Számítógépes Hálózatok

7. Előadás: Adatkapcsolati réteg
Hálózati réteg

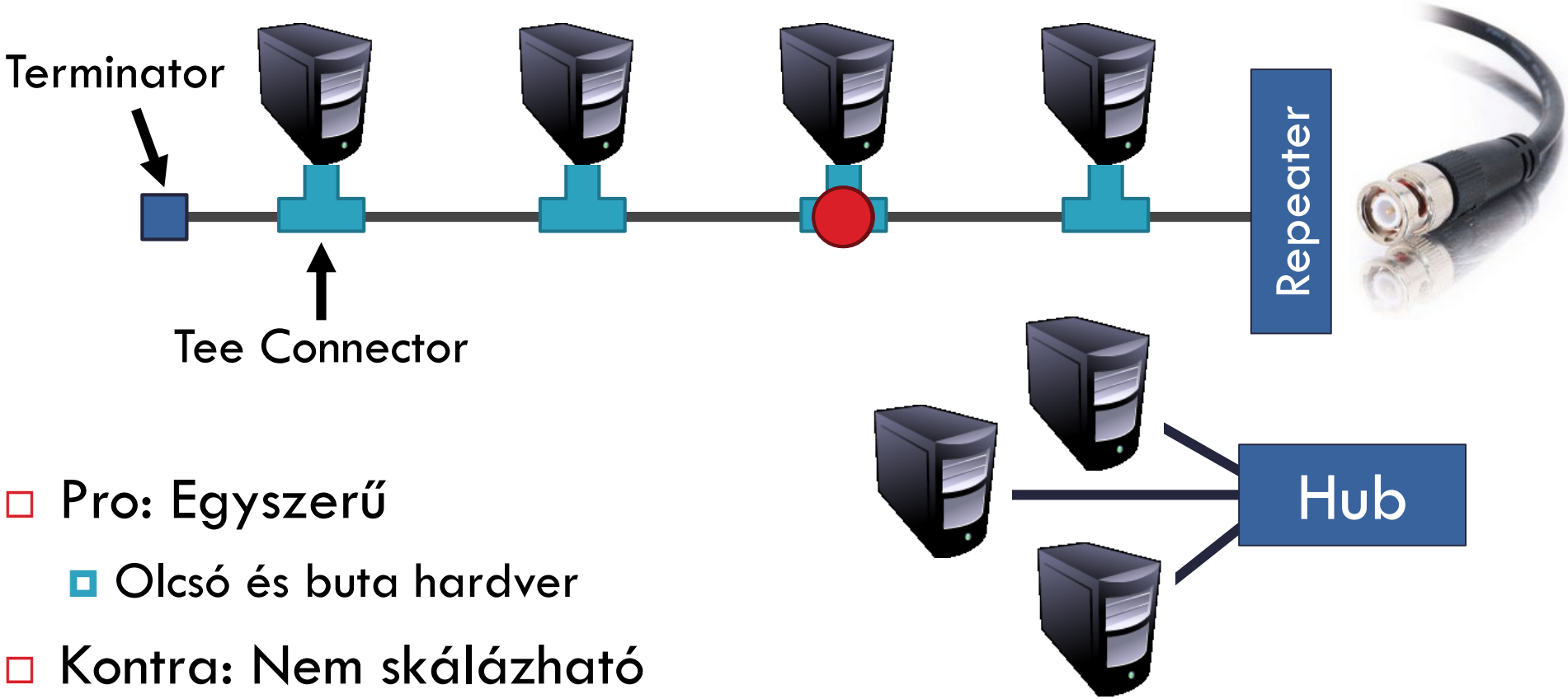
Based on slides from **Zoltán Ács ELTE** and D. Choffnes Northeastern U., Philippa Gill from StonyBrook University , Revised Spring 2016 by S. Laki

Az adatkapcsolati réteg „legtetején”…

- Bridging, avagy hidak
  - Hogyan kapcsoljunk össze LAN-okat?
- Funkciók:
  - Keretek forgalomirányítása a LAN-ok között
- Kihívások:
  - Plug-and-play, önmagát konfiguráló
  - Esetleges hurkok feloldása

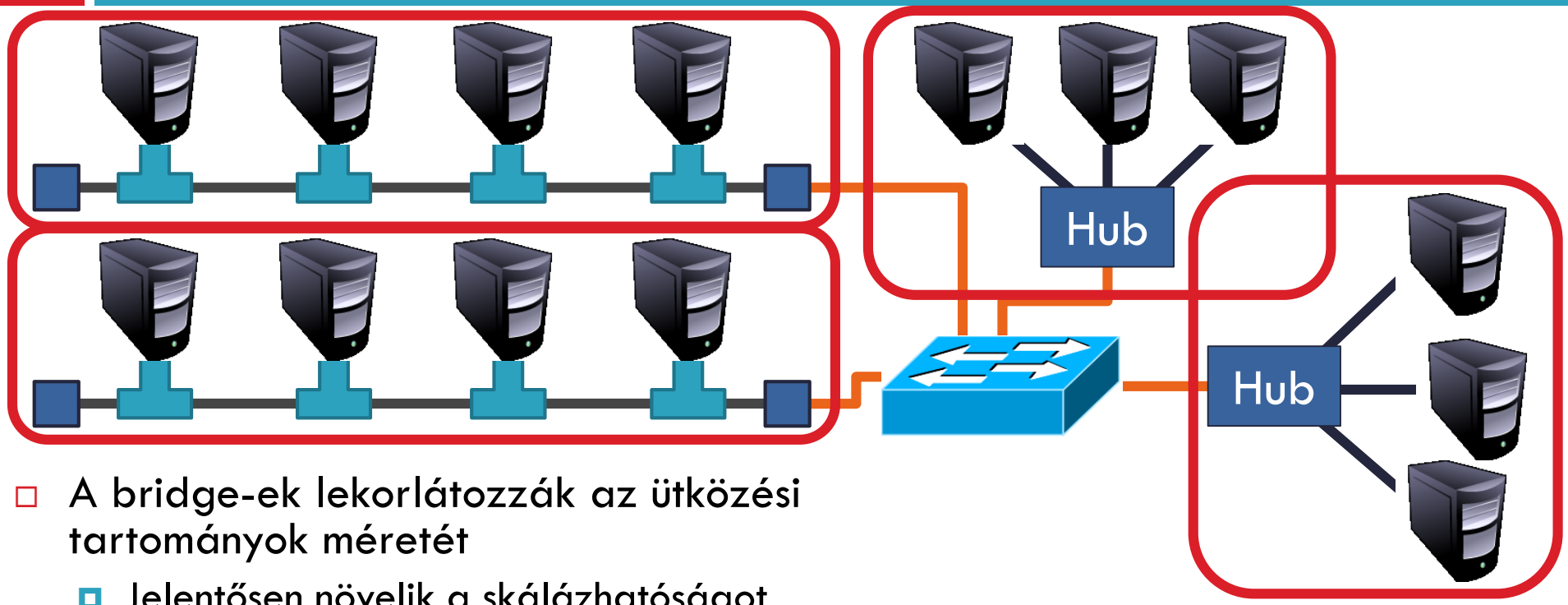
Visszatekintés

- Az Ethernet eredetileg adatszóró technológia volt



- Pro: Egyszerű
  - Olcsó és buta hardver
- Kontra: Nem skálázható
  - Több állomás = több ütközés = káosz

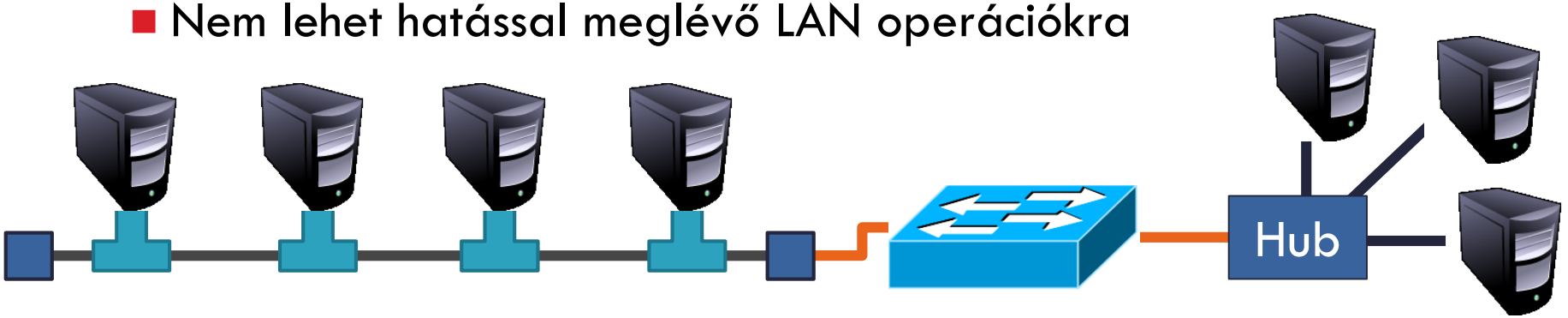
LAN-ok összekapcsolása



- A bridge-ek lekorlátozzák az ütközési tartományok méretét
  - Jelentősen növelik a skálázhatóságot
  - Kérdés: lehetne-e az egész Internet egy bridge-ekkel összekötött tartomány?
- Hátrány: a bridge-ek sokkal komplexebb eszközök a hub-oknál
  - Fizikai réteg VS Adatkapcsolati réteg
  - Memória pufferek, csomag feldolgozó hardver és routing (útválasztó) táblák szükségesek

