

# Számítógépes Hálózatok

## **1. Előadás: Internet Architektúra**

Based on slides from

# Egy kis logisztika

2

## □ Előadás

- **Nappali:** *Hétfő 12:15-13:45*  
*Déli tömb, Bolyai terem*

## □ Előadó

- Dr. Laki Sándor
- Adjunktus, Információs Rendszerek Tanszék
- [lakis@inf.elte.hu](mailto:lakis@inf.elte.hu)
- Iroda: Déli tömb, 2.506

# Mi értelme ennek a tárgynak?

3

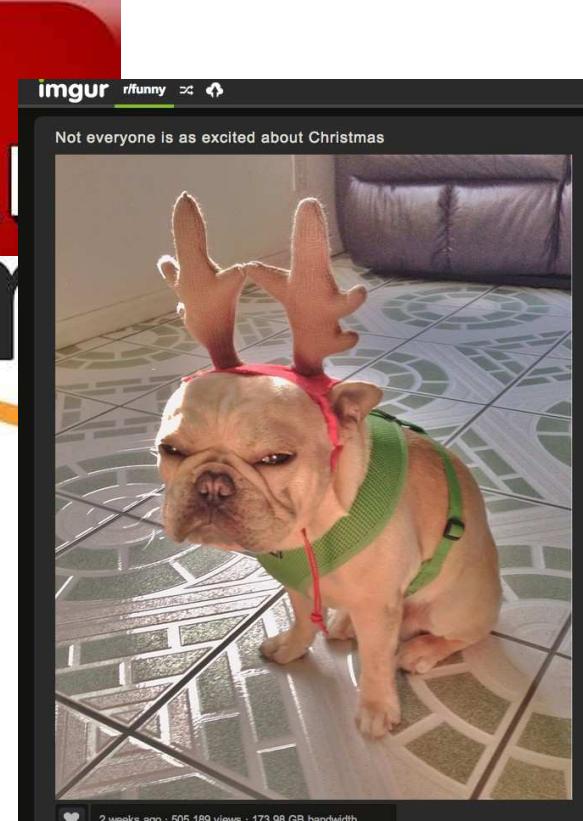
- Hányan nézték meg az e-mailjeiket, FB-ot, Twittert...
  - ma?
  - az elmúlt órában?
  - amióta elkezdtem beszélni?

# A számítógépes hálózatok mindenhol jelen vannak

4

- A hálózatok az élet minden részét érintik

- Web keresés
- Közösségi hálók
- Film nézés
- Termékek rendelése
- Időpocsékolás



# A számítógépes hálózatok mindenhol jelen vannak

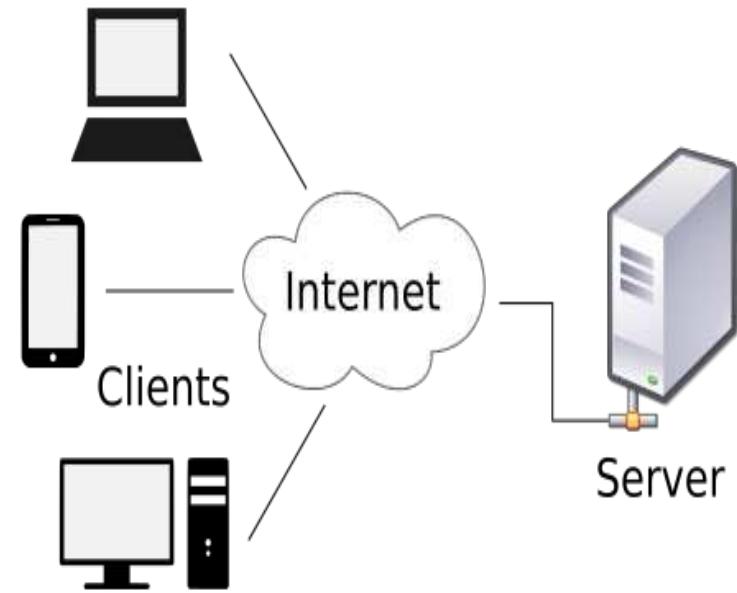
5

- A hálózatok az egyik legkritikusabb terület napjainkban
- Hálózatok nélkül nem lenne...
  - Big Data
  - Cloud
  - Apps or Mobile Computing

# Tantárgy célja 1/2

6

- „Hálózati” lehetséges témakörök:
  - 1. elosztott rendszerek,
  - 2. hálózati átvitel,
  - 3. kommunikáció.
- Hálózatokkal kapcsolatos kulcsproblémák:
  - 1. megbízhatóság,
  - 2. méretváltozás,
  - 3. erőforrások kihasználása,
  - 4. **biztonság**.



# Tantárgy célja 2/2

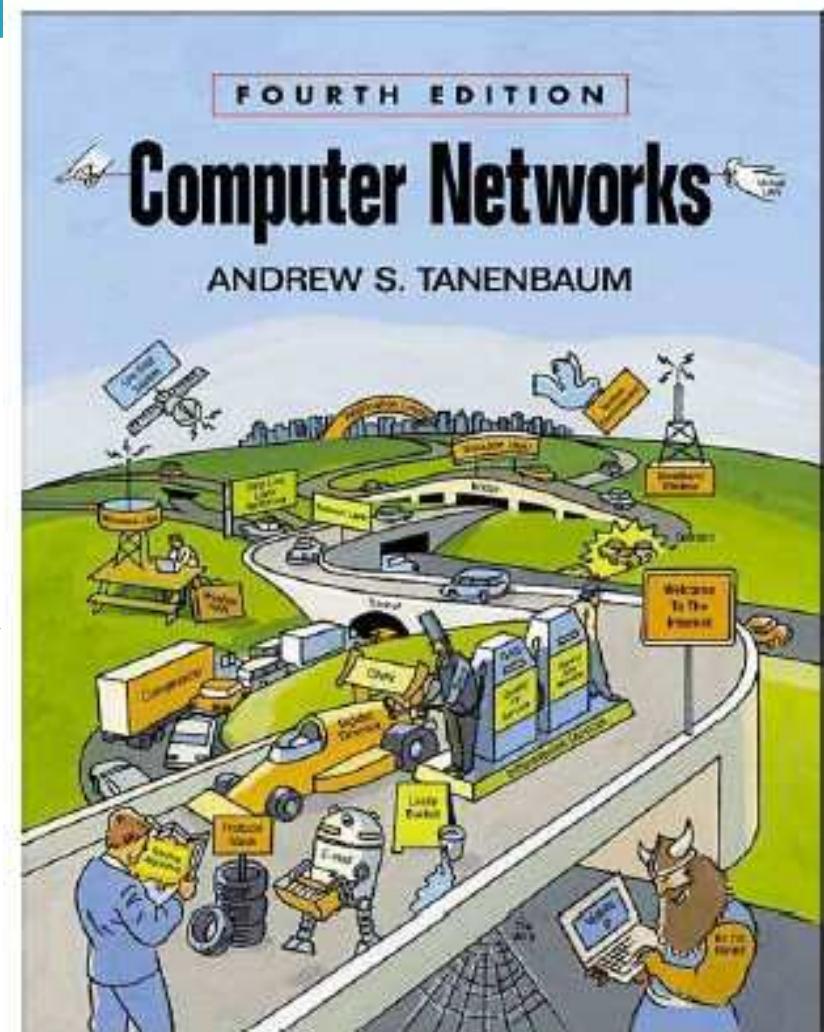
7

- Adathálózatok elveinek és gyakorlatának megismertetése.
  - útvonal választás algoritmusai, átviteli protokollok elvi kérdései,
  - hálózati alkalmazások tervezésének és implementációjának alapelvei,
  - ...
- Széles körben ismert hálózatok szolgáltatások háttérében történő folyamatok megismertetése
  - egy web alkalmazás megnyitásának folyamata a begépeléstől a kliens képernyőn történő megjelenítésig,
  - adatátvitel folyamata két eszköz között,

# Források

8

- A diák elérhetők:
  - <http://lakis.web.elte.hu>
- Könyv → → → → → → →



# Számonkérés - Gyakorlat

9

- 1) Gyakorlaton folyamatos számonkérés
  - A gyakorlati jegy 50%-át adják, és a vizsgához is alapul szolgálnak.
  - Két komponensből áll:
    - Teszt az óra elején (25%) – Előző heti előadás anyagából
      - Definíciókból, összefüggésekkel, képletekből.
    - Órai munka (25%)
      - Programozási feladatok, házi feladatok, stb.
- 2) Géptermi ZH a félév végén
  - A gyakorlati jegy másik 50%-át adja

# Számonkérés - Vizsgajegy

10

- A vizsga előfeltétele a **legalább elégséges** gyakorlati jegy.
- A vizsga **írásbeli**, azaz az egész féléves anyagra épülő elméleti és gyakorlati feladatokból összeállított kérdéssor kitöltését jelenti. A vizsga időtartama **50-60 perc**.
- Teszt részből és kifejtős részből áll.
- **A teszt rész esetén 50% minimum követelménnyel!**
- A féléves anyag a fóliákon is szereplő fogalmakat, összefüggéseket és a belőlük levonható következtetéseket jelenti.
- **Értékelés**
  - [85%, 100%] – jeles(5)
  - [75%, 85%) – jó(4)
  - [60%, 75%) – közepes(3)

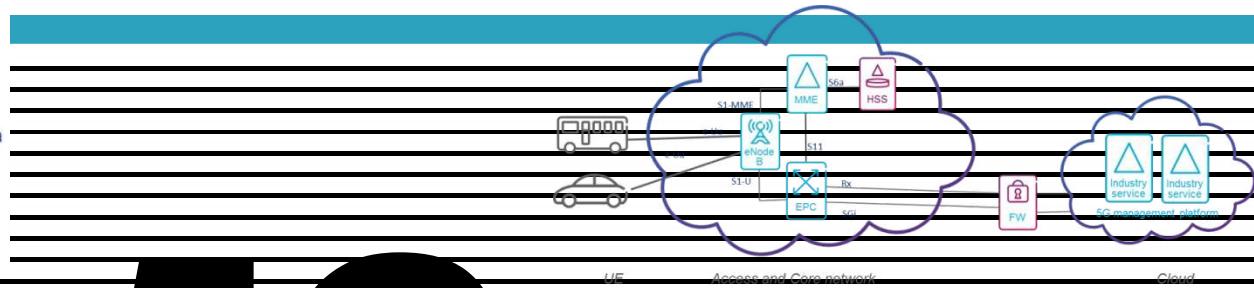
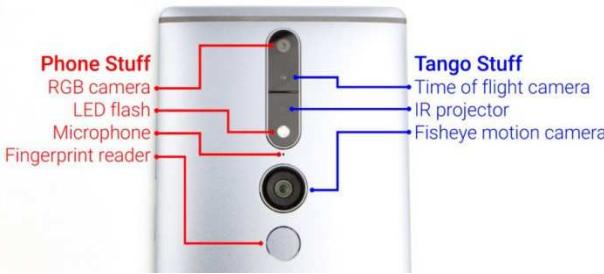
- Előadásra járni **kötelező** a tavalyi kari határozat alapján
- Az oktatónak **kötelező a jelenlét ellenőrzése**
- Katalógus minden előadáson.
- **4 igazolatlan hiányzás esetén a hallgató nem vizsgázhat**
- Ennek kivitelezése ???
  - <http://catalog.inf.elte.hu>

Bevezetés...

The

"Internet"





# 5G

/egyél részt Te is az 5G mobil hálózatok fejlesztésében!

üni partnerségi keretben támogatott kutatási projektek a Tudáskezelő rendszerek labor keretén belül ösztöndíj lehetőséggel

Témák és részletek: <http://networks.elte.hu>

Jelentkezni Laki Sándornál a [lakis@elite.hu](mailto:lakis@elite.hu) címen lehet!



# Alapfogalmak 1/6

# Alapfogalmak 2/6

16

## □ **Jel sávszélesség**

Jel feldolgozás esetén az egymást követő frekvenciák legnagyobb és legkisebb eleme közötti különbséget nevezik jel sávszélességnek. Tipikusan *Hertz*-ben mérik.

## □ **Hálózati sávszélesség**

Az adatátvittelehez elérhető vagy felhasznált kommunikációs erőforrás szabvány szolgáló mennyiségeg, amelyet bit/sec

8\*103            1 KB/s    egy kiló-bájt    ni.  
bit/sec

8\*106            1 MB/s    egy mega-  
bit/sec            bájt

8\*109            1 GB/s    egy giga-bájt  
bit/sec

8\*1012           1 TB/s    egy terra-  
bit/sec            bájt

8\*1015           1 PB/s    egy peta-bájt  
bit/sec

8\*210            1 KiB/s    egy kibi-bájt  
bit/sec

8\*220            1 MiB/s    egy mebi-  
bit/sec            bájt

8\*230            1 GiB/s    egy gibi-bájt  
bit/sec

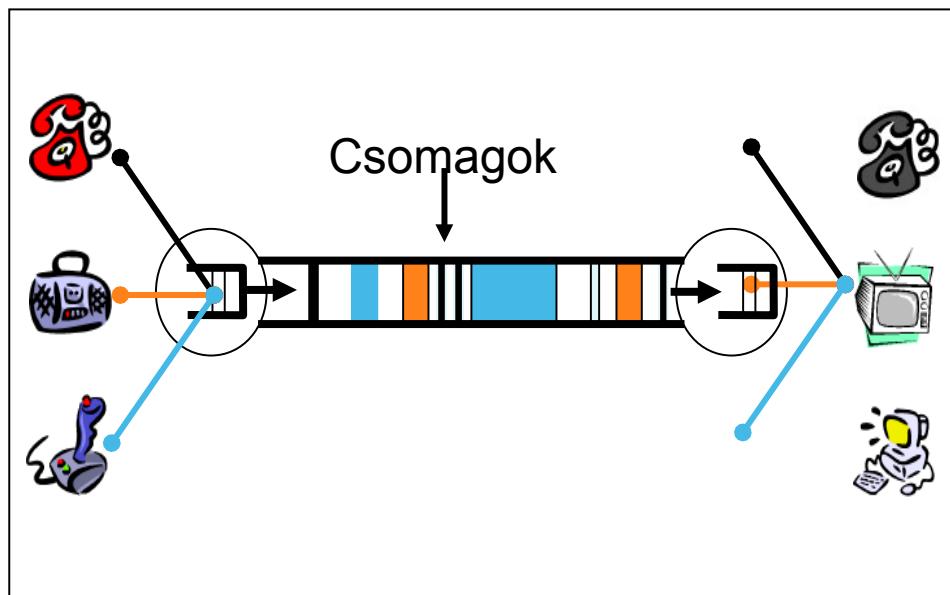
8\*240            1 TiB/s    egy tebi-bájt  
bit/sec

8\*250            1 PiB/s    egy pebi-  
bit/sec

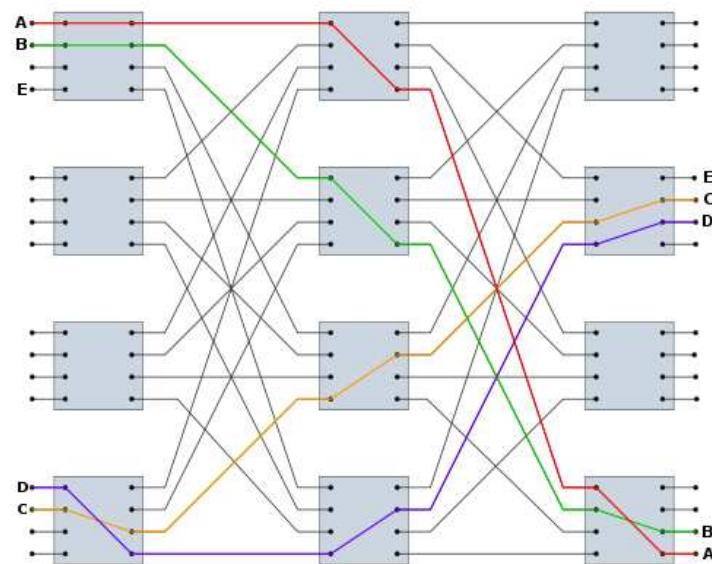
# Alapfogalmak 3/6

17

Csomagkapcsolt hálózat  
Pl. Internet



Áramkör kapcsolt hálózat  
Pl. vezetékes telefon



# Alapfogalmak 4/6

18

A hálózatokat lehet osztályozni a területi kiterjedésük alapján. (Forrás: Tar)

Processzorközi távolság	Processzorok által foglalt négyzetméter
1 m	négyzetméter
10 m	szoba
100 m	épület
1 km	kampusz
10 km	város
100 km	ország
1.000 km	kontinen
10.000 km	bolygó

Magánhálózat (angolul *Personal Area Network*)

Lokális hálózat (angolul *Local Area Network*)

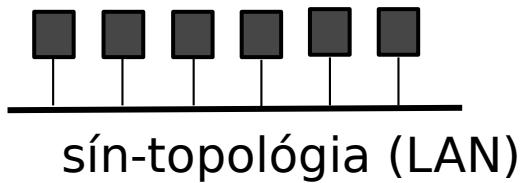
Városi hálózat (angolul *Metropolitan Area Network*)

Nagy kiterjedésű hálózat (angolul *Wide Area Network*)

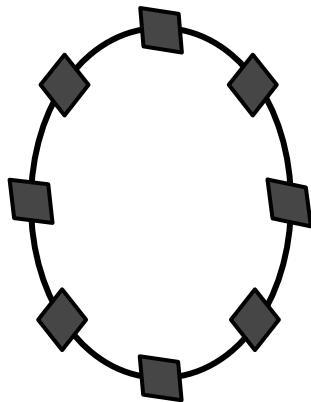
Internet

# Alapfogalmak 5/6 - példa topológiák

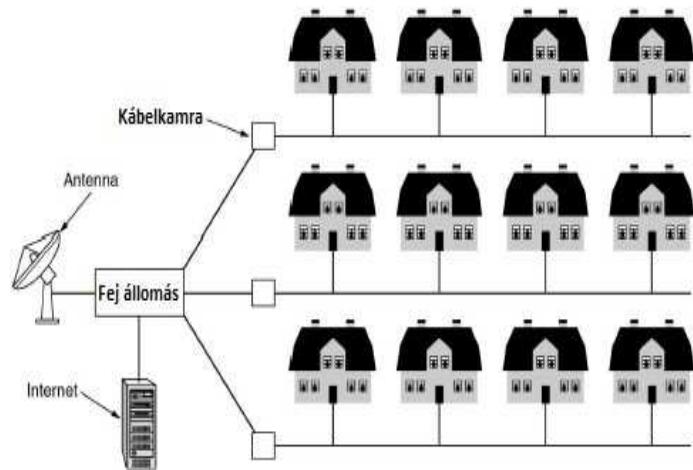
19



## sín-topológia (LAN)



## gyűrű-topológia (LAN)



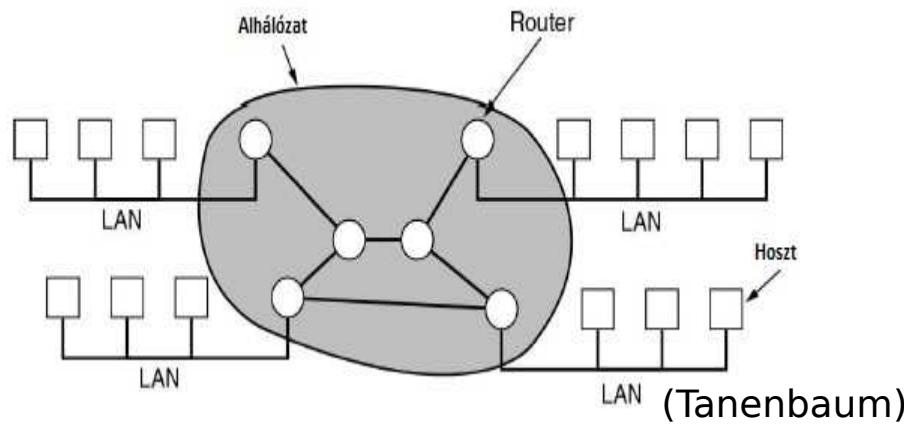
## TV-kábel alapú hálózat (MAN)

# Jelölések

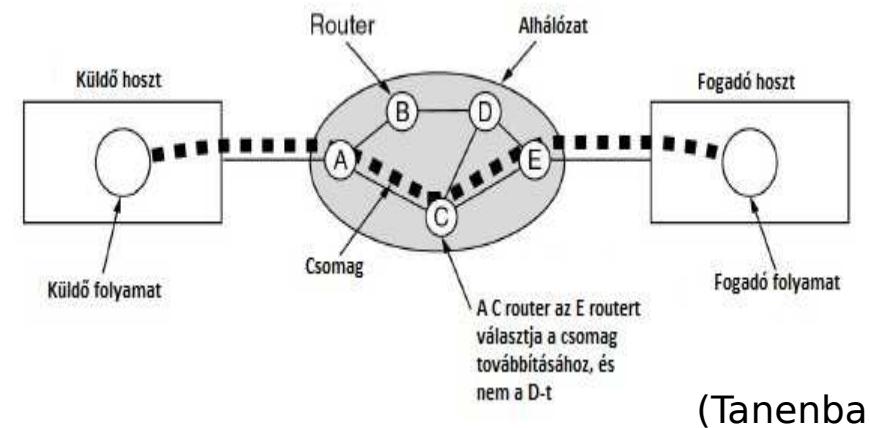
	Hoszt /
	állomás
	Kábel

# Alapfogalmak 6/6 – példa topológiák

20.



LAN-ok összekötése alhálózattal (WAN)

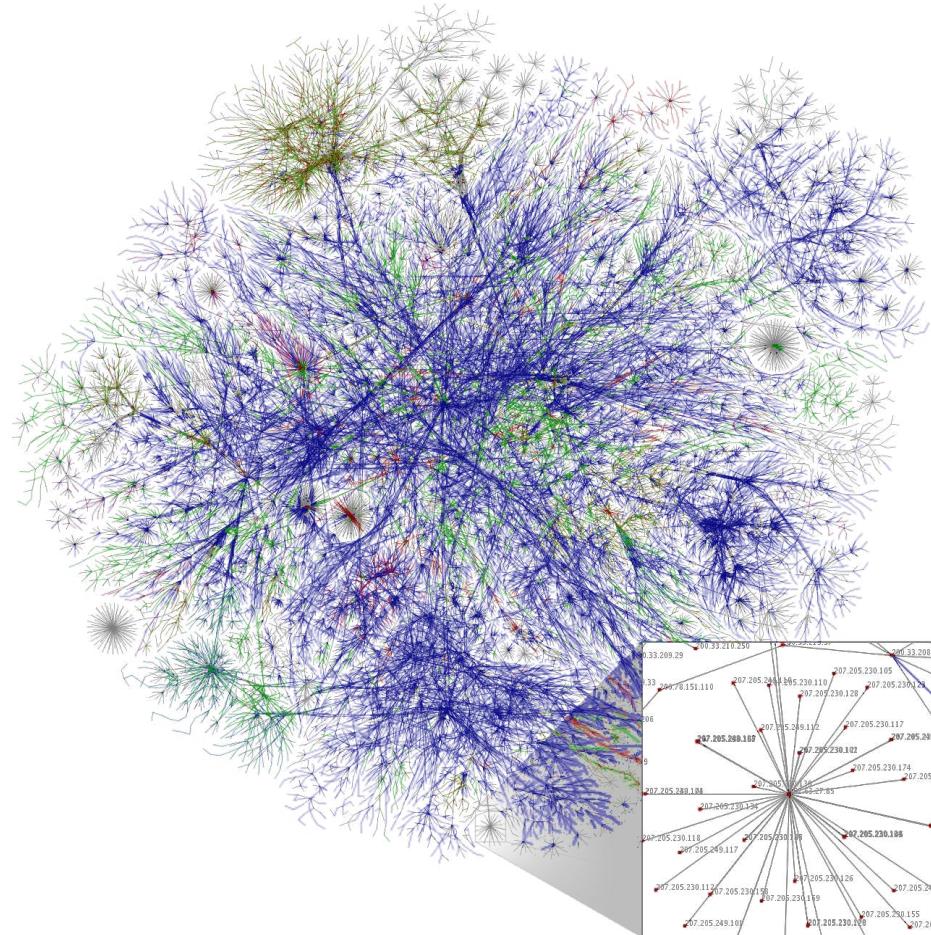


Adatfolyam szemléltetése egy WAN-on

# Mi az internet?

21

- Hálózatok hálózata
  - A világra kiterjedő nyitott WAN
  - Jellemzői
    - rendszerfüggetlenség;
    - nincs központi felügyelet;
    - építőelemei a LAN-ok;
    - globális;
    - olyan szolgáltatásokat nyújt, mint a **World Wide Web**, e-mail vagy fájlátvitel.



# Az Internet története 1/2

22

## 1957

- Sikeresen létesítettek kapcsolatot egy távoli számítógéphez.
- Szputnyik-1 műhold fellövése.

## 1958

- DARPA megalapítása.

## 1966

- ARPANET tervezésének kezdete.

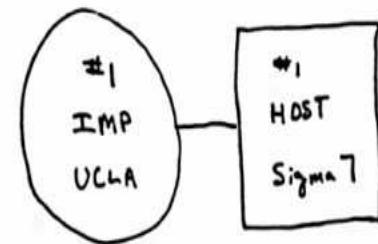
További történetileg fontos hálózatok:

- RAND – USA-ban katonai célokkal.
- NPL – Angliában kereskedelmi célokkal.
- CYCLADES – Franciaországban tudományos célokkal.

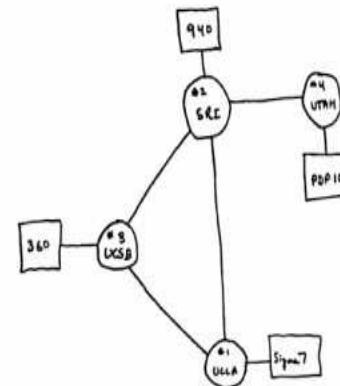
# Az Internet története 2/2 – főbb állomások

23

- 1961 július – „Packet Switching Theory” (*J.C.R. Licklider*)
- 1962 – A „Galactic Network” koncepciója (*J.C.R. Licklider*)
  - október – DARPA („**A**dvanced **R**esearch **P**rojects **A**gency”)
- 1965 – Az Internet első œse (*Thomas Merrill, Laurence G. Roberts*)
- 1967 – ARPANET tervezete
- 1969 – Az “ARPANET” első csomópontja
- 1990 – Az ARPANET megszűnése

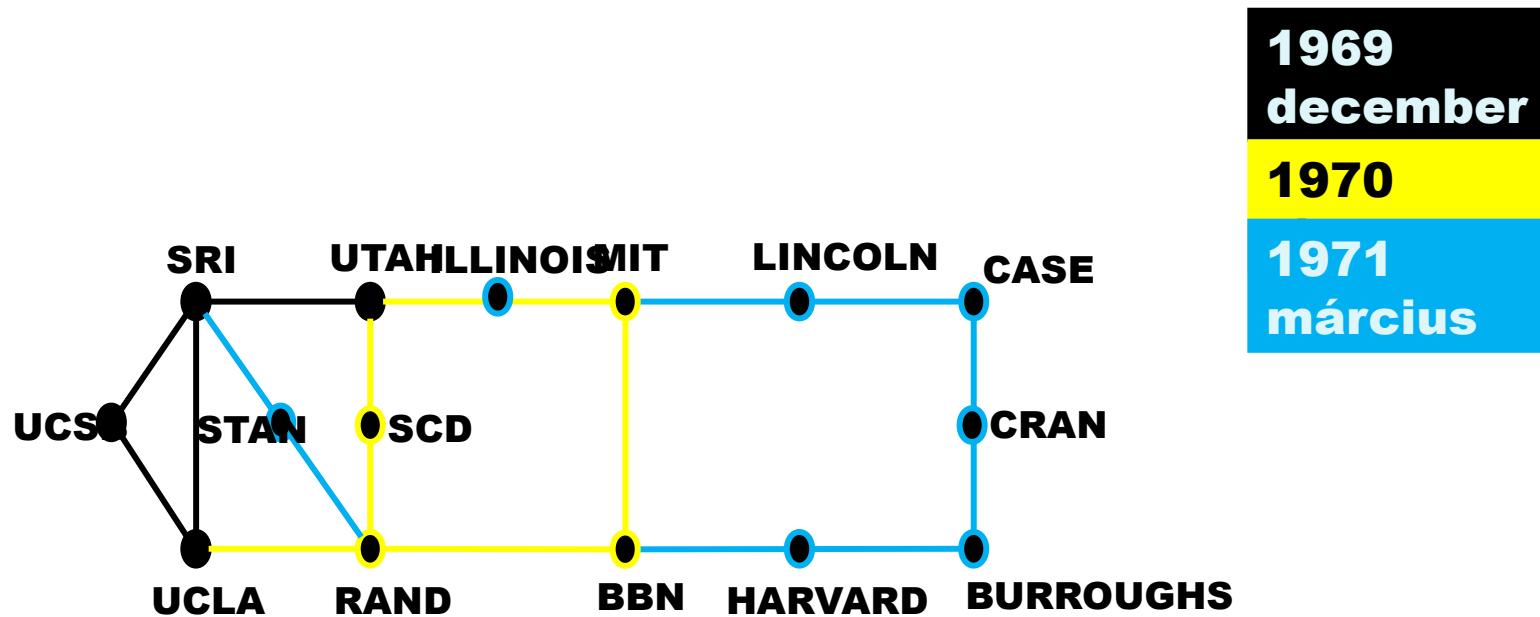


Az „ős-internet” eredeti diagramma



# ARPANET történeti ábra 1/3

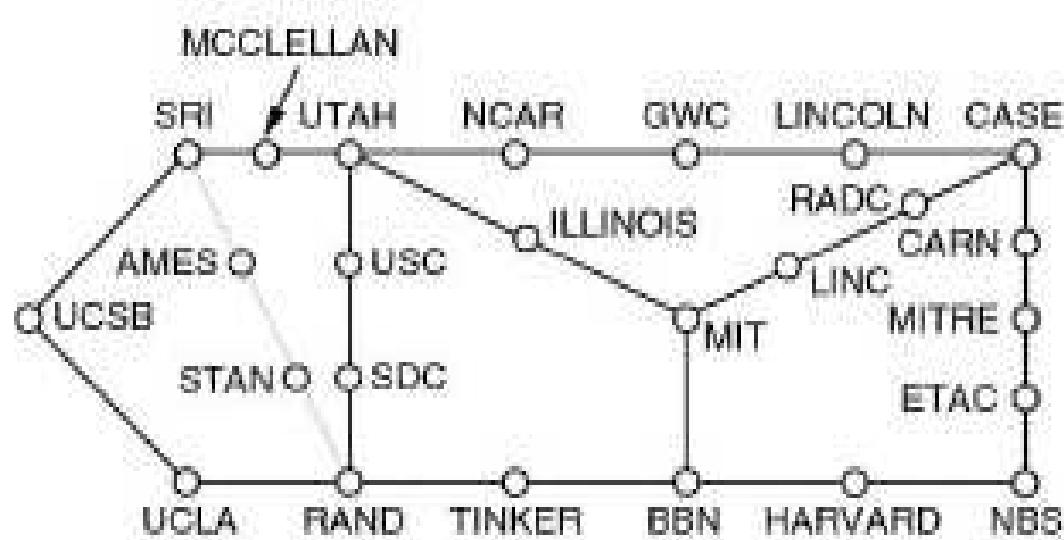
24



# ARPANET történeti ábra 2/3

25

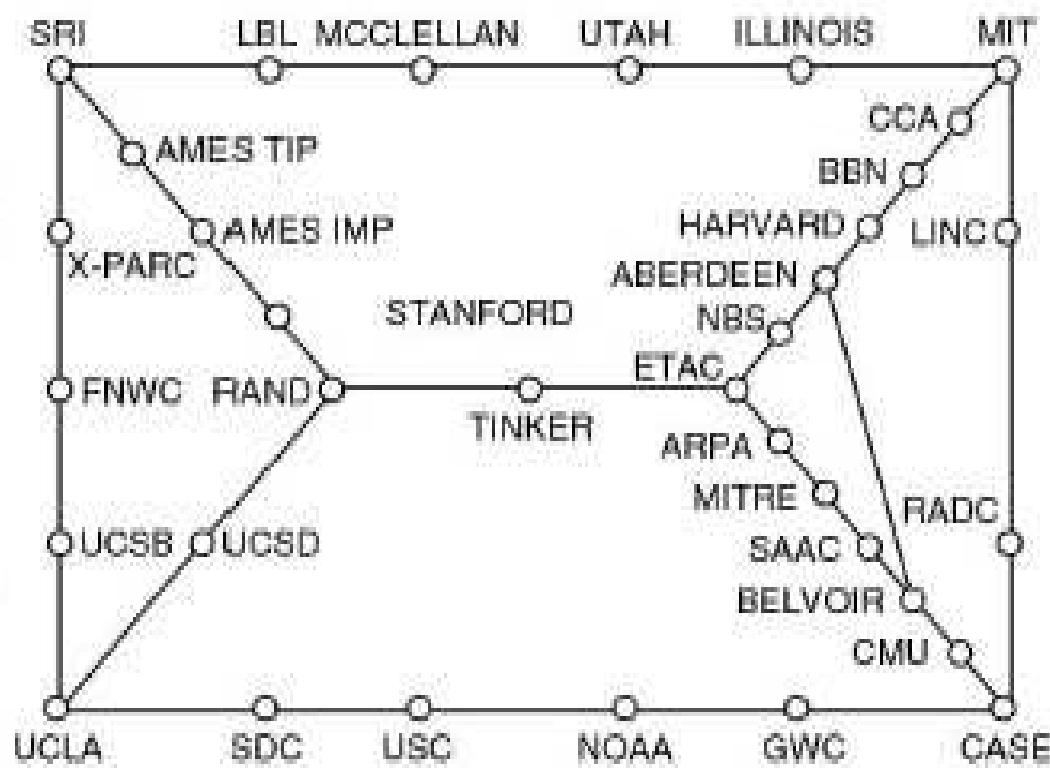
**1972 április**



# ARPANET történeti ábra 3/3

26

**1972 szeptember**



# Robert Kahn koncepciója – DARPA 1972

27

- minden (lokális) hálózat autonóm
  - önállóan dolgozik
  - nem kell elkülönítve konfigurálni a WAN-hoz
- Kommunikáció a „legjobb szándék” (angolul *best effort*) elv szerint
  - ha egy csomag nem éri el a célt, akkor törlődik
  - az alkalmazás újraküldi ilyen esetekben
- „*Black box*” megközelítés a kapcsolatokhoz
  - a *Black Box*-okat később *Gateway*-eknek és *Router*-eknek keresztelték át
  - csomaginformációk nem kerülnek megőrzésre

# Hálózati funkciók

28

- A hálózatok komponensei
  - Hálózati technológiák
    - Ethernet, Wifi, Bluetooth, Fiber Optic, Cable Modem, DSL
  - Hálózat típusok
    - Áramkör kapcsolt (Circuit switch), Csomag kapcsolt (packet switch)
    - Vezetékes (Wired), Vezeték nélküli (Wireless), Optikai, Műholdas
  - Alkalmazások
    - Email, Web (HTTP), FTP, BitTorrent, VoIP

# Probléma

29

Web



Email



Bittorrent



VoIP



- Ha ez lenne a valóság, akkor ez egy rémálom lenne
- Új alkalmazások és médiumok bevezetése költséges lenne



Ethernet



802.11



Bluetooth



Cellular

# További problémák

30

Bittorrent



Ethernet

Bittorrent



802.11

Az alkalmazási  
végpontok eltérő  
médiumot  
használnak

# Megoldás: használjunk kerülőútat

31

Web



Email



Bittorrent



VoIP



- O(1) munkával új app vagy

API

API

API

- médium közethetője
- esunáránkányszorítás úti



Ethernet



802.11



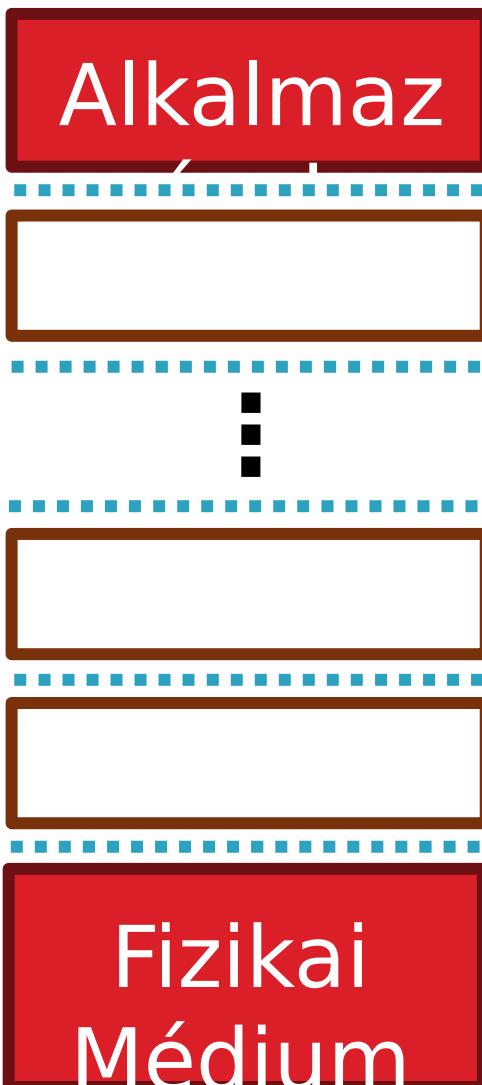
Bluetooth



Cellular

# Rétegelt Hálózati Architektúra

32



Modularitás

yered Network Stack)

- Modularitás
- A hálózati funkciókat szervezi egységekbe

Beburkolás (Encapsulation)

- Beburkolás (Encapsulation)
- Interfészek definiálják a réteg közi interakciókat
- A rétegek csak az alattuk levőkre épülnek

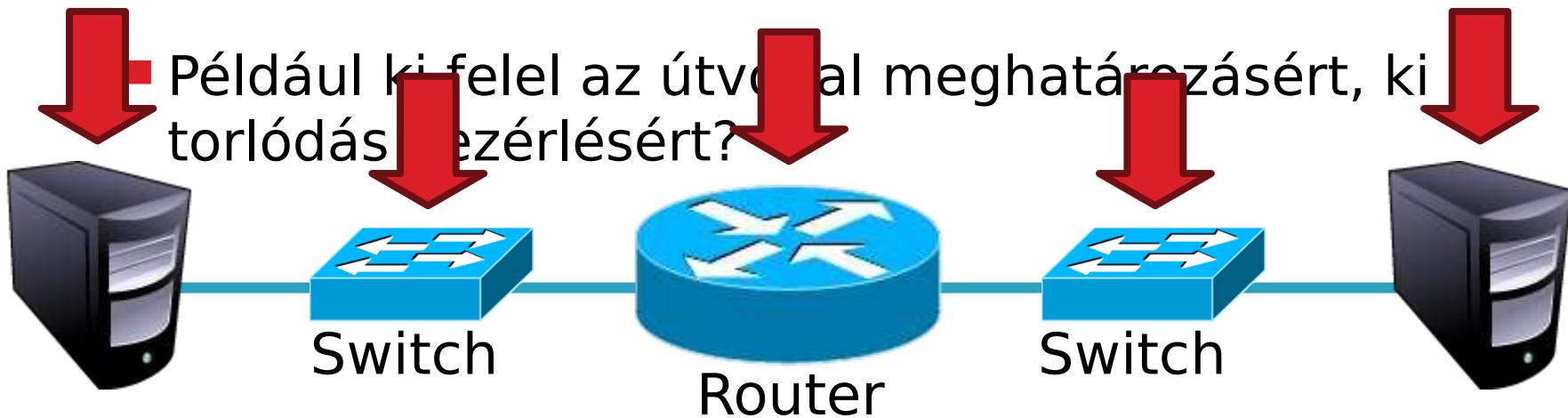
□ Rugalmasság

- Rugalmasság
- Kód újrafelhasználás a

# Fő kérdések

33

- Hogyan osszuk a funkciókat rétegekbe?
  - Útvonal meghatározás Biztonság
  - Torlódás vezérlés Fairség
  - Hiba ellenőrzés ...
- Hogyan osszuk el ezen funkciókat a hálózati eszközök között?



# Hálózatok modelljei

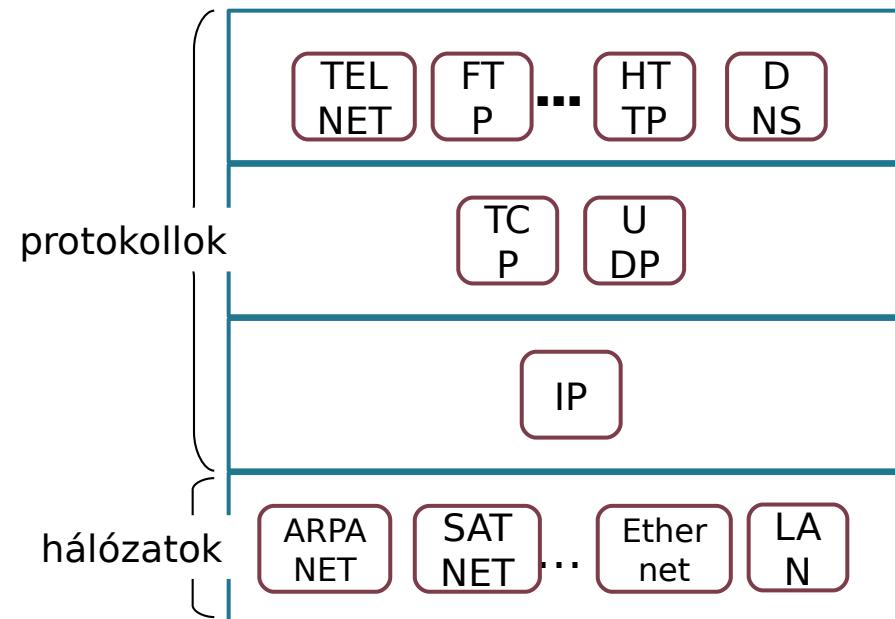
34

- Internet rétegmodelljei
  - TCP/IP modell: 4 réteget különböztet meg. 1982 márciusában az amerikai hadászati célú számítógépes hálózatok standardja lett. 1985-től népszerűsítették kereskedelmi felhasználásra.  
*(Interop)*
  - Hibrid TCP/IP modell: 5 réteget különböztet meg (*Tanenbaum, Stallings, Kurose, Forouzan*)
- Nyílt rendszerek hálózatának standard modellje
  - *Open System Interconnection Reference Model*: Röviden OSI referencia modell, amely egy 7-rétegű standard, koncepcionális modellt definiál kommunikációs hálózatok belső funkcionálitásaihoz.  
*(ISO/IEC 7498-1)*

# TCP/IP modell (RFC 1122)

35

<b>Alkalmazási réteg</b> (angolul <i>Application layer</i> )
<b>Szállítói réteg</b> (angolul <i>Transport layer</i> )
<b>Hálózati réteg</b> (angolul <i>Internet layer</i> )
<b>Kapcsolati réteg</b> (angolul <i>Link layer</i> )



# TCP/IP modell rétegei („bottom-up”)

36

- Kapcsolati réteg / Host-to-network or Link layer
  - nem specifikált
  - a LAN-tól függ
- Internet réteg / Internet or Network layer
  - speciális csomagformátum
  - útvonal meghatározás (routing)
  - csomag továbbítás (angolul *packet forwarding*)
- Szállítói réteg / Transport layer
  - **T**ransport **C**ontrol **P**rotocol
    - megbízható, kétirányú bájt-folyam átviteli szolgáltatás
    - szegmentálás, folyamfelügyelet, multiplexálás
  - **U**ser **D**atagram **P**rotocol
    - nem megbízható átviteli szolgáltatás

# ISO OSI modell

37

OSI: Open Systems Interconnect Model

Hoszt

Router/  
Switch

Hoszt

Alkalmaz

Megelen

Ülés

Szállítói

Hálózati

Adatkapc

Fizikai

Az első 2 réteget  
A rétegek peer-to-peer  
egymással  
komunikálnak

Application  
Presentation

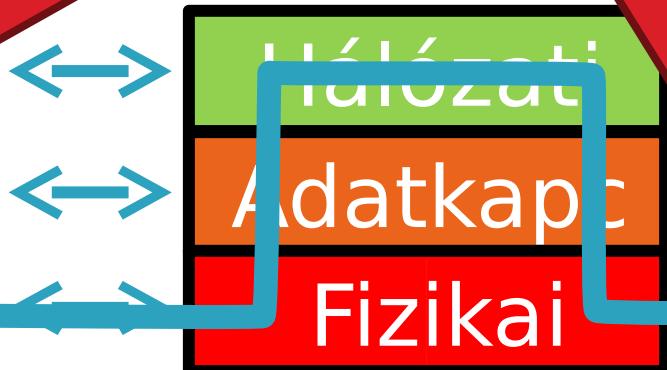
Session

Transport

Network

Adatkapcsol

Fizikai



Az első 2 réteget  
A rétegek peer-to-peer  
egymással  
komunikálnak

# Rétegek jellemzése

38

Alkalmaz

Megjelen

Ülés

Szállítói

Hálózati

Adatkapc

Fizikai

- Szolgáltatás
  - Mit csinál az adott réteg?
- Interfész
  - Hogyan férhetünk hozzá a réteghez?
- Protokoll
  - Hogyan implementáljuk a réteget?

# Fizikai réteg

□ Szolgáltatás

■ Információt visz át két fizikailag összekötött eszköz között

■ definiálja az eszköz és a fizikai átviteli közeg kapcsolatát

□ Interfész

■ Specifikálja egy bit átvitelét

□ Protokoll

■ Egy bit kódolásának sémája

■ Feszültség szintek

■ Jelek időzítése

Alkalmaz

Megjelen

Ülés

Szállítói

Hálózati

Adatkapc

Fizikai

# Adatkapcsolati réteg

- Szolgáltatás
- Adatok keretekre tördelése: határok a csomagok között

40

Alkalmaz  
Megjelen  
Ülés  
Szállítói  
Hálózati  
Adatkapc  
Fizikai

- Közeghozzáférés vezérlés (MAC)
- Per-hop megbízhatóság és folyamvezérlés
- Interfész
  - Keret küldése két közös médiumra kötött eszköz között
- Protokoll
  - Fizikai címzés (pl. MAC)

- Szolgáltatás

# Hálózati réteg

▫ Csomagtovábbítás

▫ Útvonalválasztás

▫ Csomag fragmentálás kezelése

▫ Csomag ütemezés

▫ Puffer kezelés

## ▫ Interfész

▫ Csomag küldése egy adott végpontnak

## ▫ Protokoll

▫ Globálisan egyedi címeket definiálása

Alkalmaz

Megjelen

Ülés

Szállítói

Hálózati

Adatkapc

Fizikai

# Szállítói réteg

Szolgáltatás

42

## Multiplexálás/demultiplexálás

Alkalmaz

Megjelen

Ülés

Szállítói

Hálózati

Adatkapc

Fizikai

- Torlódásvezérlés
- Megbízható, sorrendhelyes továbbítás
- Interfész
  - Üzenet küldése egy célállomásnak
- Protokoll
  - Port szám
  - Megbízhatóság/Hiba javítás
  - Felhasználószint

# Ülés v. Munkamenet réteg

43

Alkalmaz  
Megjelen  
Ülés  
Szállítói  
Hálózati  
Adatkapc  
Fizikai

- Szolgáltatás
  - kapcsolat menedzsment: felépítés, fenntarás és bontás
  - munkamenet típusának meghatározása
  - szinkronizációs pont menedzsment (checkpoint)
- Interfész
  - Attól függ...
- Protokoll
  - Token menedzsment
  - Szinkronizációs checkpoints

# Megjelenítési réteg

## □ Szolgáltatás

44

Alkalmaz  
Megjelen  
Ülés  
Szállítói  
Hálózati  
Adatkapc  
Fizikai

- Adatkonverzió különböző reprezentációk között
  - Pl. big endian to little endian
  - Pl. Ascii to Unicode
- Interfész
  - Attól függ...
- Protokoll
  - Adatformátumokat definiál
  - Transzformációs szabályokat alkalmaz

# Alkalmazási réteg

45



- Szolgáltatás
  - Bármil...
- Interfész
  - Bármil...
- Protokoll
  - Bármil...
- Példa: kapcsold be a mobilod és nézd meg milyen appok vannak rajta...

# Tananyag címszavakban

46

1. Hálózatok leírásához használt legfontosabb referencia modellek
2. Fizikai réteg áttekintése
3. Adatkapcsolati réteg
  - a) „Logical Link Control” alréteg
  - b) „Medium Access Control” alréteg
4. Hálózati réteg
5. Socket programozási alapok
6. Szállítói réteg
7. Alkalmazási réteg
8. Kis kitekintés – Software defined networks, OpenFlow, P4, 5G

Köszönöm a figyelmet!

# Számítógépes Hálózatok

## **2. Előadás: Bevezetés**

+

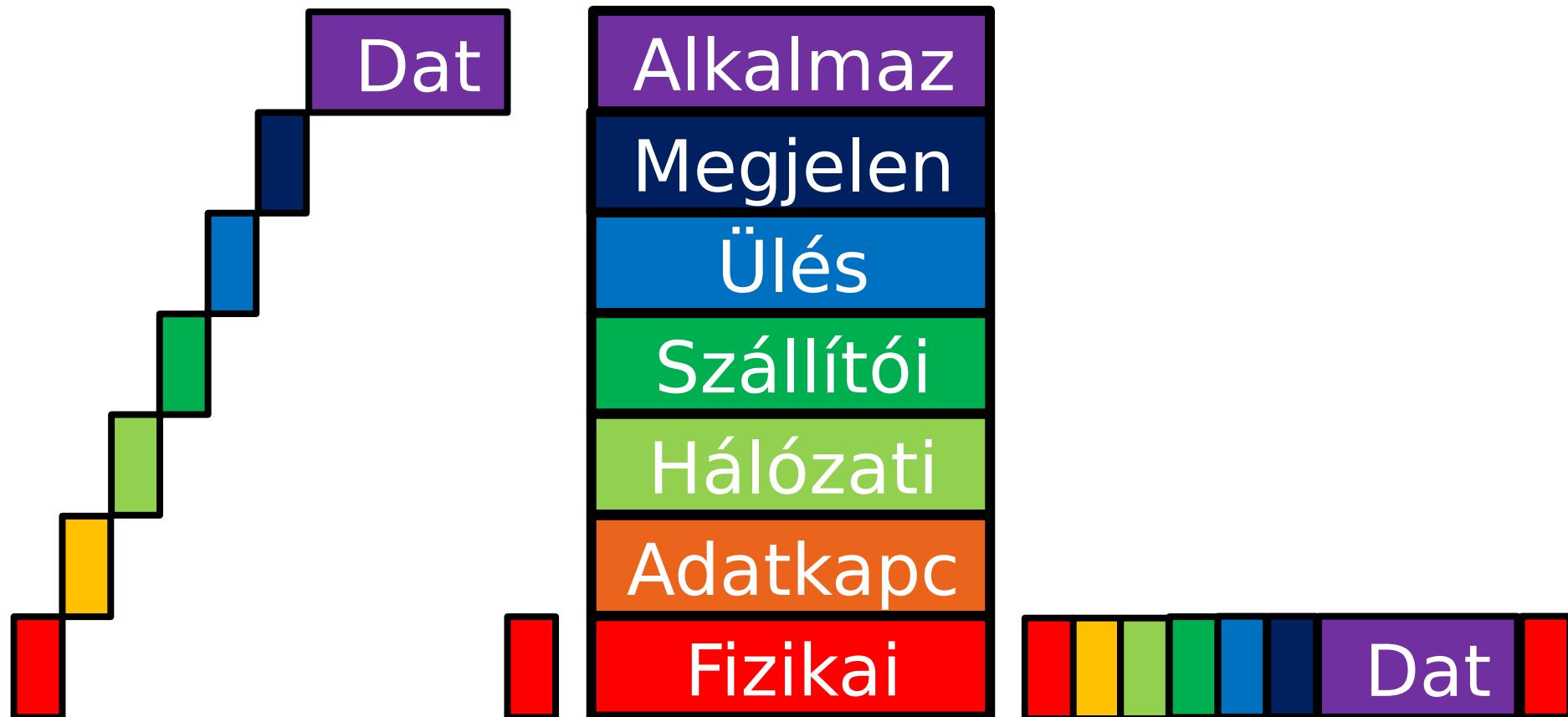
### **Fizikai réteg**

Based on slides from

# Beburkolás / enkapszuláció

2

Az adat útja a rétegeken keresztül



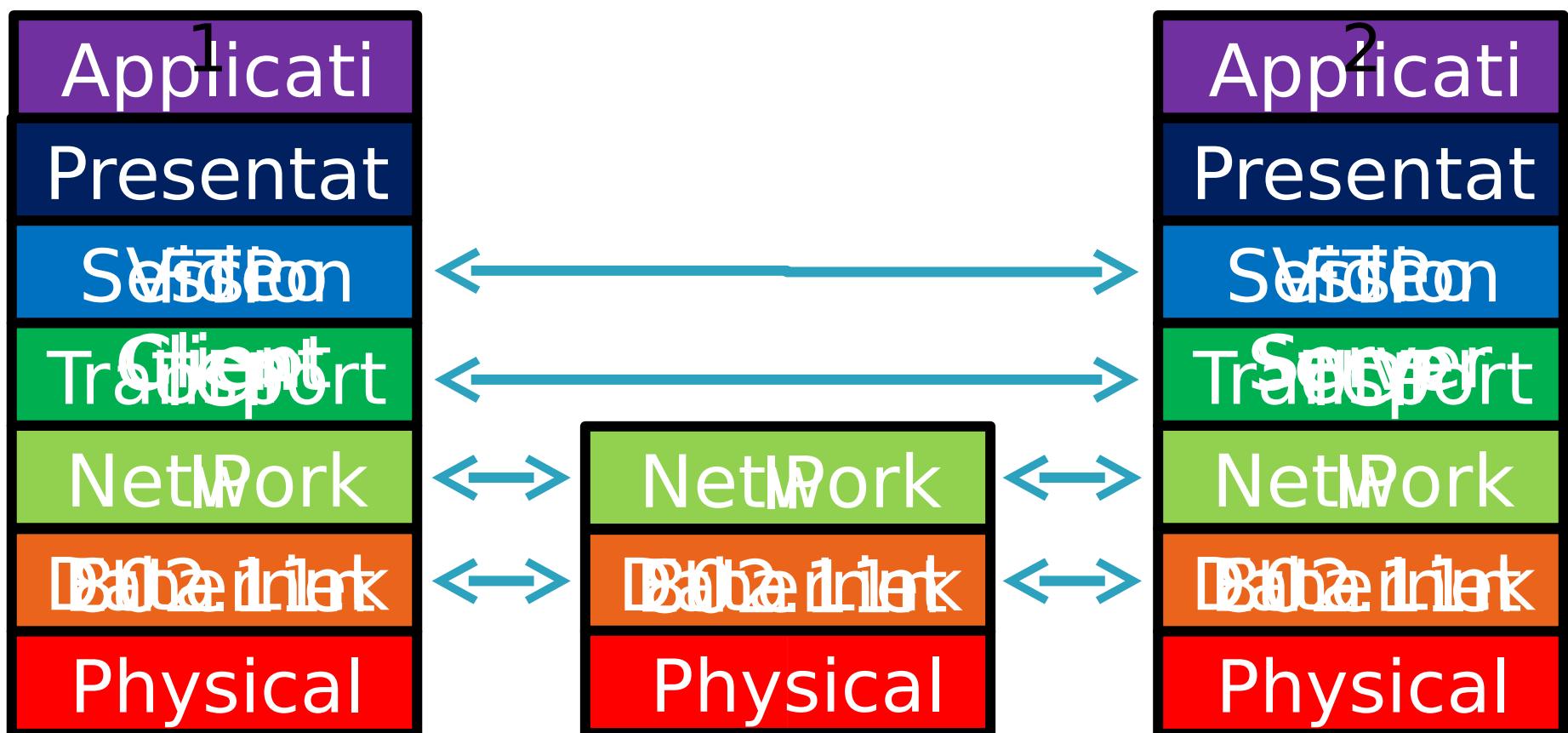
# Analógia

3



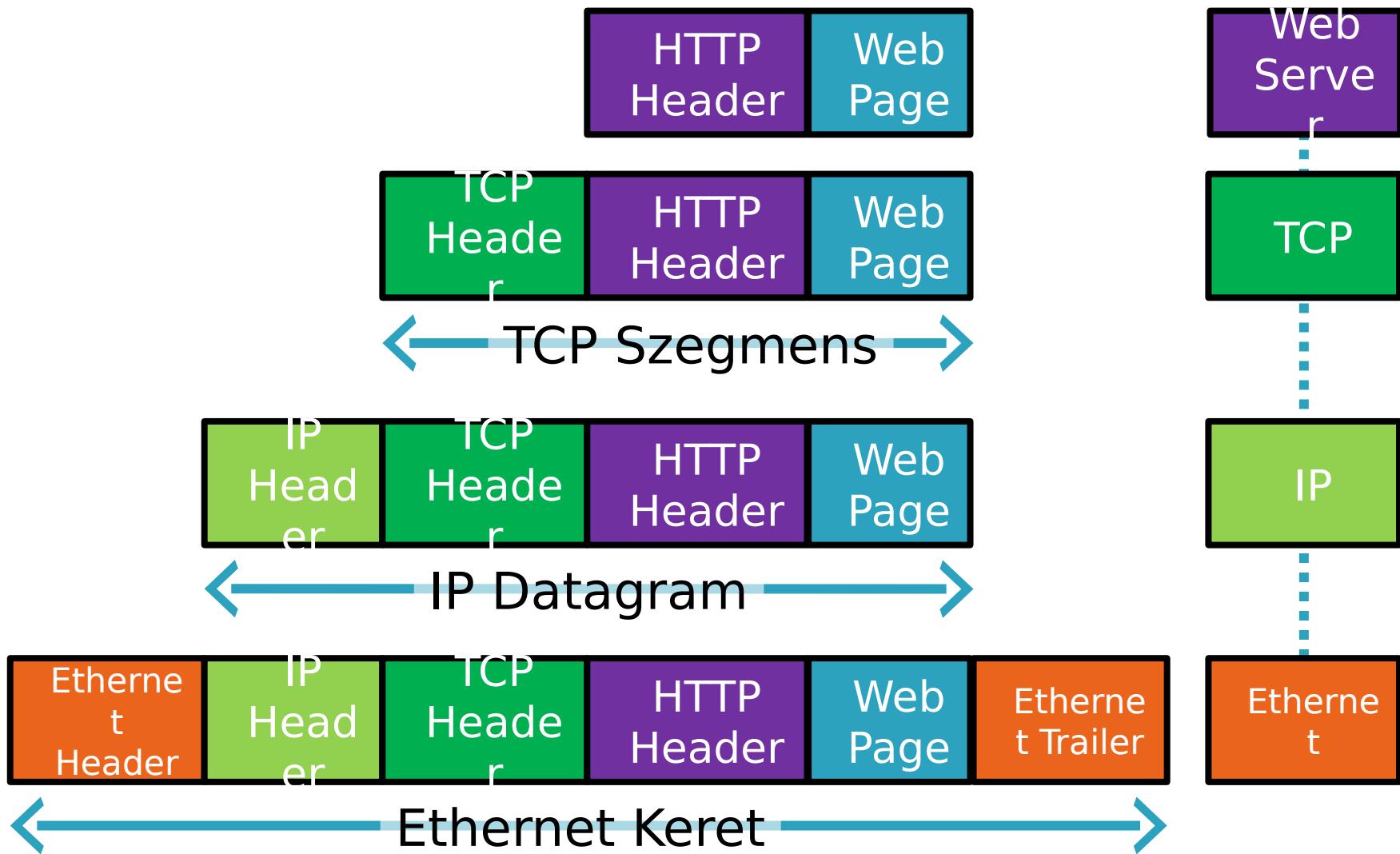
# Hálózati rétegek a gyakorlatban

## Network stack/Protocol stack



# Beburkolás – Internet példa

5



# Internet homokóra

6

- Az Internet rétegnek hála, minden hálózat képes együttműködni
- minden alkalmazás működik minden hálózaton
- Ezen réteg felett és alatt lehetőségek nyílnak újabb fejlesztések

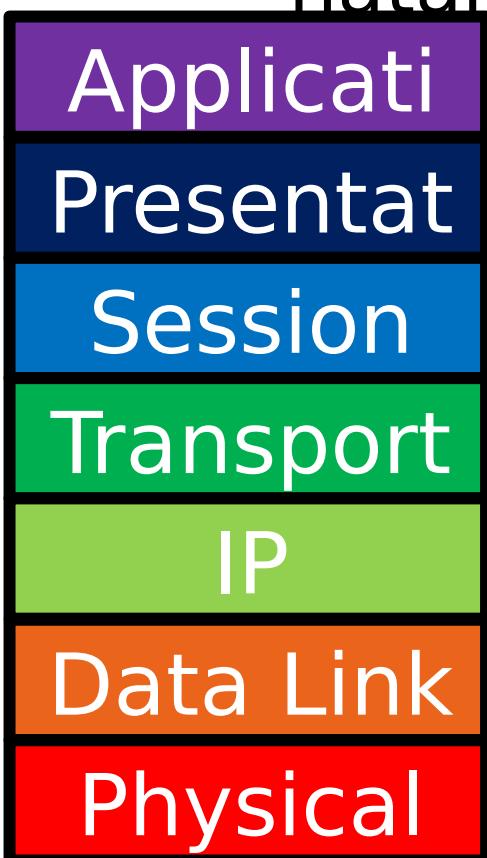
Gondoljunk az IPv6 bevezetésének

Fiber, Coax, Twisted Pair, Radio, ...  
nehéz

# Merőleges síkok

7

**Control plane/Vezérlési sík:** Hogyan határozzuk meg az **Internetes útvonalakat?**



BGP

RIP

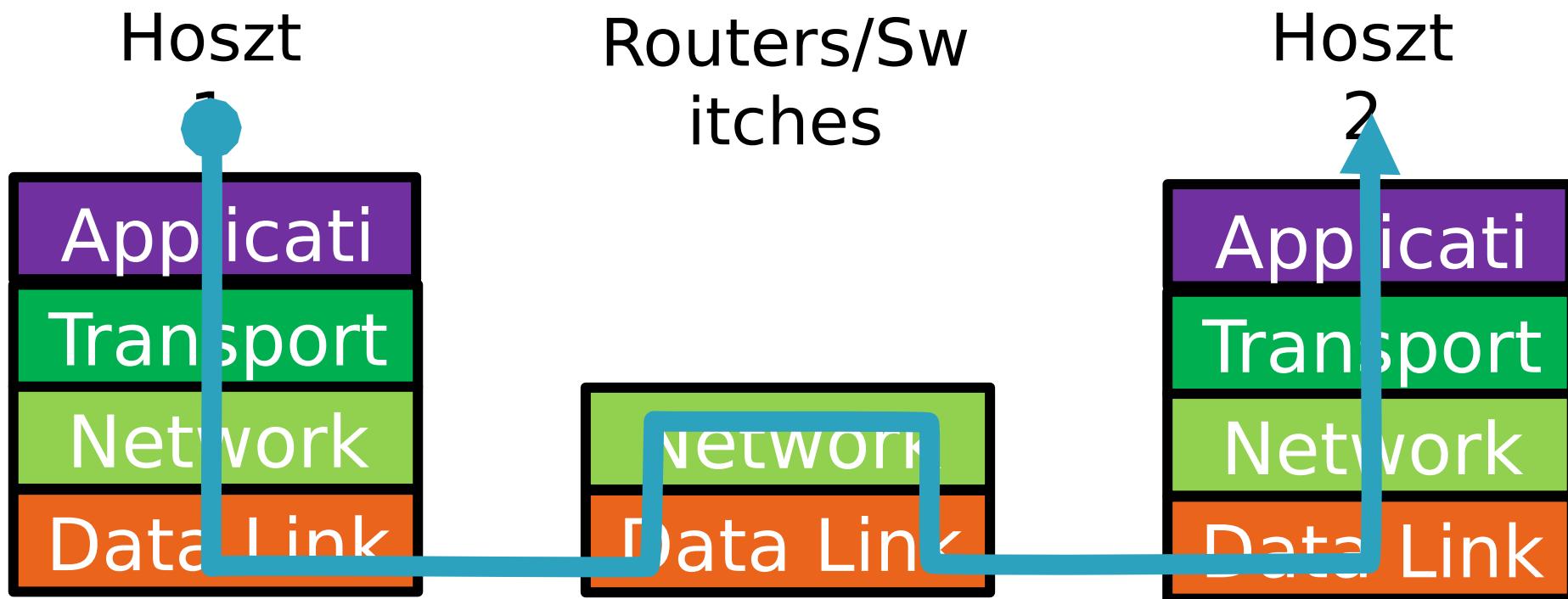
OSPF

Control Plane

# Merőleges síkok

8

**Data plane/Adat sík:** Hogyan **továbbítjuk az adatot** egy útvonal mentén?



# Valóság

9

- Az absztrakciós rétegek jól alkalmazhatók
- Vajon minden működik?

