

Számítógépes Hálózatok

7. Előadás: Adatkapcsolati réteg Hálózati réteg

CSMA/CD - CSMA ütközés detektálással (CD = Collision Detection)

- ❑ Ütközés érzékelés esetén meg lehessen szakítani az adást. („Collision Detection”)
 - Minden állomás küldés közben megfigyeli a csatornát,
 - ha ütközést tapasztal, akkor megszakítja az adást, és véletlen ideig várakozik, majd újra elkezdi leadni a keretét.
- ❑ Mikor lehet egy állomás biztos abban, hogy megszerezte magának a csatornát?
 - Az ütközés detektálás minimális ideje az az idő, ami egy jelnek a két legtávolabbi állomás közötti átviteléhez szükséges.

CSMA/CD



- Egy állomás megszerezte a csatornát, ha minden más állomás érzékeli az átvitelét.
- Az **ütközés detektálás működéséhez** szükséges a keretek hosszára egy alsó korlátot adnunk
- Ethernet a CSMA/CD-t használja

CSMA/CD

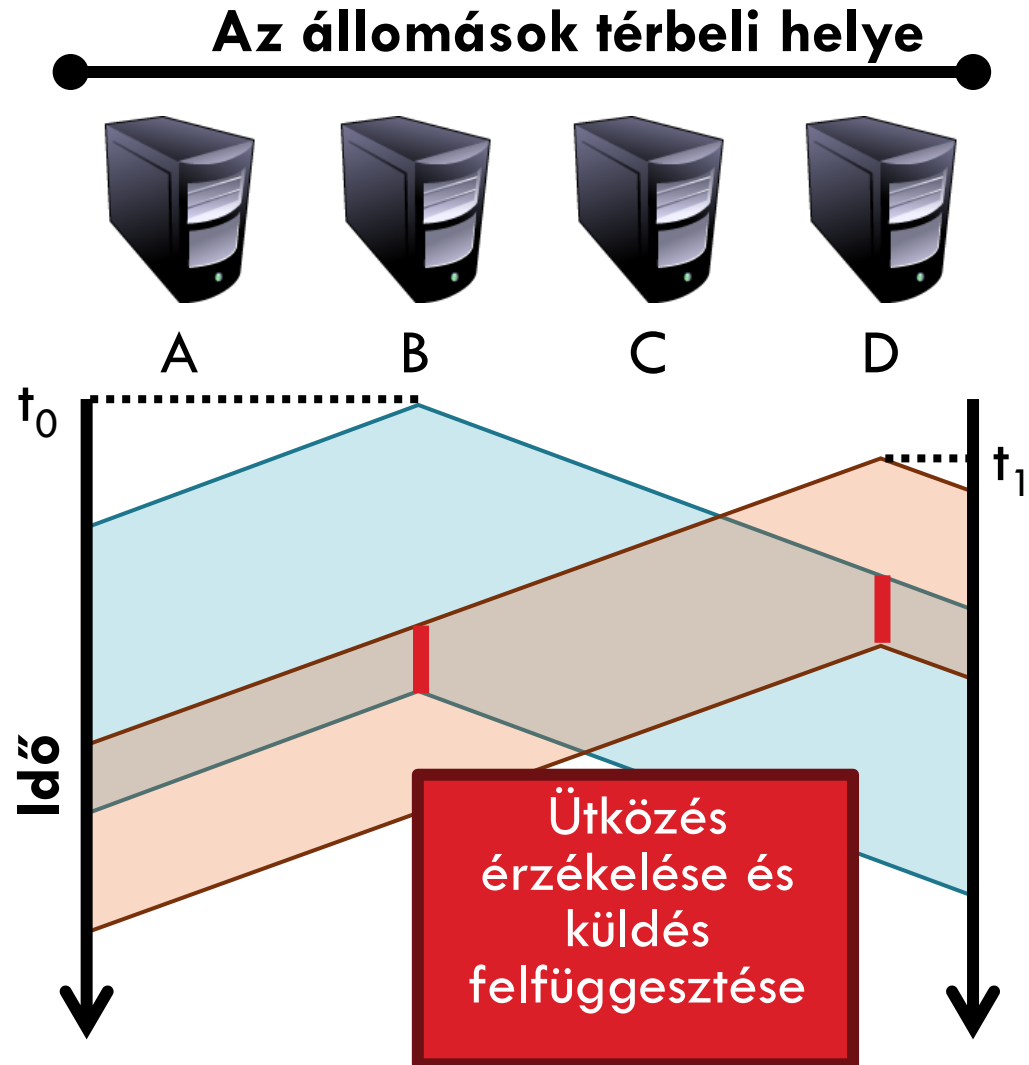
4

- ❑ Carrier sense multiple access with collision detection
- ❑ Alapvetés: a közeg lehetőséget ad a csatornába hallgatásra
- ❑ Algoritmus
 1. Használjuk valamely CSMA variánst
 2. A keret kiküldése után, figyeljük a közeget, hogy történik-e ütközés
 3. Ha nem volt ütközés, akkor a keretet leszállítottuk
 4. Ha ütközés történt, akkor azonnal megszakítjuk a küldést
 - Miért is folytatnánk hisz a keret már sérült...
 5. Alkalmazzuk az bináris exponenciális hátralék módszert az újraküldés során (binary exponential backoff)

CSMA/CD Ütközések

5

- Ütközések történhetnek
- Az ütközéseket gyorsan észleljük és felfüggesztjük az átvitelt
- Mi a szerepe a távolságnak, propagációs időnek és a keret méretének?



Binary Exponential Backoff –

Bináris exponenciális hátralék

6

- Ütközés érzékelésekor a küldő egy ún. „jam” jelet küld
 - ▣ Minden állomás tudomást szerezzen az ütközésről
- Binary exponential backoff működése:
 - ▣ Válasszunk egy $k \in [0, 2^n - 1]$ egyenletes eloszlás szerint, ahol $n =$ az ütközések száma
 - ▣ Várjunk k időegységet (keretidőt) az újraküldésig
 - ▣ n felső határa 10, 16 sikertelen próbálkozás után pedig eldobjuk a keretet
- A hátralék idő versengési résekre van osztva

Binary Exponential Backoff

7

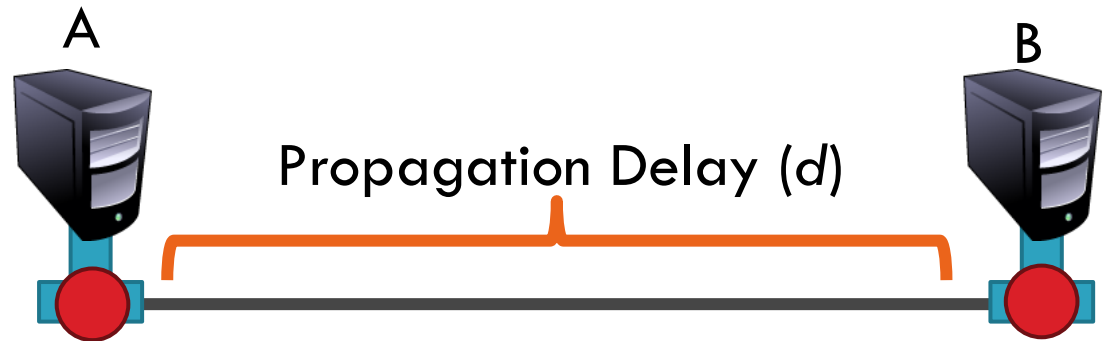
Tekintsünk két állomást, melyek üzenetei ütköztek

- Első ütközés után: válasszunk egyet a két időrés közül
 - ▣ A siker esélye az első ütközés után: 50%
 - ▣ Átlagos várakozási idő: 1,5 időrés
- Második ütközés után: válasszunk egyet a négy rés közül
 - ▣ Sikeres átvitel esélye ekkor: 75%
 - ▣ Átlagos várakozási idő: 2,5 rés
- Általában az m . ütközés után:
 - ▣ A sikeres átvitel esélye: $1 - 2^{-m}$
 - ▣ Average delay (in slots): $0,5 + 2^{(m-1)}$

Minimális keretméret

8

- Miért 64 bájt a minimális keretméret?
 - ▣ Az állomásoknak elég időre van szüksége az ütközés detektálásához
- Mi a kapcsolat a keretméret és a kábelhossz között?
 1. t időpont: Az A állomás megkezdte az átvitelt
 2. $t + d$ időpont: A B állomás is megkezdte az átvitelt
 3. $t + 2*d$ időpont: A érzékeli az ütközést



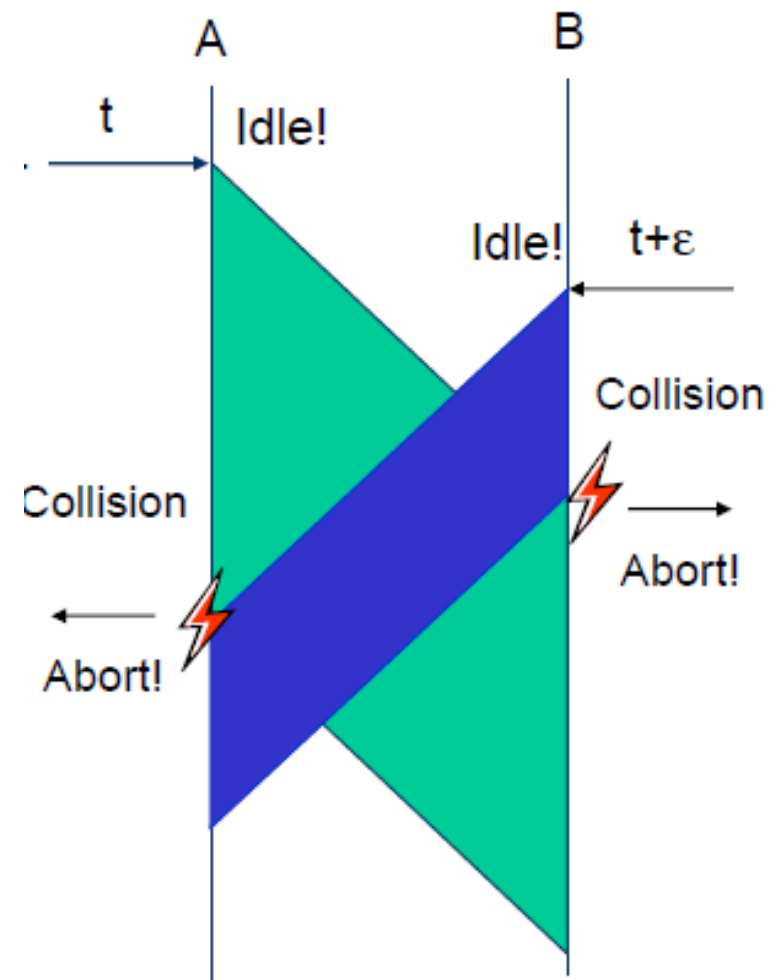
Alapötlet: Az A állomásnak $2*d$ ideig kell küldenie!