Bridge-ek (magyarul: hidak)

- Az Ethernet switch eredeti formája
- Több IEEE 802 LAN-t kapcsol össze a 2. rétegben
- Célok
  - Ütközési tartományok számának csökkentése
  - Teljes átlátszóság
    - "Plug-and-play," önmagát konfiguráló
    - Nem szükségesek hw és sw változtatások a hosztokon/hub-okon
    - Nem lehet hatással meglévő LAN operációkra

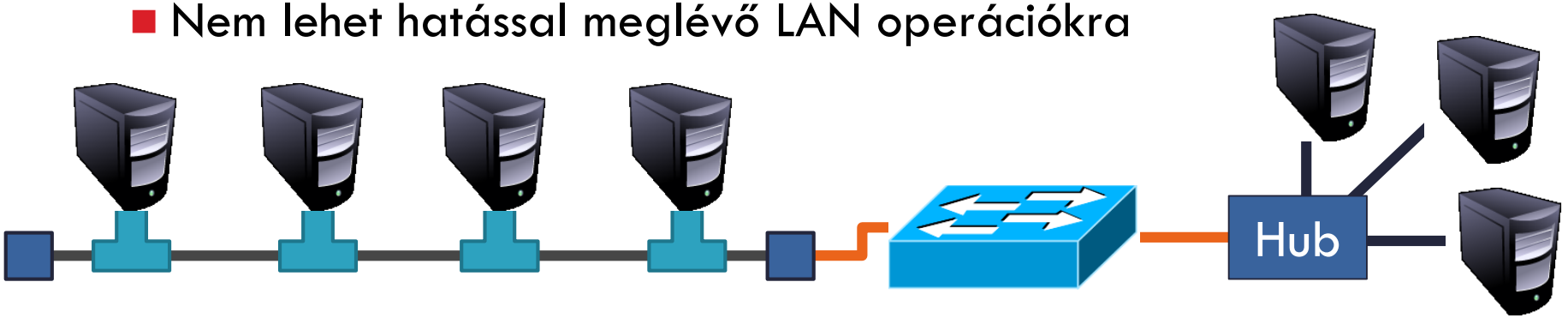


Bridge-ek (magyarul: hidak)

- Az Ethernet switch eredeti formája
-
-

1. Keretek továbbítása
2. (MAC) címek tanulása
3. Feszítőfa (Spanning Tree) Algoritmus (a hurkok kezelésére)

  - Nem szükségesek hw és sw változtatások a hosztokon/hub-okon
  - Nem lehet hatással meglévő LAN operációkra



Keret Továbbító Táblák

- Minden bridge karbantart egy továbbító táblát (forwarding table)

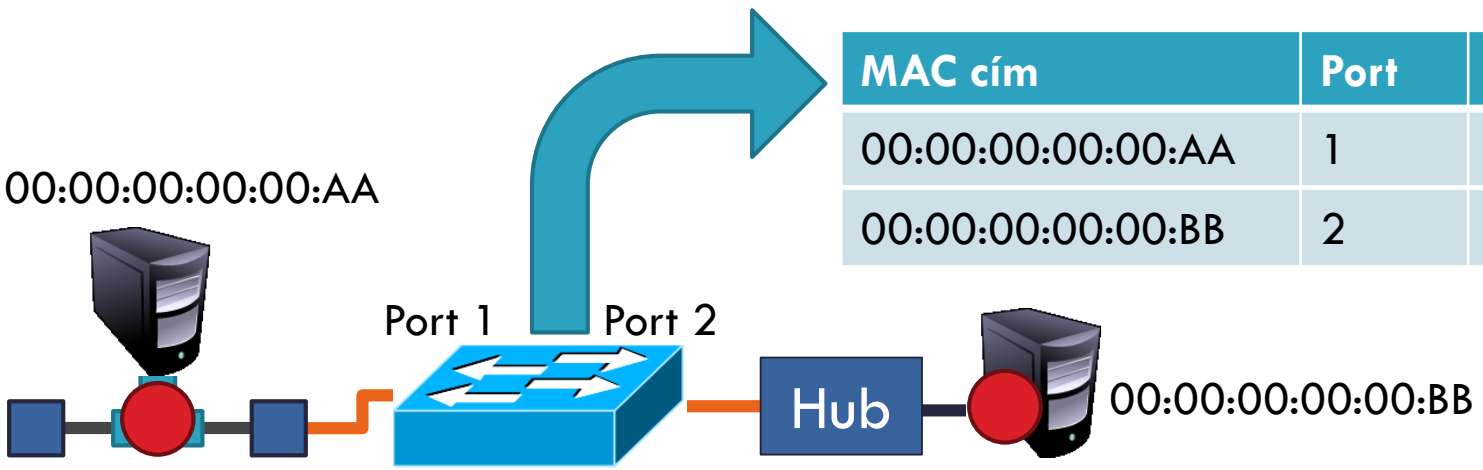
MAC Cím	Port	Kor
00:00:00:00:00:AA	1	1 perc
00:00:00:00:00:BB	2	7 perc
00:00:00:00:00:CC	3	2 mp
00:00:00:00:00:DD	1	3 perc

Címek tanulása

- Kézi beállítás is lehetséges, de…
  - Időigényes
  - Potenciális hiba forrás
  - Nem alkalmazkodik a változásokhoz (új hosztok léphetnek be és régiek hagyhatják el a hálózatot)
- Ehelyett: tanuljuk meg a címeket
  - Tekintsük a forrás címeit a különböző portok
    kereteknek --- képezzünk ebből egy táblázat