Nem.



Tűzfalak

- Alkalmazási réteg fejléceit is vizsgálhatja



Proxyk

- Alkalmazási végpontot szimulál a hálózatban

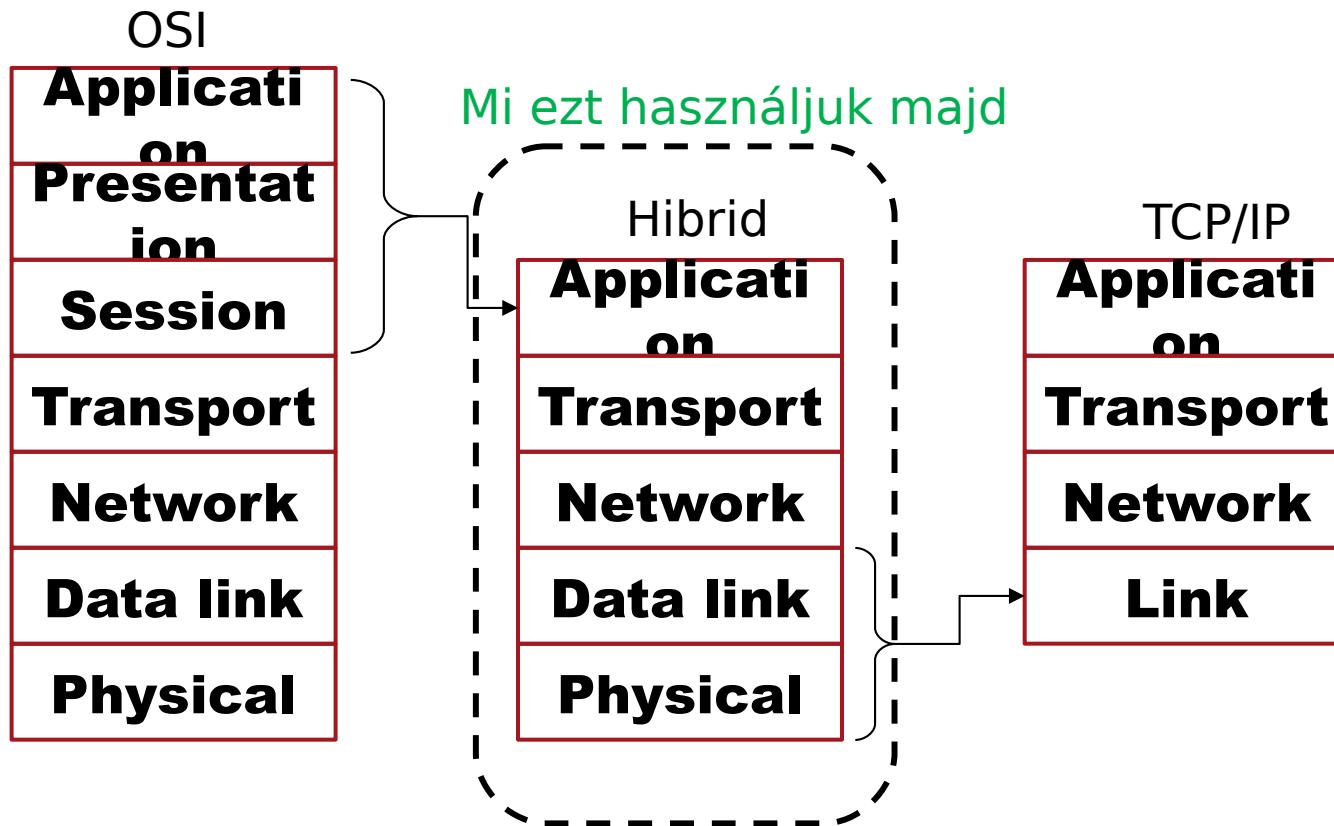


NATs

- Megtöri a végpont-végpont elérhetőséget a hálózathoz

# Konklúzió

10



# Tananyag címszavakban

11

1. Hálózatok leírásához használt legfontosabb referencia modellek
2. Fizikai réteg áttekintése
3. Adatkapcsolati réteg
  - a) „Logical Link Control” alréteg
  - b) „Medium Access Control” alréteg
4. Hálózati réteg
5. Socket programozási alapok
6. Szállítói réteg
7. Alkalmazási réteg
8. Kis kitekintés – Software defined networks, OpenFlow, P4, 5G

# Fizikai réteg

□ Szolgáltatás

□ Információt visz át két fizikailag összekötött eszköz között

□ definiálja az eszköz és a fizikai átviteli közeg kapcsolatát

□ Interfész

□ Specifikálja egy bit átvitelét

□ Protokoll

□ Egy bit kódolásának sémája

□ Feszültség szintek

□ Jelek időzítése

Alkalmaz

Megjelen

Ülés

Szállítói

Hálózati

Adatkapc

Fizikai



# Alapfogalmak

# Kihívások

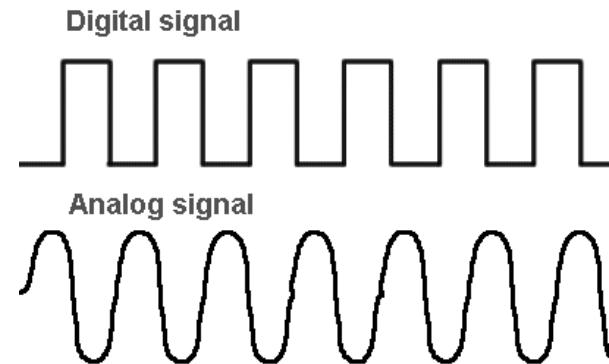
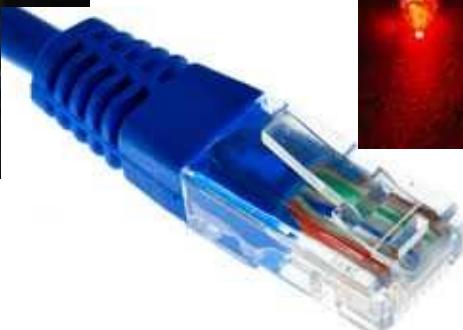
14

- Digitális számítógépek

- Nullák és egyesek

- Analóg világ

- Amplitúdék és frekvenciák



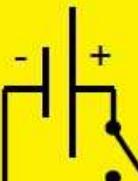
# Egyszerű adatátvitel

- 1-es bit: feszültség vagy áramerősség
- 0-ás bit: nincs feszültség

## Converting bits to voltage

Bit 1: The switch is turned on.

Bit 0: It is turned off.



## Converting voltage to bits

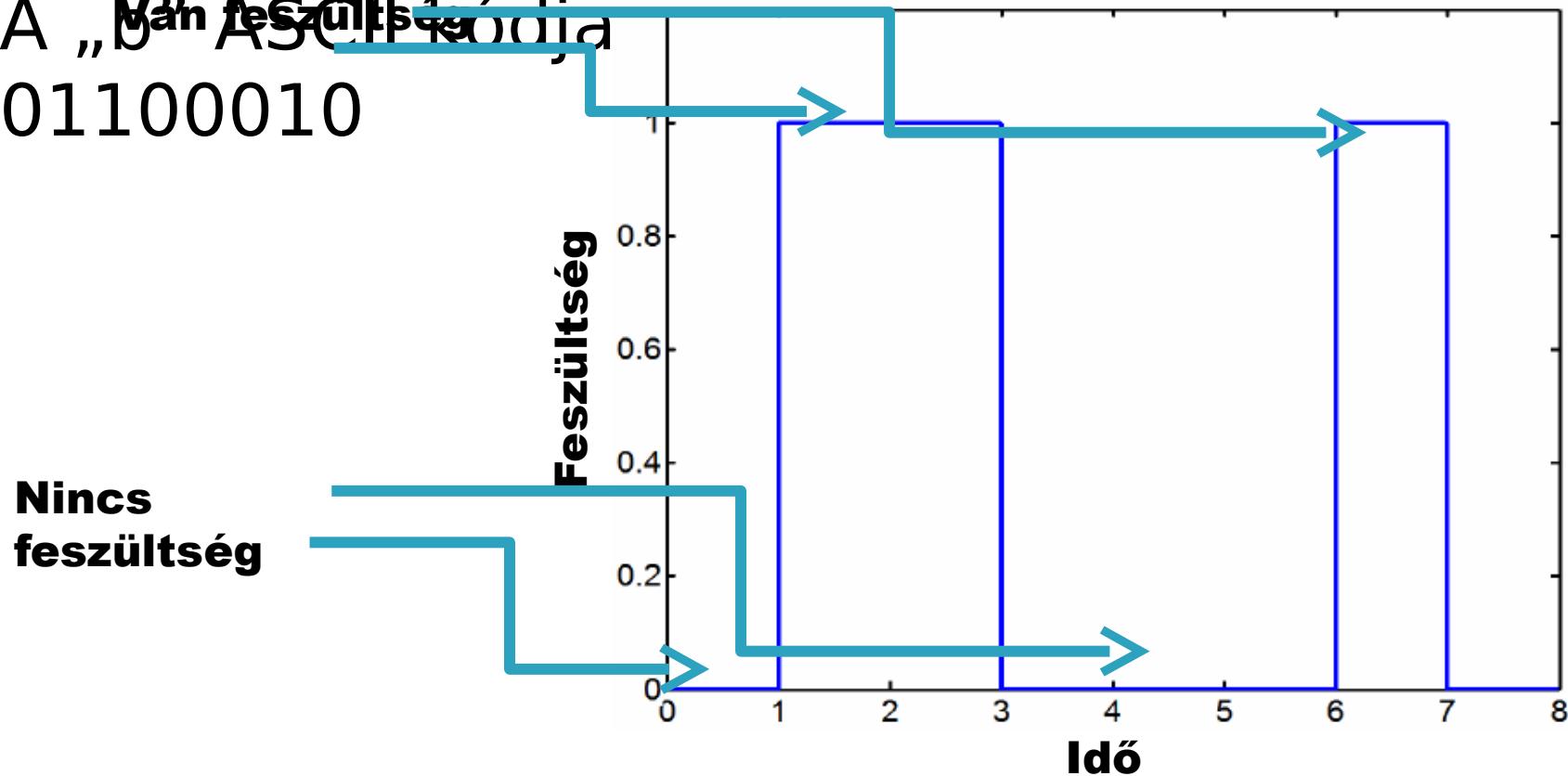
Voltage: Bit 1

No voltage: Bit 0



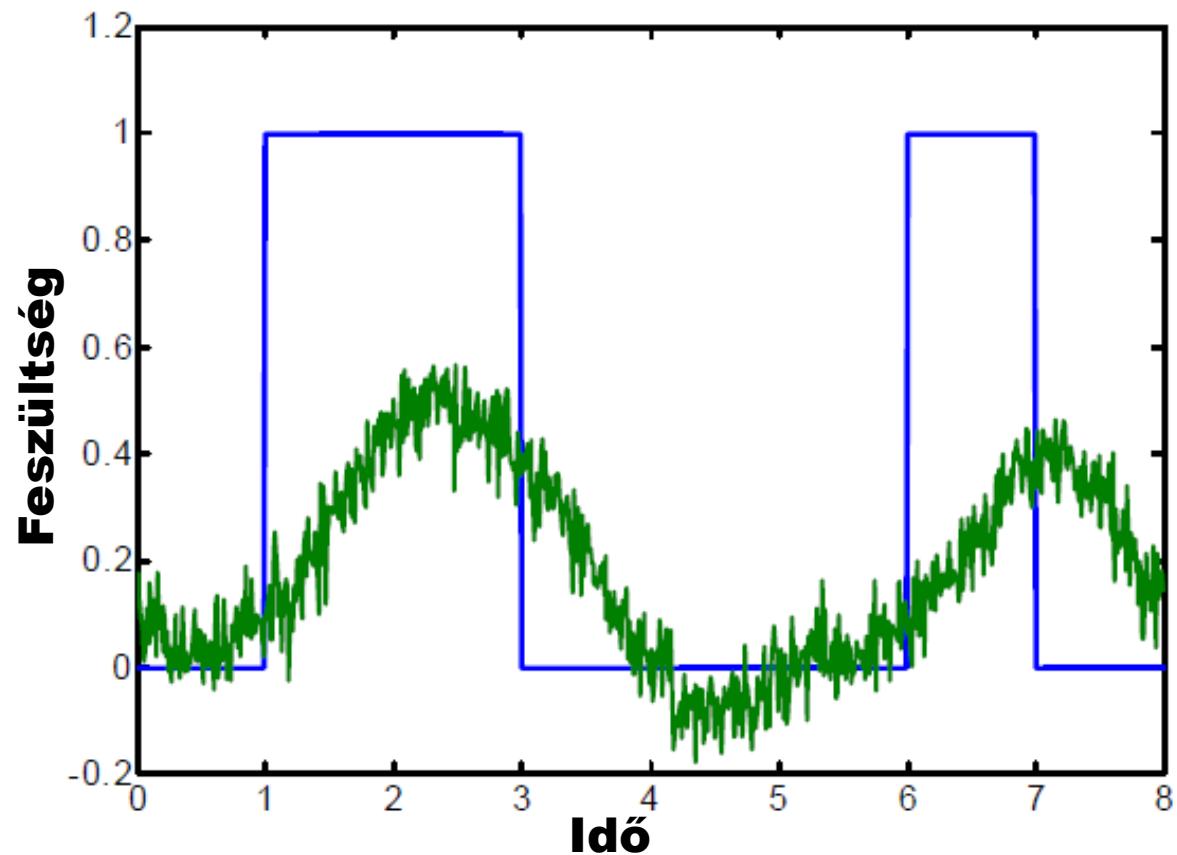
# A „b” karakter átvitele

- Egynél több bit szükséges a „b” karakter átviteléhez
- A „~~Nincs feszültség~~” kódja  
01100010



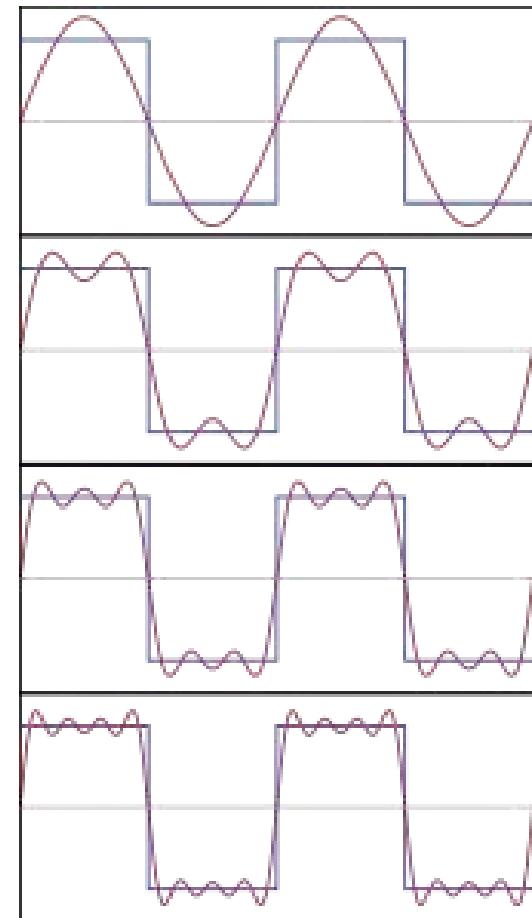
# A „b” karakter átvitele

- Túl rossz vétel



# Elméleti alapok - adatátvitel

18

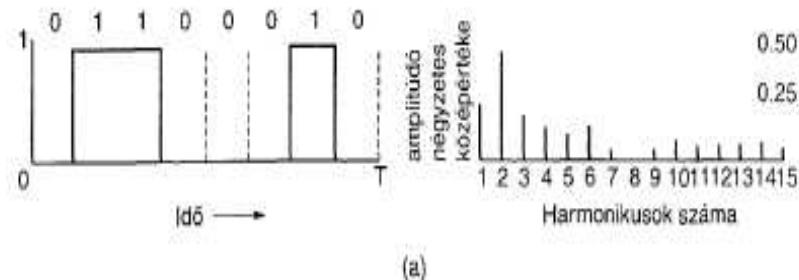


# Elméleti alapok - adatátvitel

# Elméleti alapok - adatátvitel

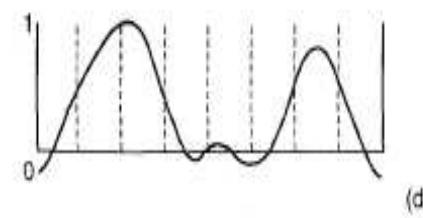
# Elméleti alapok - adatátvitel

21

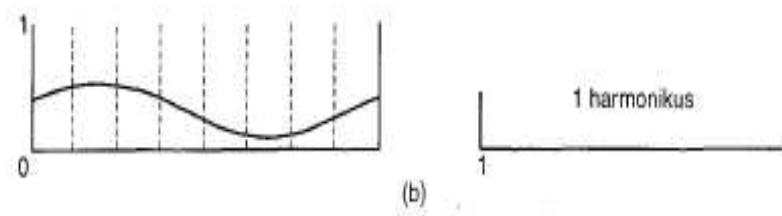


(a)

(Tanenbaum)

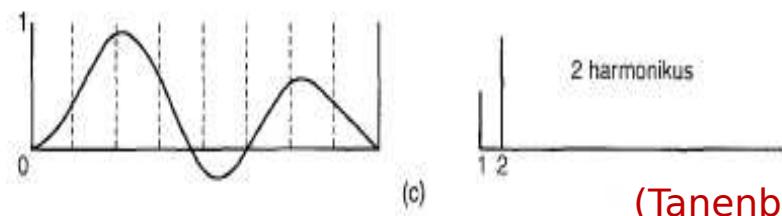


(d)



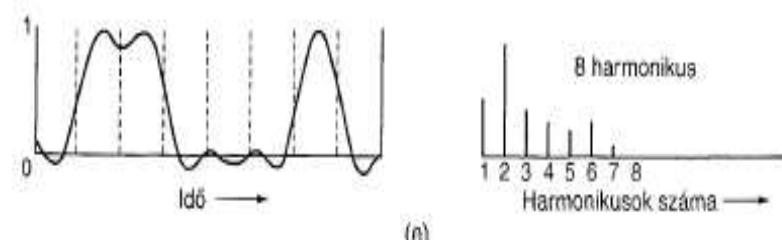
(b)

1 harmonikus.



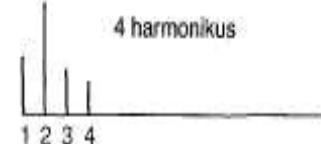
(c)

(Tanenbaum)

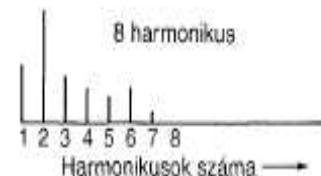


(n)

8 harmonikus



4 harmonikus

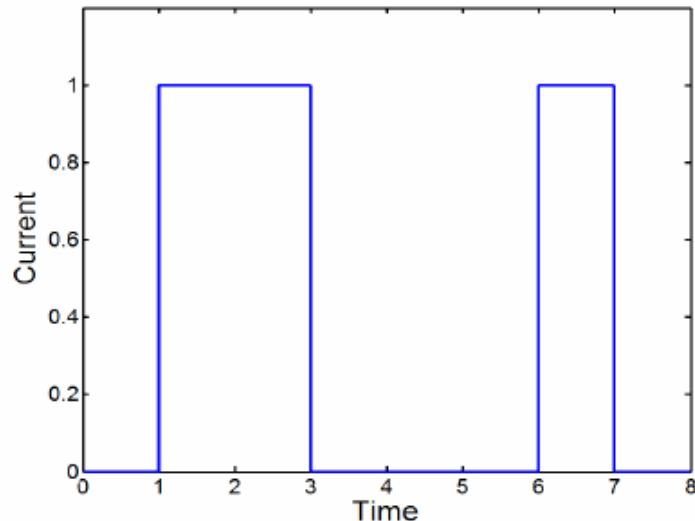


8 harmonikus

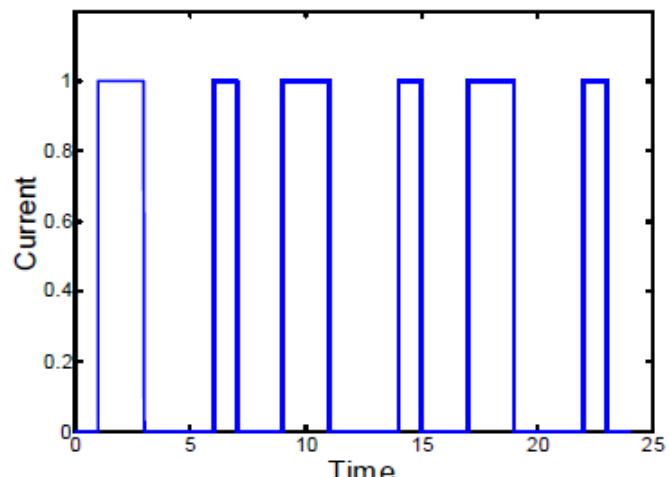
# Fourier sor felhasználása

22

- A digitális szignál nem periodikus
  - Pl. „b” ASCII kódja 8 bit hosszú
- ...de elközelhetjük, hogy végtelen sokszor ismétlődik, ami egy

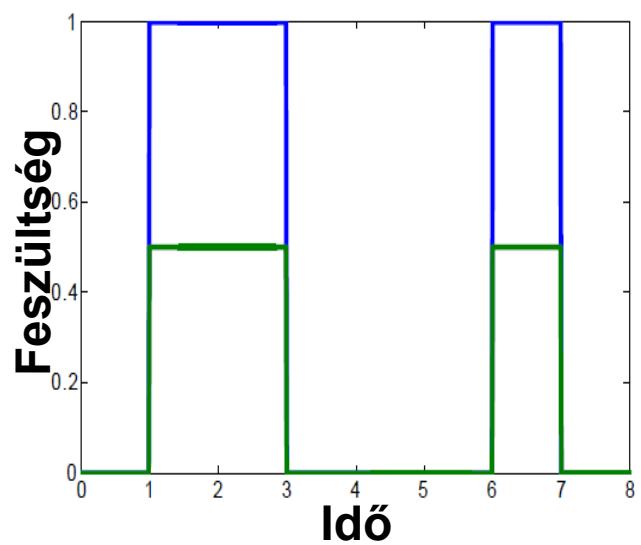


Repeated waveform for bit pattern 'b'



# Elméleti alapok - Elnyelődés

23



# Elméleti alapok - Elnyelődés

24

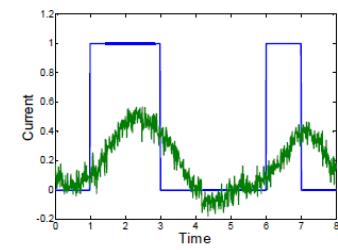
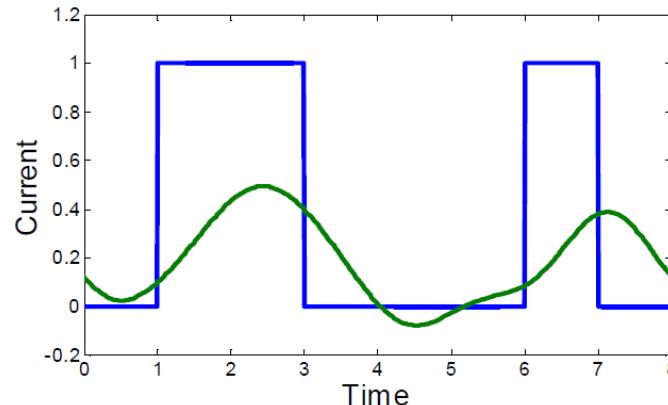
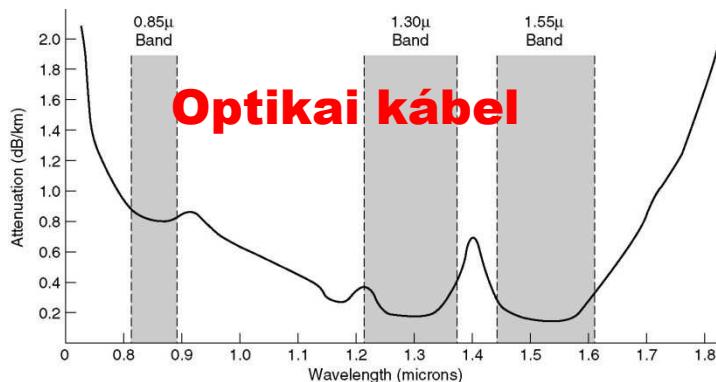
## □ Valódi közegben

### □ Frekvenciafüggő elnyelődés

### □ Fáziseltolódás

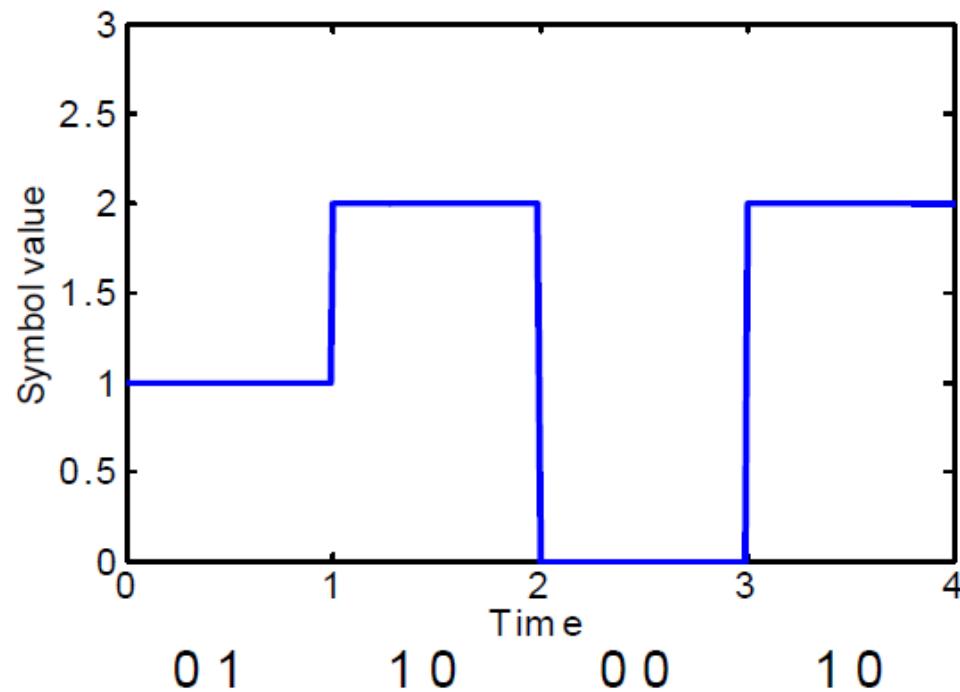
- Különböző frekvenciáknak különböző a terjedési sebessége

- Frekvenciafüggő torzítás



# Szimbólumok és bitek

- Bitek helyett szimbólumok használata az átvitelhez
- Példa:
  - Vezessünk be 4 szimbólumot:  
A(00),B(01),C(10),D(11)
  - Szimbólum ráta:  
(BAUD)
    - Elküldött szimbólumok



Példa:  
Egy 600 Baudos modemmel, ami 16 szimbólumot különböztet meg 2400 bps adatráta érhető el.

# Elméleti alapok - adatátvitel

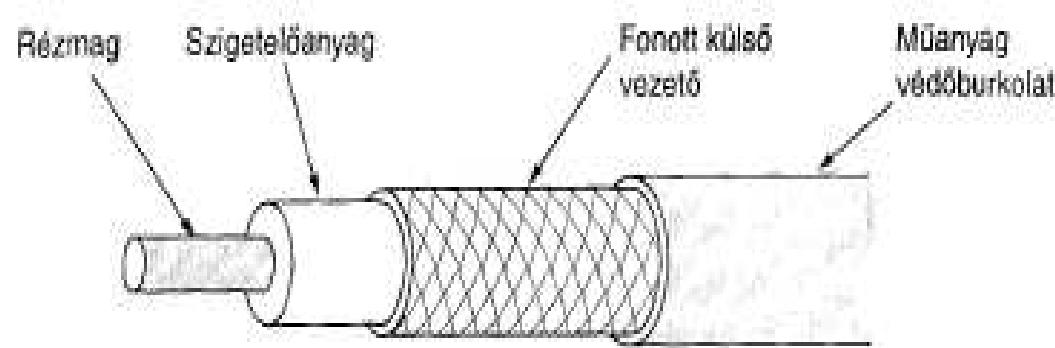
26

b/s	T (ms)	Alapharmonikus (Hz)	Elküldött harmonikusok száma
300	26,67	37,5	80
600	13,33	75	40
1 200	6,67	150	20
2 400	3,33	300	10
4 800	1,67	600	5
9 600	0,83	1200	2
19 200	0,42	2400	1
38 400	0,21	4800	0

# Átviteli közegek – vezetékes

27  
1/3

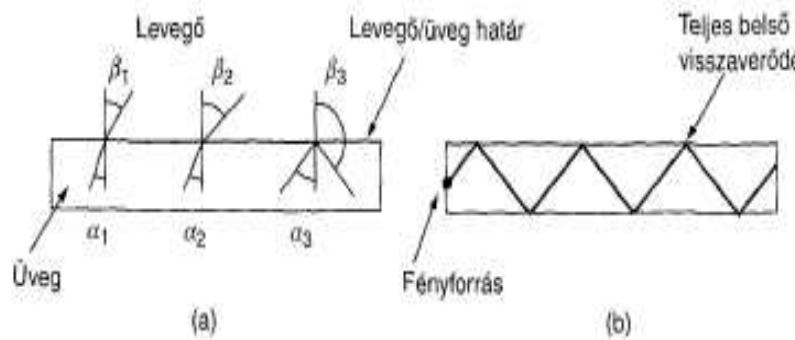
- **mágneses adathordozók** – sávszélesség jó, késleltetés nagy (nem on-line kapcsolat)
- **Sodort érpár** (angolul „*twisted pair*”) – főként távbeszélőrendszerekben használatos; dupla rézhuzal; analóg és digitális jelátvitel; UTP és STP
- **Koaxális kábel** – nagyobb sebesség és távolság érhető el, mint a sodorttal;



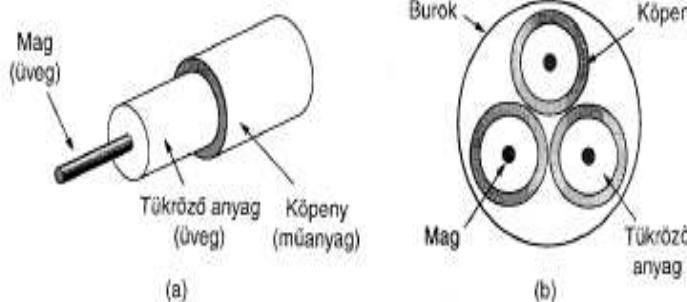
# Átviteli közegek - vezetékes

28  
2/3

- **Fényvezető szálak** – részei: fényforrás, átviteli közeg és detektor; fényimpulzus 1-es bit, nincs fényimpulzus 0-s bit; sugaraknak más-más módúsa van (határszöga < hőeső sugár szöge)



- **Fénykábelek** felépí



# Átviteli közegek - vezetékes

29 3/3

- **Fénykábelek** összevetése fényimpulzus típusa alapján

Jellemző	LED	Félvezető lézer
Adatátviteli sebesség	Alacsony	Magas
Módus	Többmóodusú	Több- vagy egymóodusú
Távolság	Kicsi	Nagy
Élettartam	Hosszú	Rövid
Hőmérsékletérzékenység	Kicsi	Jelentős
Ár	Olcsó	Drága

# Elméleti alapok - vezeték nélküli adatátvitel

30

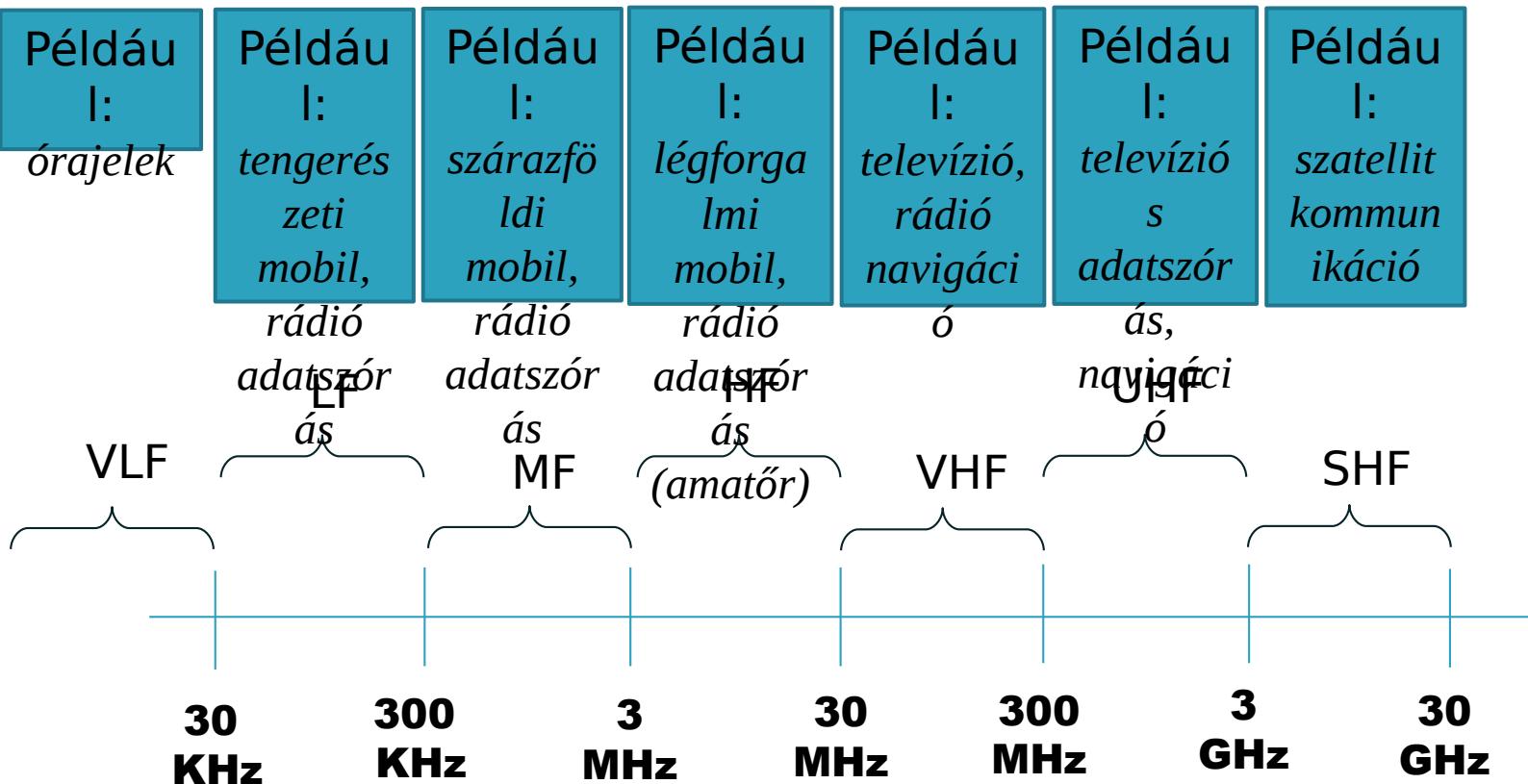
# Elméleti alapok – elektromágneses spektrum

31

Tartomány neve	Hullámhossz (centiméter)	Frekvencia (Hertz)
Rádió	>10	$< 3 * 10^9$
Mikrohullám	10 - 0.01	$3 * 10^9 - 3 * 10^{12}$
Infravörös	$0.01 - 7 \times 10^{-5}$	$3 \times 10^{12} - 4.3 \times 10^{14}$
Látható	$7 \times 10^{-5} - 4 \times 10^{-5}$	$4.3 * 10^{14} - 7.5 * 10^{14}$
Ultraibolya	$4 \times 10^{-5} - 10^{-7}$	$7.5 * 10^{14} - 3 * 10^{17}$
Röntgen sugarak	$10^{-7} - 10^{-9}$	$3 * 10^{17} - 3 * 10^{19}$
Gamma sugarak	$< 10^{-9}$	$> 3 * 10^{19}$

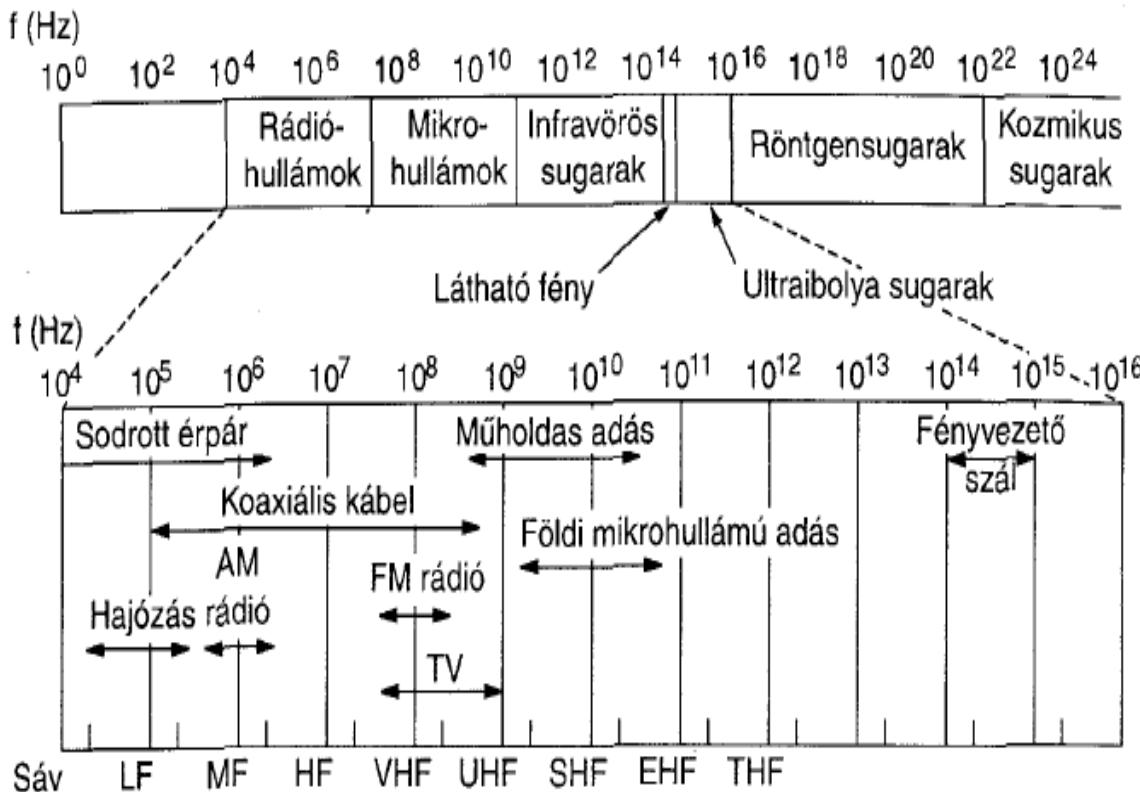
# Elméleti alapok – elektromágneses spektrum

32



# Elméleti alapok – elektromágneses spektrum

33

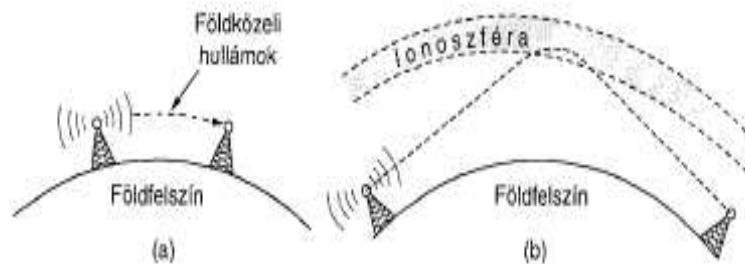


[Forrás: Tanenbaum]

# Átviteli közegek – vezeték nélküli

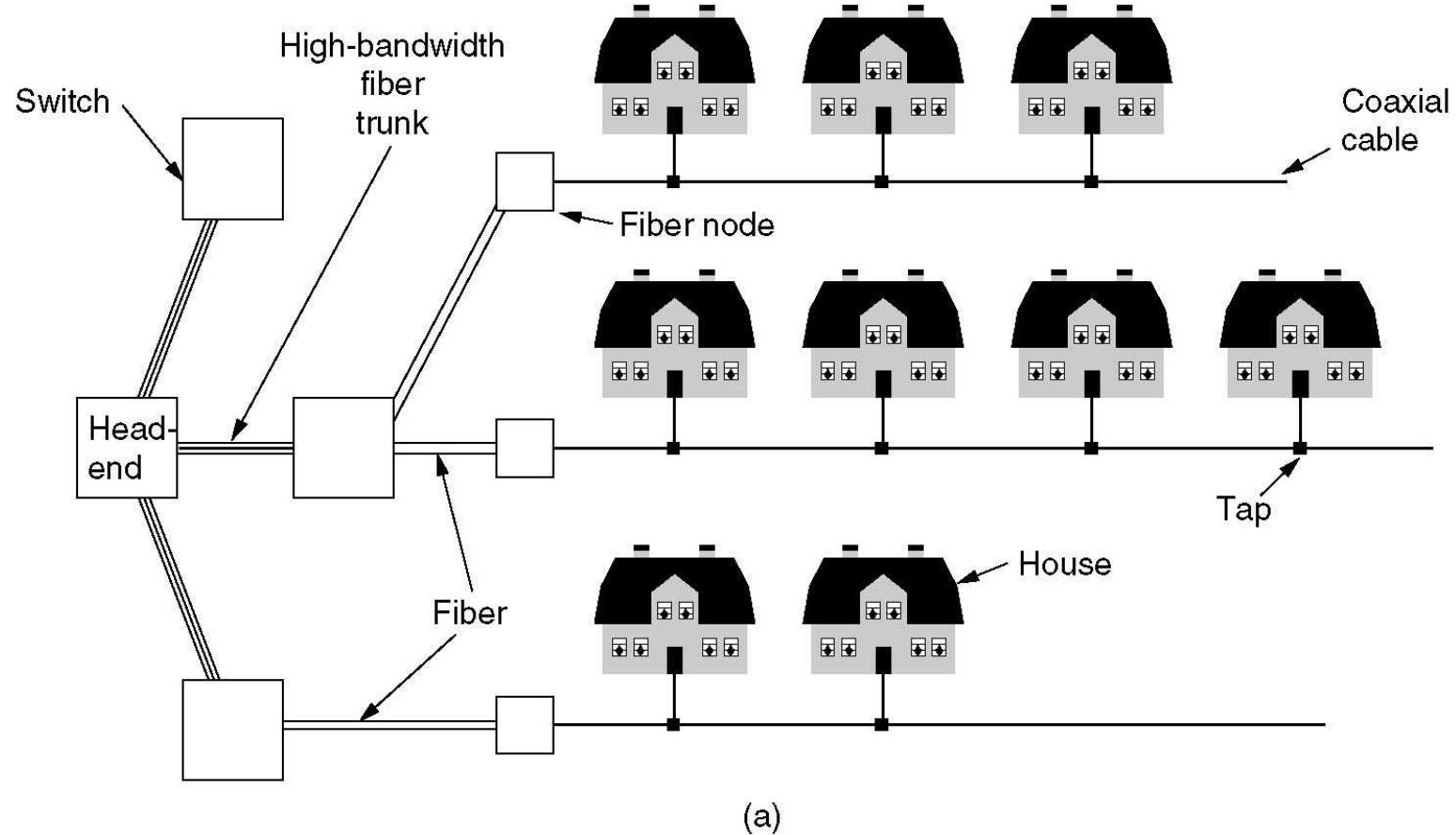
34

- **Rádiófrekvenciás átvitel** – egyszerűen előállíthatóak; nagy távolság; kültéri és beltéri alkalmazhatóság; frekvenciafüggő terjedési jellemzők

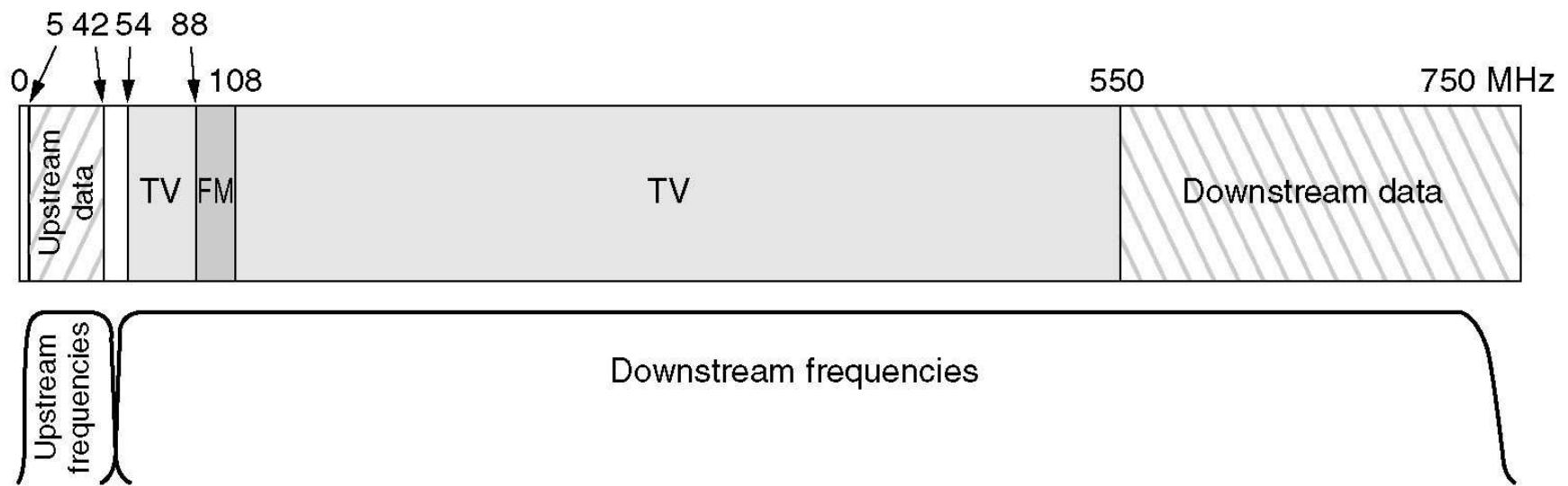


- **Mikrohullámú átvitel** – egyenes vonal mentén terjed; elhalkulás problémája; nem drága
- **Infravörös és milliméteres hullámú átvitel** – kistávolságú átvitel esetén; szilárd tárgyakon nem hatol át
- **Látható fényhullámú átvitel** – lézerforrás + fényérzékelő; nagy sávszélesség, olcsó, nem engedélyköteles; időjárás erősen

# Internet a kábel TV hálózaton



# Internet a kábel TV hálózaton



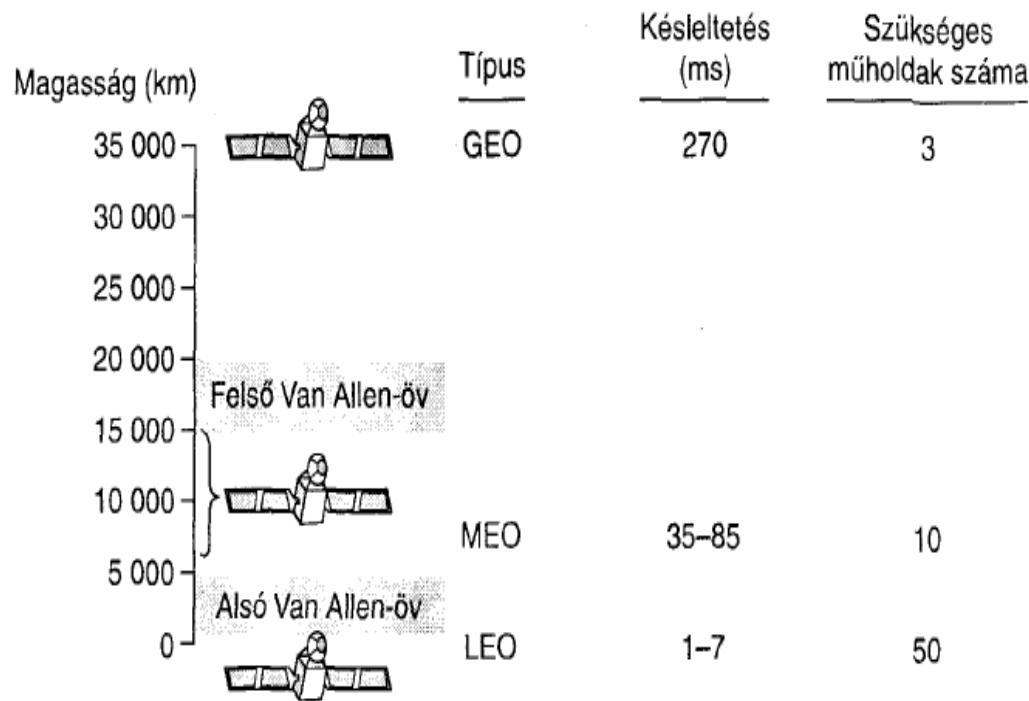
Frekvencia kiosztás egy tipikus  
kábel TV alapú Internet elérés  
esetén

# Átviteli közegek – kommunikáció műholdak

37

## Jellemzők

- Transzpondereket tartalmaz a spektrum részek figyelésére
- Jeleket felerősíti és továbbítja egy másik frekvencián
  - széles területen vagy
  - keskeny területen
- Magassággal nő a keringési idő is.



