Számítógépes Hálózatok

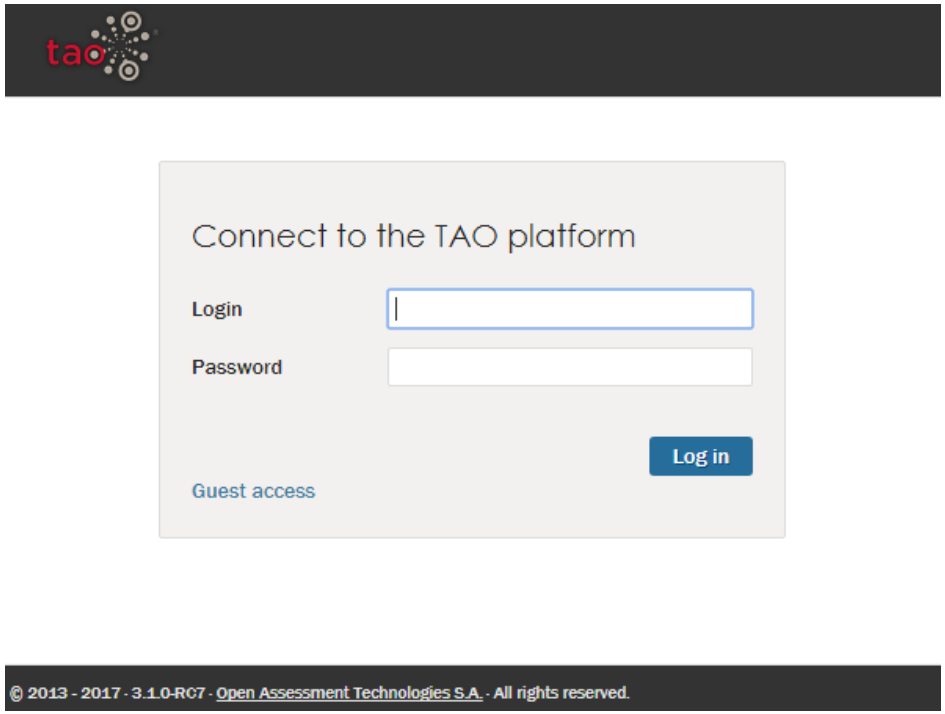
9. gyakorlat

Elérhetőségek

- honlap: http://szalaigj.web.elte.hu/
- email: szalaigindl@inf.elte.hu
- szoba: 2.507 (déli tömb)

Óra eleji kisZH

- Elérés:
  - https://oktnb16.inf.elte.hu



Gyakorlat tematika

- Forgalomirányítás
- Hálózati beállítások, forgalom figyelés

Forgalomirányítás

- **Definíció**: a hálózati réteg szoftverének azon része, amely azért a döntésért felelős, hogy a bejövő csomag melyik kimeneti vonalon kerüljön továbbításra.
- **Elosztott Bellman-Ford forgalomirányítási algoritmus**:
  - más néven távolságvektor alapú forgalomirányítás
  - minden router-nek egy táblázatot kell karbantartania,
  - amelyben minden célhoz szerepel a legrövidebb ismert távolság,
  - és annak a vonalnak az azonosítója, amelyiken a célhoz lehet eljutni.
  - A táblázatokat a szomszédoktól származó információk alapján frissítik.

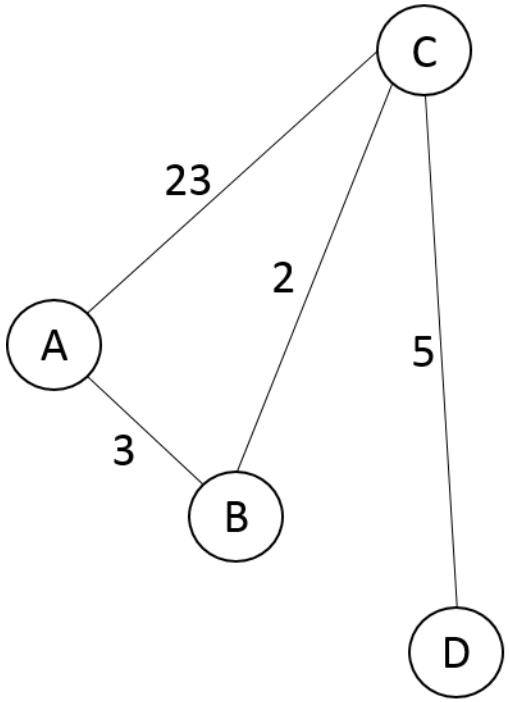
Elosztott Bellman-Ford algoritmus

- Minden csomópont csak a közvetlen szomszédjaival kommunikálhat.
- Minden állomásnak van saját távolság vektora. Ezt periodikusan elküldi a direkt szomszédoknak.
- A kapott távolság vektorok alapján minden csomópont új táblázatot állít elő.