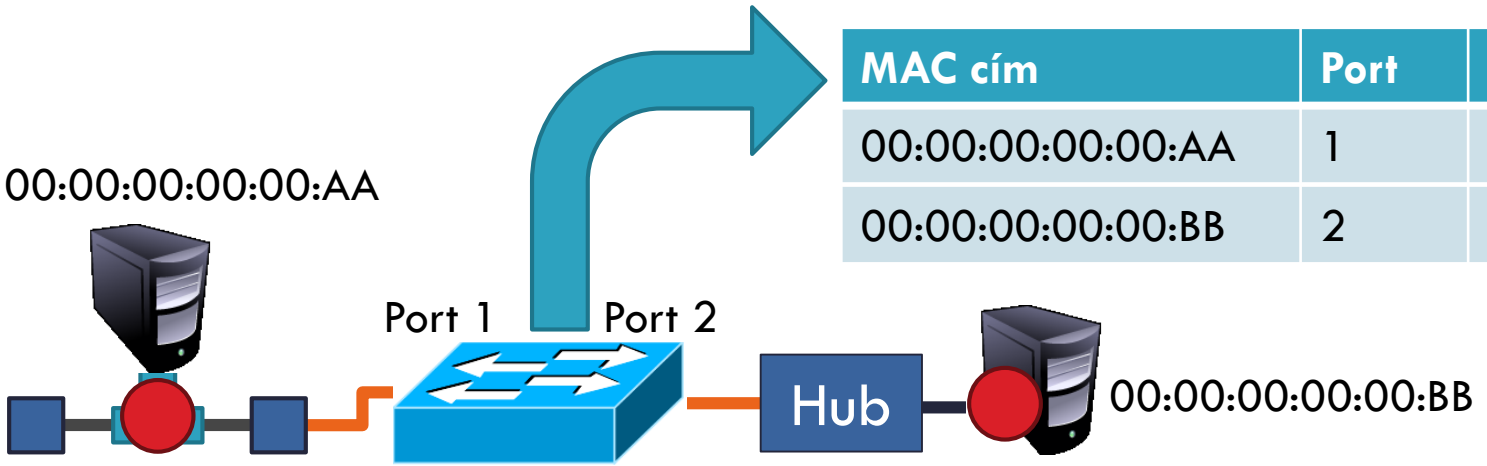
Töröljük a régi bejegyzéseket

MAC cím	Port	Kor
00:00:00:00:00:AA	1	0 minutes
00:00:00:00:00:BB	2	0 minutes



Címek tanulása

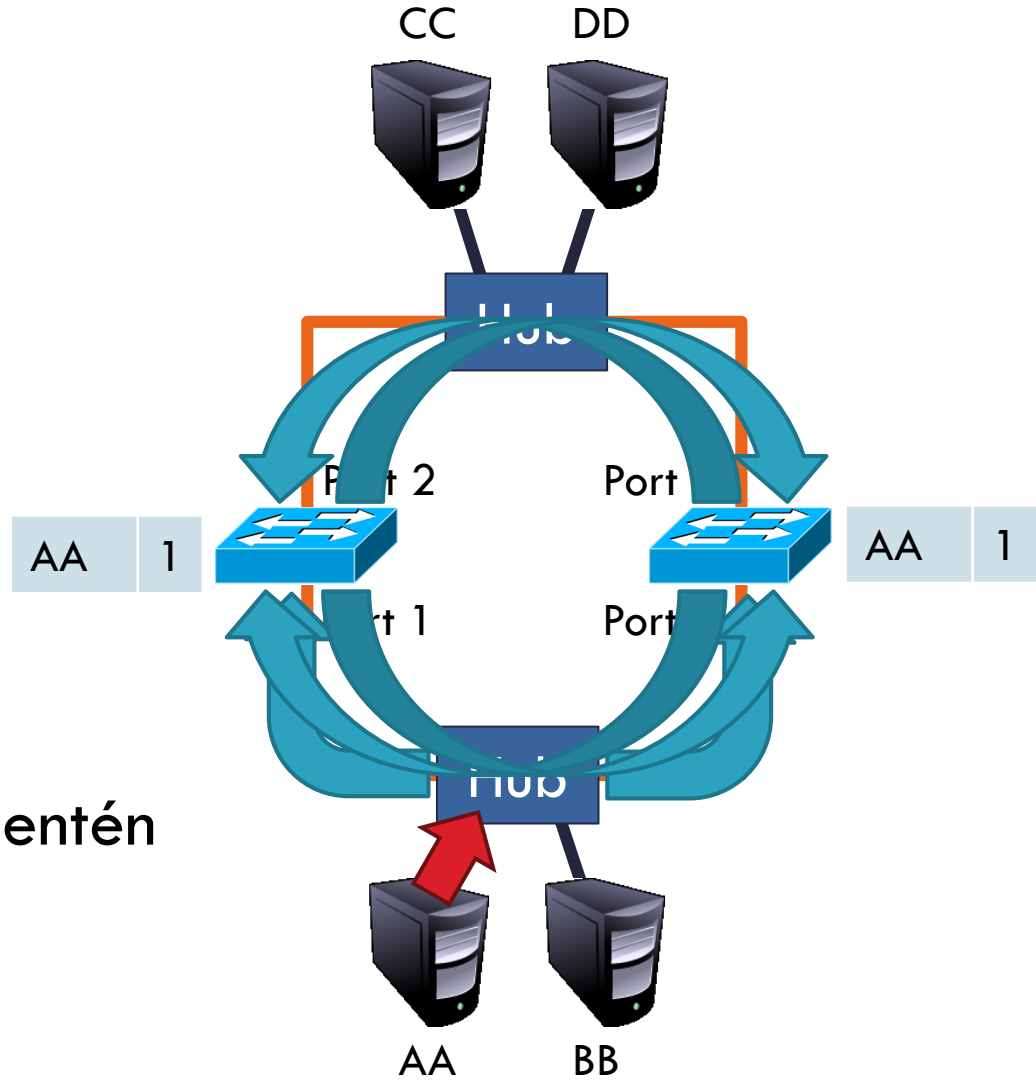
- Kézi beállítás is lehetséges, de…
  - Időigényes
  - Potenciális hiba forrás
  - Nem alkalmazkodik a változásokhoz (új hosztok léphetnek be és régiek hagyhatják el a hálózatot)
- Ehelyett: tanuljuk meg a címeket
  - Tekintsük a forrás címeit a különböző portokon beérkező kereteknek --- képezzünk ebből egy táblázatot



MAC cím	Port	Kor
00:00:00:00:00:AA	1	0 minutes
00:00:00:00:00:BB	2	0 minutes

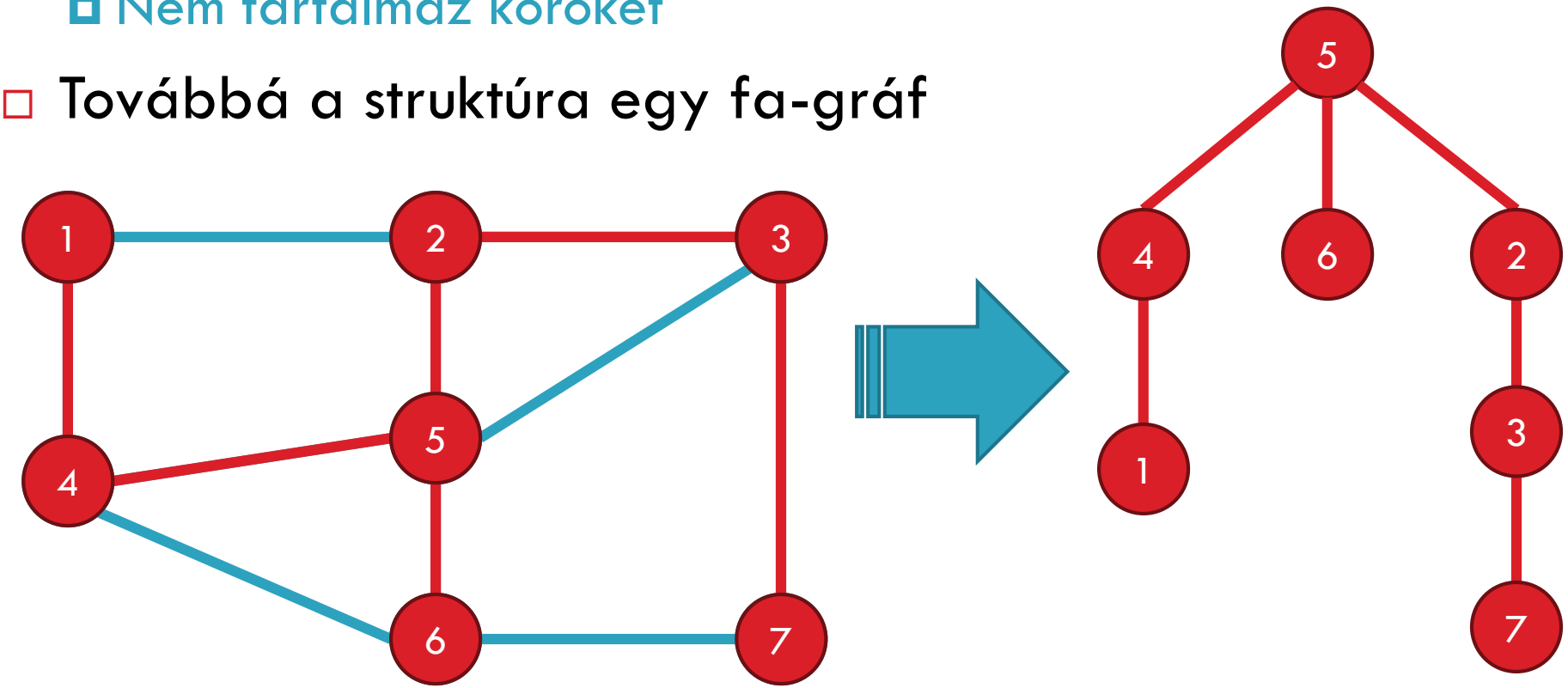
Hurkok problémája

- <Src=AA, Dest=DD>
- Ez megy a végtelenségig
  - Hogyan állítható meg?
- Távolítsuk el a hurkokat a topológiából
  - A kábelek kihúzása nélkül
- 802.1 (LAN) definiál egy algoritmust feszítőfa építéséhez és karbantartásához, mely mentén lehetséges a keretek továbbítása



Feszítőfa

- Egy gráf éleinek részhalmaza, melyre teljesül:
  - Lefed minden csomópontot
  - Nem tartalmaz köröket
- Továbbá a struktúra egy fa-gráf



A 802.1 feszítőfa algoritmusa

1. Az egyik bride-et megválasztjuk a fa gyökerének
2. Minden bridge megkeresi a legrövidebb utat a gyökérhez
3. Ezen utak unióját véve megkapjuk a feszítőfát