CSMA/CD

- CSMA/CD három állapota: versengés, átvitel és szabad.
- Ahhoz, hogy minden ütközést észleljünk szükséges:

$$T_f \geq 2T_{pg}$$

- ahol T_f egy keret elküldéséhez szükséges idő
- és T_{pg} a propagációs késés A és B állomások között



Minimális keretméret

10

- Az A küldésének $2 \cdot d$ ideig kell tartania

- $\text{Min_keret} = \text{ráta (b/s)} \cdot 2 \cdot d \text{ (s)}$

- ...

- 10 Mbps Ethernet

- Pr
- A keretméret és a kábelhossz változik (m/s)

- Aza
- a gyorsabb szabványokkal...

- $\text{Min_keret} = \text{ráta} \cdot 2 \cdot \text{távolság (m)} / \text{fényseb. (m/s)}$

- Azaz a kábel össza

- $\text{Távolság} = \text{min_keret} \cdot \text{fénysebesség} / (2 \cdot \text{ráta})$

$$(64B \cdot 8) \cdot (2 \cdot 10^8 \text{mps}) / (2 \cdot 10^7 \text{bps}) = 5120 \text{ méter}$$

Minimális keretméret

11

- Az A küldésének $2 \cdot d$ ideig kell tartania
 - ▣ $\text{Min_keret} = \text{ráta (b/s)} * 2 * d \text{ (s)}$
 - ... de mi az a d ? propagációs késés, melyet a fénysebesség ismeretében ki tudunk számolni
 - $\text{Propagációs késés (d)} = \text{távolság (m)} / \text{fénysebesség (m/s)}$
 - ▣ Azaz:
 - ▣ $\text{Min_keret} = \text{ráta (b/s)} * 2 * \text{távolság (m)} / \text{fényseb. (m/s)}$
- Azaz a kábel összhossza
 - ▣ $\text{Távolság} = \text{min_keret} * \text{fénysebesség} / (2 * \text{ráta})$

$$(64\text{B} * 8) * (2 * 10^8 \text{mps}) / (2 * 10^7 \text{bps}) = 5120 \text{ méter}$$

Kábelhossz példa

12

$$\text{min_keret} * \text{fénysebesség} / (2 * \text{ráta}) = \text{max_kábelhossz}$$
$$(64\text{B} * 8) * (2 * 10^8 \text{mps}) / (2 * 10 \text{Mbps}) = 5120 \text{ méter}$$

- Mi a maximális kábelhossz, ha a minimális keretméret 1024 bájtra változik?
 - ▣ 81,9 kilométer
- Mi a maximális kábelhossz, ha a ráta 1 Gbps-ra változik?
 - ▣ 51 méter
- Mi történik, ha mindkettő változik egyszerre?
 - ▣ 819 méter

Maximális keretméret

13

- ❑ Maximum Transmission Unit (MTU): 1500 bájt
- ❑ Pro:
 - ▣ Hosszú csomagokban levő biz hibák jelentős javítási költséget okozhatnak (pl. túl sok adatot kell újraküldeni)
- ❑ Kontra:
 - ▣ Több bájtot vesztegetünk el a fejlécekben
 - ▣ Összességében nagyobb csomag feldolgozási idő
- ❑ Adatközpontokban Jumbo keretek
 - ▣ 9000 bájtos keretek

Ütközésmentes protokollok

14

MOTIVÁCIÓ

- az ütközések hátrányosan hatnak a rendszer teljesítményére
 - ▣ hosszú kábel, rövid keret
- a CSMA/CD nem mindenhol alkalmazható

FELTÉTELEZÉSEK

- N állomás van.
- Az állomások 0-ától N-ig egyértelműen sorszámozva vannak.
- Réselt időmodellt feltételezünk.

Alapvető bittérkép protokoll

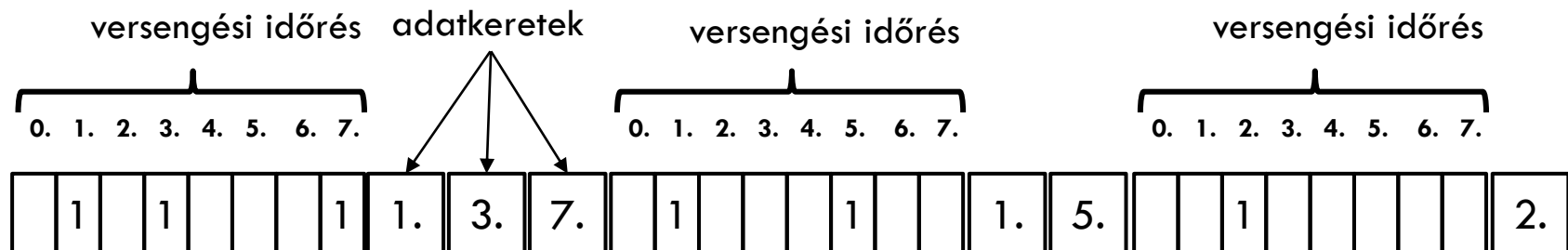
- Egy helyfoglalásos megoldás

15

- alapvető bittérkép eljárás

MŰKÖDÉS

- Az ütköztetési periódus N időrés
- Ha az i -edik állomás küldeni szeretne, akkor a i -edik versengési időrésben egy 1-es bit elküldésével jelezheti. (adatszórás)
- A versengési időszak végére minden állomás ismeri a küldőket. A küldés a sorszámok szerinti sorrendben történik meg.



Bináris visszaszámlálás protokoll 1 / 2

16

- alapvető bittérkép eljárás hátrány, hogy az állomások számának növekedésével a versengési periódus hossza is nő

MŰKÖDÉS

- Minden állomás azonos hosszú bináris azonosítóval rendelkezik.
- A forgalmazni kívánó állomás elkezd a bináris címét bitenként elküldeni a legnagyobb helyi értékű bittel kezdve. Az azonos pozíciójú bitek logikai VAGY kapcsolatba lépnek ütközés esetén. Ha az állomás nullát küld, de egyet hall vissza, akkor feladja a küldési szándékát, mert van nála nagyobb azonosítóval rendelkező küldő.

A HOSZT (0011)	0	–	–	–
B HOSZT (0110)	0	–	–	–
	1	0	1	0
C HOSZT (1010)	1	0	1	1
D HOSZT (1011)	1	0	1	1

D kerete

Bináris visszaszámlálás protokoll 2/2

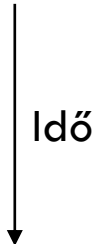
17

- **Következmény:** a magasabb címmel rendelkező állomásoknak a prioritásuk is magasabb az alacsonyabb című állomásokénál

MOK ÉS WARD MÓDOSÍTÁSA

- Virtuális állomás címek használata.
- Minden sikeres átvitel után ciklikusan permutáljuk az állomások címét.

	A	B	C	D	E	F	G	H
Kezdeti állapot	100	010	111	101	001	000	011	110



Korlátozott versenyes protokollok

18

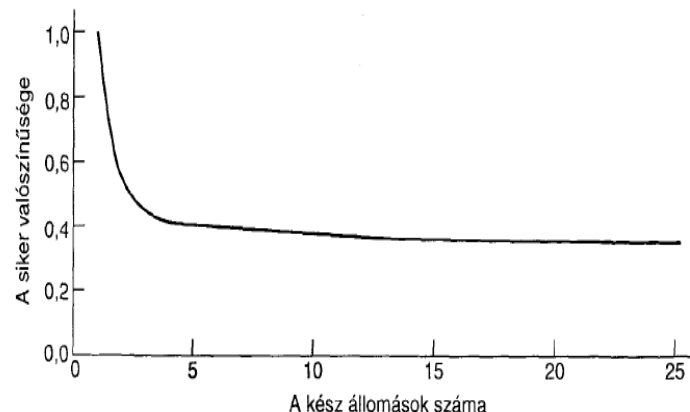
- **Cél:** Ötvözni a versenyhelyzetes és ütközésmentes protokollok jó tulajdonságait.
- **korlátozott versenyes protokoll** – Olyan protokoll, amely kis terhelés esetén versenyhelyzetes technikát használ a kis késleltetés érdekében, illetve nagy terhelés mellett ütközésmentes technikát alkalmaz a csatorna jó kihasználása érdekében.

SZIMMETRIKUS PROTOKOLLOK

- Adott részben k állomás verseng, minden állomás p valószínűséggel adhat. A csatorna megszerzésének valószínűsége: $kp(1 - p)^{k-1}$.

$$P(\text{siker optimális } p \text{ mellett}) = \left(\frac{k-1}{k}\right)^{k-1}$$

- Azaz a csatorna megszerzésének esélyeit a versenyhelyzetek számának csökkentésével érhetjük el.

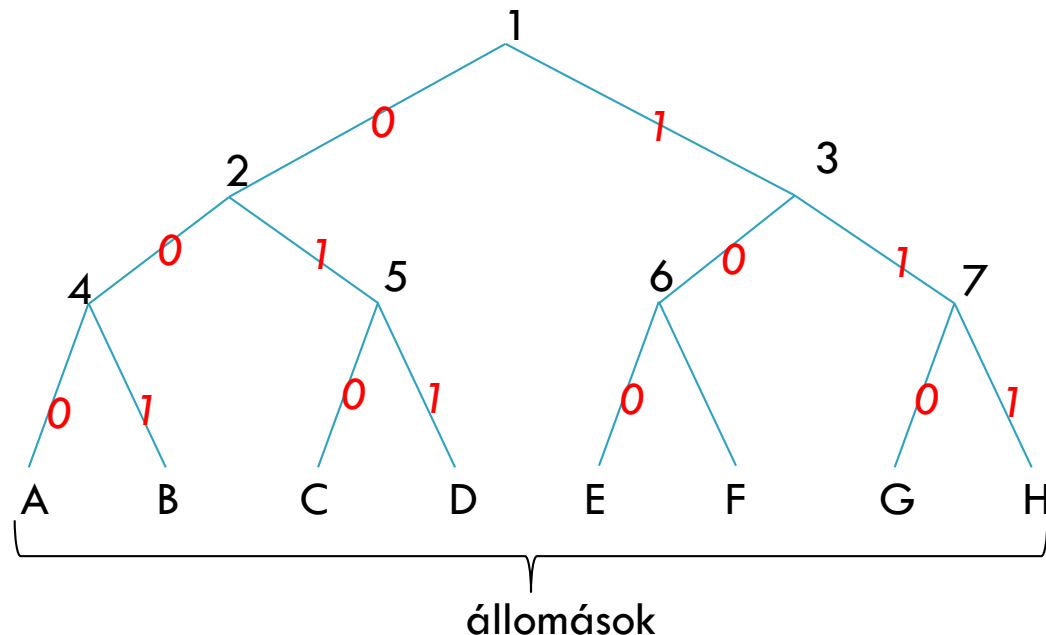


Adaptív fabejárési protokoll 1 / 2

19

Történeti háttér

- 1943 – Dorfman a katonák szifilisz fertőzöttségét vizsgálta.
- 1979 – Capetanakis bináris fa reprezentáció az algoritmus számítógépes változatával.



