Számítógépes hálózatok

HATODIK ELŐADÁS – Hálózati réteg, forgalomirányítási protokollok, címzés

KÉSZÍTETTE: ÁCS ZOLTÁN

Hálózati réteg szerepkörei

Fő feladata

A csomagok továbbítása a forrás és a cél között.

- A legalacsonyabb olyan réteg, amely két végpont közötti átvitellel foglalkozik

Elvárásokkal kapcsolatos feladatok

- Ismernie kell a kommunikációs alhálózat topológiáját.
  - Megfelelő útvonalak meghatározására.
- Ügyelni kell, hogy ne terheljen túl se bizonyos kommunikációs útvonalakat, se bizonyos *router*-eket úgy, hogy mások tétlen maradnak.

Felhasználói réteg
Szállítási réteg
Hálózati réteg
Adatkapcsolati réteg
Fizikai réteg

HÁLÓZATI RÉTEG – FORGALOMIRÁNYÍTÁSI ALGORITMUSOK

Forgalomirányító algoritmusok

DEFINÍCIÓ
A hálózati réteg szoftverének azon része, amely azért a döntésért felelős, hogy a bejövő csomag melyik kimeneti vonalon kerüljön továbbításra.

- A folyamat két jól-elkülöníthető lépésre bontható fel:
    1. Forgalomirányító táblázatok feltöltése és karbantartása.
    2. Továbbítás.

ELVÁRÁSOK
helyesség, egyszerűség, robosztusság, stabilitás, igazságosság, optimalitás és hatékonyság

ALGORITMUS OSZTÁLYOK
1. Adaptív algoritmusok
    - A topológia és rendszerint a forgalom is befolyásolhatja a döntést
2. Nem-adaptív algoritmusok
    - offline meghatározás, betöltés a router-ekbe induláskor

Forgalomirányító algoritmusok

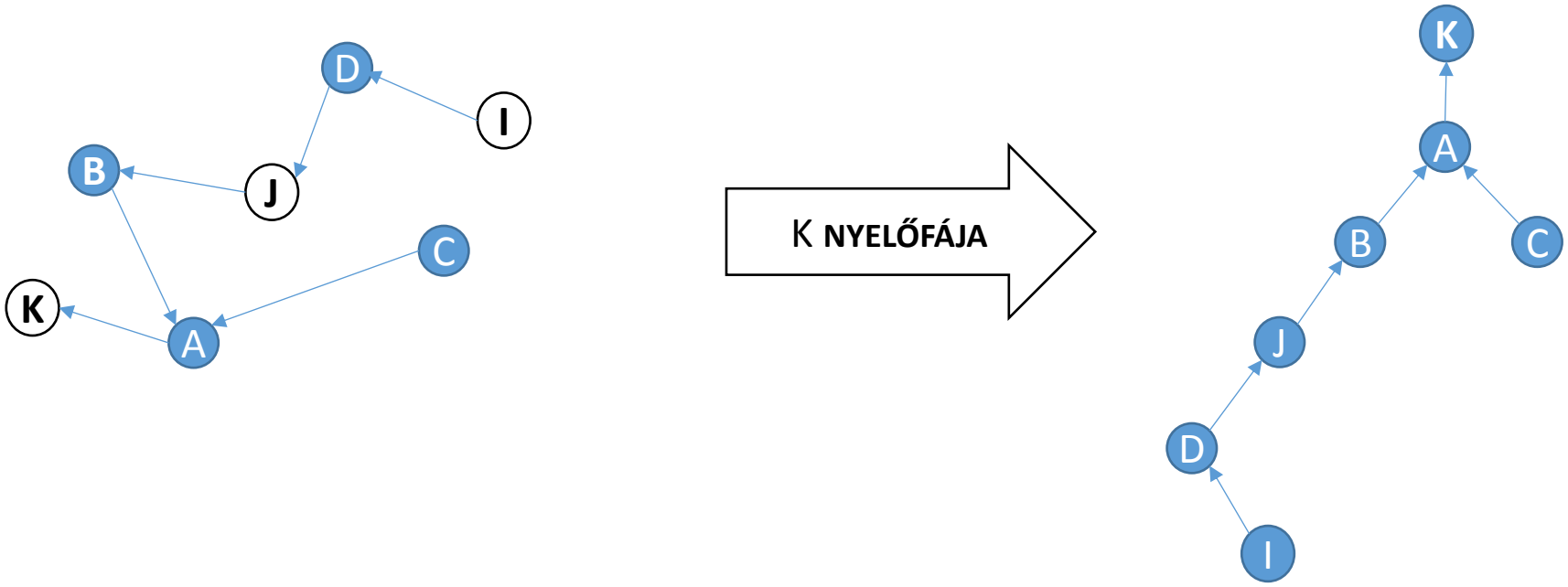
KÜLÖNBSÉGEK AZ EGYES ADAPTÍV ALGORITMUSOKBAN

1. Honnan kapják az információt?
   - szomszédok, helyileg, minden router-től
2. Mikor változtatják az útvonalakat?
   - meghatározott másodpercenként, terhelés változásra, topológia változásra
3. Milyen mértékeket használnak az optimalizáláshoz?
   - távolság, ugrások (*hops*) száma, becsült késleltetés

Optimalitási elv

Ha ***J*** router az ***I*** router-től ***K*** router felé vezető *optimális útvonalon* helyezkedik el, akkor a J-től a K-ig vezető útvonal ugyanerre esik.

- **Következmény**
  Az összes forrásból egy célba tartó optimális utak egy olyan fát alkotnak, melynek a gyökere a cél. Ezt nevezzük ***nyelőfá***nak.

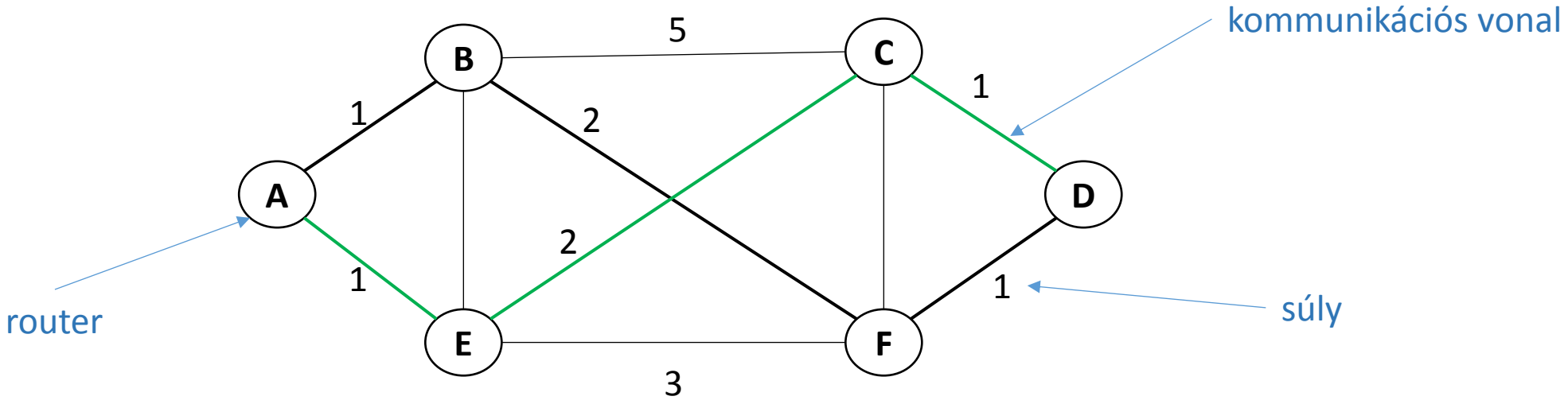


Legrövidebb út alapú forgalomirányítás

ALHÁLÓZAT REPREZENTÁCIÓJA

Az alhálózat tekinthető egy gráfnak, amelyben minden router egy csomópontnak és minden él egy kommunikációs vonalnak felel meg. Az éleken értelmezünk egy $w: E \rightarrow \mathbb{R}_0^+$ nem-negatív súlyfüggvényt, amelyek a legrövidebb utak meghatározásánál használunk.

- *G=(V,E)* gráf reprezentálja az alhálózatot
- *P* útvonal súlya: $w(P) = \sum_{e \epsilon P} w(e)$



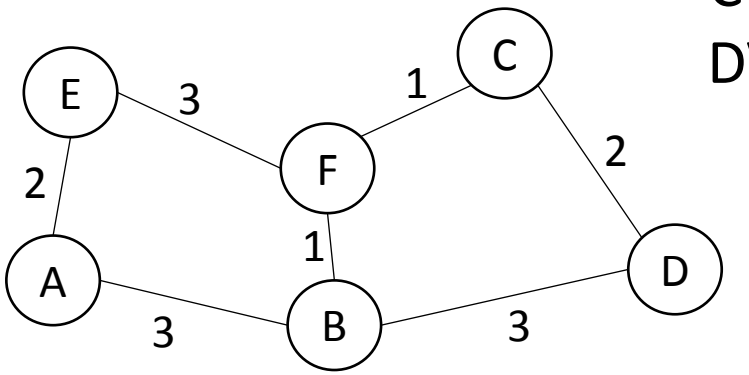
Távolságvektor alapú forgalomirányítás

- Dinamikus algoritmusoknak 2 csoportja van:
  - távolságvektor alapú illetve (distance vector routing)
  - kapcsolatállapot alapú (link-state routing)
- 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 forgalomirányítási algoritmusként is nevezik.
  - ARPANET eredeti forgalomirányító algoritmusa ez volt. RIP (Routing Information Protocol) néven is ezt használták.

