

# Számítógépes Hálózatok

## 6. Előadás: Adatkapcsolati réteg IV. & 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

# Ütközésmentes protokollok

2

## 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

3

- 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

4

- 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

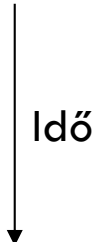
5

- **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

6

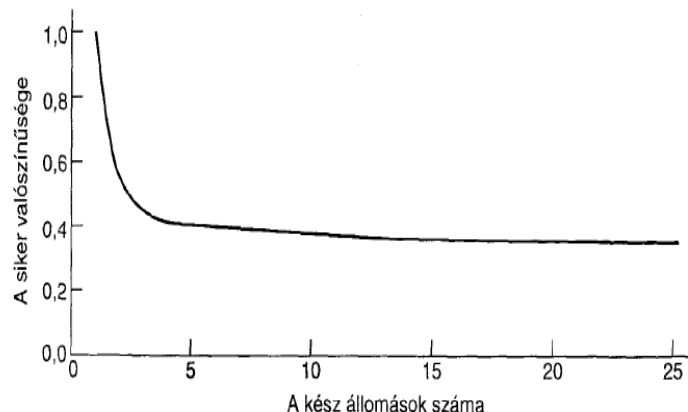
- **Cél:** Ötvözni a versenyhelyzetes és ütközésmentes protokollok jó tulajdonságait.
- **korlátozott versenyes protokoll** – Olyan protokoll, amely kis terhelés esetén versenyhelyzetes technikát használ a kis késleltetés érdekében, illetve nagy terhelés mellett ütközésmentes technikát alkalmaz a csatorna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