[Forrás: Tanenbaum]

# Átviteli közegek – kommunikáció műholdak

38

## Fajtái

- **Geoszinkron műholdak** – 270 milliszekundum késleltetés, 3 műhold szükséges a föld lefedésére, 35800 kilométeres magasságban keringenek
- **Közepes röppályás műholdak** – 35-85 milliszekundum késleltetés, 10 műhold szükséges a föld lefedésére, a két Van Allen-öv közötti magasságban keringenek
- **Alacsony röppályás műholdak** – 1-7 milliszekundum késleltetés, 50 műhold szükséges a föld lefedésére, az alsó Van Allen-öv alatti tartományban keringenek

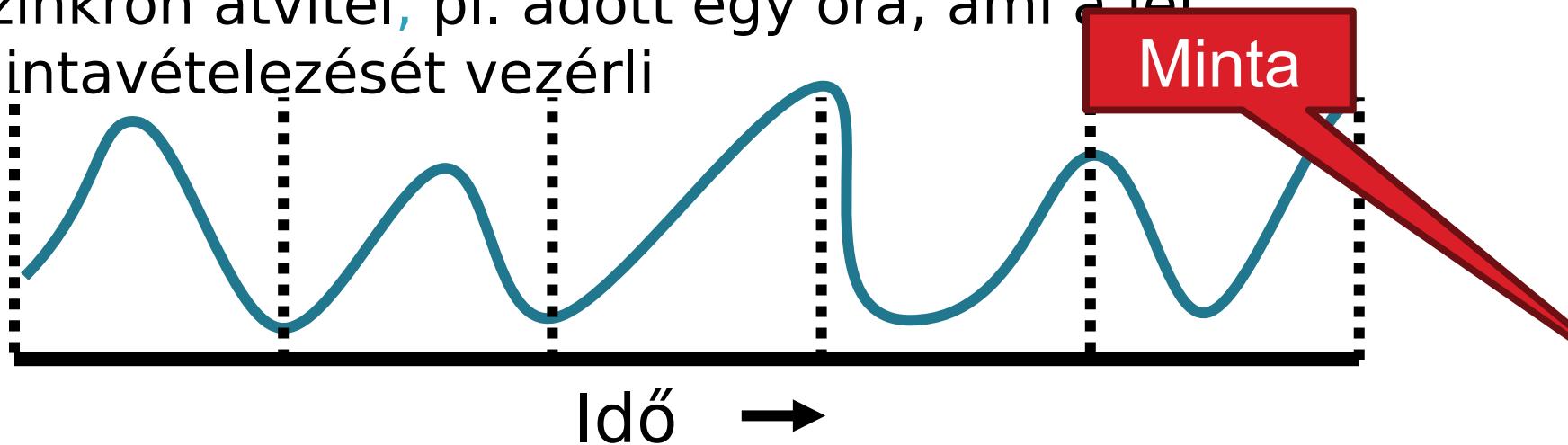


## Adatátvitel

# Kiinduló feltételek

40

- Két diszkrét jelünk van, ahol magas érték kódolja az 1-et és alacsony a 0-át.
- Szinkron átvitel, pl. adott egy óra, ami a jel mintavételezését vezérli



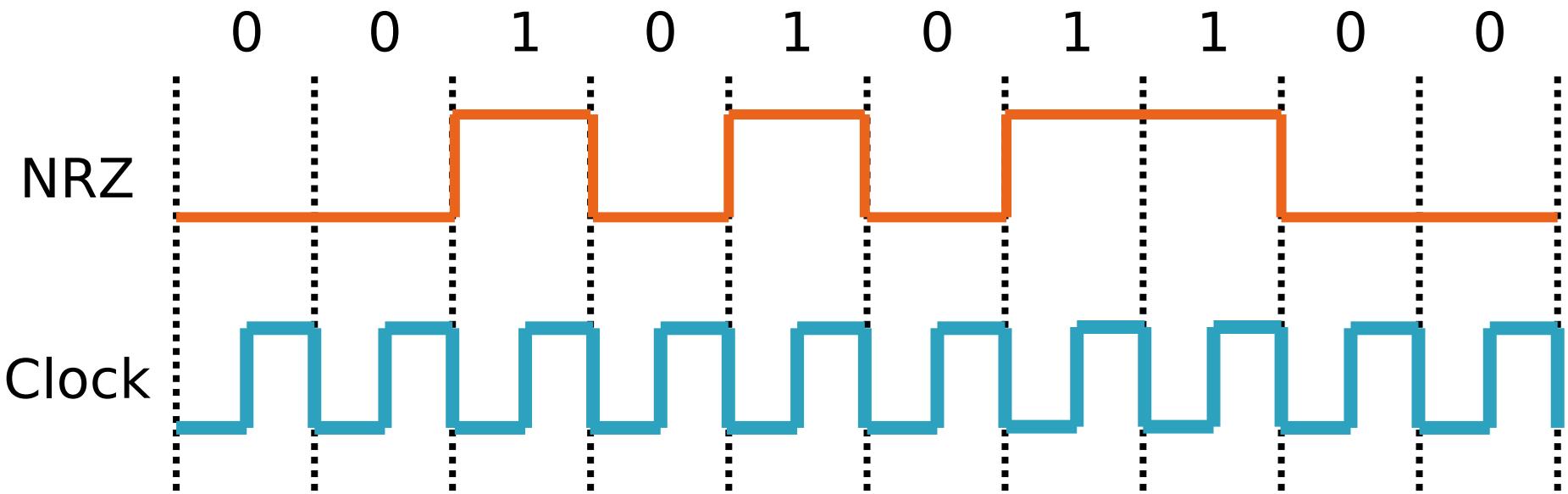
- A jel amplitúdója és az időbeli kiterjedése a fontos

# Non-Return to Zero (NRZ)

## kódolás

41

- 1 → magas jel, 0 → alacsony jel



- Probléma: 0-ákból vagy 1-esekből álló hosszú sorozatok a szinkronizáció megszűnéséhez vezetnek

↳ Hagyan különböztessük meg csak nullát ottál az

# Szinkronizáció megszűnése

42

- Probléma: mikén állítsuk vissza az órát  
osszú egyes vagy nullás sorozat után:



Az átmenetek jelzik az óra

A fogadó kihagy egy egyes bitet az órák elcsúszása miatt!!!

# Szinkronizációs megoldás

43

- Felügyelet szükséges a szinkron működéshez

## 1. Explicit órajel

- párhuzamos átviteli csatornák használata,
- szinkronizált adatok,
- rövid átvitel esetén alkalmas.

## 2. Kritikus időpontok

- szinkronizálunk például egy szimbólum vagy blokk kezdetén,
- a kritikus időpontokon kívül szabadon futnak az órák,
- feltesszük, hogy az órák rövid ideig szinkronban

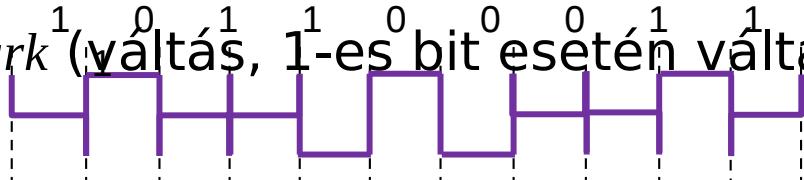
# Digitális kódok 1/3

44

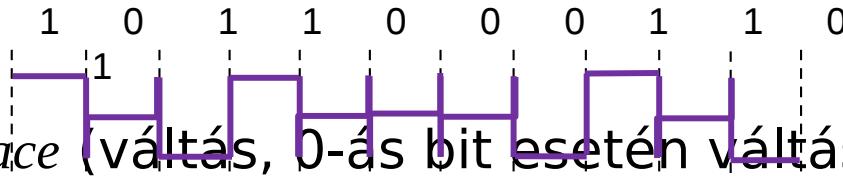
- A digitális kódok 3 lényeges momentumban térnek el:
  - i. Mi történik egy szignál intervallum elején?
  - ii. Mi történik egy szignál intervallum közepén?
  - iii. Mi történik egy szignál intervallum végén?

## Néhány konkrét digitális kód

- *Biphase-Mark* (váltás, 1-es bit esetén váltás, semmi)



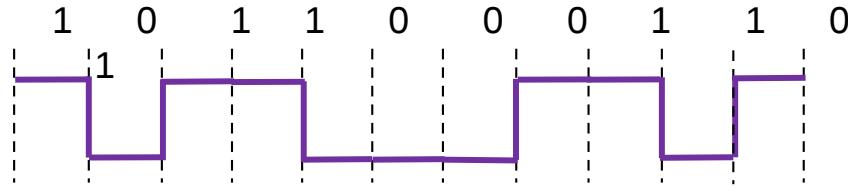
- *Biphase-Space* (váltás, 0-ás bit esetén váltás, semmi)



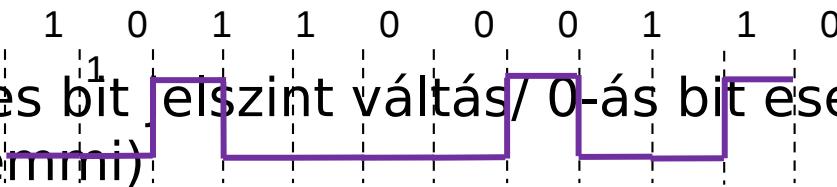
# Digitális kódok 2/3

45

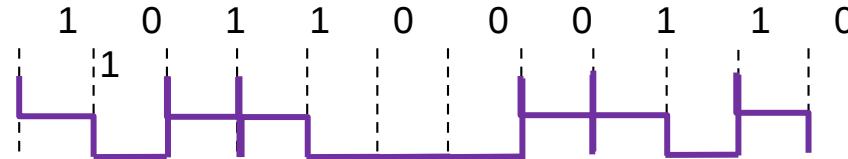
- NRZ-L (1-es bit magas jelszint/ 0-s bit alacsony jelszint, semmi, semmi)



- NRZ-M (1-es bit jelszint váltás/ 0-ás bit esetén nincs váltás, semmi, semmi)



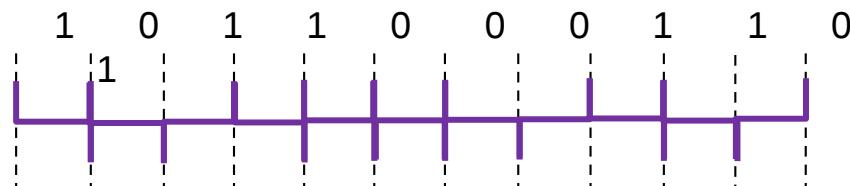
- RZ (1-es bit magas jelszint/ 0-s bit alacsony jelszint, 1-es bit esetén váltás, semmi)



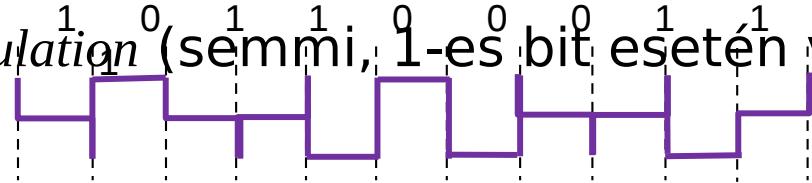
# Digitális kódok 3/3

46

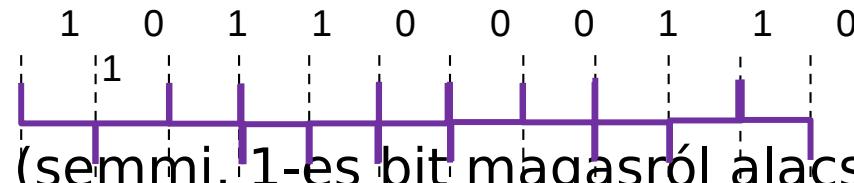
- Differential Manchester (0-s bit esetén váltás, váltás, semmi)



- Delay-Modulation (semmi, 1-es bit esetén váltás, 0-s bit következik váltás)



- Manchester (semmi, 1-es bit magasról alacsonyra/ 0-s alacsonyról magasra, semmi)

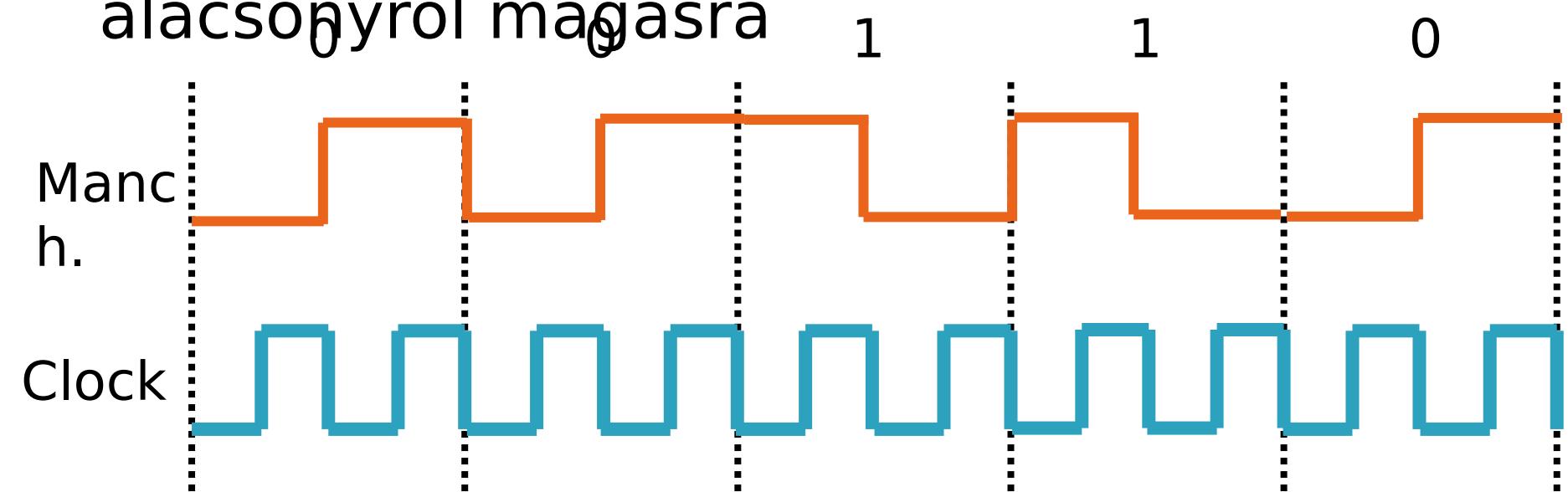


- Ethernet példa:
- 10BASE-TX
- 100BASE-TX

# Manchester (10 Mbps Ethernet 10BASE-TX)

48

- 1 → átmenet magasról alacsonyra, 0 → alacsoñyról magasra

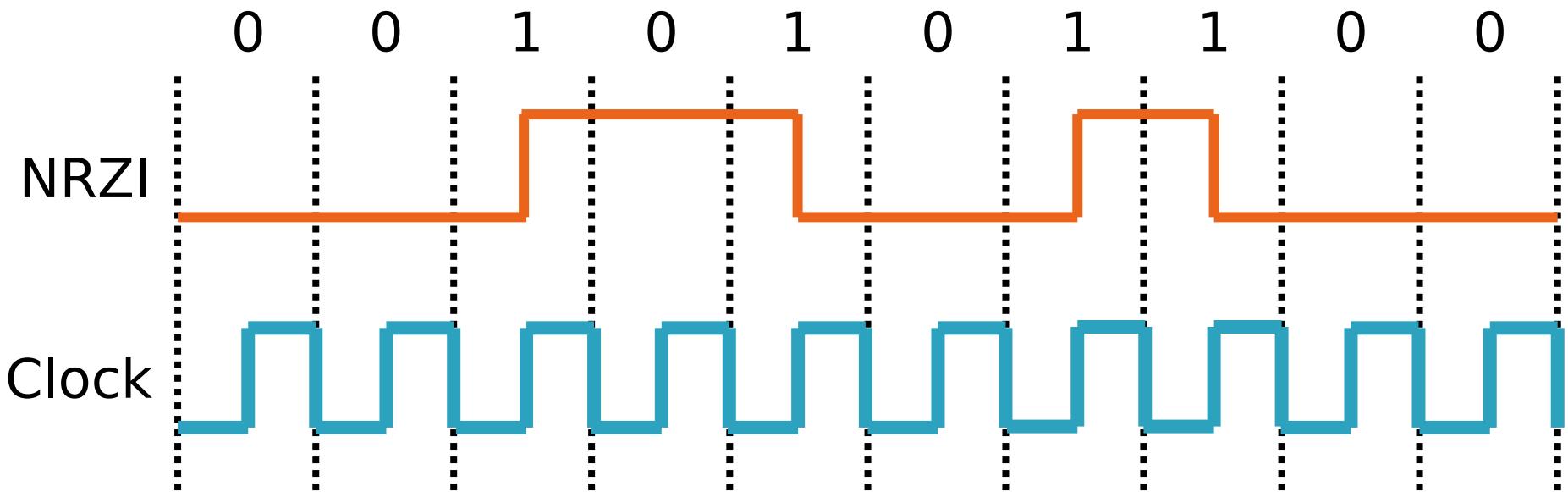


- Megoldás az órák elcsúszásának problémájára ( minden bit átmenettel kódolt)
- Negatívum, hogy az átvitel felét használja ki (létezik „0” állapot)

# Non-Return to Zero Inverted (NRZI)

49

- 1 → átmenet, 0 → ugyanaz marad



- A csupa egyes sorozat problémáját megoldja ugyan, de a csupa nulla sorozatot ez sem kezeli...

# 4-bit/5-bit kódolás NRZI előtt 50 (100 Mbps Ethernet

## Megfigyelés: **-100BASE-TX**)

- NRZI jól működik, amíg nincs csupa 0-ákból álló sorozat
- Ötlet - Kódolunk minden 4 hosszú bitsorozatot 5-bitbe:

4-bit	5-bit	4-bit	5-bit
0000	10110	0000	10000
0001	00101	0001	10010
0010	10100	0001	10011
0011	10101	0010	10100
0100	01010	0011	10110
0101	01011	0110	10111
0110	01110	1100	
0111	01111	11010	
		1110	

# 4-bit/5-bit kódolás NRZI

## (100 Mbps Ethernet)

51

- Megfizetés: 100

NRZ sorozatban, amely minden 4 bites sorozatot 5-bitbe kódolja.

- Ötlet - Kódolunk minden 4 hosszú bitsorozatot 5-bitbe:

Nem lehet több mint 1 nulla a sorozat elején, és nem lehet kettőnél több nulla végén.

4-bit	5-bit	4-bit	5-bit
0000	10110	0000	10001
0001	01001	0001	10010
0010	10100	0001	10011
0011	10101	0100	10100
0100	01010	0101	10110
0101	01011	0110	10111
0110	01110	1100	
0111	01111	11010	
		11011	

Jelátvitel

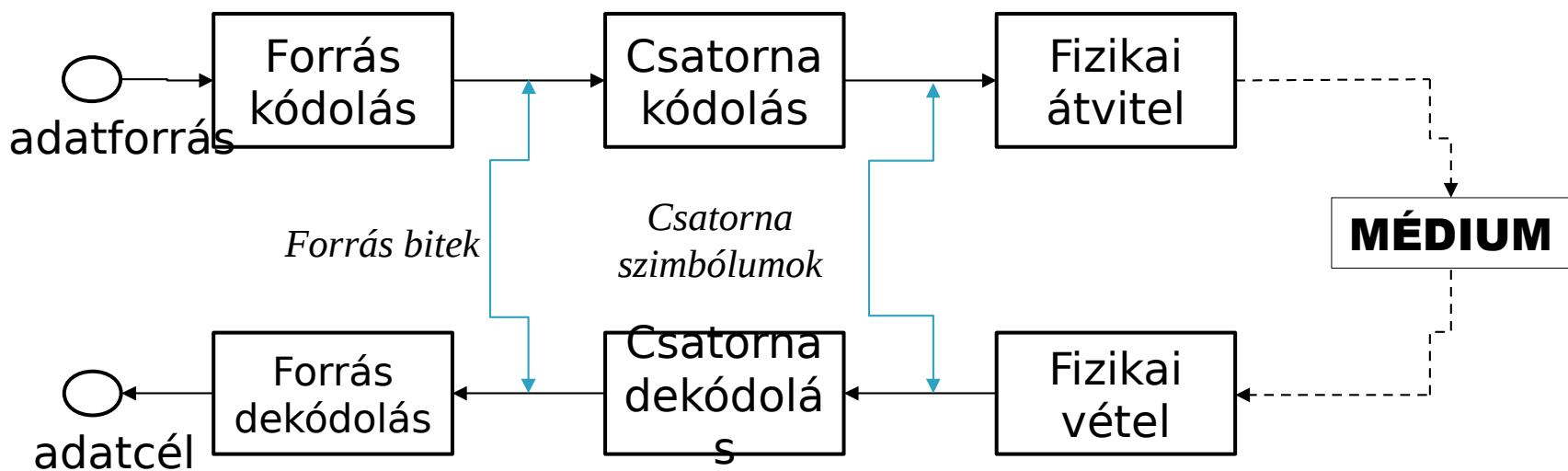
# Alapsáv és széles-sáv

53

- Alapsáv avagy angolul *baseband*
  - a digitális jel direkt árammá vagy feszültséggé alakul;
  - a jel minden frekvencián átvitelre kerül;
  - átviteli korlátok.
- Szélessáv avagy angolul *broadband*
  - Egy széles frekvencia tartományban történik az átvitel;
  - a jel modulálására az alábbi lehetőségeket használhatjuk:
    - adatok vivőhullámra „ültetése” (*amplitúdó moduláció*);
    - vivőhullám megváltoztatása (*frekvencia vagy fázis moduláció*);
    - különböző vivőhullámok felhasználása egyidejűleg

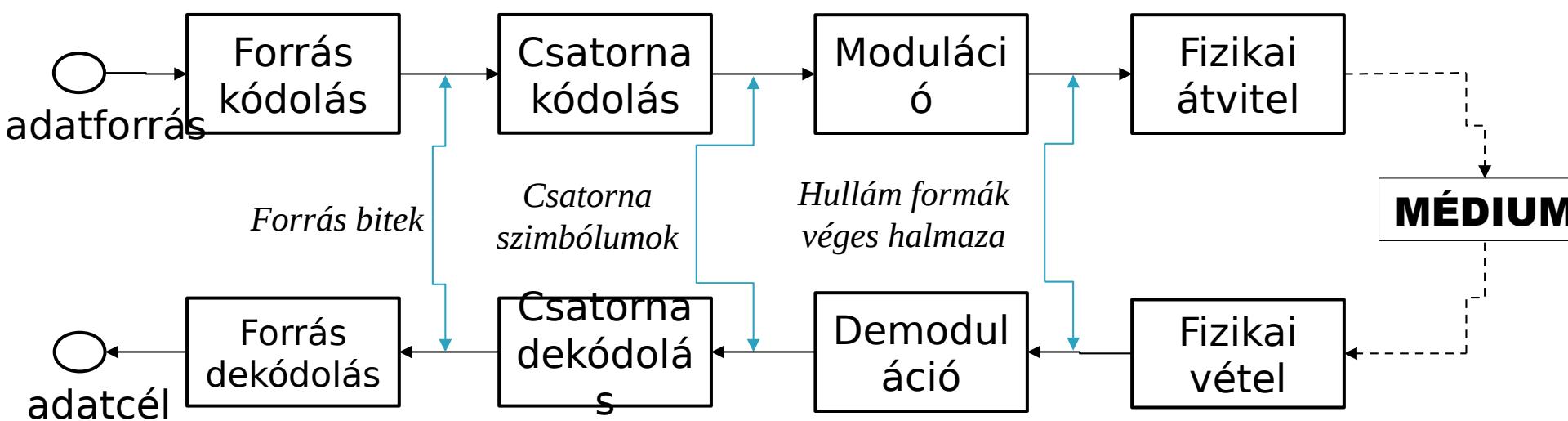
# Digitális alapsávú átvitel struktúrája

54



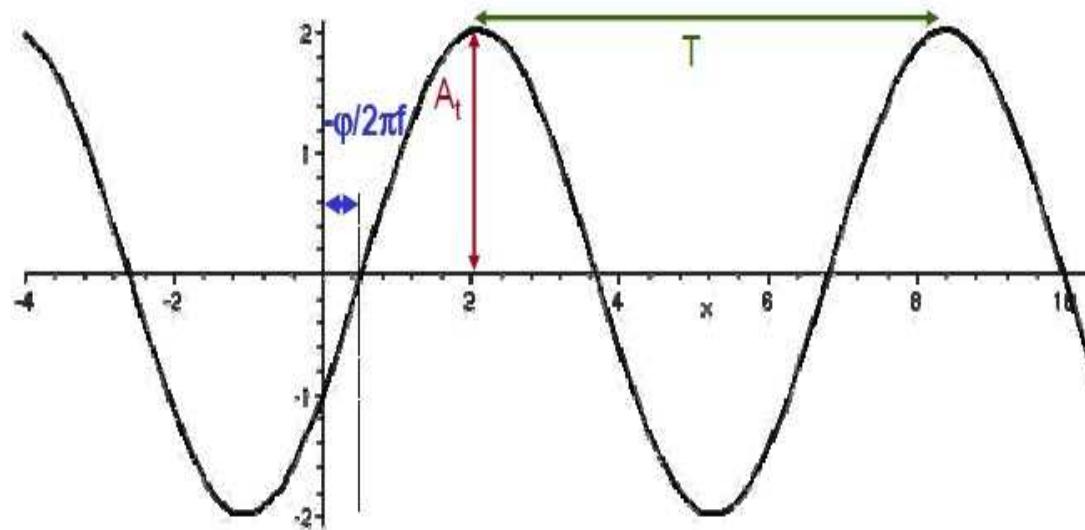
# Digitális szélessávú átvitel struktúrája

55



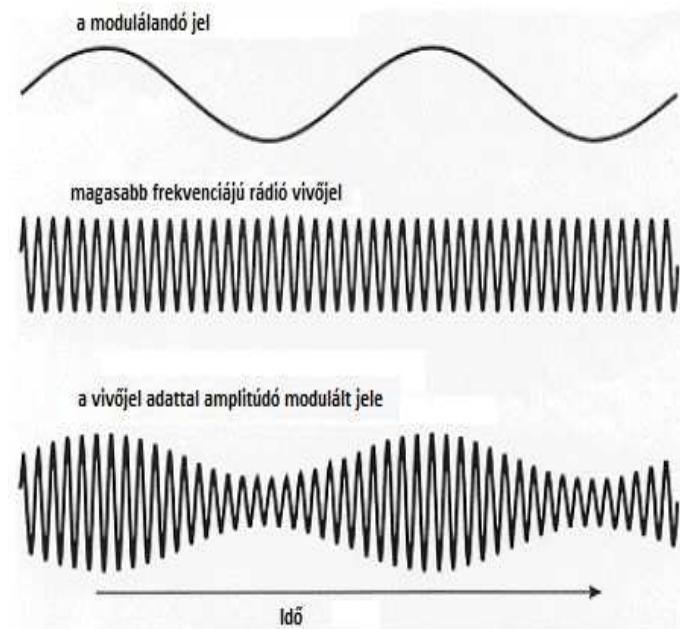
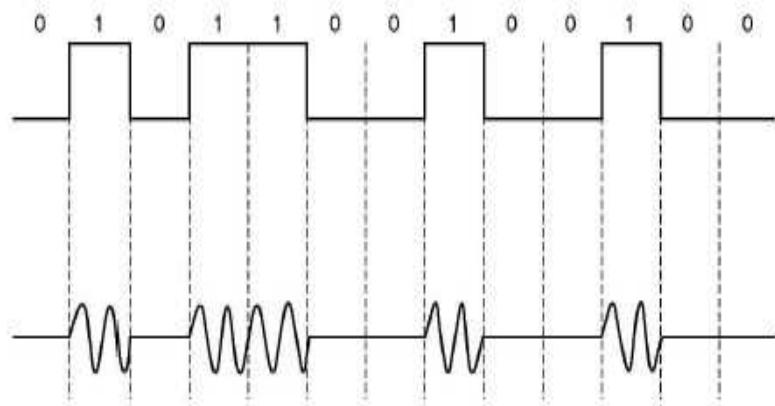
# Amplitúdó ábrázolás

56



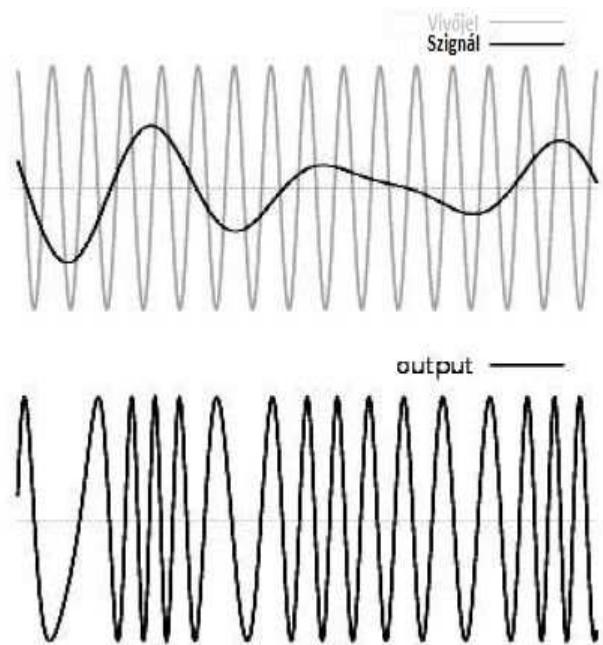
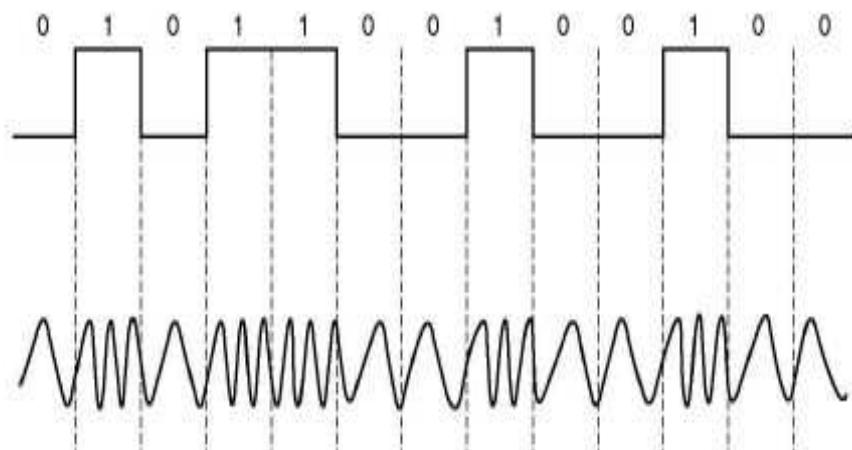
# Amplitúdó moduláció

57

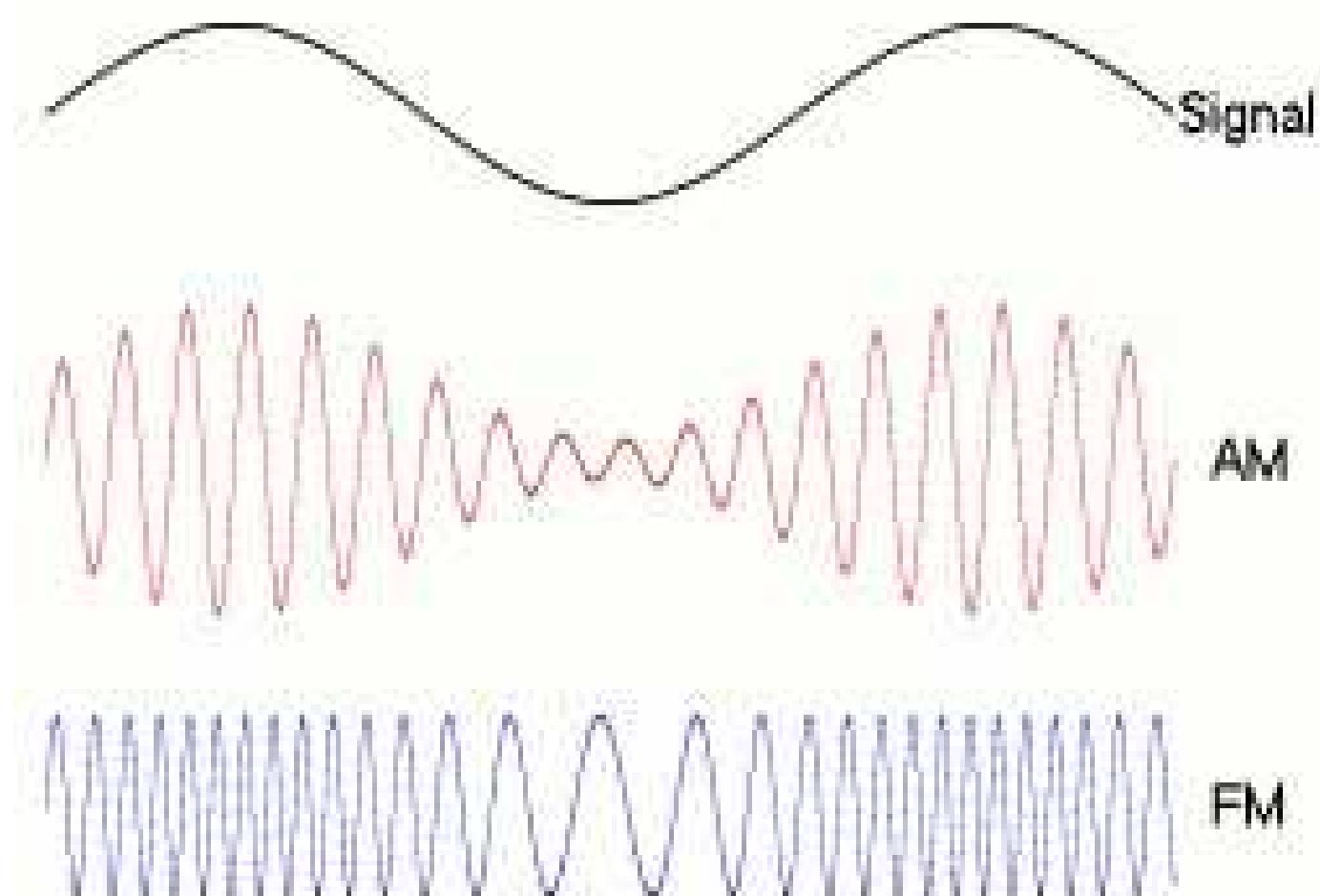


# Frekvencia moduláció

58

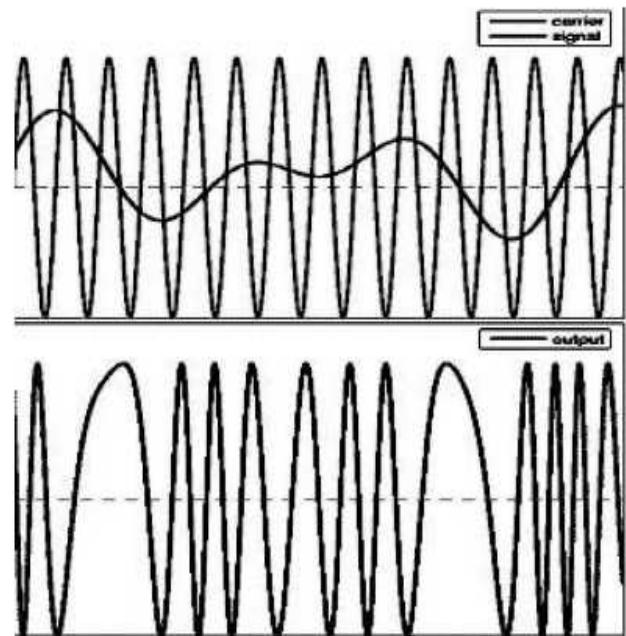
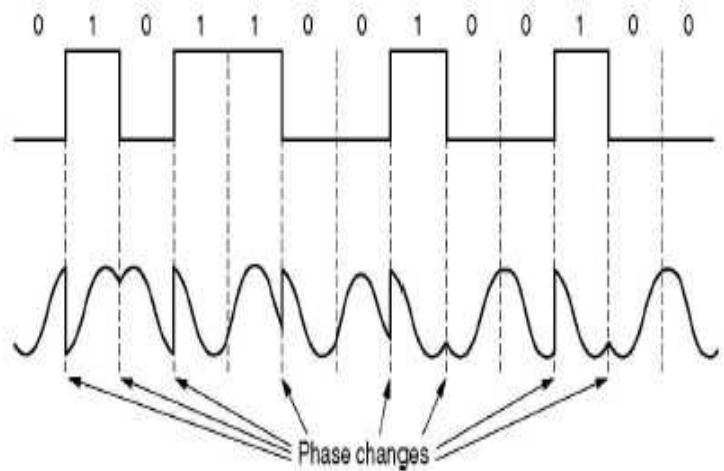


# Illusztráció - AM & FM analóg jel esetén



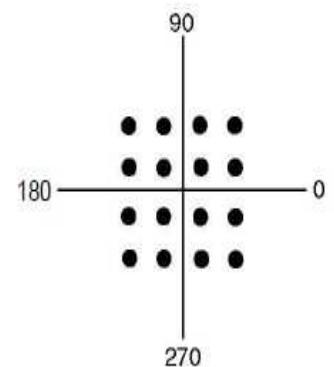
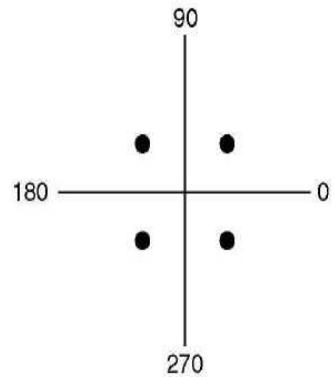
# Fázis moduláció

60



# Több szimbólum használata

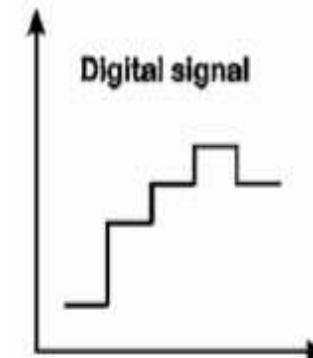
61



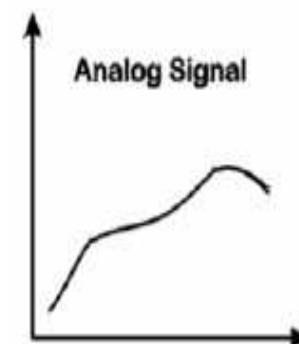
# Digitális és analóg jelek összehasonlítása

62

- *Digitális átvitel* – Diszkrét szignálok véges halmazát használja (például feszültség vagy áramerősség értékek).



- *Analóg átvitel* – Szignálok folytonos halmazát használja (például feszültség vagy áramerősség a vezetékben)



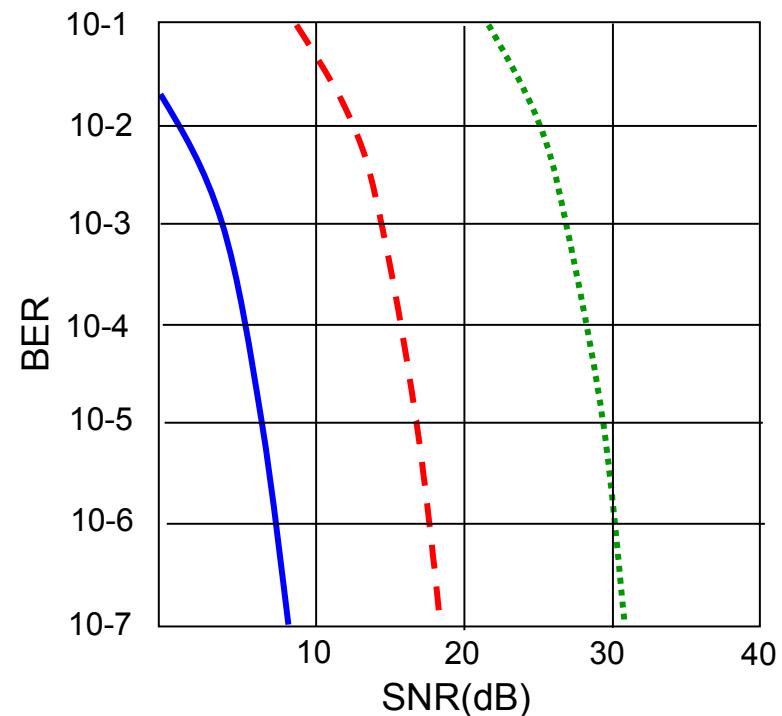
- *Digitális előnyei*
  - Lehetőség van a vételpontosság helyreállítására illetve az eredeti jel helyreállítására
- *Analóg hátránya*

# Bithiba gyakoriság és a jel-zaj arány

63

- Minél nagyobb a jel-zaj arány avagy **SNR** (*Signal-to-noise ratio*), annál kevesebb hiba lép fel
- A hibásan fogadott bitek részarányát bithiba gyakoriságnak avagy BER-nek (bit error rate) nevezzük
- A BER függ az alábbiaktól
  - a jel erőségétől,
  - a zajtól,
  - az átviteli sebességtől,
  - a felhasznált módszertől

QAM256 (8 Mbps)  
--- QAM16 (4 Mbps)  
— BPSK (1 Mbps)



- Csatorna hozzáférés módszerei
- (statikus)

# Multiplexálás

65

- Lehetővé teszi, hogy több jel egyidőben utazzon egy fizikai közegen
- Több jel átvitele érdekében a csatornát logikailag elkülönített kisebb csatornákra (alcsatornára) bontjuk
- A küldő oldalon szükséges egy speciális eszköz (multiplexer), mely a jeleket a csatorna megfelelő alcsatornáira helyezi

# Térbeli multiplexálás

66

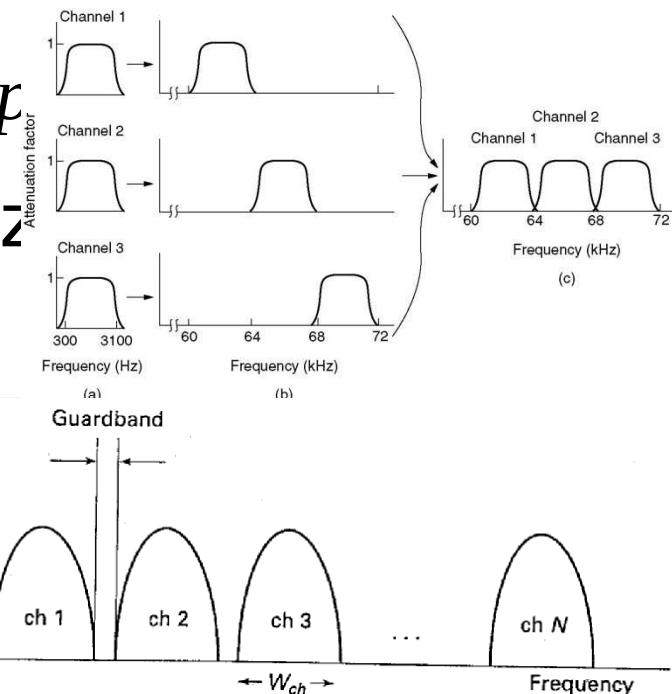
- Ez a legegyszerűbb multiplexálási módszer.
- Angolul **S**pace-**D**ivision **M**ultiplexing
- Vezetékes kommunikáció esetén minden egyes csatornához külön pont-pont vezeték tartozik.
- Vezeték nélküli kommunikáció esetén minden egyes csatornához külső rendelődik.



# Frekvencia multiplexálás

67

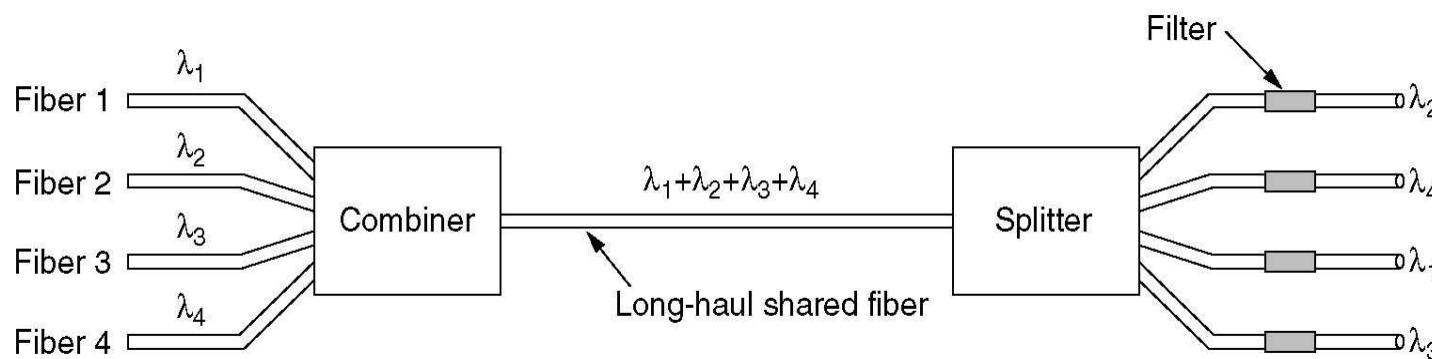
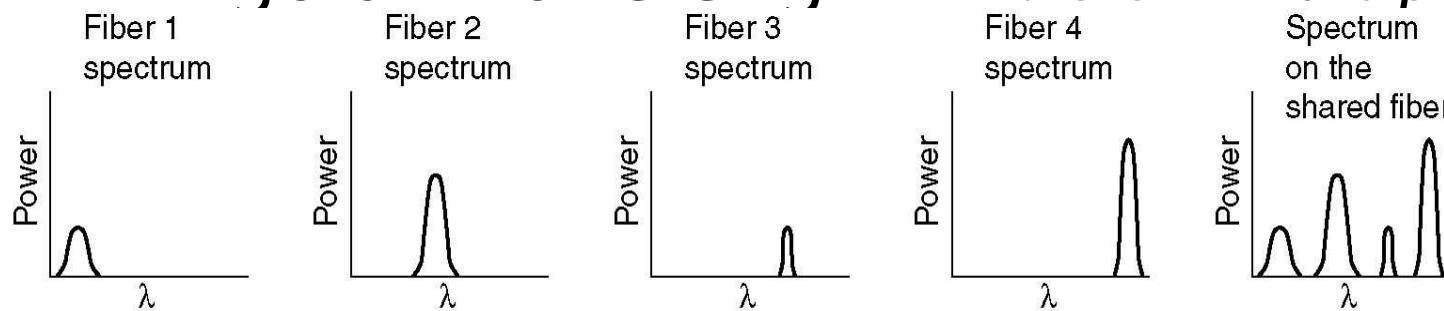
- Olyan módszertan, amelyben egy kommunikációs csatornán több szignál kombinációja adja az átvitelt.
- minden szignálhoz más frekvencia tartozik.
- Angolul **F**requency-**D**ivision **M**ultiplex
- Tipikusan analóg vonalon használják.
- Többféle megvalósítása van:
  - XOR a szignálokon véletlen bináris szám alapú
  - pszeudo véletlen szám alapú



# Hullámhossz multiplexálás

68

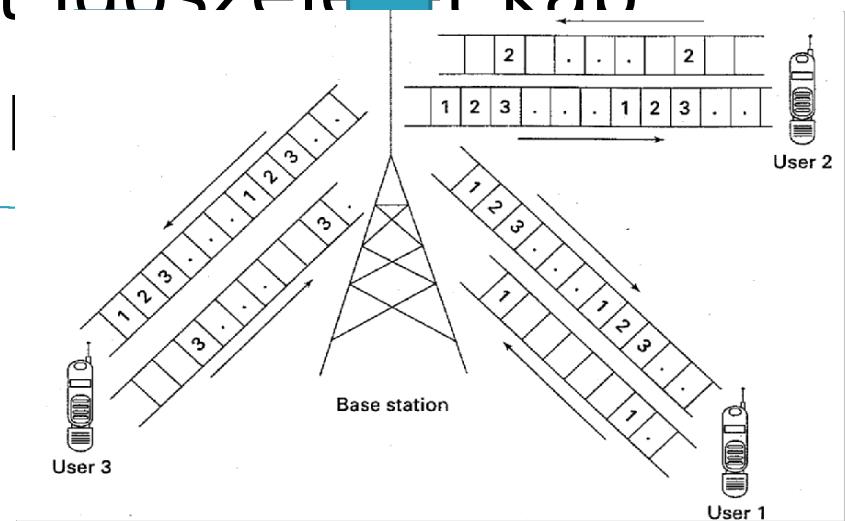
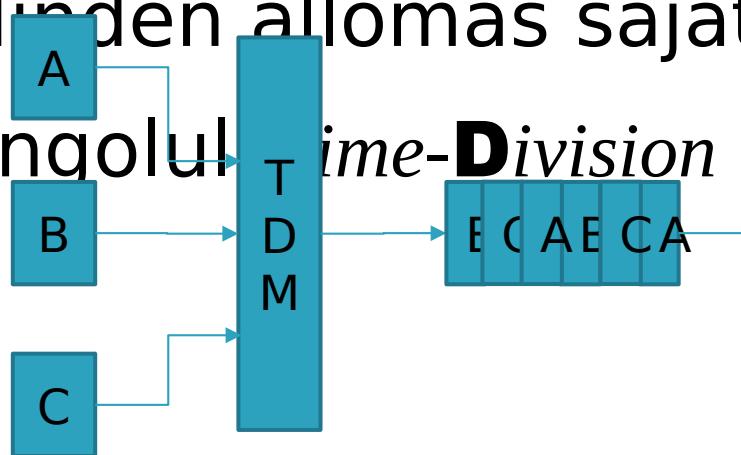
- Optikai kábeleknél alkalmazzák.
- Angolul **Wavelength-Division Multiplexing**



# Időbeli multiplexálás

69

- Több párhuzamos adatfolyam átvitelét a jelsorozat rövid időintervallumokra szegmentálásával oldja meg.
- Diszkrét időszeletek használata.  
Minden állomás saját időszeletet kan
- Angolul *Time-Division Multiplexing* |



Köszönöm a figyelmet!

# Számítógépes Hálózatok

**3. Előadás: Fizikai réteg**  
**II.rész Adatkapcsolati**  
**réteg**

Based on slides from

□ Csatorna hozzáférés  
módszerei

# Multiplexálás

3

- Lehetővé teszi, hogy több jel egy időben utazzon egy fizikai közegen
- Több jel átvitele érdekében a csatornát logikailag elkülönített kisebb csatornákra (alcsatornára) bontjuk
- A küldő oldalon szükséges egy speciális eszköz (multiplexer), mely a jeleket a csatorna megfelelő alcsatornáira helyezi

# Térbeli multiplexálás

4

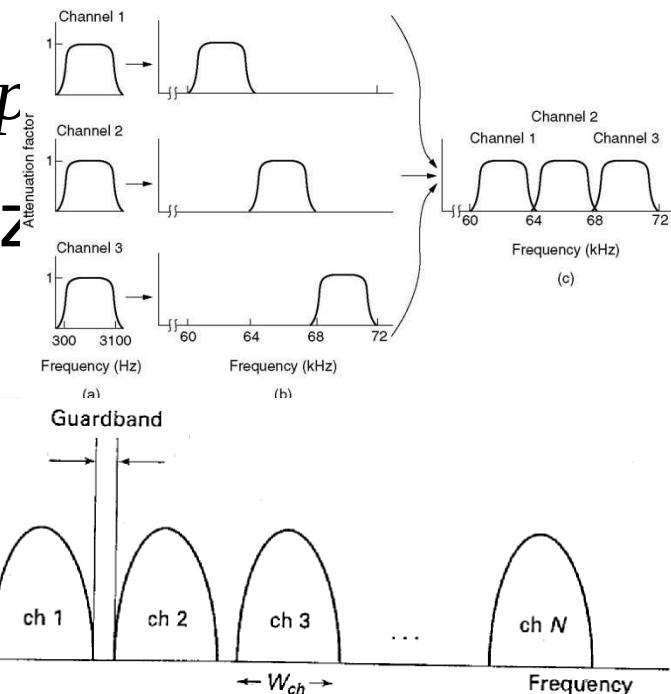
- Ez a legegyszerűbb multiplexálási módszer.
- Angolul **S**pace-**D**ivision **M**ultiplexing
- Vezetékes kommunikáció esetén minden egyes csatornához külön pont-pont vezeték tartozik.
- Vezeték nélküli kommunikáció esetén minden egyes csatornához külső antennát rendelődik.



# Frekvencia multiplexálás

5

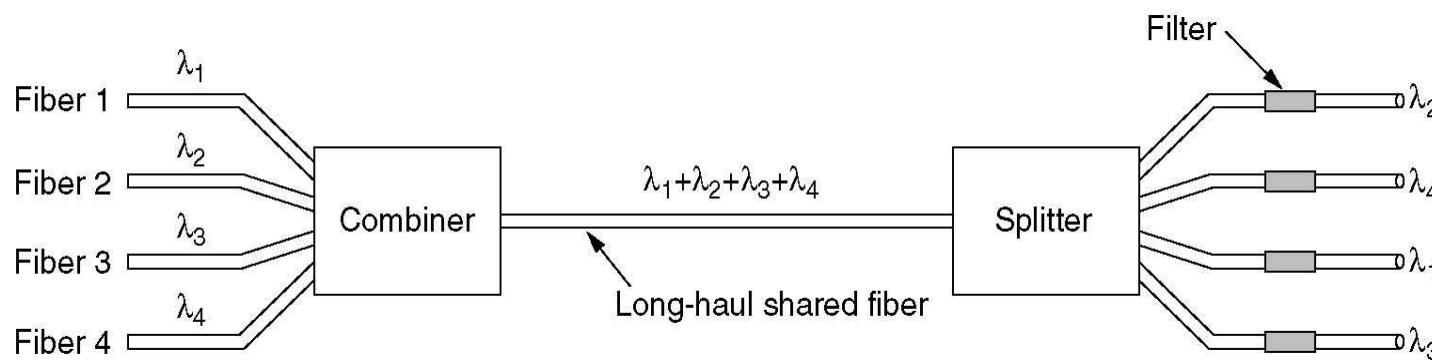
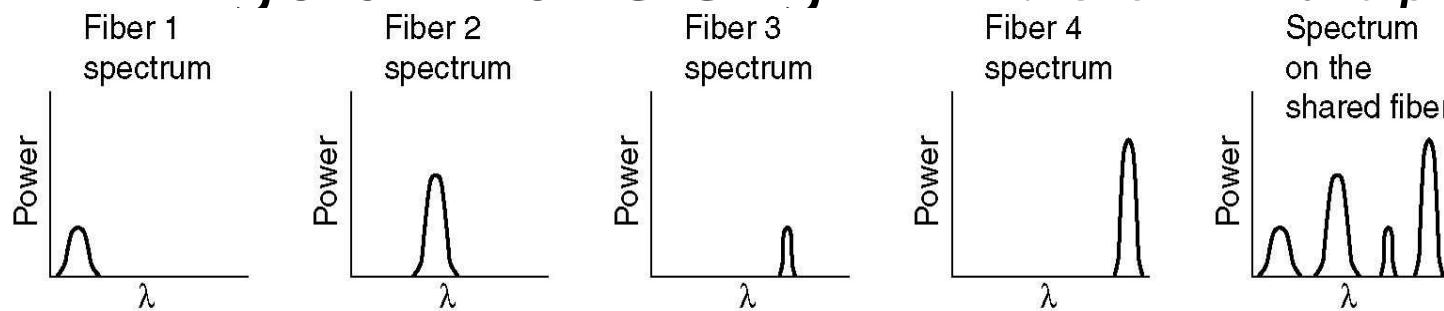
- Olyan módszertan, amelyben egy kommunikációs csatornán több szignál kombinációja adja az átvitelt.
- minden szignálhoz más frekvencia tartozik.
- Angolul **F**requency-**D**ivision **M**ultiplex
- Tipikusan analóg vonalon használják.
- Többféle megvalósítása van:
  - XOR a szignálokon véletlen bináris szám alapú
  - pszeudo véletlen szám alapú



# Hullámhossz multiplexálás

6

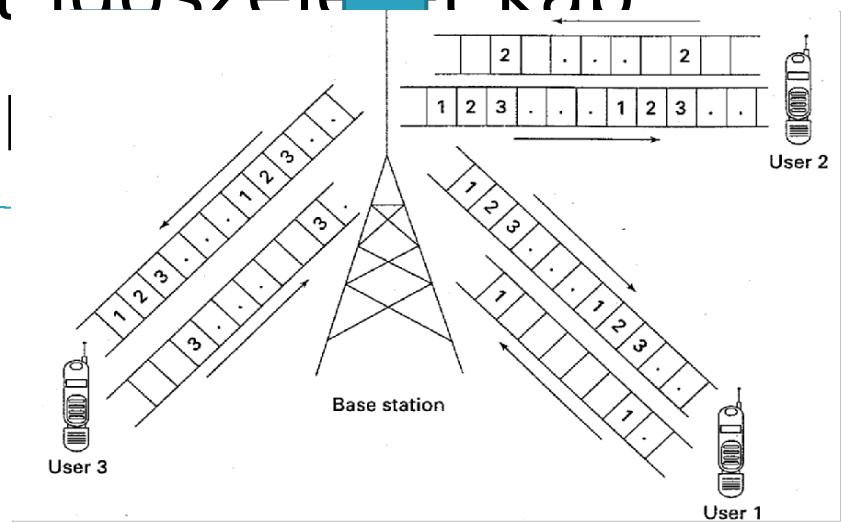
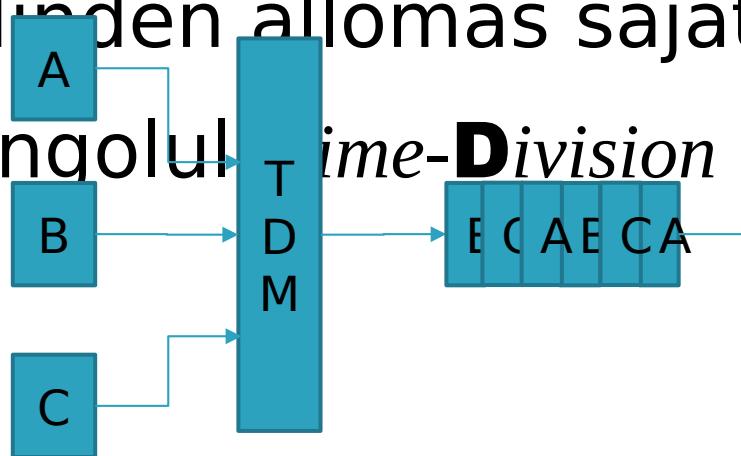
- Optikai kábeleknél alkalmazzák.
- Angolul **Wavelength-Division Multiplexing**



# Időbeli multiplexálás

7

- Több párhuzamos adatfolyam átvitelét a jelsorozat rövid időintervallumokra szegmentálásával oldja meg.
- Diszkrét időszeletek használata.  
Minden állomás saját időszeletet kan
- Angolul *Time-Division Multiplexing* |



# Code Division Multiple Access

8 1/3

- a harmadik generációs mobiltelefon hálózatok alapját képezi (IS-95 szabvány)
- minden állomás egyfolytában sugározhat a rendelkezésre álló teljes frekvenciasávon
- Feltételezi, hogy a többszörös jelek lineárisan összeadódnak.
- **Kulcsa:** a hasznos jel kiszűrése

## Algoritmus

- minden bitidőt  $m$  darab rövid intervallumra osztunk, ezek a töredékek (angolul *chip*)
- minden állomáshoz egy  $m$  bites kód tartozik, úgynevezett töredéksorozat (angolul *chip sequence*)
- Ha 1 csíkot elhagyunk, újból minden állomásról kell küldeni a saját

# Code Division Multiple Access

9 2/3

- m-szeres sávszélesség válik szükségessé, azaz szórt spektrumú kommunikációt valósít meg
- szemléltetésre bipoláris kódolást használunk:
  - bináris 0 esetén -1; bináris 1 esetén +1
  - az állomásokhoz rendelt töredék sorozatok **páronként ortogonálisak**

# Code Division Multiple Access

10 3/3

# Code Division Multiple Access

## példa

11

### A állomás

Chip kódja legyen (1,-1).