Távolságvektor alapú forgalomirányítás Elosztott Bellman-Ford algoritmus

KÖRNYEZET ÉS MŰKÖDÉS

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

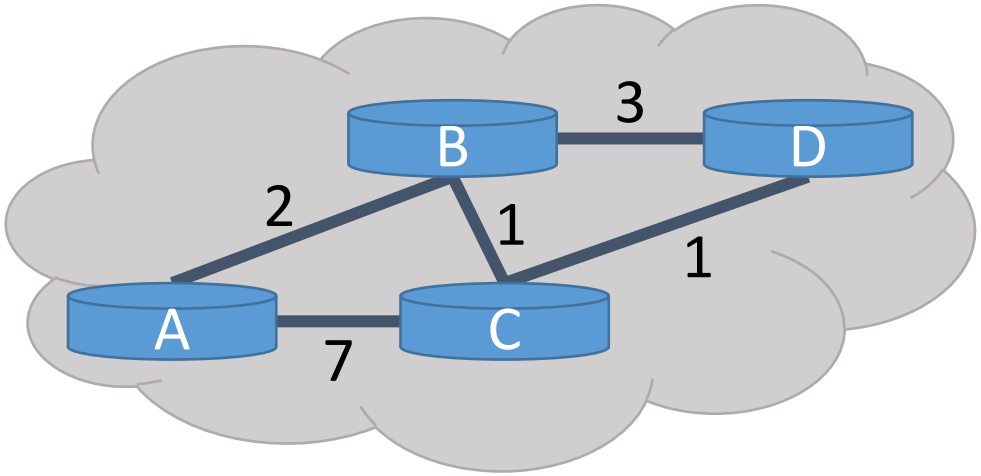


C állomás DV táblája

Cél	Ktsg.
A	5
B	2
D	2
E	4
F	1

- Nincs bejegyzés C-hez
- Kezdetben csak a közvetlen szomszédokhoz van info
  - Más célállomások költsége = ∞
- Végül kitöltött vektort kapunk

Distance Vector Initialization



Node A

Dest.	Cost	Next
B	2	B
C	7	C
D	∞	

Node B

Dest.	Cost	Next
A	2	A
C	1	C
D	3	D

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*) = ∞;

…

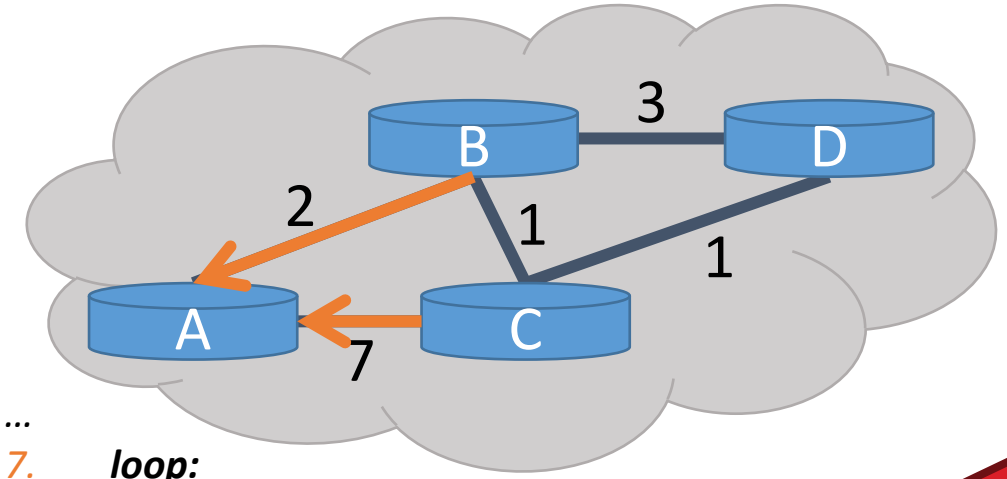
Node C

Dest.	Cost	Next
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	Next
A	∞	
B	3	B
C	1	C

Distance Vector: 1st Iteration




```
...
7.   loop:
...
12.  else if (update D(V, Y) received from V)
13.    for all destinations
14.      if (destination Y t
15.        D(A,Y) = D(A,V) +
16.      else
17.        D(A, Y) =
                   min(D(A,
                   D(A, V) + D(V, Y));
18.  if (there is a new min. for dest. Y)
19.    send D(A, Y) to all neighbors
20.  forever
```

Node A

Dest.	Cost	Next
B	2	B
C	3	B
D	5	B

Node B

Dest.	Cost	Next
A	2	A
C	1	C
D	2	C

D(A,C), D(A,B)+D(B,C))

D(A,D) = min(D(A,D), D(A,B)+D(B,D))
= min(8, 3 + 3) : B

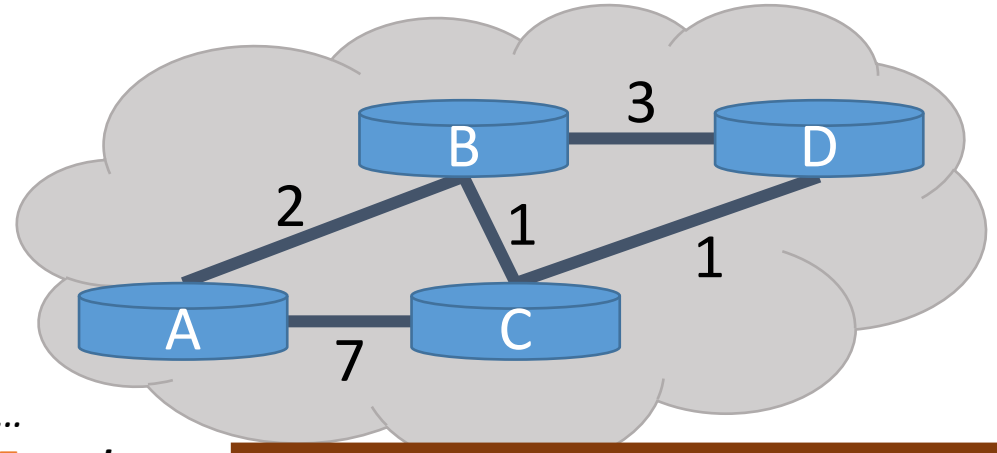
e D

		Next
		B
B	3	B
C	1	C

4

B	1	B
D	1	D

Distance Vector: End of 3rd Iteration



Node A

Dest.	Cost	Next
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	Next
A	2	A
C	1	C
D	2	C

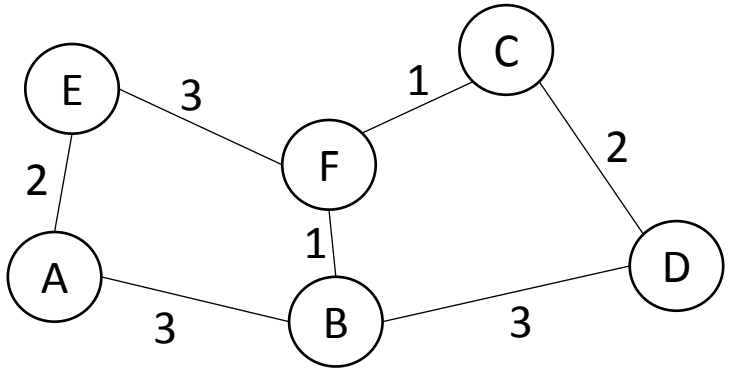
Dest.	Cost	Next
A	3	B
B	1	B
D	1	D

Dest.	Cost	Next
A	4	C
B	2	C
C	1	C

- Nothing changes, algorithm terminates
- Until something changes…

```
…
7.   loop:
…
12.  else if (
13.    for all
14.      if (de
15.        D(A
16.      else
17.        D(A, Y) =
                 min(D(A, Y),
                 D(A, V) + D(V, Y));
18.  if (there is a new min. for dest. Y)
19.    send D(A, Y) to all neighbors
20.  forever
```

Elosztott Bellman-Ford algoritmus – *példa*



Becsült késleltetés A-tól kezdetben

A	cost	N. Hop
B	3	B
C	∞	-
D	∞	-
E	2	E
F	∞	-

B vektora A-nak

A	3
B	0
C	∞
D	3
E	∞
F	1

E vektora A-nak

A	2
B	∞
C	∞
D	∞
E	0
F	3

Új becsült késleltetés A-tól

A	cost	N. Hop
B	3	B
C	∞	-
D	6	B
E	2	E
F	4	B

A vektora B-nek és E-nek

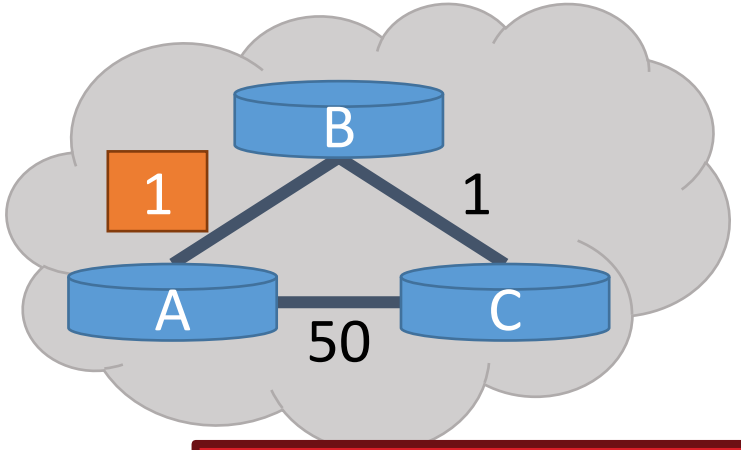
A	0
B	3
C	∞
D	6
E	2
F	4

...

Új becsült késleltetés A-tól

A	cost	N. Hop
B	3	B
C	5	B
D	6	B
E	2	E
F	4	B

```
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.
19.
20.
```



Link C
Algor

Good news travels fast

Algorithm Terminates

Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

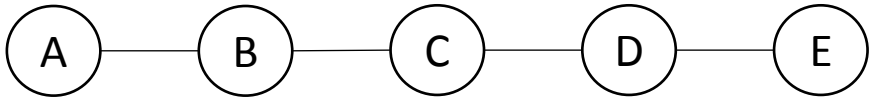
D	C	N
A	2	B
B	1	B

D	C	N
A	2	B
B	1	B

Time

Elosztott Bellman-Ford algoritmus – *Végtelenig számolás problémája*

A állomás meghibásodik

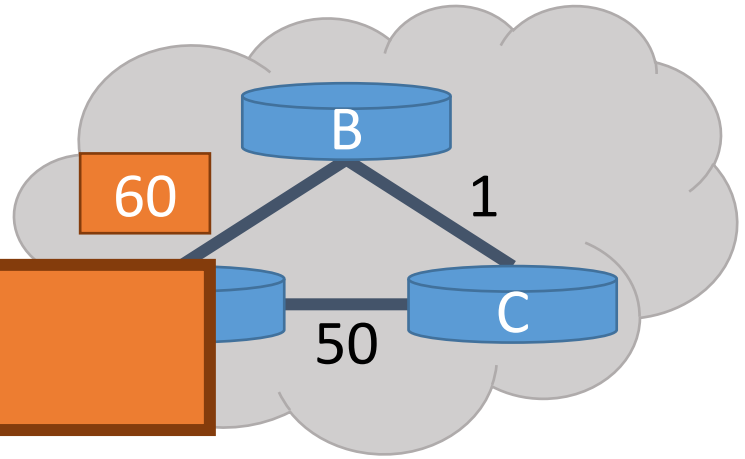


1	2	3	4	Kezdetben
3	2	3	4	1 csere után
3	4	3	4	2 csere után
5	4	5	4	3 csere után
	. . .			
∞	∞	∞	∞	