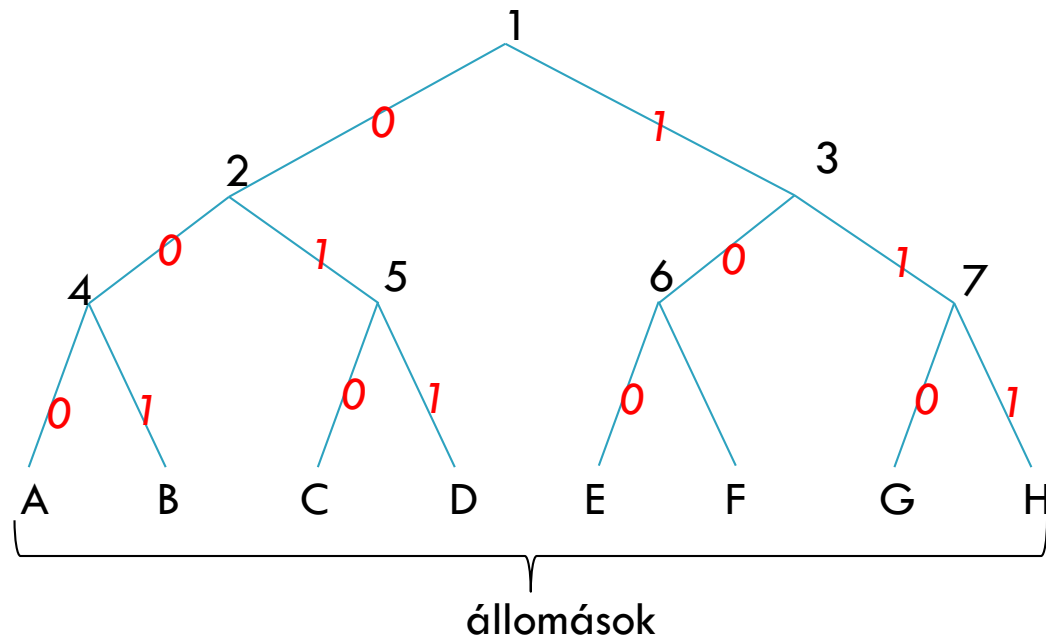


# Adaptív fabejárési protokoll 1 / 2

7

## Történeti háttér

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



# Adaptív fabejárási protokoll 2/2

8

## 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.

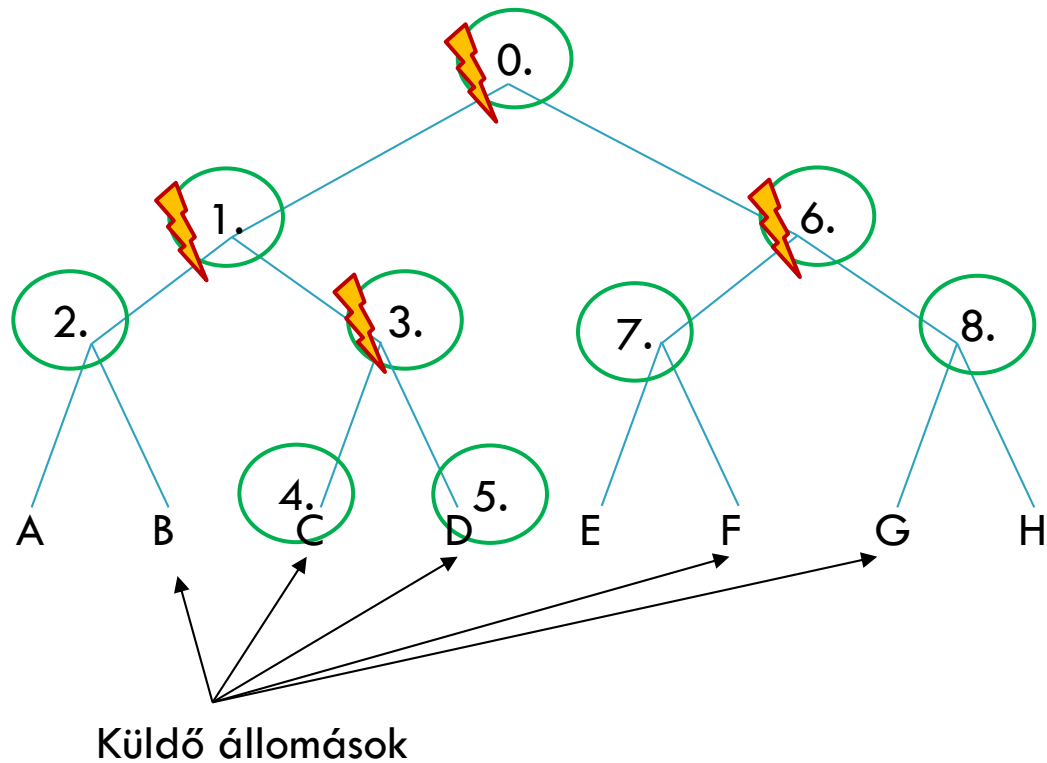
## Következmény

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



# Adaptív fabejárás példa

9



# Az adatkapcsolati réteg „legtetején”...

10

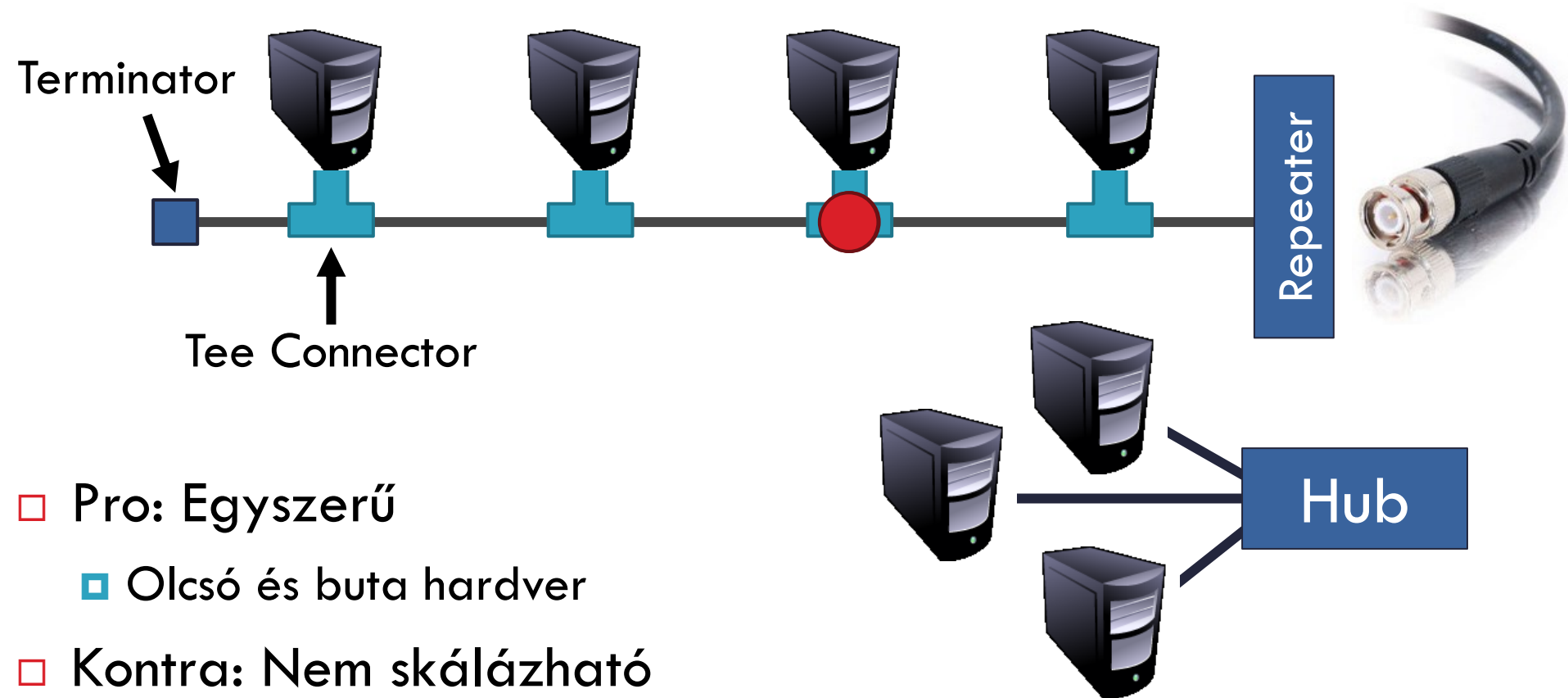


- Bridging, avagy hidak
  - ▣ Hogyan kapcsoljunk össze LAN-okat?
- Funkciók:
  - ▣ Keretek forgalomirányítása a LAN-ok között
- Kihívások:
  - ▣ Plug-and-play, önmagát konfiguráló
