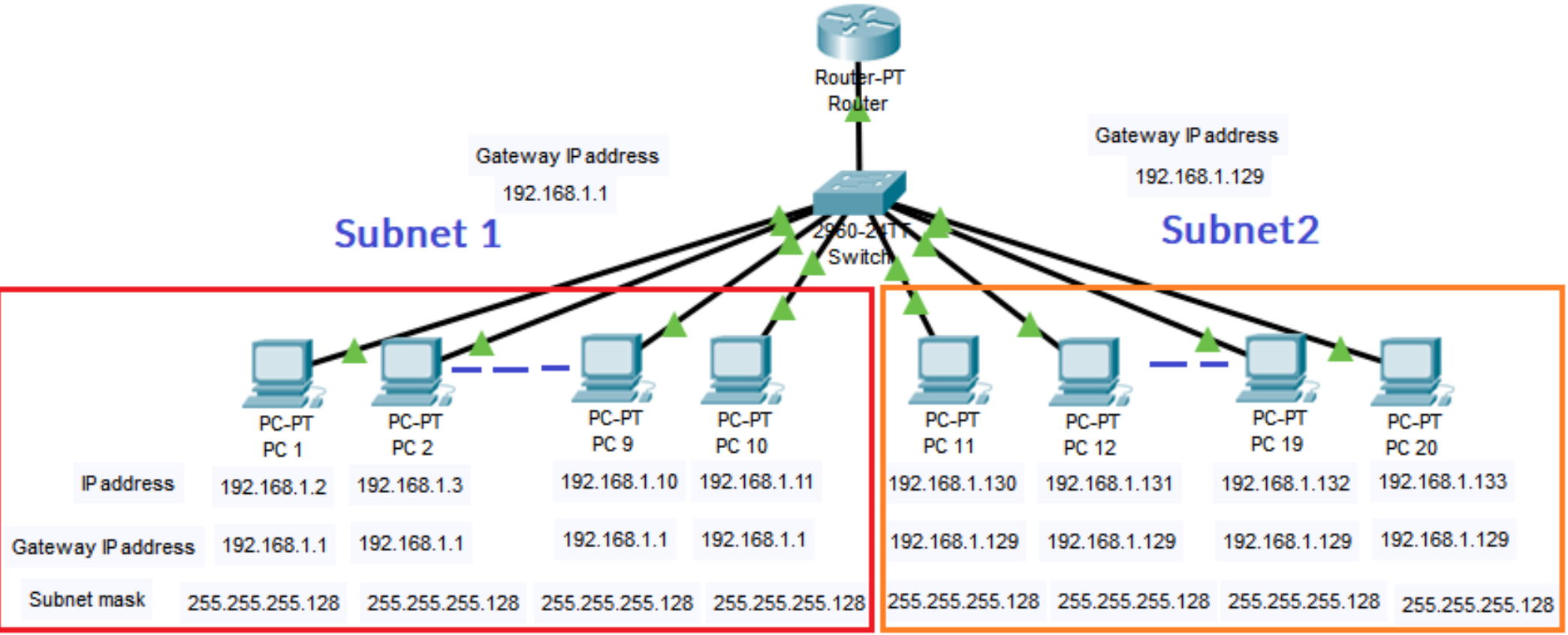
Számítógépes Hálózatok

7. gyakorlat

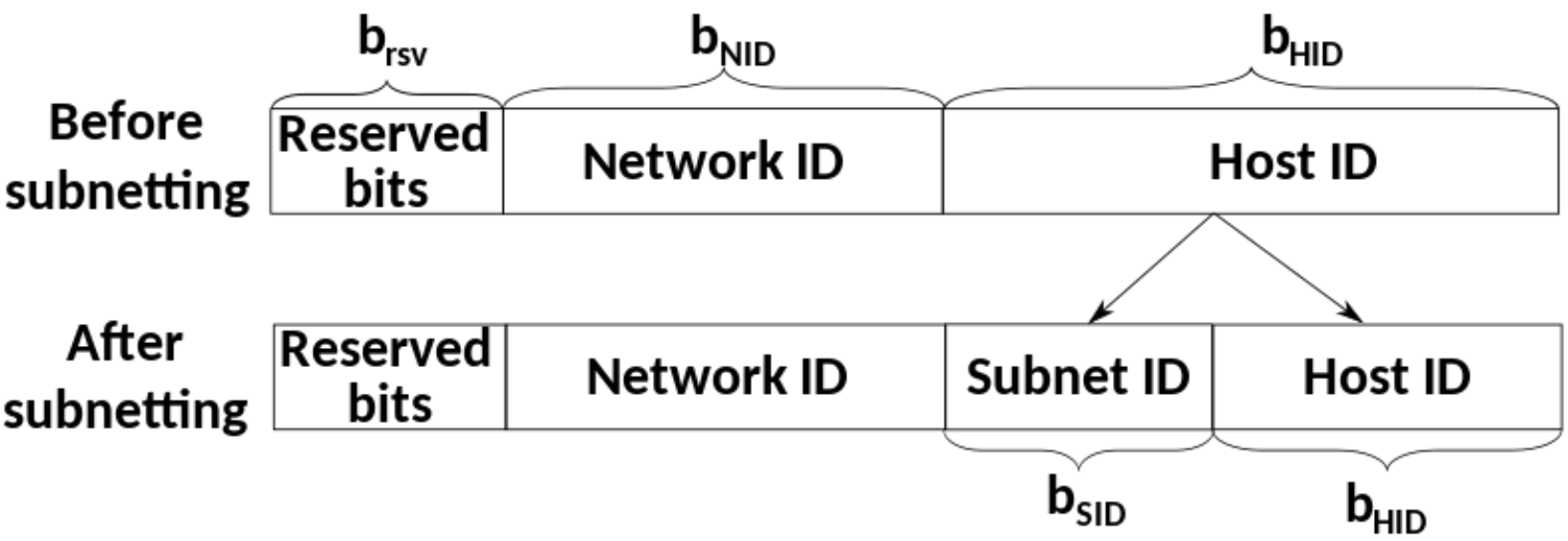
NETMASK (ALHÁLÓZATOK)