Count to Infinity Problem

- Node B knows D(C, A) = 5
- However, B does not know the path is C → B → A
- Thus, D(B,A) =

Bad news travels slowly



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	8	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	7	B
B	1	B

D	C	N
A	7	B
B	1	B

Time

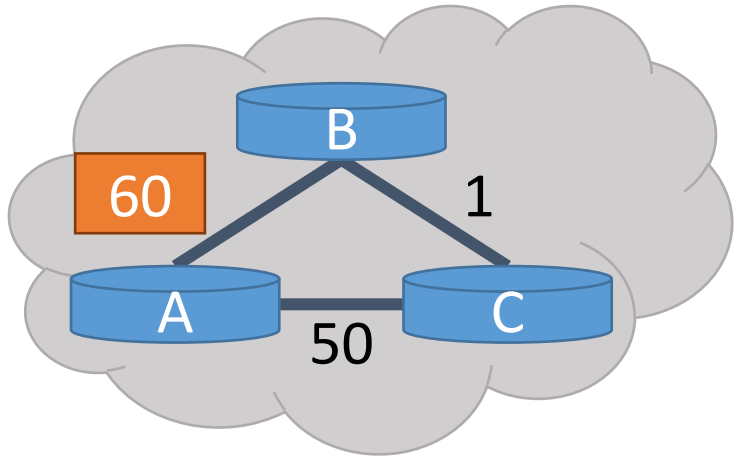
Elosztott Bellman-Ford algoritmus – *Végtelenig számolás problémája*

PROBLÉMA

- A „jó hír” gyorsan terjed.
- A „rossz hír” lassan terjed.
- Azaz ciklusok keletkezhetnek.
- Lehetséges megoldás:
  - **„split horizon with poisoned reverse”**: negatív információt küld vissza arról a szomszédjának, amit tőle „tanult”. (*RFC 1058*)

Poisoned Reverse

- If C routes through B to get to A
  - C tells B that D(C, A) = ∞
  - Thus, B won't route to A via C



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	51	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	50	A
B	1	B

D	C	N
A	50	A
B	1	B

Time

Kapcsolatállapot alapú forgalomirányítás
Link-state routing

MOTIVÁCIÓ

1. Eltérő sávszélek figyelembevétele.
2. Távolság alapú algoritmusok lassan konvergáltak.

AZ ALAPÖTLET ÖT LÉPÉSBŐL TEVŐDIK ÖSSZE

1. Szomszédok felkutatása, és hálózati címeik meghatározása.
2. Megmérni a késleltetést vagy költséget minden szomszédhoz.
3. Egy csomag összeállítása a megismert információkból.
4. Csomag elküldése az **összes többi** router-nek.
5. Kiszámítani a legrövidebb utat az összes többi router- hez.
   - Dijkstra algoritmusát használják.

Kapcsolatállapot alapú forgalomirányítás működése

1. A router beindulásakor az első feladat a szomszédok megismerése, ezért egy speciális `HELLO` csomag elküldésével éri el, amelyet minden kimenő vonalán kiküld. Elvárás, hogy a vonal másik végén lévő router válaszolt küldjön vissza, amelyben közli az azonosítóját (, ami globálisan egyedi!).
2. A késleltetés meghatározása, amelynek legközvetlenebb módja egy speciális `ECHO` csomag küldése, amelyet a másik oldalnak azonnal vissza kell küldenie. A körbeérési idő felével becsülhető a késleltetés. (Javítás lehet a többszöri kísérlet átlagából számított érték.)
3. Az adatok összegzése, és csomag előállítása a megismert információkról. A kapcsolatállapot tartalma: a feladó azonosítója, egy sorszám, egy korérték és a szomszédok listája. Minden szomszédhoz megadják a felé tapasztalható késleltetést. Az előállítás történhet periodikusan vagy hiba esemény esetén. (Un. LSA – Link State Advertisment, azaz kapcsolatállapot hírdetés)

Kapcsolatállapot alapú forgalomirányítás működése

4. A kapcsolat csomagok megbízható szétosztása. Erre használható az elárasztás módszere, viszont a csomagban van egy sorszám, amely minden küldésnél 1-gyel nő. A router-ek számon tartanak minden (forrás,sorszám) párt, amelyet látnak. Ha új érkezik, akkor azt küldik minden vonalon, kivéve azon, amin érkezett. A másod példányokat eldobják. A kisebb sorszámúakat elavultnak tekintik, és nem küldik tovább.

Probléma	Megoldás
Sorszámok egy idő után körbe érnek	32 bites sorszám használata
Router összeomlik	Kor bevezetése. A kor értéket másod-percenként csökkenti a router, ha a kor eléri a nullát, akkor el kell dobni.
A sorszám mező megsérül	

- **További finomítások:** tároló területre kerül először a csomag és nem a küldési sorba; nyugtázás

Kapcsolatállapot alapú forgalomirányítás működése

5. Új útvonalak számítása. Amint egy router a kapcsolatállapot csomagok egy teljes készletét összegyűjtötte, megszerkesztheti az alhálózat teljes gráfját, mivel minden kapcsolat képviselve van. Erre lefuttatható Dijkstra algoritmusa, eredményeképp pedig megkapjuk a forgalomirányító táblát.

JELLEMZŐK

- A router-ek és a router-ek szomszédinak átlagos számával arányos tárterület kell az algoritmus futtatásához. *O(kn)*, ahol *k* a szomszédok száma és *n* a router-ek száma. Azaz nagy hálózatok esetén a számítás költséges és memória igényes lesz.
- A hardver- és szoftver-problémák komoly gondot okozhatnak. A hálózat méretének növekedésével a hiba valószínűsége is nő.

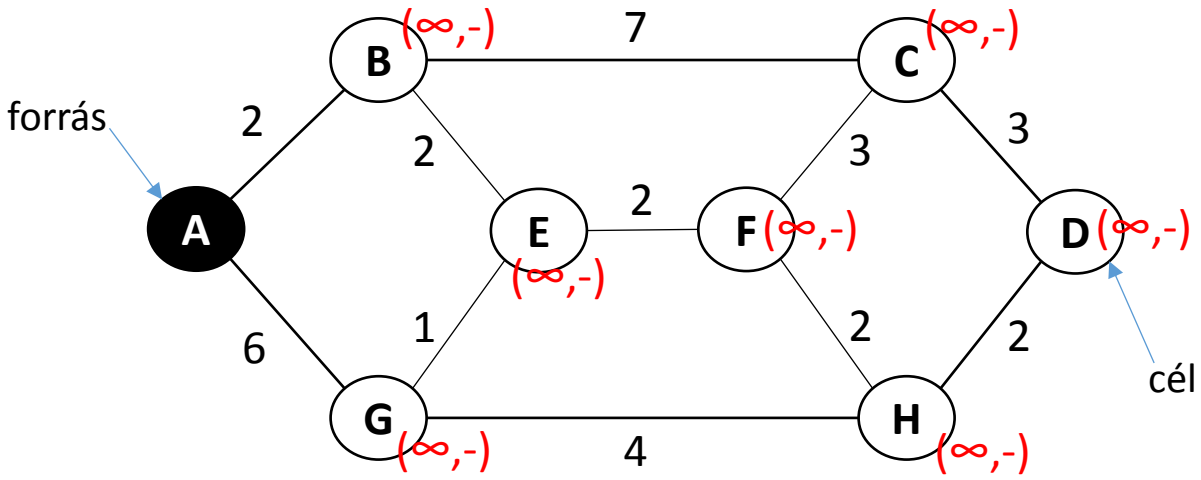
Dijkstra algoritmus (1959)

- Statikus algoritmus
- **Cél:** két csomópont közötti legrövidebb út meghatározása.

INFORMÁLIS LEÍRÁS

- Minden csomópontot felcímkézünk a forrás csomóponttól való legrövidebb ismert út mentén mért távolságával.
  - Kezdetben a távolság végtelen, mivel nem ismerünk útvonalat.
- Az algoritmus működése során a címkék változhatnak az utak megtalálásával. Két fajta címkét különböztetünk meg: ideiglenes és állandó. Kezdetben minden címke ideiglenes. A legrövidebb út megtalálásakor a címke állandó címkévé válik, és továbbá nem változik.