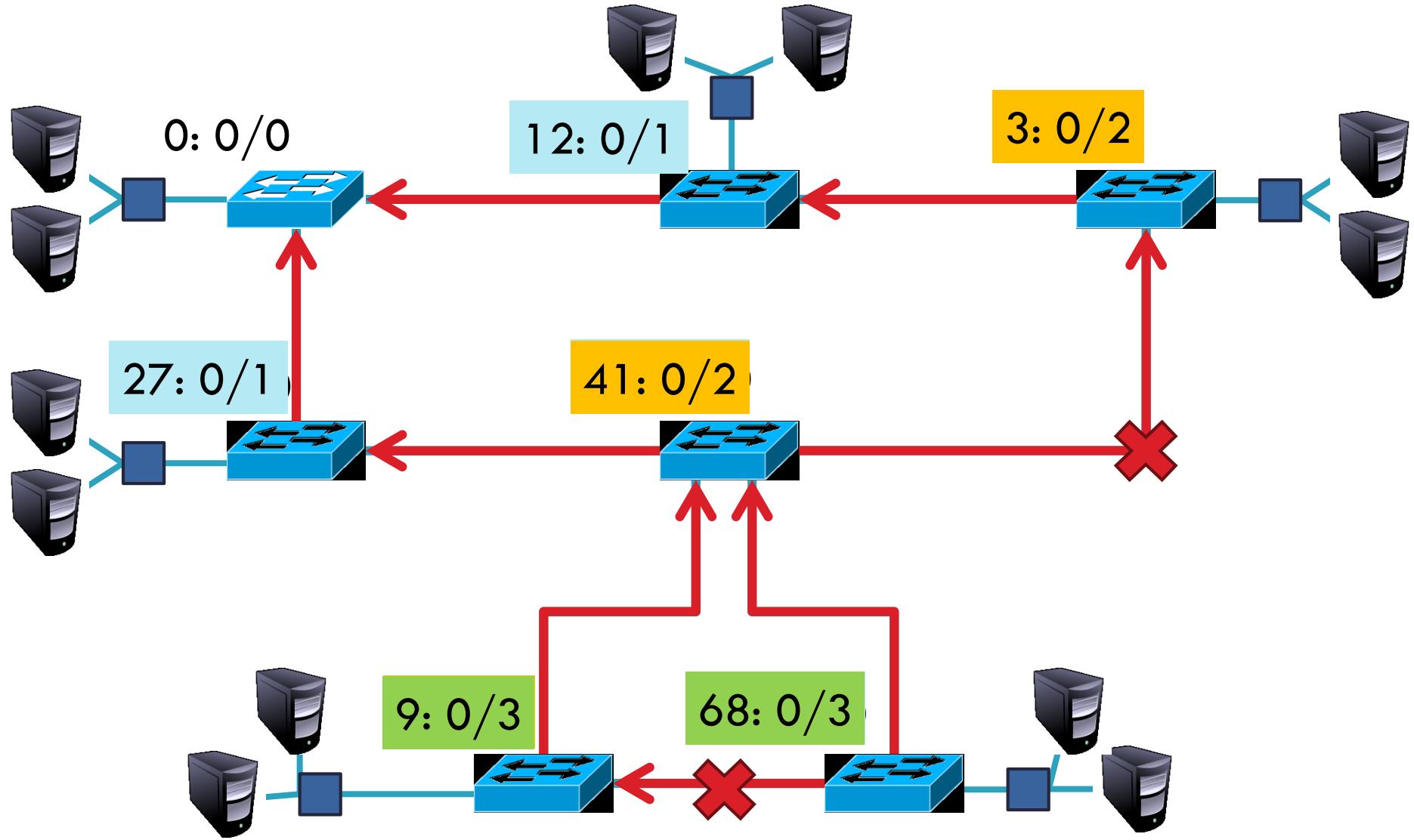
- A fa építése során a bridge-ek egymás között konfigurációs üzeneteket (Configuration Bridge Protocol Data Units [BPDUs]) cserélnek
  - A gyökér elem megválasztásához
  - A legrövidebb utak meghatározásához
  - A gyökérhez legközelebbi szomszéd (next hop) állomás és a hozzá tartozó port azonosításához
  - A feszítőfához tartozó portok kiválasztása

Gyökér meghatározása

- Kezdetben minden állomás feltételezi magáról, hogy gyökér
- Bridge-ek minden irányba szétküldik a BPDU üzeneteiket:

- A fogadott BPDU üzenet alapján, minden switch választ:
  - Egy új gyökér elemet (legkisebb ismert Gyökér ID alapján)
  - Egy új gyökér portot (melyik interfész megy a gyökér irányába)
  - Egy új kijelölt bridge-et (a következő állomás a gyökérhez vezető úton)

Feszítőfa építése



Bridge-ek vs. Switch-ek
Hidak vs. Kapcsolók

- A bridge-ek lehetővé teszik hogy növeljük a LAN-ok kapacitását
  - Csökkentik a sikeres átvitelhez szükséges elküldendő csomagok számát
  - Kezeli a hurkokat
- A switch-ek a bridge-ek speciális esetei
  - Minden port egyetlen egy hoszthoz kapcsolódik
    - Lehet egy kliens terminál
    - vagy akár egy másik switch
  - Full-duplex link-ek
  - Egyszerűsített hardver: nincs szükség CSMA/CD-re!
  - Különböző sebességű/rátájú portok is lehetségesek

Kapcsoljuk össze az Internetet

- Switch-ek képességei:
  - MAC cím alapú útvonalválasztás a hálózatban
  - Automatikusan megtanulja az utakat egy új állomáshoz
  - Feloldja a hurkokat
- Lehetne a teljes internet egy ily módon összekötött tartomány?

NEM

Korlátok

- Nem hatékony
  - Elárasztás ismeretlen állomások megtalálásához
- Gyenge teljesítmény
  - A feszítőfa nem foglalkozik a terhelés elosztással
  - Hot spots
- Nagyon gyenge skálázhatóság
  - Minden switch-nek az Internet összes MAC címét ismerni kellene a továbbító táblájában!
- Az IP fogja ezt a problémát megoldani…

Hálózati réteg

- Szolgáltatás
  - Csomagtovábbítás
  - Útvonalválasztás
  - Csomag fragmentálás kezelése
  - Csomag ütemezés
  - Puffer kezelés
- Interfész
  - Csomag küldése egy adott végpontnak
- Protokoll
  - Globálisan egyedi címeket definiálása
  - Routing táblák karbantartása
- Példák: Internet Protocol (IPv4), IPv6

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

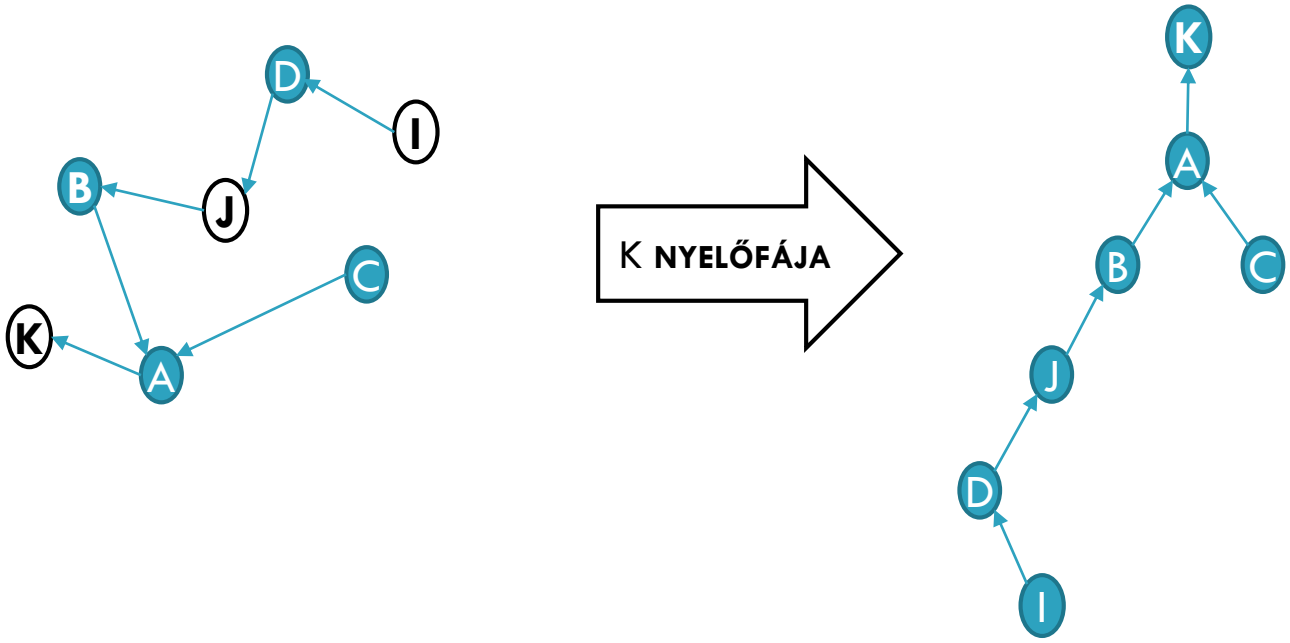
1. Honnan kapják az információt?
   - szomszédok, helyileg, minden router-től
2. Mikor változtatják az útvonalakat?
   - meghatározott másodpercenként, terhelés változásra, topológia változásra
3. Milyen mértékeket használnak az optimalizáláshoz?
   - távolság, ugrások (*hops*) száma, becsült késleltetés

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.