Adaptív fabejárási protokoll 2/2

20

Működés

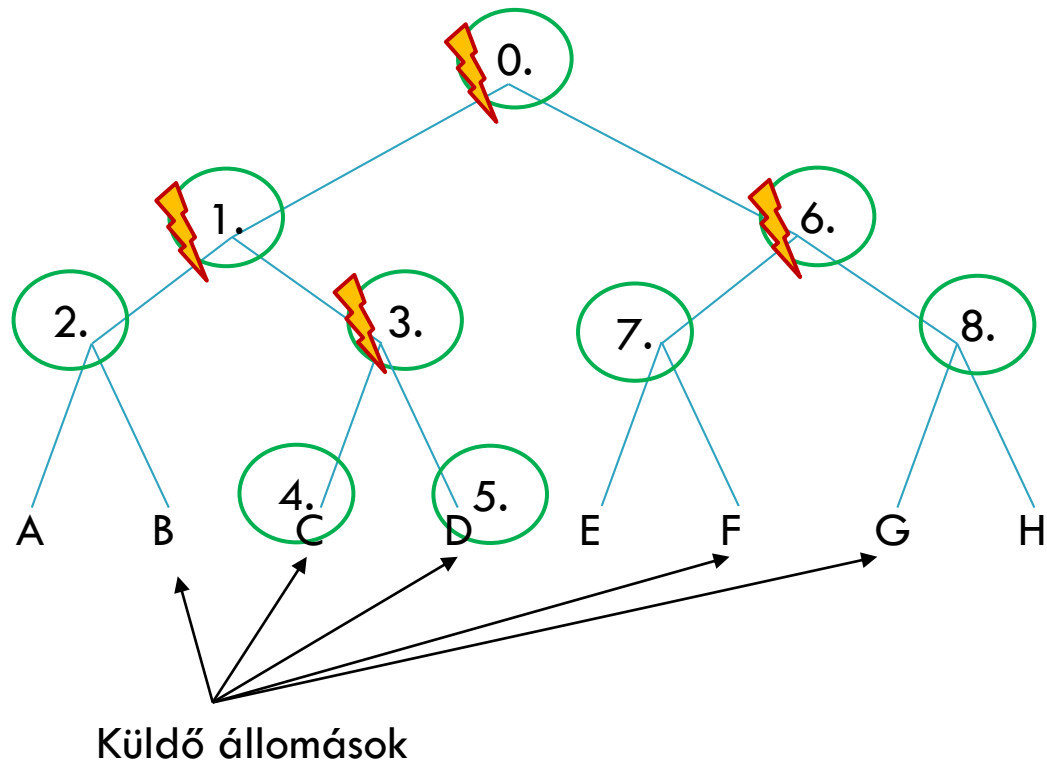
- 0-adik időrésben mindenki küldhet.
 - ▣ Ha ütközés történik, akkor megkezdődik a fa *mélységi bejárása*.
- A rések a fa egyes csomópontjaihoz vannak rendelve.
- Ütközéskor rekurzívan az adott csomópont bal illetve jobb gyerekcsomópontjánál folytatódik a keresés.
- Ha egy bitrés kihasználatlan marad, vagy pontosan egy állomás küld, akkor a szóban forgó csomópont keresése befejeződik.

Következmény

- Minél nagyobb a terhelés, annál mélyebben érdemes kezdeni a keresést.

Adaptív fabejárás példa

21



Az adatkapcsolati réteg „legtetején”...

22

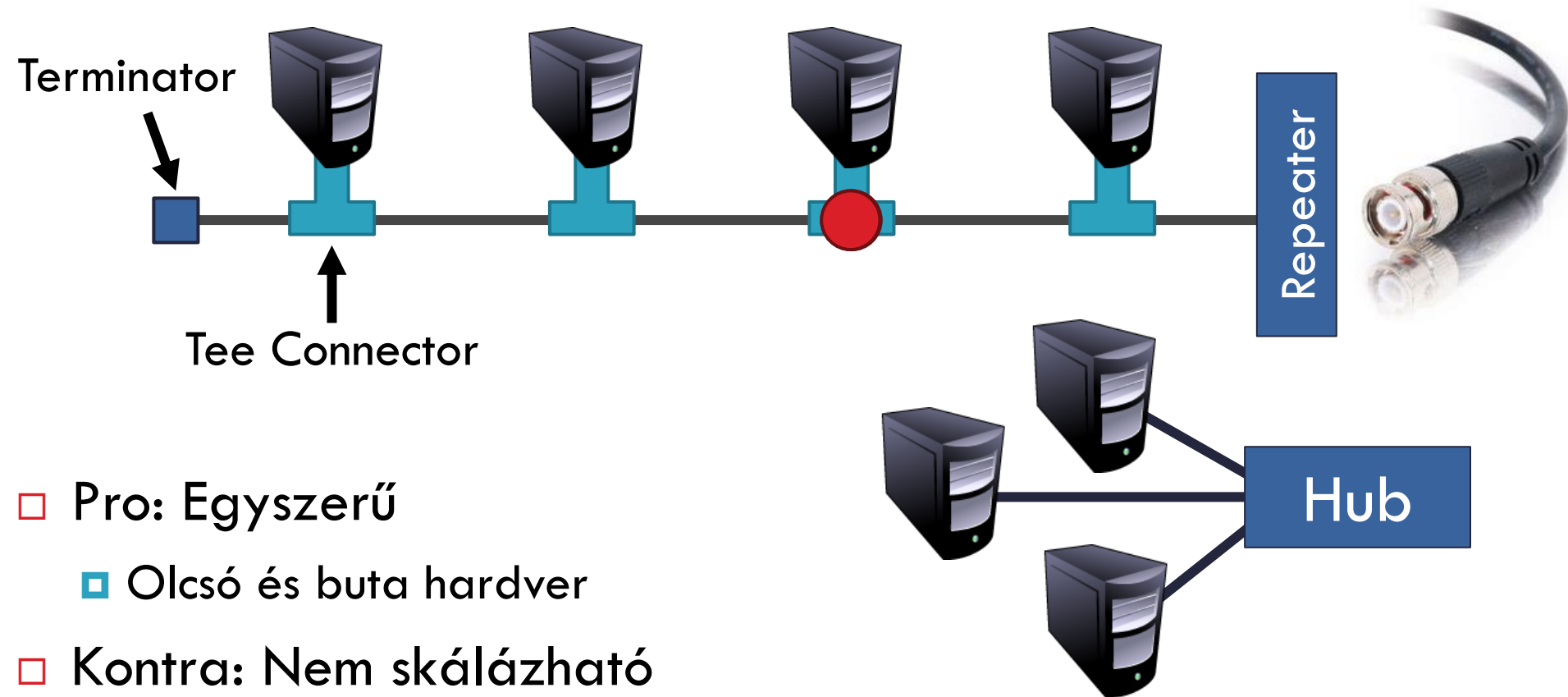


- Bridging, avagy hidak
 - ▣ Hogyan kapcsoljunk össze LAN-okat?
- Funkciók:
 - ▣ Keretek forgalomirányítása a LAN-ok között
- Kihívások:
 - ▣ Plug-and-play, önmagát konfiguráló
 - ▣ Esetleges hurkok feloldása

Visszatekintés

23

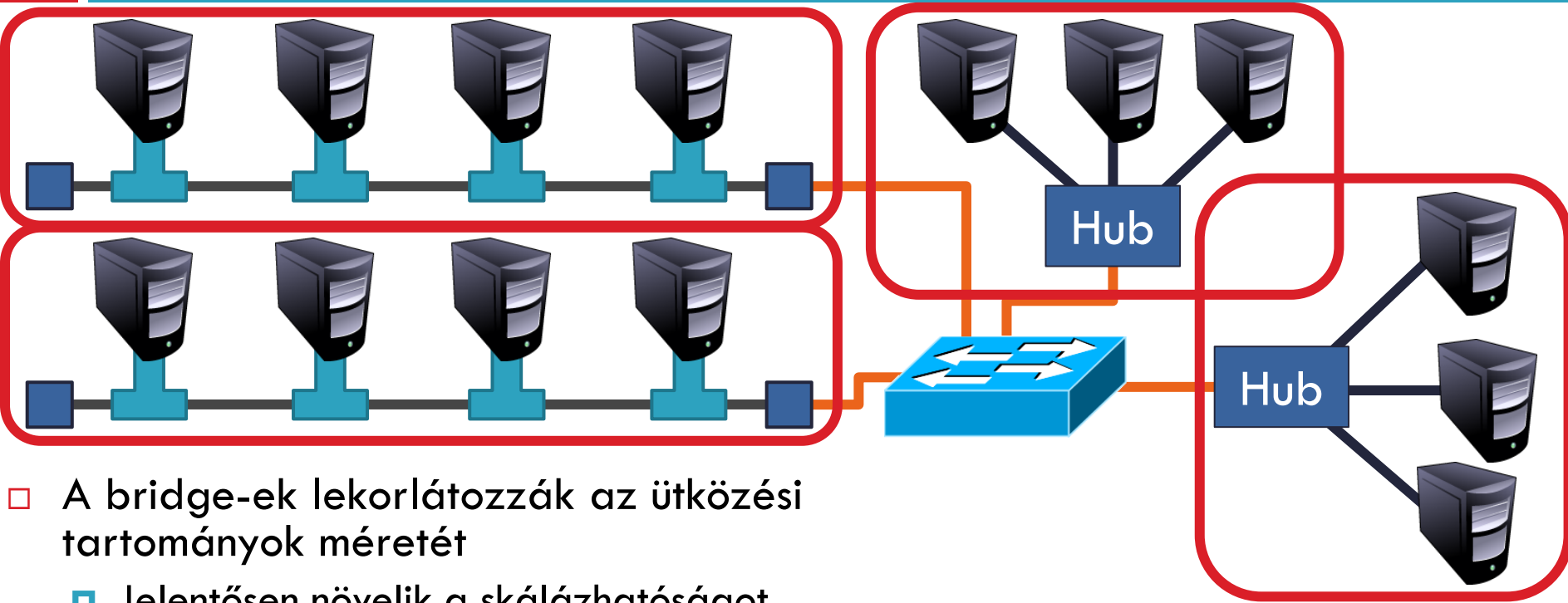
- Az Ethernet eredetileg adatszóró technológia volt



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

LAN-ok összekapcsolása

24

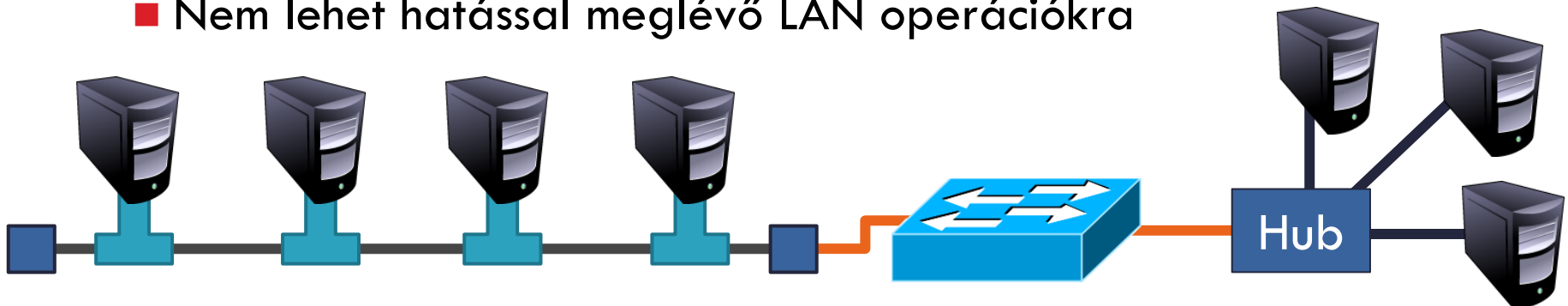


- A bridge-ek lekorlátozzák az ütközési tartományok méretét
 - ▣ Jelentősen növelik a skálázhatóságot
 - ▣ Kérdés: lehetne-e az egész Internet egy bridge-ekkel összekötött tartomány?
- Hátrány: a bridge-ek sokkal komplexebb eszközök a hub-oknál
 - ▣ Fizikai réteg VS Adatkapcsolati réteg
 - ▣ Memória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