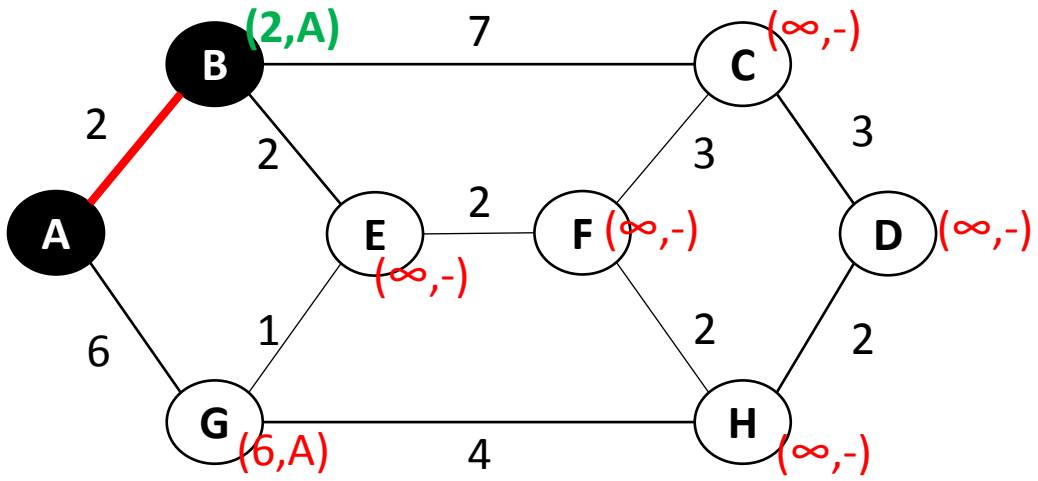
Dijkstra algoritmus - Példa



E´ = {}

Q = {(B,2),(G,6)}

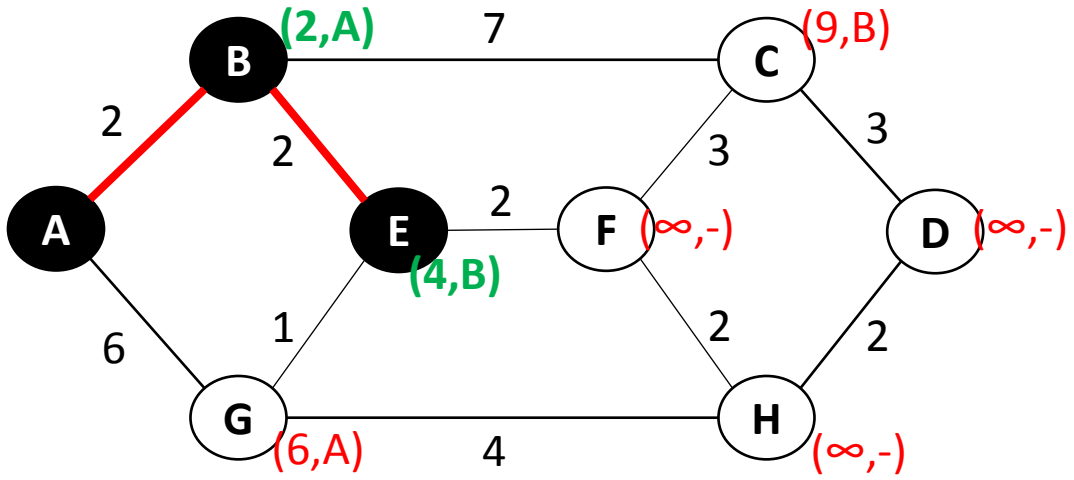
Dijkstra algoritmus - Példa



E´ = {(A,B)}

Q = {(E,4),(G,6),(C,9)}

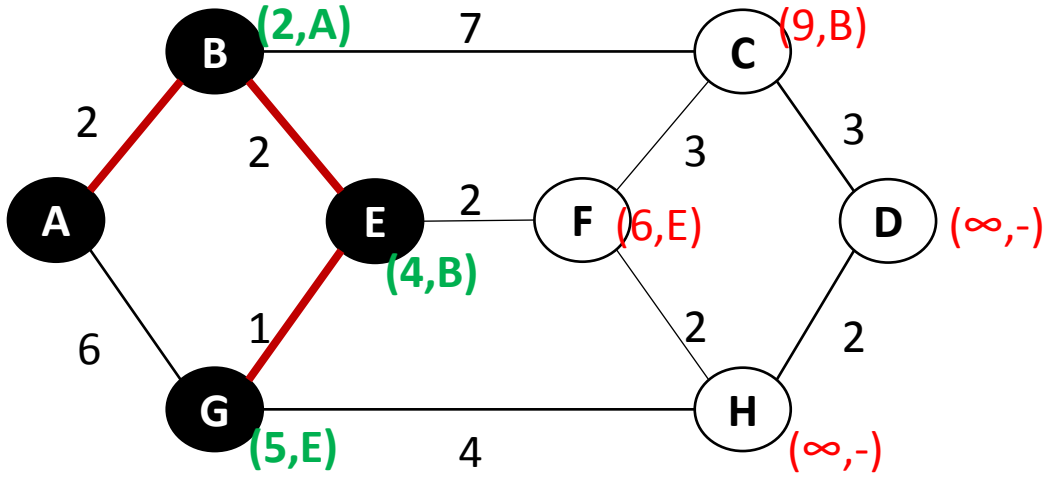
Dijkstra algoritmus - Példa



E´ = {(A,B),(B,E)}

Q = {(G,5),(F,6),(C,9)}

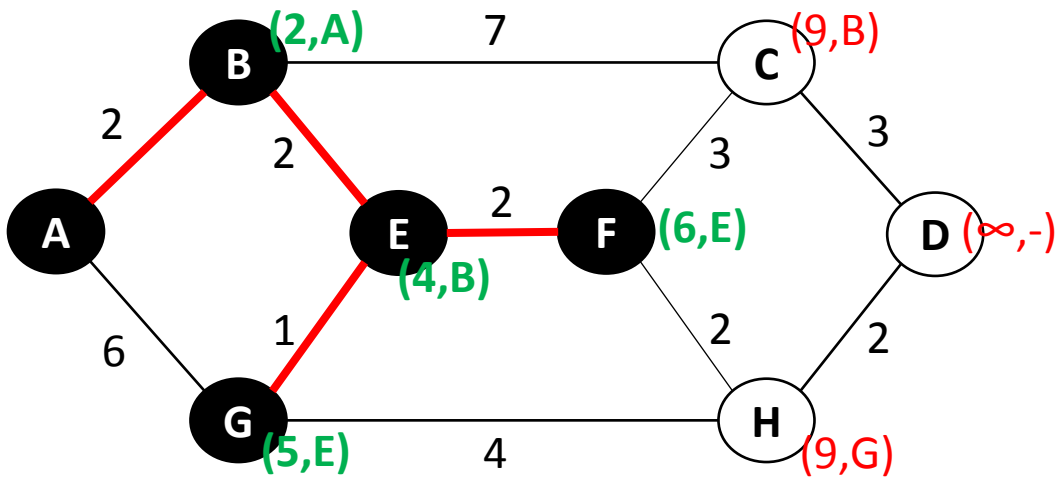
Dijkstra algoritmus - Példa



E´ = {(A,B),(B,E),
(E,G)}

Q = {(F,6),(C,9)
(H,9)}

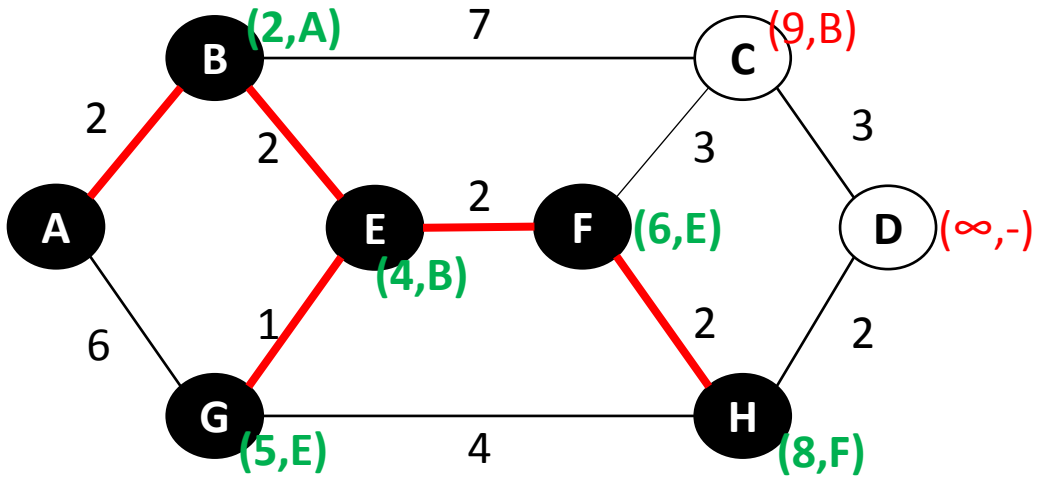
Dijkstra algoritmus - Példa



$E' = \{(A,B),(B,E),(E,G),(E,F)\}$

$Q = \{(H,8),(C,9)\}$

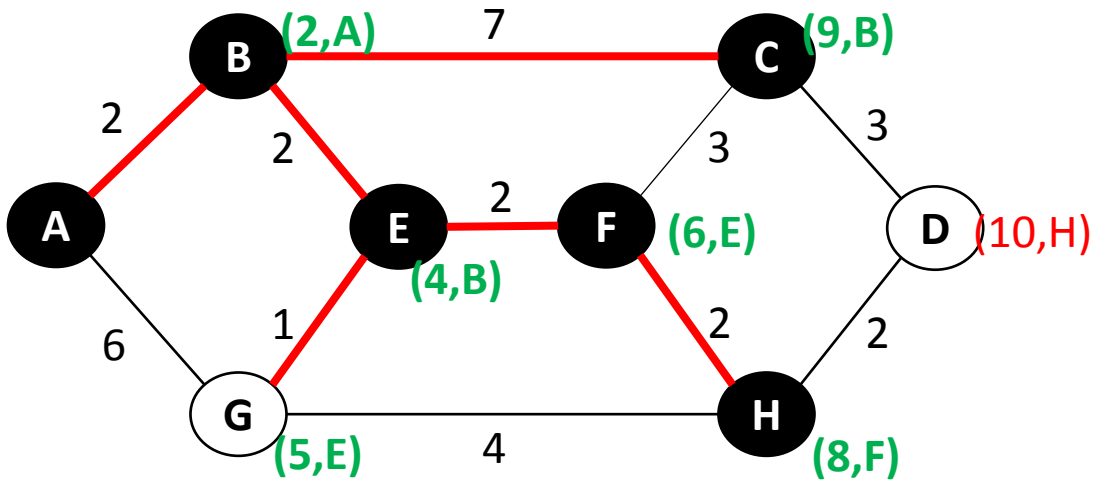
Dijkstra algoritmus - Példa




```
E´ = {(A,B),(B,E),
      (E,G),(E,F),
      (F,H)}

Q = {(C,9),(D,10)}
```

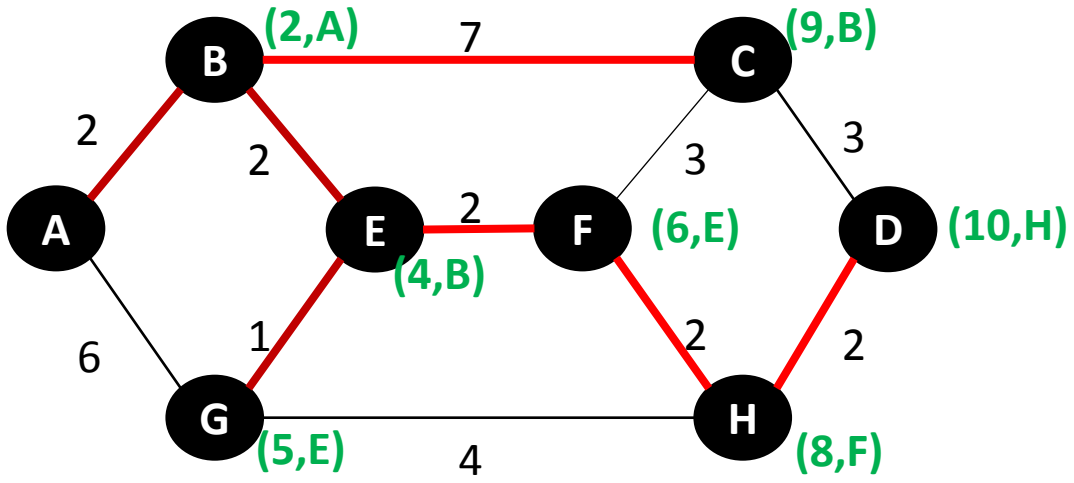
Dijkstra algoritmus - Példa




```
E´ = {(A,B),(B,E),
      (E,G),(E,F),
      (F,H),(B,C)}

Q = {(D,10)}
```