Átvitelre szánt adat legyen 1011

1. Egyedi szignál előállítása az (1,0,1,1) vektorra:

$$((1,-1),(-1,1),(1,-1),\\(1,-1))$$

~~((1+(-1),(-1)+(-1),((-1)+(-1),1+(-1)),(1+1,(-1)+1),(1+1,(-1)+1)) =~~

Szignál modulálása a csatornára.

$$(0,-2,-2,0,2,0,2,0)$$

### B állomás

Chip kódja legyen (1,1).  
Átvitelre szánt adat legyen 0011

1. Egyedi szignál előállítása az (0,0,1,1) vektorra:  
 $((-1,-1),(-1,-1),(1,1),\\(1,1))$
2. Szignál modulálása a csatornára.

# Code Division Multiple Access

## nélda

$$1(1+(-1),(-1)+(-1)),((-1)+(-1),1+(-1)),(1+1,(-1)+1),(1+1,(-1)+1)) =$$

$$((0,-2),(-2,0),(2,0),(2,0))$$

### Vevő 1

Ismeri B chip kódját: (1,1).

1. Visszakódolás az ismert kóddal:

$$((0,-2)*(1,1),(-2,0)*(1,1),(2,0)*(1,1),\\(2,0)*(1,1))$$

2. Kapott (-2,-2,2,2) eredmény értelmezése:

(-,-,+,), azaz 0011 volt az üzenet B-től.

### Vevő 2

Ismeri A chip kódját: (1,-1).

1. Visszakódolás az ismert kóddal:

$$((0,-2)*(1,-1),(-2,0)*(1,-1),(2,0)*(1,-1),(2,0)*(1,-1))$$

2. Kapott (2,-2,2,2) eredmény értelmezése:

(+,-,+,), azaz 1011 volt az üzenet A-tól.

# Médium többszörös használata összefoglalás

13

- Tér-multiplexálás avagy *SDM* (párhuzamos adatátviteli csatornák)
  - cellurális hálózatok
- Frekvencia-multiplexálás avagy *FDM*(a frekvencia tartomány felosztása és küldőhöz rendelése)
  - „**D**irect **S**equence **SS” (XOR a szignálokon véletlen bitsorozattal)**
  - „**F**requency **H**opping **SS” (pszeudo véletlen szám alapú választás)**
- Idő-multiplexálás avagy *TDM* (a médium használat időszeletekre osztása és küldőhöz rendelése)
  - diszkrét idő szeletek (*slot*)
  - koordináció vagy merev felosztás kell hozzá
- Hullámhossz-multiplexálás avagy *WDM* (színes szigetelés)

# Adatkapcsolati réteg

# Adatkapcsolati réteg

- Szolgáltatás
- Adatok keretekre tördelése:  
határok a csomagok között

15



- Közeghozzáférés vezérlés (MAC)
- Per-hop megbízhatóság és folyamvezérlés
- Interfész
  - Keret küldése két közös médiumra kötött eszköz között
- Protokoll
  - Fizikai címzés (pl. MAC address, IP address)

# Adatkapcsolati réteg

16



- Funkciók:
  - Adat blokkok (*keretek/frames*) küldése eszközök között
  - A fizikai közeghez való hozzáférés szabályozása
- Legfőbb kihívások:
  - Hogyan *keretezzük* az adatokat?
  - Hogyan ismerjük fel a *hibát*?

# Adatkapcsolati réteg

## Feladatai

- jól definiált szolgálati interfész biztosítása a hálózati rétegnek:
  - nyugtázatlan összeköttetés alapú szolgálat;
  - nyugtázott összeköttetés nélküli szolgálat;
  - nyugtázott összeköttetés alapú szolgálat (3 fázis);
- átviteli hibák kezelése;
- adatforgalom szabályozása (elárasztás elkerülése)

Keret képzés /  
Keretezés / Framing

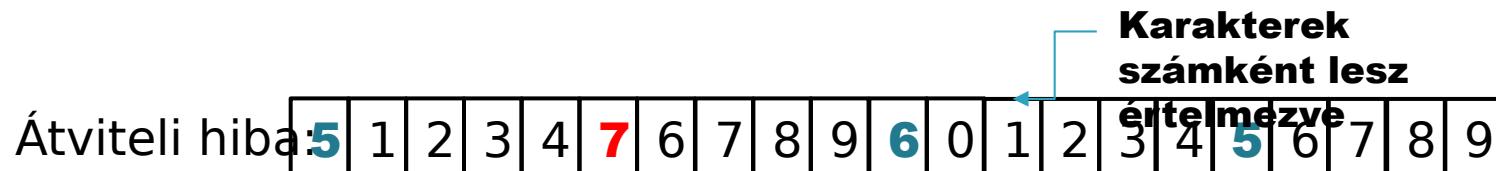
# Keret

## <sup>19</sup> Képzés/Keretezés/Framing

- A bitek kódolását a fizikai réteg határozza meg
- A következő lépés az adatblokkok „kódolása”
  - I Csomag-kapcsolt hálózatok
    - minden csomag útvonal (routing) információt is tartalmaz
    - az adathatárokat ismernünk kell a fejlécek olvasásához
  - I a fizikai réteg nem garantál hibamentességet, az adatkommunikáció réteg feladata a hibaelzárás illetve

# Bájt alapú: Karakterszámlálás

- a keretben lévő karakterek számának megadása a keret fejlécében lévő mezőben
- a vevő adatkapcsolati rétege tudni fogja a keret végét
- *Probléma:* nagyon érzékeny a hibára a módszer



# Bájt alapú: Bájt beszúrás (Byte Stuffing)



- Egy speciális **FLAG** bájt (jelölő bájt) jelzi az adat keret elejét és végét
  - Korábban két speciális bájtot használtak: egyet a keret elejéhez és egyet a végéhez
- Probléma: Mi van, ha a **FLAG** szerepel az adat bájtok között is?
  - Szűrjunk be egy speciális **ESC** (Escape) bájtot az „adat” **FLAG** elé
  - Mi van ha **ESC** is szerpel az adathban?

# Bájt beszúrás példa

**Keretezendő adat**

H	E	L	L	O	[SPAC E]	[ESC]
---	---	---	---	---	-------------	-------



**Keretezett adat**

[FLAG ]	H	E	L	L	O	[SPAC E]	[ESC]	[ESC]	[FLAG ]
------------	---	---	---	---	---	-------------	-------	-------	------------

# Bit alapú: Bit beszúrás (Bit stuffing)

23

0111111  
0

Adat

0111111  
0

- minden keret speciális bitmintával kezdődik és végződik (hasonlóan a bájt beszúráshoz)
  - A kezdő és záró bitsorozat ugyanaz
  - Például: 01111110 a High-level Data Link Protocol (HDLC) esetén
- A Küldő az adatban előforduló minden 11111 részsorozat előtt 0 bitet szűr be
  - Ezt nevezzük bit beszúrásnak

# Példa bit beszúrásra

az átvitelre szánt bitsorozat bitbeszúrás előtt  
az átvitelre szánt bitsorozat  
bitbeszúrás után (fejléc nélkül)  
a vevőnél megjelenő üzenet a  
redundáns bitek eltávolítása után

0►1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

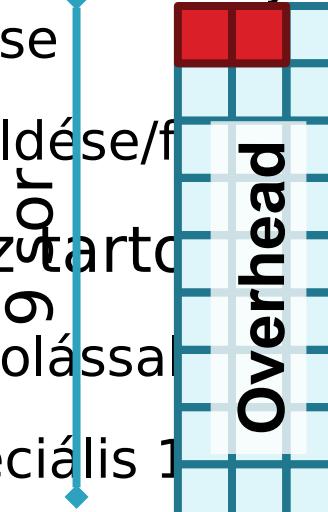
0►1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Beszúrt bitek

# Óra alapú keretezés: SONET

- **Synchronous Optical Network**
  - Nagyon gyors optikai kábelen való átvitel
  - STS- $n$ , e.g. STS-1: 51.84 Mbps, STS-768: 36.7 Gbps
- Az STS-1 keretei rögzített mérettel rendelkeznek
  - 9\*99 – 910 bájt → 810 bájt fogadása 90 órában a keret-kezdő

Speciális kezdő  
mintázat



90 órában a keret-kezdő

Payload/szállított adat

- A fizikai részhez tartozó:
  - A bitek NRZ kódolással
  - Payload egy speciális 1
  - A hosszú 0 és 1 sorozatok elkerülése végett

Hiba felügyelet

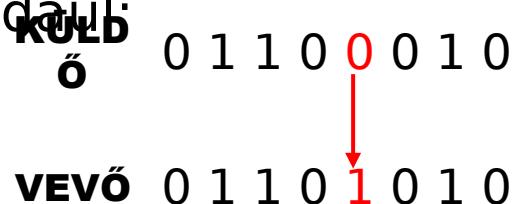
# Zaj kezelése

27

- A fizikai világ eredendően zajos
  - Interferencia az elektromos kábelek között
  - Áthallás a rádiós átvitelek között, mikrosütő, ...
  - Napviharok
- Hogyan detektáljuk a bithibákat az átvitelben?
- Hogyan állítsuk helyre a hibát?

# Bithibák definíciók és példák

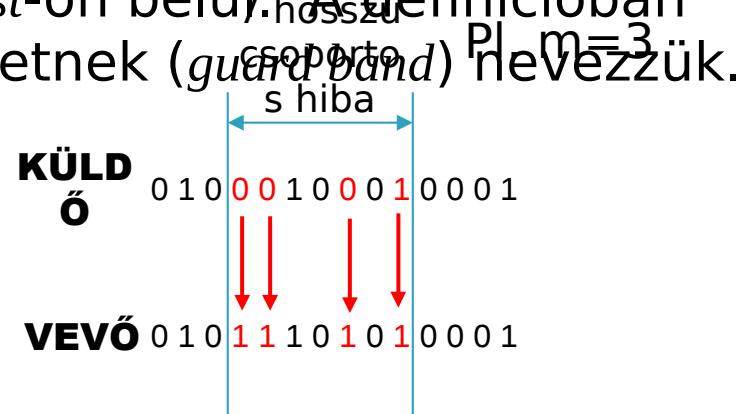
- *egyszerű bithiba* – az adategység 1 bitje nulláról egyre avagy egyről nullára változik. Például:



- *csoportos hiba* (angolul *burst error*) – Az átviteli csatornán fogadott bitek egy olyan folytonos sorozata, amelynek az első és utolsó szimbóluma hibás, és nem létezik ezen két szimbólummal határolt részsorozatban olyan  $m$  hosszú részsorozat, amelyet helyesen fogadtunk volna a hiba *burst*-ön belül. A definícióban használt  $m$  paramétert védelmi övezetnek (*guard band*) nevezik. (Gilbert-Elliott modell)



VEVŐ 0 1 0 1 1 1 0 1 0 1 0 0 0 1



# Naiv hibadetectálás

29

- Ötlet: küldjünk két kópiát minden egyes keretből
  - if (memcmp(frame1, frame2) != 0) { JAJ, HIBA TÖRTÉNT! }
- Miért rossz ötlet ez?
  - Túl magas ára van / a hatékonyság jelentősen lecsökken
  - Gyenge hibavédelemmel rendelkezik
    - Lényegében a duplán elküldött adat azt jelenti, hogy kétszer akkora esélye lesz a meghibásodásnak

# Paritás Bit

30

- ❑ Ötlet: egy extra bitet adunk a bitsorozathoz úgy, hogy az egyesek száma végül **páros** legyen

0101001 1101000 1011111 0001111 0110101  
Példa: 7-bites ASCII karakterek + 1 paritásbit  
10



- ❑ 1-bit hiba detektálható
- ❑ 2-bit hiba nem detektálható
- ❑ Nem megbízható burstös hibák esetén

# Hiba vezérlés

- Stratégiák
  - Hiba javító kódok
    - Előre hibajavítás
    - Forward Error Correction (FEC)
    - kevésbé megbízható csatornákon célszerűbb
  - Hiba detektálás és újraküldés
    - Automatic Repeat Request (ARQ)
    - hibás csatornának újraadásához

# Hiba vezérlés

## □ Célok

### □ Hiba detektálás

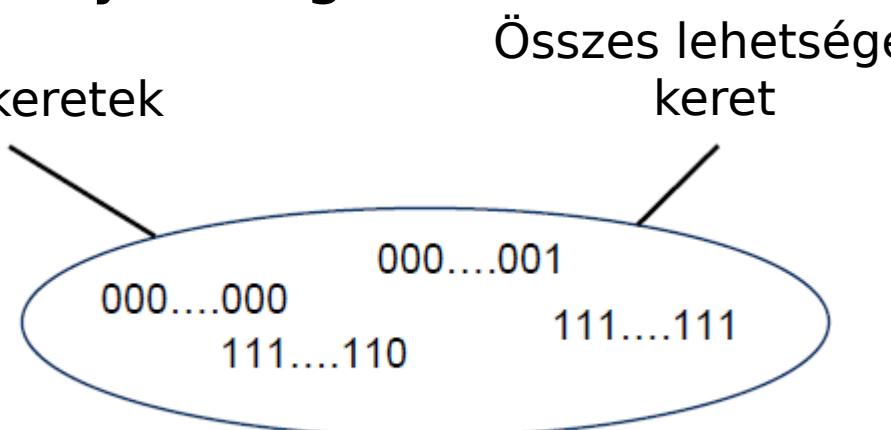
- javítással
  - Forward error correction
  - Javítás nélkül -> pl. eldobjuk a keretet
    - Utólagos hibajavítás
    - A hibás keret újraküldése

### □ Hiba javítás

- Hiba detektálás nélkül
  - Pl. hangátvitel

# Redundancia

- Redundancia szükséges a hiba vezérléshez
- Redundancia nélkül
  - 2m lehetséges üzenet írható le m biten
  - Mindegyik helyes (legal) üzenet és fontos adatot tartalmazhat
  - Ekkor minden hiba egy új helyes (legal) üzenetet eredményez
    - A hiba felismerése lehetővé teszi a helyes üzenetet
- Hogyan ismerjük fel



# Redundancia

## □ Egy keret felépítése:

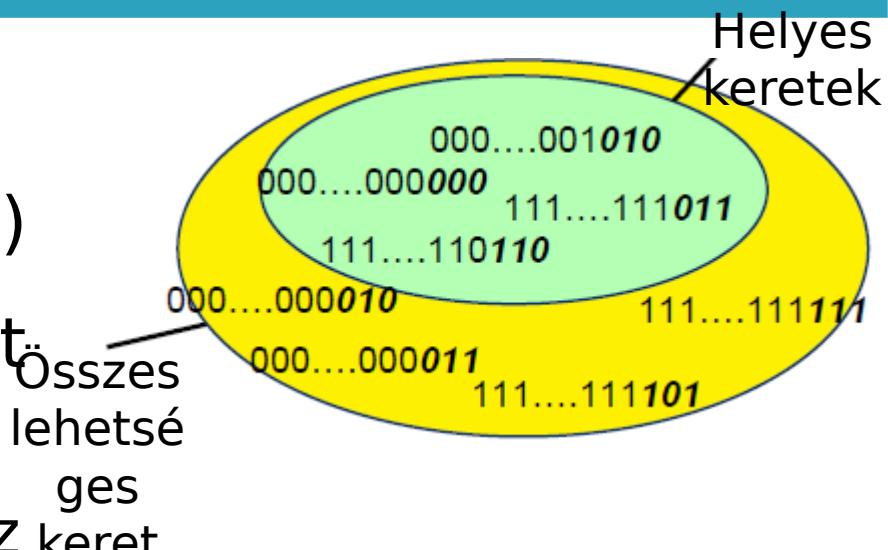
■ m adat bit (ez az üzenet)

■ r redundáns/ellenőrző bit

■ Az üzenetből számolt, új információt nem hordoz keret

■ A teljes keret hossza:  $n = m + r$

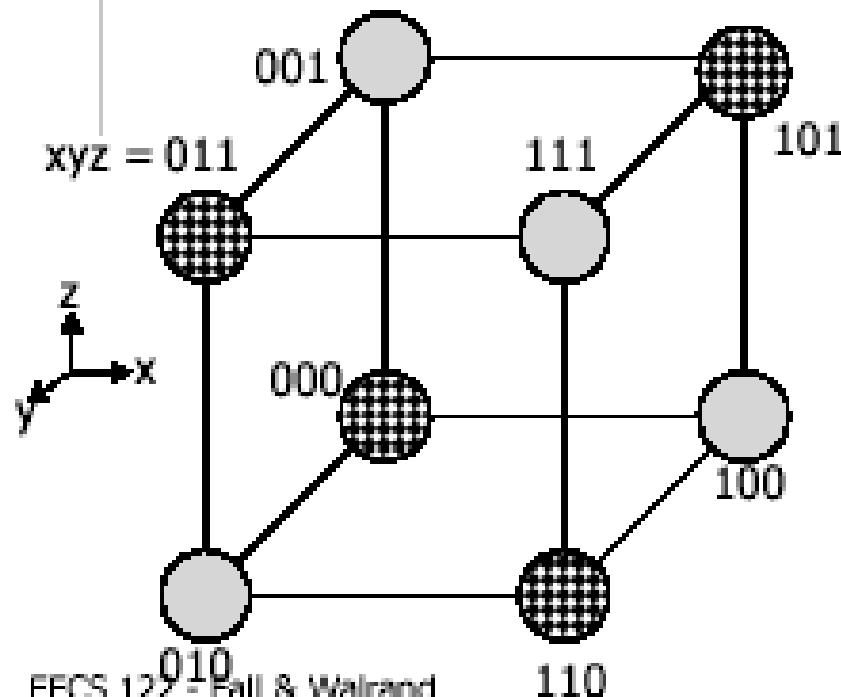
□ Az így előálló n bites bitsorozatot n hosszú kódszónak nevezzük!



# Error Control Codes

## How Codes Work: Words and Codewords

- ◆ Code = subset of possible words: Codewords
- ◆ Example:
  - 3 bits => 8 words; codewords: subset



Words:  
000, 001, 010, 011  
100, 101, 110, 111

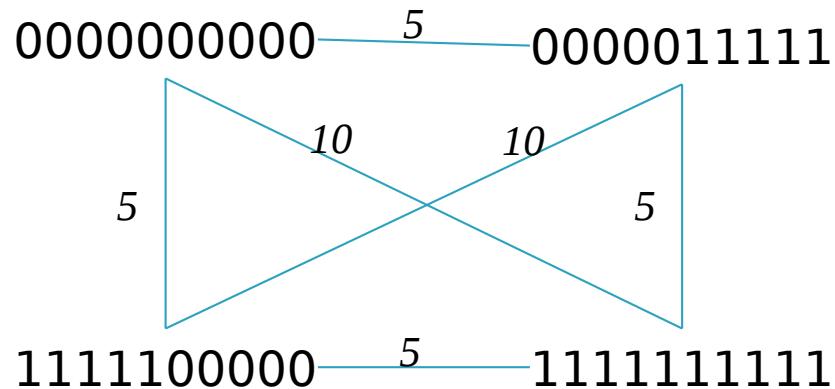
Code:  
000, 011, 101, 110

Send only codewords

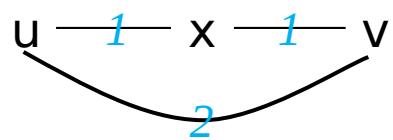
# Elméleti alapok



# Példa Hamming távolságra



# Hamming távolság használata

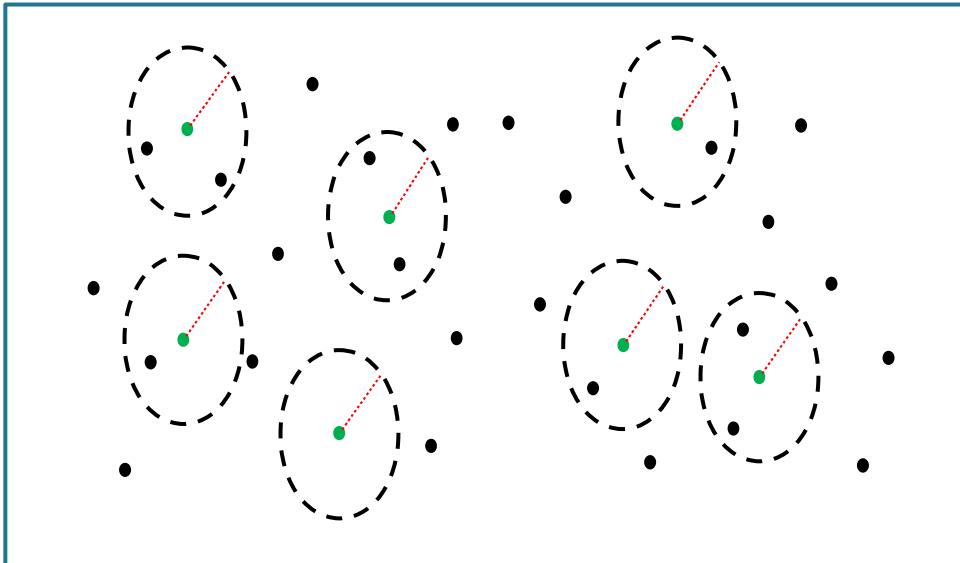


# Hamming korlát bináris kódkönyvre 1/3

# Hamming korlát bináris kódkönyvre 2/3

# Hamming korlát bináris kódkönyvre 3/3

□



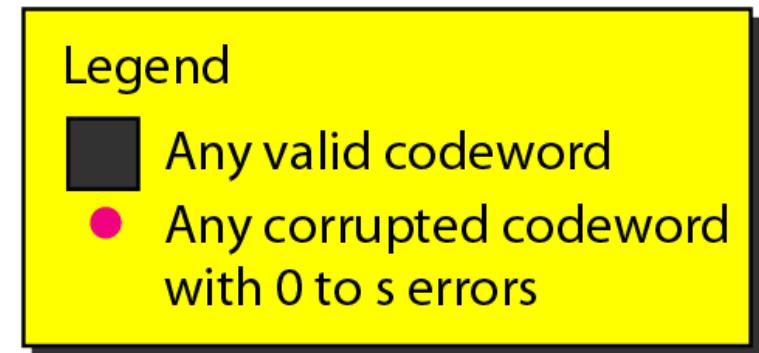
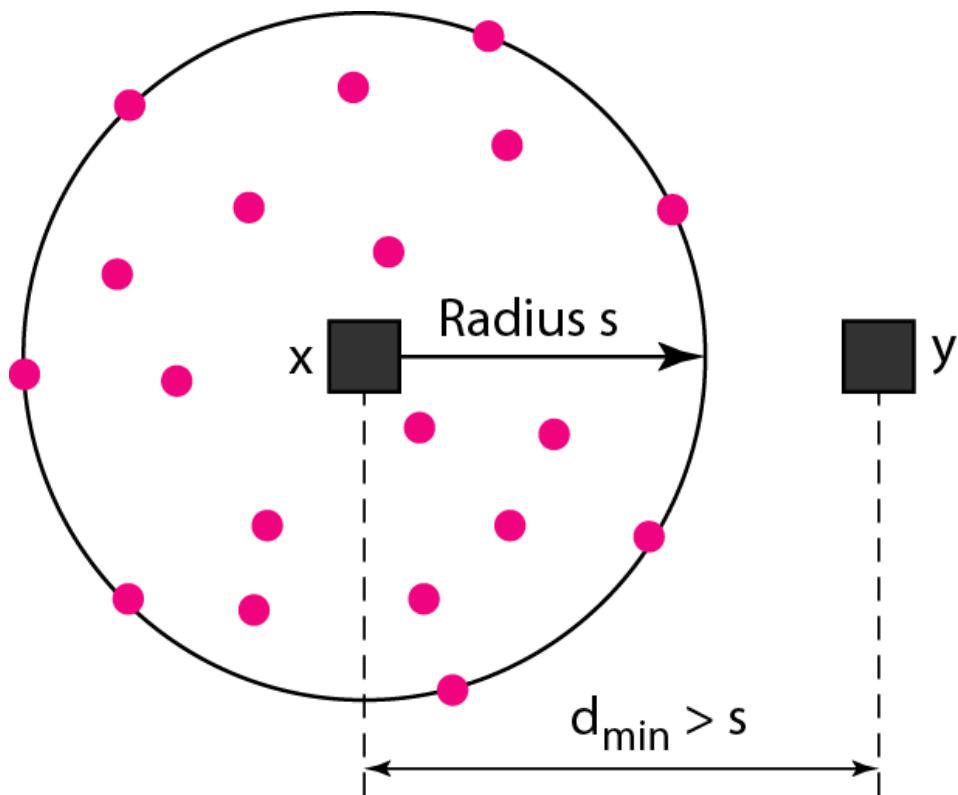
## Jelmagyarázat

- Kódszó
- Bitszó, amely nem kódszó

# Hibafelismerés és javítás Hamming távolsággal

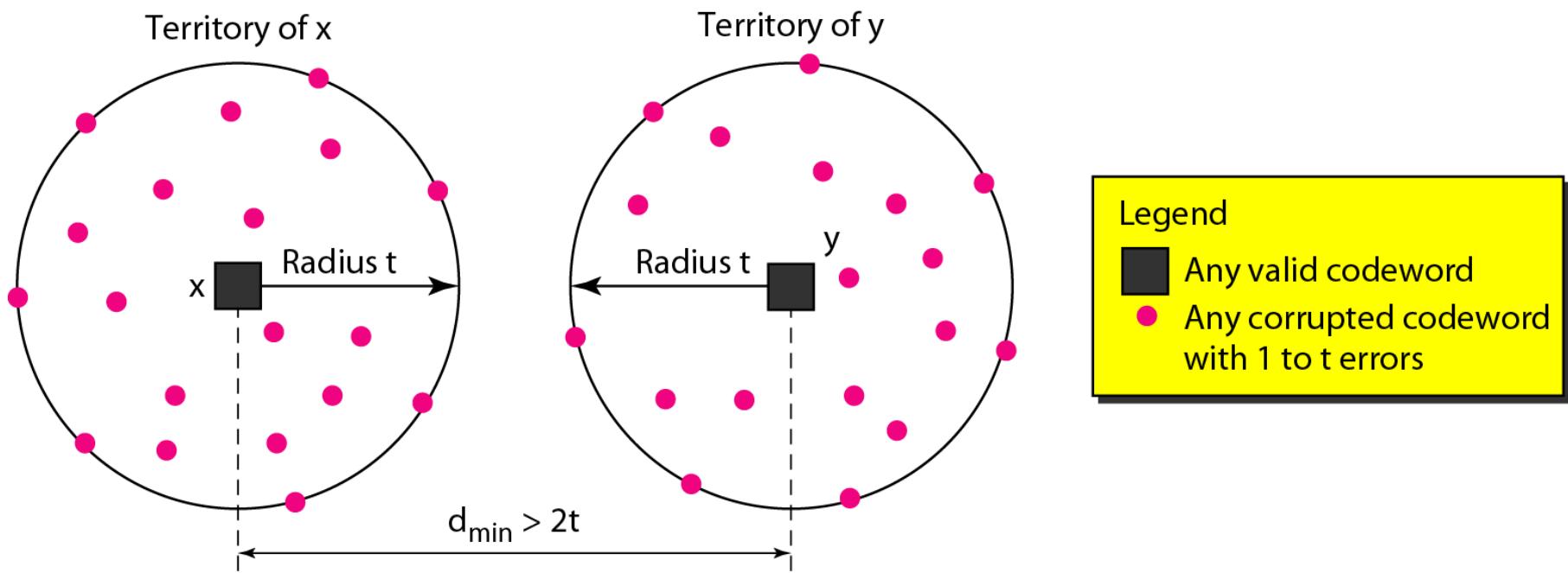
# Hiba felismerés

d bithiba felismeréséhez legalább  $d+1$  Hamming távolságú kód szükséges.



# Hiba javítás

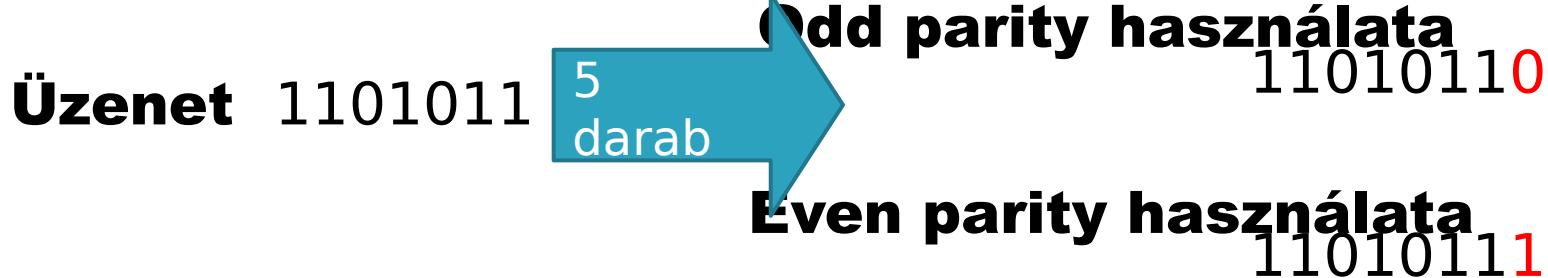
$d$  bithiba javításához legalább  $2d+1$  Hamming-távolságú kód szükséges.



# Újra a paritás bit használata

1/4

- a paritásbitet úgy választjuk meg, hogy a kódszóban levő 1-ek száma páros (vagy páratlan)
  - **Odd parity** - ha az egyesek száma páratlan, akkor 0 befűzése; egyébként 1-es befűzése
  - **Even parity** - ha az egyesek száma páros, akkor 0 befűzése; egyébként 1-es befűzése



# Paritás bit használata 2/4

## Egy paritást használó módszer (*Hamming*)

- a kódszó bitjeit számozzuk meg 1-gyel kezdődően;
- 2 egészhatvány sorszámú pozíciói lesznek az ellenőrző bitek, azaz  $1, 2, 4, 8, 16, \dots$ ;
- a maradék helyeket az üzenet bitjeivel töltjük fel;
- mindegyik ellenőrző bit a bitek valamelyen csoportjának a paritását állítja be párosra (vagy páratlanra)
- egy bit számos paritásszámítási csoportba tartozhat:
  - pozíciót írunk fel kettő hatványok összegeként

# Paritás bit használata - példa

3/4

ASCII karakter	ASCII decimális	Üzenet forrás bitjei	Az előállít kódszavak
E	69	1000101	<b>10100000101</b>
L	76	1001100	<b>10110011100</b>
T	84	1010100	<b>00110101100</b>
E	69	1000101	<b>10100000101</b>
	32	0100000	<b>10001100000</b>
I	73	1001001	<b>11110011001</b>
K	75	1001011	<b>00110010011</b>

# Paritás bit használata 4/4

- a vevő az üzenet megérkezésekor 0-ára állítja a számlálóját, ezt követően megvizsgálja a paritás biteket, ha a  $k$ -adik paritás nem jó, akkor a számlálóhoz ad  $k$ -t
- Ha a számláló 0 lesz, akkor érvényes kódszónak tekinti a vevő a kapott üzenetet; ha a számláló nem nulla, akkor a hibás bit sorszámát tartalmazza, azaz ha a fogadott üzenet első, a második 4 és nyolcadik bit helyében, akkor a megváltozott bit a tizenegyedik.

Fogadott  $L$  karakterek: ~~1010010101~~ Számláló != 0  
Fogadott  $L$  karakterek: ~~1110011100~~ Számláló = 2

Köszönöm a figyelmet!

# Számítógépes Hálózatok

## **4. Előadás: Adatkapcsolati réteg II.**

Based on slides from

Hibajelző kódok

# Hibajelző kódok


$$\begin{array}{r} 11110000 \\ - 10100110 \\ \hline 01010110 \end{array}$$
$$\begin{array}{r} 10011011 \\ + 11001010 \\ \hline 01010001 \end{array}$$

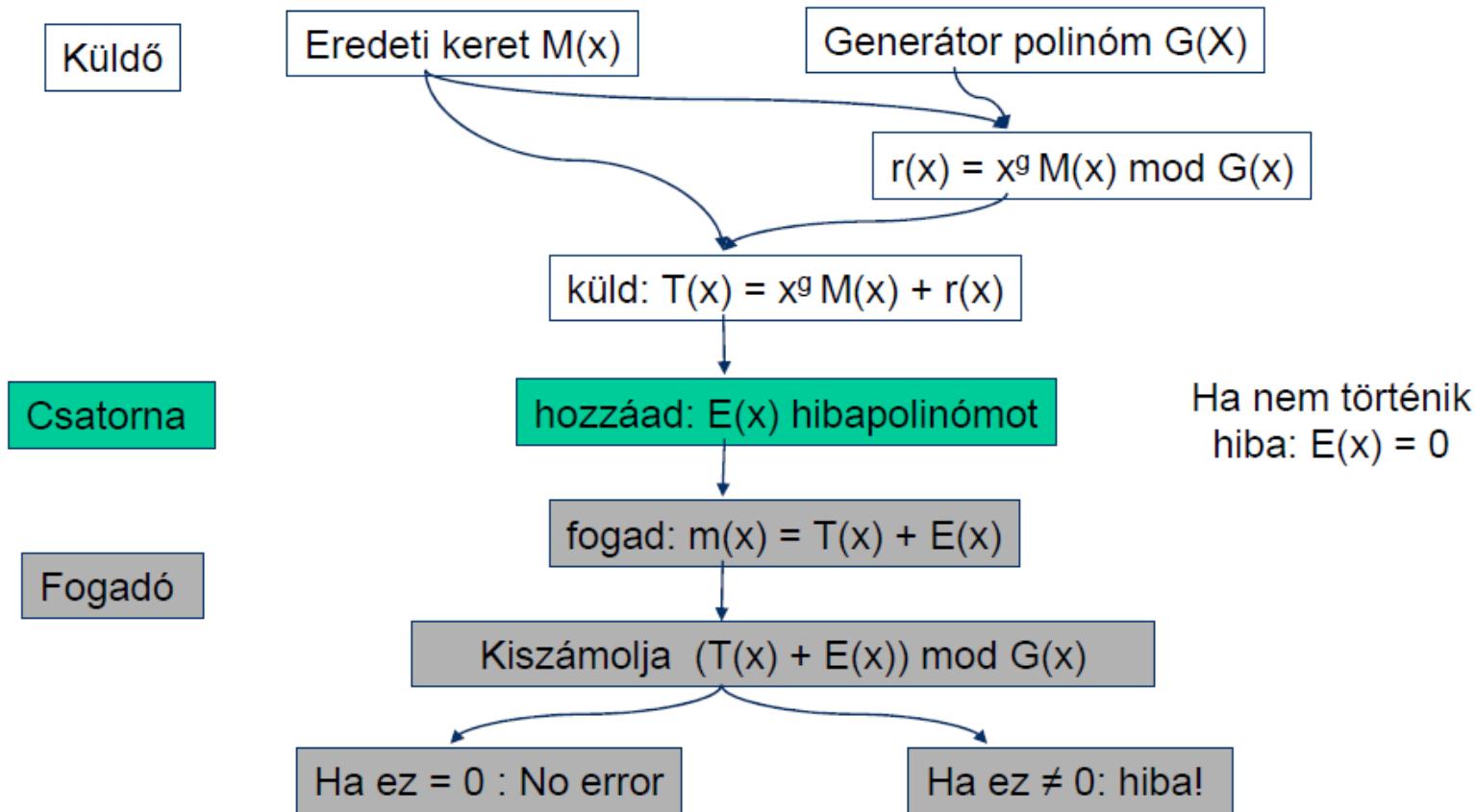
# CRC



# CRC áttekintés

5

- Forrás: Dr. Lukovszki Tamás fóliái

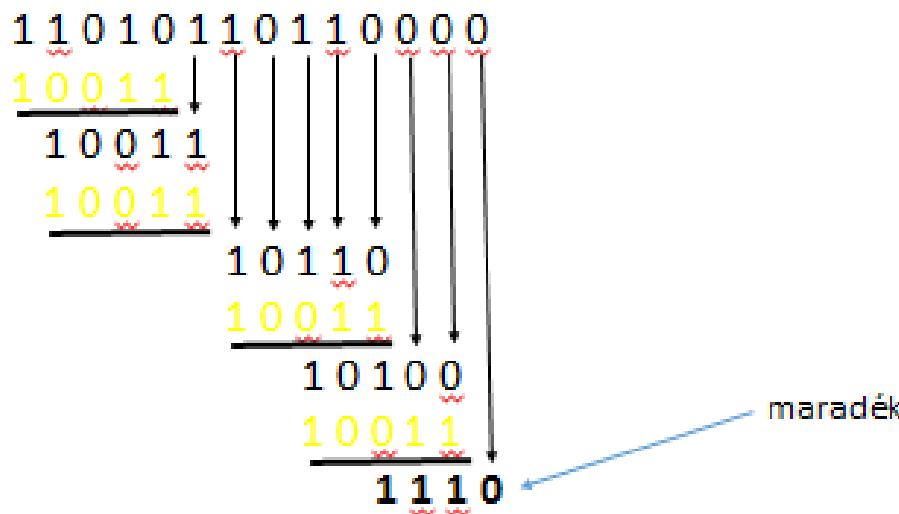


# Példa CRC számításra

**Keret:** 1101011011

# Generátor: 10011

**A továbbítandó üzenet:** 11010110111110



# CRC áttekintés



# CRC a gyakorlatban





## Forgalomszabályozás

# Forgalomszabályozás

- gyors adó lassú vevő problémája (*elárasztás*)
- még hibamentes átvitel esetén se lesz képes a vevő kezelní a bejövő kereteket

## Megoldási lehetőségek

1. visszacsatolás alapú forgalomszabályozás (avagy angolul *feedback-based flow control*)
  - engedélyezés
2. Sebesség alapú forgalomszabályozás (avagy angolul *rate-based flow control*)
  - protokollba integrált sebességkorlát
  - az adatkapcsolati réteg nem használja

# Elemi adatkapcsolati protokollok

## Feltevések

- A fizikai, az adatkapcsolati és a hálózati réteg független folyamatok, amelyek üzeneteken keresztül kommunikálnak egymással.
- Az A gép megbízható, összeköttetés alapú szolgálat alkalmazásával akar a B gépnek egy hosszú adatfolyamot küldeni. (Adatok előállítására sosem kell várnia A gépnek.)
- A gépek nem fagynak le.
- Adatkapcsolati fejrészben vezérlési információk; adatkapcsolati lábrészben ellenőrző összeg

## Kommunikációs fajták

- *szimplex kommunikáció* – a kommunikáció pusztán egy irányba lehetséges
- *fél-duplex kommunikáció* – minden két irányba folyhat

# Korlátozás nélküli szimplex protokoll

a legegyszerűbb protokoll („utópia”)

## A környezet

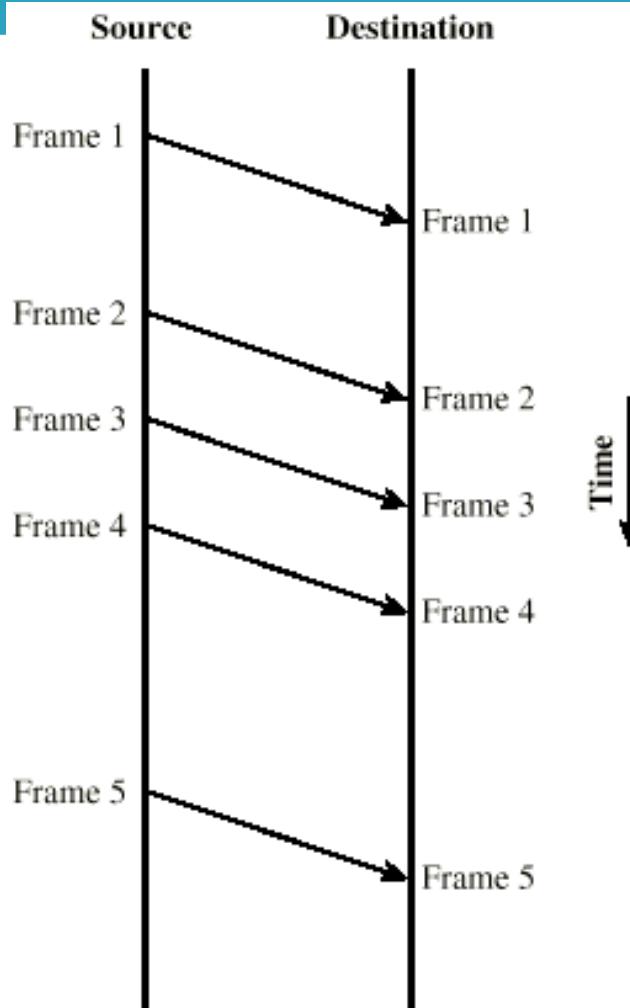
- mind az adó, mind a vevő hálózati rétegei mindenkorban készen állnak;
- a feldolgozási időktől eltekintünk;
- végtelen puffer-területet feltételezünk;
- Az adatkapcsolati rétegek közötti kommunikációs csatorna sosem rontja vagy veszíti el a kereteket;

## A protokoll

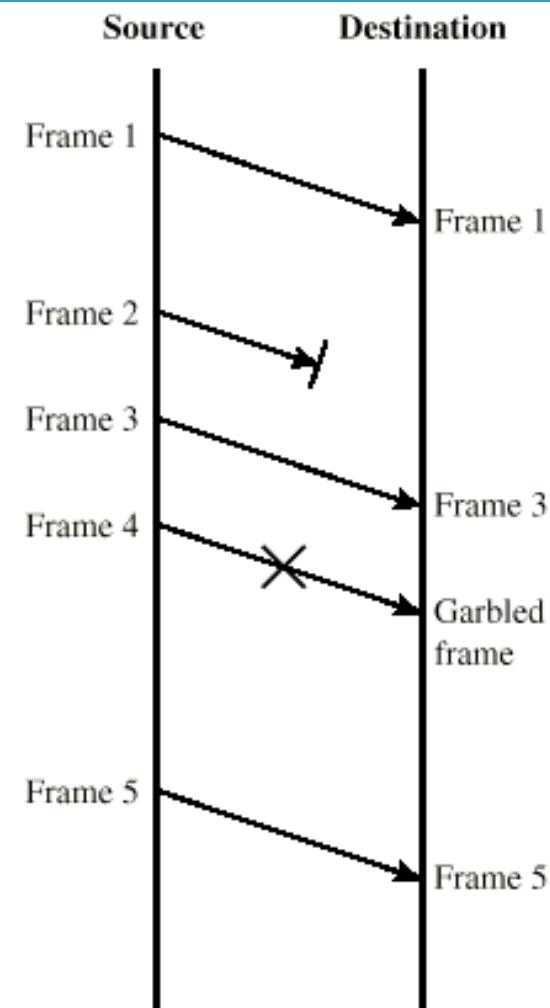
- résznevők: *küldő* és *vevő*;
- nincs sem sorszámozás, sem nyugta;
- küldő végtelen ciklusban küldi kifele a kereteket

# Átvitel hiba nélkül és hibával

13



(a) Error-free transmission



(b) Transmission with losses and errors

# Szimplex megáll-és-vár protokoll

(stop-and-wait protocol)

# Szimplex protokoll zajos csatornához

# Megáll-és-vár

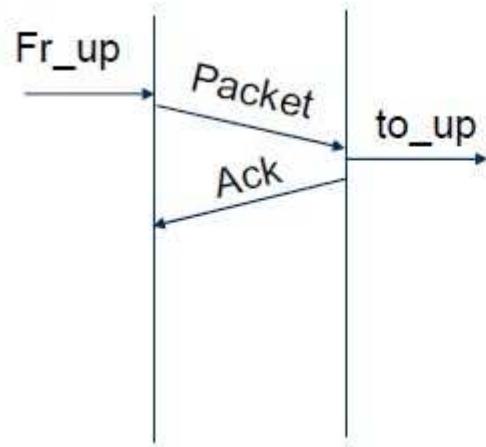
16

Egyszerű de nem hatékony  
nagy távolságok és nagy  
sebességű hálózat esetén.

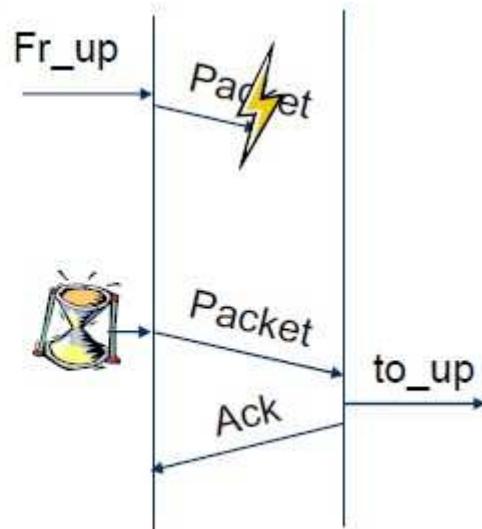
Küldhetnénk egymás után  
folyamatosan???

# Mi is a probléma?

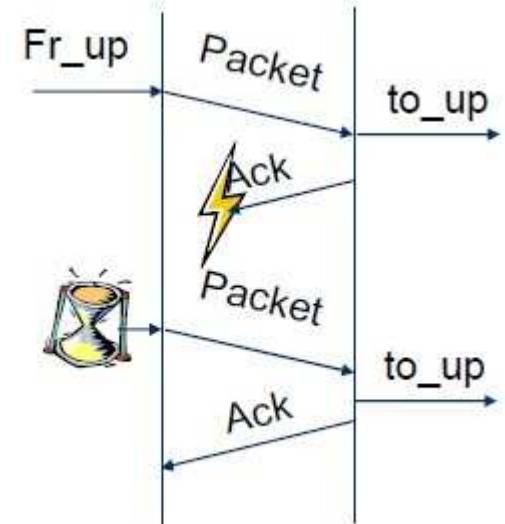
Általában



Csomagvesztés esetén

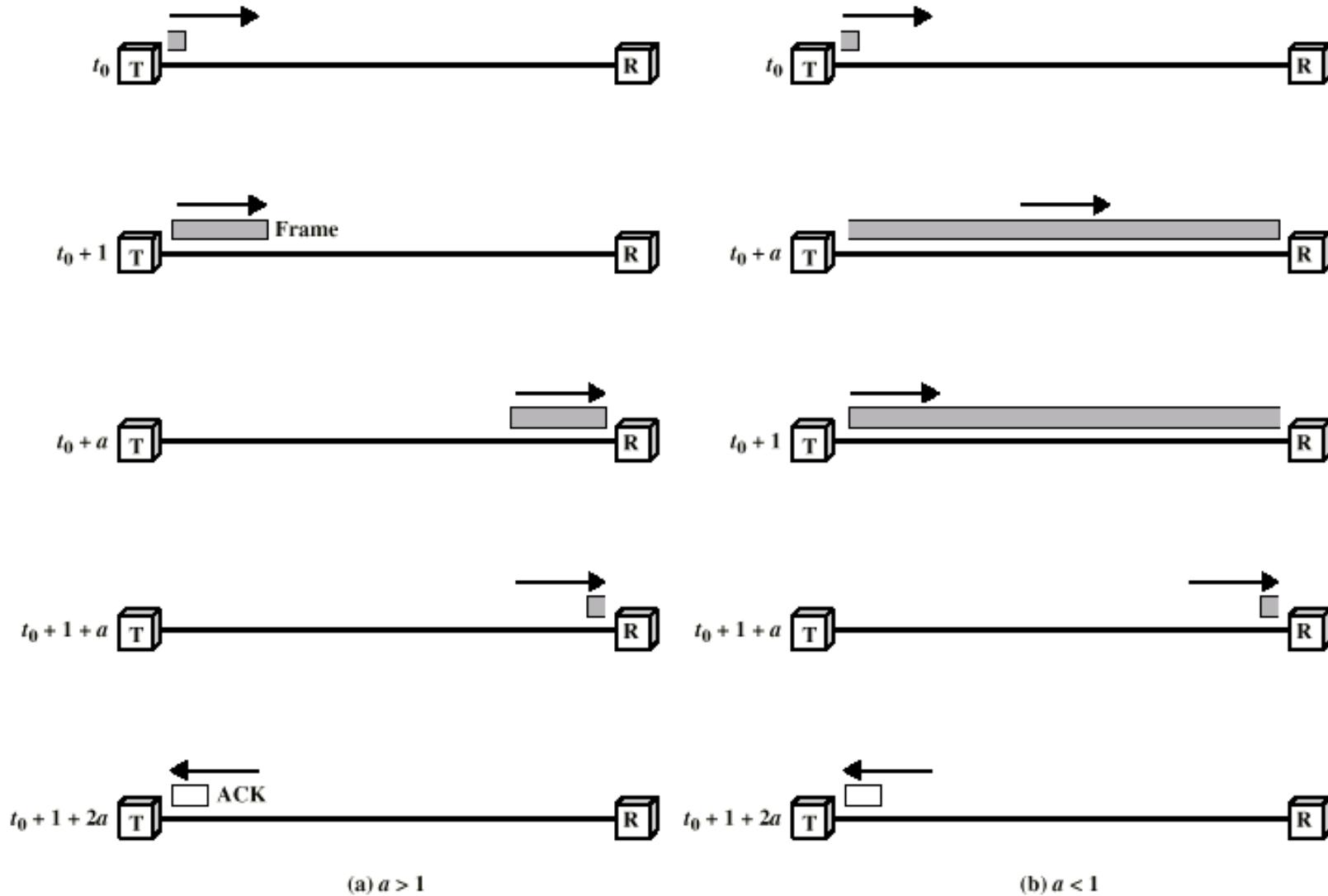


ACK vesztés esetén



# Csatorna kihasználtság

18

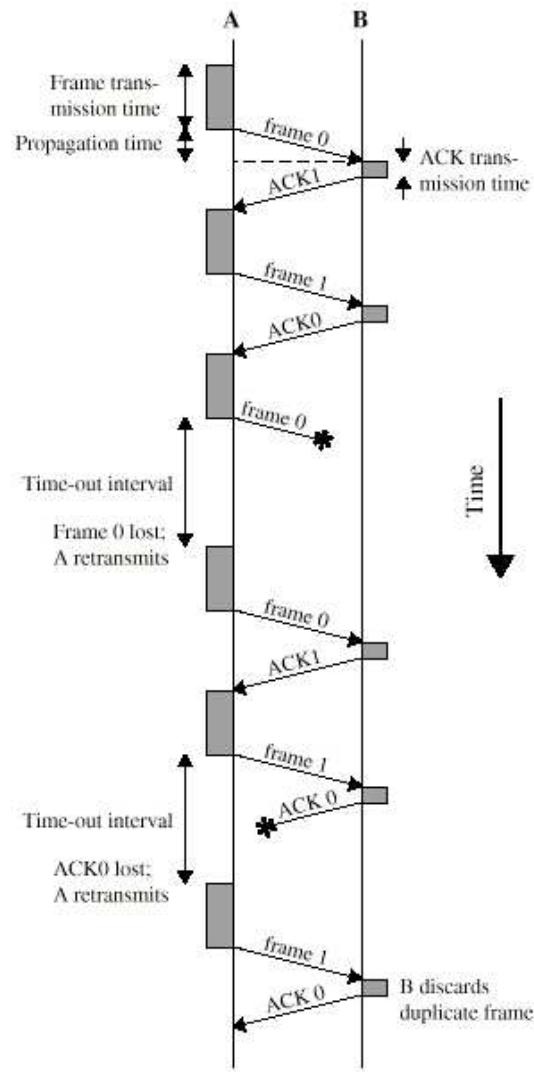


# Alternáló-bit protokoll (ABP)



# ABP

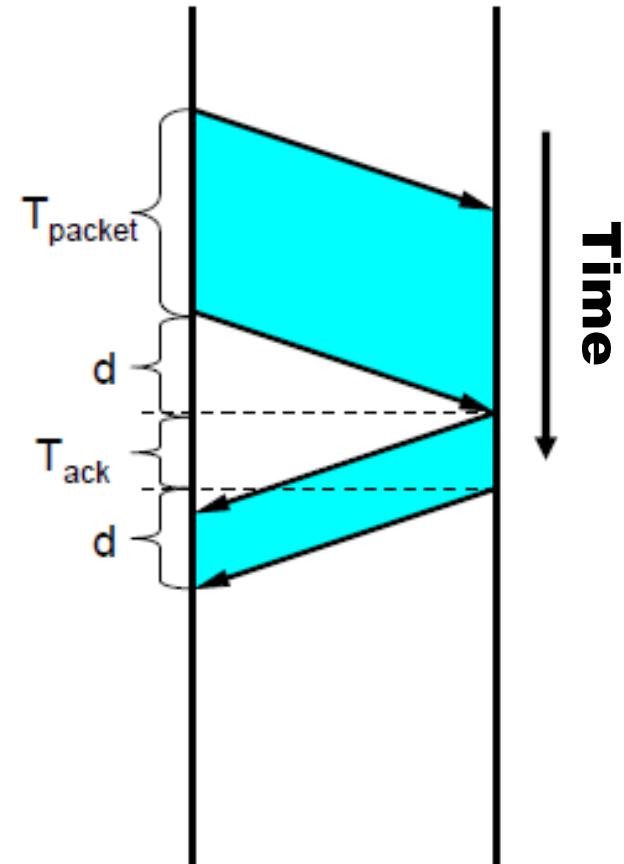
20



# ABP – Csatorna kihasználtság

- Kihasználtság ( $\eta$ ) a következő két elem aránya

- A csomag elküldéséhez szükséges idő ( $T_{packet}$ )
  - Az idő, ami a következő keret küldéséig eltelik
    - Az ábrán:  $(T_{packet} + d + T_{ack} + d)$

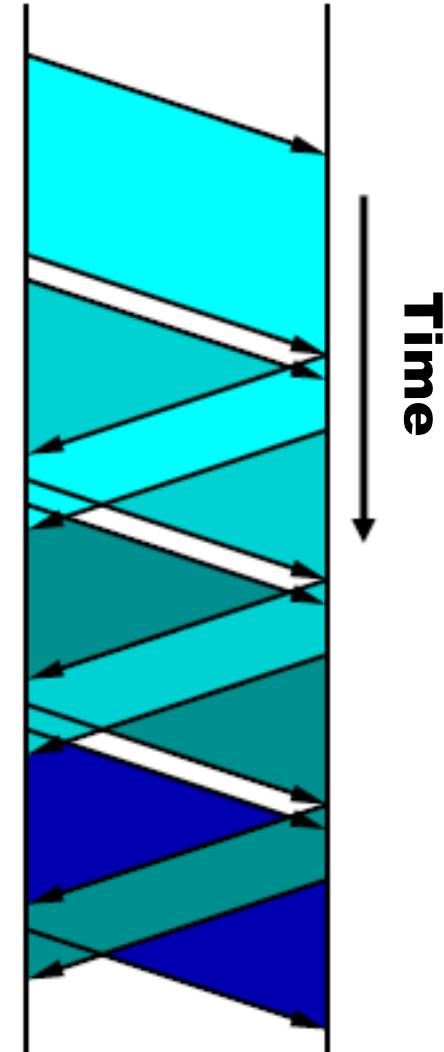


- ABP esetén:

- $\eta = T_{packet} / (T_{packet} + d)$

# Hogyan javítsunk a hatékonyságon?

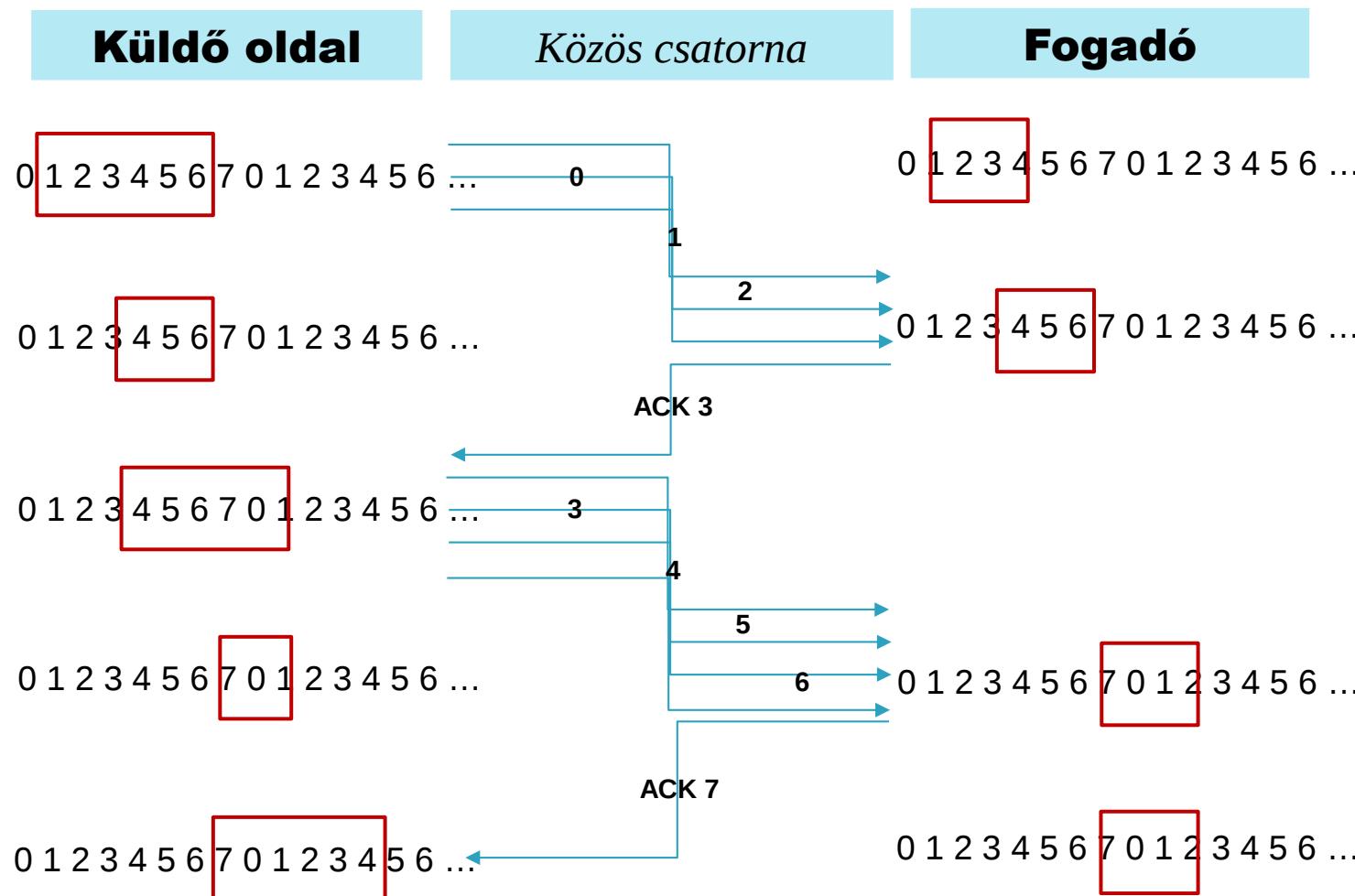
- A küldők egymás után küldik a kereteket
  - Több keretet is kiküldünk, nyugta megvárása nélkül.
  - Pipeline technika
- ABP kiterjesztése
  - Sorszámok bevezetésével



# Csúszó-ablak protokollok 1/2



# Példa 3-bites csúszó-ablak protokollra



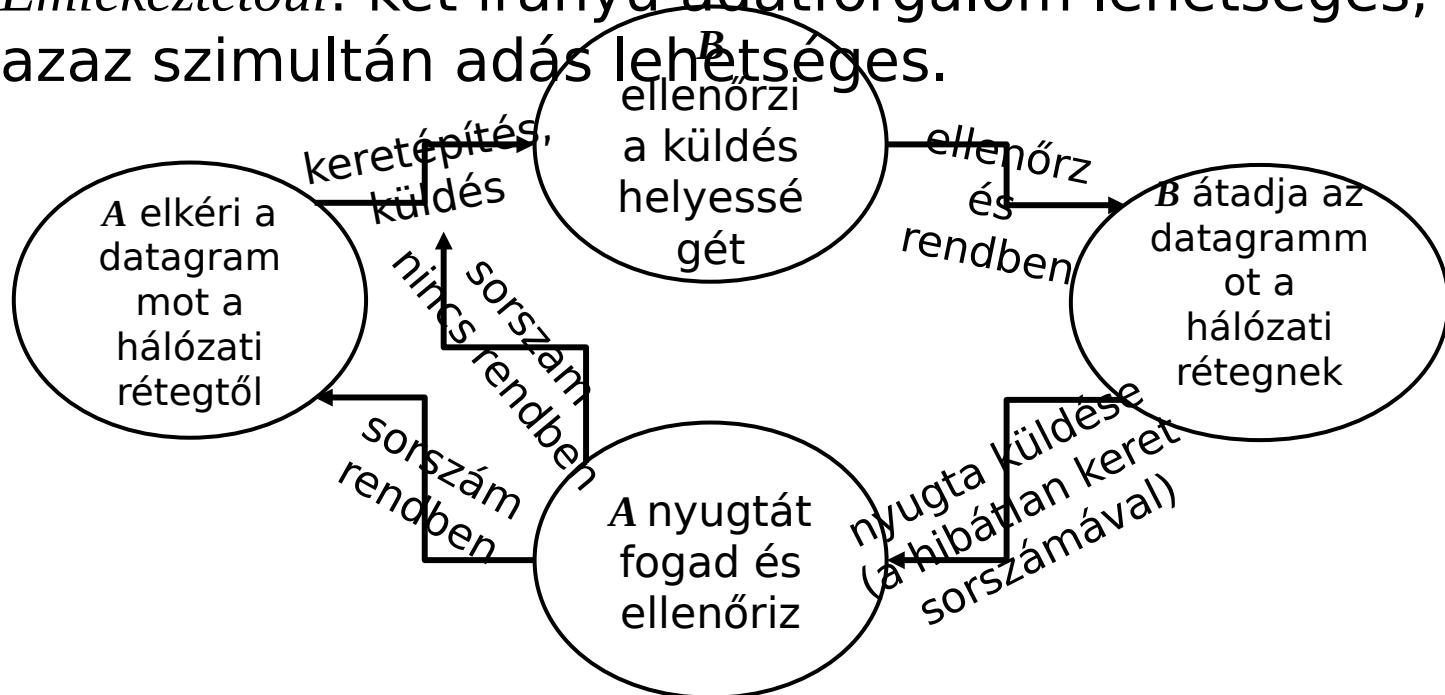
# Csúszó-ablak protokollok 2/2



# Egybites csúszó-ablak protokoll állapotátmenetei

## Környezet

- A maximális ablak méret legyen 1.
- *Emlékeztetőül:* két irányú adatforgalom lehetséges, azaz szimultán adás lehetséges.