Bridge-ek (magyarul: hidak)

25

- ❑ 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)

26

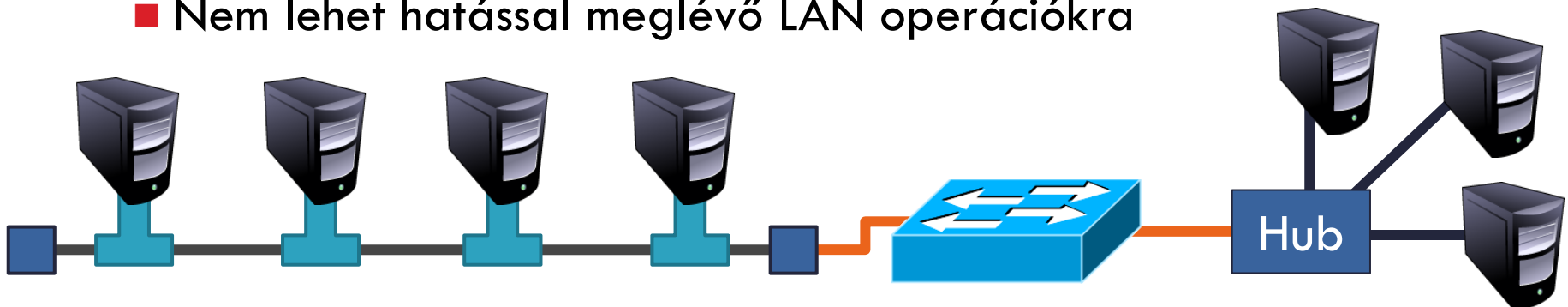
- Az Ethernet switch eredeti formája

□

□

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

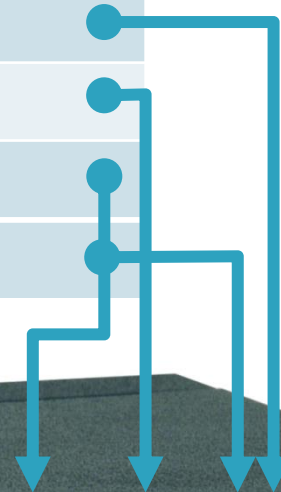


Keret Továbbító Táblák

27

- 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



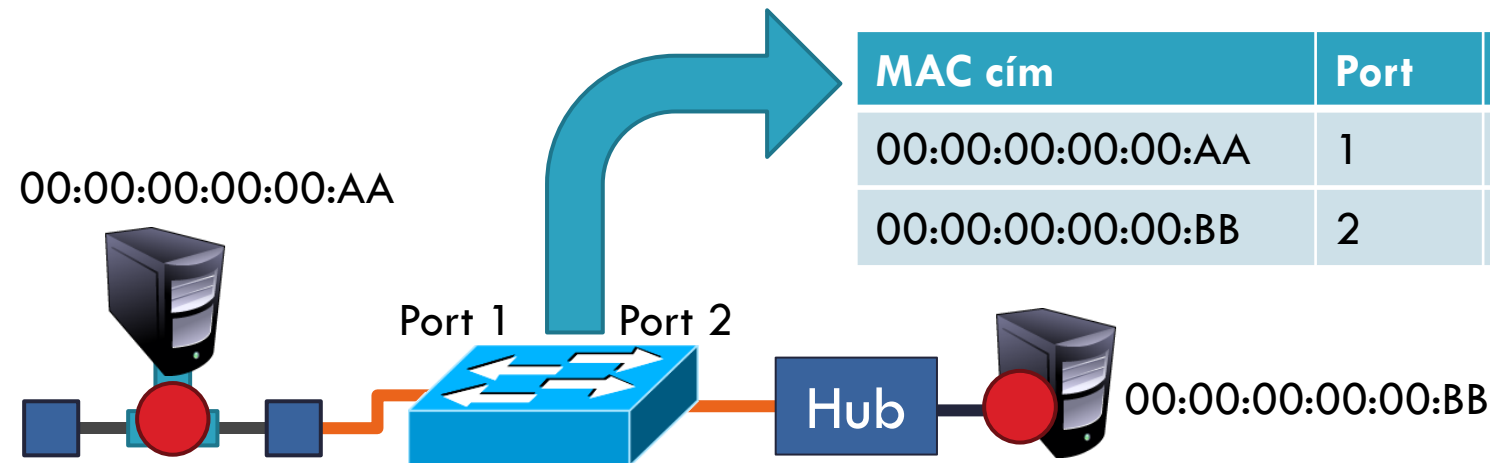
Címek tanulása

28

- ❑ 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 címeket** a különböző portokhoz tartozó kereteknek --- képezzünk ebből egy táblázatot

Töröljük a régi bejegyzéseket

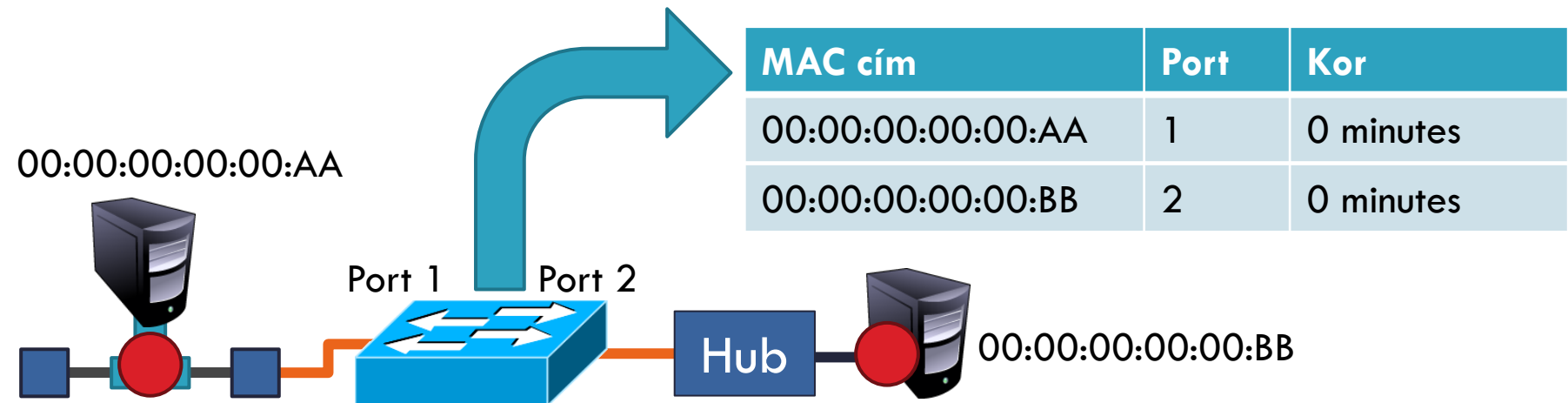
MAC cím	Port	Kor
00:00:00:00:00:AA	1	0 minutes
00:00:00:00:00:BB	2	0 minutes



Címek tanulása

29

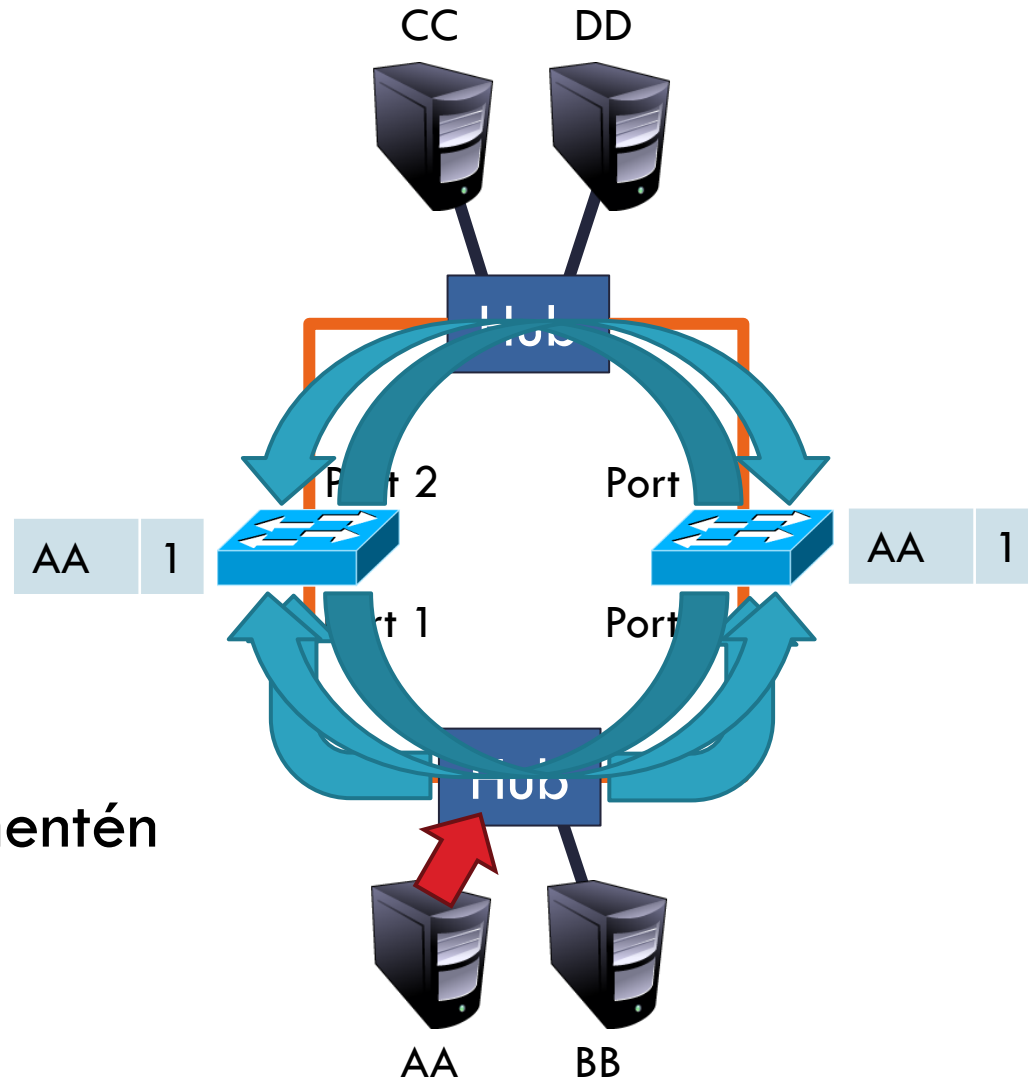
- ❑ Kézi beállítás is lehetséges, de...
 - ▣ Időigényes
 - ▣ Potenciális hiba forrás
 - ▣ Nem alkalmazkodik a változásokhoz (új hosztok léphetnek be és régiek hagyhatják el a hálózatot)
- ❑ Ehelyett: tanuljuk meg a címeket
 - ▣ Tekintsük a **forrás címeket** a különböző portokon beérkező kereteknek ---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