Dijkstra algoritmus - Példa




```
E´ = {(A,B),(B,E),
      (E,G),(E,F),
      (F,H),(B,C)
      (H,D)}

Q = {}
```

Dijkstra algoritmus pszeudo-kód

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

OSPF vs. IS-IS

- Két eltérő implementáció a link-state routing stratégiának

OSPF

- Cégek és adatközpontok
- Több lehetőséget támogat
- IPv4 felett
  - LSA-k IPv4 feletti küldése
  - OSPFv3 szükséges az IPv6-hoz

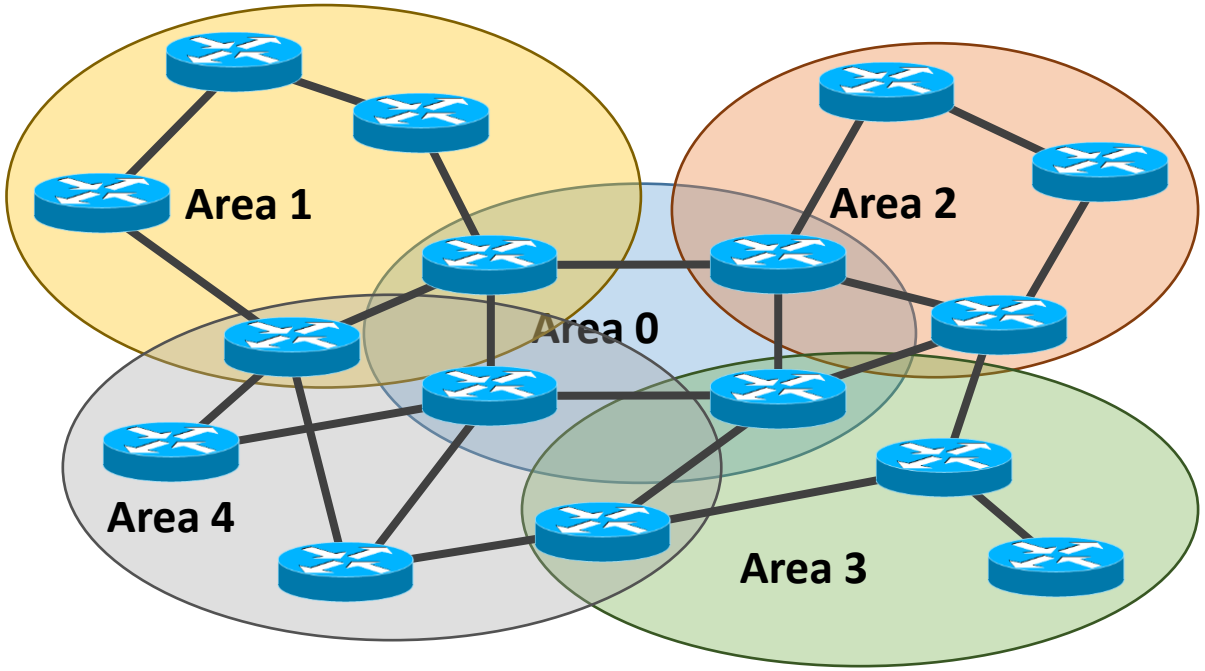
IS-IS

- Internet szolgáltatók által használt
- Sokkal tömörebb
  - Kisebb hálózati overhead
  - Több eszközt támogat
- Nem kötődik az IP-hez
  - Működik mind IPv4-gyel és IPv6-tal

Eltérő felépítés

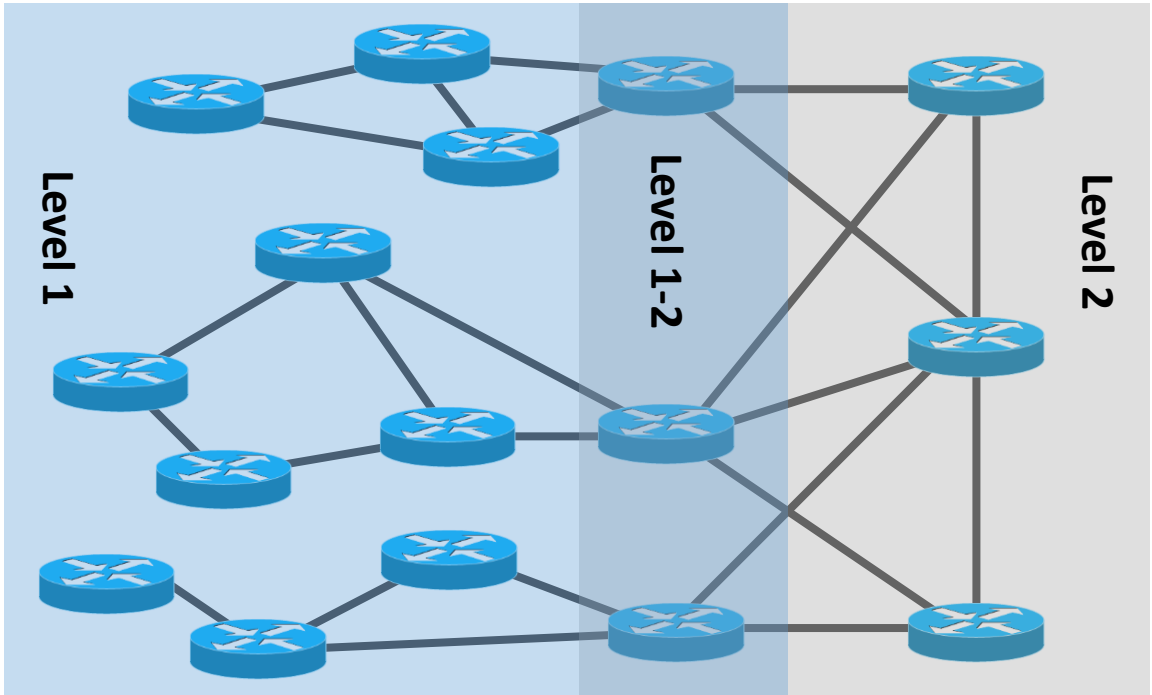
OSPF

- Átfedő területek köré szerveződik
- Area 0 a hálózat magja



IS-IS

- 2-szintű hierarchia
- A 2. szint a gerinchálózat

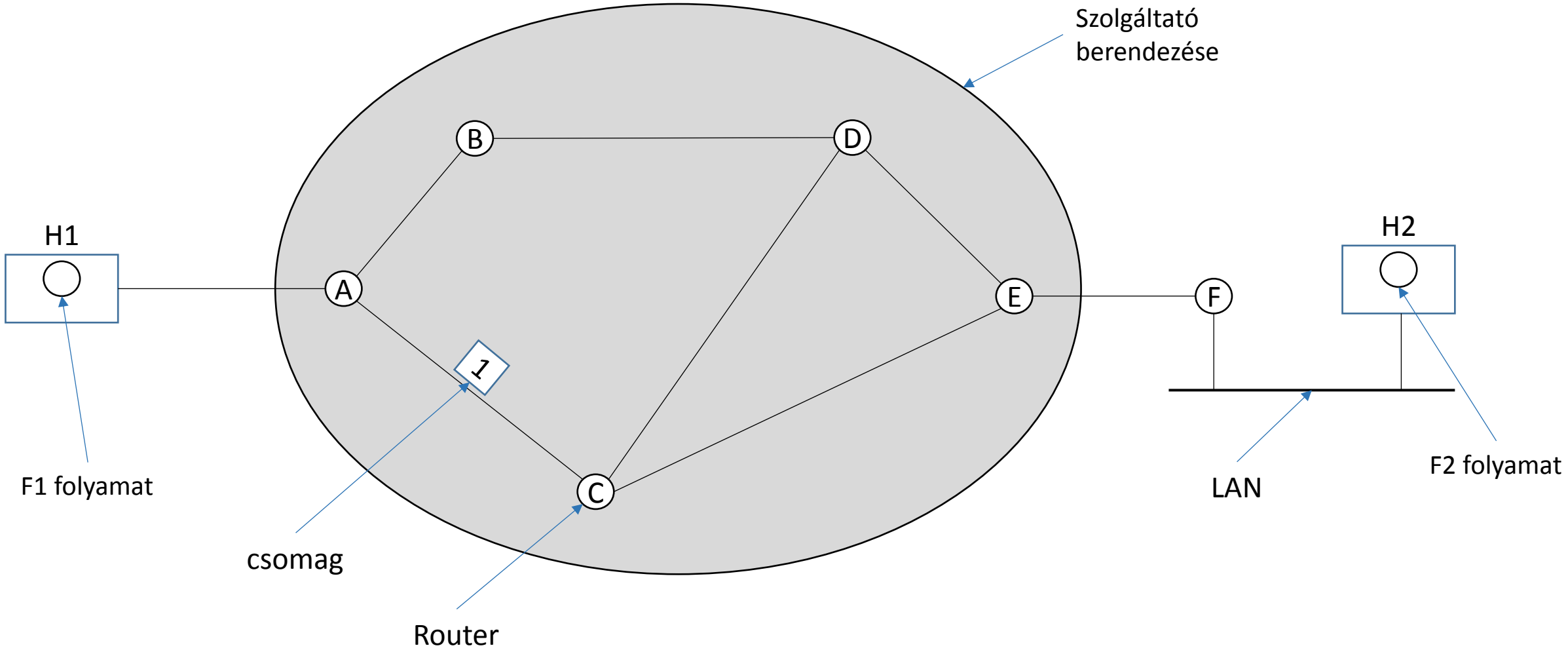


Link State vs. Distance Vector

	Link State	Distance Vector
Message Complexity	$O(n^2*e)$	O(d*n*k)
Time Complexity	O(n*log n)	O(n)
Convergence Time	O(1)	O(k)
Robustness	• Nodes may advertise incorrect link costs • Each node computes their own table	• Nodes may advertise incorrect path cost • Errors propagate due to sharing of DV tables

- Which is best?
- In practice, it depends.
- In general, link state is more popular.