Elosztott Bellman-Ford algoritmus pszeudokódja

```
1.  Initialization:
2.    for all neighbors V  do
3.      if V adjacent to A
4.        D(A, V) = c(A,V);
5.    else
6.        D(A, V) = ∞;
7.  loop:
8.    wait (link cost update or update message)
9.    if (c(A,V) changes by d)
10.     for all destinations Y through V do
11.       D(A,Y) =  D(A,Y) + d
12.   else if (update D(V, Y) received from V)
13.     for all destinations Y do
14.       if (destination Y through V)
15.         D(A,Y) = D(A,V) + D(V, Y);
16.       else
17.         D(A, Y) = min(D(A, Y), D(A, V) + D(V, Y));
18.   if (there is a new minimum for destination Y)
19.     send D(A, Y) to all neighbors
20. forever
```

Elosztott Bellman-Ford algoritmus – példa



A kezdeti vektora		
A-ból	költség	keresztül
B-hez	3	B-n
C-hez	23	C-n
D-hez	∞	-

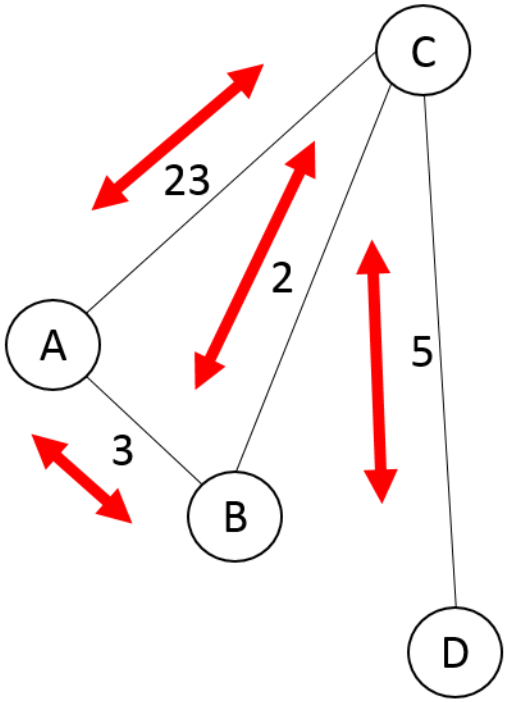
B kezdeti vektora		
B-ből	költség	keresztül
A-hoz	3	A-n
C-hez	2	C-n
D-hez	∞	-

C kezdeti vektora		
C-ből	költség	keresztül
A-hoz	23	A-n
B-hez	2	B-n
D-hez	5	D-n

D kezdeti vektora		
D-ből	költség	keresztül
A-hoz	∞	-
B-hez	∞	-
C-hez	5	C-n

- A példa a https://en.wikipedia.org/wiki/Distance-vector_routing_protocol alapján készült.

Elosztott Bellman-Ford algoritmus – példa



A vektora B és C fogadása után

A-ból	költség	keresztül
B-hez	3	B-n
C-hez	5	B-n
D-hez	28	C-n

B vektora A és C fogadása után

B-ből	költség	keresztül
A-hoz	3	A-n
C-hez	2	C-n
D-hez	7	C-n

C vektora A, B és D fogadása után

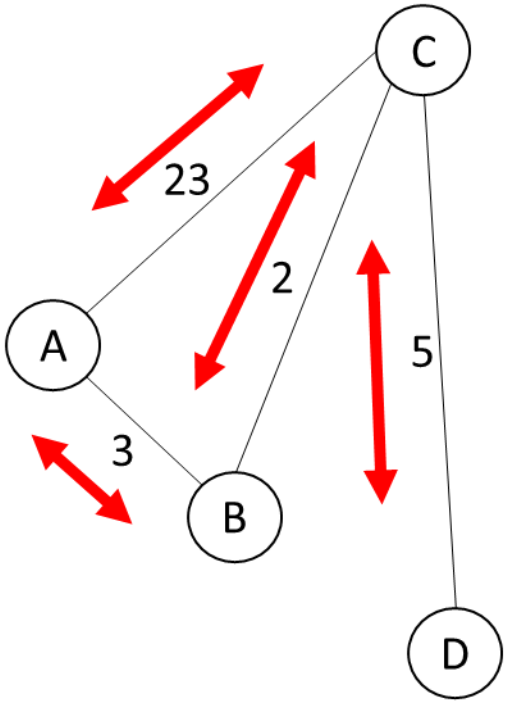
C-ből	költség	keresztül
A-hoz	5	B-n
B-hez	2	B-n
D-hez	5	D-n