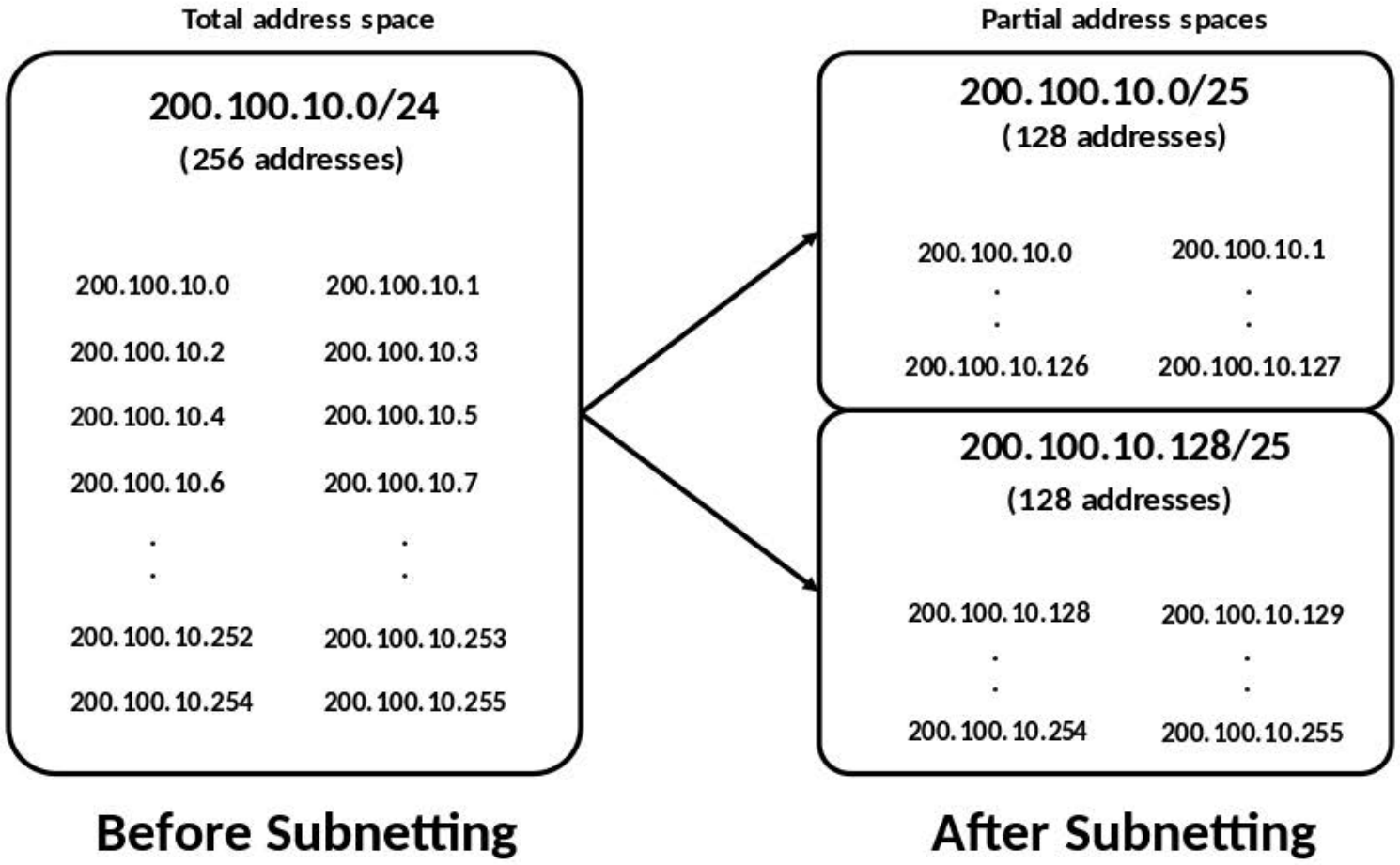
Alhálózatok kialakítása





Forrés: https://www.seomining.com/web-technologies/module4/purpose-subnetting-subnetMasks.php és https://bytesofgigabytes.com/networking/what-is-subnetting-in-networking/





Forrés: https://en.wikipedia.org/wiki/Subnetwork

Netmask

Address (Host or Network) **Netmask** (i.e. 24) **Netmask** for sub/supernet (optional)

192.168.0.1 / 24 move to:

Calculate

```
No host given
No netmask given (using default netmask of your network's class)
```

```
Address:   192.168.0.1          11000000.10101000.00000000 .00000001
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111 .00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000 .11111111
=>
Network:   192.168.0.0/24       11000000.10101000.00000000 .00000000 (Class C)
Broadcast: 192.168.0.255        11000000.10101000.00000000 .11111111
HostMin:   192.168.0.1          11000000.10101000.00000000 .00000001
HostMax:   192.168.0.254        11000000.10101000.00000000 .11111110
Hosts/Net: 254                  (Private Internet)
```

} 2 foglalt cím

Alhálózati maszk

- Az alhálózat egy logikai felosztása egy IP hálózatnak. Az IP cím ezért két részből áll: hálózatszámból és hoszt azonosítóból.
- A szétválasztás a 32 bites alhálózati maszk segítségével történik, amellyel bitenkénti ÉS-t alkalmazva az IP címre megkapjuk a hálózat-, komplementerével pedig a hoszt azonosítót.
- Ez arra jó, hogy meg tudjuk határozni, hogy a címzett állomás a helyi alhálózaton van-e, vagy sem.
- Az utóbbi esetben az alapértelmezett router felé továbbítják a csomagot (default gateway).

Alhálózati maszk

- CIDR jelölés: kompakt reprezentációja egy IP címnek és a hozzátartozó hálózatszámnak
- → IP cím + ’/’ + decimális szám.
- Pl.: 135.46.57.14/24 esetben 135.46.57.14 az IP cím,
- 255.255.255.0 a hálózati maszk (24 db. 1-es bit az elejétől),
- így 135.46.57.0 a hálózat azonosító.

Alhálózati maszk – példa

	10000111	00101110	00111001	00001110	**135.46.57.14**
AND	11111111	11111111	11111111	00000000	**255.255.255.0**
	10000111	00101110	00111001	00000000	**135.46.57.0**

135.46.57.14/24 → 135.46.57.0

Feladat 3

- Hány cím érhető el a következő alhálózati maszkokkal? Adjuk meg a minimális és maximális címet is!
- 188.100.22.12/32
- 188.100.22.12/20
- 188.100.22.12/10

Feladat 3 megoldása

- 188.100.22.12/32 : egy darab a 188.100.22.12
- 188.100.22.12/20 : $2^{32-20} = 2^{12} = 4096$ darab lenne, de valójában ebből még kettőt le kell vonni, mert speciális jelentéssel bírnak:
  - csupa 0: az alhálózat hálózati címe (magára az alhálózatra vonatkozik)
  - csupa 1-es: broadcast a helyi hálózaton
- 4094 lesz, így a min. cím: 188.100.16.1, a max. cím: 188.100.31.254
- 188.100.22.12/10 : $2^{32-10} - 2 = 4194302$, min. cím: 188.64.0.1, a max. cím: 188.127.255.254.

NAT,PAT,PORTFORWARDING

Hálózati címfordítás (NAT)

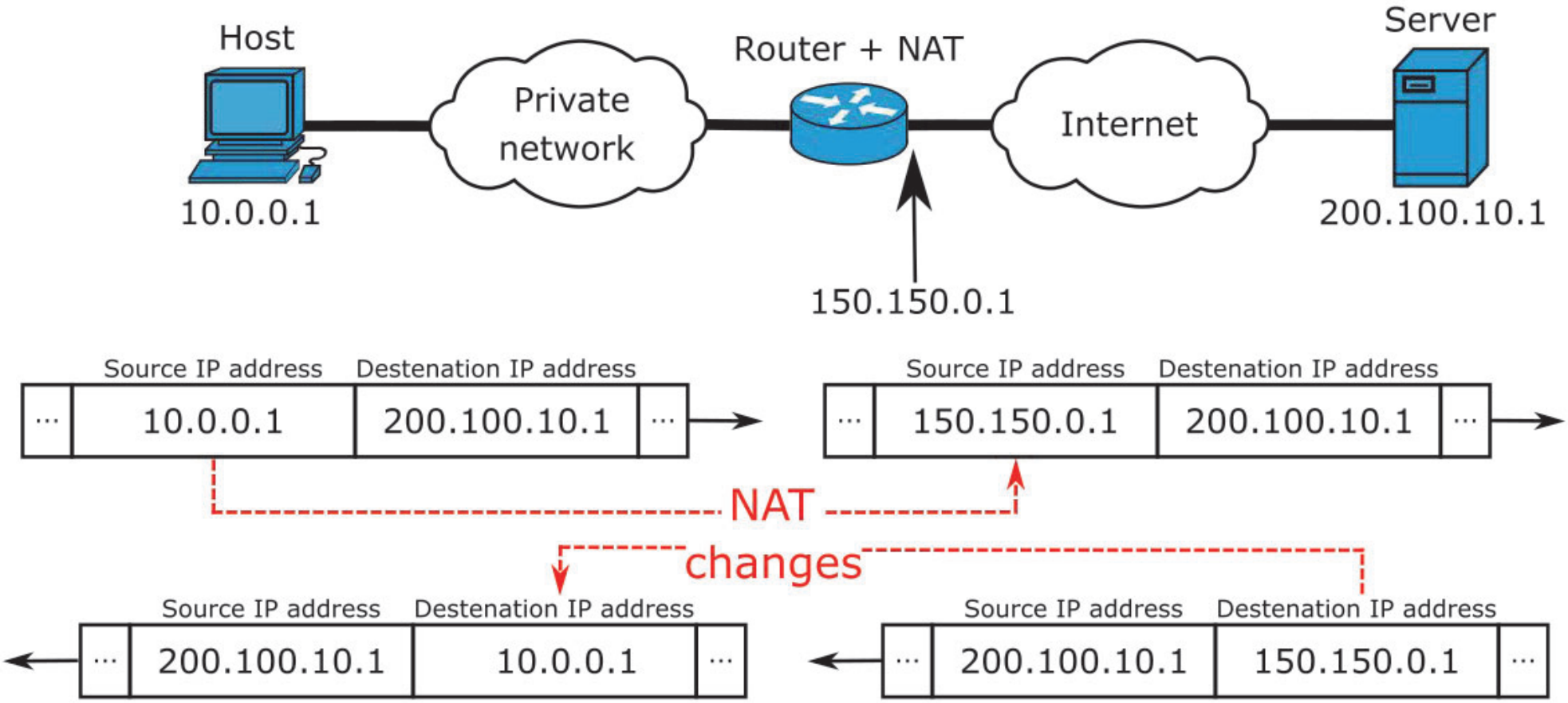
- Gyors javítás az IP címek elfogyásának problémájára.
- Az internet forgalomhoz minden cégnek egy vagy legalábbis kevés IP címet adnak (publikus IP cím(ek))
- A publikus IP cím hozzá van rendelve egy router-hez, a helyi hálózat (LAN) „szélén”, - minden eszközhöz egy privát IP cím van rendelve
- A privát IP címek csak a LAN-on belül érvényesek (vannak IP cím tartományok erre a célra foglalva)