Hálózati réteg protokolljai - *Környezet*



Szállítási réteg felé nyújtott szolgálatok

VEZÉRELVEK

1. A szolgálat legyen független az alhálózat kialakításától.
2. A szállítási réteg felé el kell takarni a jelenlevő alhálózatok számát, típusát és topológiáját.
3. A szállítási réteg számára rendelkezésre bocsájtott hálózati címeknek egységes számozási rendszert kell alkotniuk, még *LAN*-ok és *WAN*-ok esetén is.

SZOLGÁLATOK KÉT FAJTÁJÁT KÜLÖNBÖZTETIK MEG

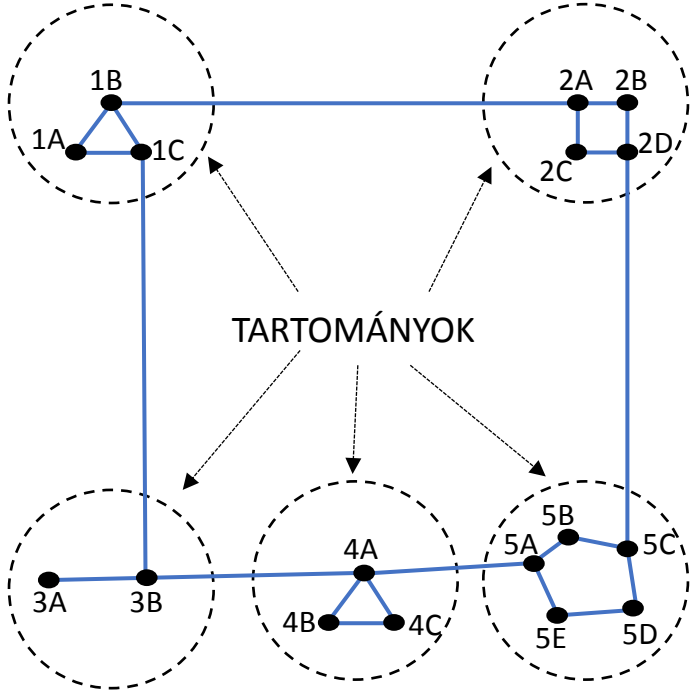
- Összeköttetés nélküli szolgálat (*Internet*)
  - datagram alhálózat
- Összeköttetés alapú szolgálat (*ATM*)
  - virtuális áramkör alhálózat

HÁLÓZATI RÉTEG – FORGALOMIRÁNYÍTÁS

Hierarchikus forgalomirányítás

MOTIVÁCIÓ

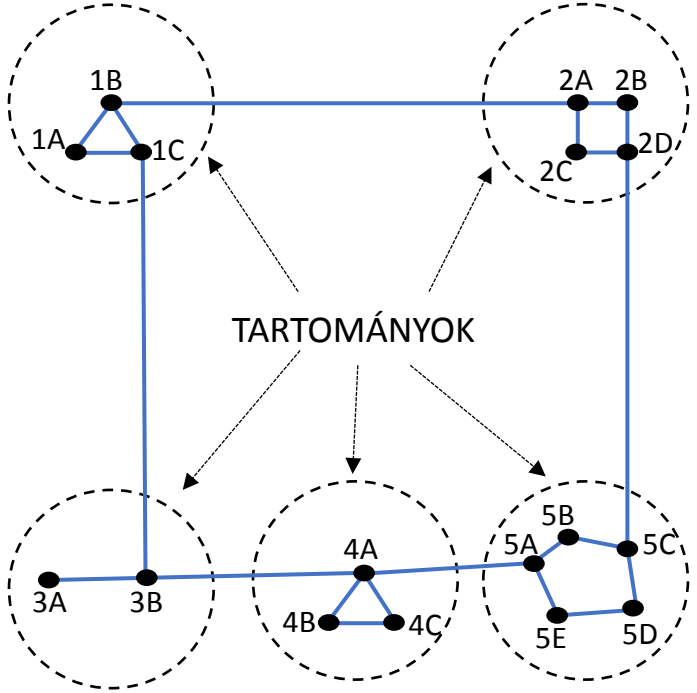
- A hálózat méretének növekedésével a router-ek forgalomirányító táblázatai is arányosan nőnek.
  - ➢A memória, a CPU és a sávszélesség igény is megnövekszik a router-eknél.
- *Ötlet:* telefonhálózatokhoz hasonlóan hierarchikus forgalomirányítás alkalmazása.



Hierarchikus forgalomirányítás

JELLEMZŐK

- A router-eket tartományokra osztjuk. A saját tartományát az összes router ismeri, de a többi belső szerkezetéről nincs tudomása.
- Nagy hálózatok esetén többszintű hierarchia lehet szükséges.
- N darab router-ből álló alhálózathoz az optimális szintek száma $\ln N$, amely router-enként $e * \ln N$ bejegyzést igényel. (*Kamoun és Kleinrock, 1979*)



Adatszóró forgalomirányítás

- **Adatszórás** (vagy angolul *broadcasting*) – egy csomag mindenhová történő egyidejű küldése.
- Több féle megvalósítás lehetséges:
  1. **Külön csomag küldése** minden egyes rendeltetési helyre
     - *sávszélesség pazarlása, lista szükséges hozzá*
  2. **Elárasztás.**
     - *kétpontos kommunikációhoz nem megfelelő*

Adatszóró forgalomirányítás

3. **Többcélú forgalomirányítás** (vagy angolul *multidestination routing*). Csomagban van egy lista a rendeltetési helyekről, amely alapján a router-ek eldöntik a vonalak használatát, mindegyik vonalhoz készít egy másolatot és belerakja a megfelelő célcím listát.
4. **A forrás router-hez tartozó nyelőfa használata**. A feszítőfa (vagy angolul *spanning tree*) az alhálózat részhalmaza, amelyben minden router benne van, de nem tartalmaz köröket. Ha minden router ismeri, hogy mely vonalai tartoznak a feszítőfához, akkor azokon továbbítja az adatszóró csomagot, kivéve azon a vonalon, amelyen érkezett.
   - *nem mindig ismert a feszítőfa*

Adatszóró forgalomirányítás 2/2

5. **Visszairányú továbbítás** (vagy angolul *reverse path forwarding*). Amikor egy adatszórásos csomag megérkezik egy routerhez, a router ellenőrzi, hogy azon a vonalon kapta-e meg, amelyen rendszerint ő szokott az adatszórás forrásához küldeni. Ha igen, akkor nagy esély van rá, hogy az adatszórásos csomag a legjobb utat követte a router-től, és ezért ez az első másolat, amely megérkezett a router-hez. Ha ez az eset, a router kimásolja minden vonalra, kivéve arra, amelyiken érkezett. Viszont, ha az adatszórásos csomag más vonalon érkezett, mint amit a forrás eléréséhez előnyben részesítünk, a csomagot eldobják, mint valószínű másodpéldányt.

Többes-küldéses forgalomirányítás

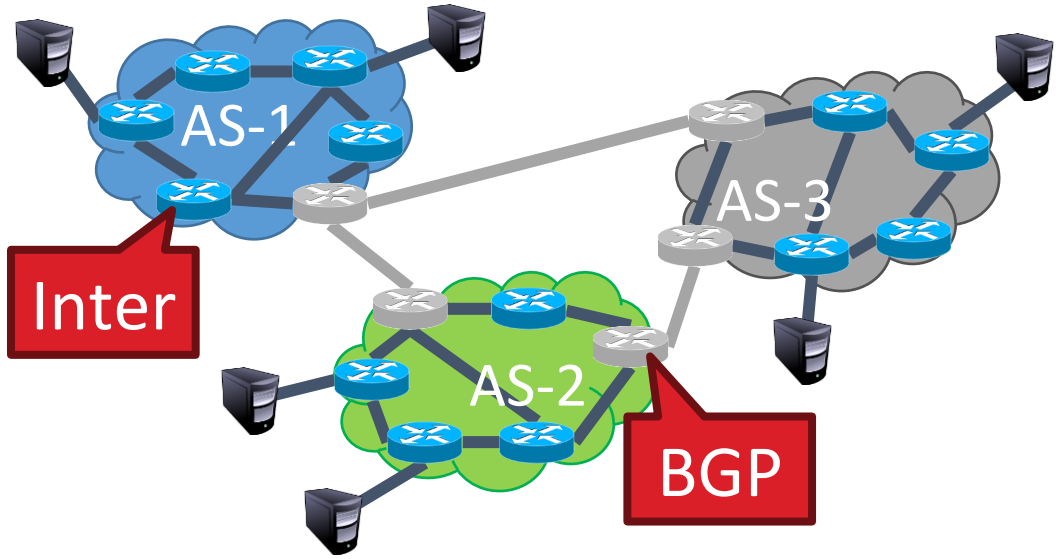
- **Többes-küldés** (vagy angolul *multicasting*) – egy csomag meghatározott csoporthoz történő egyidejű küldése.

MULTICAST ROUTING

- Csoport kezelés is szükséges hozzá: létrehozás, megszüntetés, csatlakozási lehetőség és leválasztási lehetőség. (Ez nem a forgalomirányító algoritmus része!)
- Minden router kiszámít egy az alhálózatban az összes többi *router*t lefedő feszítőfát.
- Többes-küldéses csomag esetén az első router levágja a feszítőfa azon ágait, amelyek nem csoporton belüli hoszthoz vezetnek. A csomagot csak a csonkolt feszítőfa mentén továbbítják.