D vektora C fogadása után

D-ből	költség	keresztül
A-hoz	28	C-n
B-hez	7	C-n
C-hez	5	C-n

- Mivel az összes csúcsnál van új legrövidebb út a kezdeti fázis után, ezért mindegyik csúcs küldeni fogja a vektorát az összes szomszédjának.
- A csúcsok újraszámítják a legrövidebb utakat ezek alapján.
- Zöld háttér → új legrövidebb út

Elosztott Bellman-Ford algoritmus – példa



A vektora B és C fogadása után

A-ból	költség	keresztül
B-hez	3	B-n
C-hez	5	B-n
D-hez	10	B-n

B vektora A és C fogadása után

B-ből	költség	keresztül
A-hoz	3	A-n
C-hez	2	C-n
D-hez	7	C-n

C vektora A, B és D fogadása után

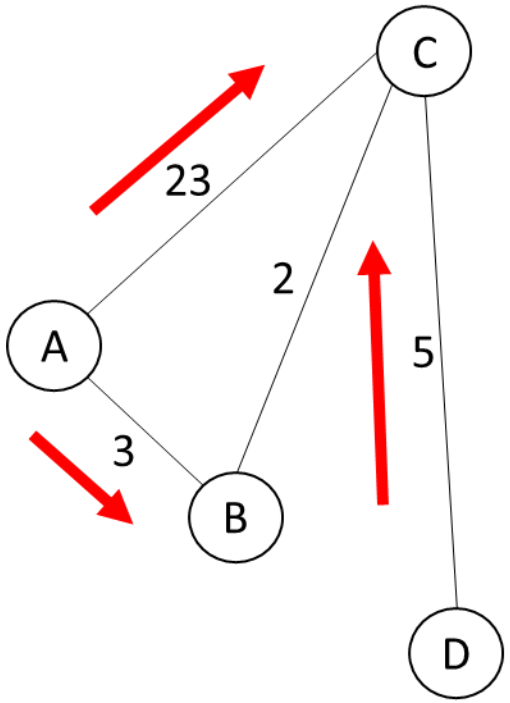
C-ből	költség	keresztül
A-hoz	5	B-n
B-hez	2	B-n
D-hez	5	D-n

D vektora C fogadása után

D-ből	költség	keresztül
A-hoz	10	C-n
B-hez	7	C-n
C-hez	5	C-n

- Most viszont már csak *A* és *D* csúcsnak változott a vektora, ezért csak ezek fogják elküldeni vektoraikat szomszédjaiknak.

Elosztott Bellman-Ford algoritmus – példa



A végső vektora

A-ból	költség	keresztül
B-hez	3	B-n
C-hez	5	B-n
D-hez	10	B-n

B vektora A fogadása után

B-ből	költség	keresztül
A-hoz	3	A-n
C-hez	2	C-n
D-hez	7	C-n

C vektora A és D fogadása után

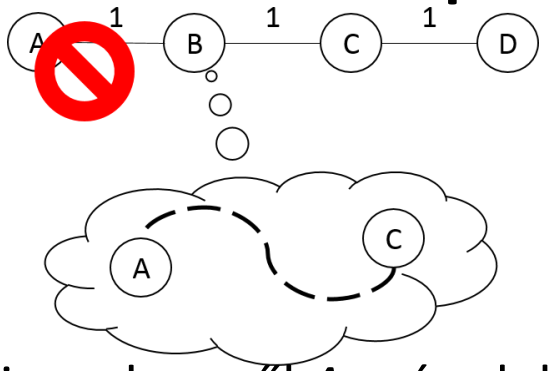
C-ből	költség	keresztül
A-hoz	5	B-n
B-hez	2	B-n
D-hez	5	D-n

D végső vektora

D-ből	költség	keresztül
A-hoz	10	C-n
B-hez	7	C-n
C-hez	5	C-n

- Mivel egyik vektor sem hordoz információt legrövidebb utakról, így nem változnak a táblázatok.
- Nincs üzenetszóró csúcs → az algoritmus leáll…
- … legalábbis amíg meg nem változik valami

Elosztott Bellman-Ford algoritmus – Végtelenig számolás problémája



- Tegyük fel, hogy a fenti rendszerről *A* csúcs lekapcsolódik.
- *B* érzékeli, hogy *A* felé már nincs meg az él
- A probléma, hogy *C*-től fog kapni egy vektort, amely tévesen azt az információt hordozza, hogy *C A*-tól 2 távolságra van (*C*-*B*-*A*), **mivel** *C* **nem tudja, hogy** *A* **nincs már bent**.
- Ezért *B* frissíti az *A*-hoz vezető utat 2 + 1-gyel a táblázatban, mivel nem tudja, hogy a *C*-féle 2 költségű úton **ő is rajta van**.
- *B*-nek változott a vektora, amit *C* felé továbbít, aki továbbra is úgy tudja, hogy *A B*-n keresztül elérhető, emiatt *C* az *A*-hoz vezető utat 3 + 1-re frissíti → ez így fog menni a „végtelenig”