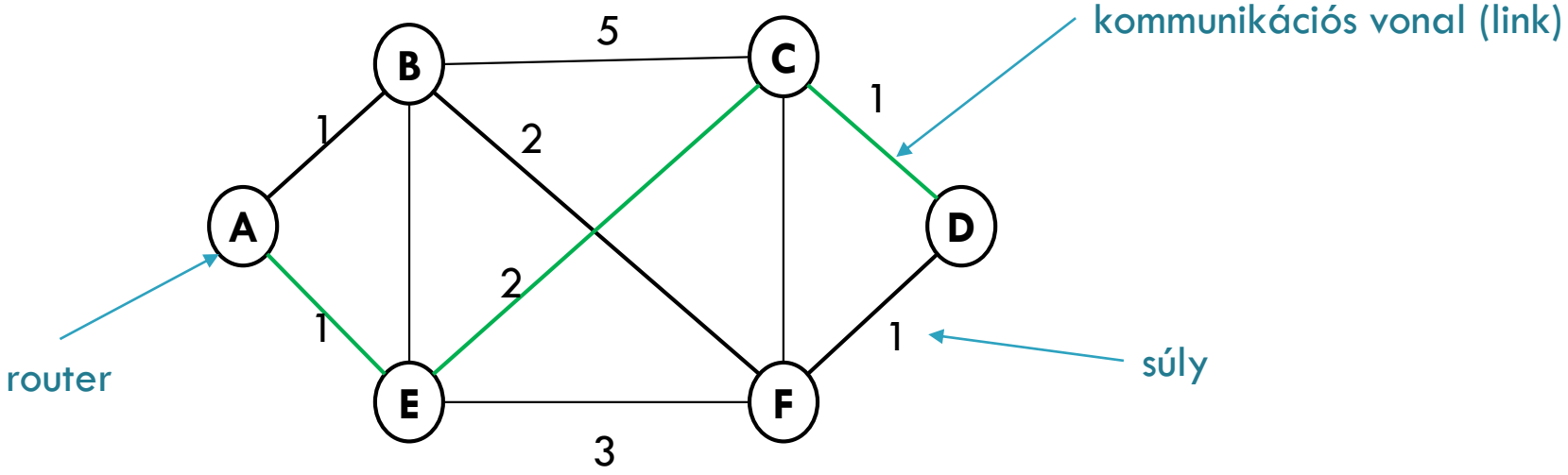


Legrövidebb út alapú forgalomirányítás

ALHÁLÓZAT REPREZENTÁCIÓJA

Az alhálózat tekinthető egy gráfnak, amelyben minden router egy csomópontnak és minden él egy kommunikációs vonalnak (link) 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

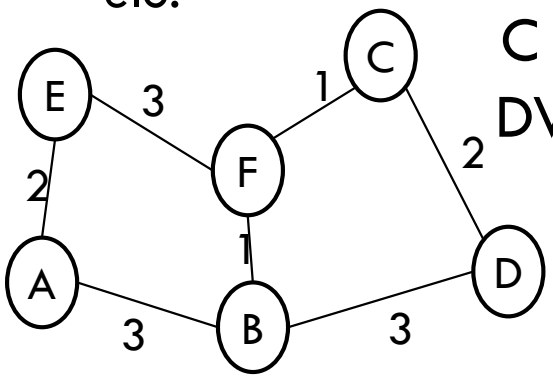
- Dinamikus algoritmusoknak 2 csoportja van:
  - távolságvektor alapú illetve (distance vector routing)
  - kapcsolatállapot alapú (link-state routing)

- **Távolságvektor alapú**: 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.

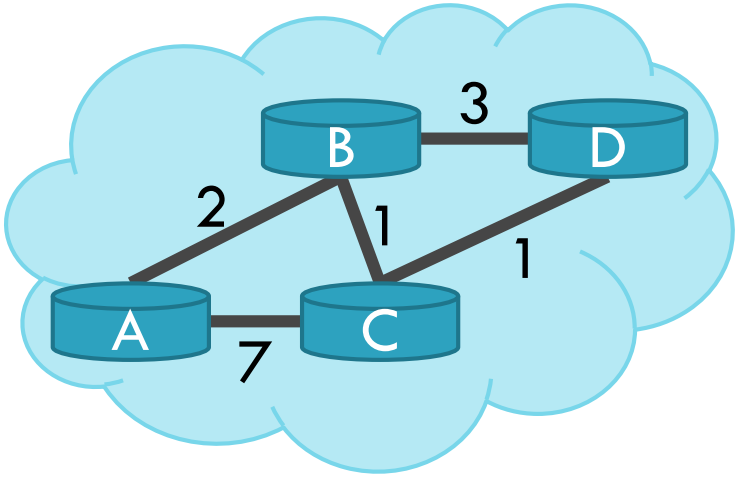


C állomás DV táblája

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) = ∞;
…
```

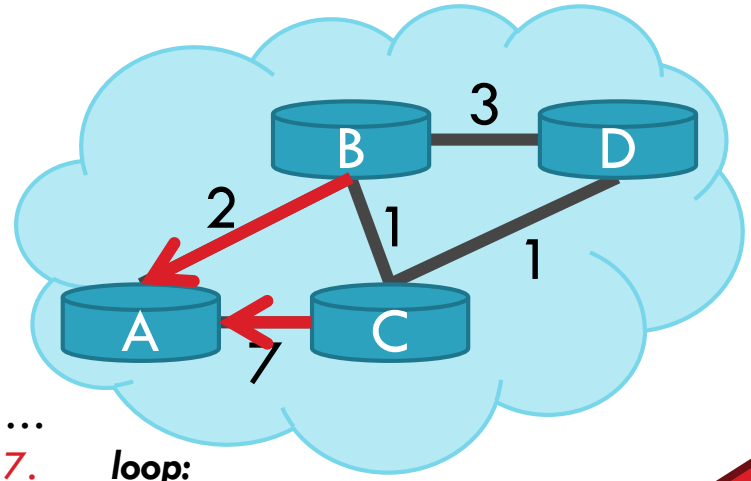
Node C

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



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

…

7. ***loop:***

…

12. **else if** (update D(*V*, *Y*) rece
13. **for all** desti
14. **if** (destina
15. D(*A*,*Y*) =
16. **else**
17. D(A, Y) =
mi
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**

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

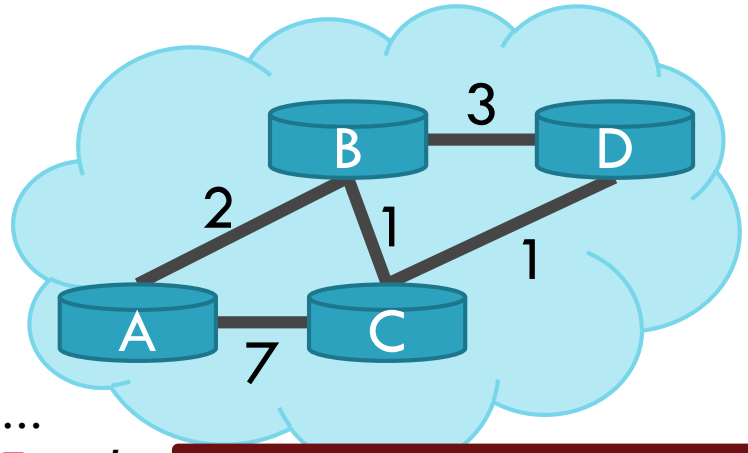
D(A,D) = min(D(A,D), D(A,B)+D(B,D))
= min(8, 3 + 3) = 5

Node D

Dest.	Cost	Next
B	4	B
B	3	B
C	1	C

Dest.	Cost	Next
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

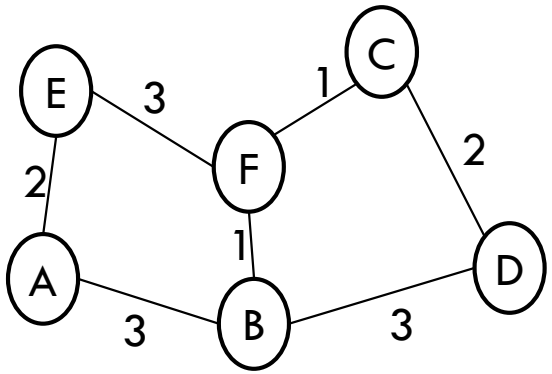
- Nothing changes, algorithm terminates
- Until something changes…

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

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

```
…
7.   loop
…
12.  else
13.    for
14.      i
15.
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