Hierarchikus forgalomirányítás IP

- Hierarchikus (2 szintű)
  - AS-ek közötti:
    - EGP
    - Exterior Gateway Protocols
    - Tartományok közötti
  - AS-en belüli
    - IGP
    - Interior Gateway Protocols
    - Tartományon belüli
- AS – Autonom System – Autonóm Rendszer



Hálózati réteg az Interneten

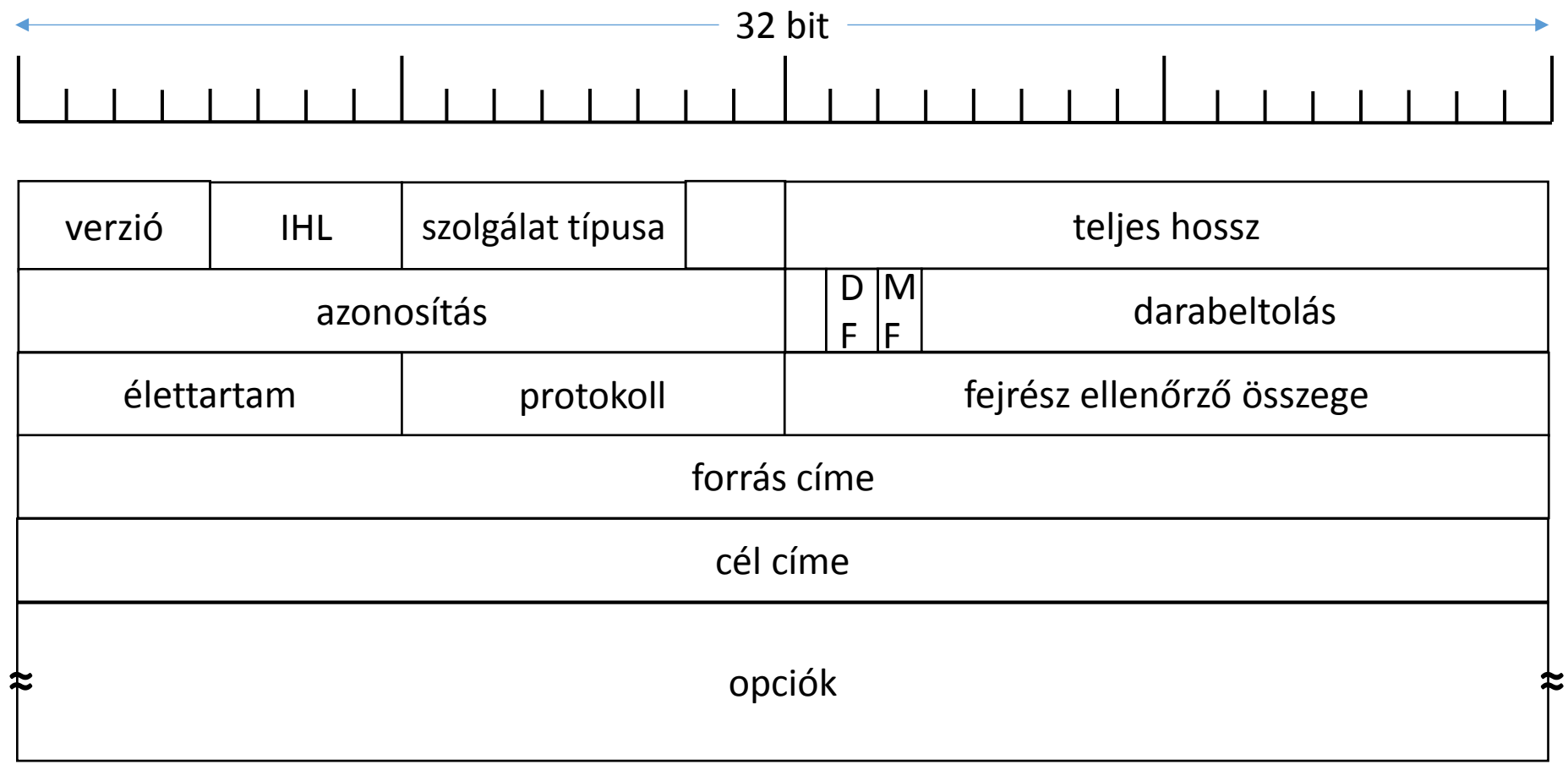
- A hálózati réteg szintjén az internet autonóm rendszerek összekapcsolt együttesének tekinthető.
  - Nincs igazi szerkezete, de számos főbb *gerinchálózata* létezik.
  - A gerinchálózatokhoz csatlakoznak a területi illetve regionális hálózatok.
  - A regionális és területi hálózatokhoz csatlakoznak az egyetemeken, vállalatoknál és az internet szolgáltatóknál lévő LAN-ok.
- Az internet protokollja, az IP.

Hálózati réteg az Interneten

- Az Interneten a kommunikáció az alábbi módon működik:
  1. A szállítási réteg viszi az adatfolyamokat és datagramokra tördeli azokat.
  2. Minden datagram átvitelre kerül az Interneten, esetleg menet közben kisebb egységekre darabolva.
  3. A célgép hálózati rétege összeállítja az eredeti datagramot, majd átadja a szállítási rétegének.
  4. A célgép szállítási rétege beilleszti a datagramot a vételi folyamat bemeneti adatfolyamába.

Hálózati réteg – Címzés

Az IP fejrésze



Az IP fejrésze

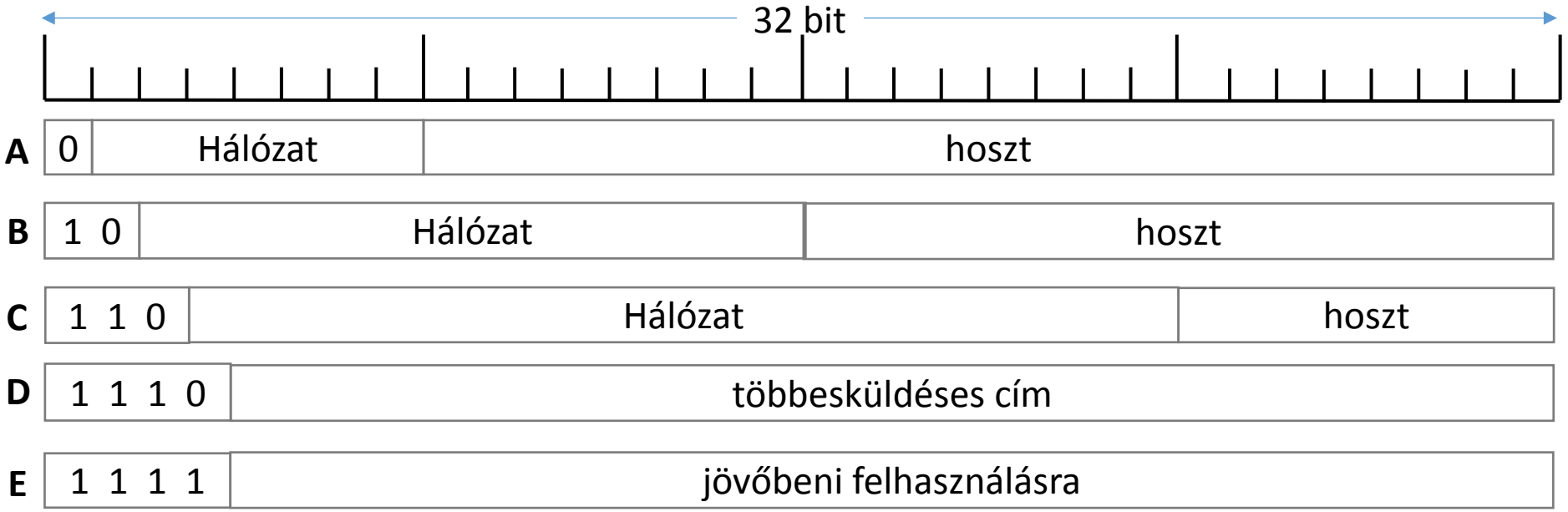
- **verzió:** IP melyik verzióját használja (jelenleg 4 és 6 közötti átmenet zajlik)
- **IHL**: a fejléc hosszát határozza meg 32-bites szavakban mérve, legkisebb értéke 5.
- **szolgálat típusa**: szolgálati osztályt jelöl (3-bites precedencia, 3 jelzőbit [D,T,R])
- **teljes hossz:** fejléc és adatrész együttes hossza bájtokban
- **azonosítás:** egy datagram minden darabja ugyanazt az *azonosítás* értéket hordozza.
- **DF:** „ne darabold” flag a router-eknek
- **MF**: „több darab” flag minden darabban be kell legyen állítva, kivéve az utolsót.
- **darabeltolás**: a darab helyét mutatja a datagramon belül. (elemi darab méret 8 bájt)

Az IP fejrésze

- **élettartam**: másodpercenként kellene csökkenteni a mező értékét, minden ugrásnál csökkentik eggyel az értékét
- **protokoll:** szállítási réteg protokolljának azonosítóját tartalmazza
- **ellenőrző összeg:** a router-eken belüli rossz memóriaszavak által előállított hibák kezelésére használt ellenőrző összeg a fejrészre, amelyet minden ugrásnál újra kell számolni
- **forrás cím** és **cél cím**: IP cím (később tárgyaljuk részletesen)
- **opciók:** következő verzió bővíthetősége miatt hagyták benne. Eredetileg 5 opció volt. (router-ek általában figyelmen kívül hagyják)

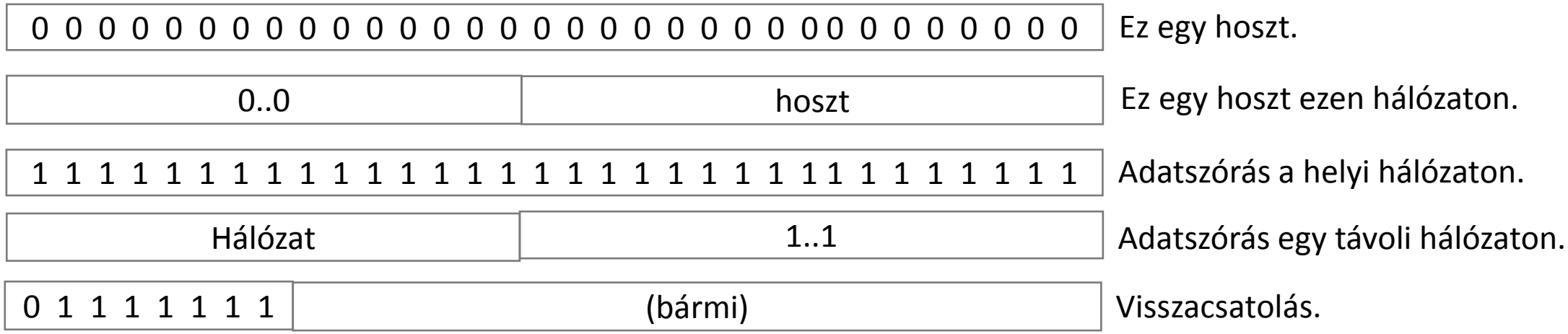
IP cím

- Minden hoszt és minden router az Interneten rendelkezik egy IP-címmel, amely a hálózat számát és a hoszt számát kódolja. (*egyedi kombináció*)
- 4 bájton ábrázolják az IP-címet.
- Több évtizeden keresztül 5 osztályos címzést használtak: A,B, C, D és E.