Elosztott Bellman-Ford algoritmus – Végtelenig számolás problémája

- Lehetséges megoldások:
- **„split horizon”**: olyan utakat nem küld vissza a csomópont a szomszédjának, amit tőle „tanult”. (*A*-ról nem küld információt *C* *B*-nek)
- **„split horizon with poison reverse”**: negatív információt küld vissza arról a szomszédjának, amit tőle „tanult”. (*C* a $D(C,A) = \infty$ információt küldi *B*-nek, így *B*-nek nem lesz útvonala *A*-hoz *C* keresztül.)

Feladat 1

- Tegyük fel, hogy egy távolság alapú forgalomirányítási protokollban az *A* és *B* routerek távolság vektora az alábbi ábrán található. A protokoll elosztott Bellman-Ford algoritmust használ az útvonalak meghatározására. A költségek szimmetrikusak, azaz minden élen mindkét irányban azonosak.

A vektora		
A-ból	költség	keresztül
B-hez	4	B-n
C-hez	6	C-n
D-hez	11	B-n
E-hez	10	C-n

B vektora		
B-ből	költség	keresztül
A-hoz	4	A-n
C-hez	10	A-n
D-hez	7	D-n
E-hez	14	A-n

- Tegyük fel, hogy a csomópontok a „split horizon” szabályt használják a távolságvektorok átadására. Adja meg azt a távolságvektort, amit *B* elküld *A*-nak, miután *E* és *B* közötti közvetlen kapcsolat költsége 5-re változik.
- Adja meg azt a távolságvektort, amit *B* az előző pontban küldene *A*-nak, ha „split horizon with poison reverse” szabályt használna.

Feladat 1 megoldása

- „split horizon” szabály esetén:

B vektora		
B-ből	költség	keresztül
D-hez	7	D-n
E-hez	5	E-n

- „split horizon with poison reverse” szabály esetén:

B vektora		
B-ből	költség	keresztül
A-hoz	∞	-
C-hez	∞	-
D-hez	7	D-n
E-hez	5	E-n

Dijkstra algoritmus

- Statikus (nem adaptív) forgalomirányító algoritmus
- Az eredeti algoritmus célja: két csúcs közti legrövidebb út meghatározása.
- Leggyakrabban használt változatban: egyik csúcsot kinevezzük forrás csúcsnak, és onnan a legrövidebb utakat kiszámítjuk az összes többi csúcshoz.
- Egy *v* csúcsnak kétféle címkéje lehet: ideiglenes (*ready*[*v*] = *false*) vagy állandó (*ready*[*v*] = *true*). Kezdetben csak a forrás csúcs állandó, a többi ideiglenes.
- A legrövidebb út megtalálásakor a címke állandó címkévé válik, és továbbá nem változik.
- Van még két segédhalmazunk: *E'* és *Q*. Az *E'*-ben az állandó címkéjűeket tároljuk.

Dijkstra algoritmus

- *Q*-ban kezdetben a forrás csúcs szomszédjai szerepelnek az odavezető élsúlyoknak megfelelő prioritással (a legkisebb prioritás van *Q* legtetején)
- Amíg *Q* ki nem ürül, addig mindig levesszük a tetejéről a csúcsot (a legkisebb költségűt), beletesszük *E'*-be, és állandó címkét kap. Jelöljük ezt a csúcsot *v*-vel.
- Vesszük a szomszédjait:
  - Ha egy *u* szomszédja még nincs *Q*-ban → betesszük, prioritása a *v*-hez vezető legrövidebb út hosszának és *v*,*u* közti élsúlynak az összege lesz
  - Ha benne van, de kisebb költségű úttal is elérhető → csökkentjük a prioritását

Dijkstra algoritmus pszeudokódja (az előadás diasorról)

Dijkstra(G,s,w)
Output: egy legrövidebb utak fája T=(V,E´) G-ben s gyökérrel

```
01 E´ := Ø;
02 ready[s] := true;
03 ready[v] := false; ∀ v ∈ V \ {s};
04 d[s] := 0;
05 d[v] := ∞; ∀ v ∈ V \ {s};
06 priority_queue Q;
07 forall v ∈ Adj[s] do
08    pred[v] := s;
09    d[v] := w(s,v);
10    Q.Insert(v,d[v]);
11 od
12 while Q ≠ Ø do
13    v := Q.DeleteMin();
14    E´:= E´ U {(pred[v],v)};
15    ready[v] := true;
16    forall u ∈ Adj[v] do
17       if u ∈ Q and d[v] + w(v,u) < d[u]) then
18          pred[u] := v;
19          d[u] := d[v] + w(v,u);
20          Q.DecreasePriority(u,d[u]);
21       else if u ∉ Q and not ready[u] then
22          pred[u] := v;
23          d[u] := d[v] + w(v,u);
24          Q.Insert(u,d[u]);
25       fi
26    od
27 od
```