# Pipelining

- Eddig feltételeztük, hogy *a keret vevőhöz való megérkezéséhez és a nyugta visszaérkezéséhez együttesen szükséges idő elhanyagolható.*
  - a nagy RTT a sávszélesség kihasználtságra hatással lehet
  - **Ötlet:** egyszerre több keret küldése
  - Ha az adatsebesség és az RTT szorzata nagy, akkor érdemes nagyméretű adási ablakot használni.  
*(pipelining)*
- Mi van ha egy hosszú folyam közepén történik egy keret hiba?
  1. „visszalépés N-nel”, avagy angolul *go-back-n*
  2. „szelektív ismétlés”, avagy angolul *selective-repeat*

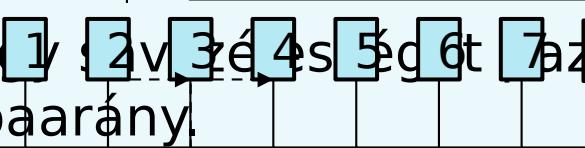
# „visszalépés N-nel” stratégia

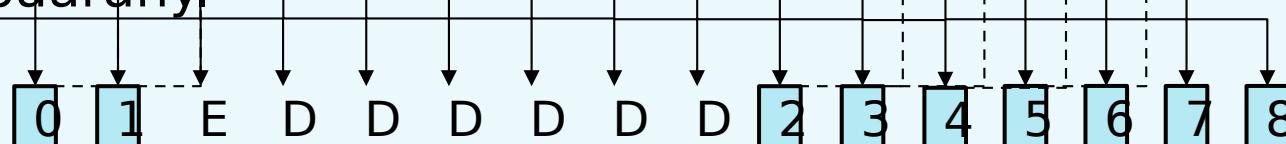
## Stratégia lényege

- Az összes hibás keret utáni keretet eldobja és nyugtát sem küld róluk.
- Mikor az adónak lejár az időzítője, akkor újraküldi az összes nyugtáztalan keretet, kezdve a sérült vagy elveszett kerettel.

## Következmények

- Egy méretű vételi ablakot feltételezünk.  

-   
hibaarány



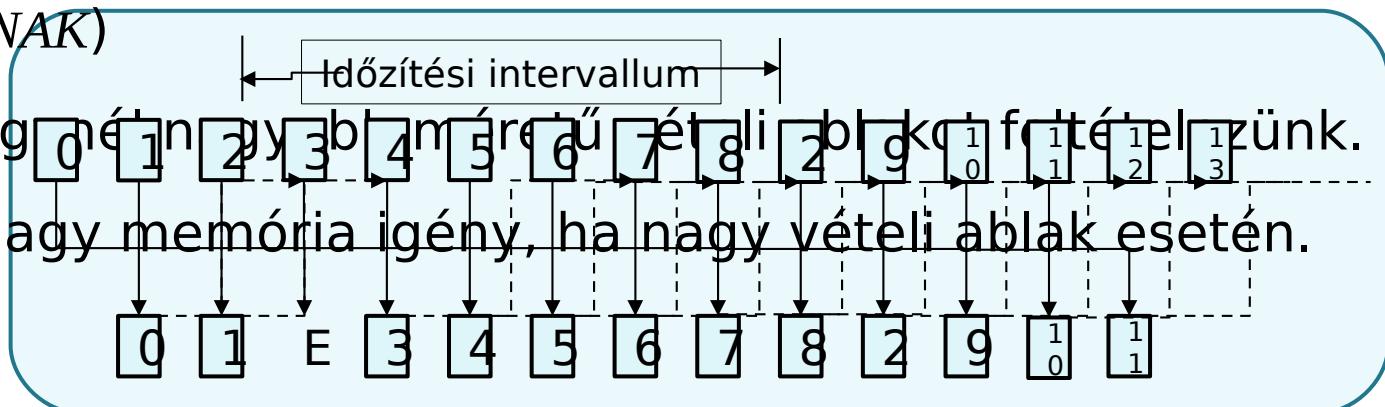
# „szelektív ismétlés” stratégia

## Stratégia lényege

- A hibás kereteket eldobja, de a jó kereteket a hibás után puffereli.
- Mikor az adónak lejár az időzítője, akkor a legrégebbi nyugtázatlan keretet küldi el újra.

## Következmények

- Javíthat a hatékonyságon a negatív nyugta használata.  
(NAK)
- Egészén igyeben mérőrételi blokkfuttatásunk.
- Nagy memória igény, ha nagy vételi ablak esetén.



□ Példák az  
adatkapcsolati réteg  
protokolljaira

# HDLC 1/4

31

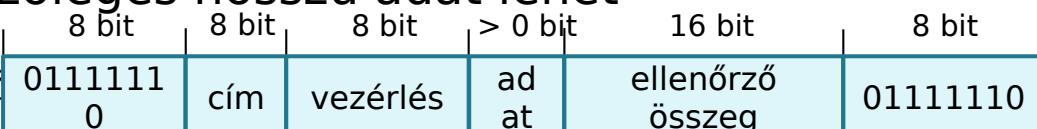
## Áttekintés

- IBM mainframe világában használták először az SDLC protokoll;
  - ANSI szabvány változat ADCCP
  - ISO szabvány változat HDLC
  - Később a CCITT adoptálta a LAP és a LAPB protokolljához.
- A különböző szabványok közös elveken nyugodnak
  - Bitorientáltság (bitbeszúrás alkalmazása)
  - Apróbb sajátosságok

# HDLC 2/4

32

## Általános keretfelépítés

- *cím* mező
  - több vonallal rendelkező terminálok esetén van jelentősége,
  - pont-pont kapcsolatnál parancsok és válaszok megkülönböztetésére használják
- *vezérlés* mező
  - sorszámozás, nyugtázás és egyéb feladatok ellátására
- *adat* mező
  - tetszőleges hosszú adat lehet
- *ellenőrző összeg*
  - The diagram shows a horizontal line divided into six fields by vertical lines. From left to right: 1) An 8-bit field labeled 'cím' (address) containing binary '0111111'. 2) An 8-bit field labeled 'vezérlés' (control) containing binary '0'. 3) A variable-length field labeled 'adat' (data) containing binary '1111111111111111'. 4) A 16-bit field labeled 'ellenőrző összeg' (checksum) containing binary '01111110'. 5) An 8-bit field labeled 'trailer' containing binary '11111111'.
    - CRC kontrollösszeg a CRC-CCITT generátor polinom

# HDLC 3/4

33

- 3 bites csúszó-ablak protokoll használ a sorszámozáshoz
- Három típusú keretet használ:
  - információs
  - felügyelő

## Típusok

0. típus - nyugtakeret (RECEIVE READY);

1. típus - negatív nyugtakeret (REJECT);

2. típus - vételre nem kész, amely nyugtáz minden keretet a

# HDLC 4/4

34

## **Legelterjedtebb parancsok számoszatlan keretek esetén**

- DISC – a csatlakozás bontására vonatkozó szándék bejelentésére
- SNRM – visszacsatlakozó állomás bejelentse a jelenlétét (*aszinkron*)
- SABM – alaphelyzetbe állítja a vonalat (*egyenrangúság*)
- Kiterjesztett keretformátum (7 bites sorszám) engedélyezésekor: SAMBE, SNRME.
- FRMR – keretelutasítás illegális keretek jelzésére

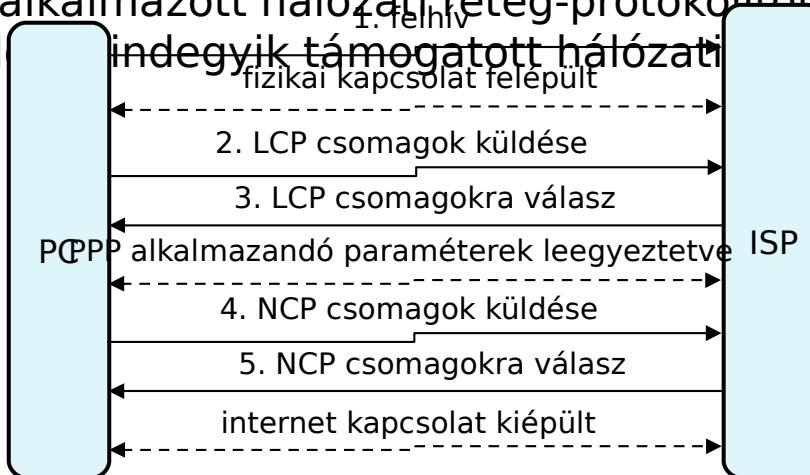
# PPP 1/2

35

- RFC 1661, 1662, 1663

- **Három dolgot biztosít**

1. Keretezési módszert (egyértelmű kerethatárok).
2. Kapcsolatvezérlő protokollt a vonalak felélesztésére, tesztelésére, az opció egyeztetésére és a vonalak elengedésére. LCP protokoll. (szinkron/aszinkron áramkörök, bájtalapú/bitalapú kódolás)
3. Olyan módot a hálózati réteg-opciók megbeszélésére, amely független az alkalmazott hálózati réteg-protokolloktól. Külön-külön NCP protokollon keresztül. ~~independently of the underlying network protocols~~ ghez.



# PPP 2/2

36

- bájt alapú keretszerkezet, azaz a legkisebb adategység a bájt
- Alapértelmezésben nem biztosít megbízható átvitelt.

## □ **Mezők fontosabb tulajdonságai**

- Vezérlő mező alapértéke a számoszatlan keretet jelzi
- Protokoll mezőben protokoll kód lehet az LCP, NCP, IP, IPX, AppleTalk vagy más protokollhoz.

### ■ 0-s bittel kezdődő kódok a hálózati

Jelző 0111111 0	Cím 1111111	Vezérlő 0000001 1	Protokoll	Adatmez ő	Változó ID VNC	2 vagy 4 bitb	1

egyeztetést igénylő protokollokhoz  
tartoznak (LCP, NCP, stb.)

# Ethernet keret

802.3 Ethernet frame structure									
Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interframe gap	
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	42[note 2]–1500 octets	4 octets	12 octets	
					64–1522 octets				
					72–1530 octets				
					84–1542 octets				

Köszönöm a figyelmet!

# Számítógépes Hálózatok

**5. Előadás: Adatkapcsolati  
réteg III.**

Based on slides from

- Közeg hozzáférés vezérlése
- Media Access Control (MAC)

# Mi az a közeg hozzáférés ?

3

- Ethernet és a Wifi is többszörös hozzáférést biztosító technológiák
  - Az átviteli közegen több résztvevő osztozik
    - Adatszórás (broadcasting)
  - Az egyidejű átvitel **ütközést** okot
    - Lényegében meghiúsítja az átvitelt
- Követelmények a Media Access Control (MAC) protokolljaival szemben
  - Szabályok a közeg megosztására
  - Stratégiák az ütközések detektálásához.

# MAC alréteg

4

- Eddigi tárgyalásaink során pont-pont összeköttetést feltételeztünk.
- Most az adatszóró csatornát (angolul *broadcast channel*) használó hálózatok tárgykörével foglalkozunk majd.
  - **Kulcskérdez:** *Melyik állomás kapja a csatornahasználat jogát?*
  - A csatorna kiosztás történhet:
    1. statikus módon (FDM, TDM)
    2. dinamikus módon
      - a) verseny vagy ütközés alapú protokollok (ALOHA, CSMA, CSMA/CD)
      - b) verseny-mentes protokollok (bittérkép-alapú protokollok, bináris visszaszámlálás)

# Statikus csatornakiosztás

5

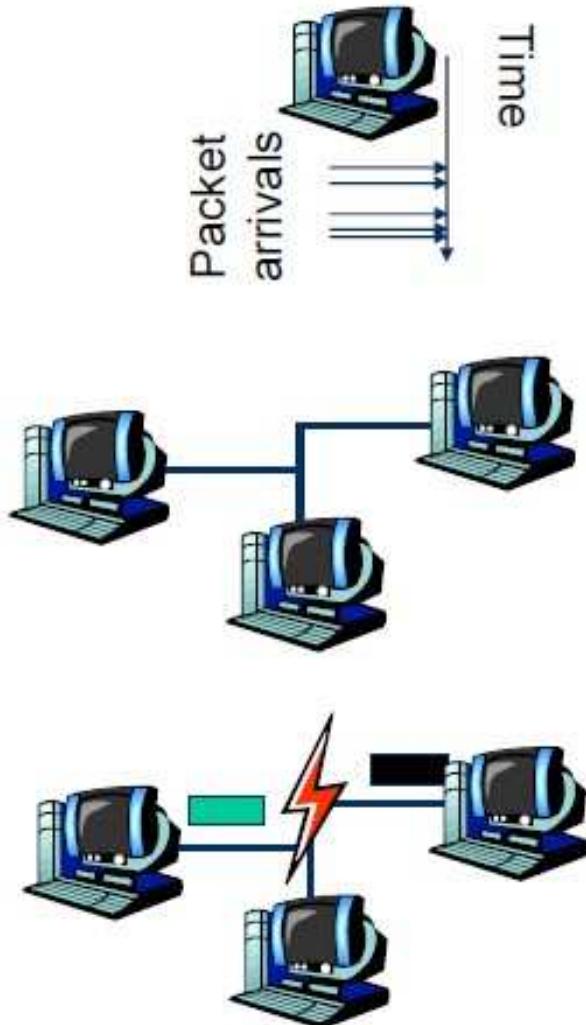
## Frekvenciaosztásos nyalábolás

- $N$  darab felhasználót feltételezünk, a sávszélet  $N$  egyenlő méretű sávra osztják, és minden egyes sávhoz hozzárendelnek egy felhasználót.
- Következésképpen az állomások nem fogják egymást zavarni.
- Előnyös a használata, ha fix számú felhasználó van és a felhasználók nagy forgalmi igényt támasztanak.
- Löketszerű forgalom esetén használata problémás.

## Időosztásos nyalábolás

- $N$  darab felhasználót feltételezünk, az időegységet  $N$  egyenlő méretű időrésre - úgynévezett *slot*-ra - osztják, és minden egyes réshez hozzárendelnek egy felhasználót.
- Löketszerű forgalom esetén használata nem hatékony.

# Dinamikus csatornakiosztás



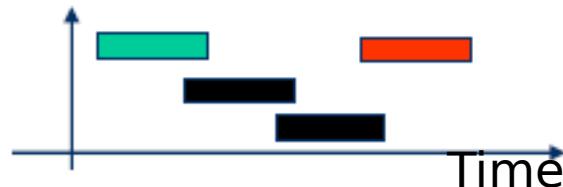
## 1. Állomás modell

- N terminál/állomás
- Annak a valószínűsége, hogy  $\Delta t$  idő alatt csomag érkezik  $\lambda \Delta t$ , ahol  $\lambda$  az érkezési folyam rátája.

## 2. Egyetlen csatorna feltételezés

- minden állomás egyenrangú.
- minden kommunikáció

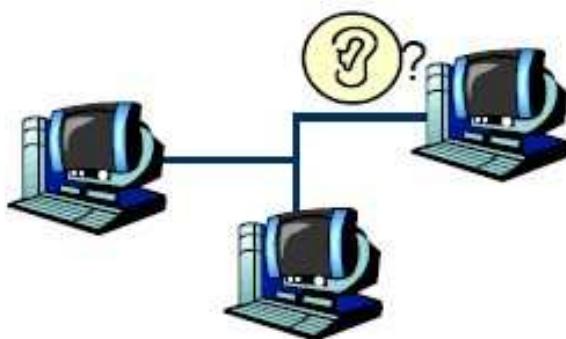
# Dinamikus csatornakiosztás



## Használt időmodell

Kétféle időmodellt különböztetünk meg:

- a) **Folytonos** - Mindegyik állomás tetszőleges időpontban megkezdheti a küldésre kész keretének sugárzását.
- b) **Diszkrét** - Az időt diszkrét részekre osztjuk. Keret továbbítás csak időrés elején lehetséges. Az időrés lehet üres, sikeres vagy ütközéses.



**Vivőjel érzékelési képesség**

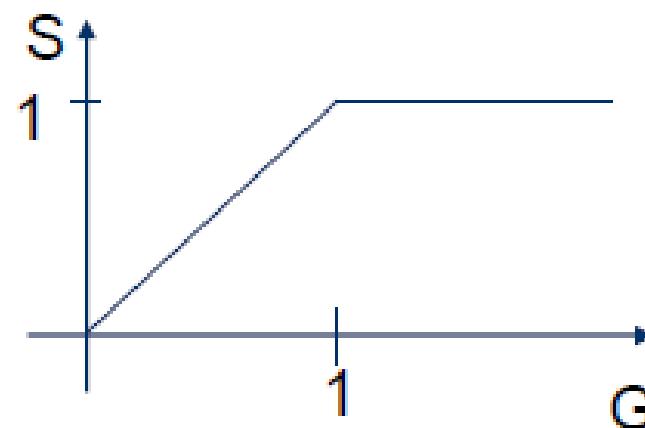
# Hogyan mérjük a hatékonyságot?

- **Átvitel [Throughput] (S)**
  - A sikeresen átvitt csomagok/keretek száma egy időegység alatt
- **Késleltetés [Delay]**
  - Egy csomag átviteléhez szükséges idő
- **Fairség [Fairness]**
  - minden állomás egyenrangúként van kezelve

# Átvitel és terhelés

## □ Terhelés ( $G$ )

- A protokoll által kezelendő csomagok száma egy időegység alatt (beérkezési hármasok)
- $G > 1$ : túlterhelés
- A csatorna egy kérést tüntetni elvezetni

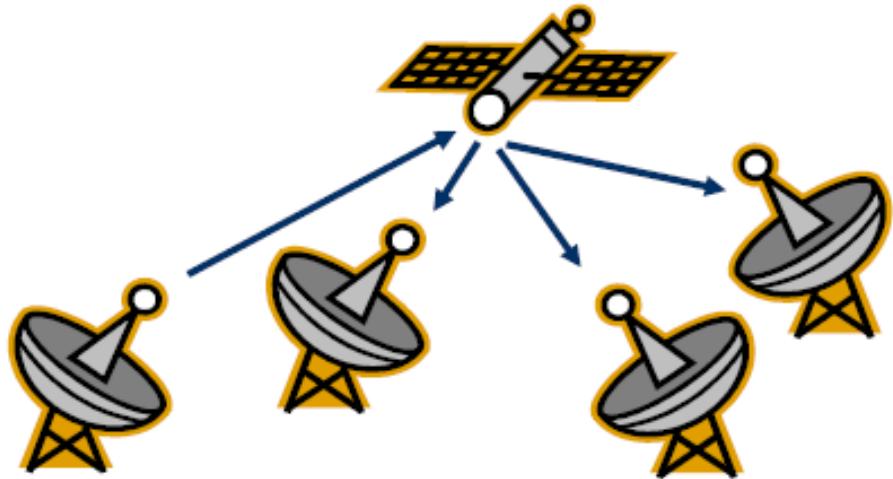
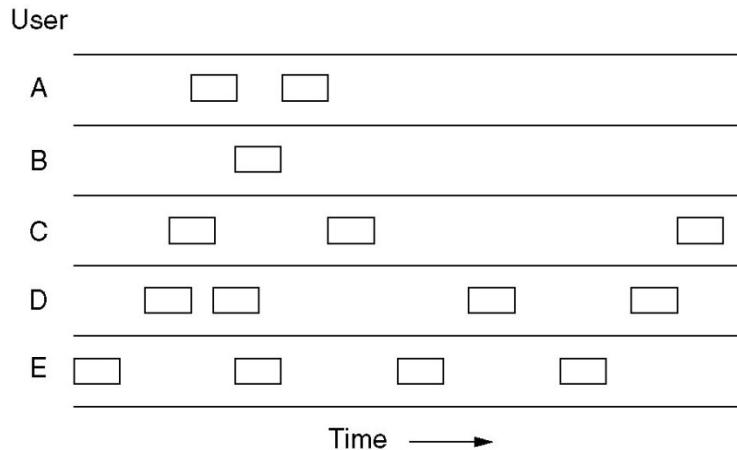
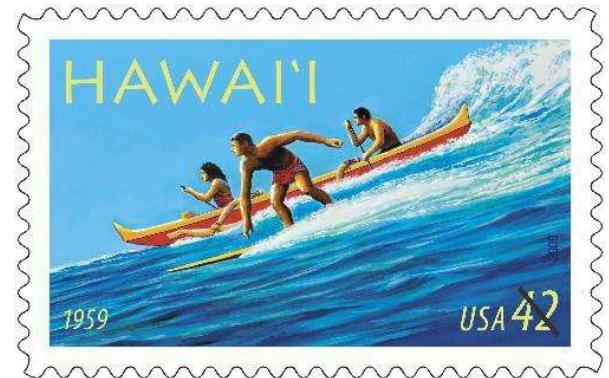


## □ Ideális esetben

- Ha  $G < 1$ ,  $S = G$
- Ha  $G \geq 1$ ,  $S = 1$

# (Tiszta) ALOHA

- Az algoritmust a 70-es években a Uni. of Hawaii fejlesztette
  - **Ha van elküldendő adat, akkor elküldi**
  - Alacsony költségű, nagyon egyszerű megoldás



# ALOHA

11

□ Topológia: broadcast rádió több állomással

□ Protokoll:

- Az állomások azonnal küldenek
- A fogadók minden csomagot nyugtáznak

- Egyszerű, de radikális megoldás
- Korábbi megoldások, mind felosztották a csatornát
  - TDMA, FDMA, etc.
- Kévés küldő esetére készült

# Teljesítmény elemzés -Poisson Folyam

- A „véletlen érkezések” egyik ünnepelt modellje a sorban-állás elméletben a Poisson folyam.
- A modell feltételezései:
  - Egy érkezés valószínűsége egy rövid  $\Delta t$  intervallum alatt arányos az intervallum hosszával és nem függ az intervallum kezdetétől (ezt nevezzük **memória nélküli** tulajdonságnak)
  - Annak a valószínűsége, hogy több érkezés történik egy rövid  $\Delta t$  intervallum alatt közelít a nullához.

# Teljesítmény elemzés –Poisson eloszlás

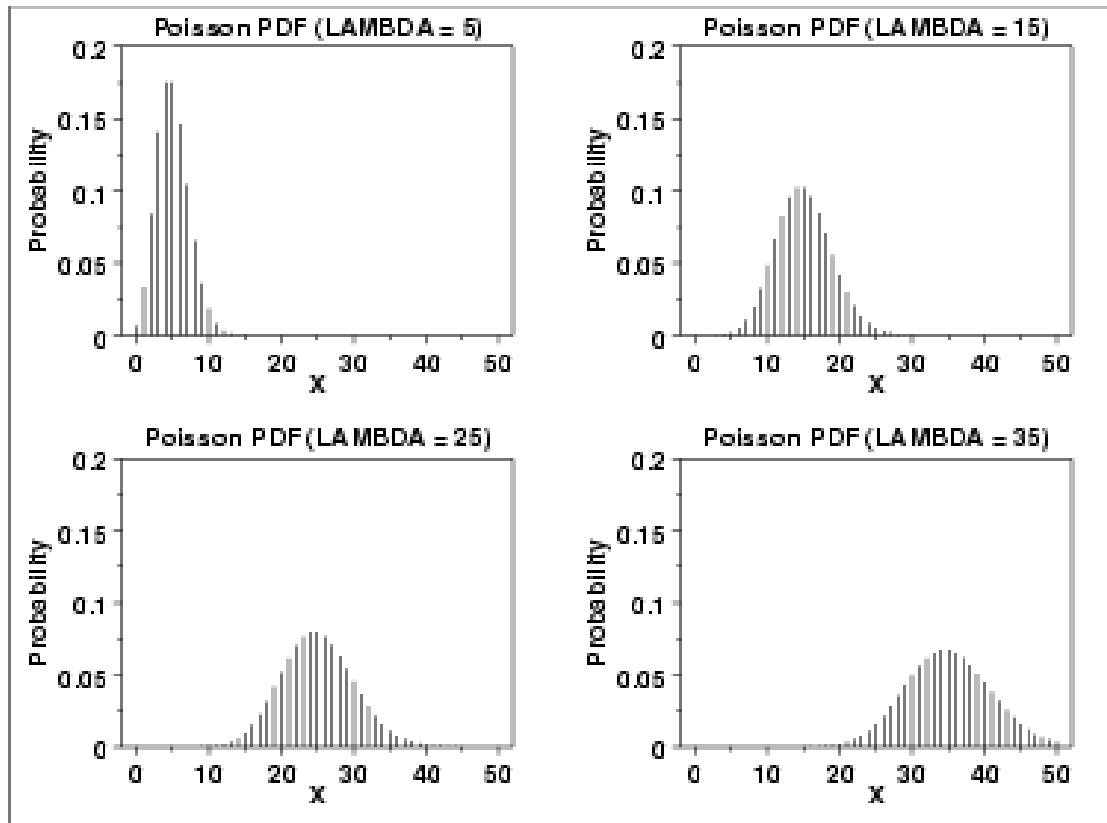
Annak a valószínűsége, hogy  $k$  érkezés történik egy  $t$  hosszú intervallum során:

$$P_k(t) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

ahol  $\lambda$  az érkezési ráta. Azaz ez egy egy-paraméteres modell, ahol csak  $\lambda$ -át kell ismernünk.

# Poisson Eloszlás példák

14



# ALOHA vizsgálata

- Jelölés:
  - $T_f$  = keret-idő (feldolgozási, átviteli és propagációs)
  - $S$ : A sikeres keret átvitelek átlagos száma  $T_f$  idő alatt; (*throughput*)
  - $G$ :  $T_f$  idő alatti összes átviteli kísérletek átlagos száma
  - $D$ : Egy keret küldésre kész állapota és a sikeres átvitele között eltelt átlagos idő
- Feltételezések
  - minden keret konstans/azonos méretű
  - A csatorna zajmentes, hibák csak ütközések miatt történnek

# ALOHA vizsgálata

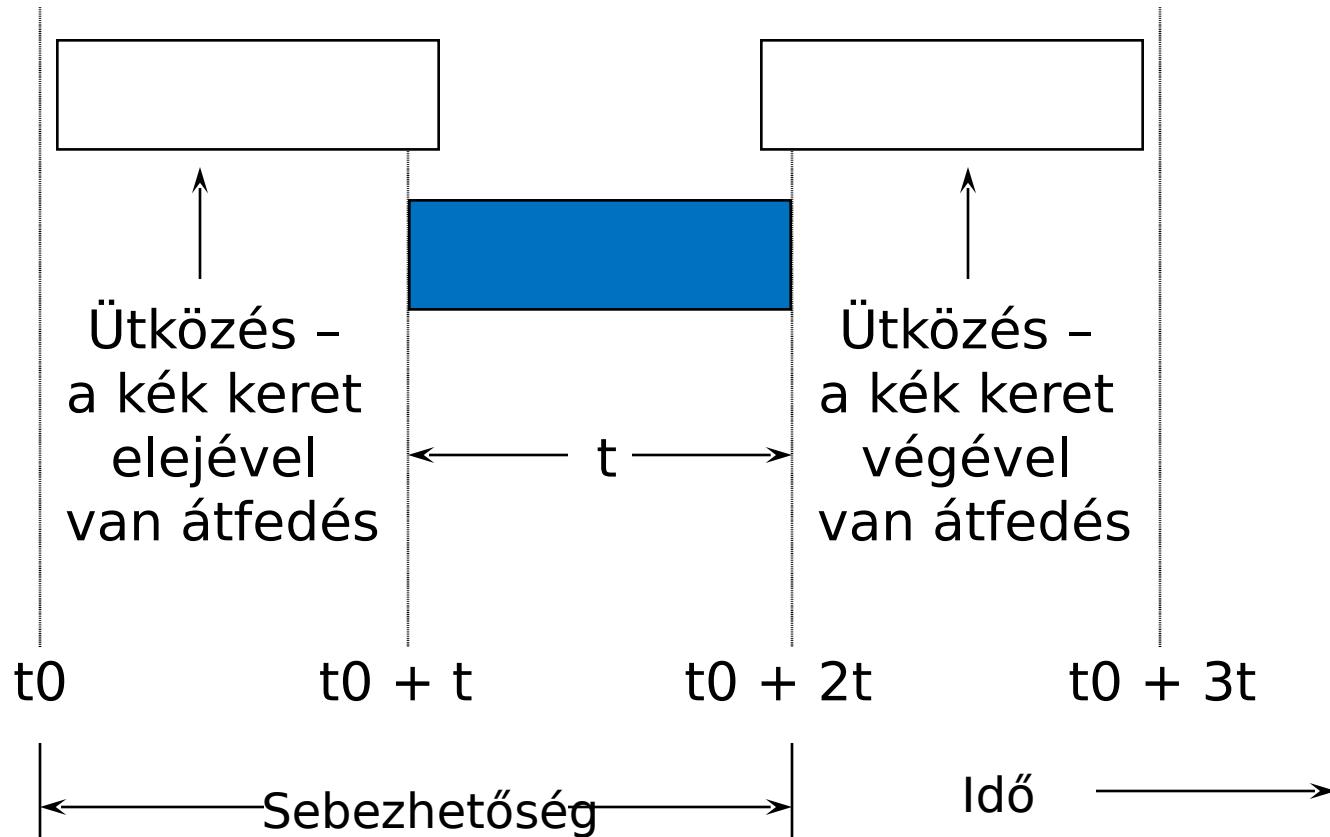
- Mivel  $S$  jelöli a „jó” átviteleket egy keret idő alatt és  $G$  jelöli az összes átviteli kísérletet egy keret idő alatt, így a következő összefüggést írhatjuk:

$$S = S(G) = G \times (\text{A „jó” átvitelek valószínűsége})$$

- A sebezetőségi idő egy keret sikeres átviteléhez:  $2T_f$
- Azaz a „jó” átvitel valószínűsége megegyezik

# ALOHA vizsgálata

17



Sebezhetőségi időintervallum a kékkel jelölt kerethez

# ALOHA vizsgálata

Tudjuk, hogy:

$$P_k(t) = \frac{(-t)^k e^{-t}}{k!}$$

Azaz most  $t = 2T_f$  és  $k = 0$  (t legyen a seb. Idő, k=0, hogy ne érkezzen új keret a kék küldése során)

$$P_0(2T_f) = \frac{(\lambda \cdot 2T_f)^0 e^{-\lambda 2T_f}}{0!} = e^{-2G}$$

because  $\lambda = \frac{G}{T_f}$ . Thus,  $S = G \cdot e^{-2G}$

# ALOHA vizsgálata

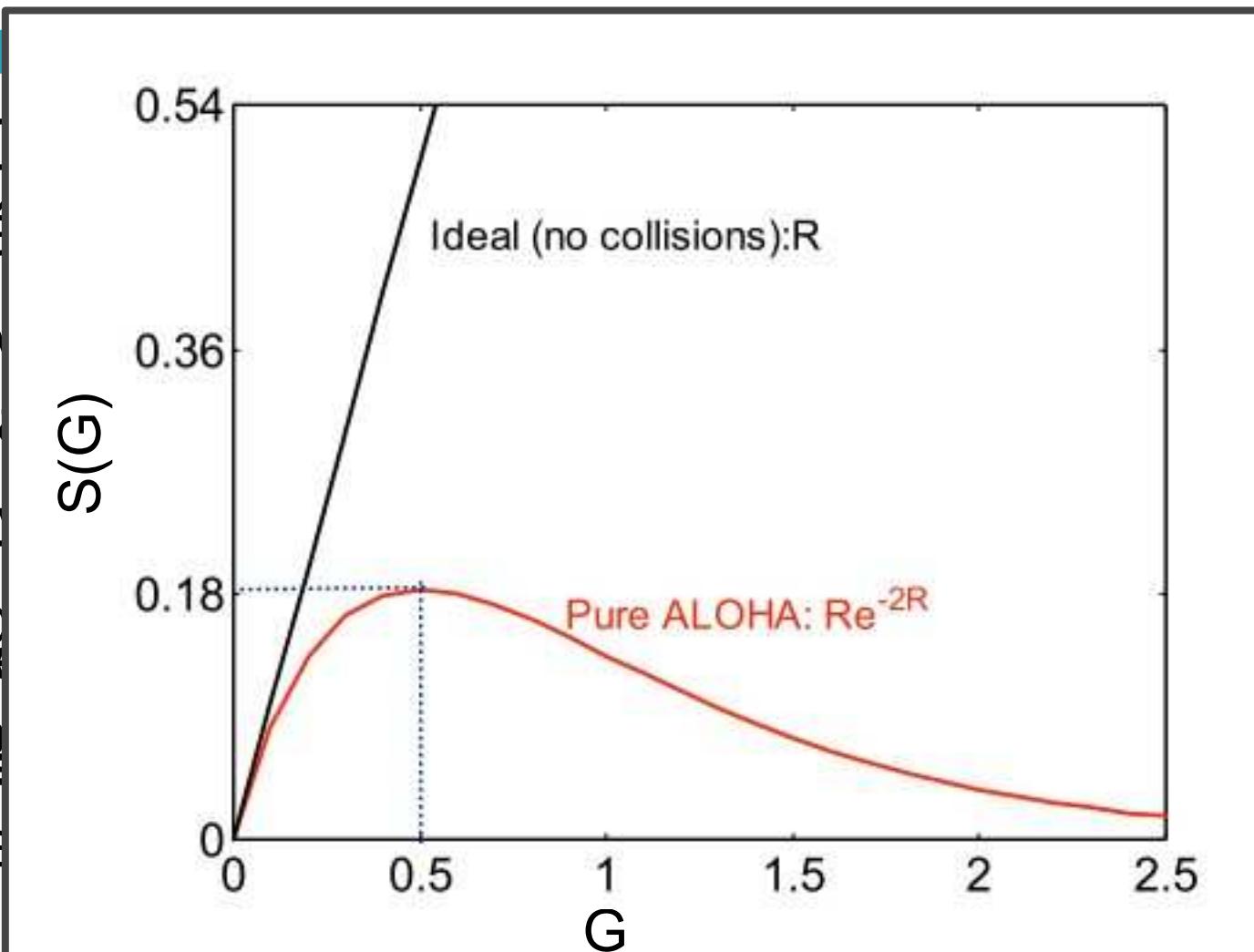
19

- $S(G) = Ge - 2G$  függvényt G szerint deriválva és az eredményt nullának tekintve az egyenlet megoldásával megkapjuk a maximális sikeres átvitelhez tartozó G értéket:

$$G = 0.5,$$

melyre  $S(G) = 1/2e = 0.18$ . Azaz a maximális throughput csak 18%-a a teljes kapacitásnak!!!

# ALOHA vs TDMA



20  
□ A TDMA-körében érhető el a csatorna kapacitás 10%-a

Az ALOHA-szabálytól függetlenül minden sendernek van egy saját szabályai

Az ALOHA különösen a külső sendereknél használható

Sender

De

Máximálisan a csatorna kapacitás 10%-a

aját

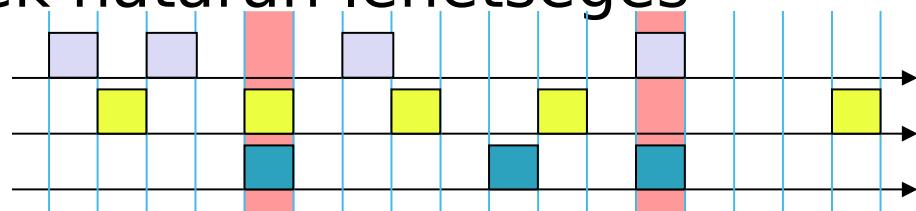
al

→

# Réselt ALOHA

21

- A csatornát azonos időrésekre bontjuk, melyek hossza pont egy keret átviteléhez szükséges idő.
- Átvitel csak az időrések határán lehetséges



- Algoritmus:
  - Amikor egy új A keret küldésre kész:
    - Az A keret kiküldésre kerül a (következő) időrés-határon

# A réselt ALOHA vizsgálata

□ A sebezhetőségi idő a felére csökken!!!

□ Tudjuk, hogy  $P_k(t) = \frac{(-t)^k e^{-t}}{k!}$

Ez esetben  $t = T_f$  és továbbra is  $k = 0$ , amiből kapjuk, hogy:

$$P_0(T_f) = \frac{(\lambda \cdot T_f)^0 e^{-\lambda T_f}}{0!} = e^{-G}$$

because  $\lambda = \frac{G}{T_f}$ . Thus,  $S = G \cdot e^{-G}$

# Résult ALOHA

23

Prot

Ug

■

Cs

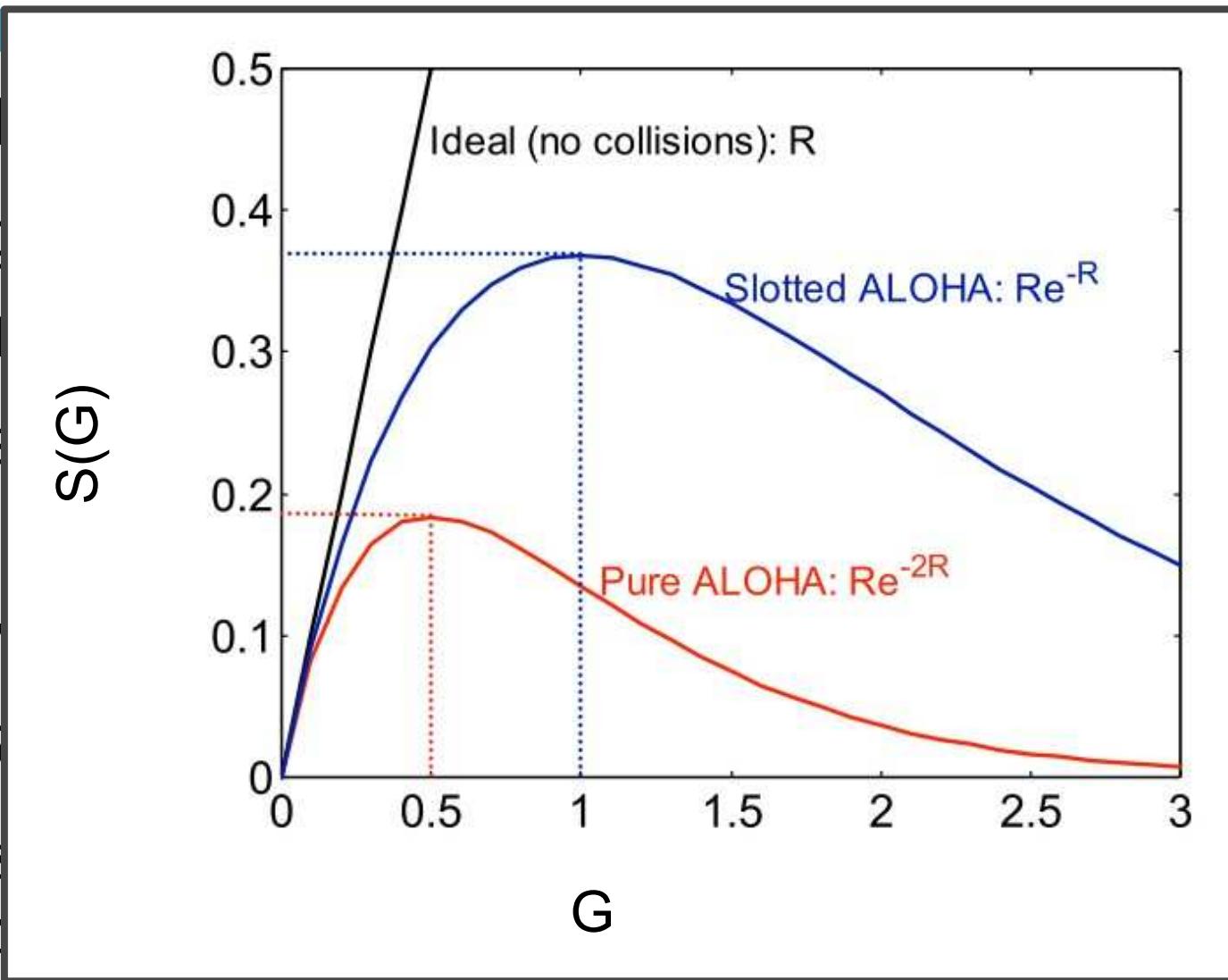
Aza  
egy

37

Az

SZ

vagy  
etén)



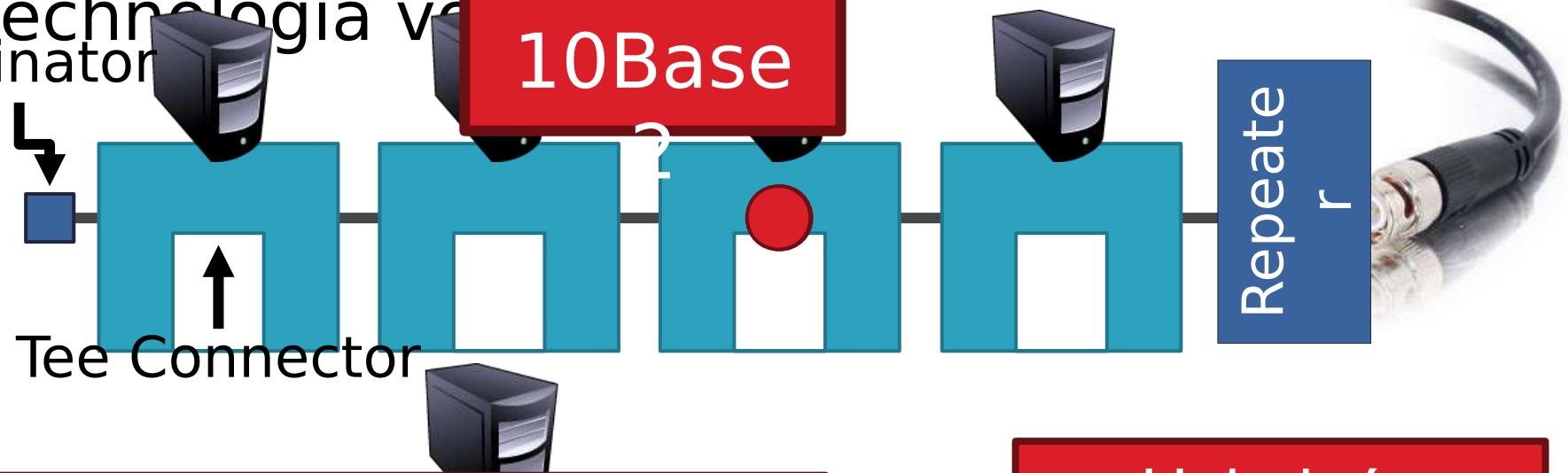
# Adatszóró (Broadcast)

## Ethernet

24

- Eredetileg az Ethernet egy adatszóró technológiával működött.

Terminator



- 10BaseT és 100BaseT

java

b

Hubok és repeaterek minden 1. rétegbeli eszközök (csak fizikai)

# Vivőjel érzékelés

## Carrier Sense Multiple Access (CSMA)

- További feltételezés

- minden állomás képes belehallgatni a csatornába és így el tudja dönteni, hogy azt más állomás használja-e átvitelre

# 1-perzisztens CSMA protokoll

26

- Vivőjel érzékelés van, azaz minden állomás belehallgathat a csatornába.
- Folytonos időmodellt használ a protokoll

## Algoritmus

- Keret leadása előtt belehallgat a csatornába:
  - a) Ha foglalt, akkor addig vár, amíg fel nem szabadul. Szabad csatorna esetén azonnal küld. (*perzisztens*)
  - b) Ha szabad, akkor küld.
- Ha ütközés történik, akkor az állomás véletlen hosszú ideig vár, majd újrakezdi a keret leadását.

## Tulajdonságok

- A terjedési késleltetés nagymértékben befolyásolhatja a teljesítményét

# Nem-perzisztens CSMA protokoll

27

- Vivőjel érzékelés van, azaz minden állomás belehallgathat a csatornába.
- Folytonos időmodellt használ a protokoll
- Mohóság kerülése

## Algoritmus

- Keret leadása előtt belehallgat a csatornába:
  - a) Ha foglalt, akkor véletlen ideig vár (nem figyeli a forgalmat), majd kezdi előről a küldési algoritmust. (*nem-perzisztens*)
  - b) Ha szabad, akkor küld.
- Ha ütközés történik, akkor az állomás véletlen hosszú ideig vár, majd újrakezdi a keret leadását.

## Tulajdonságok

# p-perzisztens CSMA protokoll

28

- Vivőjel érzékelés van, azaz minden állomás belehallgathat a csatornába.
- Diszkrét időmodellt használ a protokoll

## Algoritmus

- Adás kész állapotban az állomás belehallgat a csatornába:
  - a) Ha foglalt, akkor vár a következő időrésig, majd megismétli az algoritmust.
  - b) Ha szabad, akkor  $p$  valószínűsséggel küld, illetve  $1-p$  valószínűsséggel visszalép a szándékától a következő időrésig. Várakozás esetén a következő időrésben megismétli az algoritmust. Ez addig folytatódik, amíg el nem küldi a keretet, vagy amíg egy másik állomás el nem kezd küldeni, mert ilyenkor úgy viselkedik, mintha ütközés történt volna.
- Ha ütközés történik, akkor az állomás véletlen hosszú ideig vár, majd újra elkezdi a csatornától való távolságba lépni.

# CSMA áttekintés

29

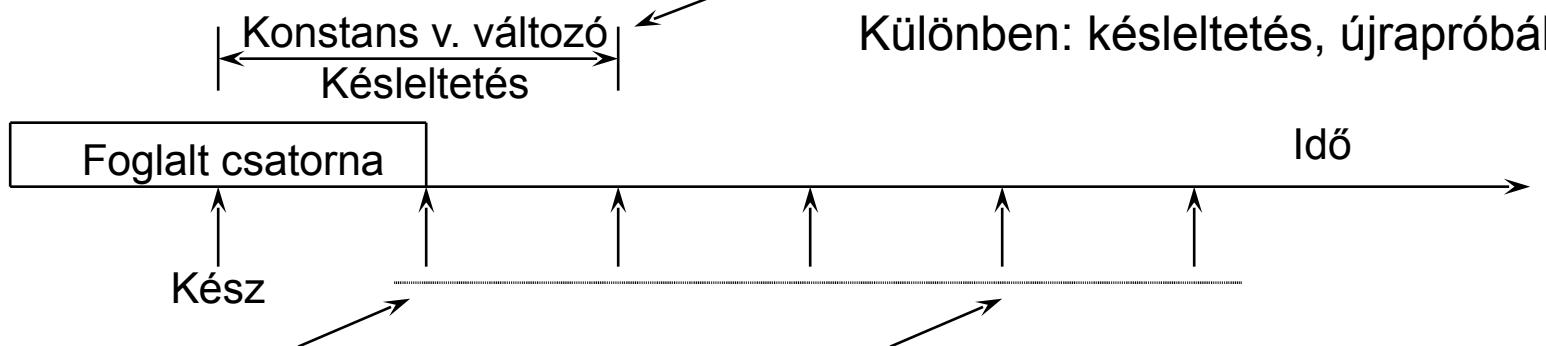
- Nem-perzisztens
- 1-perzisztens
- $p$ -perzisztens

## CSMA perzisztencia

### **Nem-perzisztens:**

Átvitel ha szabad

Különben: késleltetés, újrapróbáljuk



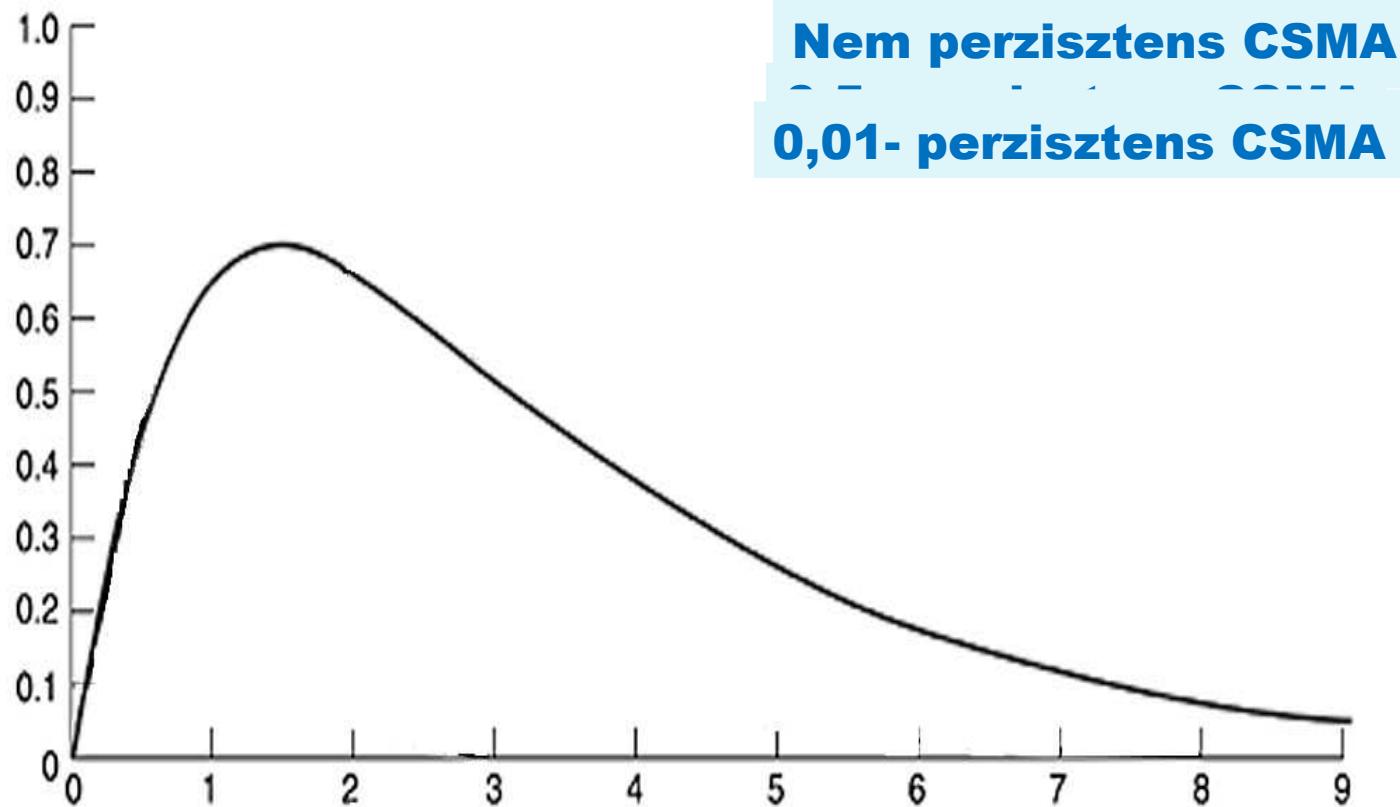
### **1-perzisztens:**

Átvitel amint a csatorna szabad  
Ütközés esetén visszalépés,  
majd újrapróbáljuk

### **$p$ -perzisztens:**

Átvitel  $p$  valószínűsséggel, ha a csatorna szabad  
Különben: várunk 1 időegységet és újrapróbáljuk

# CSMA és ALOHA protokollok összehasonlítása



Forrás: [1]

# CSMA/CD - CSMA ütközés detektálással

- 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

# 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

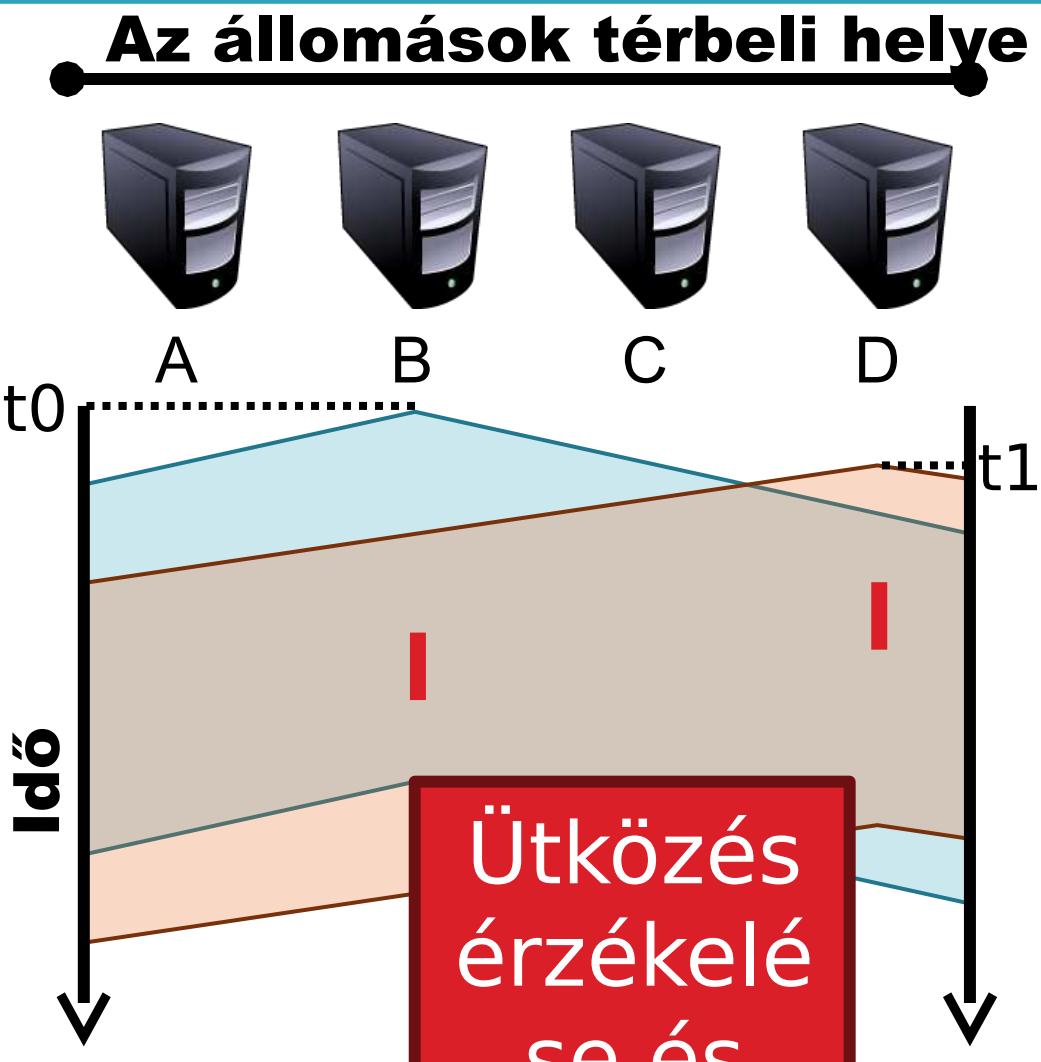
33

- 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

# CSMA/CD Ütközések

34

- 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

35

- Ütközés érzékelésekor a küldő egy ún. „jam” jelet küld
  - minden állomás tudomást szerezzen az ütközésről
- Binary exponential backoff működése:
  - Válasszunk egy  $k \in [0, 2n - 1]$  egyenletes eloszlás szerint, ahol  $n =$  az ütközések száma
  - Várunk  $k$  időegységet (keretidőt) az újraküldésig
  - $n$  felső határa 10, 16 sikertelen próbálkozás után

# Binary Exponential Backoff

36

Tekintsünk két állomást, melyek üzenetei ütköztek

- Első ütközés után: válasszunk egyet a két időrés közül
  - A sikeres átvitel 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

# Minimális keretméret

37

- Miért 64 bájt a minimális keretméret?
  - I Az állomásoknak elég időre van szüksége az ütközés detektálásához
- Mi a kapcsolat a keretméret és a kábelhossz

1. központi? Az A

állomás megkezdi az átvitelt

2.  $t + d$  időpont: A B

állomás.js  
Alapötlet: Az A állomásnak  $2*d$  ideig kell megkezdeni az átvitelt



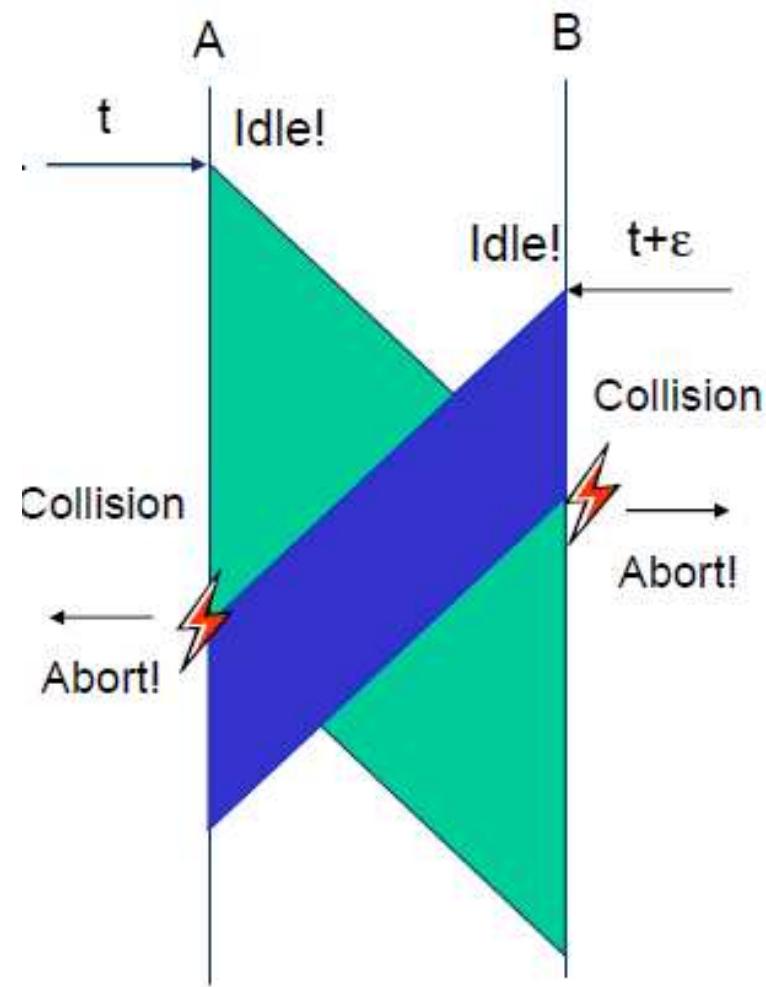
Alapötlet: Az A állomásnak  $2*d$  ideig kell küldenie!