Hálózati címfordítás (NAT)

- Ha a helyi hálózaton lévő másik géppel akarunk kapcsolatot létesíteni → közvetlenül el tudjuk érni
- Amikor helyi eszközről akarunk egy külső eszközt elérni, mi történik?
- Szükségünk van port mezők használatára, ami TCP-nél vagy UDP-nél van (Igazából PAT (Port Address Translation))

Hálózati címfordítás (NAT)

Forrás: wikipedia.org/wiki/Network_adress_translation

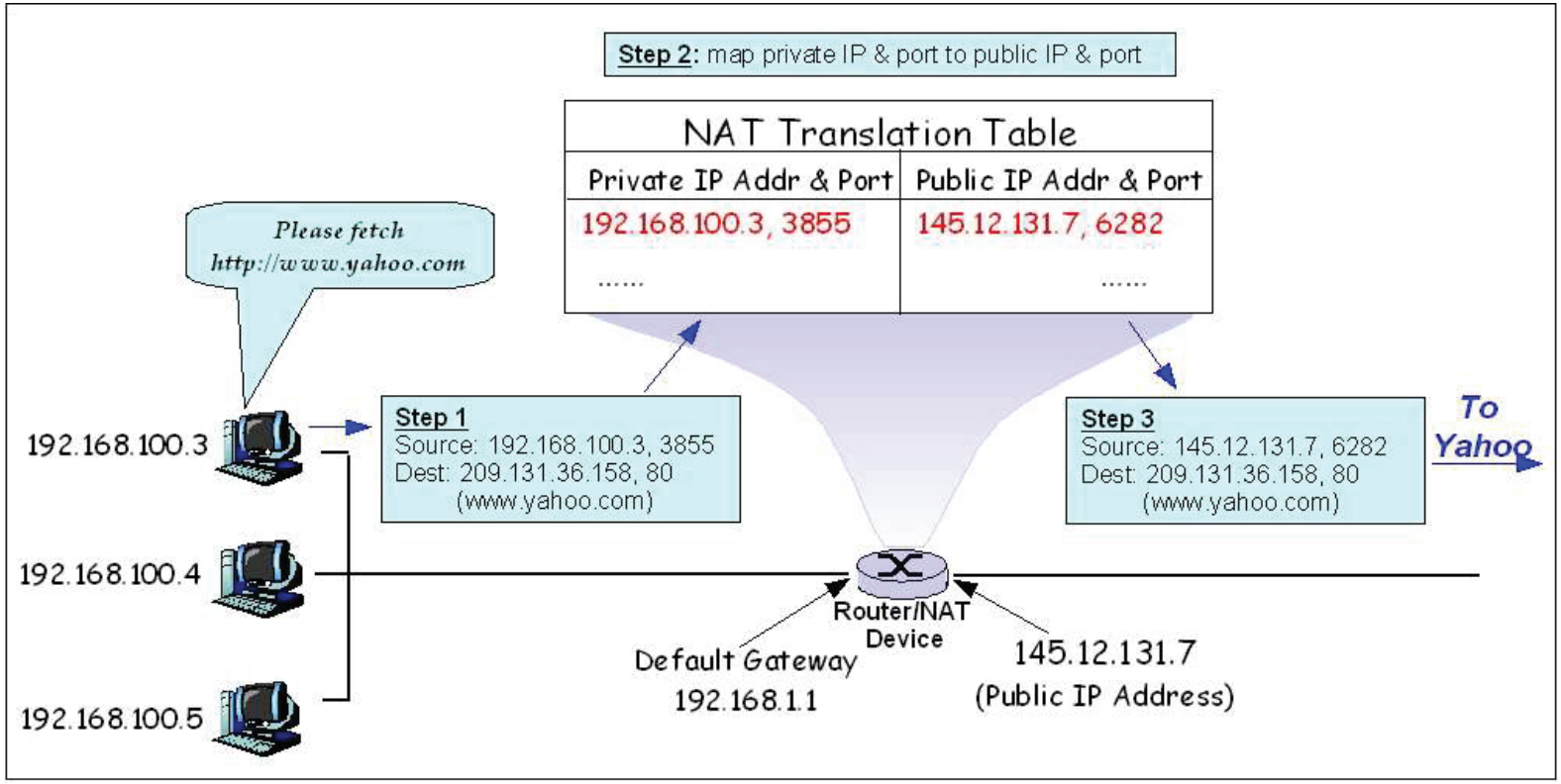


NAT és PAT

- NAT(Network Address Translation): statikusan fordítja az IP címeket, azaz a lokális IP címből csinál publikusat, ezzel lehet a helyi hálózatot elválasztani az internettől. alternatíva: dinamikusan, ahol egy IP poolban, több nyilvános IP van és azokat osztja ki.