Hurkok problémája

30

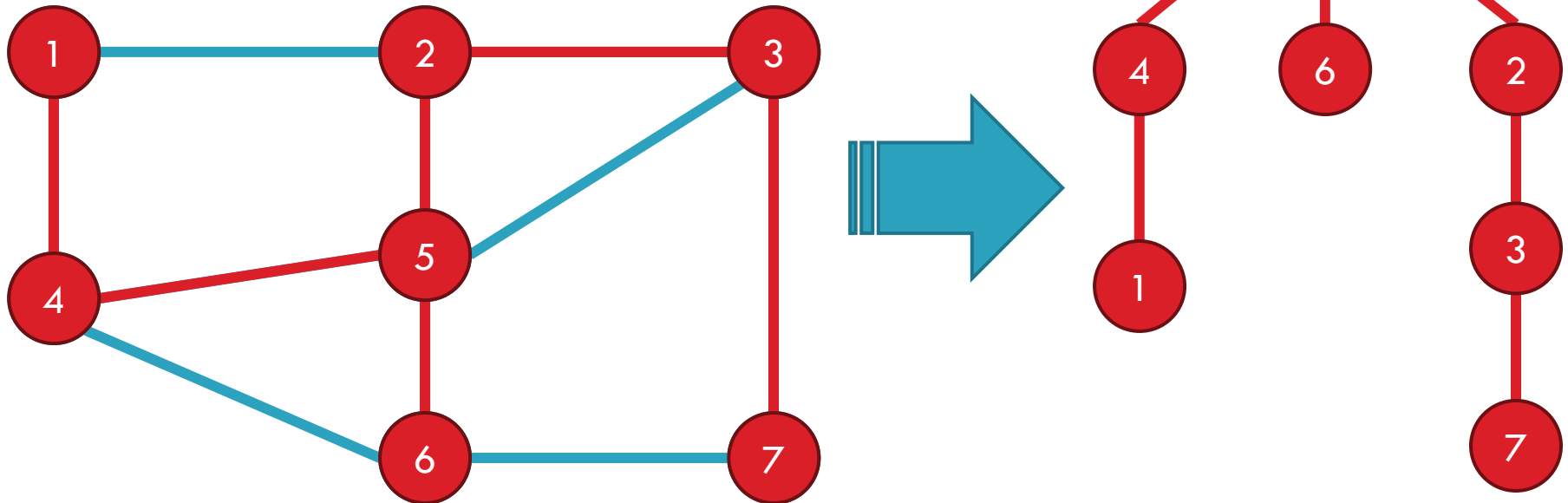
- ❑ $\langle \text{Src}=\text{AA}, \text{Dest}=\text{DD} \rangle$
- ❑ Ez megy a végtelenségig
 - ▣ Hogyan állítható meg?
- ❑ Távolítsuk el a hurkokat a topológiából
 - ▣ A kábelek kihúzása nélkül
- ❑ 802.1 (LAN) definiál egy algoritmust **feszítőfa** építéséhez és karbantartásához, mely mentén lehetséges a keretek továbbítása



Feszítőfa

31

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



A 802.1 feszítőfa algoritmus

32

1. Az egyik bridge-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

33

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

Bridge ID

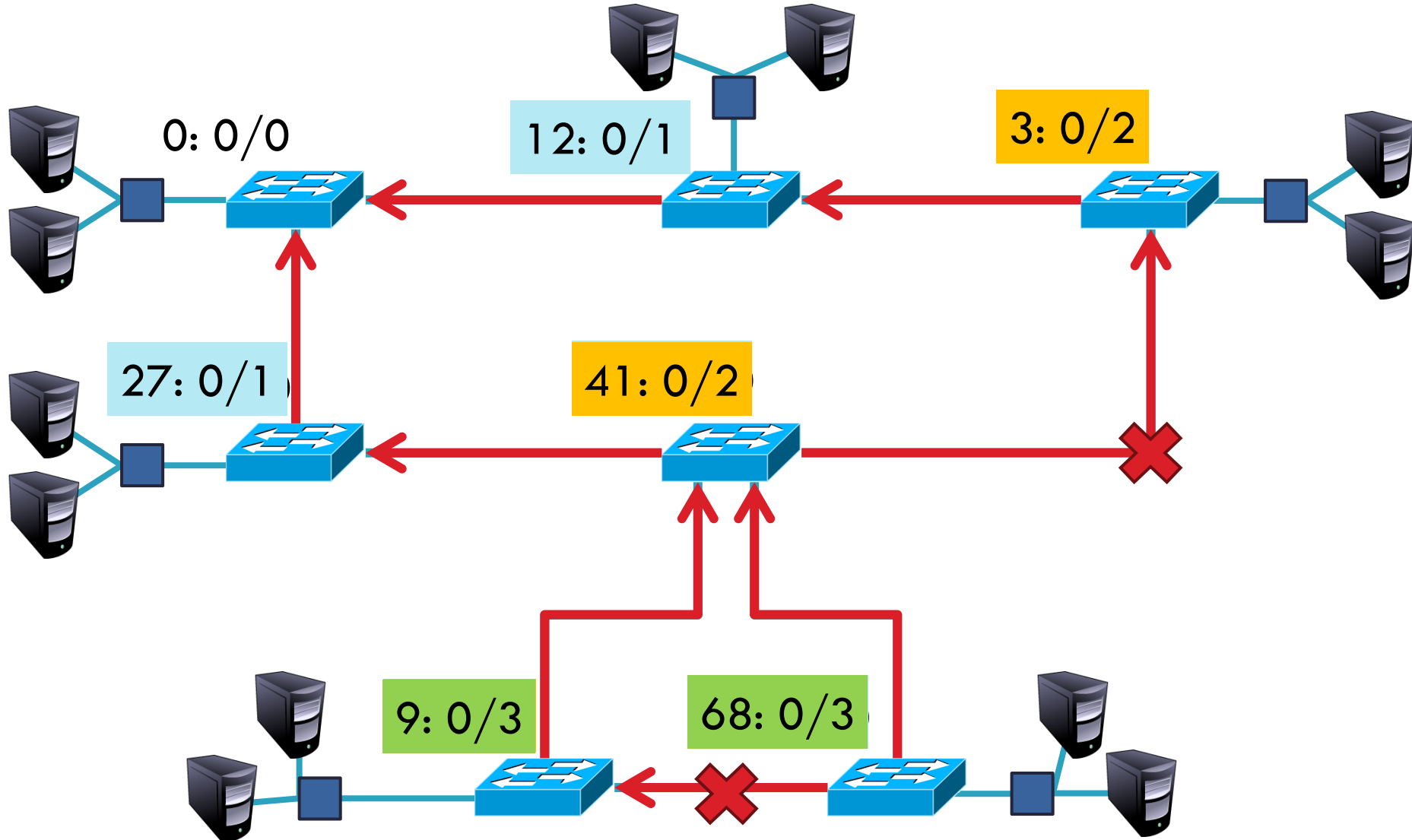
Gyökér ID

Út költség a gyökérhez

- ❑ 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

34



Bridge-ek vs. Switch-ek

Hidak vs. Kapcsolók

35

- ❑ 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

36

- ❑ 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

37

- ❑ 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

38



- 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

39

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

40

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

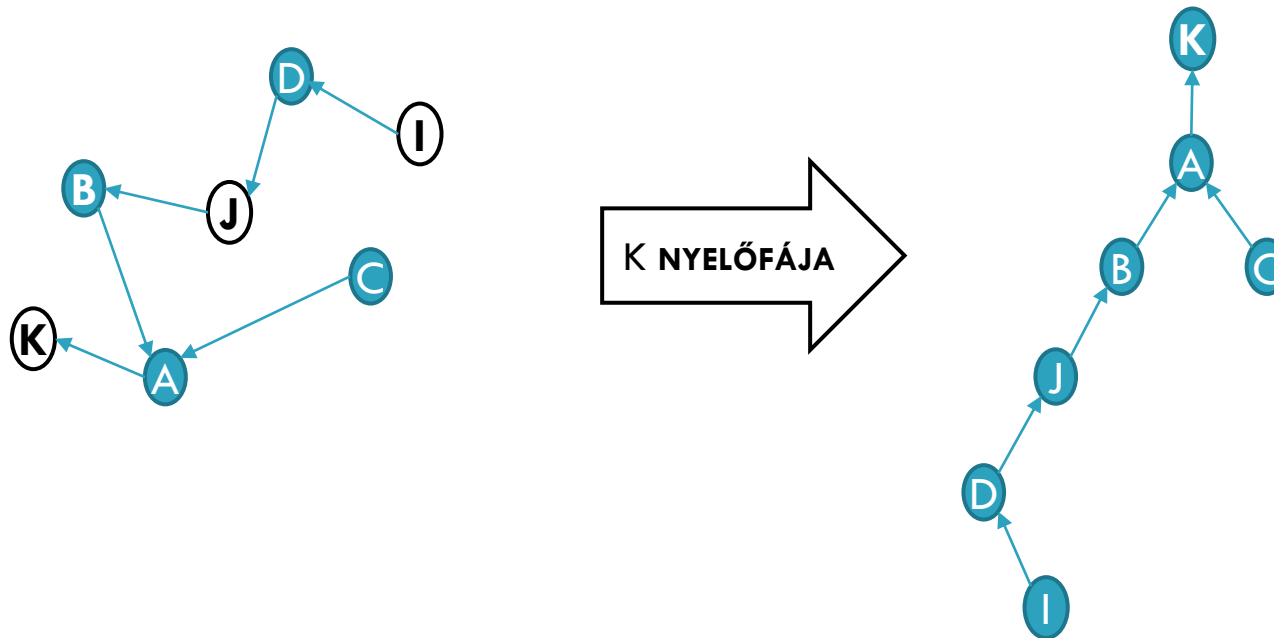
Optimalitási elv

41

Ha J router az I router-től K router felé vezető *optimális útvonalon* helyezkedik el, akkor a J -tól 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 *nyelőfának*.