  - ▣ Esetleges hurkok feloldása

# Visszatekintés

11

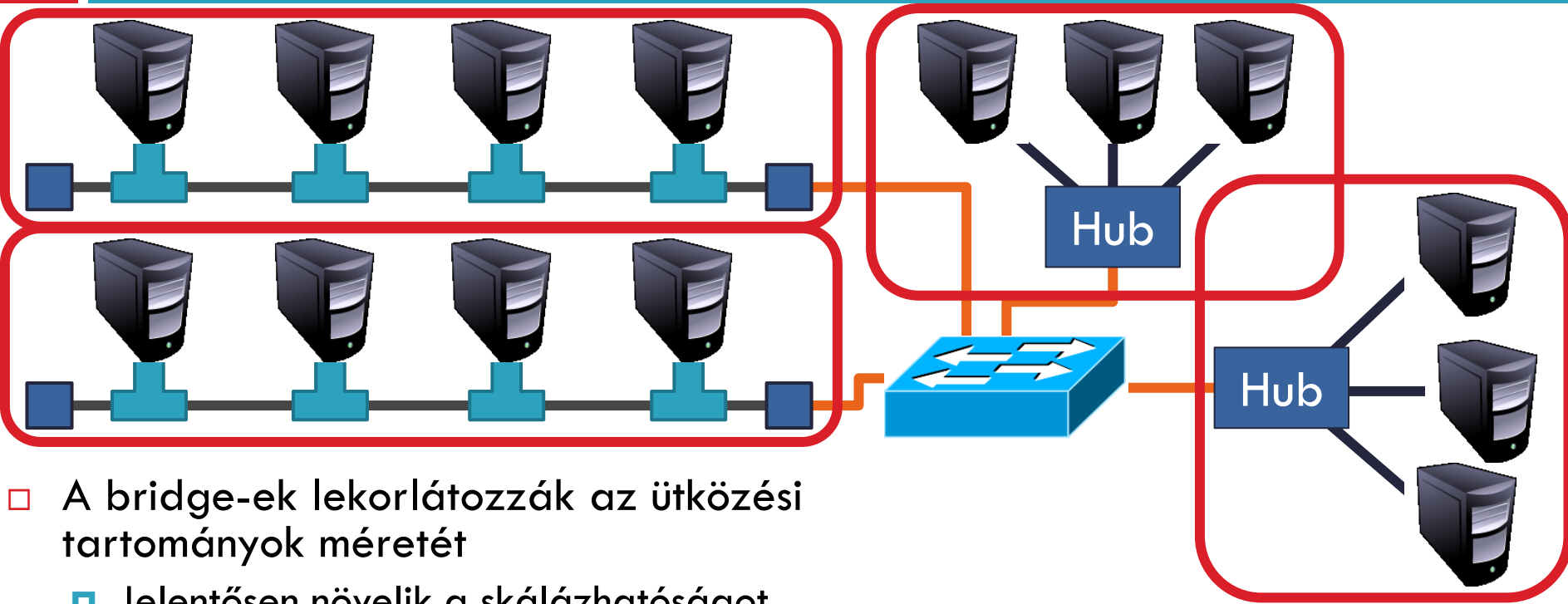
- Az Ethernet eredetileg adatszóró technológia volt



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

# LAN-ok összekapcsolása

12

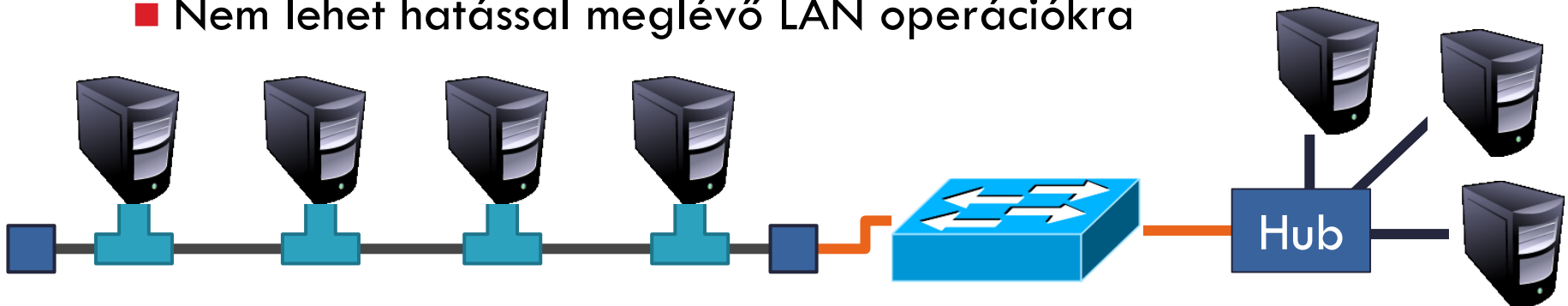


- A bridge-ek lekorlátozzák az ütközési tartományok méretét
  - ▣ Jelentősen növelik a skálázhatóságot
  - ▣ Kérdés: lehetne-e az egész Internet egy bridge-ekkel összekötött tartomány?
- Hátrány: a bridge-ek sokkal komplexebb eszközök a hub-oknál
  - ▣ Fizikai réteg VS Adatkapcsolati réteg
  - ▣ Memória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