- PAT(Port Address Translation): megengedi, hogy több eszköz a LAN hálózaton egy publikus IP-hez legyen rendelve, és a portok szerint legyenek megkülönböztetve.

Hálózati címfordítás (PAT)

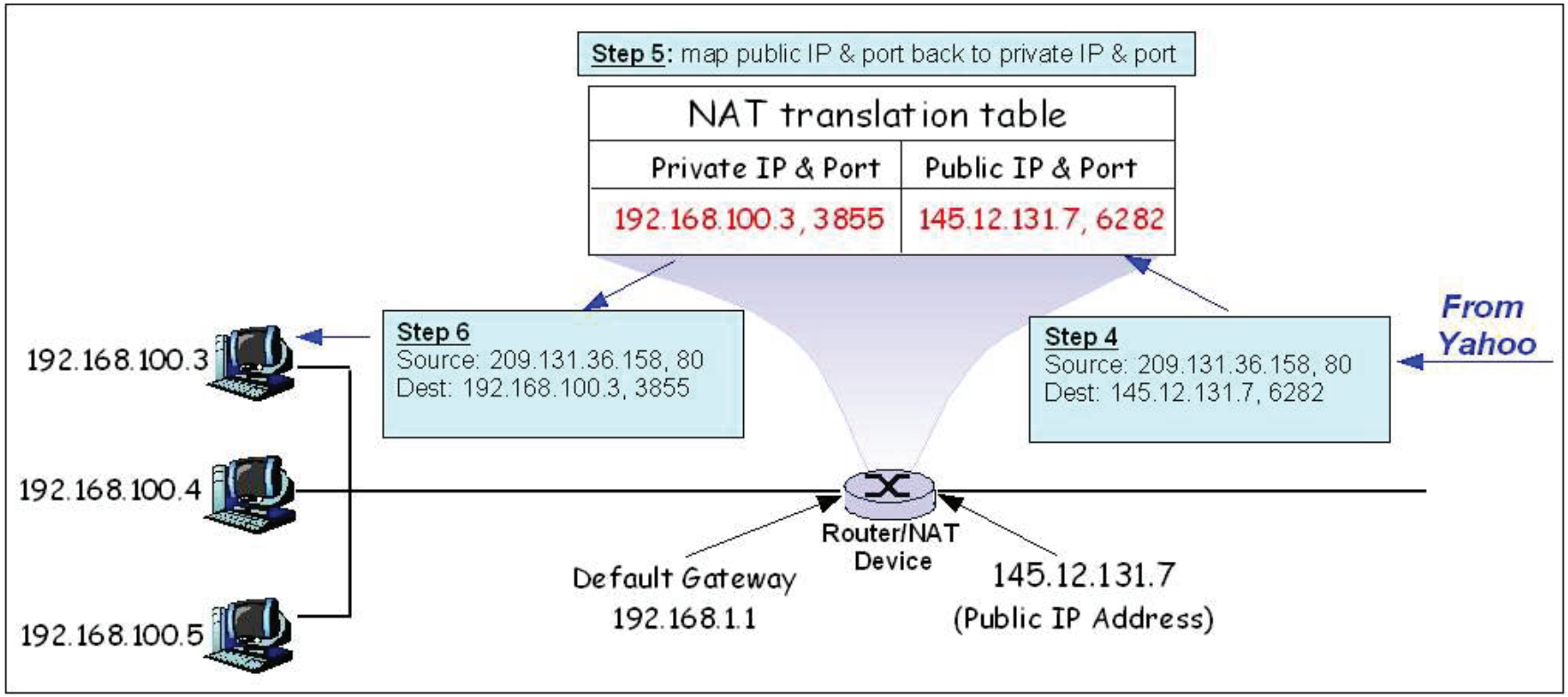
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- 192.168.100.3 privát IP című gépről HTTP kérés, 3855 porton → Default gateway (192.168.1.1): megnézi a translation tábláját:
  - Ha létezik már a (192.168.100.3, 3855) párhoz (publikus IP cím, port) bejegyzés → lecseréli a küldő forrását arra
  - Ha nincs létrehoz egy új bejegyzést (egyedi lesz!), és azt használja fel a cseréhez