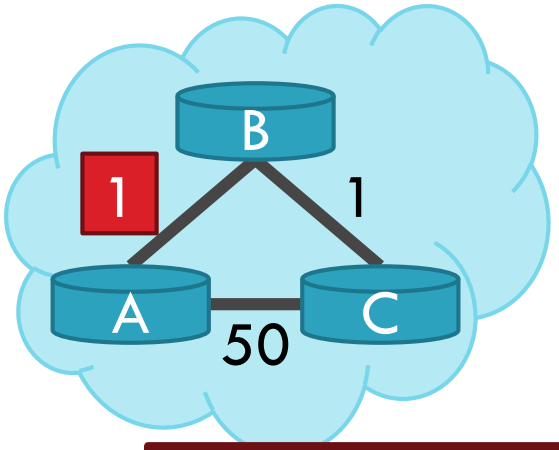
A	0
B	3
C	∞
D	6
E	2
F	4

…

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 Co
Algor

Good news travels fast

Algorithm
erminates

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

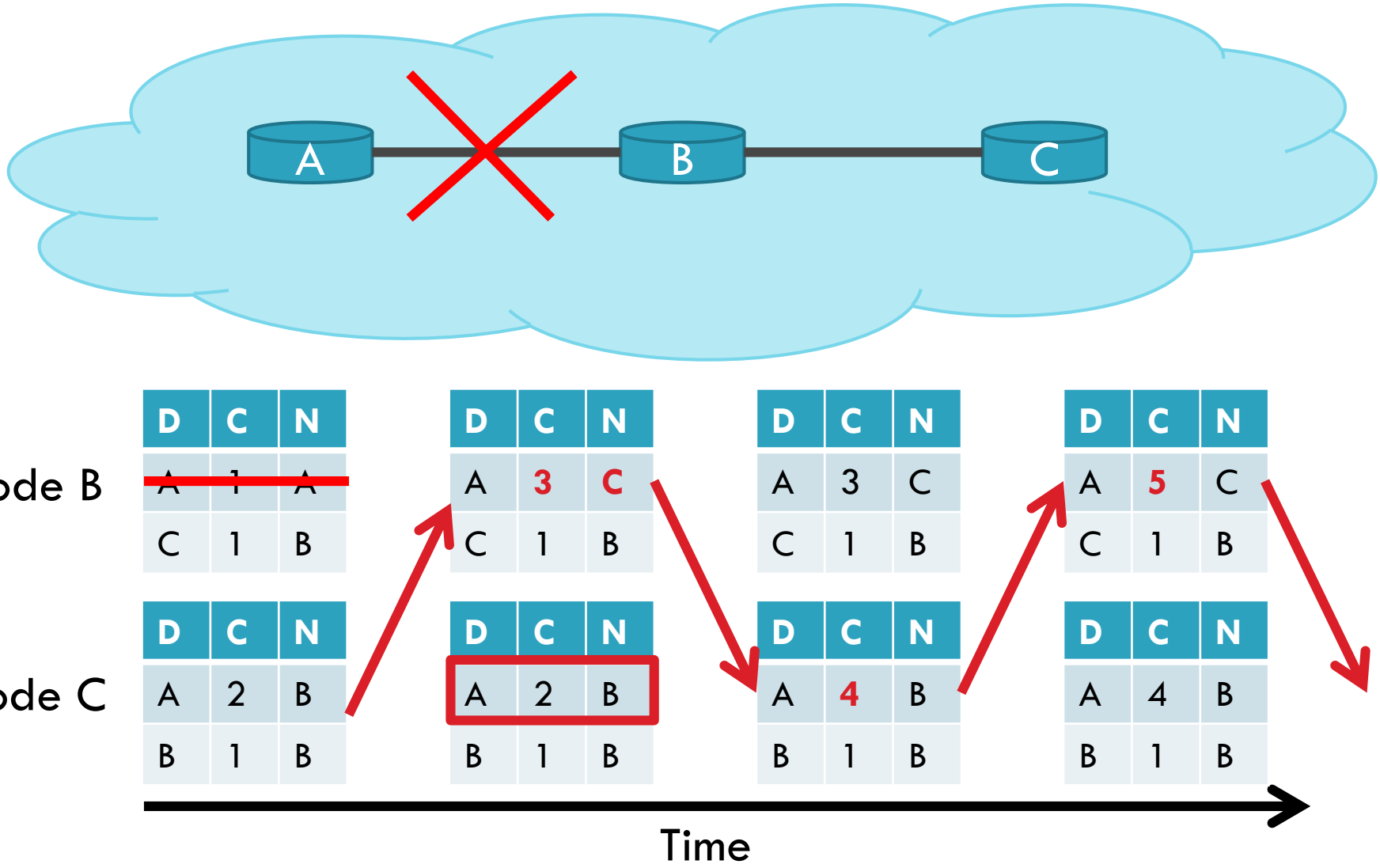
D	C	N
A	5	B
B	1	B

D	C	N
A	2	B
B	1	B

D	C	N
A	2	B
B	1	B

Time

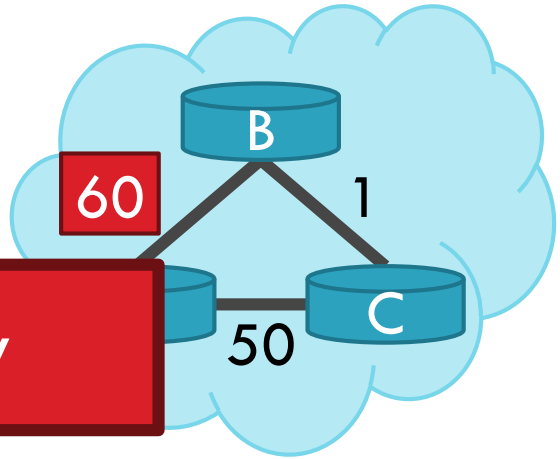
Távolság vektor protokoll – *Végtelenig számolás problémája (count to infinity)*



Példa - Count to Infinity Problem

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

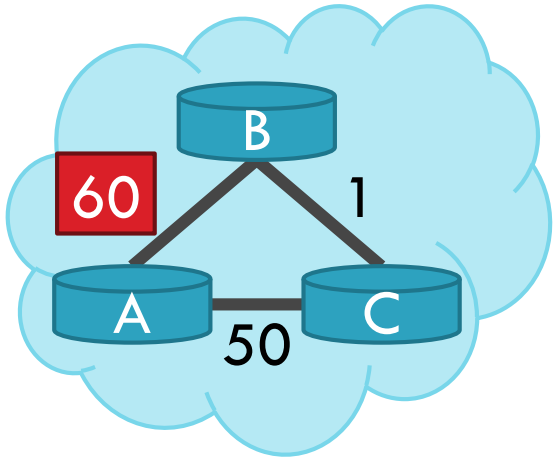
Elosztott Bellman-Ford algoritmus – *Végtelenig számolás problémája*

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*)

Split horizon with Poisoned Reverse

- Ha C B-n keresztül irányítja a forgalmat A állomáshoz
  - C állomás B-nek D(C, A) = ∞ távolságot küld
  - Azaz B állomás nem fog C-n keresztül irányítani az A-ba menő forgalmat



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 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