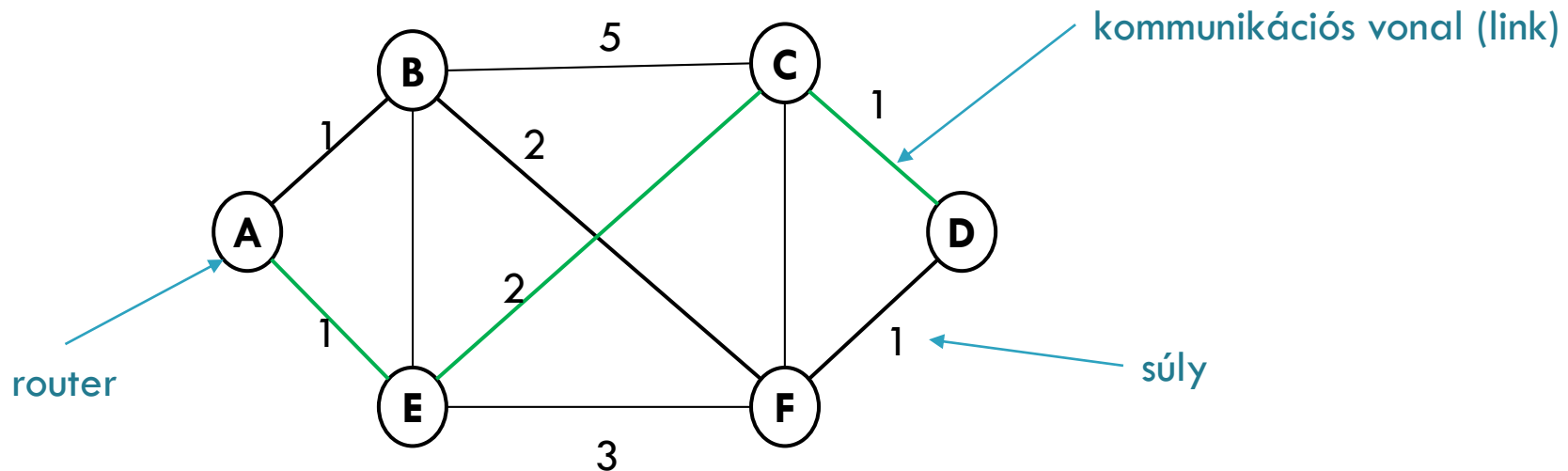
Legrövidebb út alapú forgalomirányítás

42

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: $w(P) = \sum_{e \in P} w(e)$



Távolságvektor alapú forgalomirányítás

43

- 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 táblázatok a szomszédoktól származó információk alapján frissítik.
 - ▣ Elosztott Bellman-Ford forgalomirányítási algoritmusként is nevezik.
 - ▣ ARPANET eredeti forgalomirányító algoritmus ez volt. RIP (Routing Information Protocol) néven is ezt használták.

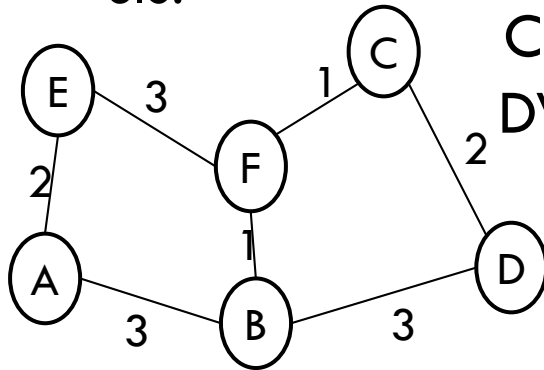
Távolságvektor alapú forgalomirányítás

Elosztott Bellman-Ford algoritmus

44

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.



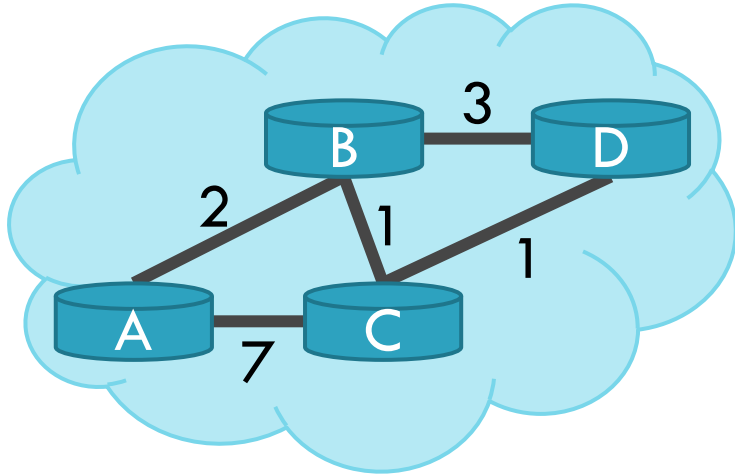
C állomás
DV táblája

Cél	Ktsz.
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

45



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

Node C

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

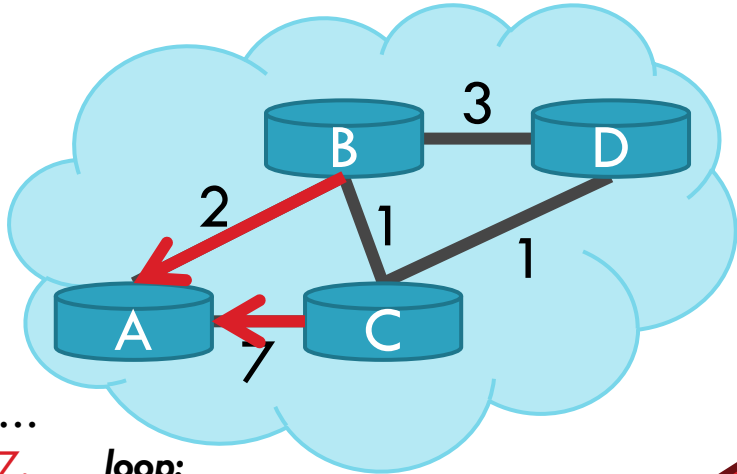
Node D

Dest.	Cost	Next
A	∞	
B	3	B
C	1	C

1. **Initialization:**
2. **for all** neighbors V **do**
3. **if** V adjacent to A
4. $D(A, V) = c(A, V);$
5. **else**
6. $D(A, V) = \infty;$
- ...

Distance Vector: 1st Iteration

46



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) received)
13.   for all destinations Y
14.     if (destination Y is not A)
15.       D(A, Y) = min(D(A, Y), D(A, B) + D(B, Y))
16.     else
17.       D(A, Y) = min(D(A, Y), D(A, C) + D(C, Y))
18.   if (there is a new min. for dest. Y)
19.     send D(A, Y) to all neighbors
20. forever
  
```

$D(A, C)$

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

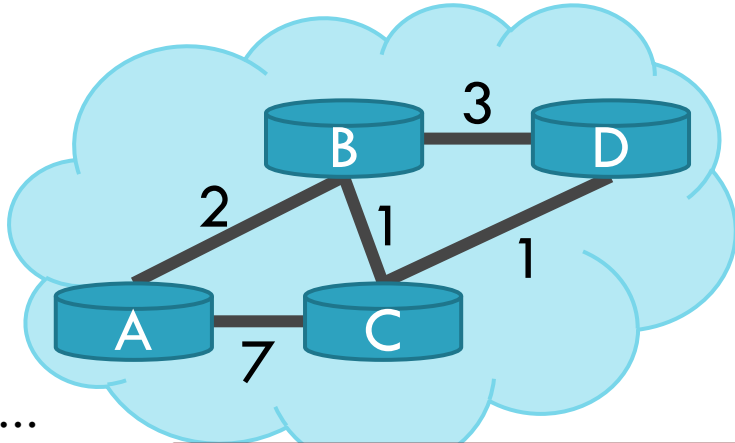
$D(A, D) = \min(D(A, D), D(A, B) + D(B, D))$
 $= \min(8, 3 + 3) = 5$

4

		Next
B	1	B
D	1	D

Distance Vector: End of 3rd Iteration

47



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...

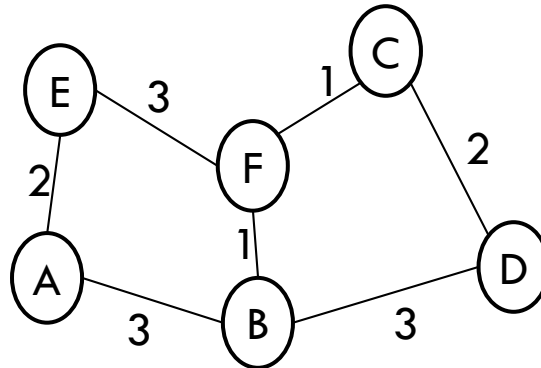
Dest.	Cost	Next
A	3	B
B	1	B
D	1	D

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

```

...
7.  loop
...
12. else
13.  for
14.    if
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
  
```

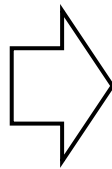
Elosztott Bellman-Ford algoritmus – példa



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 becslét 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

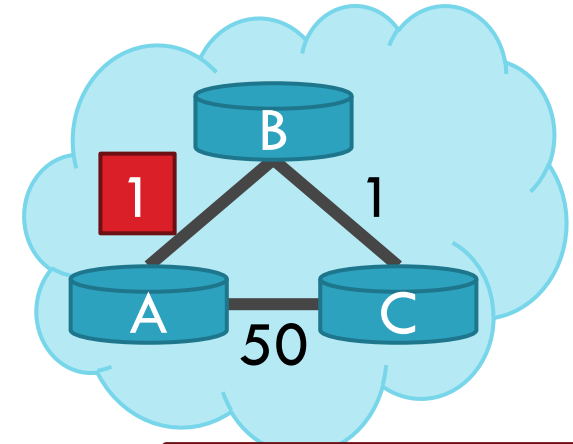


Új becslét késleltetés A-tól		
A	cost	N. Hop
B	3	B
C	5	B
D	6	B
E	2	E
F	4	B


```

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));