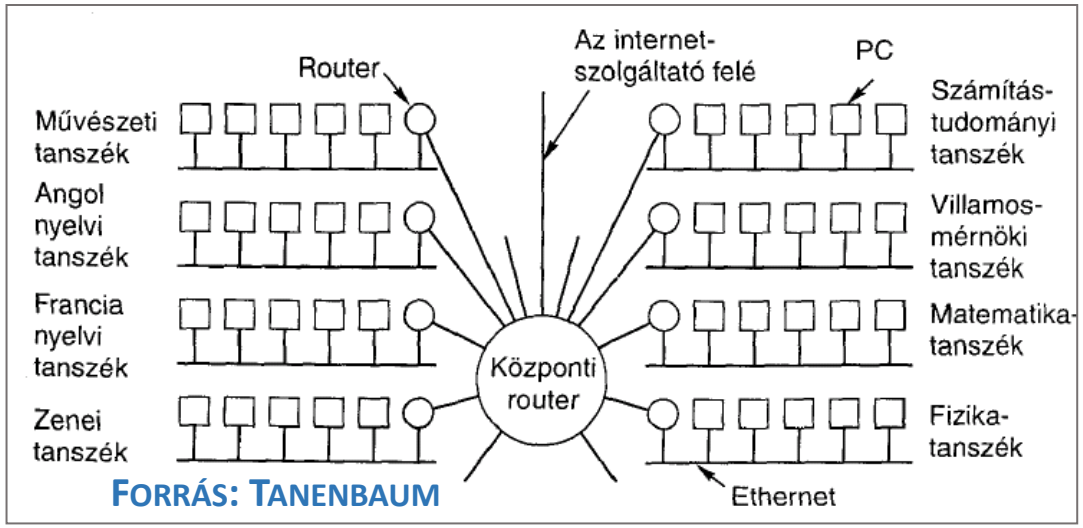


IP cím

- Az IP-t pontokkal elválasztott decimális rendszerben írják. Például: *192.168.0.1*
- Van pár speciális cím. Lásd az alábbiakban.



IP cím – alhálózatok



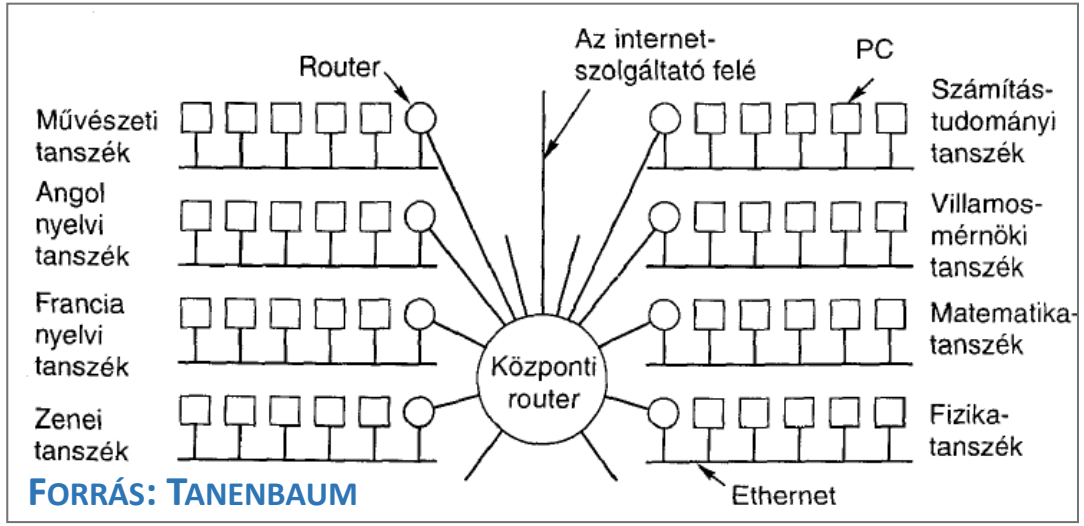
FORRÁS: TANENBAUM

- Az azonos hálózatban lévő hosztok ugyanazzal a hálózatszámmal rendelkeznek.
- Egy hálózat belső felhasználás szempontjából több részre osztódhat, de a külvilág számára egyetlen hálózatként jelenik meg.
  - Alhálózat (avagy angolul *subnet*)

IP cím – alhálózatok

AZONOSÍTÁS

- alhálózati maszk (avagy angolul *subnet mask*) ismerete kell a routernek
  - Két féle jelölés *IP-cím jellegű* vagy a *fix pozíciók száma.*
- A forgalomirányító táblázatba a router-eknél *(hálózat,0)* és *(saját hálózat, hoszt)* alakú bejegyzések.
- Ha nincs találat, akkor az alapértelmezett router felé továbbítják a csomagot.



FORRÁS: TANENBAUM

IP cím – CIDR

- IP címek gyorsan fogytak. 1996-ban kötötték be a 100.000-edik hálózatot.
  - Az osztályok használata sok címet elpazarolt. (B osztályú címek népszerűsége)
- ***Megoldás***: osztályok nélküli környezetek közötti forgalomirányítás (CIDR).
  - Például 2000 cím igénylése esetén 2048 méretű blokk kiadása.
- Forgalomirányítás megbonyolódik:
  - Minden bejegyzés egy 32-bites maszkkal egészül ki.
  - Egy bejegyzés innentől egy hármassal jellemezhető: (*ip-cím*, *alhálózati maszk*, *kimeneti vonal*)
  - Új csomag esetén a cél címből kimaszkolják az alhálózati címet, és találat esetén a leghosszabb illeszkedés felé továbbítják.
- Túl sok bejegyzés keletkezik.
  - Csoportos bejegyzések használata.

CIDR címzés példa

Mi történik, ha a router egy 135.46.57.14 IP cím felé tartó csomagot kap?

/22-ES CÍM ESETÉN

10001011 00101110 00111001 00001110
AND 11111111 11111111 11111100 00000000
10001011 00101110 00111000 00000000

Kimaszkolás eredménye

/23-ES CÍM ESETÉN

10001011 00101110 00111001 00001110
AND 11111111 11111111 11111110 00000000
10001011 00101110 00111000 00000000

- Vagyis *135.46.56.0/22*-as vagy *135.46.56.0/23*-as bejegyzést kell találni, azaz jelen esetben a *0.interface* felé történik a továbbítás.

Cím/maszk	Következő ugrás
135.46.56.0/22	0.interface
135.46.60.0/23	1.interface
192.53.40.0/23	1.router
Alapértelmezett	2.router

CIDR bejegyzés aggregálás példa

- Lehet-e csoportosítani a következő bejegyzéseket, ha feltesszük, hogy a *következő ugrás* mindegyiknél az *1.router*: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21, 57.6.120.0/21?

00111001 00000110 01100000 00000000
00111001 00000110 01101000 00000000
00111001 00000110 01110000 00000000
00111001 00000110 01111000 00000000

- Azaz az (57.6.96.0/19, 1.router) bejegyzés megfelelően csoportba fogja a 4 bejegyzést.

Forgalomirányítási tábla példa

Network Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	192.168.0.1	192.168.0.100	10
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
192.168.0.0	255.255.255.0	192.168.0.100	192.168.0.100	10
192.168.0.100	255.255.255.255	127.0.0.1	127.0.0.1	10
192.168.0.255	255.255.255.255	192.168.0.100	192.168.0.100	10

NAT

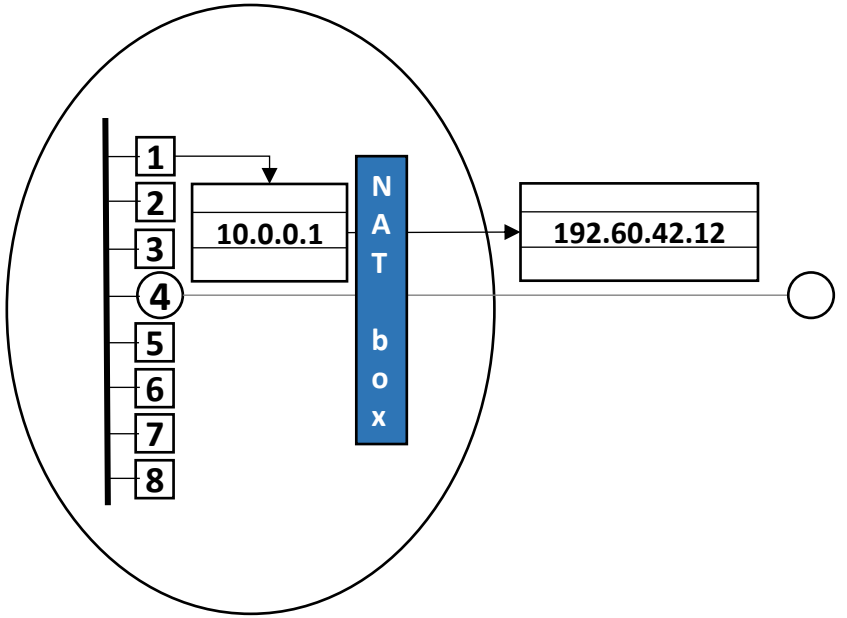
- Gyors javítás az IP címek elfogyásának problémájára. (hálózati címfordítás)

ALAPELVEK

- Az internet forgalomhoz minden cégnek egy vagy legalábbis kevés IP-címet adnak. A vállalaton belül minden számítógéphez egyedi IP-címet használnak a belső forgalomirányításra.
- A vállalaton kívüli csomagokban a címfordítást végzünk.
- 3 IP-címtartományt használunk:
  - 10.0.0.0/8, azaz 16 777 216 lehetséges hoszt;
  - 172.16.0.0/12, azaz 1 084 576 lehetséges hoszt;
  - 192.168.0.0/16, azaz 65 536 lehetséges hoszt;
- *NAT box* végzi a címfordítást

NAT

- Hogyan fogadja a választ?
  - A *port* mezők használata, ami mind a TCP, mind az UDP fejlécben van
  - Kimenő csomagnál egy mutatót tárolunk le, amit beírunk a *forrás port* mezőbe. *65536* bejegyzésből álló fordítási táblázatot kell a *NAT box*-nak kezelni.
  - A fordítási táblázatban benne van az eredeti IP és forrás port.
- **Ellenérvek**: sérti az IP architekturális modelljét, összeköttetés alapú hálózatot képez, rétegmodell alapelveit sérti, kötöttség a TCP és UDP fejléchez, szöveg törzsében is lehet az IP, szűkös port tartomány



Vége

- Köszönöm a figyelmet!