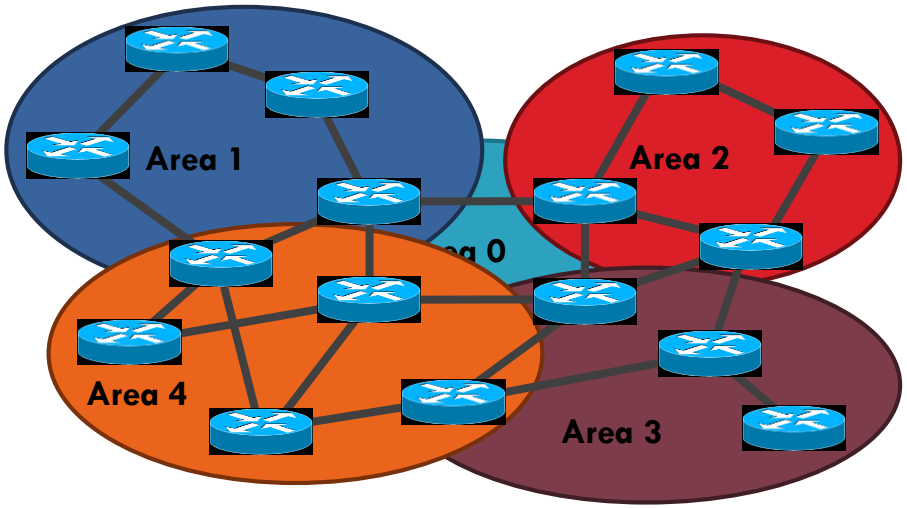
IS-IS

- 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

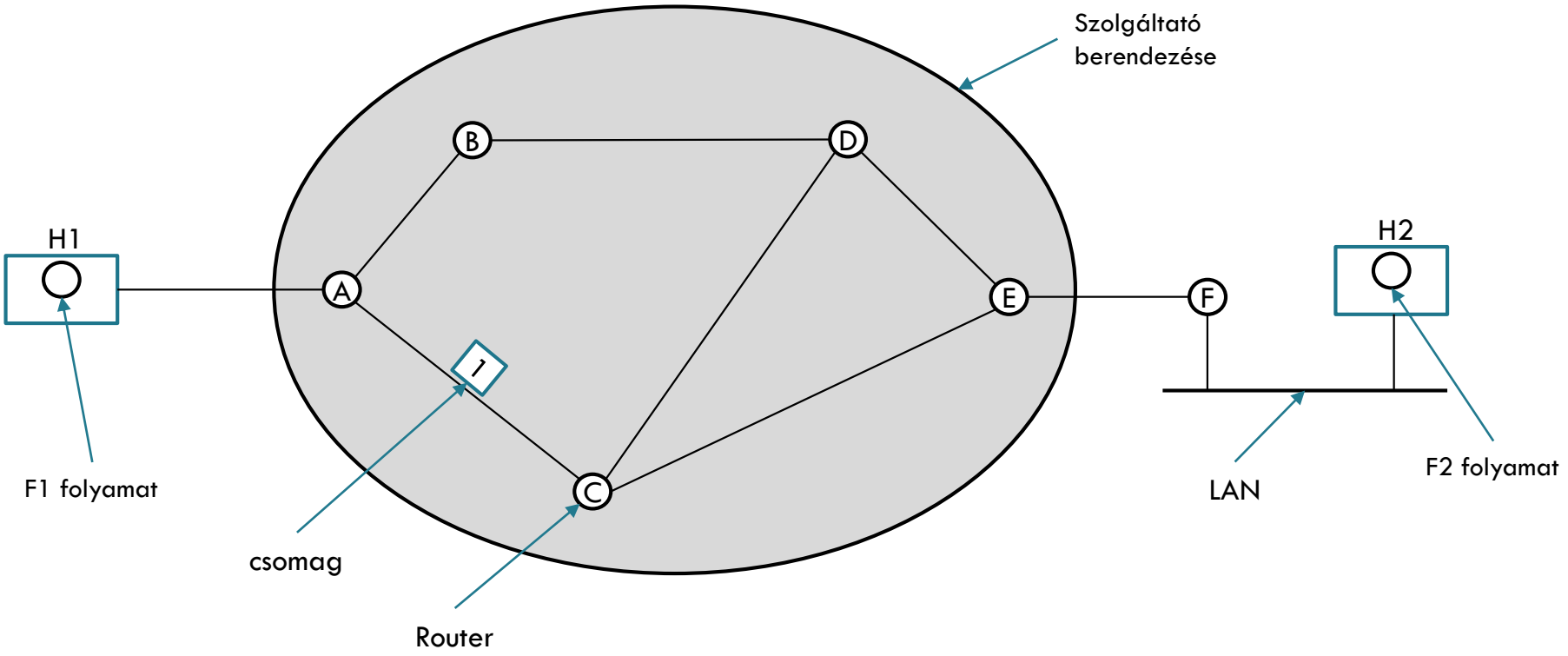
- Átfedő területek köré szerveződik
- Area 0 a hálózat magja



IS-IS

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

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

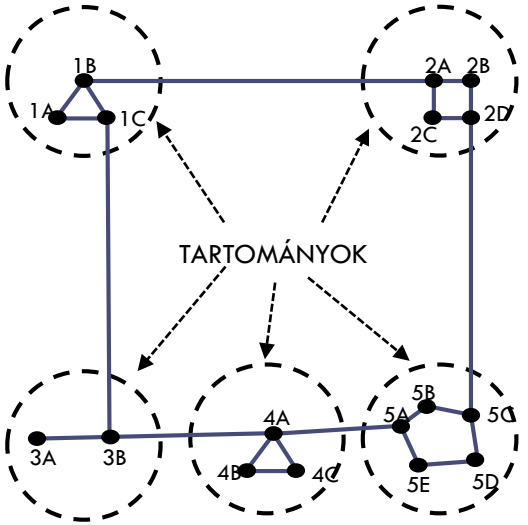
- Összeköttetés nélküli szolgálat (*Internet*)
  - datagram alhálózat
- Összeköttetés alapú szolgálat (*ATM*)
  - virtuális áramkör alhálózat

Hálózati réteg – forgalomirányítás

Hierarchikus forgalomirányítás

MOTIVÁCIÓ

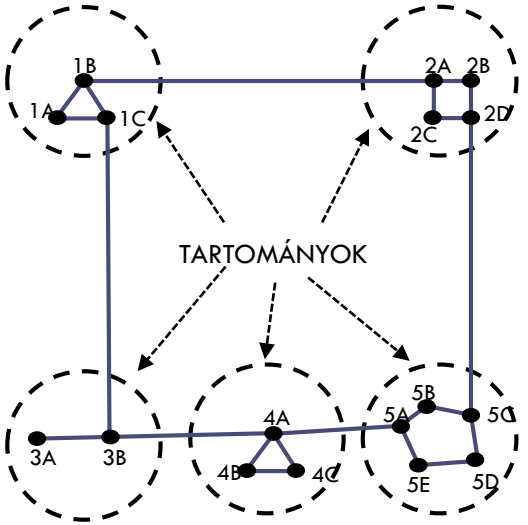
- A hálózat méretének növekedésével a router-ek forgalomirányító táblázatai is arányosan nőnek.
  - A memória, a CPU és a sávszélesség igény is megnövekszik a router-eknél.
- *Ötlet:* telefonhálózatokhoz hasonlóan hierarchikus forgalomirányítás alkalmazása.



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

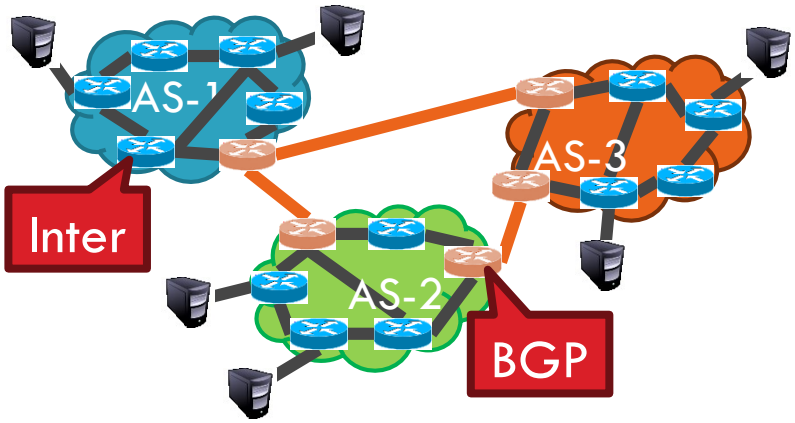
- **Többes-küldés** (vagy angolul *multicasting*) – egy csomag meghatározott csoporthoz történő egyidejű küldése.

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

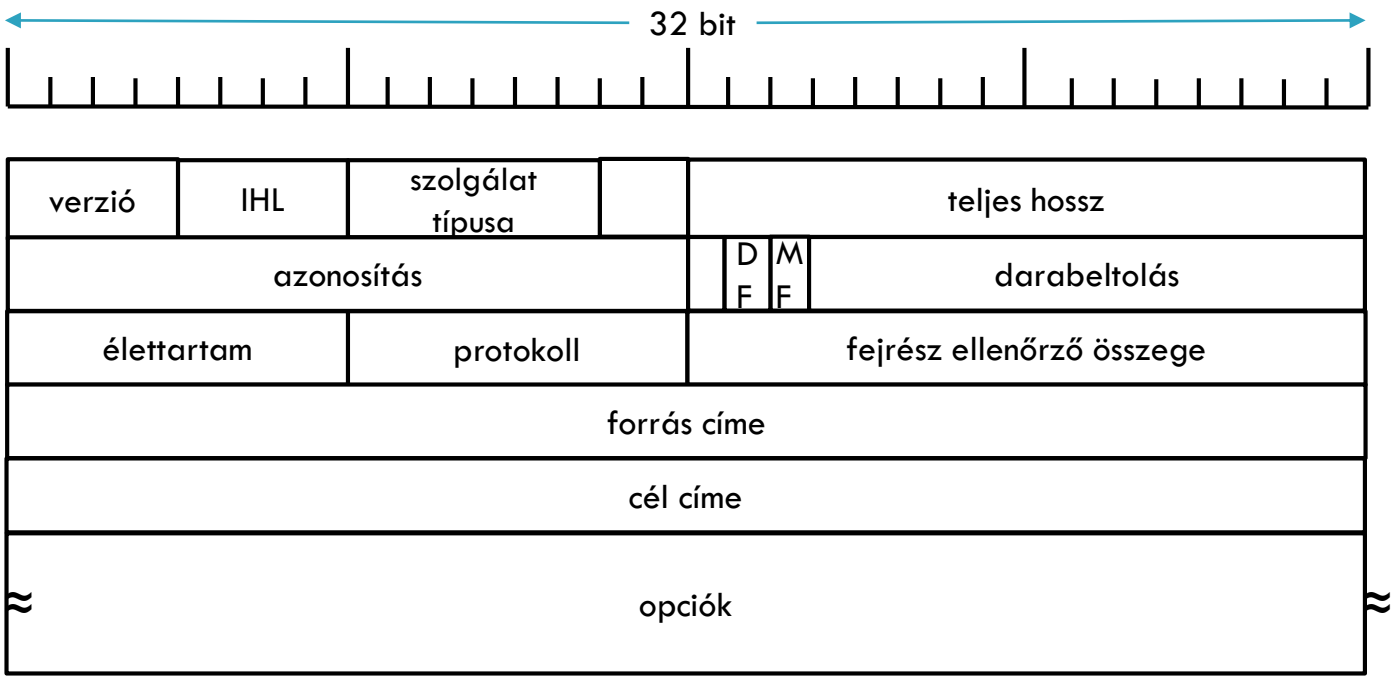
- A hálózati réteg szintjén az internet autonóm rendszerek összekapcsolt együttesének tekinthető.
  - Nincs igazi szerkezete, de számos főbb *gerinchálózata* létezik.
  - A gerinchálózatokhoz csatlakoznak a területi illetve regionális hálózatok.
  - A regionális és területi hálózatokhoz csatlakoznak 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 IPv4 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)

Címzés


What's your address ?
173.168.15.10
No,your local address
127.0.0.1
I mean your physical address !
28:05:FF:58:31:05