# Bridge-ek (magyarul: hidak)

13

- ❑ 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)

14

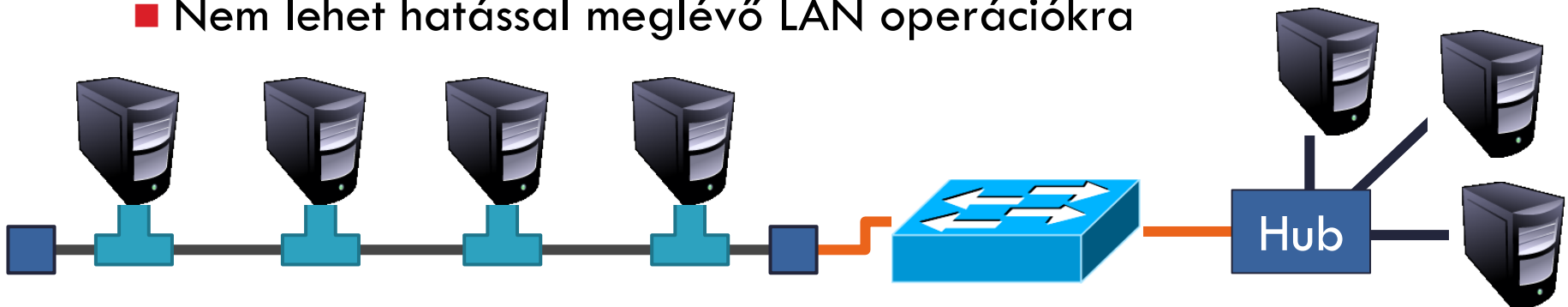
- Az Ethernet switch eredeti formája

□

□

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

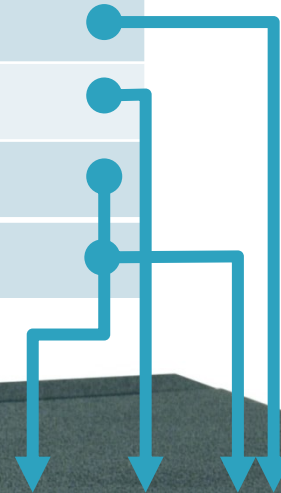


# Keret Továbbító Táblák

15

- 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



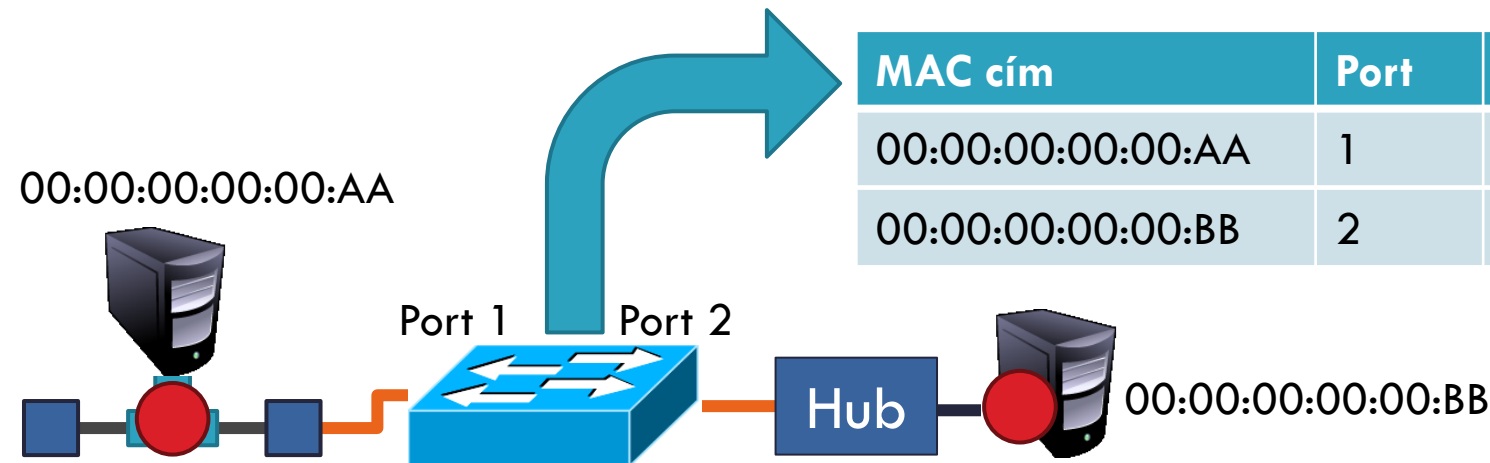
# Címek tanulása

16

- ❑ 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

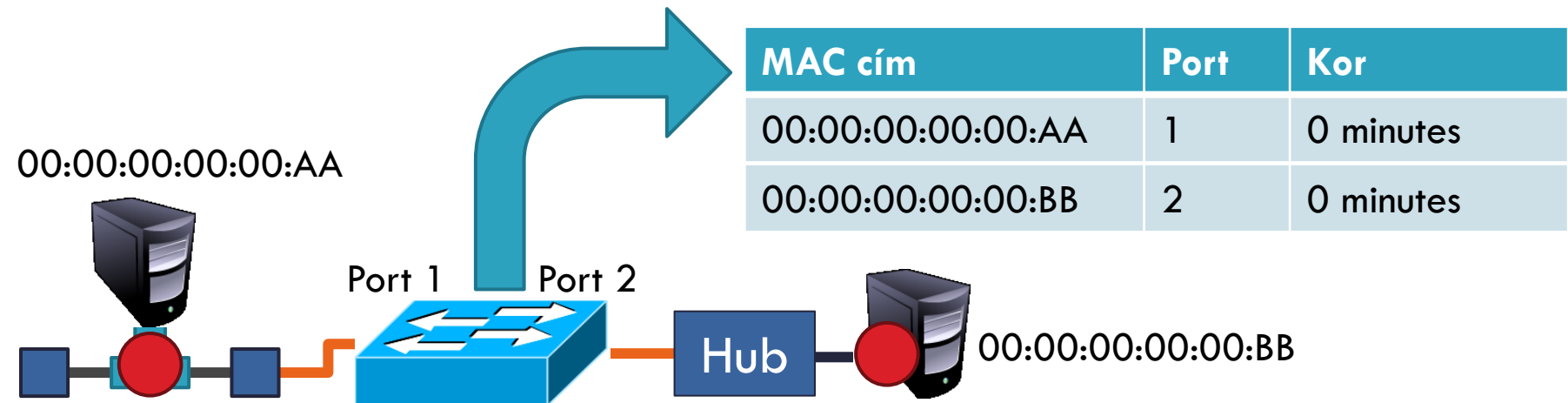




# Címek tanulása

17

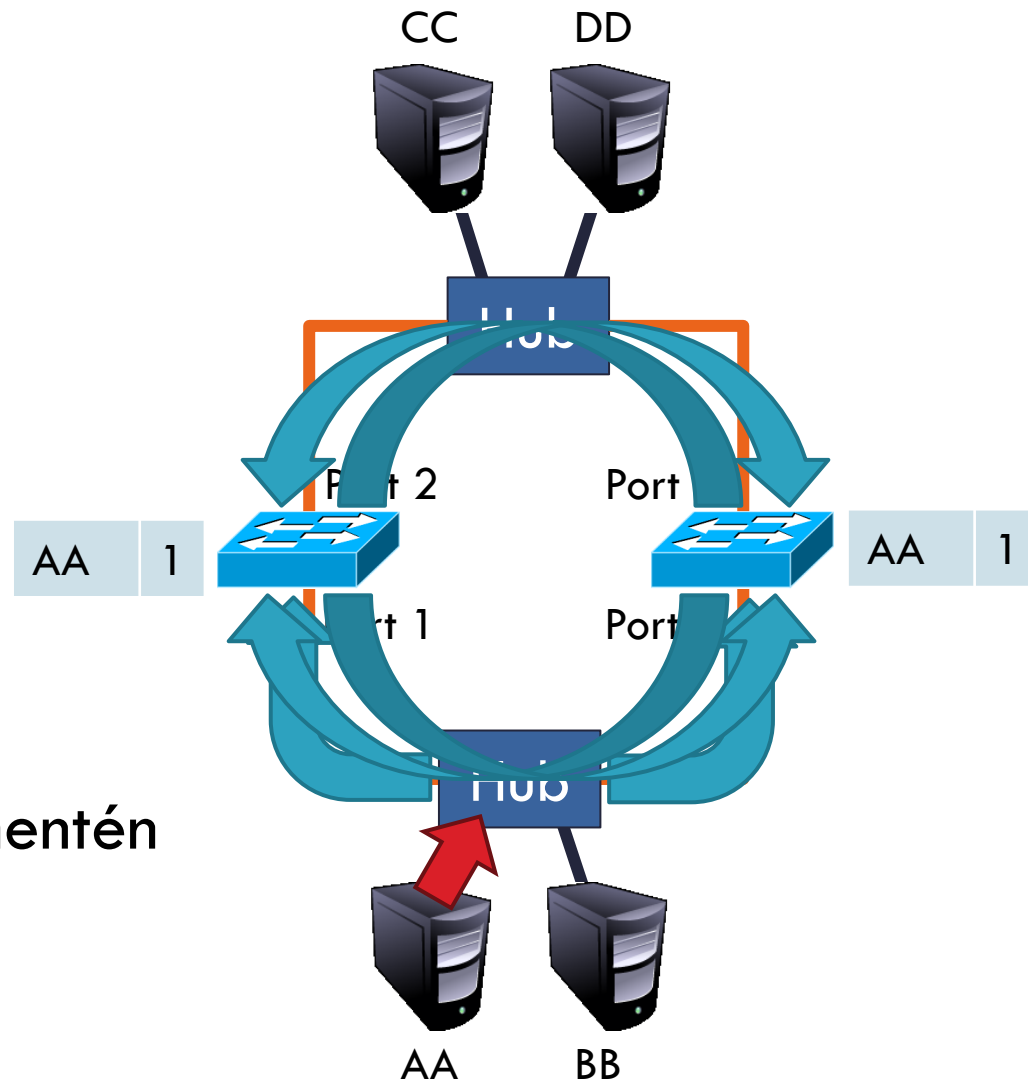
- ❑ Kézi beállítás is lehetséges, de...
  - ▣ Időigényes
  - ▣ Potenciális hiba forrás
  - ▣ Nem alkalmazkodik a változásokhoz (új hosztok léphetnek be és régiek hagyhatják el a hálózatot)
- ❑ Ehelyett: tanuljuk meg a címeket
  - ▣ Tekintsük a **forrás címeket** a különböző portokon beérkező kereteknek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



# Hurkok problémája

18

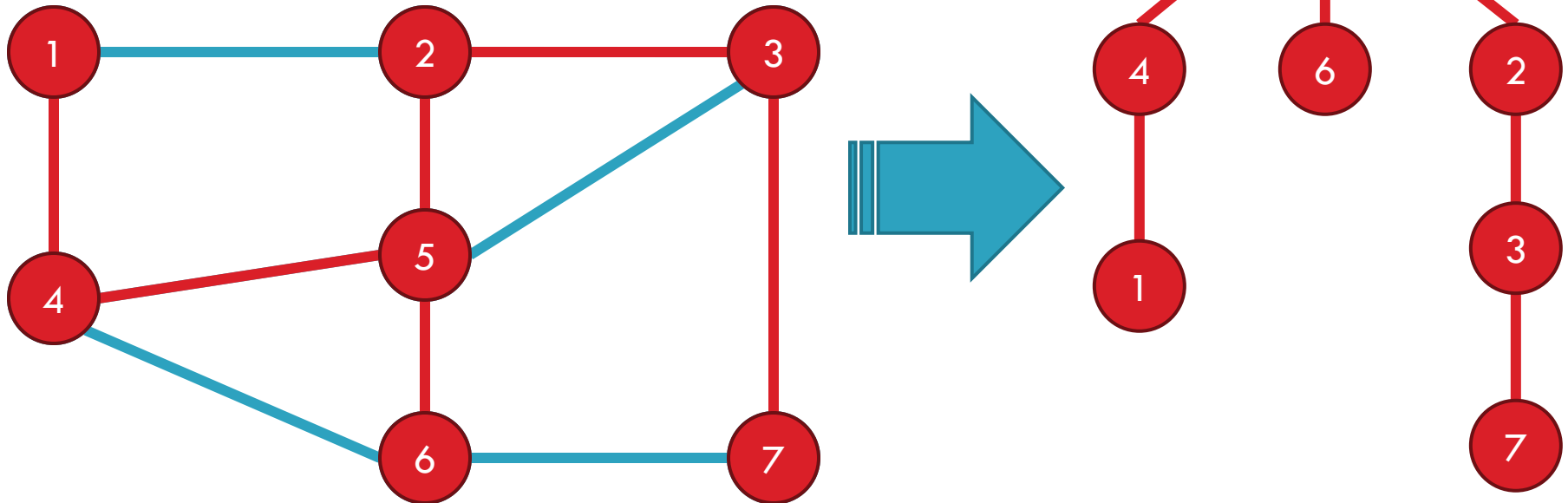
- ❑  $\langle \text{Src}=\text{AA}, \text{Dest}=\text{DD} \rangle$
- ❑ Ez megy a végtelenségig
  - ▣ Hogyan állítható meg?
- ❑ Távolítsuk el a hurkokat a topológiából
  - ▣ A kábelek kihúzása nélkül
- ❑ 802.1 (LAN) definiál egy algoritmust **feszítőfa** építéséhez és karbantartásához, mely mentén lehetséges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