INICIALIZÁCIÓS FÁZIS

ITERÁCIÓS LÉPÉSEK

JAVÍTÓ ÚT

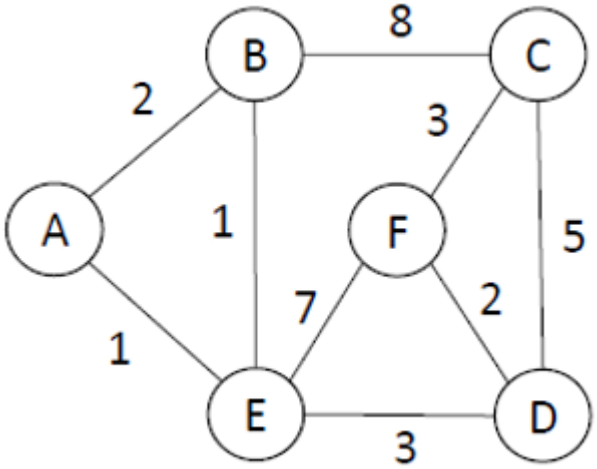
ÚJ ÚT

Feladat 2

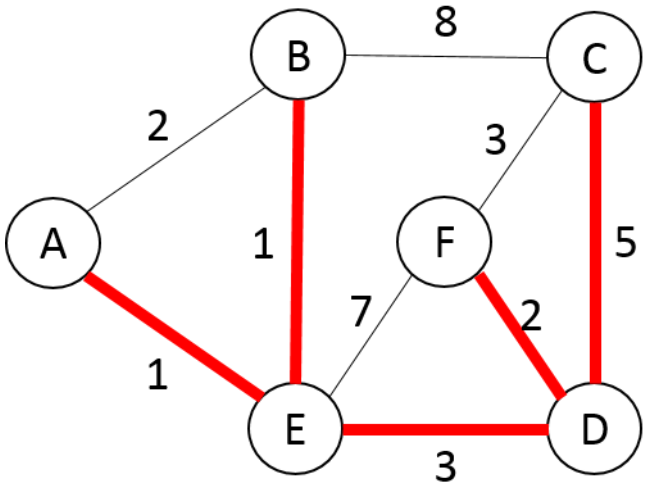
Tekintsük a $G = (V ; E)$ gráfot az alábbi ábrán egy hálózat reprezentánsának.

Először számítsa ki Dijkstra algoritmusával egy legrövidebb utak fáját D csomópontból minden más csomóponthoz a pszeudokód segítségével. Minden iteráció után jelölje a „kész” csomópontokat és adja meg minden u 2 V csomóponthoz $d[u]$ és $pred[u]$ értéket egy táblázatban.

Ezt követően rajzolja fel a kiszámított legrövidebb utak fáját.



Feladat 2 megoldása



	d	pred	d	pred	d	pred	d	pred	d	pred	d	pred
A	∞	-	∞	-	∞	-	**4**	E	**4**	E	**4**	E
B	∞	-	∞	-	∞	-	4	E	**4**	E	**4**	E
C	∞	-	5	D	5	D	5	D	5	D	**5**	D
D	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-
E	∞	-	3	D	**3**	D	**3**	D	**3**	D	**3**	D
F	∞	-	**2**	D	**2**	D	**2**	D	**2**	D	**2**	D

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: még nincs ip címe a gépnek
  - 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.

ping, traceroute / tracert

- A ping program segítségével derítsük ki mennyi ideig tart, amíg a csomag a tartózkodási helyétől az alábbi helyhez eljut:
- Berkeley.edu – Berkeley, CA, USA
- Csomagok útvonalának (path) meghatározására, és az átviteli késleltetés mérésére használjuk a traceroute/tracert programot!