Hálózati címfordítás (PAT)

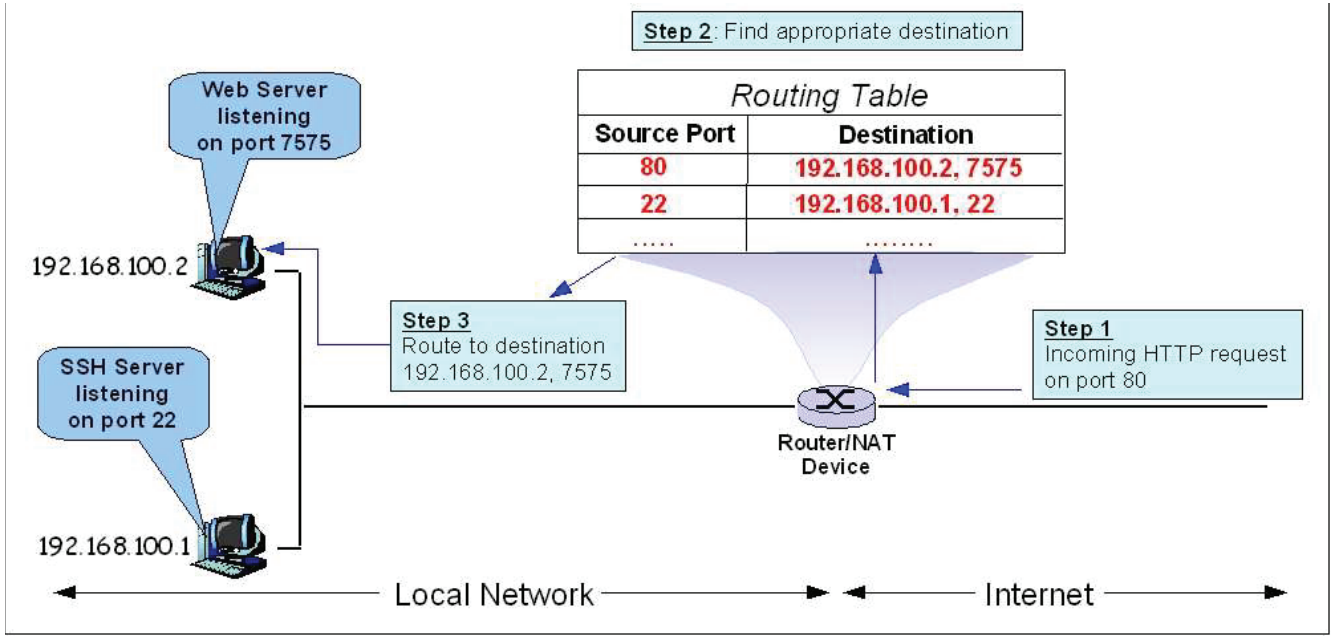
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- A HTTP válasz a yahoo-tól ugyanúgy a router translation tábláján keresztül megy végbe, csak fordított irányban
- Egy különbség: hiányzó bejegyzés esetén a csomagot eldobja a router

Porttovábbítás (port forwarding)

Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



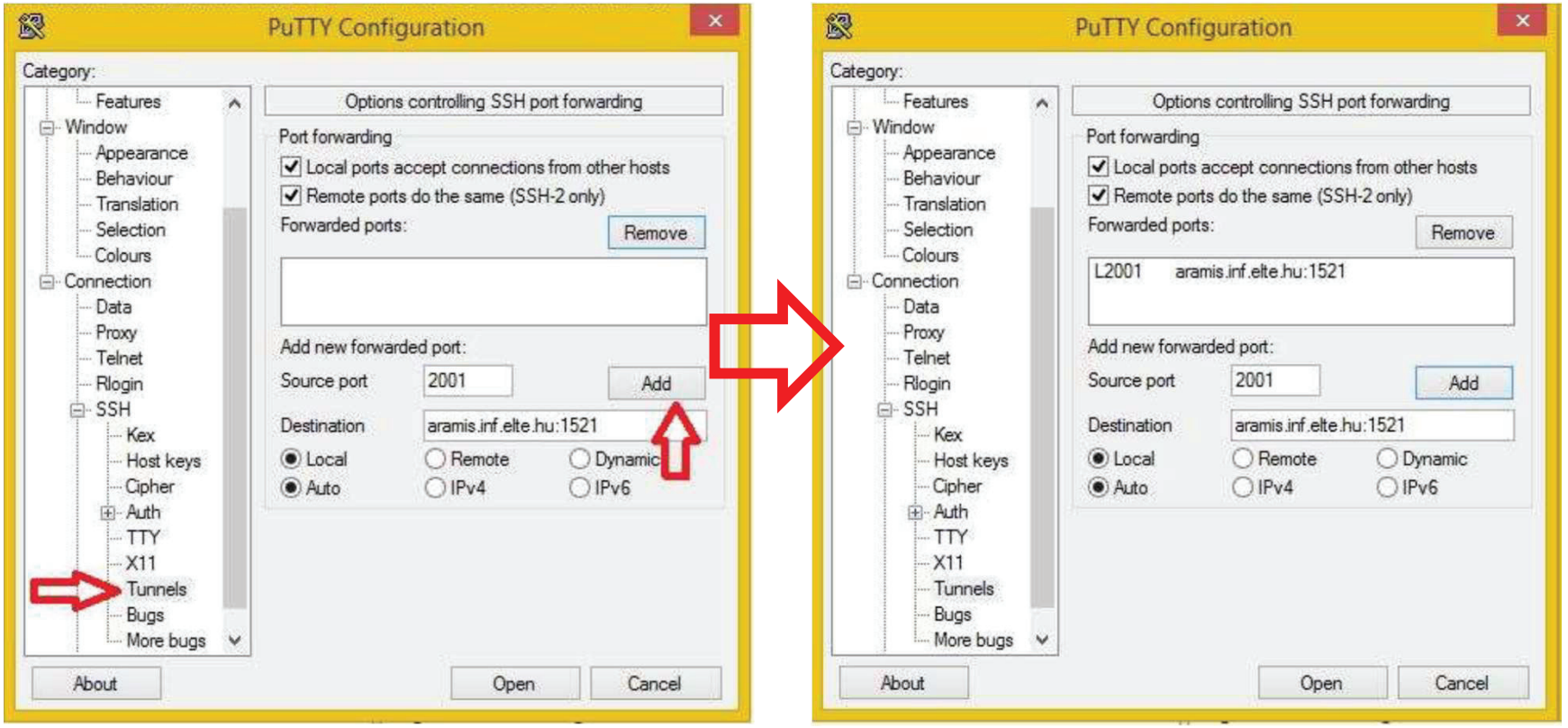
- Az előző példánál a címfordítás transzparens volt (csak a router tudott arról, hogy IP konverzió zajlik). Mit lehet tenni, ha pl. egy belső hálózaton lévő HTTP szervert akarunk elérni kívülről?
- **Porttovábbítás** lehetővé teszi adott lokális hálózaton (LAN) lévő privát IP címek külső elérését egy megadott porton keresztül
- Gyakorlatilag ez a *statikus* NAT alkalmazása

Portforwardingra példa

SSH TUNNEL

SSH Tunnel

- A porttovábbítás egyik tipikus alkalmazása
- Windows (putty) beállítások
  - (Nyitni kell egy ssh kapcsolatot a caesar.elte.hu-ra)



https://people.inf.elte.hu/nikovits/DB1/ssh_tunnel.pdf

SSH Tunnel

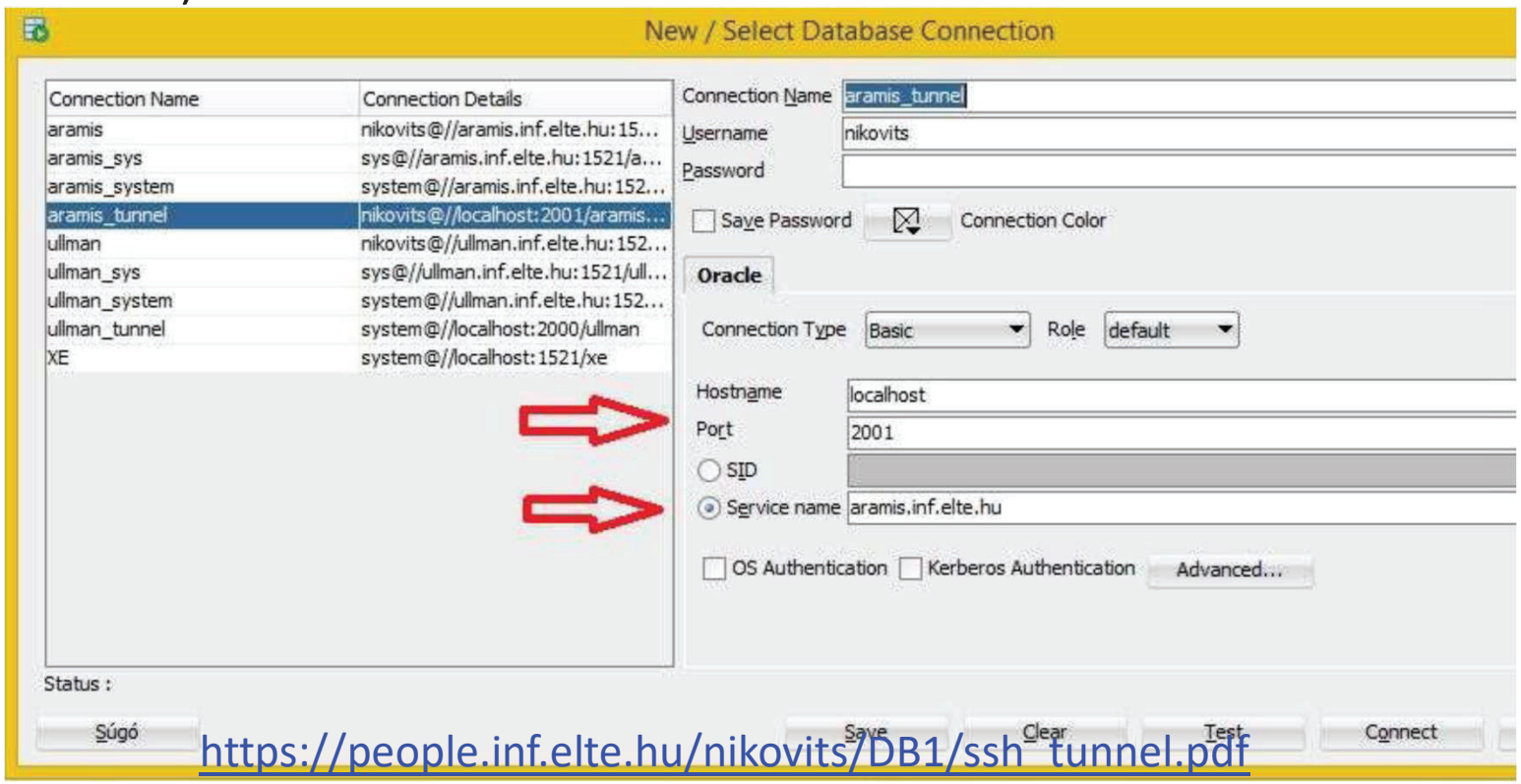
- Linux

```
ssh –L 2001:aramis.inf.elte.hu:1521 user@hostname
```