# Feszítőfa

19

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



# A 802.1 feszítőfa algoritmus

20

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

21

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

Bridge ID

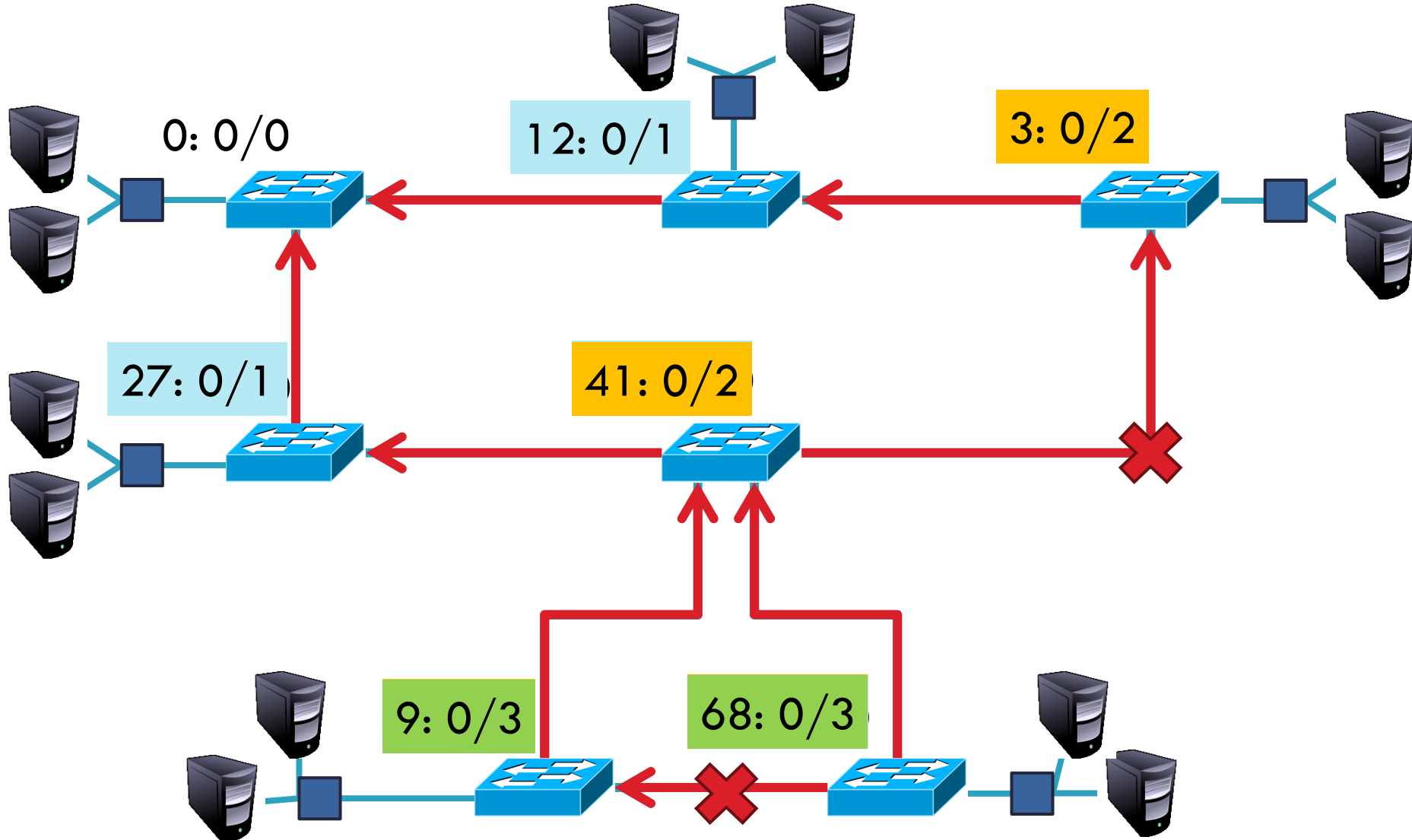
Gyökér ID

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

22



# Bridge-ek vs. Switch-ek

## Hidak vs. Kapcsolók

23

- ❑ 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

24

- ❑ 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

25

- ❑ 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

26



- 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

27

## 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

28

## 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

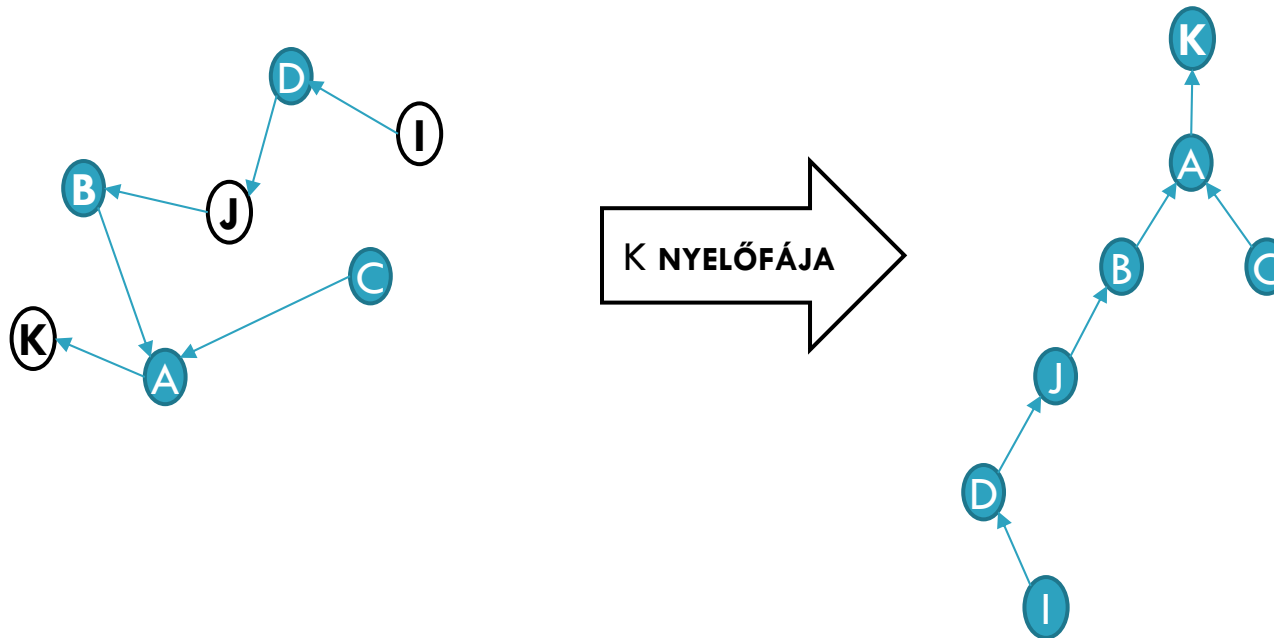
# Optimalitási elv

29

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 *nyelőfának*.



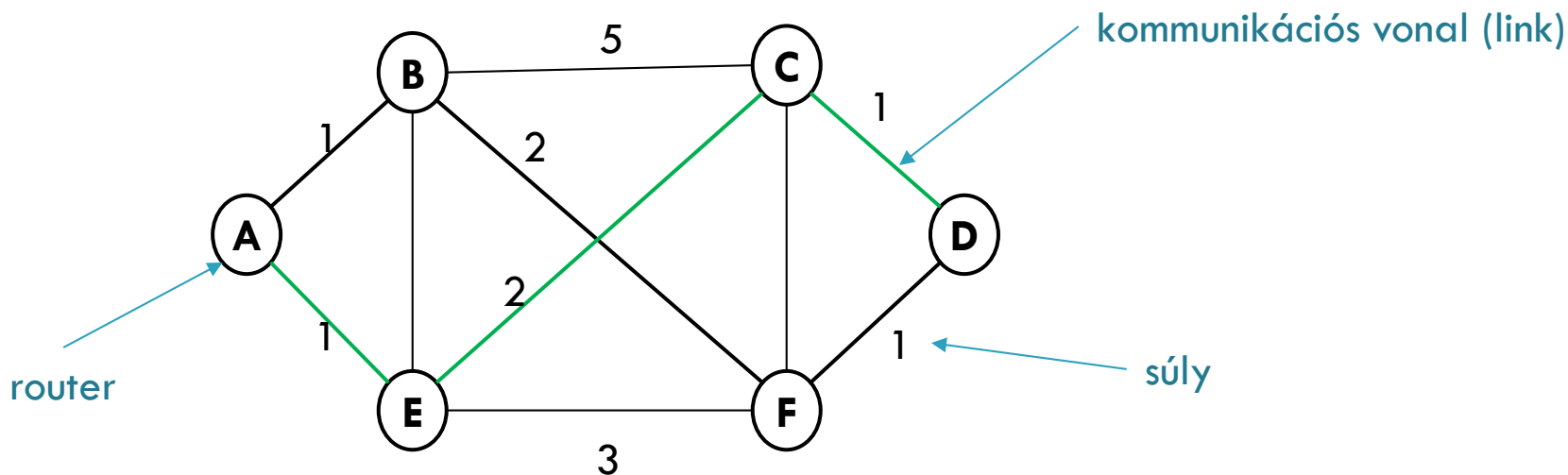
# Legrövidebb út alapú forgalomirányítás

30

## ALHÁLÓZAT REPRESENTÁCIÓJA

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

31

- 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.