# 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



# Minimális keretméret

39

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

- $\text{Min\_keret} = \text{ráta (b/s)} * 2 * d (\text{s})$

- ...
      - 10 Mbps Ethernet
      - A keretméret és a kábelhossz változik a gyorsabb fénysebességek miatt
    - Problémák a hosszú kábelben

- Azaz:

- $\text{Min\_keret} = (\text{ráta (b/s)} * 2 * \text{távolság (m)}) / \text{fényseb. (m/s)}$

- Azaz a kábel hossza ...  
$$\frac{(64B*8)*(2*108mps)}{(2*107bps)} = 5120 \text{ meter}$$

# Minimális keretméret

40

- Az A küldésének  $2*d$  ideig kell tartania
  - Min\_keret = ráta (b/s) \* 2 \* d (s)
    - ... de mi az a  $d$ ? propagációs késés, melyet a fénysebesség ismeretében ki tudunk számolni
    - Propagációs késés ( $d$ ) = távolság (m) / fénysebesség (m/s)
  - Azaz:
  - Min\_keret = ráta (b/s) \* 2 \* távolság (m) / fényseb. (m/s)
- Azaz a kábel  $\frac{(64B*8)*(2*108mps)}{(2*107bps)} = 5120$  meter

# Kábelhossz példa

41

$$\text{min\_keret} * \text{fénysebesség} / (2 * \text{ráta}) = \text{max\_kábelhossz}$$

$$(64B * 8) * (2 * 108 \text{ mps}) / (2 * 10 \text{ Mbps}) = 5120 \text{ méter}$$

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

# Maximális keretméret

42

- 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 vesztegettünk el a fejlécekben
  - Összességében nagyobb csomag feldolgozási idő
- Adatközpontokban Jumbo keretek
  - 9000 bájtos keretek

Köszönöm a figyelmet!

# Számítógépes Hálózatok

**6. Előadás: Adatkapcsolati  
réteg IV.  
& Hálózati réteg**

Based on slides from

# Ütközésmentes protokollok

2

## Motiváció

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

## Feltételezések

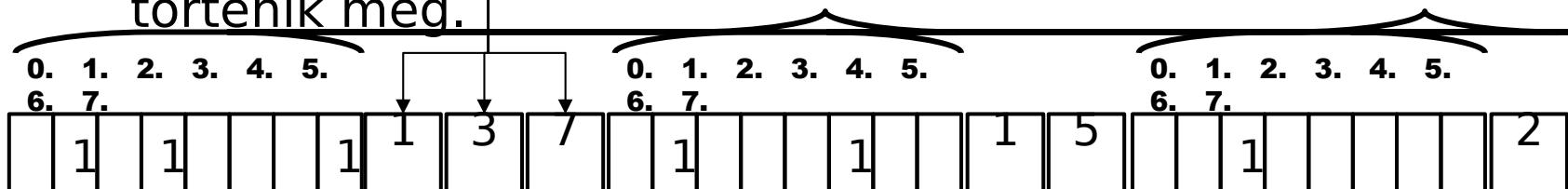
- N állomás van.
- Az állomások 0-ától N-ig egyértelműen sorszámozva vannak.
- Réselt időmodellt feltételezünk.

# Alapvető bittérkép protokoll

## 3 - Egy helyfoglalásos megoldás

### 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 sorszámpok szerinti sorrendben történik meg.



# Bináris visszaszámítás protokoll 1/2

4

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

## Működés

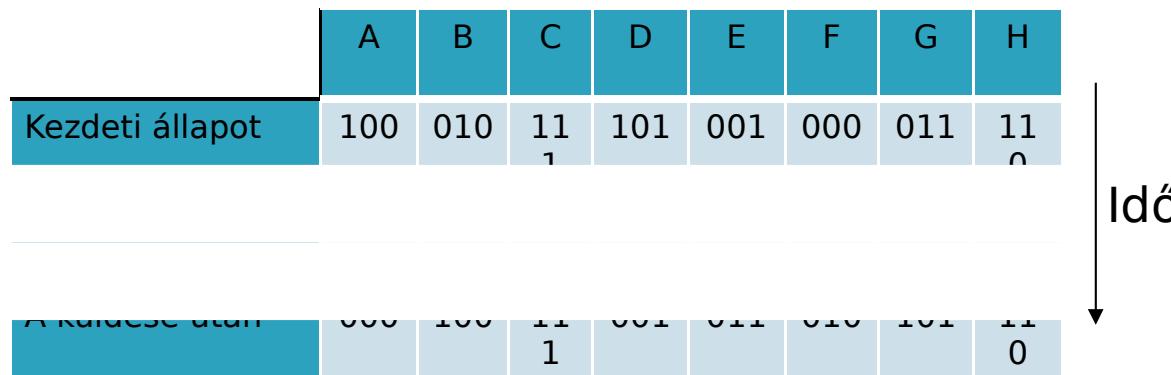
- minden állomás azonos hosszú bináris azonosítóval rendelkezik.
- A forgalmazni kívánó állomás elkezdi a bináris címét bitenként elküldeni a legnagyobb helyi értékű bittel kezdve. Az ~~A<sub>hoszt(0011)</sub>~~<sup>A<sub>hoszt(0011)</sub></sup> pozíciójú bitek logikai VAGY kapcsolatba lépnek) ütközés esetén. Ha az állomás nullát küld, de egyet hall vissza, akkor feladja a küldési szándékát, mert van nála nagyobb azonosítóval rendelkező küldő.

# Bináris visszaszámítás protokoll 2/2

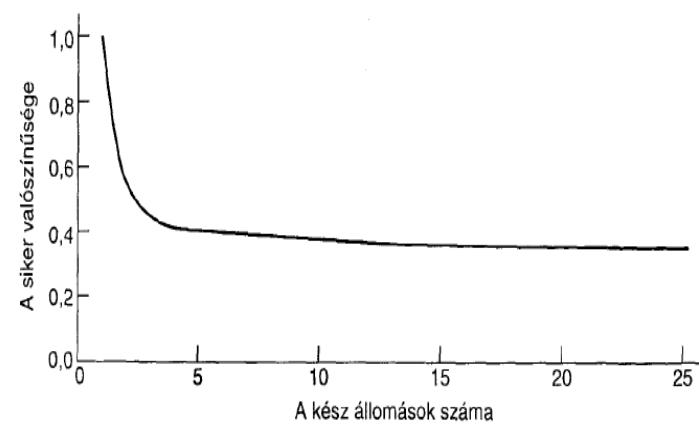
- **Következmény:** a magasabb címmel rendelkező állomásoknak a prioritásuk is magasabb az alacsonyabb című állomásokénál

## Mok és Ward módosítása

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



# Korlátosztott versenyes protokollok

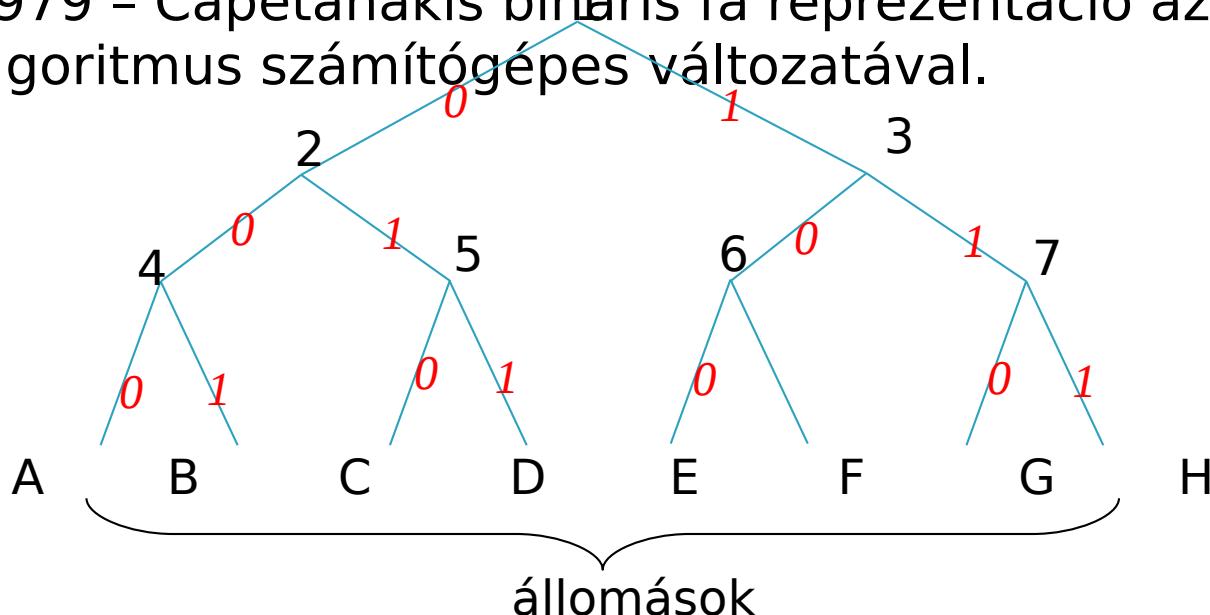


# Adaptív fabejárási protokoll

7 1/2

## Történeti háttér

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



# Adaptív fabejárási protokoll

8 2/2

## Működés

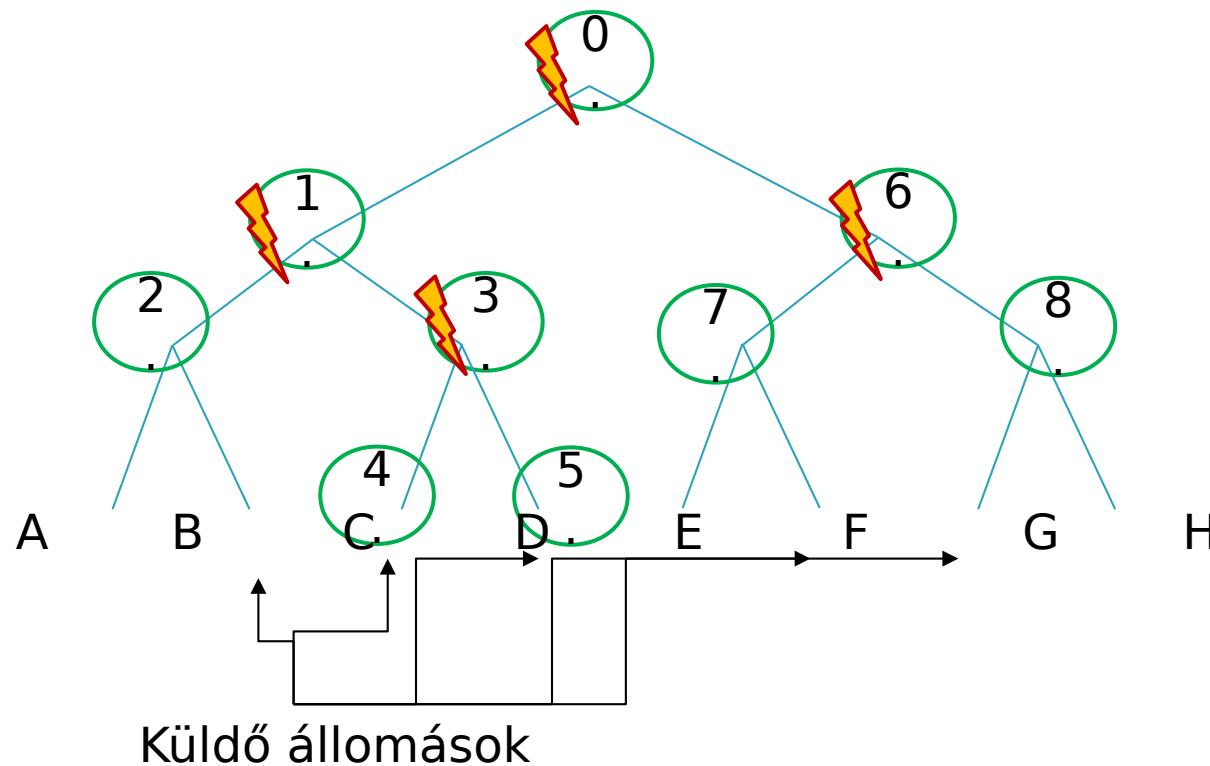
- 0-adik időrésben mindenki küldhet.
  - Ha ütközés történik, akkor megkezdődik a fa *mélységi bejárása*.
- A rések a fa egyes csomópontjaihoz vannak rendelve.
- Ütközéskor rekurzívan az adott csomópont bal illetve jobb gyerekcsomópontjánál folytatódik a keresés.
- Ha egy bitrész kihasználatlan marad, 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” ...



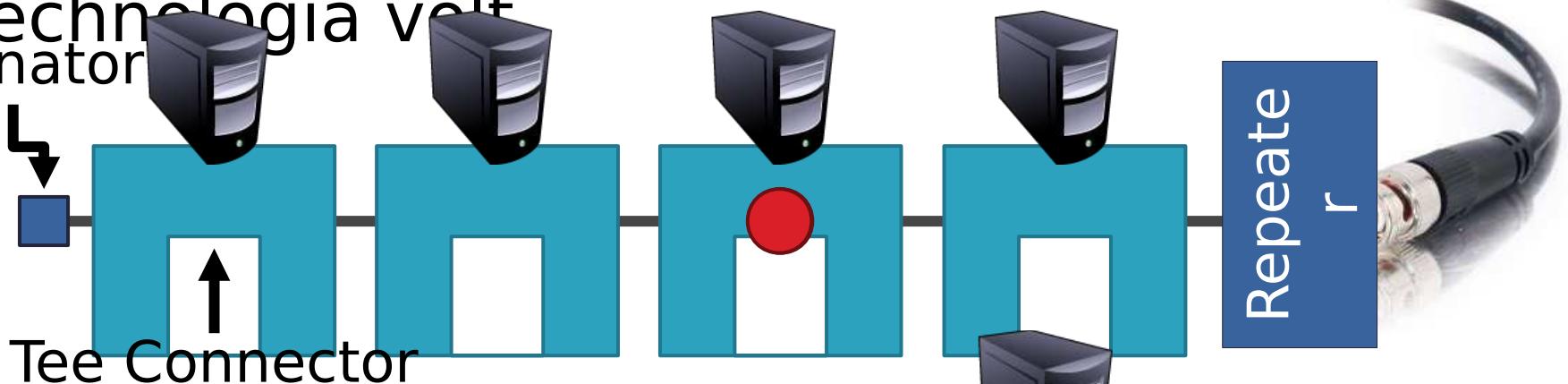
- Bridging, avagy hidak
  - Hogyan kapcsoljunk össze LAN-okat?
- Funkciók:
  - Keretek forgalomirányítása a LAN-ok között
- Kihívások:
  - Plug-and-play, önmagát konfiguráló
  - Esetleges hurkok feloldása

# Visszatekintés

11

- Az Ethernet eredetileg adatszóró technológia volt

Terminator

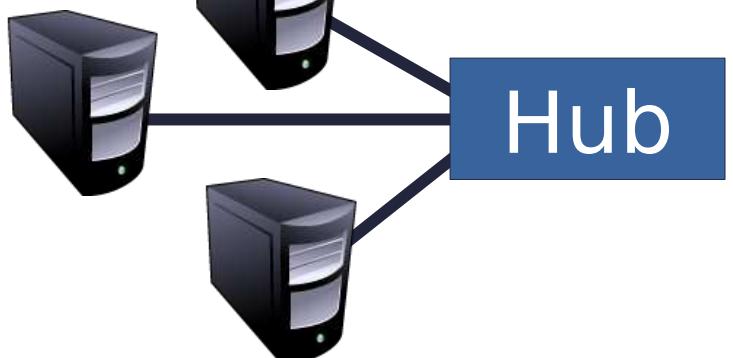


- Pro: Egyszerű

- Olcsó és buta hardver

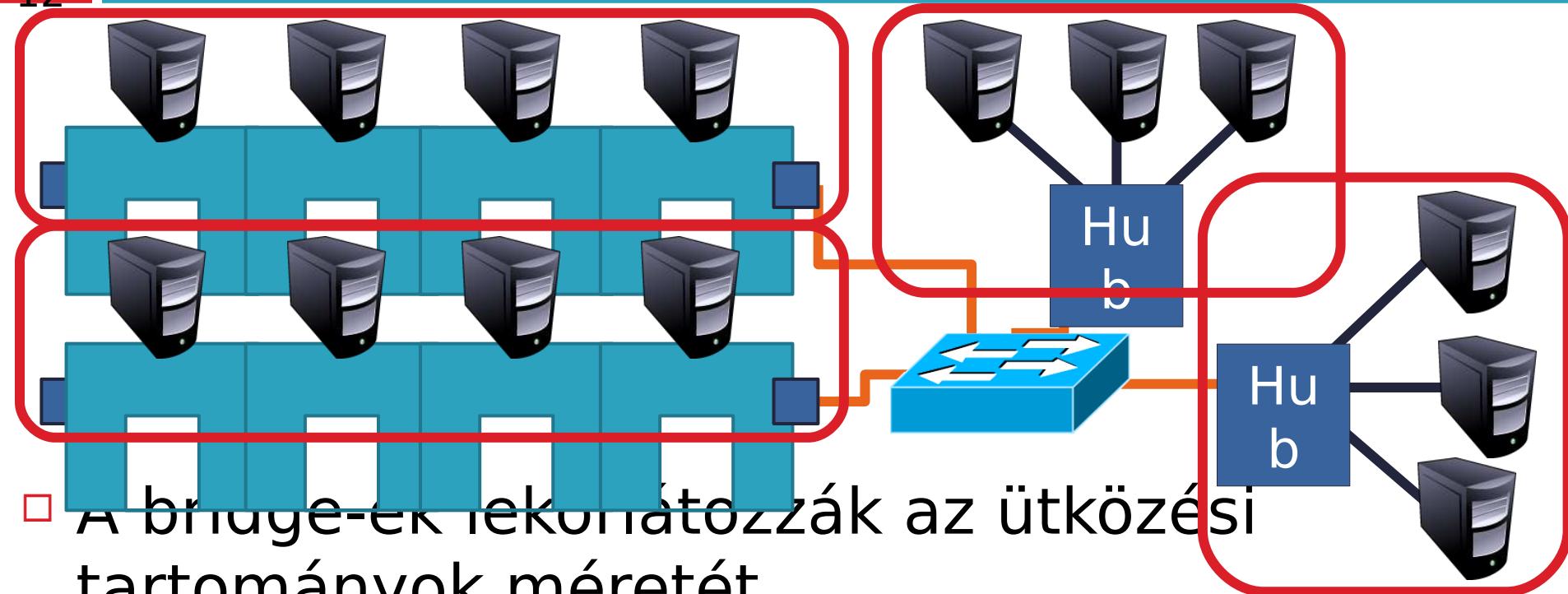
- Kontra: Nem skálázható

- 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?

# 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



# 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)**
- Teljes átlátszóság
  - “Plug-and-play,” önmagát konfiguráló

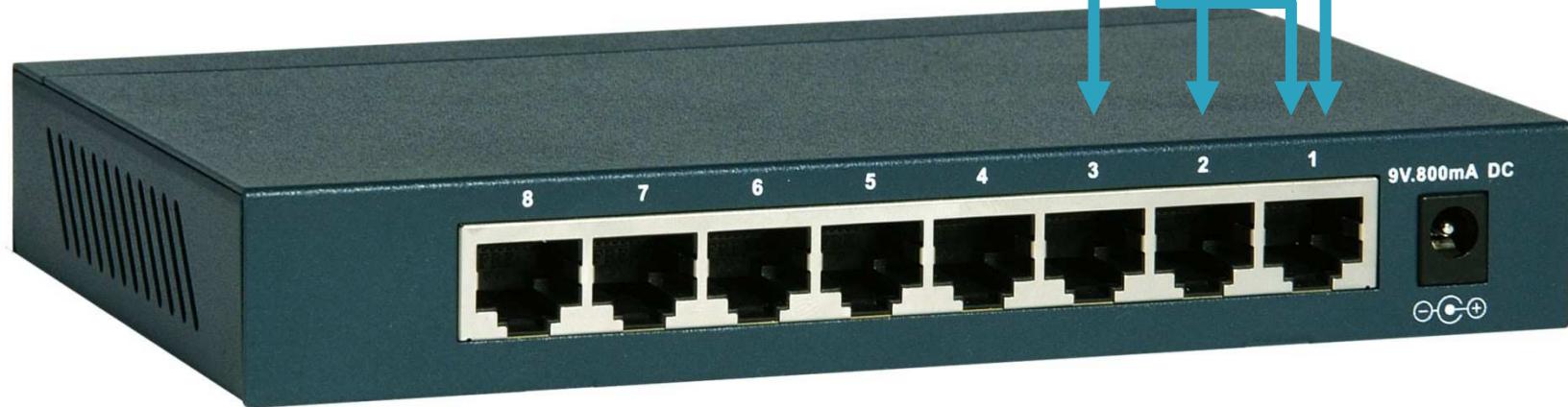
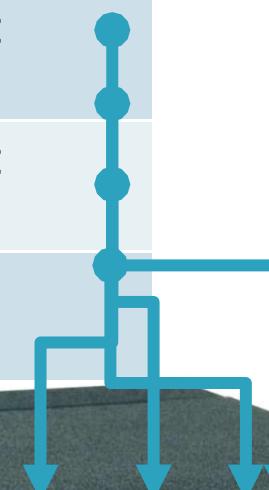


# Keret Továbbító Táblák

15

- Minden bridge karbantart egy **továbbító táblát** (*forwarding table*)

MAC Cím	Port	Kor
00:00:00:00:00:A A	1	1 perc
00:00:00:00:00:B B	2	7 perc
00:00:00:00:00:C C	3	2 mp



# Címek tanulása

16

- Kézi beállítás is lehetséges, de...

- Időigényes
- Potenciális hiba forrás
- Nem alkalmazkodik a változásokhoz  
léphető időszakban, mivel régi címek hagyhatják el a rendszert

Töröljük a régi  
bejegyzéseket

MAC cím	Poszíció	Idő
00:00:00:00:00:A	1	10 minutes
A		
B		



# Címek tanulása

17

- Kézi beállítás is lehetséges, de...
  - Időigényes
  - Potenciális hiba forrás
  - Nem alkalmazkodik a változásokhoz (új hosztok léphetnek fel, régiok elszűnések hagyhatják el a hálózatot)

MAC cím	Port	Kor
00:00:00:00:00:A	1	0 minutes
A		
B		

Hu  
b



00:00:00:00:00:BB

# Hurkok problémája

18

- <Src=AA, Dest=DD>

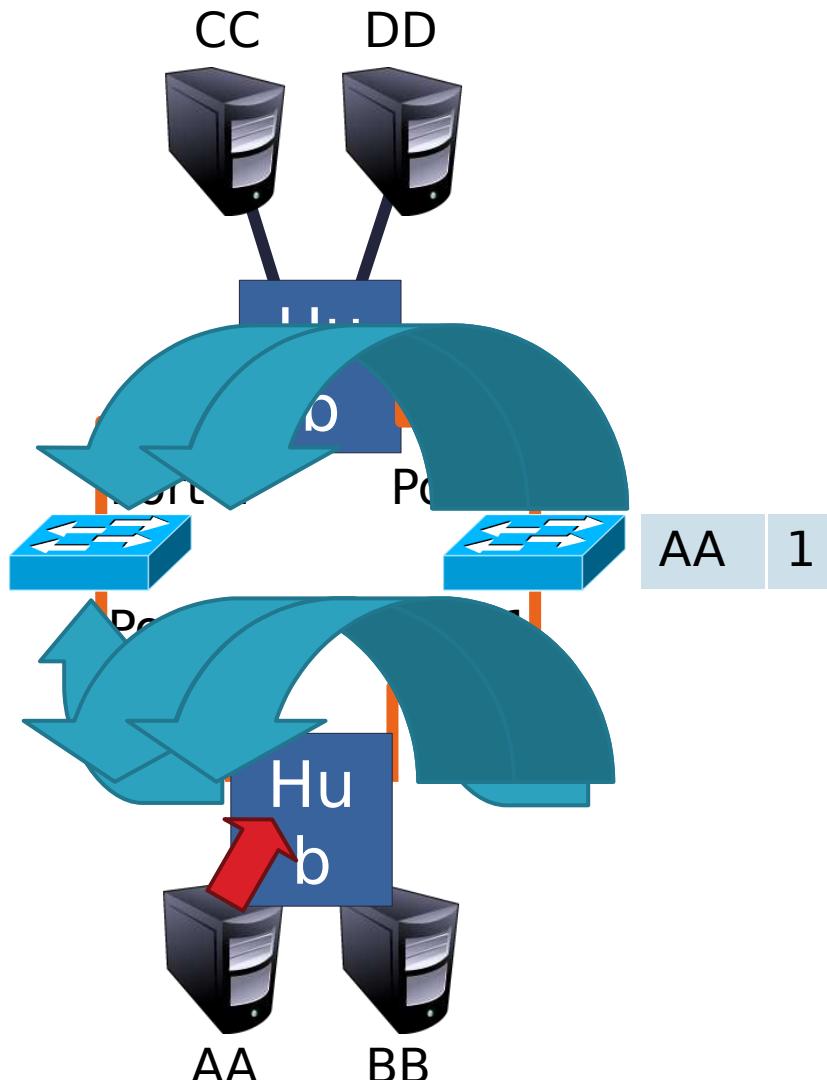
- Ez megy a végtelenséggig

- Hogyan állítható meg?

- Távolítsuk el a hurkot a topológiából

- A kábelek kihúzása nélkül

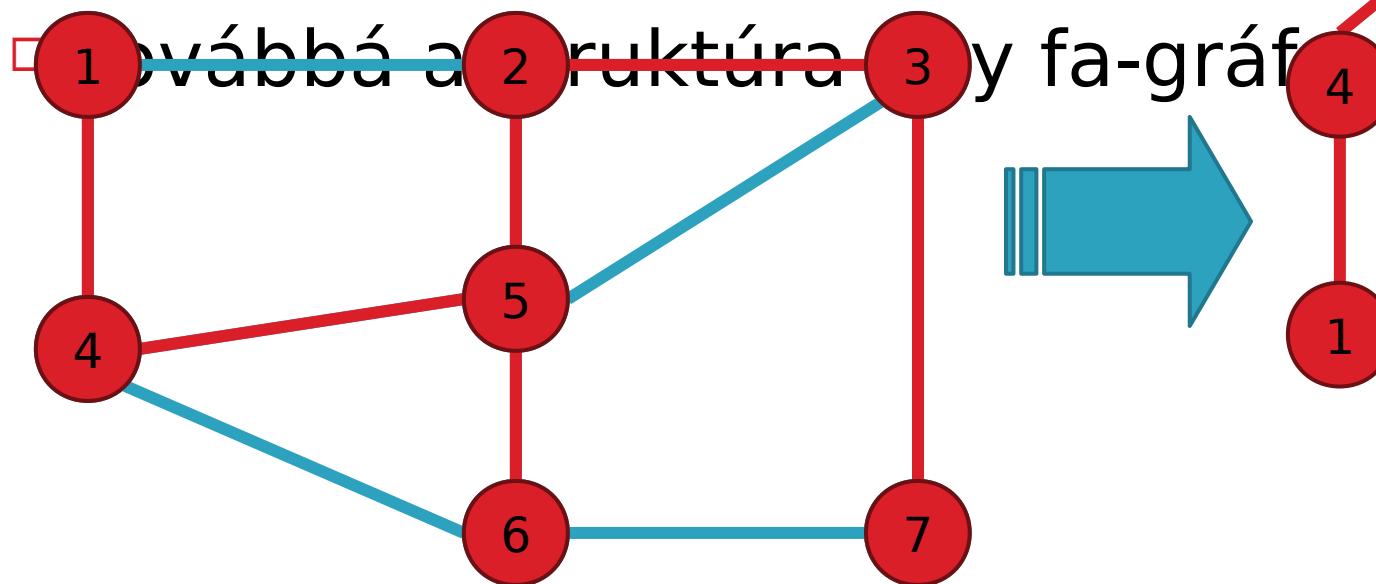
- 802.1 (LAN) definiál egy algoritmust



# Feszítőfa

19

- Egy gráf éleinek részhalmaza, melyre teljesül:
  - Lefed minden csomópontot
  - Nem tartalmaz köröket
- Elvábbá a struktúra a folyamatos gráfban



# A 802.1 feszítőfa algoritmusa

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)

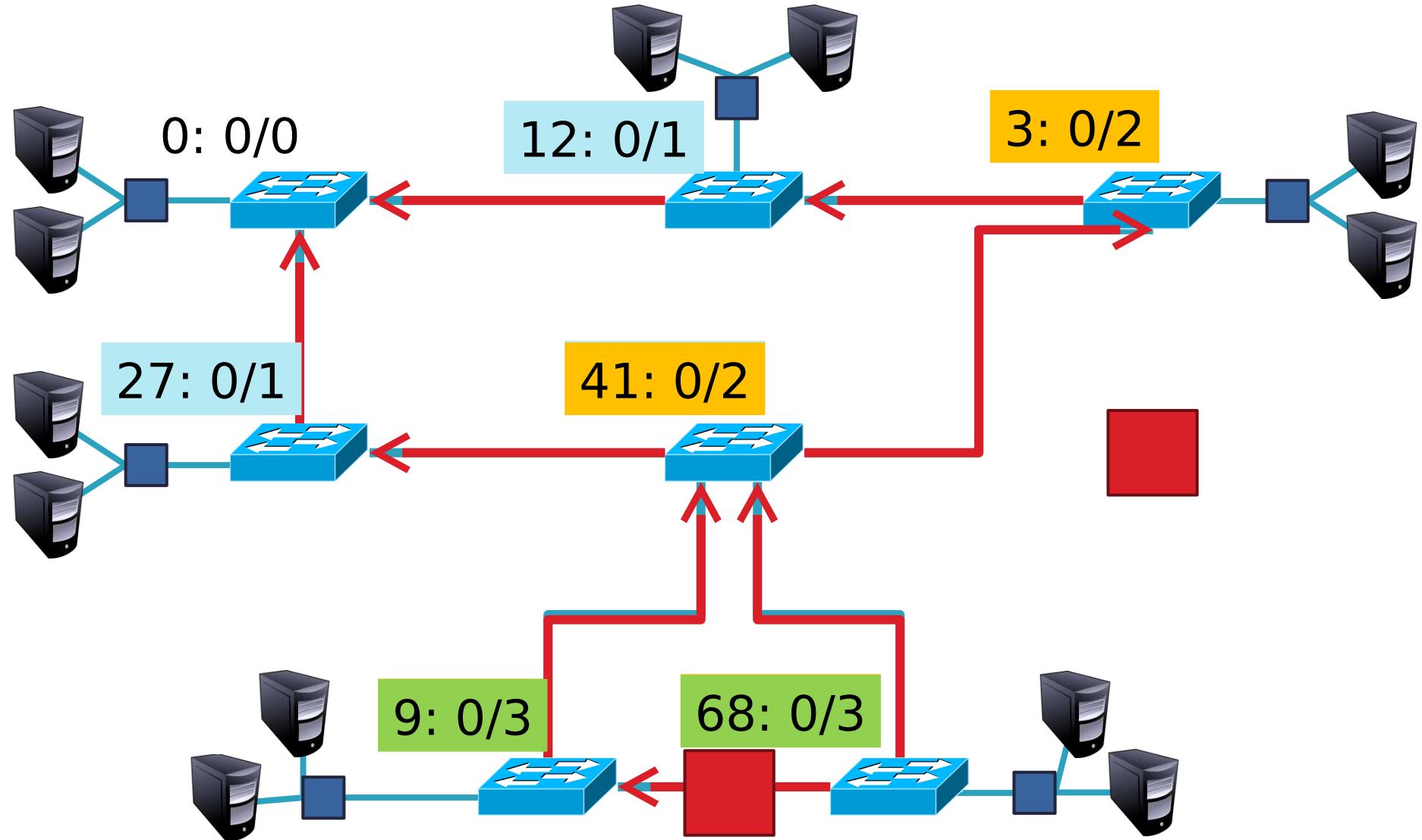
# Gyökér meghatározása

21

- Kezdetben minden állomás feltételezi magáról, hogy gyökér
- Bridge-ek minden irányba szétküldik a BPDU üzeneteket.
  - 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)

# 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

# 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!

- Szolgáltatás

# Hálózati réteg

▫ Csomagtovábbítás

▫ Útvonalválasztás

▫ Csomag fragmentálás kezelése

▫ Csomag ütemezés

▫ Puffer kezelés

## ▫ Interfész

▫ Csomag küldése egy adott végpontnak

## ▫ Protokoll

▫ Globálisan egyedi címeket definiálása

Alkalmaz

Megjelen

Ülés

Szállítói

Hálózati

Adatkapc

Fizikai

# 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. Forgalmirányító táblázatok feltöltése és karbantartása.
  2. Továbbítás.

## Elvárások

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

## Algoritmus osztályok

- 3 Adatív algoritmusok

# 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 látáldási idő

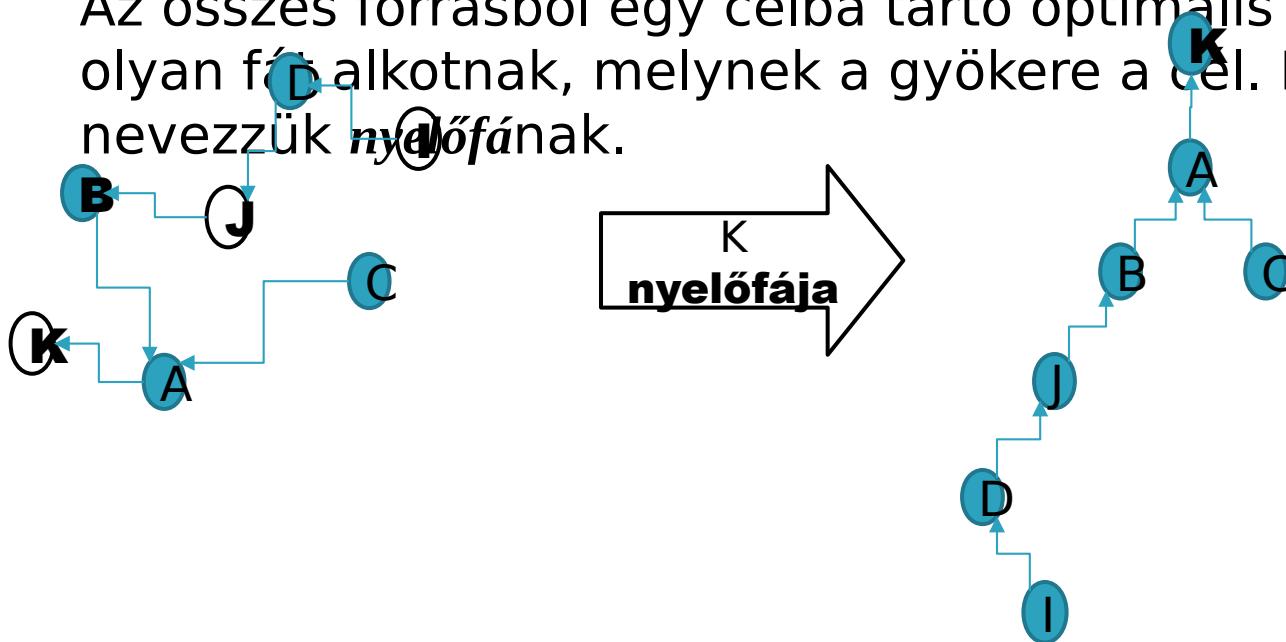
# Optimalitási elv

29

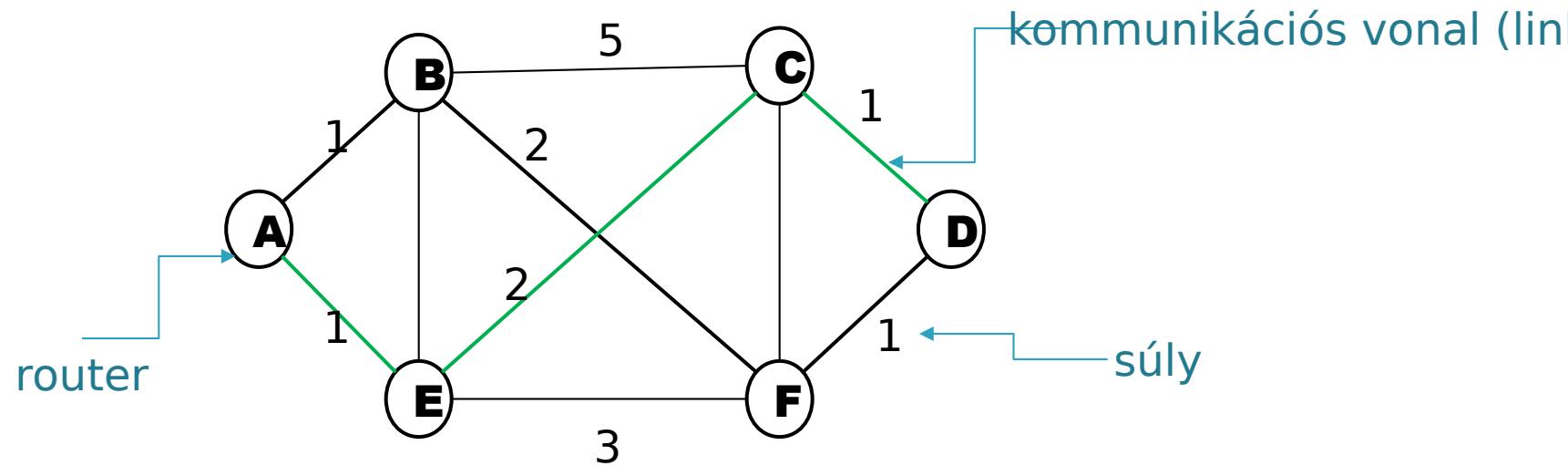
Ha  $J$  router az  $I$  router-től  $K$  router felé vezető optimális útvonalon helyezkedik el, akkor a  $J$ -től a  $K$ -ig vezető útvonal ugyanerre esik.

## ▫ 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



# 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 táblázatokat a szomszédoktól származó információk alapján frissítik.
  - Elosztott Bellman-Ford forgalomirányítási algoritmusként is nevezik.

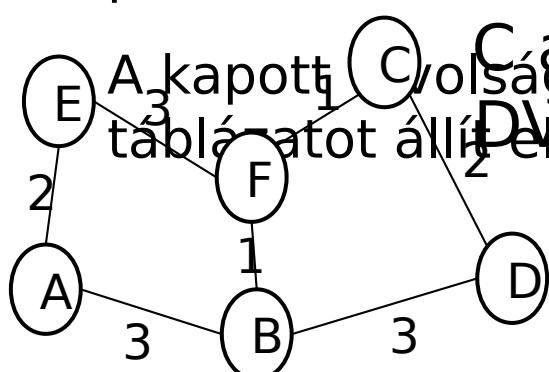
# Távolságvektor alapú forgalomirányítás

32

## Flosztott Bellman-Ford Környezet és működés algoritmus

- minden csomópont csak a közvetlen szomszédjaival kommunikálhat.

- Aszinkron működés.
- minden állomásnak van saját távolság vektora. Ezt periodikusan elküldi a direkt szomszédoknak.



C állomás  
távolság vektora  
DV-táblája

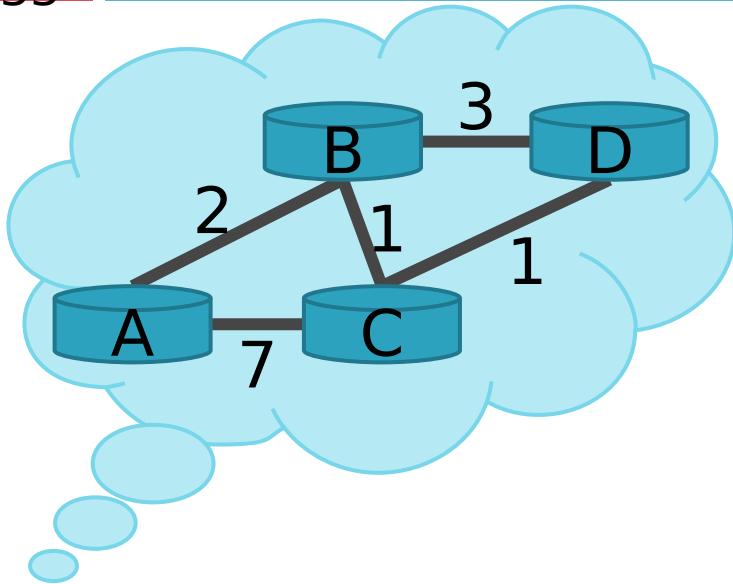
Cél	Ktsg.
A	5
B	2
D	2
E	4
F	1

▫ Nincs bejegyzés C-  
nél, mivel nincs szomszédság C-nél

- Kezdetben csak a közvetlen szomszédokhoz van info
- Más célállomások költsége =  $\infty$