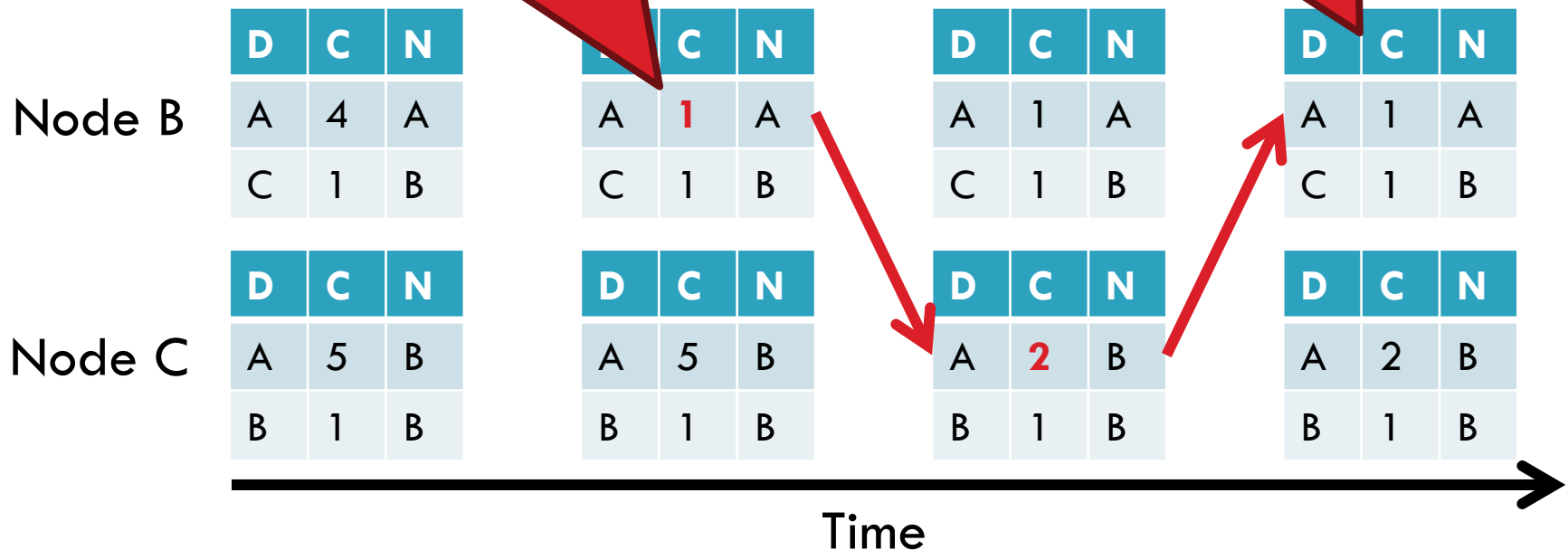
```



Link Cost
Algorithm

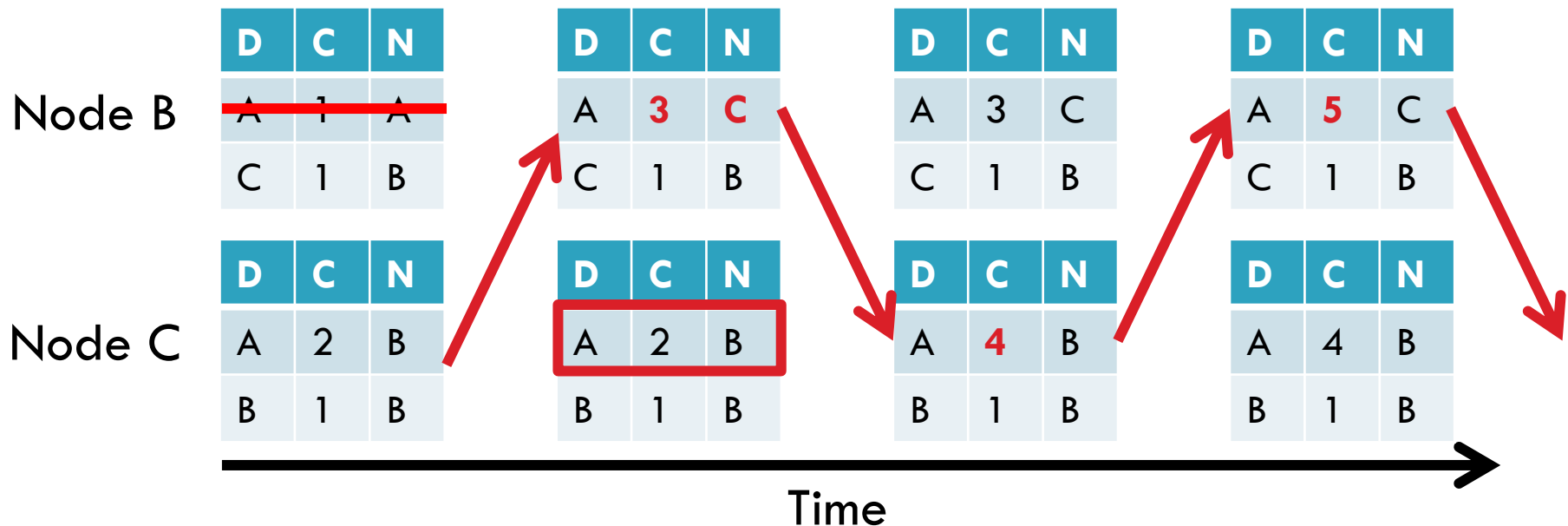
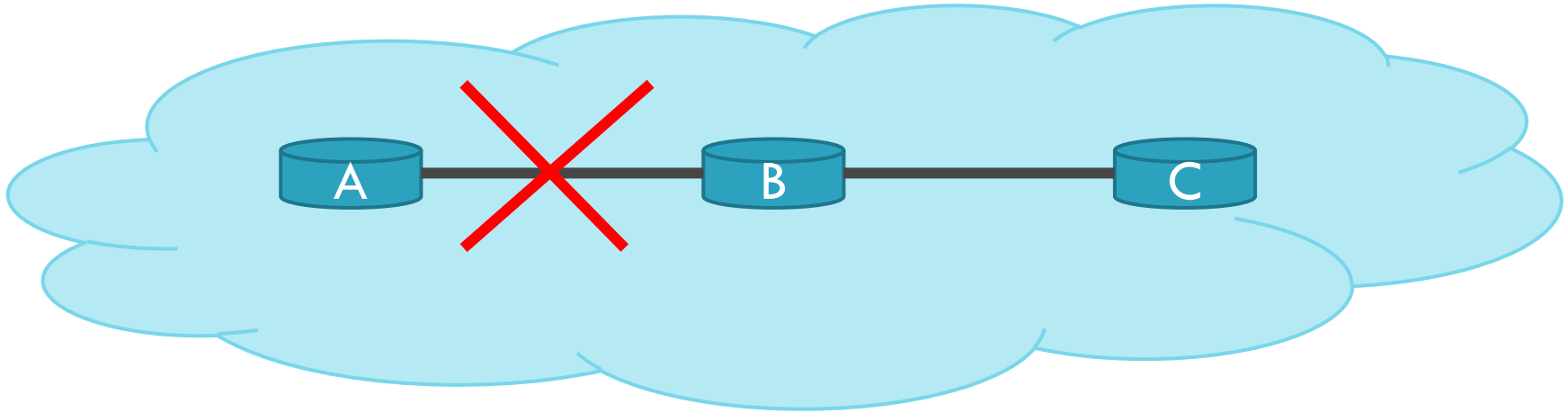
Good news travels fast

Algorithm
terminates



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

50

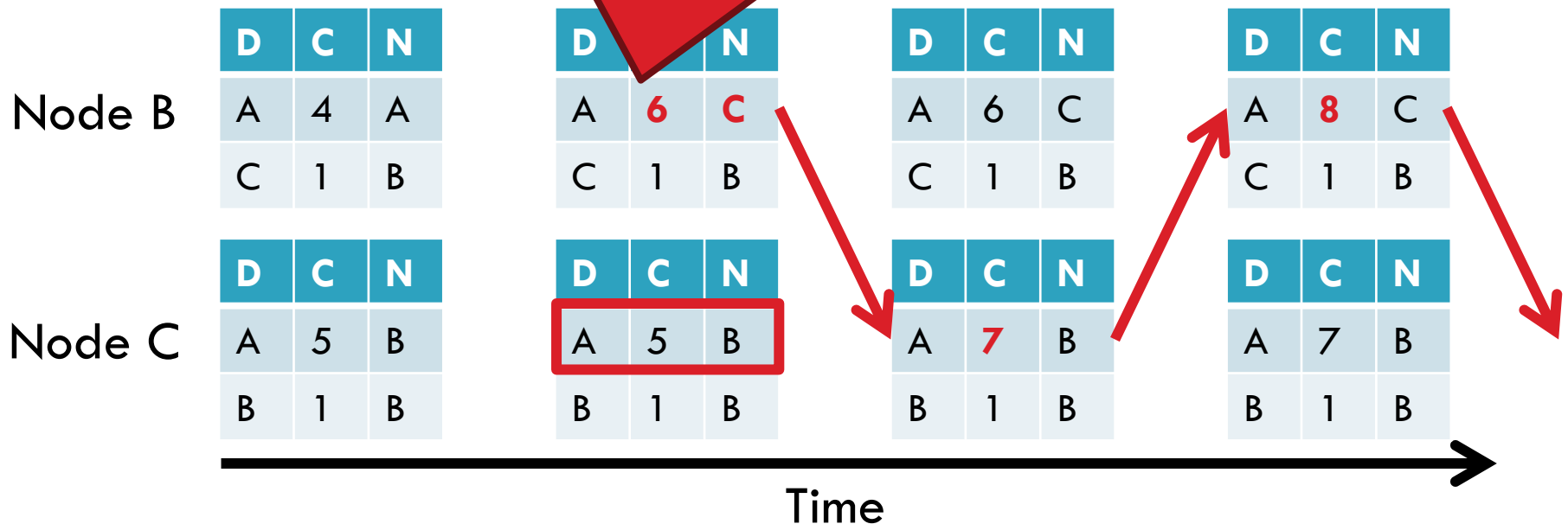
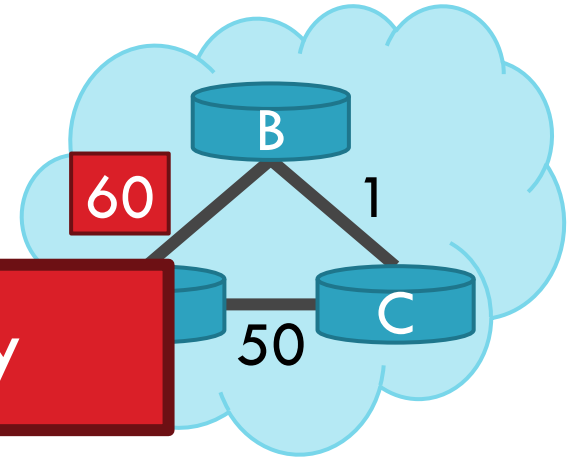


Példa - Count to Infinity Problem

51

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

Bad news travels slowly



Elosztott Bellman-Ford algoritmus – *Végtelenig számolás problémája*

52

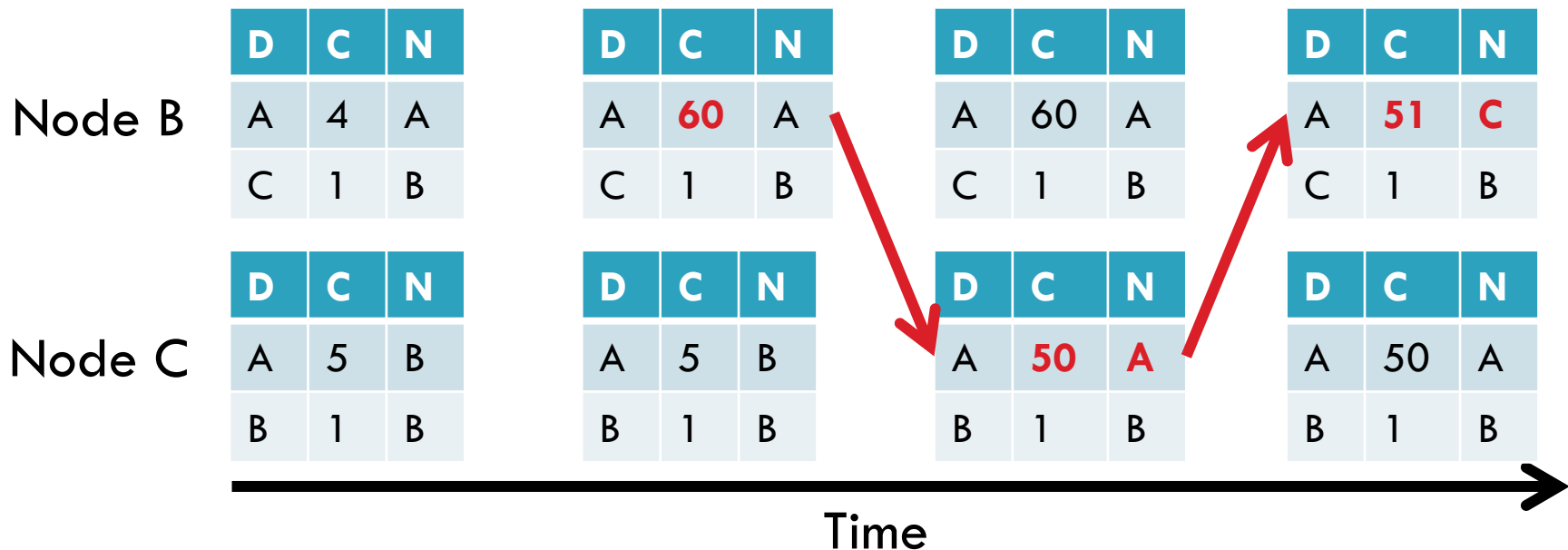
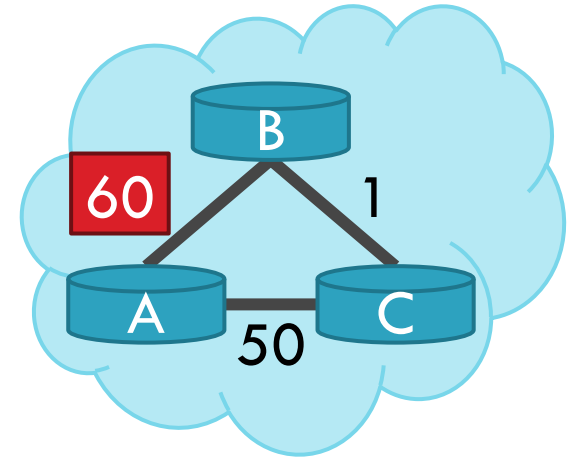
PROBLÉMA

- 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

53

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



Kapcsolatállapot alapú forgalomirányítás

Link-state routing

54

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

55

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 oldálnak 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 Advertishment, azaz kapcsolatállapot hírdetés)

Kapcsolatállapot alapú forgalomirányítás működése

56

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

57

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 algoritmus, 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)

58

- 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 pseudo-kód

59

Dijkstra(G, s, w)

Output: egy legrövidebb utak fája $T=(V, E')$ G -ben s gyökérrel