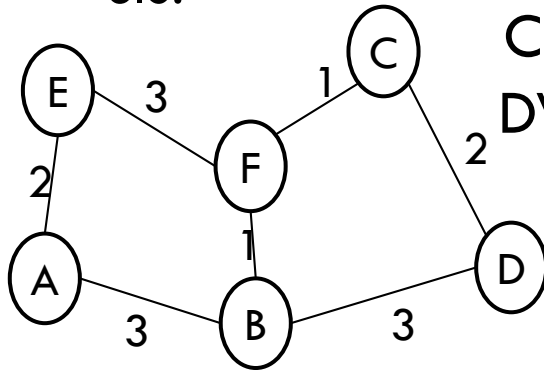
# Távolságvektor alapú forgalomirányítás

## Elosztott Bellman-Ford algoritmus

32

### 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

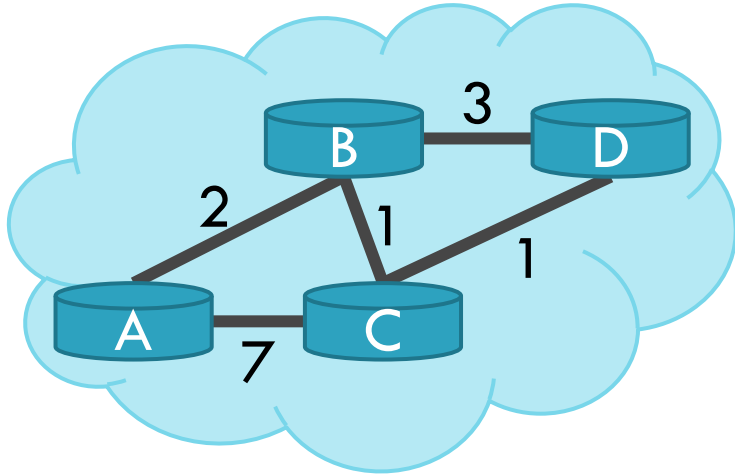
Cél	Ktsz.
A	5
B	2
D	2
E	4
F	1

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



# Distance Vector Initialization

33



Node A

Dest.	Cost	Next
B	2	B
C	7	C
D	$\infty$	

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

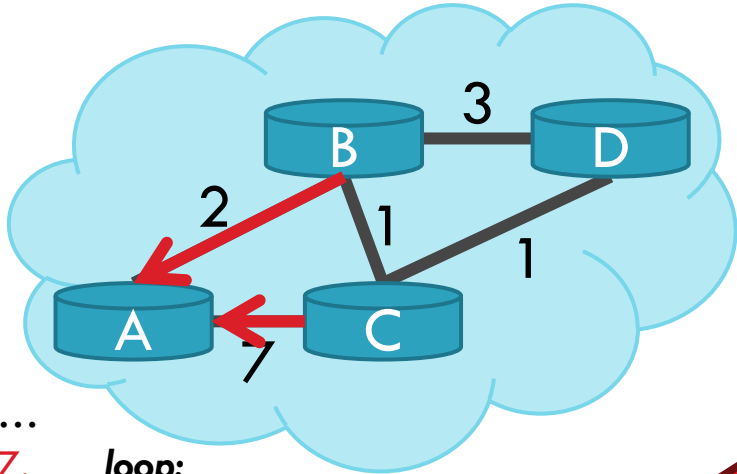
Node D

Dest.	Cost	Next
A	$\infty$	
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: 1<sup>st</sup> Iteration

34



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

$D(A, C)$

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

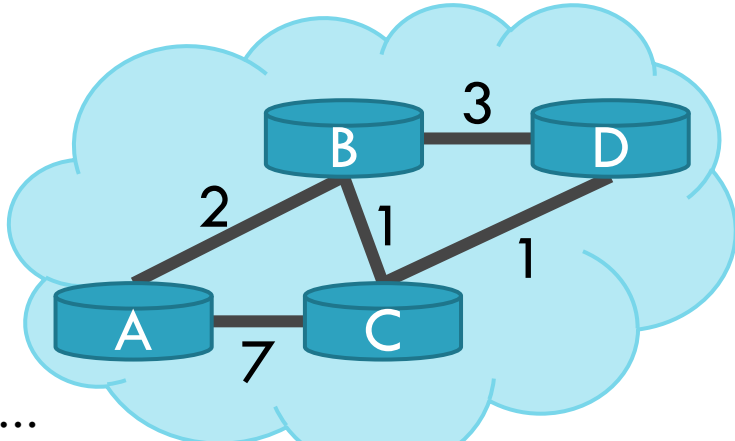
$D(A, D) = \min(D(A, D), D(A, B) + D(B, D))$   
 $= \min(8, 3 + 3) = 5$

4

		Next
B	1	B
D	1	D

# Distance Vector: End of 3<sup>rd</sup> Iteration

35



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	Dest.	Cost	Next
A	3	B	A	4	C
B	1	B	B	2	C
D	1	D	C	1	C

loop

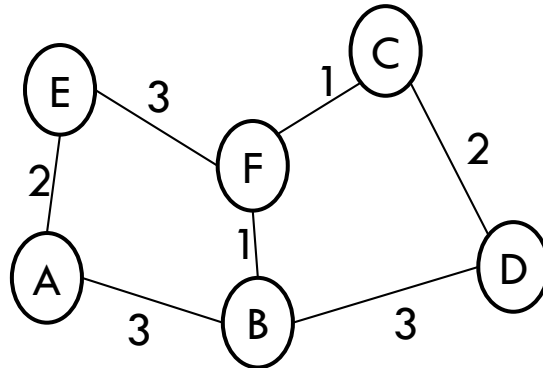
else

for

```

D(A, Y) =
    min(D(A, Y),
        D(A, V) + D(V, Y));
if (there is a new min. for dest. Y)
    send D(A, Y) to all neighbors
forever
  
```

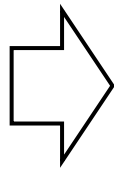
# Elosztott Bellman-Ford algoritmus – példa



Becsült késleltetés A-tól kezdetben		
A	cost	N. Hop
B	3	B
C	$\infty$	-
D	$\infty$	-
E	2	E
F	$\infty$	-