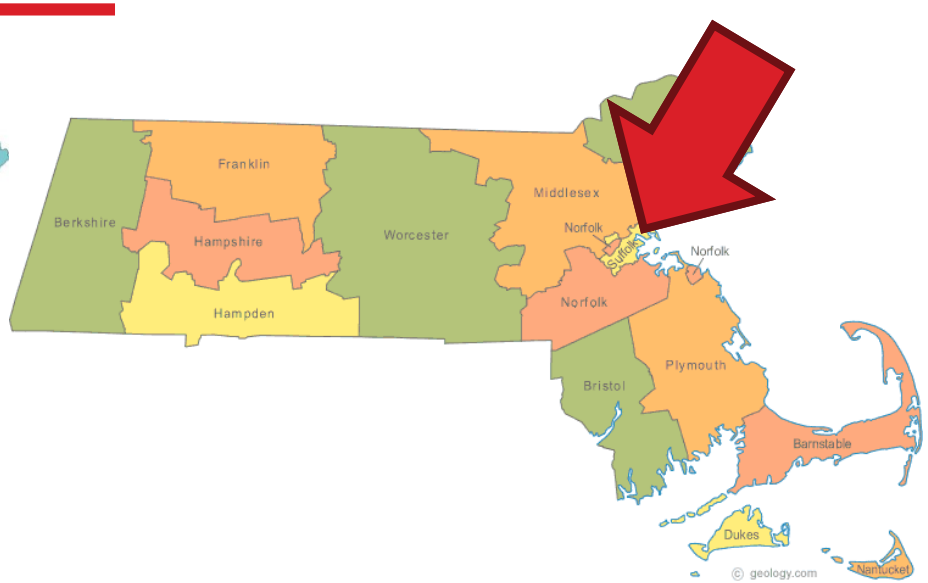
Lehetséges címzési struktúrák

- Sík - Flat
  - Pl. minden hosztot egy 48-bites MAC címmel azonosítunk
  - A routernek minden hoszthoz kell bejegyzés a táblájába
    - Túl nagy
    - Túl nehéz karbantartani (hosztok jönnek, mennek)
    - Túl lassú
- Hierarchikus
  - Címek szegmensekre bonthatók
  - Egy szegmens egy adott szintű konkrét területet fed le

Példa: Telefonszámok

1-617-373- 3278

Nagyon általános



Northeastern University

West Village H

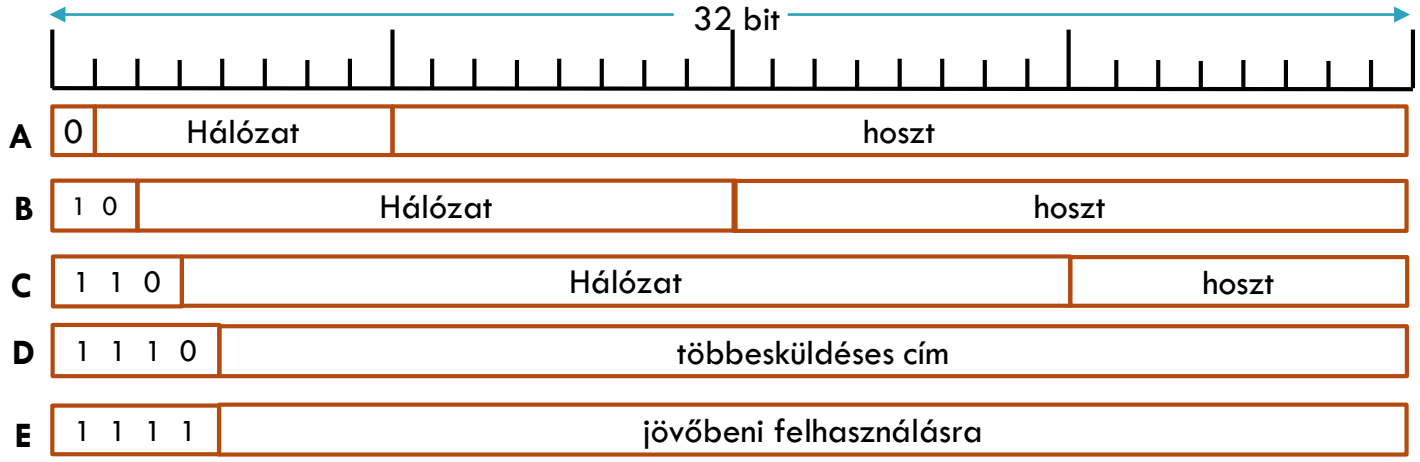
Room 254

Helyi frissítések

Nagy[illegible] konkrét

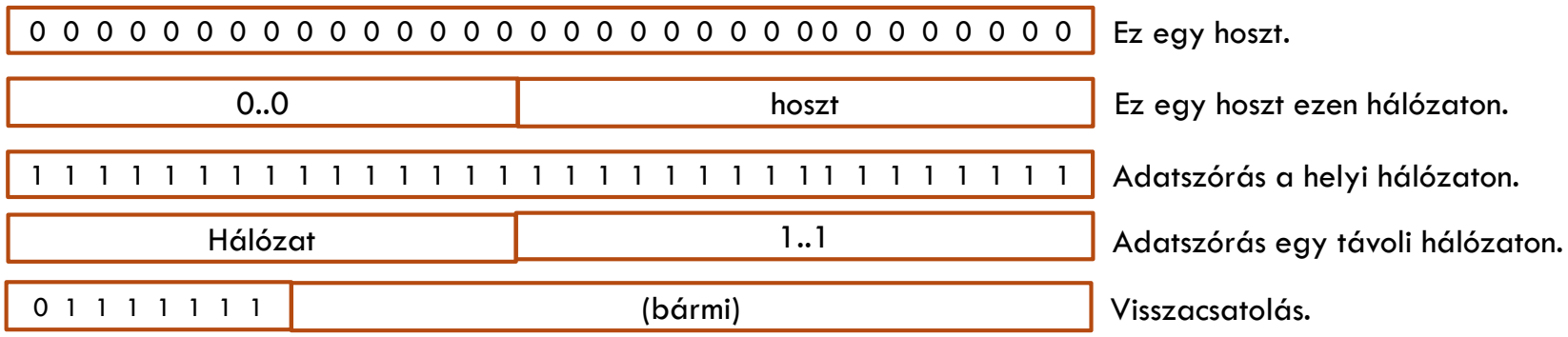
IP cím

- 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.

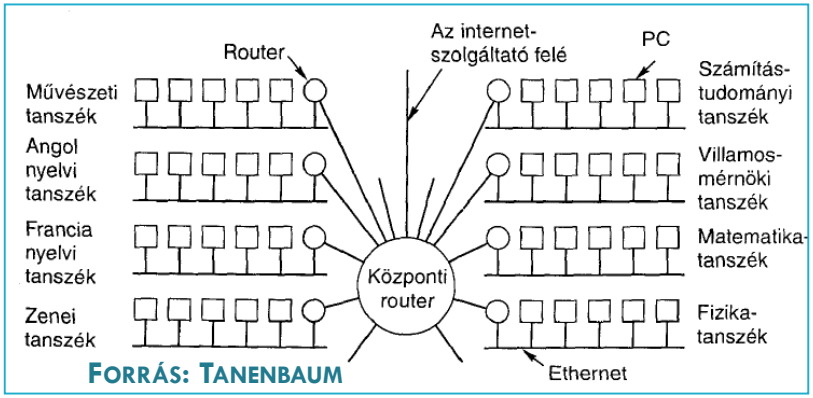


IP cím

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



IP cím – alhálózatok



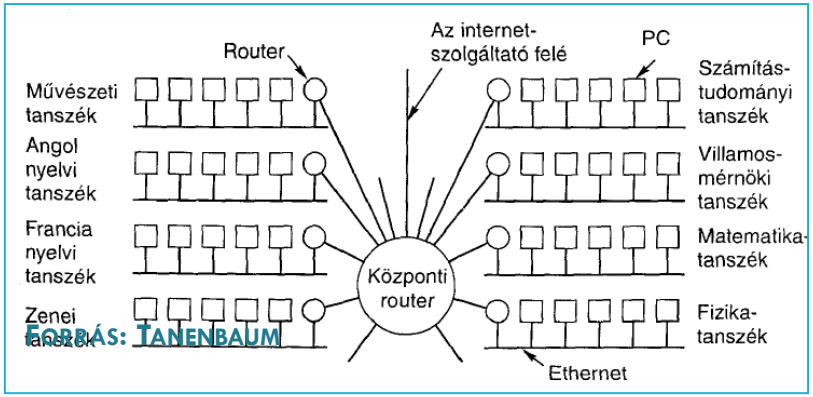
FORRÁS: TANENBAUM

- 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

/23-ES CÍM ESETÉN

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

Kimaszkolás eredménye

- 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 01100 000 00000000
00111001 00000110 01101 000 00000000
00111001 00000110 01110 000 00000000
00111001 00000110 01111 000 00000000

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

Köszönöm a figyelmet!