# Distance Vector Initialization

33



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

```

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

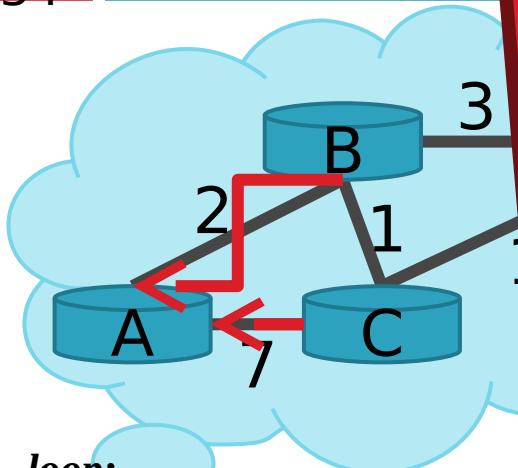
Node B		
Dest.	Cost	Next
A	2	A
C	1	C
D	3	D

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

# Distance Vector: 1st Iteration

34



loop:

**else if** (update  $D(V, Y)$  re-

**for all** destination

**if** (destination

$D(A, Y) = D(A, V)$

**else**

$D(A, Y) =$

$\min(D(A, V)$

$+ D(V, Y))$

**if** (there is a new min. for dest.  $Y$ )  
**send**  $D(A, Y)$  to all neighbors

**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

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

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

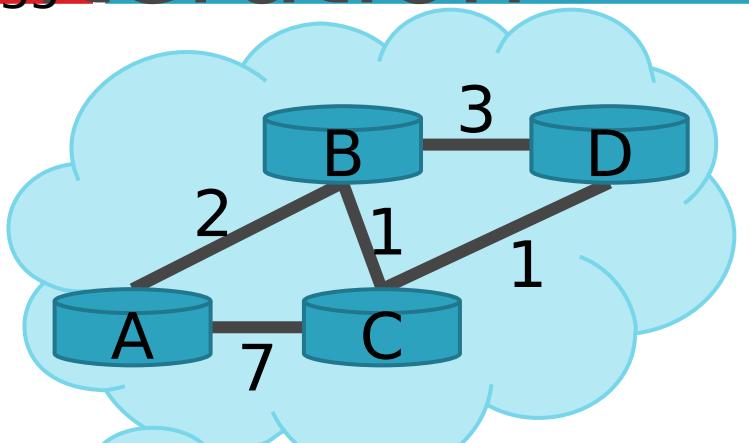
$= \min(8, 3 + 3, B = 5)$

Dest.	Cost	Next
B	1	B
D	1	D

Dest.	Cost	Next
B	3	B
C	1	C

# Distance Vector: End of 3rd Iteration

35



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

for all

if (

Do

else

$D(A, Y) =$

$\min(D(A, Y),$

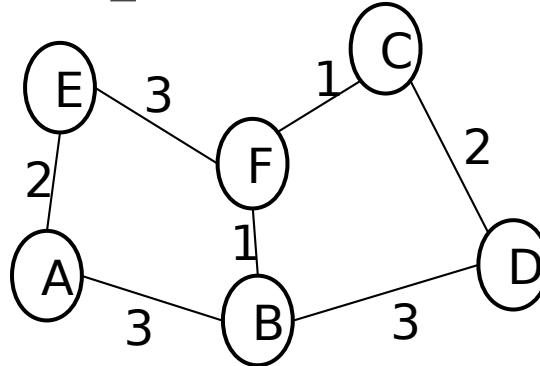
$D(A, V) + D(V, Y);$

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	
	cost	N. Hop
A	t	
B	3	B
C	$\infty$	-
D	$\infty$	-
E	2	E
F	$\infty$	-

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

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

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

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

	Új becsült késleltetés A-tól	
	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.     **Link Cost**

19.     **Algorithm fact**

20.     **Node B**

D	C	N
A	4	A
C	1	B

D	C	N
A	5	B
B	1	B

Good news travels

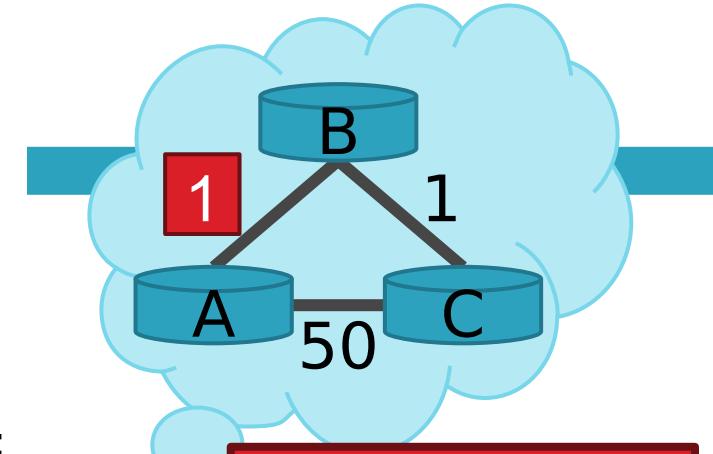
Algorithm terminates

Node C

D	C	N
A	1	A
C	1	B

D	C	N
A	5	B
B	1	B

Time



D	C	N
A	1	A
C	1	B

D	C	N
A	2	B
B	1	B

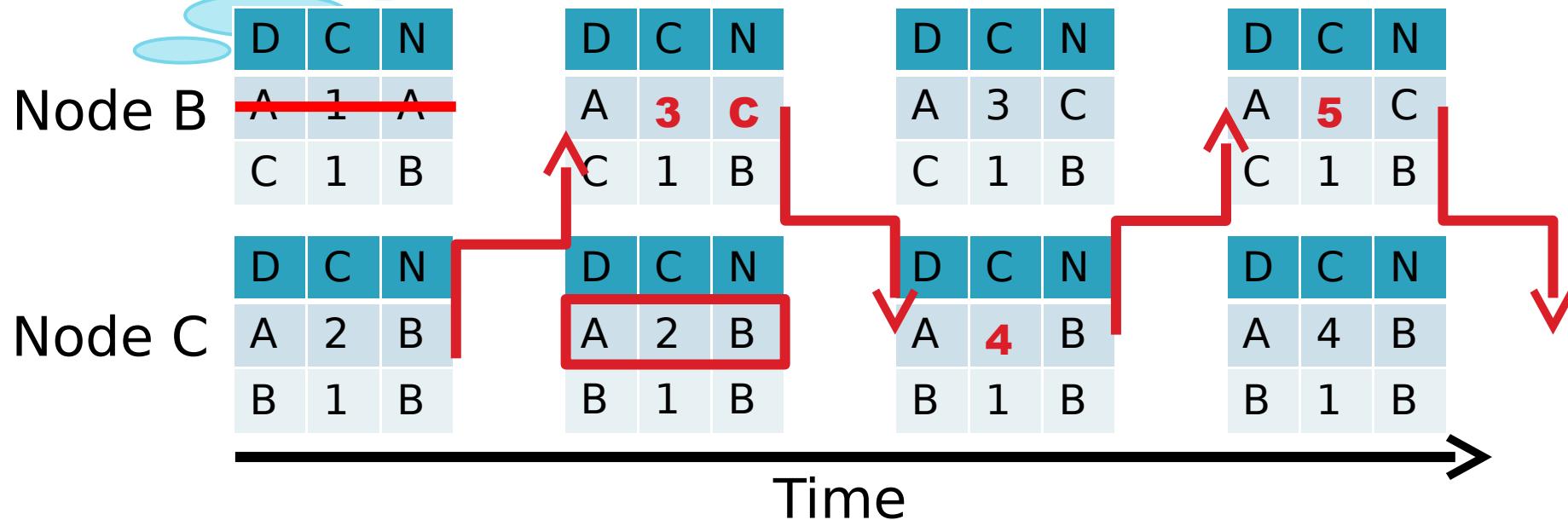
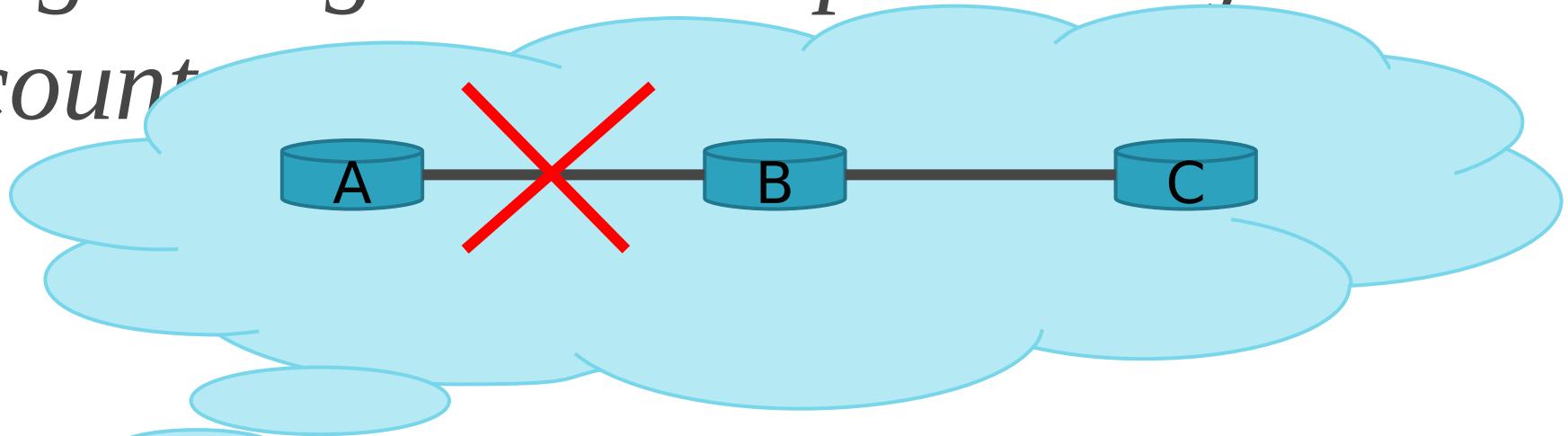
C	N
1	A
1	B

C	N
2	B
1	B

# Távolság vektor protokoll –

## <sup>38</sup> Végtelenig számolás problémája

(count)

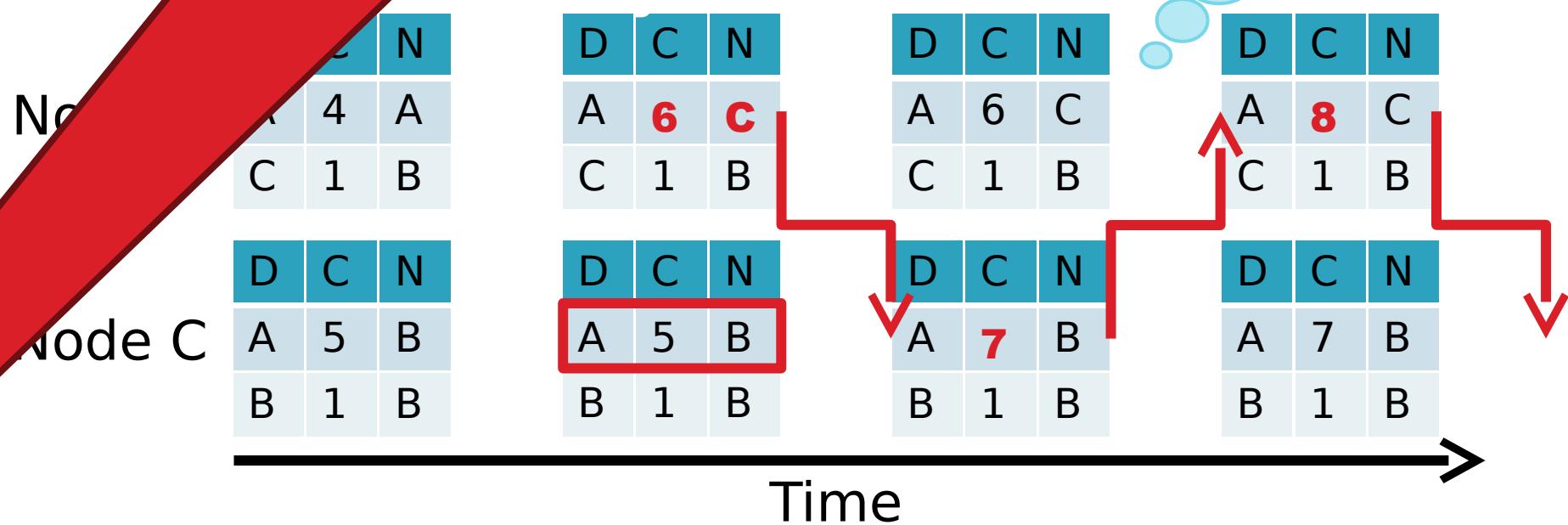


# Példa - Count to Infinity Problem

39

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

Bad news travels slowly



# Elosztott Bellman-Ford algoritmus – Végtelenig számolás problémája

- 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 Node B-t A-ba menő forgalmat
- 
- Diagram illustrating the split horizon with poisoned reverse mechanism. It shows three nodes (A, B, C) and their link costs:
- Node A: D 60 A
  - Node B: C 1 B
  - Node C: A 50 A
- The diagram shows the sequence of events over time:
- Initial state: Node A: D 60 A, Node B: C 1 B, Node C: A 50 A.
  - After a delay, Node B updates its routing table: D 51 C (link to A is now infinity).
  - After another delay, Node C updates its routing table: A 50 A (link to B is now infinity).
  - Final state: All nodes have updated their tables to reflect the correct shortest paths.
- Red arrows indicate the flow of information between the nodes, showing how Node C's update to infinity for Node B's link to Node A prevents the loop.

# Vége

□ Köszönöm a figyelmet!

# Számítógépes hálózatok

Nyolcadik előadás – Hálózati réteg, forgalomirányítási protokollok, címzés

Készítette: Ács Zoltán  
Kiegészítette: Laki Sándor

# 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
<b>Hálózati réteg</b>
Adatkapcsolati réteg
Fizikai réteg

# Hálózati réteg – forgalomirányítási algoritmusok

# 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 korétek és a szomszédok lista. minden szomszédhoz megadják a felé tapasztalható késleltetést. Az előállítás történhet periodikusan vagy hiba esemény esetén. (Un. LSA – Link State

# 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 másodpercenként 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 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.

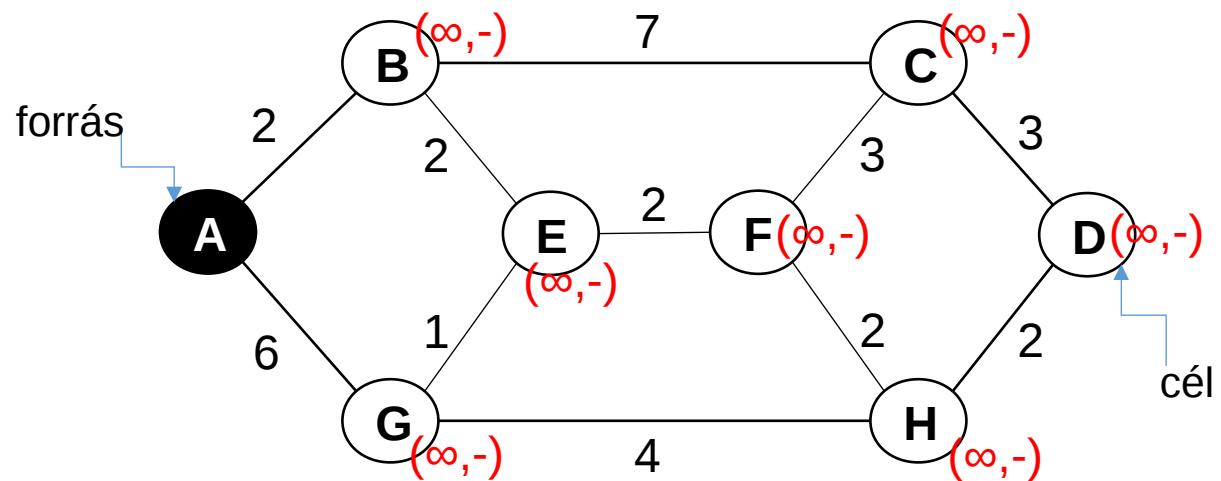
# Dijkstra algoritmus (1959)

- 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.

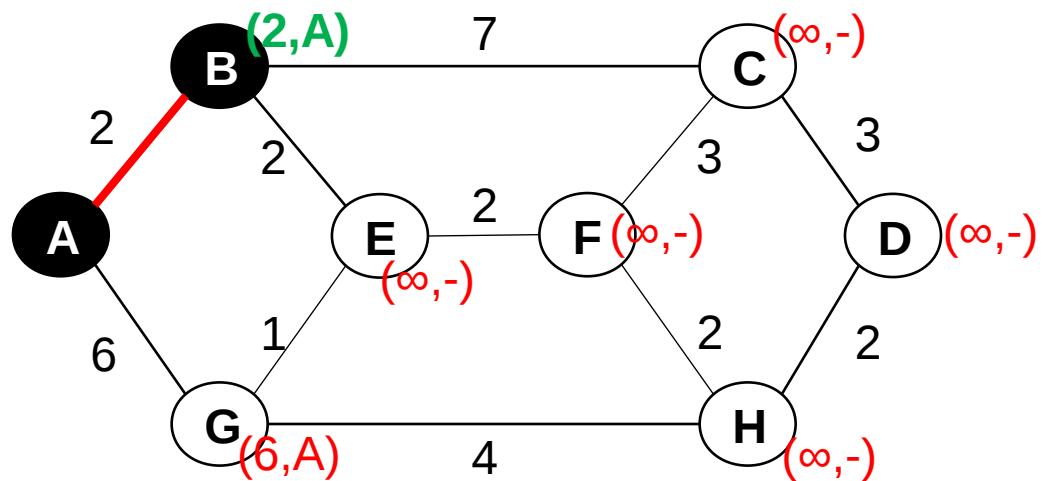
# Dijkstra algoritmus - Példa



E' = { }

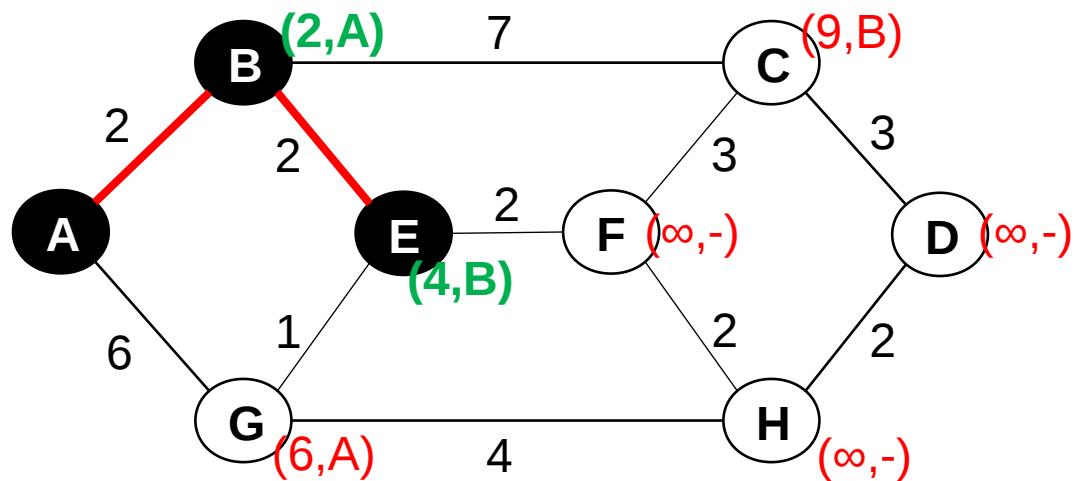
Q = { (B, 2), (G, 6) }

# Dijkstra algoritmus - Példa



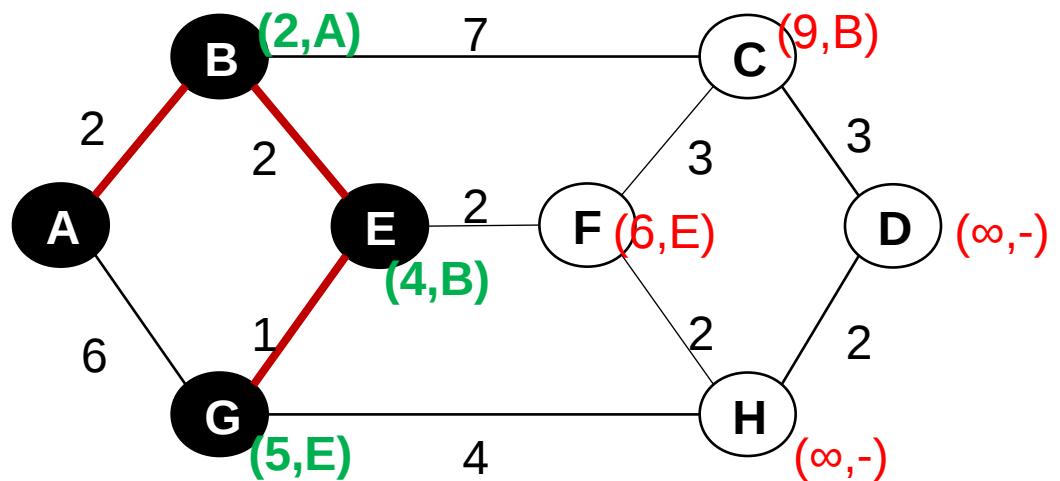
```
E' = { (A, B) }  
Q = { (E, 4), (G, 6),  
      (C, 9) }
```

# Dijkstra algoritmus - Példa



$$E' = \{ (A, B), (B, E) \}$$
$$Q = \{ (G, 5), (F, 6), (C, 9) \}$$

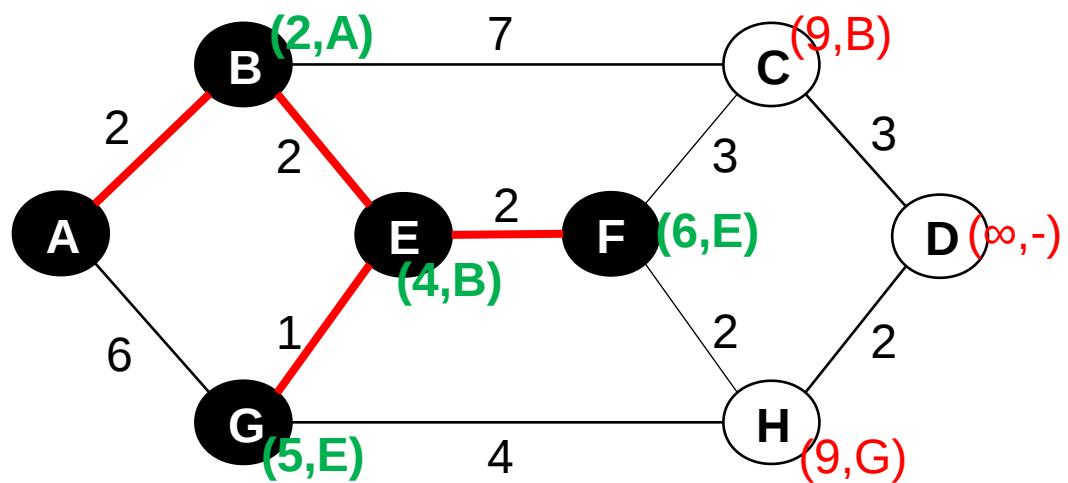
# Dijkstra algoritmus - Példa



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

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

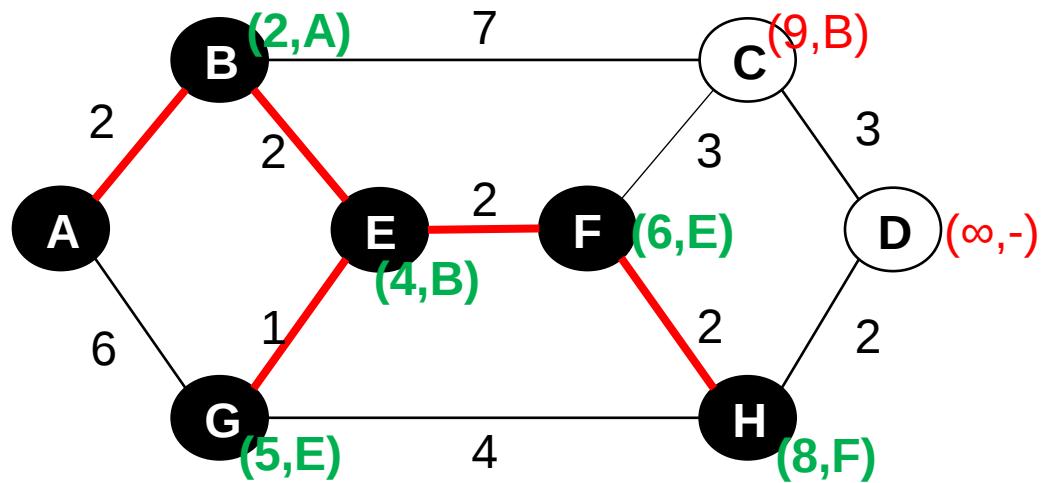
# Dijkstra algoritmus - Példa



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

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

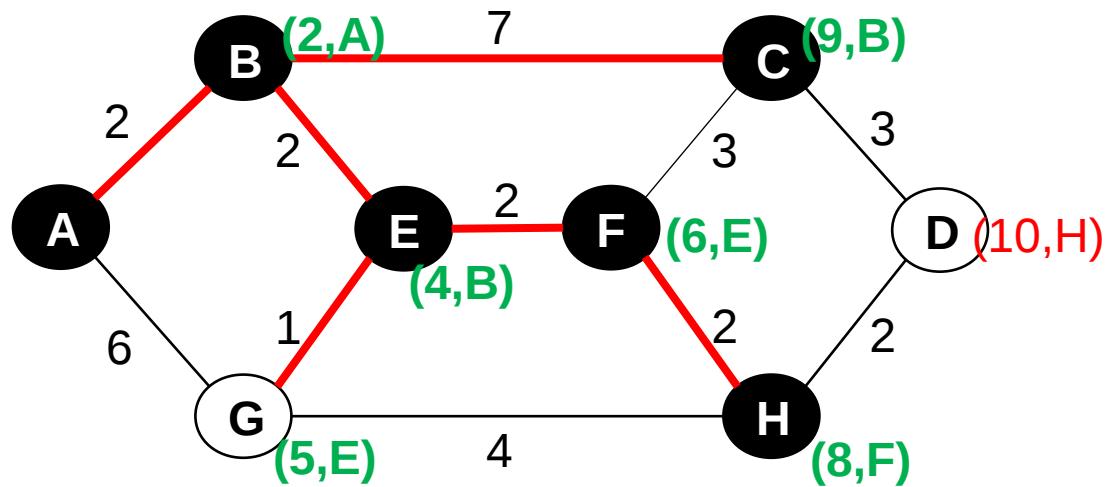
# Dijkstra algoritmus - Példa



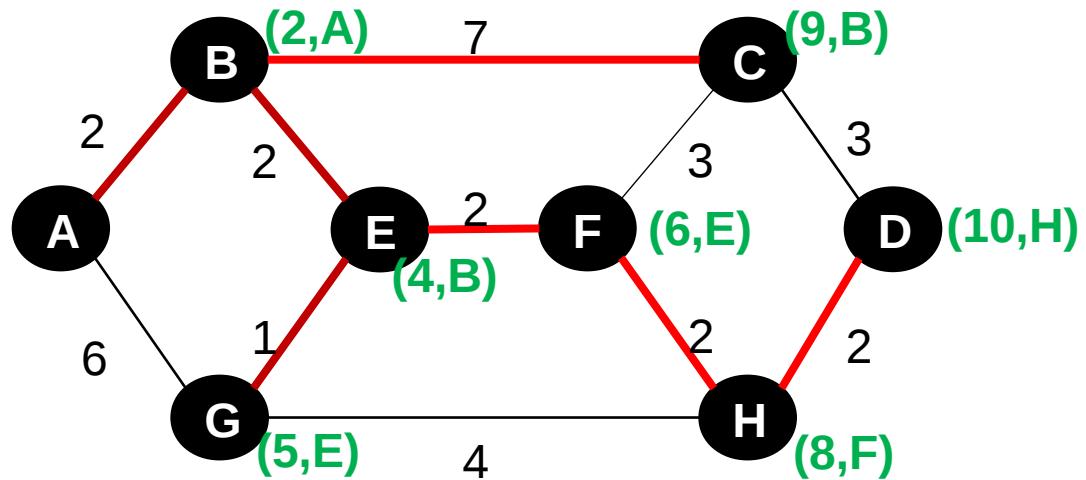
$$E' = \{ (A, B), (B, E), (E, G), (E, F), (F, H) \}$$

$$Q = \{ (C, 9), (D, 10) \}$$

# 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 = { }
```

# Díjkstra algoritmus pszeudo-kód

## ITERÁCIÓS LÉPÉSEK

JAVÍTÓ ÚT

ÚJ ÚT

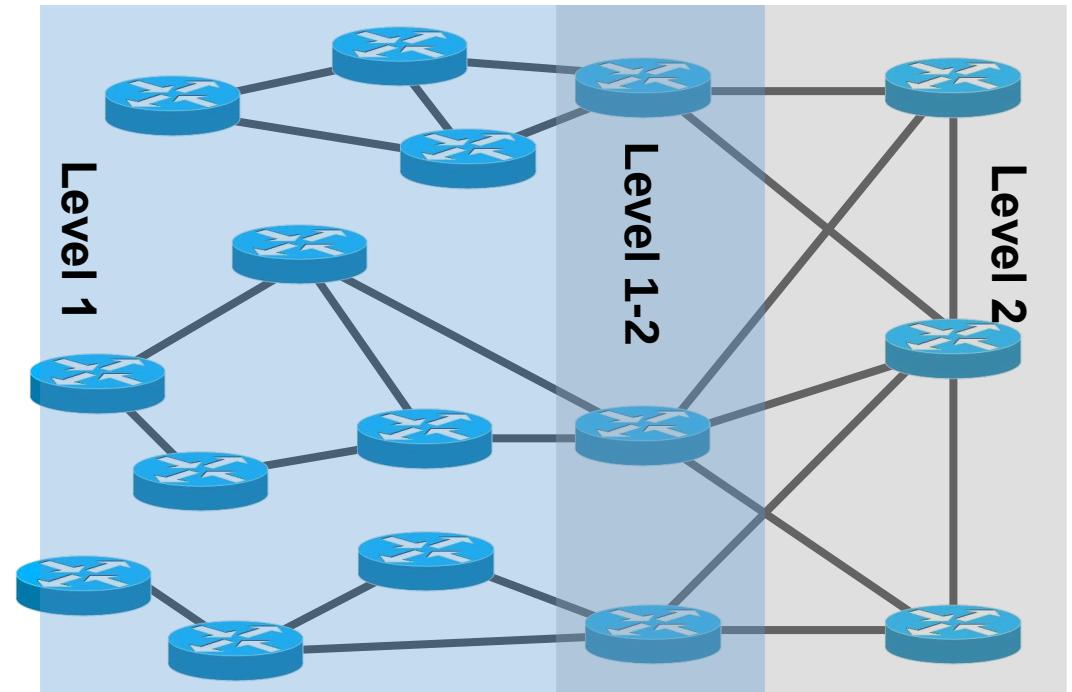
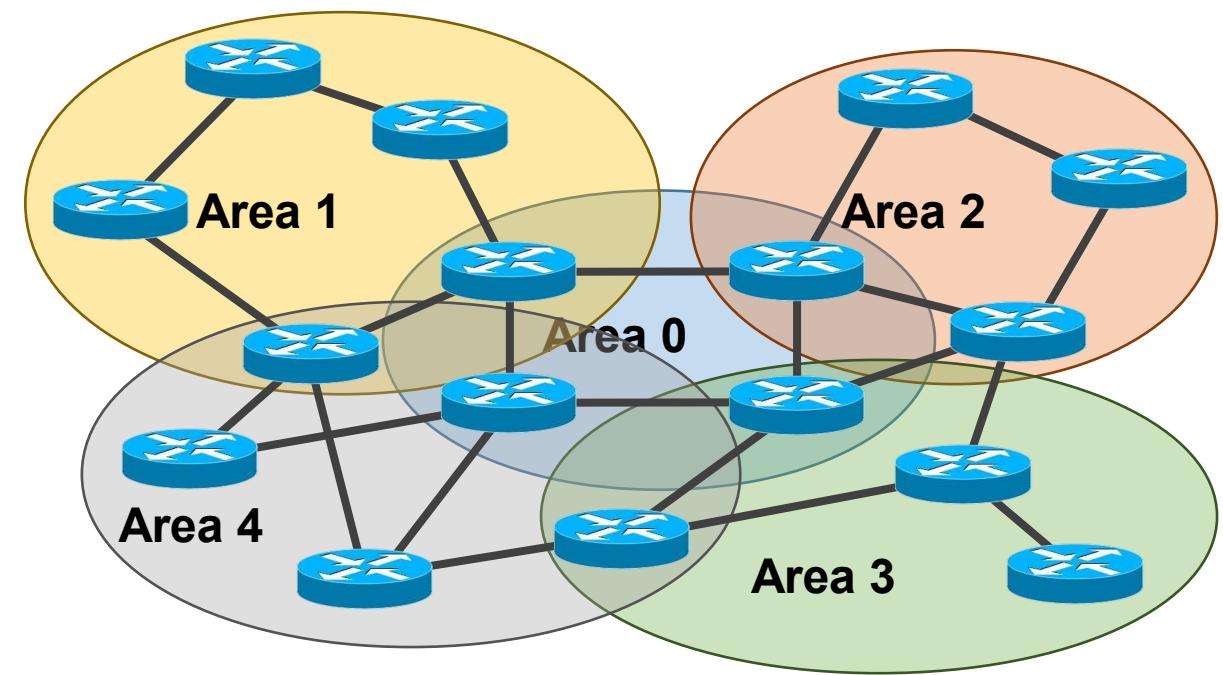
## INICIALIZÁCIÓS FÁZIS

# OSPF vs. IS-IS

- Két eltérő implementáció a link-state routing stratégiának
  - OSPF
  - IS-IS
- Cégek és adatközpontok
- Több lehetőséget támogat
- IPv4 felett
  - LSA-k IPv4 feletti küldése
  - OSPFv3 szükséges az IPv6-hoz
- 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

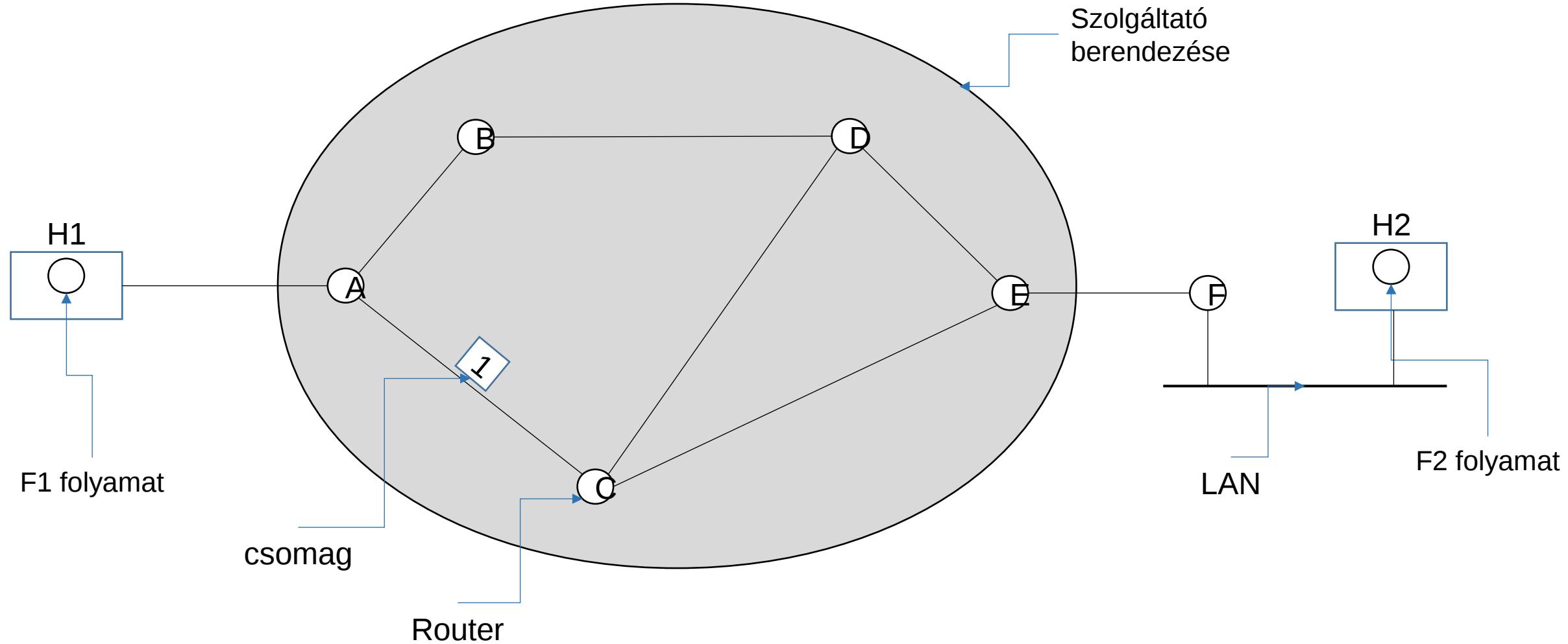


# Link State vs. Distance Vector

	Link State	Distance Vector
Message Complexity	$O(n^2 \cdot e)$	$O(d \cdot n \cdot k)$
Time Complexity	$O(n \cdot \log n)$	$O(n)$
Convergence Time	$O(1)$	$O(k)$
Robustness	<ul style="list-style-type: none"><li>Nodes may advertise incorrect link costs</li><li>Each node computes their own table</li></ul>	<ul style="list-style-type: none"><li>Nodes may advertise incorrect path cost</li><li>Errors propagate due to sharing of DV tables</li></ul>

- 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 bocsátott hálózati címeknek egységes számozási rendszert kell alkotniuk, még *LAN*-ok és *WAN*-ok esetén is.

## Szolgálatok két fajtáját különböztetik meg

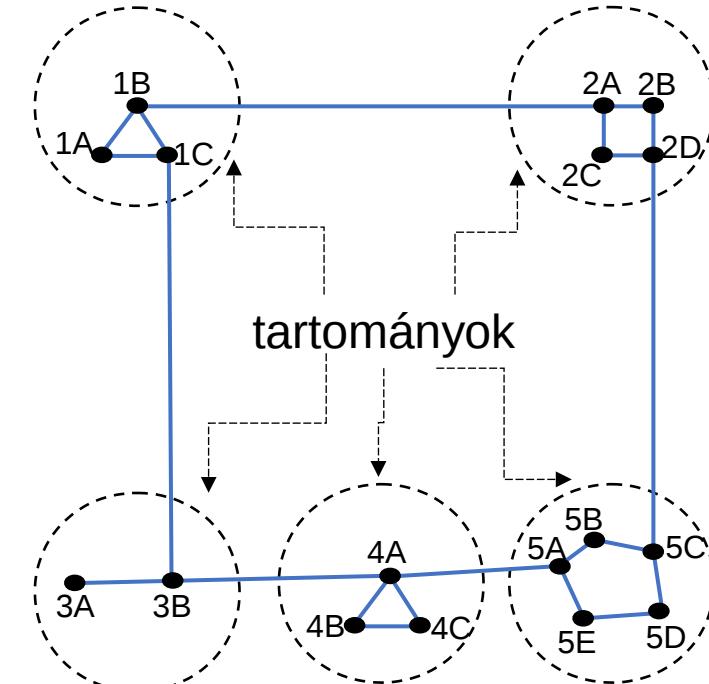
- Összeköttetés nélküli szolgálat (*Internet*)
  - datagram alhálózat
- Összeköttetés alapú szolgálat (*ATM*)
  - virtuális áramkör alhálózat

# Hálózati réteg – forgalomirányítás

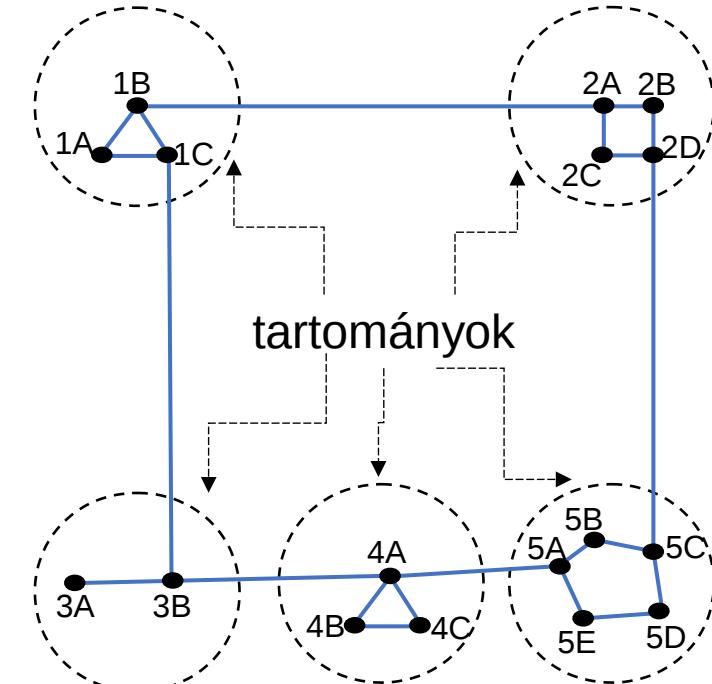
# Hierarchikus forgalomirányítás

## 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



# Adatszóró forgalomirányítás

- **Adatszórás** ( vagy angolul *broadcasting*) – egy csomag mindenholára 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, minden egyik 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

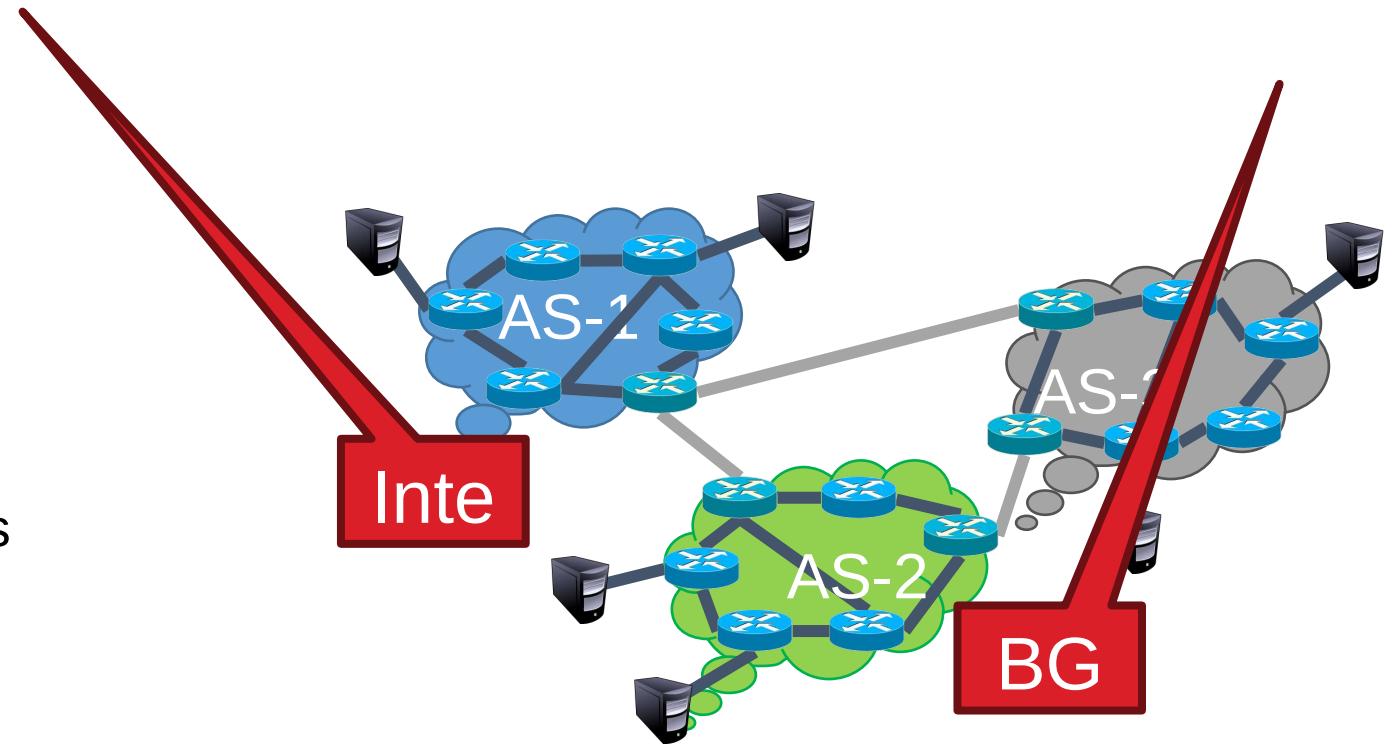
- **Többes-küldés** ( vagy angolul *multicasting*) – egy csomag meghatározott csoporthoz történő egyidejű küldése.

## 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 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

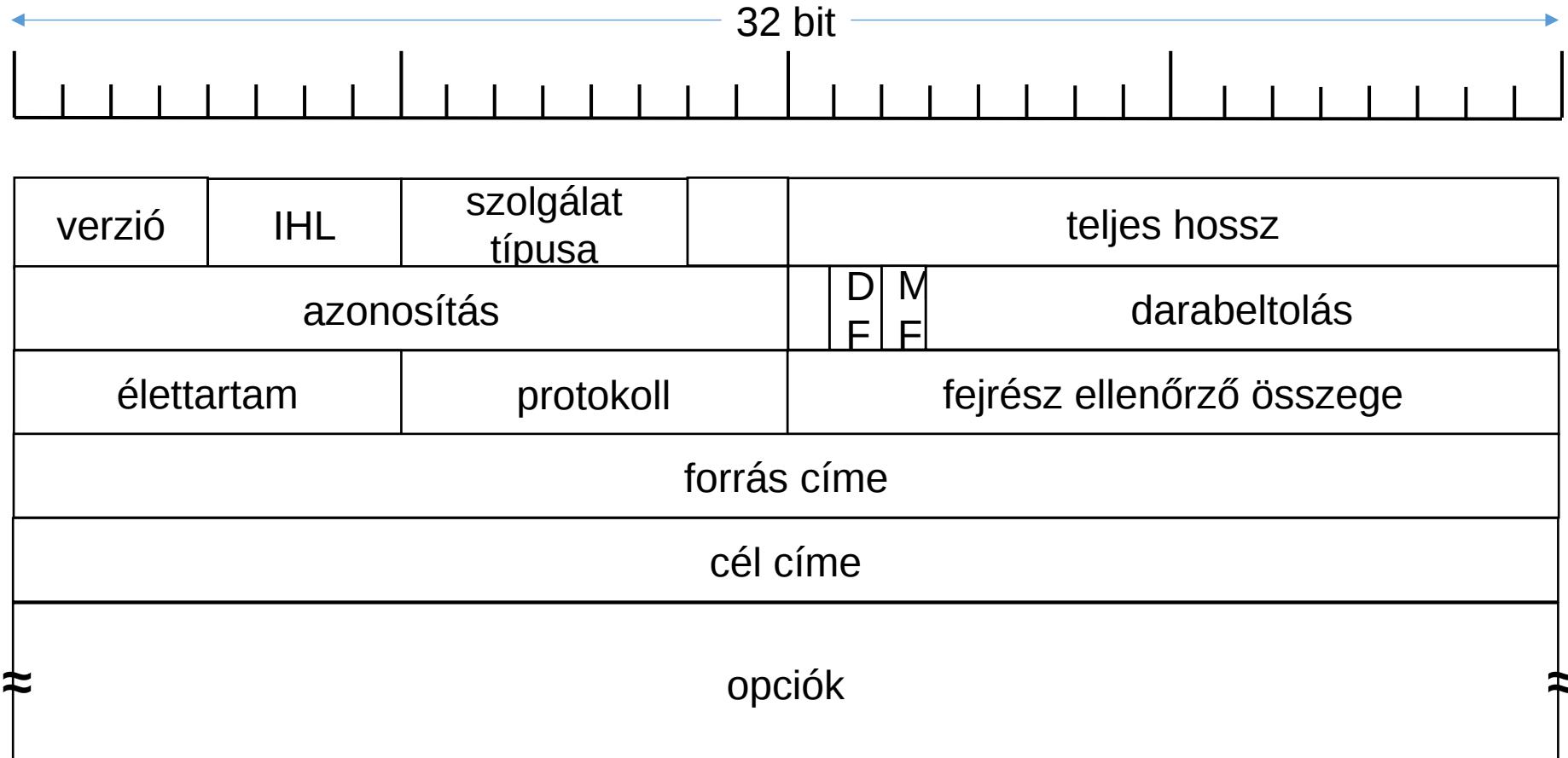
- A hálózati réteg szintjén az internet autonóm rendszerek összekapcsolt együttesének tekinthető.
  - Nincs igazi szerkezete, de számos főbb *gerinchálózata* létezik.
  - A gerinchálózatokhoz csatlakoznak a területi illetve regionális hálózatok.
  - A regionális és területi hálózatokhoz csatlakoznak 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)
- **opciónk:** következő verzió bővíthetősége miatt hagyták benne. Eredetileg 5 opción volt. (router-ek általában figyelmen kívül hagyják)

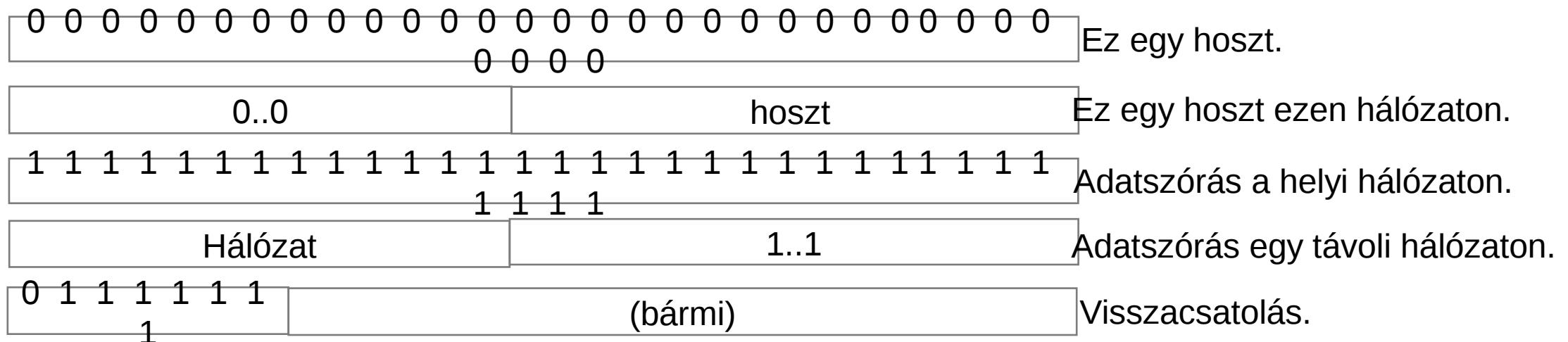
# IP cím

- 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.

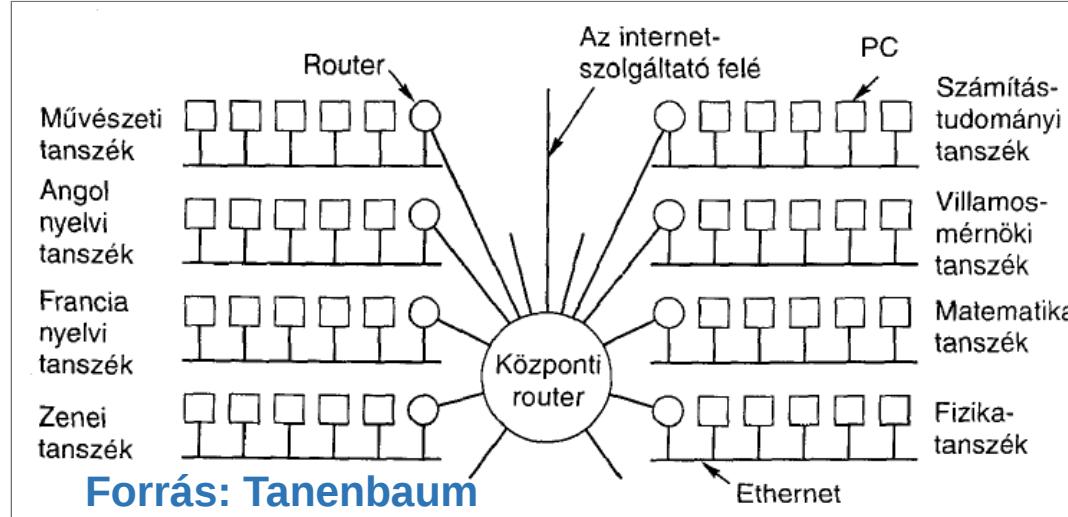


# IP cím

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



# IP cím – alhálózatok

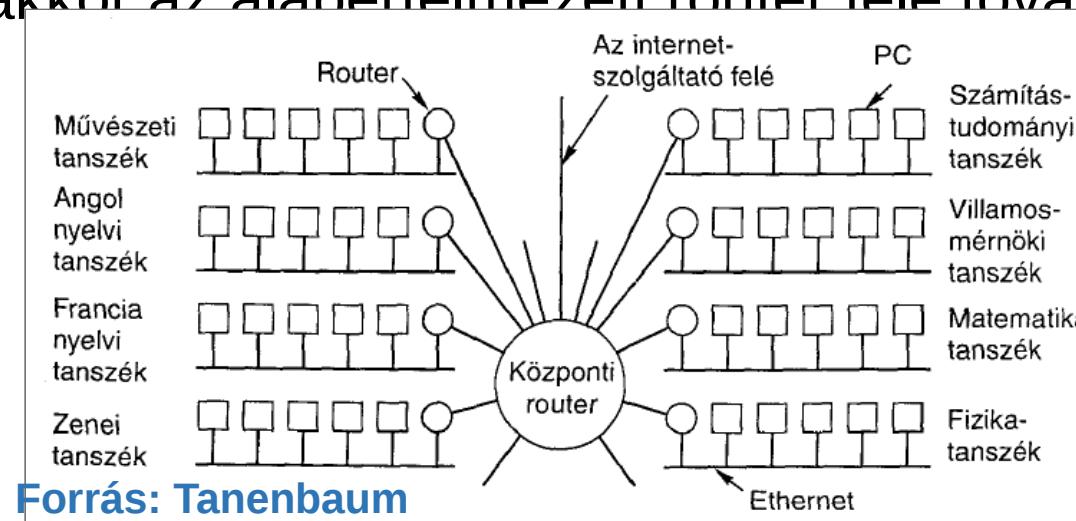


- Az azonos hálózatban lévő hosztok ugyanazzal a hálózatszámmal rendelkeznek.
- Egy hálózat belső felhasználás szempontjából több részre osztódhat, de a külvilág számára egyetlen hálózatként jelenik meg.
  - Alhálózat (avagy angolul *subnet*)

# 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 alanértelmezett router felé továbbítják a csomagot.



# 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 bejeavezés keletkezik.

# 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

- 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.

Kimaszkolás eredménye

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 feltezzük, 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.

# Forgalomirányítási tábla példa

<b>Network Destination</b>	<b>Netmask</b>	<b>Gateway</b>	<b>Interface</b>	<b>Metric</b>
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

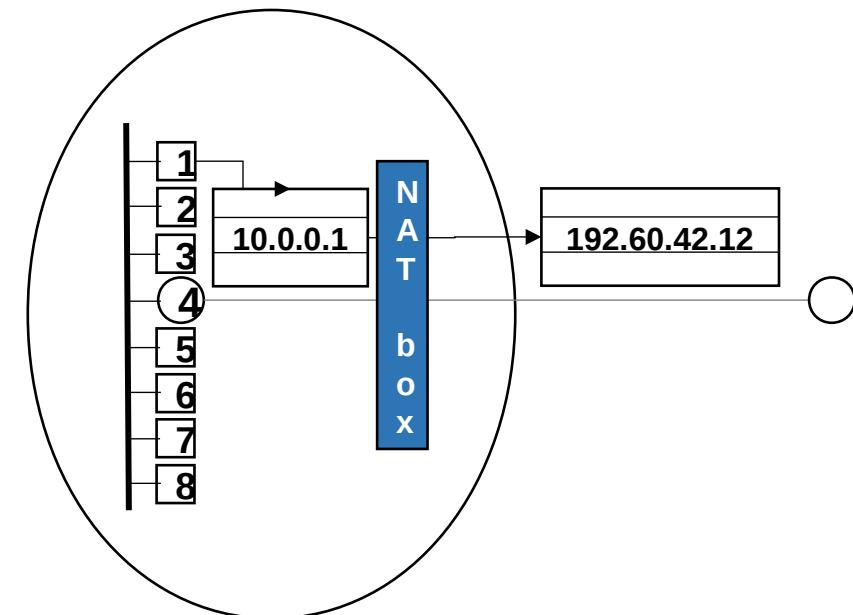
- Gyors javítás az IP címek elfogyásának problémájára. (hálózati címfordítás)

## Alapelvek

- Az internet forgalomhoz minden cégnak 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 kezelní.
  - 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!

# Számítógépes Hálózatok

## 7a. Előadás: Hálózati réteg

Based on slides from

# 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.

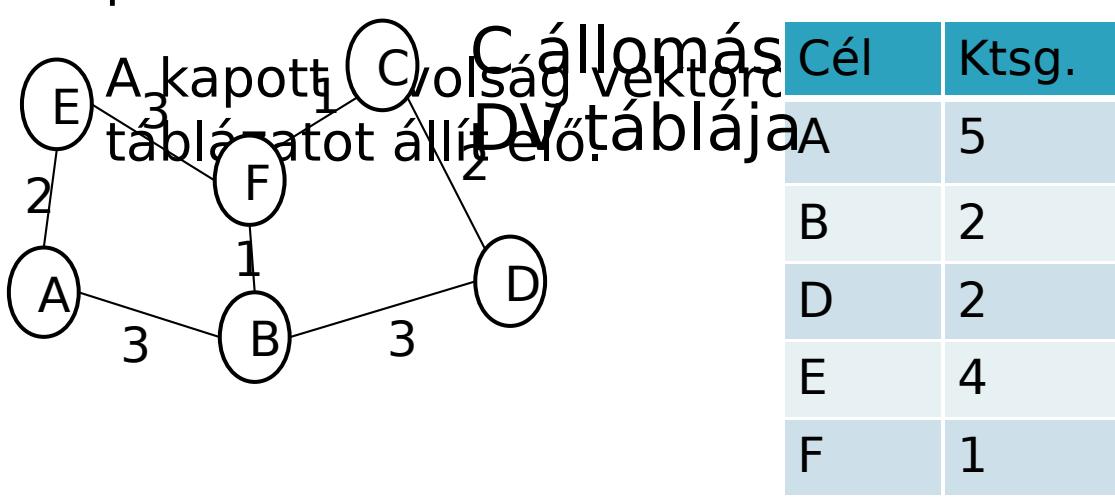
# Távolságvektor alapú forgalomirányítás

## Flosztott Bellman-Ford

**Környezet és működés**

▫ minden csomópont csak a közvetlen szomszédjaival kommunikálhat.

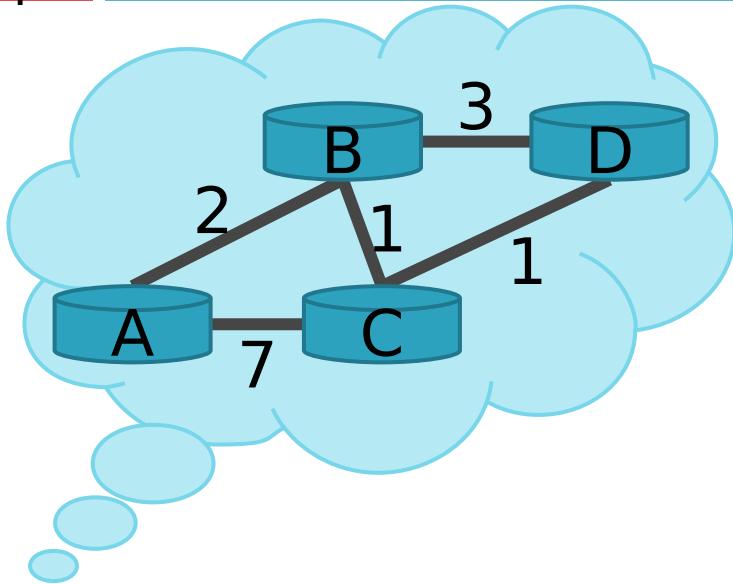
- Aszinkron működés.
- minden állomásnak van saját távolság vektora. Ezt periodikusan elküldi a direkt szomszédoknak.



- Nincs bejegyzés C-nél, mert nincs csomópont új
- Kezdetben csak a közvetlen szomszédokhoz van info
- Más célállomások költsége =  $\infty$

# Distance Vector Initialization

4



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

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

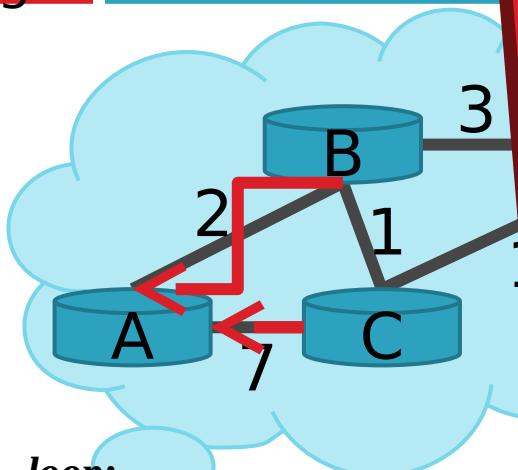
Node B		
Dest.	Cost	Next
A	2	A
C	1	C
D	3	D

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

# Distance Vector: 1st Iteration

5



loop:

**else if** (update  $D(V, Y)$  re-

**for all** destination

**if** (destination

$D(A, Y) = D(A, V)$

**else**

$D(A, Y) =$

$\min(D(A, V)$

$+ D(V, Y))$

**if** (there is a new min. for dest.  $Y$ )  
**send**  $D(A, Y)$  to all neighbors

**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

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

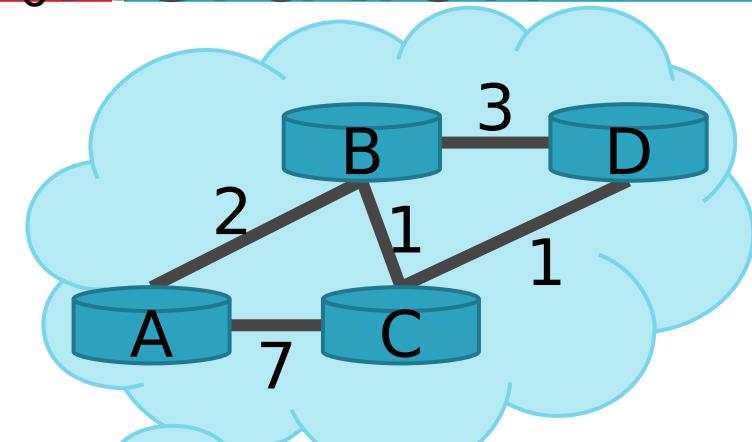
$D(A, C)$

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

Dest.	Cost	Next
B	1	B
D	1	D

Dest.	Cost	Next
B	3	B
C	1	C

# Distance Vector: End of 3rd Iteration



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

Red arrow points from Node A to Node B.

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

loop:

```
else if  
for all  
if (c  
D)  
else
```

$D(A, Y) = \min(D(A, Y), D(A, V) + D(V, Y))$ ;

**if** (there is a new min. for dest. Y)  
**send**  $D(A, Y)$  to all neighbors  
**forever**

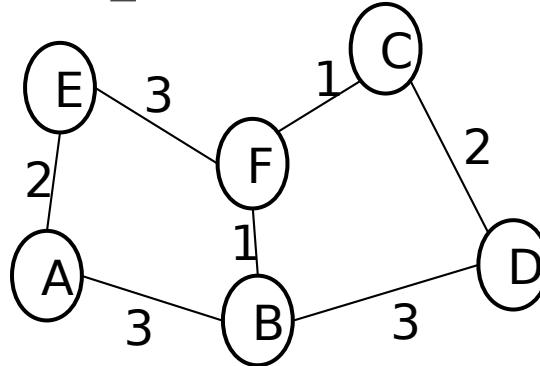
- Nothing changes, algorithm terminates
- Until something changes

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

Red arrow points from Node B to Node A.

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

# Elosztott Bellman-Ford algoritmus – példa



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

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

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

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

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

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

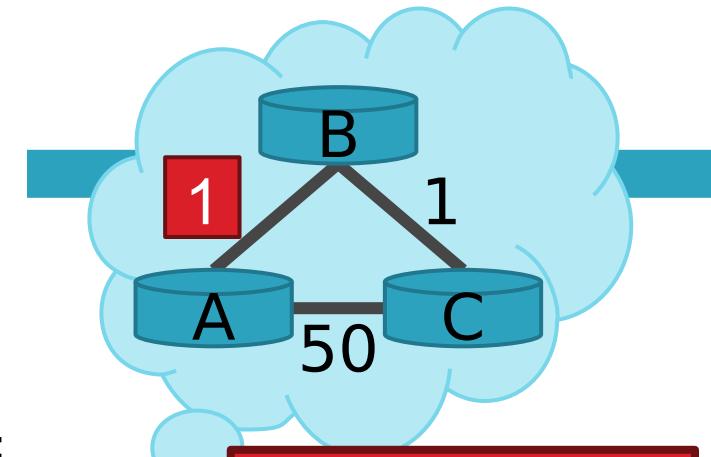
7.  
8.  
9.  
10.  
11.  
12.  
13.  
14.  
15.  
16.  
17.  
18.  
19.  
20.