B vektora A-nak	
A	3
B	0
C	$\infty$
D	3
E	$\infty$
F	1

E vektora A-nak	
A	2
B	$\infty$
C	$\infty$
D	$\infty$
E	0
F	3



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

A vektora B-nek és E-nek	
A	0
B	3
C	$\infty$
D	6
E	2
F	4

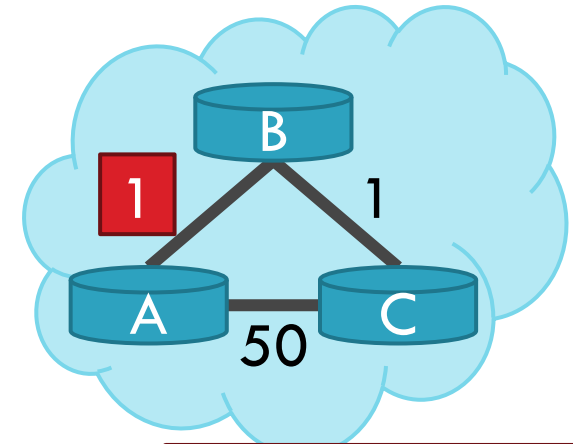


Új becslét késleltetés A-tól		
A	cost	N. Hop
B	3	B
C	5	B
D	6	B
E	2	E
F	4	B

```

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));

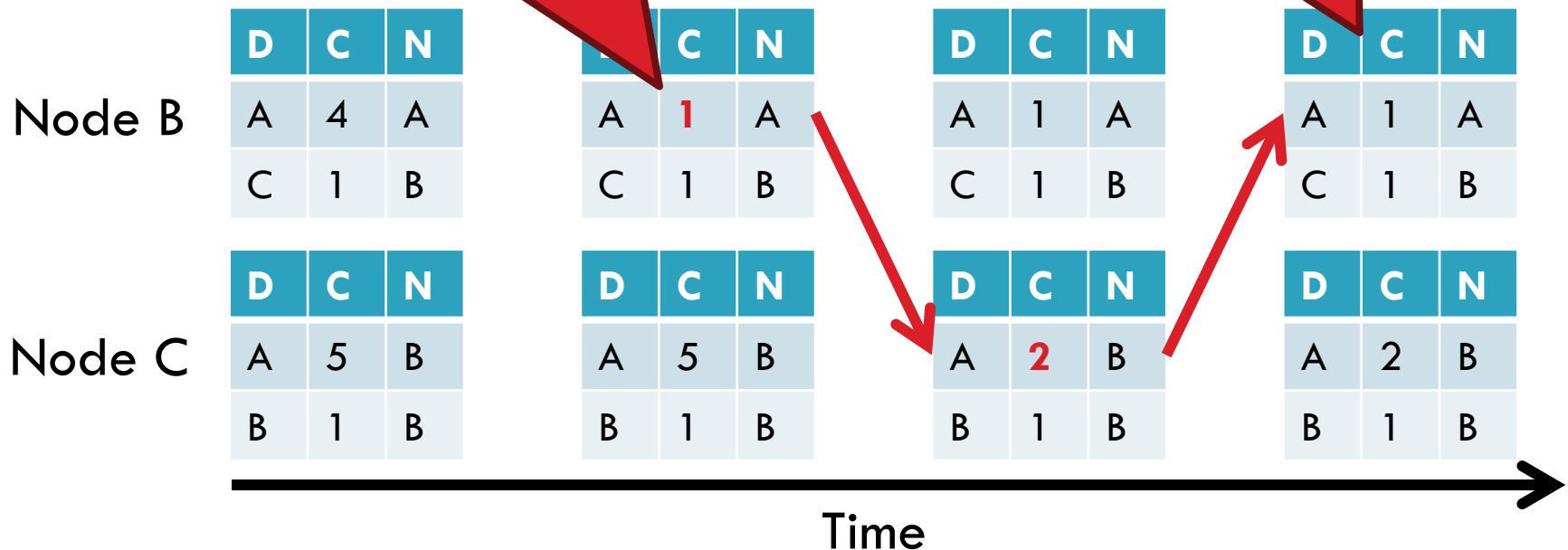
```



Link Cost  
Algorithm

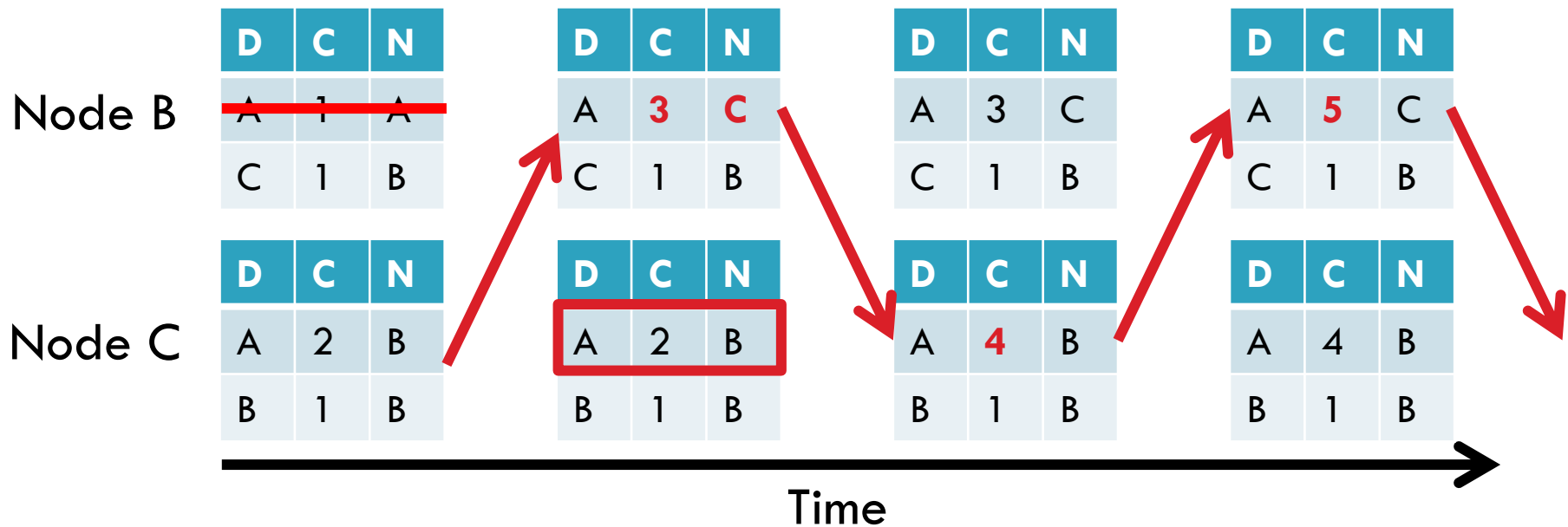
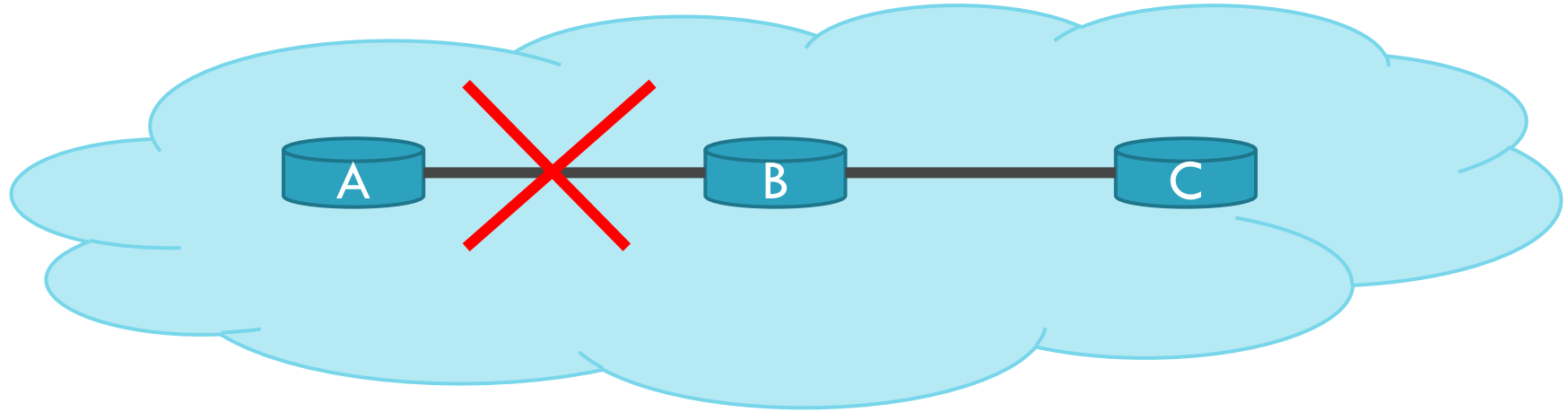
Good news travels fast