ping, traceroute / tracert

```
C:\[redacted]>ping Berkeley.edu

Berkeley.edu [128.32.203.137] pingelése - 32 bájtnyi adattal:
Válasz 128.32.203.137: bájt=32 idő=187 ms TTL=49
Válasz 128.32.203.137: bájt=32 idő=188 ms TTL=49
Válasz 128.32.203.137: bájt=32 idő=186 ms TTL=49
Válasz 128.32.203.137: bájt=32 idő=186 ms TTL=49

128.32.203.137 ping-statisztikája:
    Csomagok: küldött = 4, fogadott = 4, elveszett = 0
                        (0% veszteség),
Oda-vissza út ideje közelítőlegesen, milliszekundumban:
    minimum = 186ms, maximum = 188ms, átlag = 186ms
```

```
C:\[redacted]>tracert Berkeley.edu

Útvonal követése a következőhöz: Berkeley.edu [128.32.203.137]
legfeljebb 30 ugrással:

  1     1 ms     1 ms     1 ms  192.168.0.1
  2    14 ms     8 ms    12 ms  catv-89-133-39-254.catv.broadband.hu [89.133.39.254]
  3     8 ms    10 ms     8 ms  catv-89-135-214-109.catv.broadband.hu [89.135.214.109]
  4    38 ms    34 ms    34 ms  hu-bud02a-ra3-ae26-2030.aorta.net [84.116.241.22]
  5    35 ms    35 ms    36 ms  84.116.137.206
  6    37 ms    38 ms    36 ms  84.116.140.194
  7     *        *        *     A kérésre nem érkezett válasz a határidőn belül.
  8    35 ms    35 ms    34 ms  uk-lon01b-ri1-ae23-0.aorta.net [84.116.135.30]
  9    37 ms    34 ms    36 ms  195.66.225.24
 10    44 ms    36 ms    39 ms  nl-sar.nordu.net [109.105.97.124]
 11   115 ms   115 ms   114 ms  us-man.nordu.net [109.105.97.139]
 12   125 ms   124 ms     *     64.57.21.54
 13     *        *        *     A kérésre nem érkezett válasz a határidőn belül.
 14     *        *        *     A kérésre nem érkezett válasz a határidőn belül.
 15     *        *        *     A kérésre nem érkezett válasz a határidőn belül.
 16   199 ms   195 ms   195 ms  dc-lax-agg6--lax-agg7-100g.cenic.net [137.164.11.10]
 17   190 ms   184 ms   184 ms  dc-svl-agg4--lax-agg6-100ge.cenic.net [137.164.11.1]
 18   186 ms   190 ms   189 ms  dc-oak-agg4--svl-agg4-100ge.cenic.net [137.164.46.144]
 19   189 ms   186 ms   188 ms  ucb--oak-agg4-10g.cenic.net [137.164.50.31]
 20   188 ms   187 ms   188 ms  t2-3.inr-201-sut.berkeley.edu [128.32.0.37]
 21     *      188 ms   188 ms  e3-48.inr-310-ewdc.berkeley.edu [128.32.0.97]
 22   189 ms   188 ms   189 ms  calweb-farm-prod.ist.berkeley.edu [128.32.203.137]

Az útvonalkövetés elkészült.
```

ifconfig (Linux)

- Hálózati interfészek konfigurálására/lekérdezésére
- Pl.: segítségével megtudhatjuk az IP címünket (inet addr) és MAC címünket (HWaddr):
  - (forrás: https://en.wikipedia.org/wiki/Ifconfig)

```
eth0      Link encap:Ethernet HWaddr 00:0F:20:CF:8B:42
          inet addr:217.149.127.10 Bcast:217.149.127.63 Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:2472694671 errors:1 dropped:0 overruns:0 frame:0
          TX packets:44641779 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1761467179 (1679.7 Mb) TX bytes:2870928587 (2737.9 Mb)
          Interrupt:28
```

- Ehhez hasonló Windows-nál az ipconfig

További hasznos hálózati segédeszközök

- netstat (Linux és Windows) : kilistázza az aktív hálózati kapcsolatokat
- Pl. kideríthető vele, hogy a programok melyik portokat használják
- tcpdump (Linux) : forgalom figyelő eszköz, a hálózati interfészről jövő csomagokat tudja olvasni

Hálózati címfordítás (NAT)