- ssh –L <localport>:<remote host>:<remote port> <gateway you can ssh in>
  - localport: a localhost ezen porján lesz elérhető a távoli szerver/szolgáltatás
  - remote host:remote port: ide csatlakozik a tunnel végpont, minden, amit a localportra küldünk ide fog továbbítódni és vissza. A gateway-ről elérhetőnek kell lennie!
  - gateway: a gép, amire be tudunk sshval lépni!

SSH Tunnel

- Használat SqlDeveloper-nél:
  - (ssh kapcsolatnak fenn kell állnia végig az adatbázis kapcsolat ideje alatt)

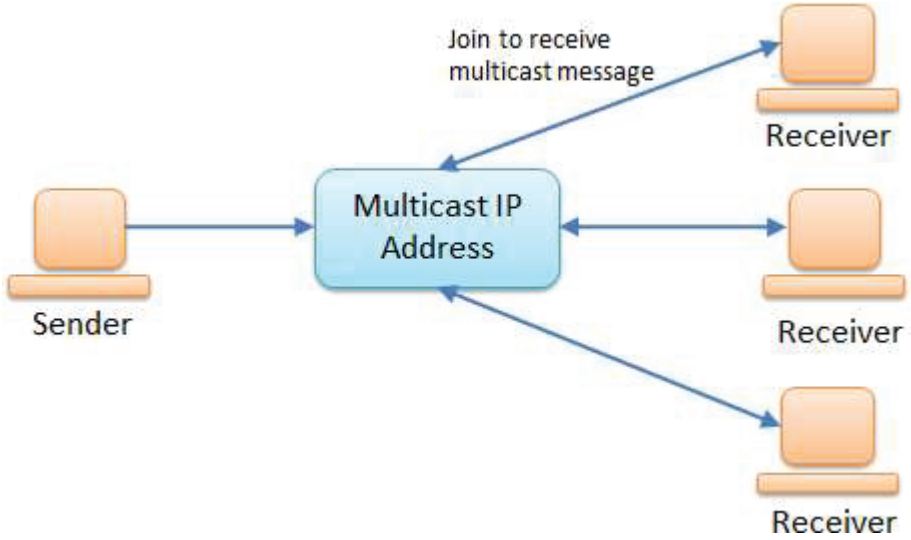


https://people.inf.elte.hu/nikovits/DB1/ssh_tunnel.pdf

MULTICAST

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
        +-+-+-+-+-+-------------------+
```



- Ugyanazt az infót külön-külön elküldeni a társaknak nem optimális a erőforrás kihasználtság szempontjából
- A multicast egy időben több végpontnak is tudja szállítani az üzenetet → jobb hatékonyság
- A pont-pont összeköttetés sokféle kommunikációs igényt ki tud szolgáln

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 életidő (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)

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)
```

Feladat Multicast

- Készítsünk egy multicast fogadó és küldő alkalmazást!
- A saját gépünkön fusson a küldő és a fogadó is

  (TTL értéke 1)

MINTA ZH ÉS GYAKORLÁS

Minta ZH

- ZHMintafeladatok.pdf
- A pdf-ben 1-5ig vannak feladatok, amiket meg kell oldani fokozatosan a jobb jegyért, a feladatokon belül lévő 'abcd' alpontok a feldatsorok randomalizálást reprezentálják, ahol minden hallgató egy-egy véletlen kombinációját kapja a feladatsornak.

ZH gyakorlás

- 3 script: naviClient, naviServer, mapBank
- A naviClient 4 számot kap a standard inputról/parancssori argumentumként/fájlból (2 koordináta, óra, perc), ezeket + hard codeolva a Neptun kódodat egy struktúrba csomagolja / egy bytestringbe rakja valamilyen elválasztó karakterrel kiegészítve és UDP/TCP-n keresztül elküldi a naviServernek.
- A naviServer kiírja a kapott infokat a konzolra majd egy TCP/UDP 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 naviCl ientnek.

VÉGE
KÖSZÖNÖM A FIGYELMET!