```
01  $E' := \emptyset$ ;  
02  $ready[s] := true$ ;  
03  $ready[v] := false; \forall v \in V \setminus \{s\}$ ;  
04  $d[s] := 0$ ;  
05  $d[v] := \infty; \forall v \in V \setminus \{s\}$ ;  
06  $priority\_queue\ Q$ ;  
07 forall  $v \in Adj[s]$  do  
08    $pred[v] := s$ ;  
09    $d[v] := w(s, v)$ ;  
10    $Q.Insert(v, d[v])$ ;
```

INICIALIZÁCIÓS FÁZIS

```
11 od  
12 while  $Q \neq \emptyset$  do
```

```
13    $v := Q.DeleteMin()$ ;  
14    $E' := E' \cup \{(pred[v], v)\}$ ;  
15    $ready[v] := true$ ;  
16   forall  $u \in Adj[v]$  do
```

```
17     if  $u \in Q$  and  $d[v] + w(v, u) < d[u]$  then  
18        $pred[u] := v$ ;  
19        $d[u] := d[v] + w(v, u)$ ;
```

JAVÍTÓ ÚT

```
20      $Q.DecreasePriority(u, d[u])$ ;
```

```
21     else if  $u \notin Q$  and not  $ready[u]$  then  
22        $pred[u] := v$ ;  
23        $d[u] := d[v] + w(v, u)$ ;
```

ÚJ ÚT

```
24      $Q.Insert(u, d[u])$ ;  
25   fi
```

```
26 od  
27 od
```

ITERÁCIÓS LÉPÉSEK

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

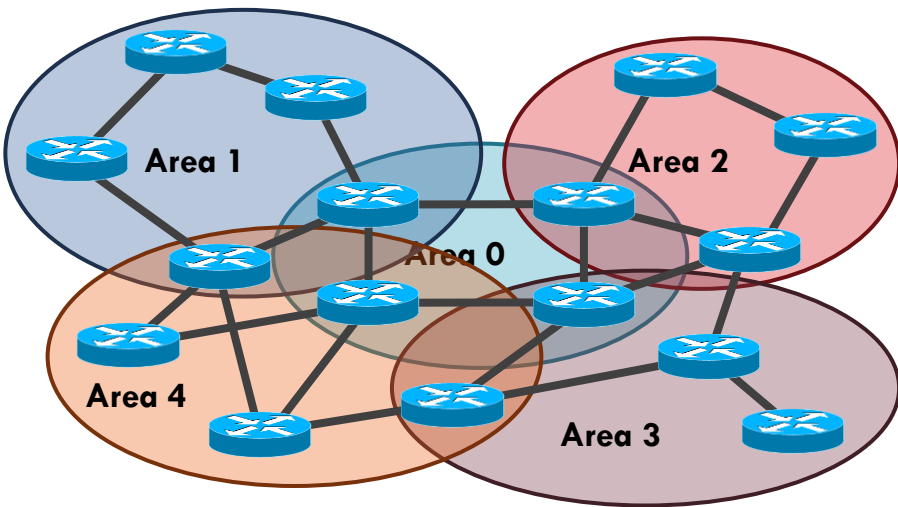
IS-IS

- 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

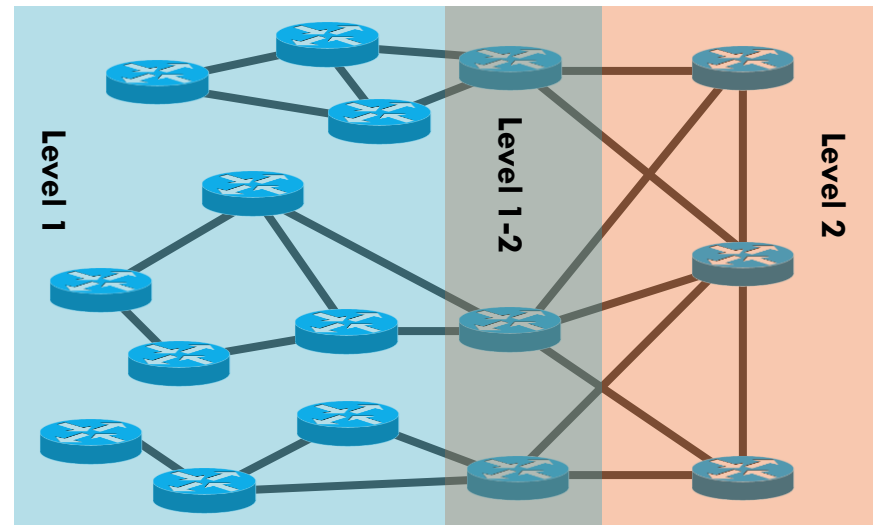
OSPF

- Átfedő területek köré szerveződik
- Area 0 a hálózat magja



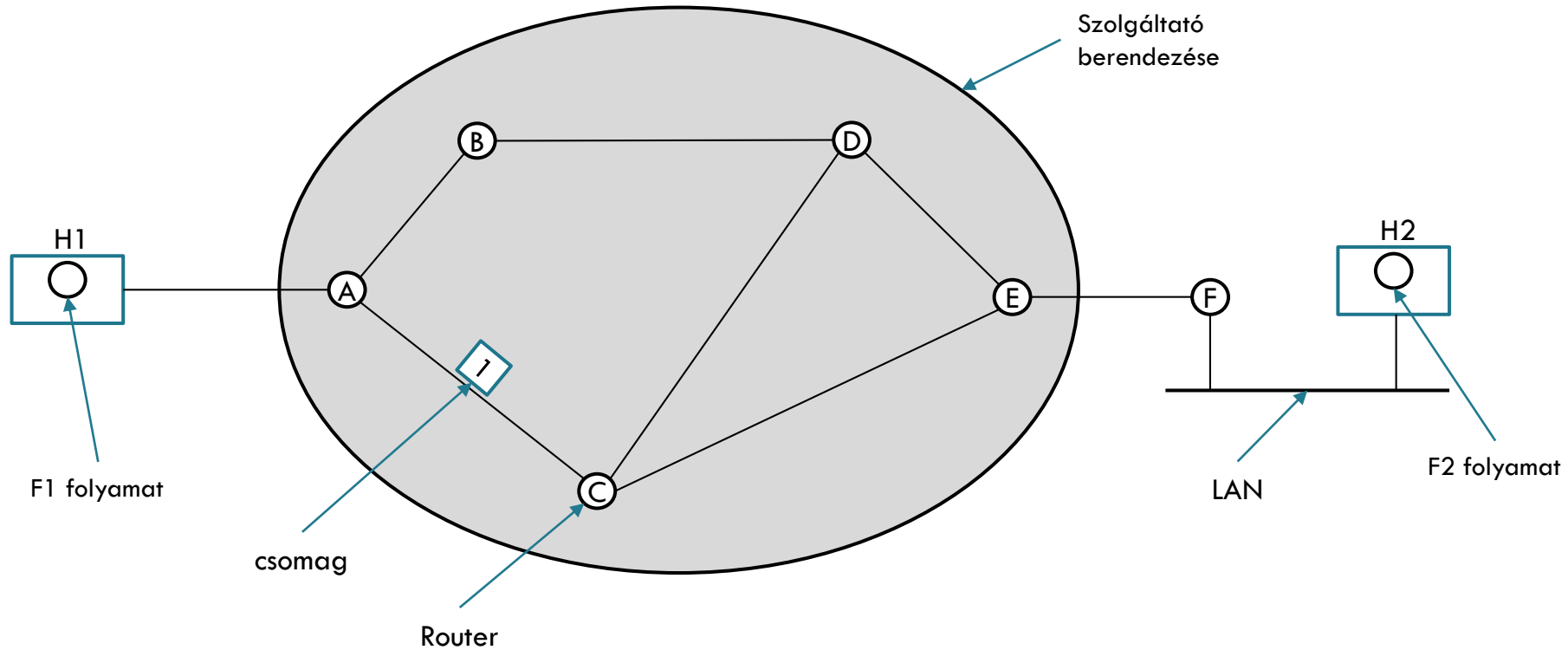
IS-IS

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



Hálózati réteg protokolljai - Környezet

62



Szállítási réteg felé nyújtott szolgáltatok

63

VEZÉRELVEK

1. A szolgáltat 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 szolgáltat (*Internet*)
 - datagram alhálózat
- Összeköttetés alapú szolgáltat (*ATM*)
 - virtuális áramkör alhálózat



HÁLÓZATI RÉTEG — FORGALOMIRÁNYÍTÁS

Unicast forgalomirányítás

65

- Legegyszerűbb és legáltalánosabb eset
- Csomag küldése két végpont között
- Forrás és cél egyedi azonosítóval rendelkezik (Internet esetén: IPv4 v. IPv6 címek).

Adatszóró forgalomirányítás

66

- **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

67

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

68

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

69

- **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*et 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.

Köszönöm a figyelmet!