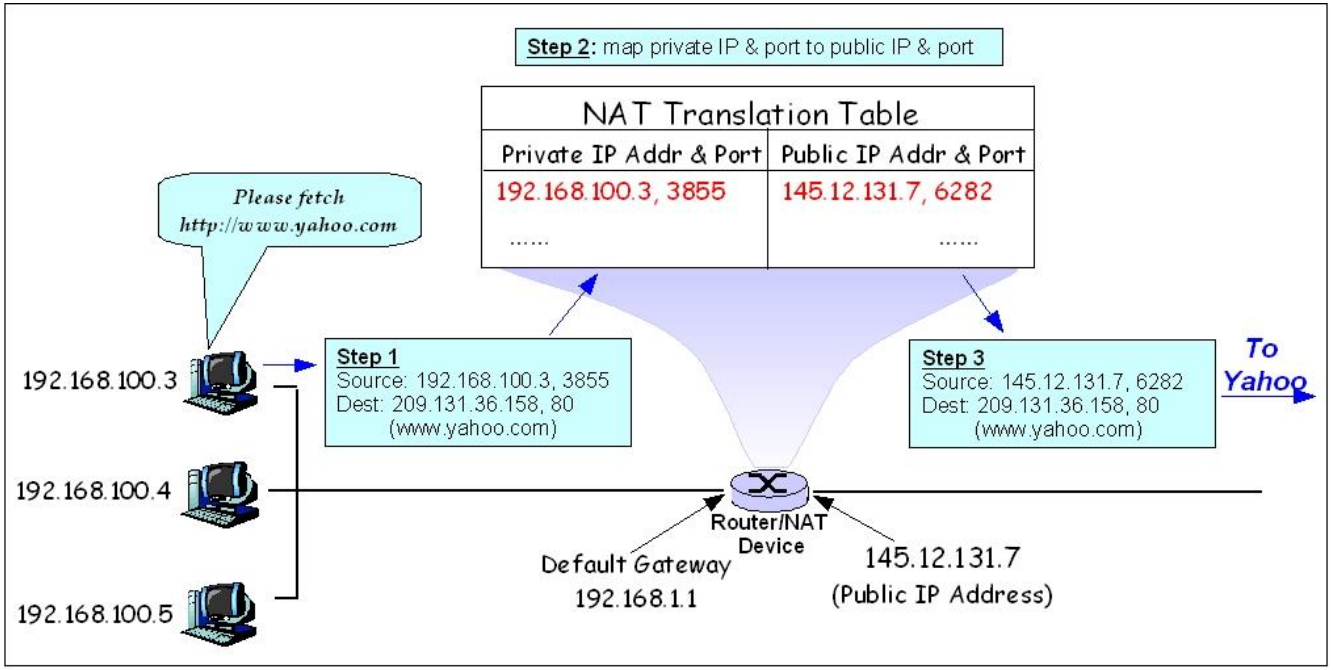
- Gyors javítás az IP címek elfogyásának problémájára.
- Az internet forgalomhoz minden cégnek egy vagy legalábbis kevés IP címet adnak (publikus IP cím(ek))
- A publikus IP cím hozzá van rendelve egy router-hez, a helyi hálózaton (LAN) belül, - amely emögött van, - minden eszközhöz egy privát IP cím van rendelve
- A privát IP címek csak a LAN 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

Hálózati címfordítás (NAT)

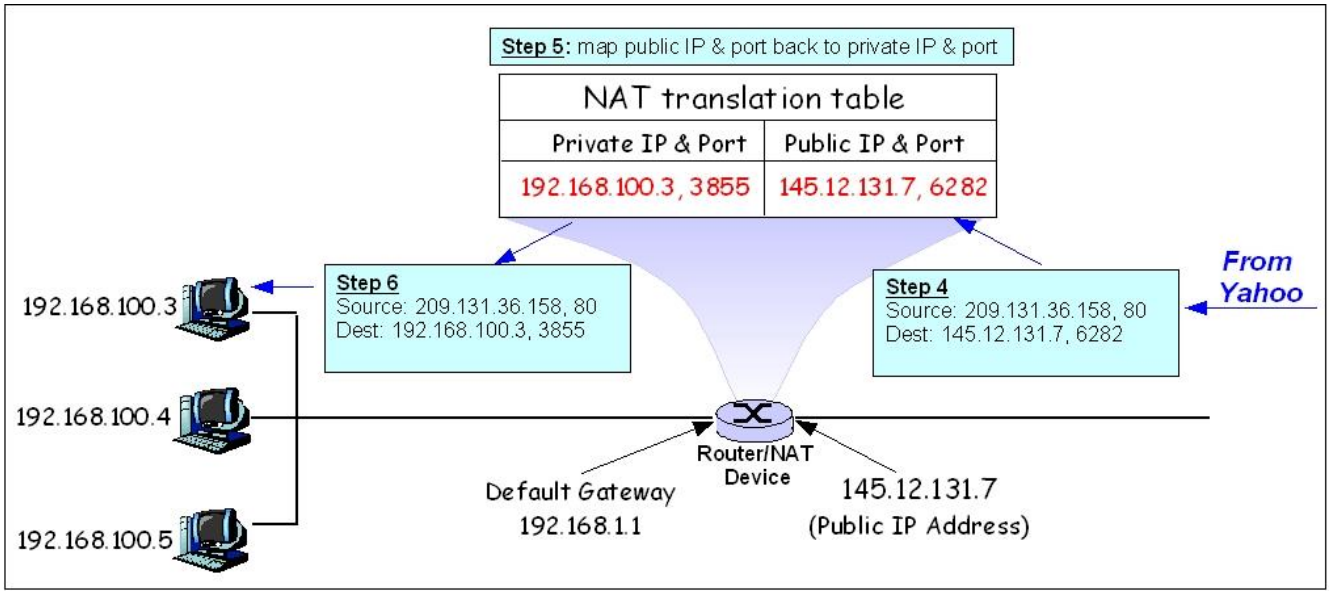
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- 192.168.100.3 privát IP című gépről HTTP kérés, 3855 porton → Default gateway (192.168.1.1): megnézi a transzlációs 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 (NAT)