Algorithm  
terminates



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

38

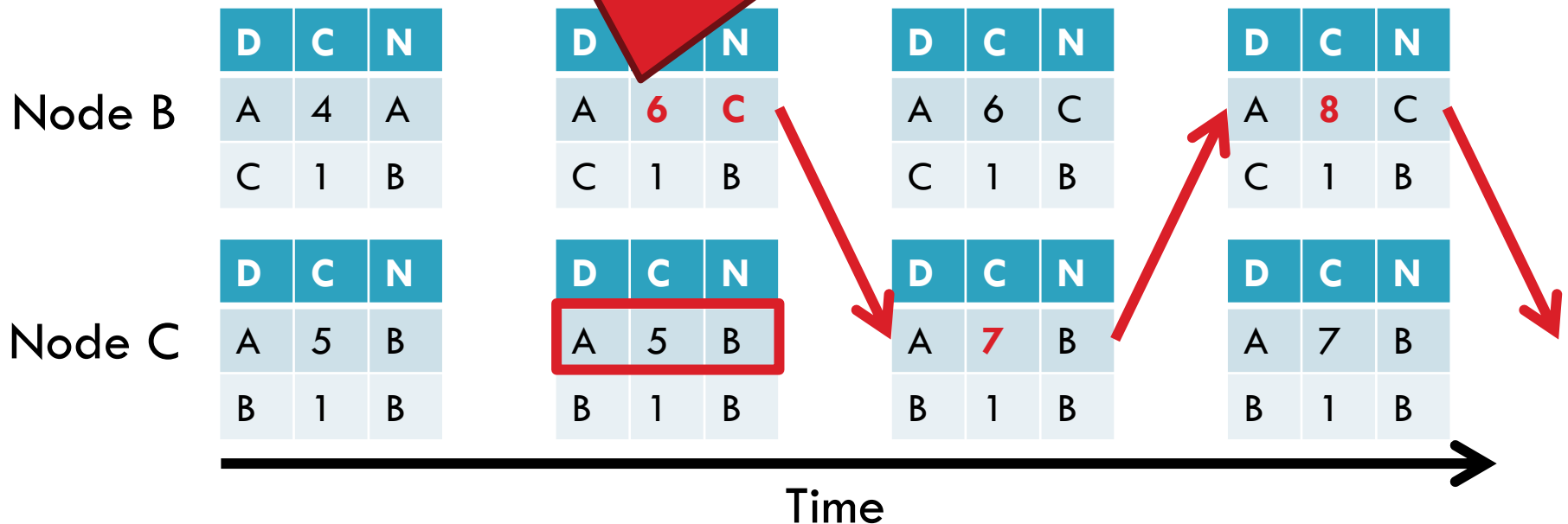
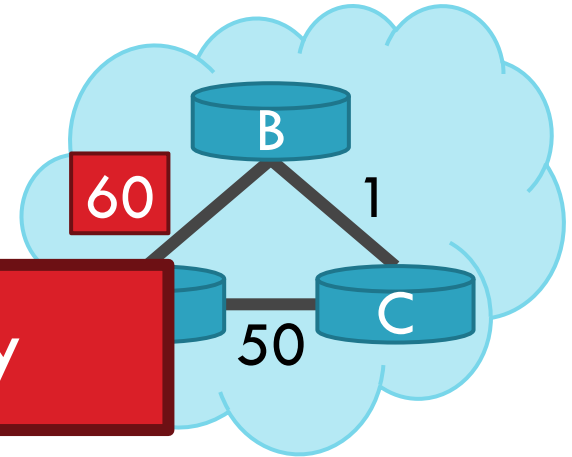


# Példa - Count to Infinity Problem

39

- 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*

40

## PROBLÉMA

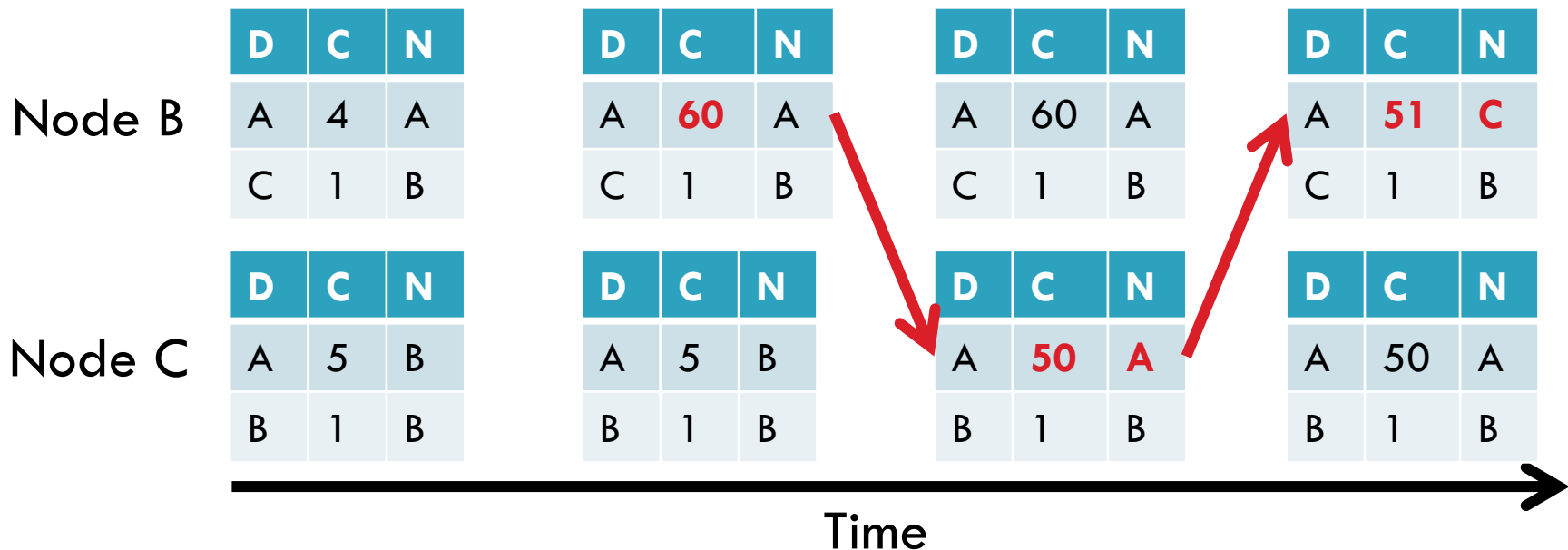
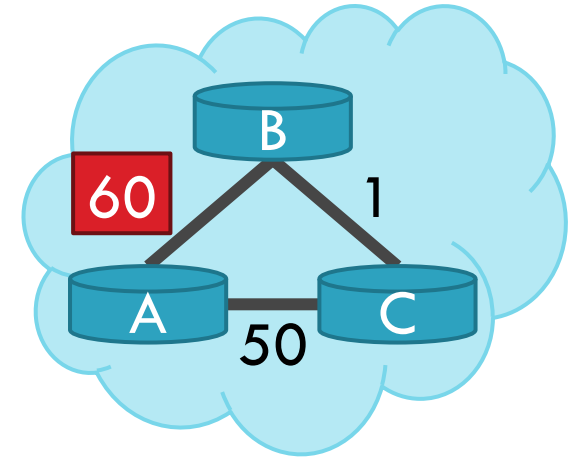
- A „jó hír” gyorsan terjed.
- A „rossz hír” lassan terjed.
- Azaz ciklusok keletkezhetnek.
- Lehetséges megoldás:
  - ▣ **„split horizon with poisoned reverse”**: negatív információt küld vissza arról a szomszédjának, amit tőle „tanult”. (RFC 1058)



# Split horizon with Poisoned Reverse

41

- 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



# Vége



□ Köszönöm a figyelmet!