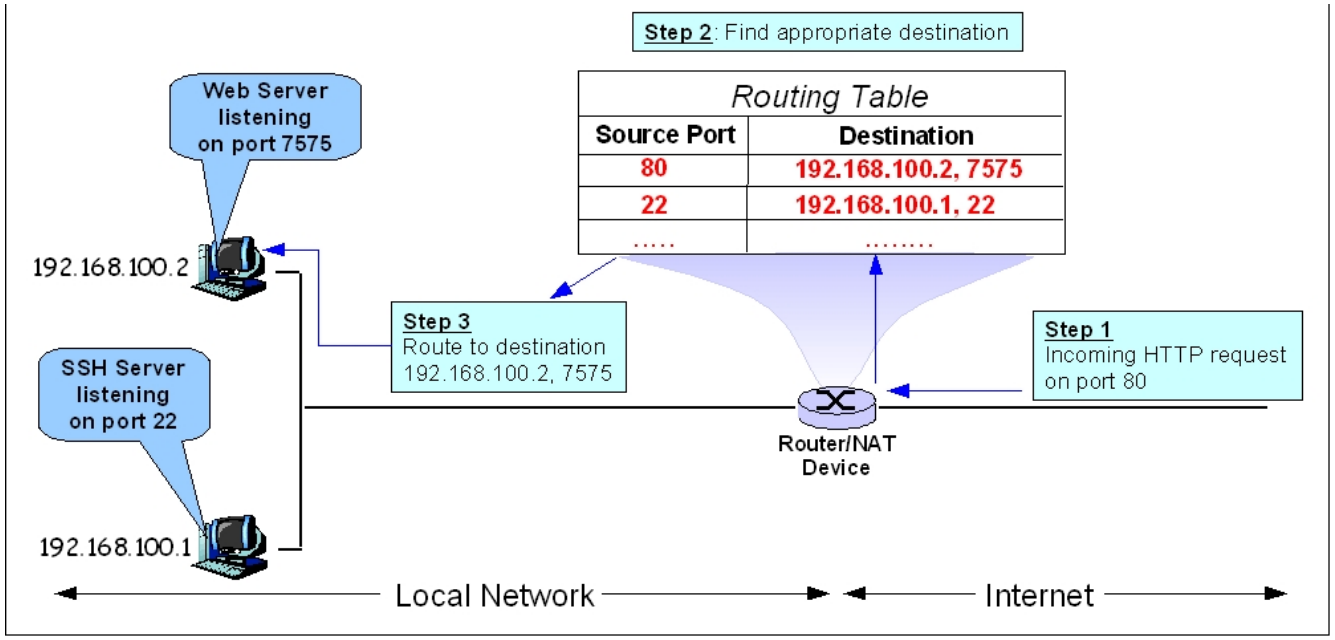
Forrás: https://en.wikibooks.org/wiki/Communication_Networks/NAT_and_PAT_Protocols



- A HTTP válasz a yahoo-tól ugyanúgy a router transzlációs 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** lehetové teszi adott lokális hálózaton (LAN) lévő privát IP címek külso elérését egy megadott porton keresztül
- Gyakorlatilag ez a *statikus* NAT alkalmazása

iptables (Linux)

- Egy Linux program csomagszűrési/csomagtovábbítási szabályok, NAT módosítása/lekérdezése
- Például szeretnénk a 192.168.1.10 IP címhez és 80-as porthoz jövő csomagot a 192.168.1.20 IP című géphez küldeni a 80-as portjához, akkor az alábbi parancsok (is) kelleni fognak:

```
iptables -t nat -A PREROUTING -p tcp –dport 80 -j DNAT –to-destination 192.168.1.20:80
```

```
iptables -t nat -A POSTROUTING -p tcp -d 192.168.12.20 –dport 80 -j SNAT –to-source 192.168.12.10
```

- (-t kapcsolóval a táblát határozzuk meg, -A PREROUTING : a szabályt a PREROUTING lánc végére szúrja be, -j a csomagcél megadására (SNAT: Source NAT, DNAT: Destination NAT))

VÉGE