**loop:**

**wait** (link cost update or update message)  
**if** ( $c(A,V)$  changes by  $d$ )  
**for all** destinations  $Y$  through  $V$  **do**  
 $D(A,Y) = D(A,Y) + d$   
**else if** (update  $D(V,Y)$  received from  $V$ )  
**for all** destinations  $Y$  **do**  
**if** (destination  $Y$  through  $V$ )  
 $D(A,Y) = D(A,V) + D(V,Y);$   
**else**  
 $D(A,Y) = \min(D(A,Y), D(A,V) + D(V,Y));$

Link Cost  
Algorithm

Good news travels  
fast



Node B

D	C	N
A	4	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	2	B
B	1	B

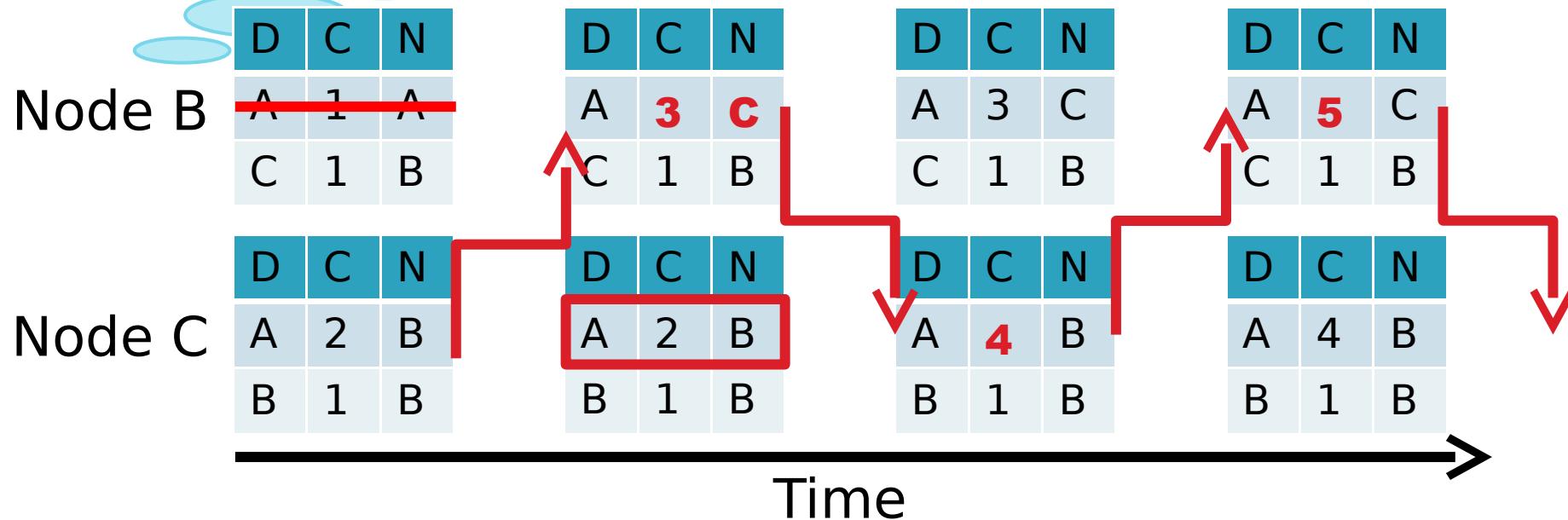
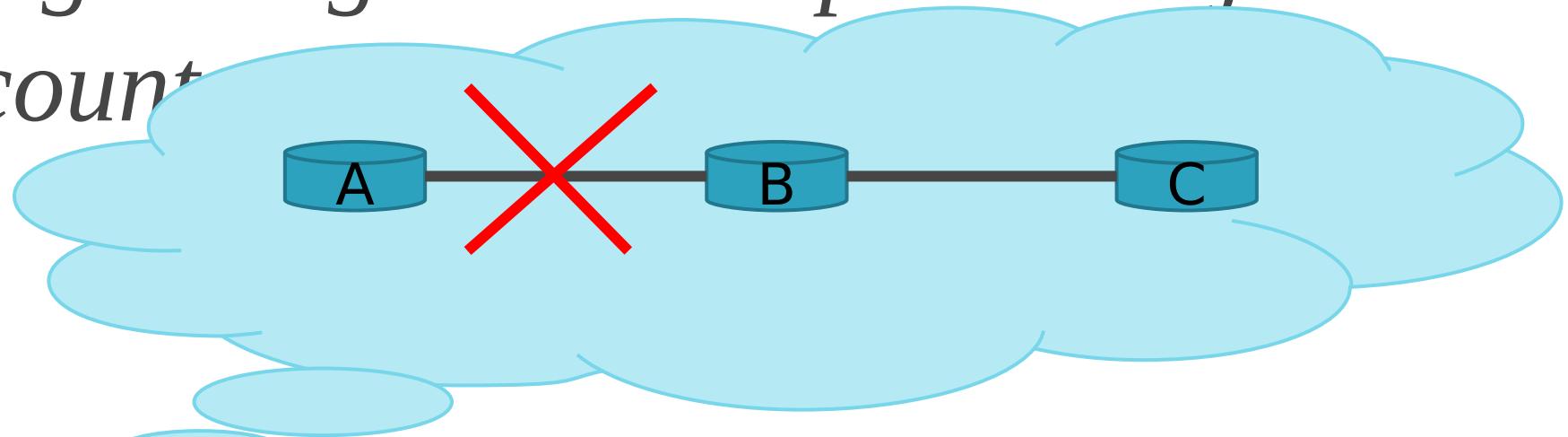
Algorithm  
terminates

Time

# Távolság vektor protokoll –

## „Végtelenig számolás problémája

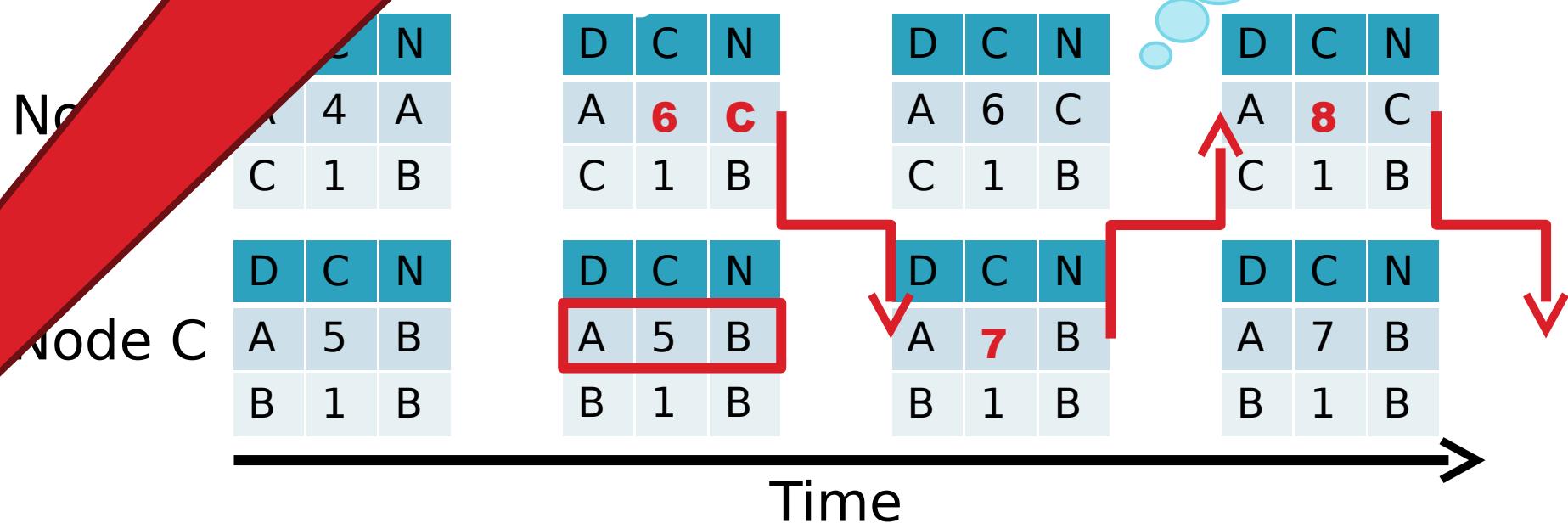
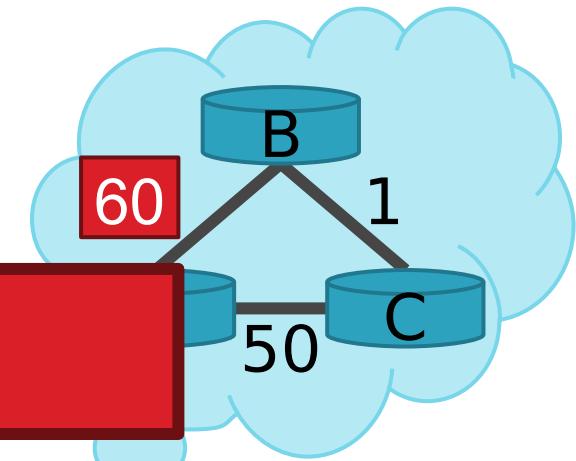
(count)



# Példa - Count to Infinity Problem

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

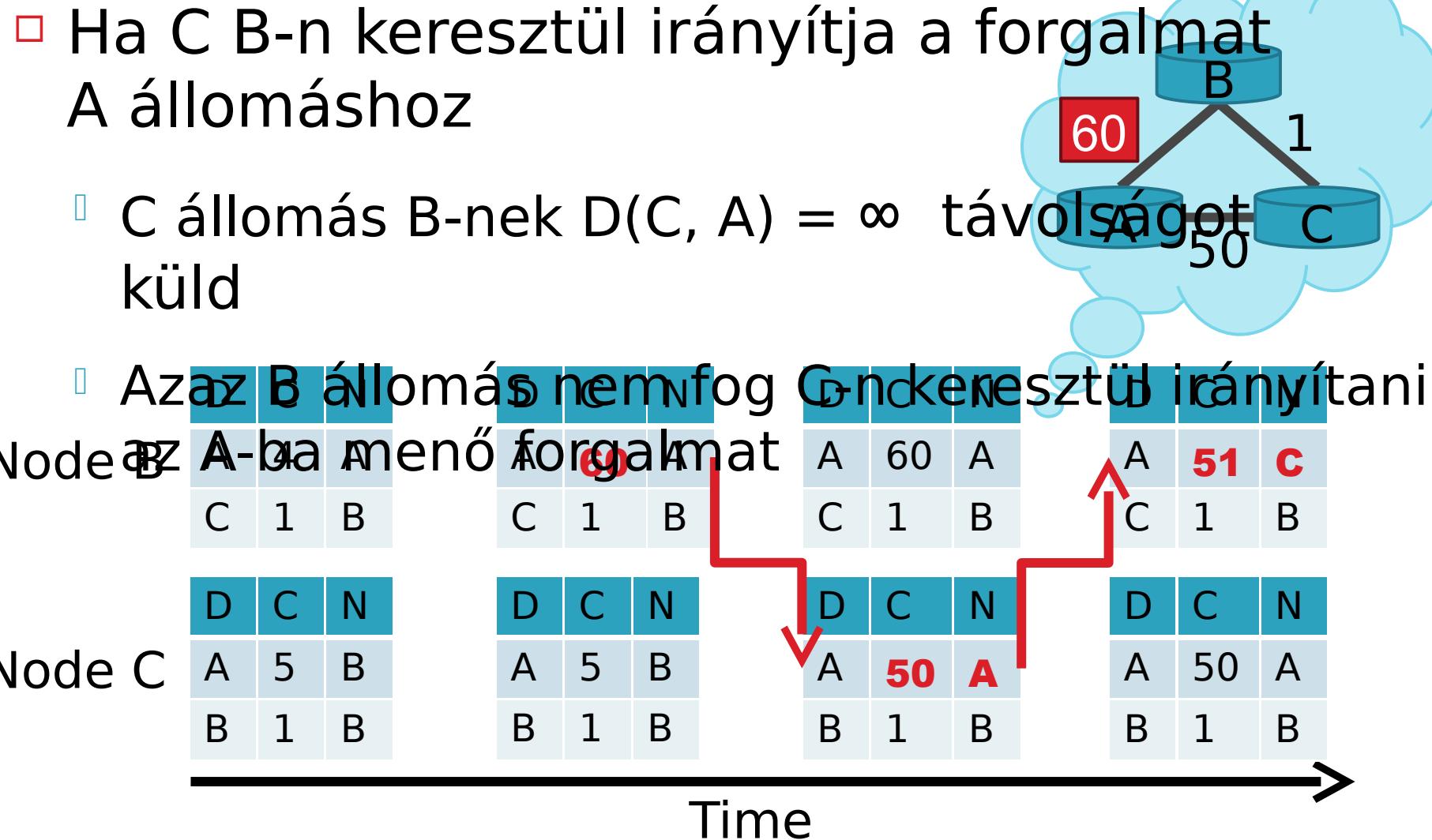
Bad news travels slowly



# Elosztott Bellman-Ford algoritmus – Végtelenig számolás problémája

- 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



# Vége

□ Köszönöm a figyelmet!

# Számítógépes Hálózatok

## 8. Előadás: Hálózati réteg II.

Based on slides from

# Hálózati réteg

2



## □ Feladatok:

- Csomagok végpontok közötti leszállítása, akár több közbenső állomáson keresztül

## □ Kihívások:

- Címek ábrázolása
- Útvonal meghatározás
  - Skálázhatóság
  - Konvergencia

# Forgalomirányítási tábla példa

3

<b>Network Destination</b>	<b>Netmask</b>	<b>Gateway</b>	<b>Interface</b>	<b>Metric</b>
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 – Network Address Translation

4

- Gyorsjavítás az IP címek elfogyásának problémájára.  
(hálózati címfordítás)

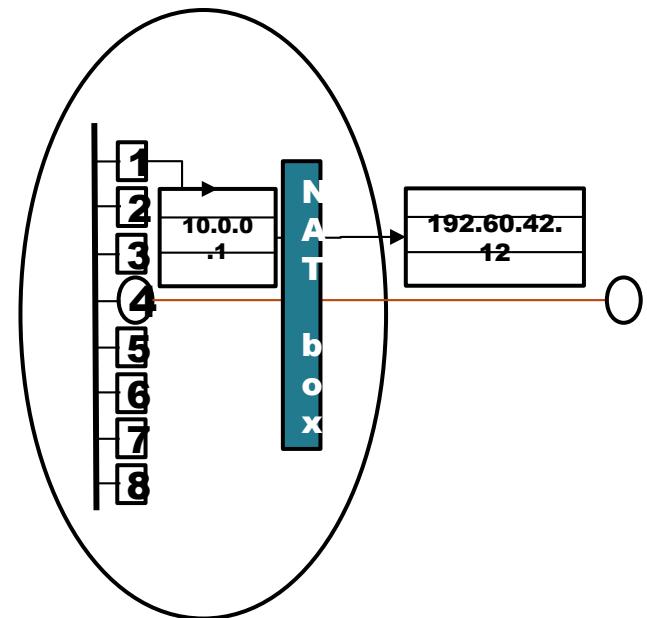
## Alapelvek

- Az internet forgalomhoz minden cégnak 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;

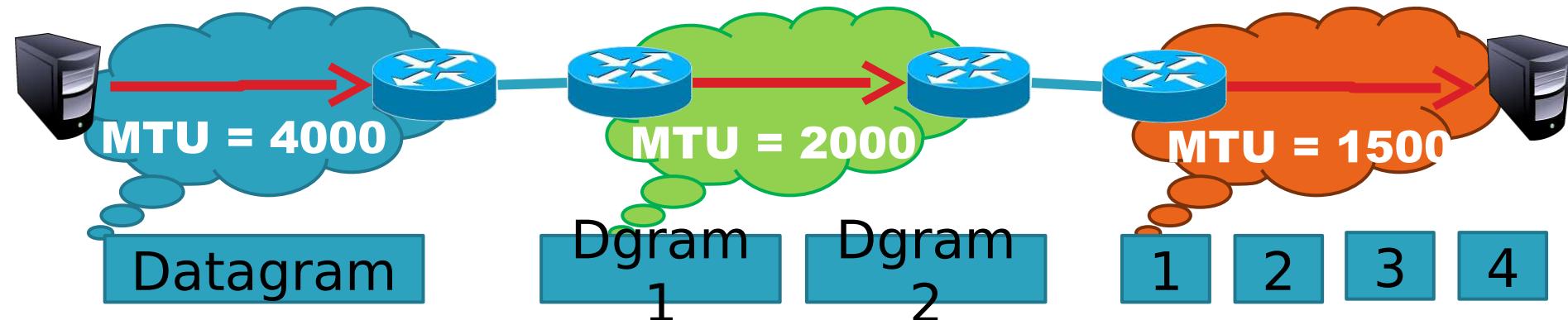
# NAT

5

- 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 kezelní.
  - A fordítási táblázatban benne van az eredeti IP és forrás port.
- **Ellenérvek:** sérti az IP



# IP Fragmentation – IP Fragmentáció (darabolás)



- Probléma: minden hálózatnak megvan a maga MTU-ja
  - MTU: Maximum Transmission Unit - lényegében a maximális használható csomag méret egy hálózatban
  - DARPA/Internet alapelve: hálózatok heterogénnek lehetnek

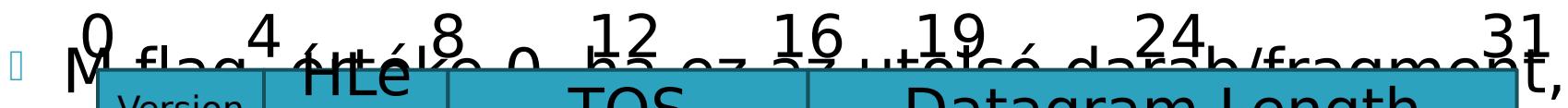
# IP fejléc: 2. szó

7

- Identifier (azonosító):

- egyedi azonosító minden IP datagramhoz (csomaghoz)

- Flags (jelölő bitek):

- 

- Offset (utca szélessége):

- 

- 

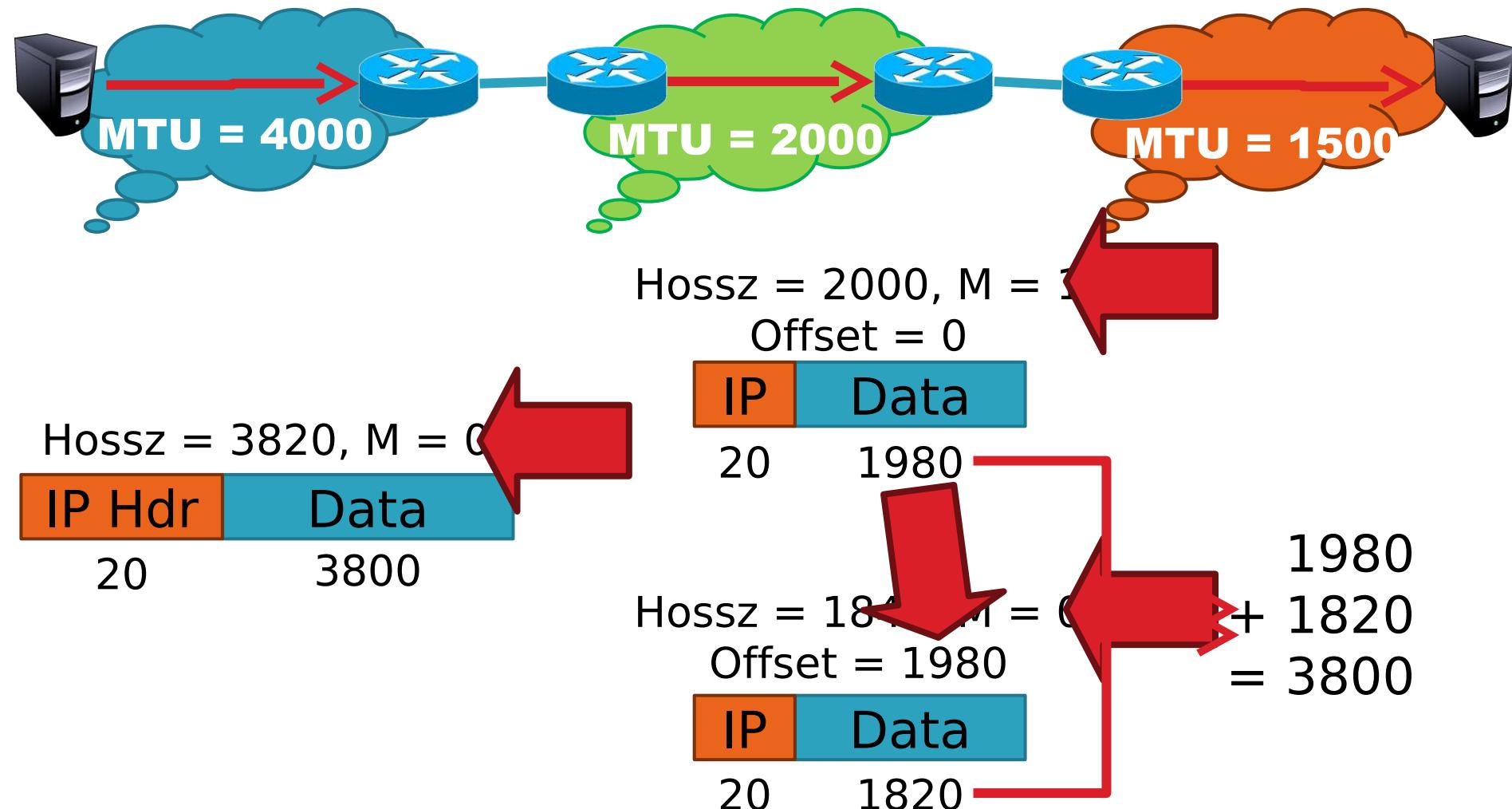
- 

- 

- 

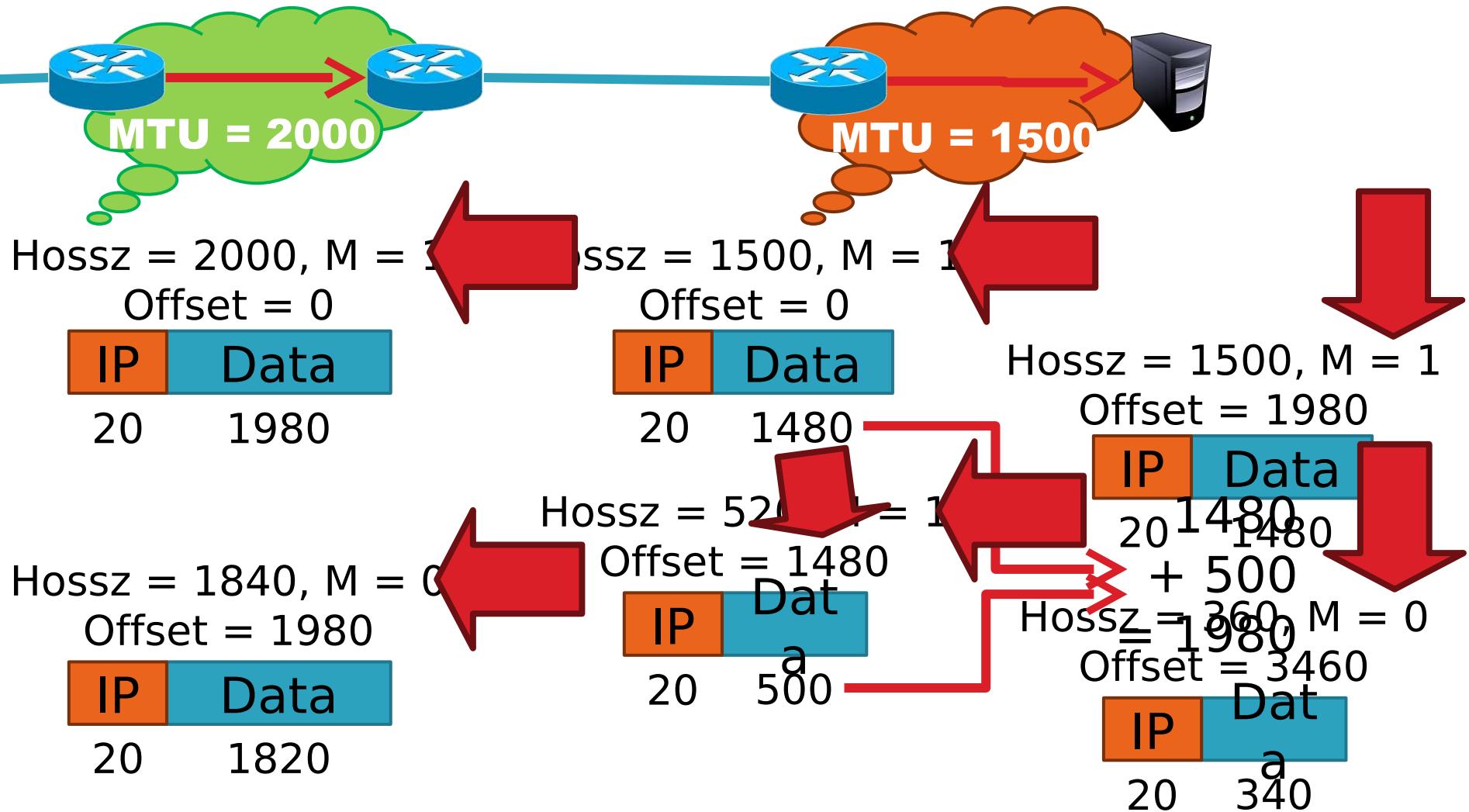
# Példa

8



# Példa

9



# IP csomag helyreállítása

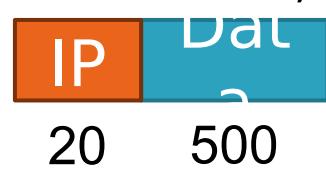
10

ossz = 1500, M = 1, Offset = 0



20 1480

ossz = 520, M = 1, Offset = 1480



20 500

ossz = 1500, M = 1, Offset = 1980



20 1480

ossz = 360, M = 0, Offset = 3460



20 340

- A végponton történik
- M = 0, akkor ebből a darabból tudjuk a teljes adatmennyiséget

▀ Hossz - IPHDR\_hossz  
+ Offset

▀ 360 - 20 + 3460 =  
3800

- Kihívások:

# Fragmentáció

11

- Az Internet esetén
  - Elosztott és heterogén
    - minden hálózat maga választ MTU-t
  - Kapcsolat nélküli datagram/csomag alapú protokoll
    - minden darab tartalmazza a továbbításhoz szükséges összes információt
    - a darabok függetlenül kerülnek leszállításra, akár különböző útvonalon keresztül
  - Legjobb szándék elve szerint (best effort)
    - A routerok és a fogadó is oldhatat darabokat

# Fregmantáció a valóságban

12

- A fragmentáció költséges
  - I Memória és CPU költés a csomag visszaállításához
  - I Ha lehetséges, el kell kerülni
- MTU felderítő protokoll
  - I Csomagküldés a “don’t fragment” flag bittel
  - I Folyamatosan csökkentjük a csomag méretét, amíg egy meg nem érkezik
  - I Lehetséges “can’t fragment” hiba egy routertől, ami közvetlenül tartalmazza az adott hálózatban

IPv6

# Fogyó IPv4 címek

14

- Probléma: az IPv4 címtartomány túl kicsi
  - $2^{32} = 4,294,967,296$  lehetséges cím
  - Ez kevesebb mint egy emberenként
- A világ egy részén már nincs kiosztható IP blokk

Régió	Regional Internet Registry (RIR)	Utolsó IP blokk kiosztása
Asia/Pacific	APNIC	April 19, 2011
Europe/Middle East	RIPE	September 14, 2012
North America	ARIN	13 Jan 2015 (Projected)
South America	LACNIC	13 Jan 2015 (Projected)
Africa	AFRINIC	17 Jan 2022(Projected)

# IPv6

15

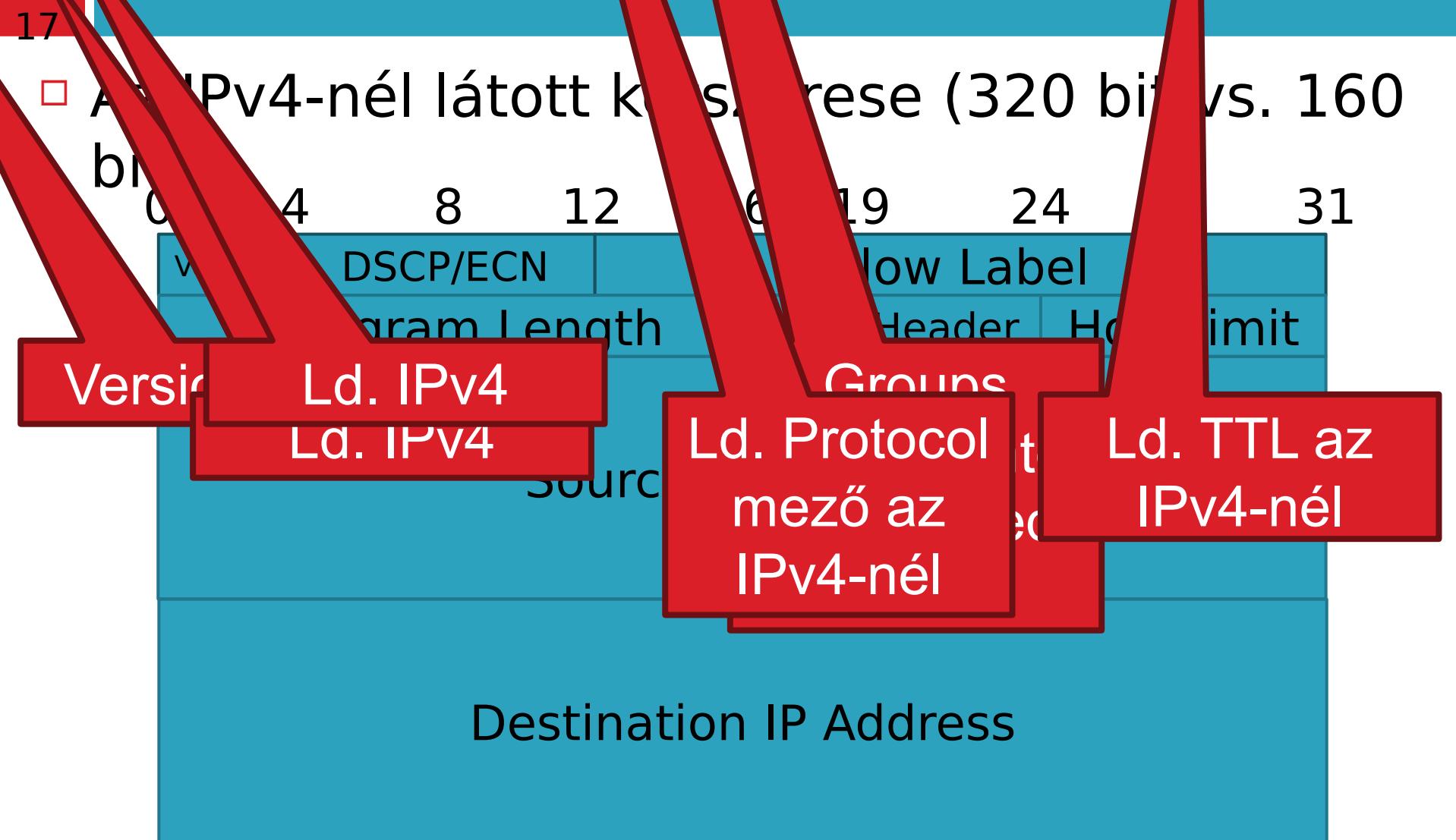
- IPv6, 1998(!)-ban mutatták be
  - 128 bites címek
  - $4.8 * 1028$  cím/ember
- Cím formátum
  - 16 bites értékek 8 csoportba sorolva (':'-tal elválasztva)
  - minden csoport elején szereplő nulla sorozatok elhagyhatók
  - Csupa nulla csoportok elhagyhatók , ekkor '::'

# IPv6

16

- Ki tudja a localhost IPv4 címét?
  - 127.0.0.1
- Mi ez az IPv6 esetén?
  - ::1

# IPv6 Fejléc



# Különbségek az IPv4-hez képest

18

- Számos mező hiányzik az IPv6 fejlécből
  - Fejléc hossza – beépült a Next Header mezőbe
  - Checksum – nem igazán használták már korábban se...
  - Identifier, Flags, Offset
    - IPv6 routerek nem támogatják a fragmentációt
    - Az állomások MTU felderítést alkalmaznak
- Az Internet felhasználás súlypontjainak megváltozása
  - Napjaink hálózatai sokkal homogénebbek, mint

# Teljesítmény növekmény

19

- Nincsenek ellenőrizendő kontrollösszegek (checksum)
- Nem szükséges a fragmentáció kezelése a routerekben
- Egyszerű routing tábla szerkezet
  - A cím tér nagy
  - Nincs szükség CIDR-re (de aggregáció szükséges)
  - A szabványos alhálózat méret 264 cím
- Egyszerű auto-konfiguráció

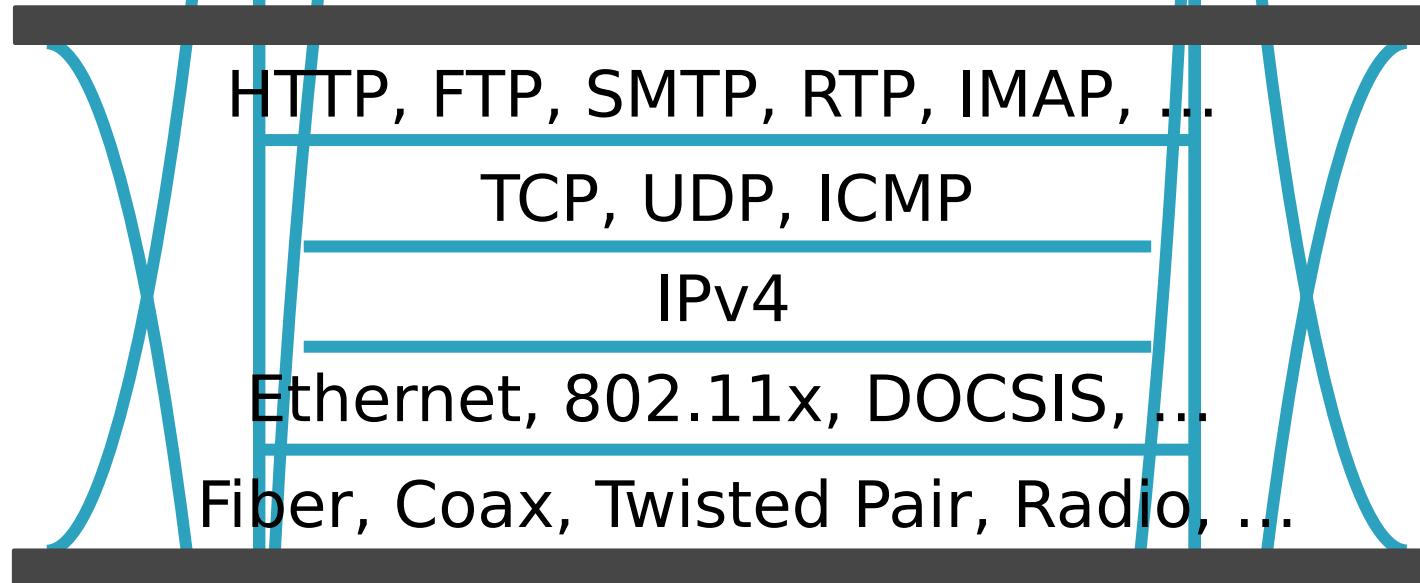
# További IPv6 lehetőségek

20

- Forrás Routing
  - Az állomás meghatározhatja azt az útvonalat, amelyen a csomagjait továbbítani szeretné
- Mobil IP
  - Az állomások magukkal vihetik az IP címüket más hálózatokba
  - Forrás routing használata a csomagok irányításához
- Privacy kiterjesztések
  - Véletlenszerűen generált állomás azonosítók

# Bevezetési nehézségek

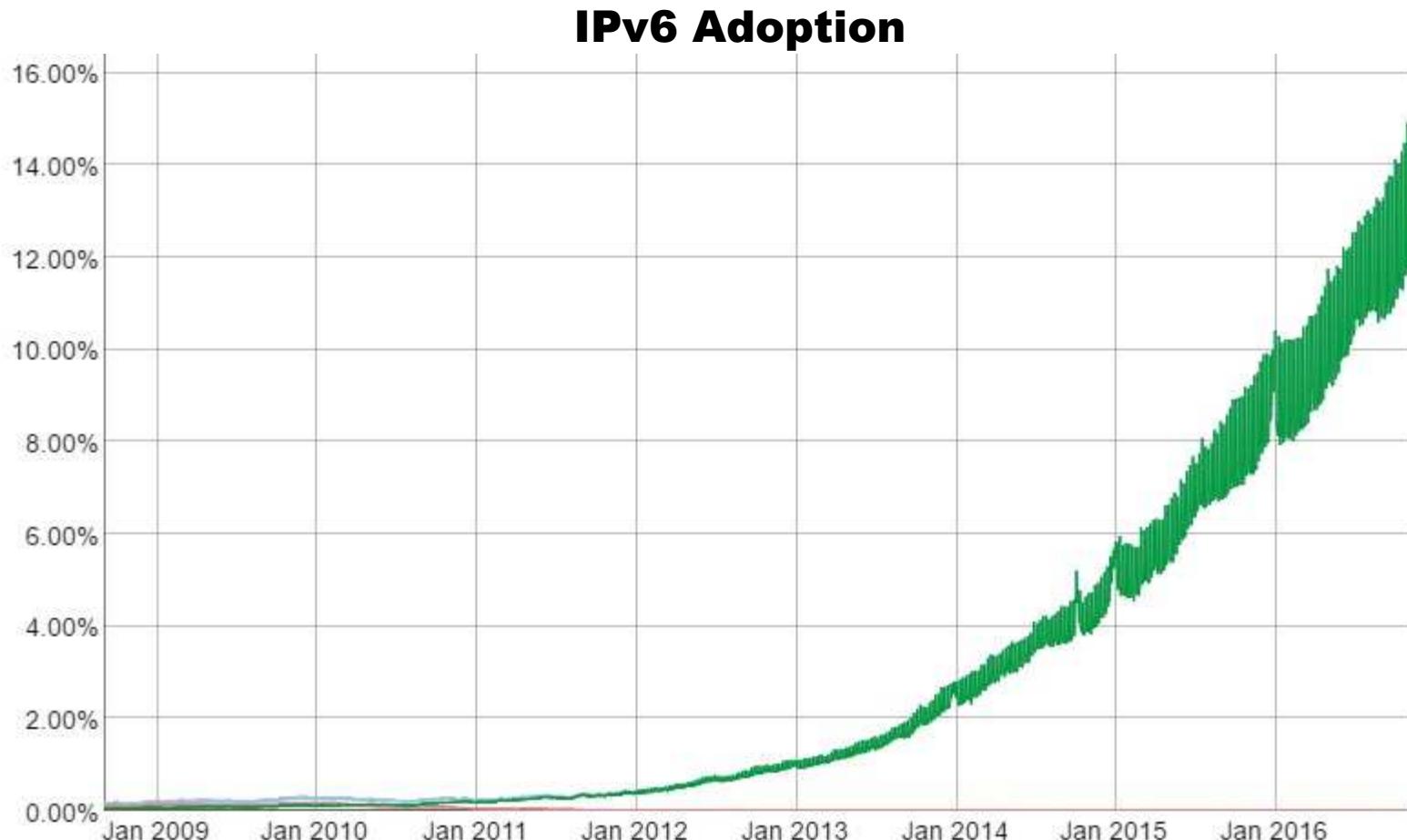
21



- IPv6 bevezetése a teljes Internet frissítését jelentené
  - minden router, minden hoszt
  - ICMPv6, DHCPv6, DNSv6

<https://www.google.com/intl/en/ipv6/statistics.html>

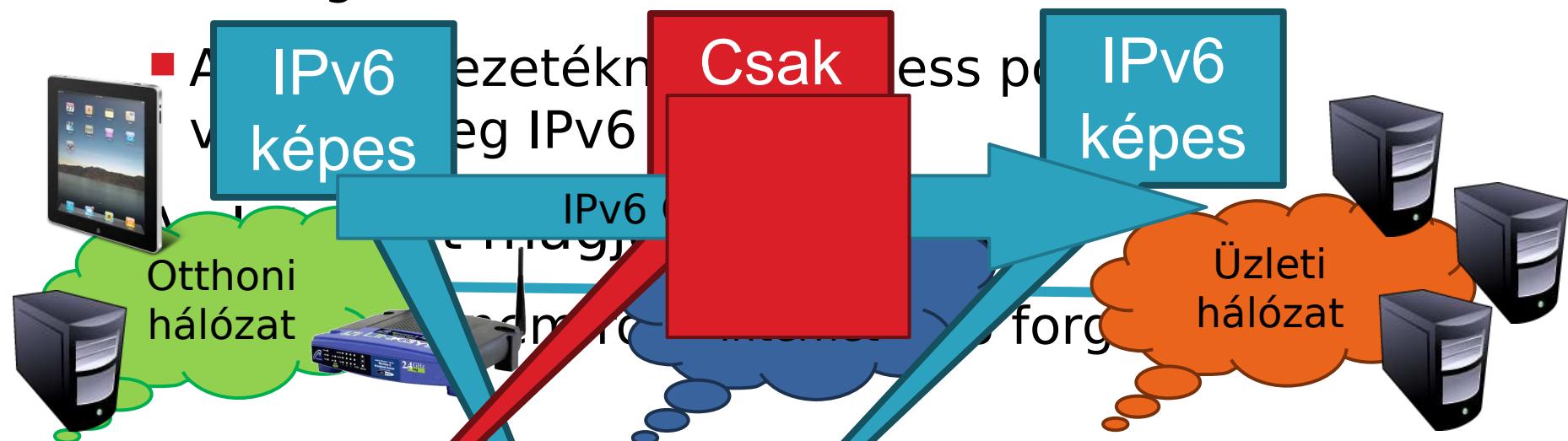
22



# Átmenet IPv6-ra

23

- Hogyan történhet az átmenet IPv4-ről IPv6-ra?
  - Napjainkban a legtöbb végpont a hálózat széleken támogatja az IPv6-ot
    - Windows/OSX/iOS/Android minden tartalmaz IPv6 támogatást
    - A vezetéknél leggyakrabban használják az IPv6-t



# Átmeneti megoldások

24

- Azaz hogyan routoljunk IPv6 forgalmaz IPv4 hálózat felett?
- Megoldás
  - Használunk tunneleket az IPv6 csomagok becsomagolására és IPv4 hálózaton való továbbítására
  - Számos különböző implementáció
    - 6to4
    - IPv6 Rapid Deployment (6rd)
    - Teredo

Routing 2. felvonás

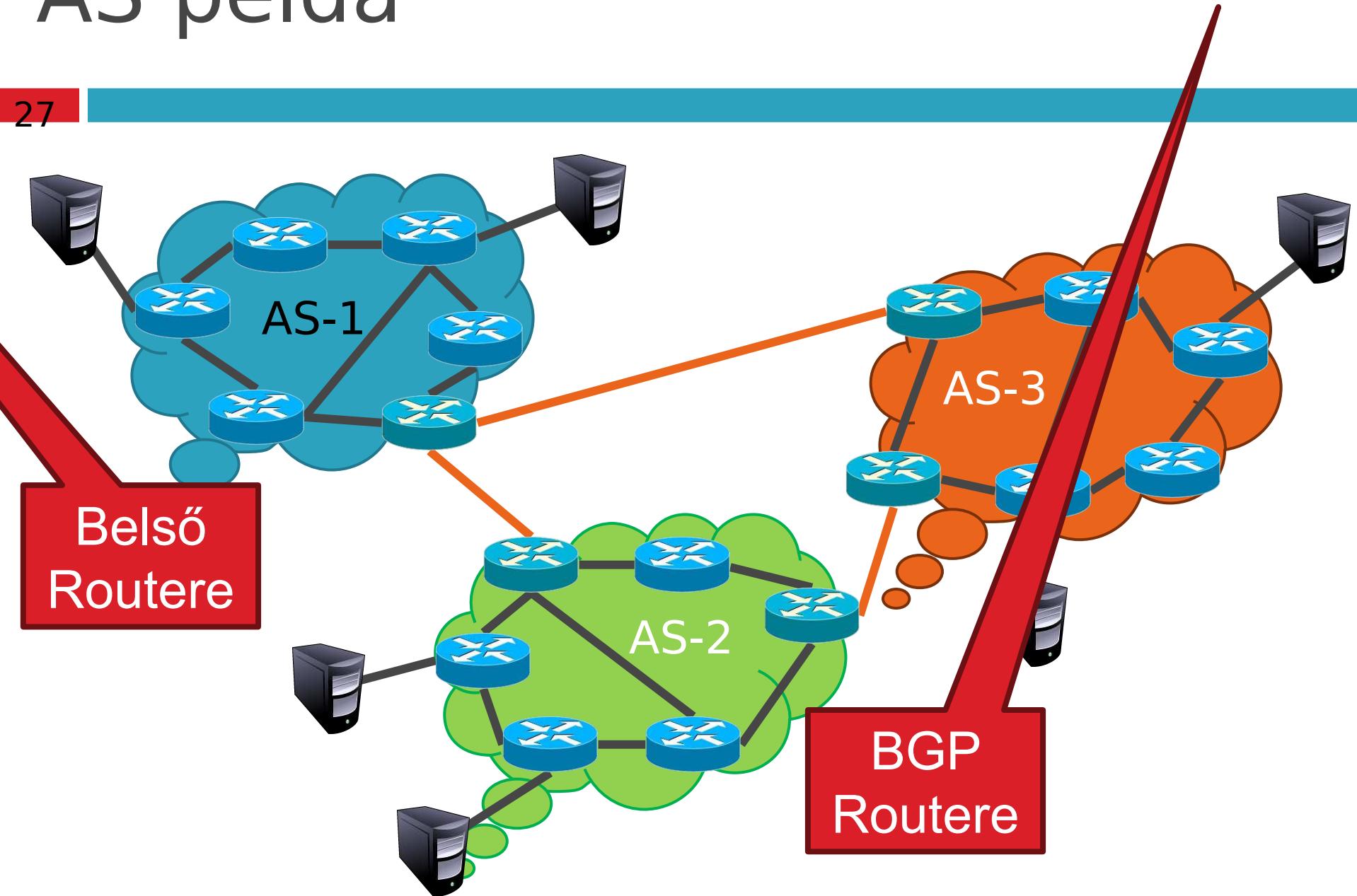
# Újra: Internet forgalom irányítás

26

- Az Internet egy két szintű hierarchiába van szervezve
- Első szint – autonóm rendszerek (AS-ek)
  - AS – egy adminisztratív tartomány alatti hálózat
  - Pl.: ELTE, Comcast, AT&T, Verizon, Sprint, ...
- AS-en belül ún. intra-domain routing protokollokat használunk
  - Distance Vector, pl.: Routing Information Protocol (RIP)
  - Link State, pl.: Open Shortest Path First (OSPF)

# AS példa

27



# Miért van szükség AS-ekre?

28

- A routing algoritmusok **nem elég hatékonyak** ahhoz, hogy a teljes Internet topológián működjenek
- Különböző szervezetek **más-más politika** mentén akarnak forgalom irányítást (policy)
- Lehetőség, hogy a szervezetek **elrejtsék a belső hálózatuk szerkezetét**
- Lehetőség, hogy a szervezetek **elrejtsék, hogy a belső hálózatuk szerkezetét**,
  - Egyszerűbb az útvonalak forgalma számítása

# AS számok

29

- minden AS-t egy AS szám (ASN) azonosít
  - 16 bites érték (a legújabb protokollok már 32 bites azonosítókat is támogatnak)
  - 64512 – 65535 más célra foglalt
- Jelenleg kb. 40000 AS szám létezik
  - AT&T: 5074, 6341, 7018, ...
  - Sprint: 1239, 1240, 6211, 6242, ...
  - ELTE: 2012
  - Google 15169, 36561 (formerly YT), + others
  - Facebook 22224

# Inter-Domain Routing

30

- A globális összeköttetéshez szükséges!!!
  - Azaz minden AS-nek **ugyanazt** a protokollt kell használnia
  - Szemben az intra-domain routing-gal
- Milyen követelmények vannak?
  - Skálázódás
  - Rugalmas útvonal választás
    - Költség
    - Forgalom irányítás egy hiba kikerülésére
- Milyen protokollt válasszunk?

# Border Gateway Protocol

31

## Általános

AS-ek közötti (*exterior gateway protocol*).

Eltérő célok vannak forgalomirányítási szempontból, mint az AS-eken belüli protokollnál.

Politikai szempontok szerepet játszhatnak a forgalomirányítási döntésben.

## Néhány példa forgalomirányítási korlátozásra

- Ne legyen átmenő forgalom bizonyos AS-eken keresztül.
- Csak akkor haladjunk át Albánián, ha nincs más út a célhoz.
- Az IBM-nél kezdődő illetve végződő forgalom ne menjen át a Microsoft-on.
- A politikai jellegű szabályokat kézzel konfigurálják a BGP-routeren.
- A BGP router szempontjából a világ AS-ekből és a közöttük

# Border Gateway Protocol

32

## Hálózatok csoportosítása az átmenő forgalom szempontjából

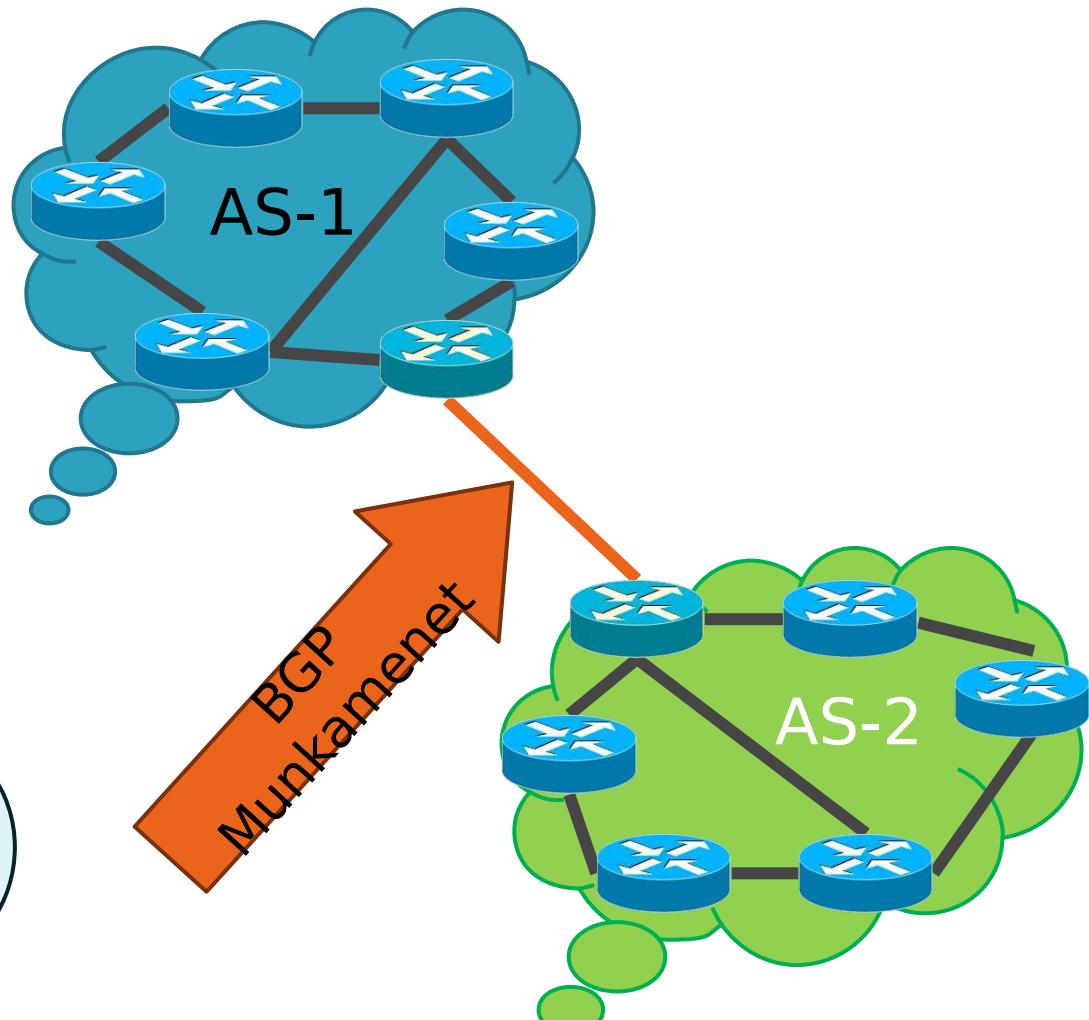
1. **Csonka hálózatok**, amelyeknek csak egyetlen összeköttetésük van a BGP gráffal.
2. **Többszörösen bekötött hálózatok**, amelyeket használhatna az átmenő forgalom, de ezek ezt megtagadják.
3. **Tranzit hálózatok**, amelyek némi megkötéssel, illetve általában fizetség ellenében, készek kezelni harmadik fél csomagjait.

## Jellemzők

- A BGP router-ek páronként TCP-összeköttetést létrehozva kommunikálnak egymással.

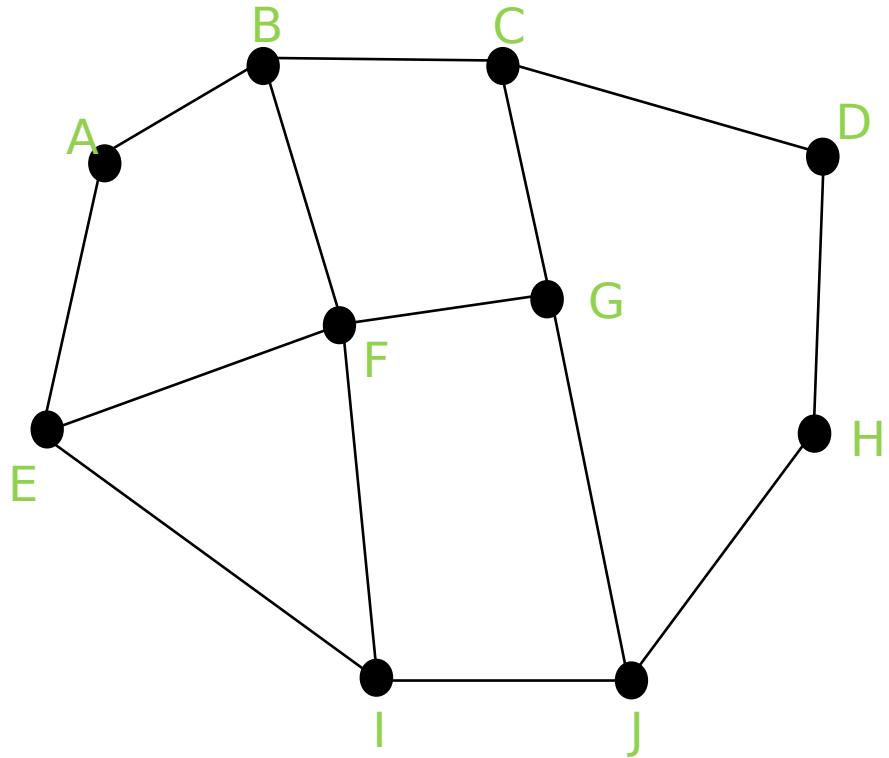
# BGP egyszerűsített működése

33



# Border Gateway Protocol

34



A *F* által a szomszédjaitól kapott *D*-re vonatkozó információ az alábbi:

*B*-től: „Én a *BCD*-t használom”

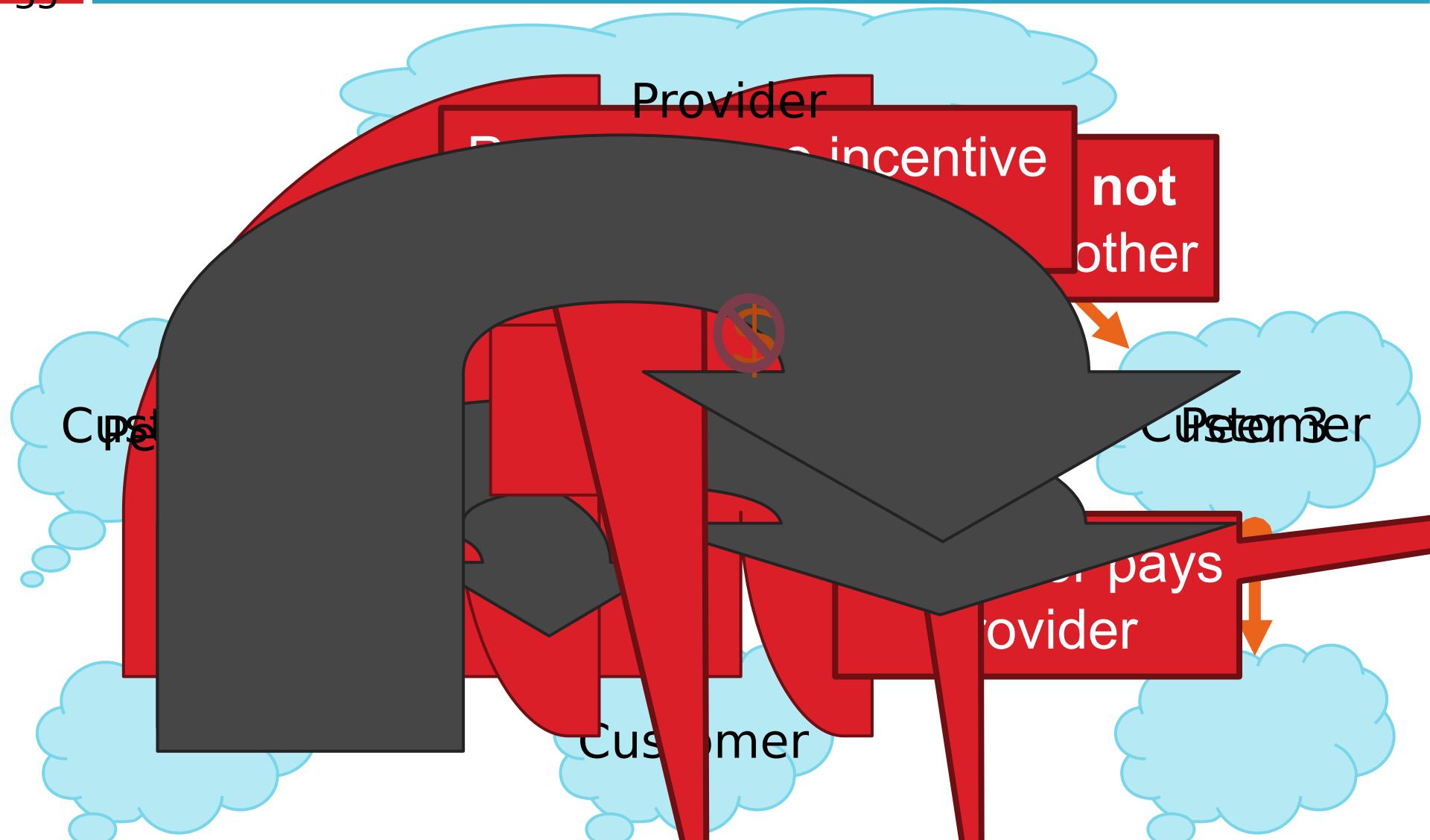
*G*-től: „Én a *GCD*-t használom”

*I*-től: „Én a *IFGCD*-t használom”

*E*-től: „Én a *EFGCD*-t használom”

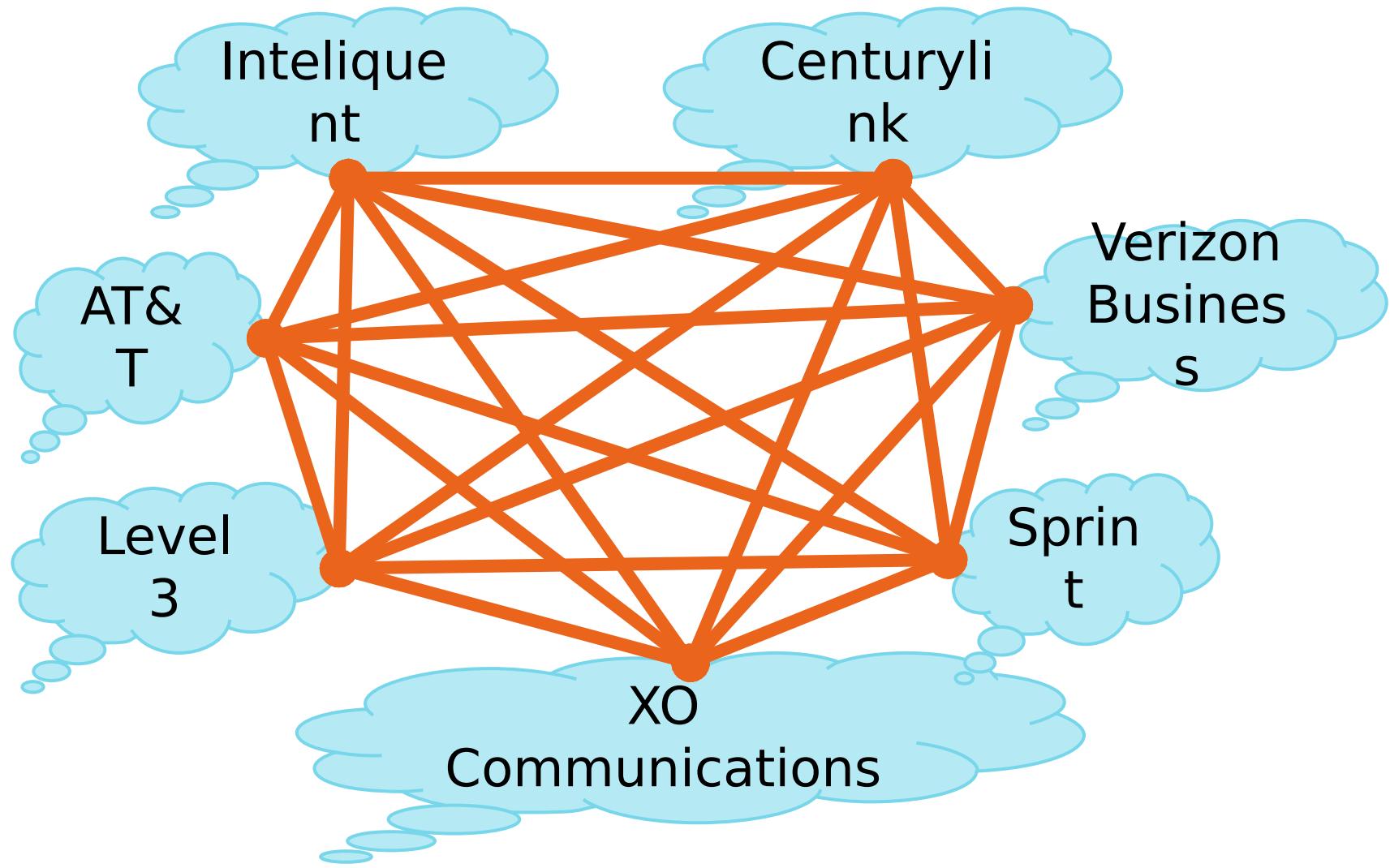
# BGP kapcsolatok

35



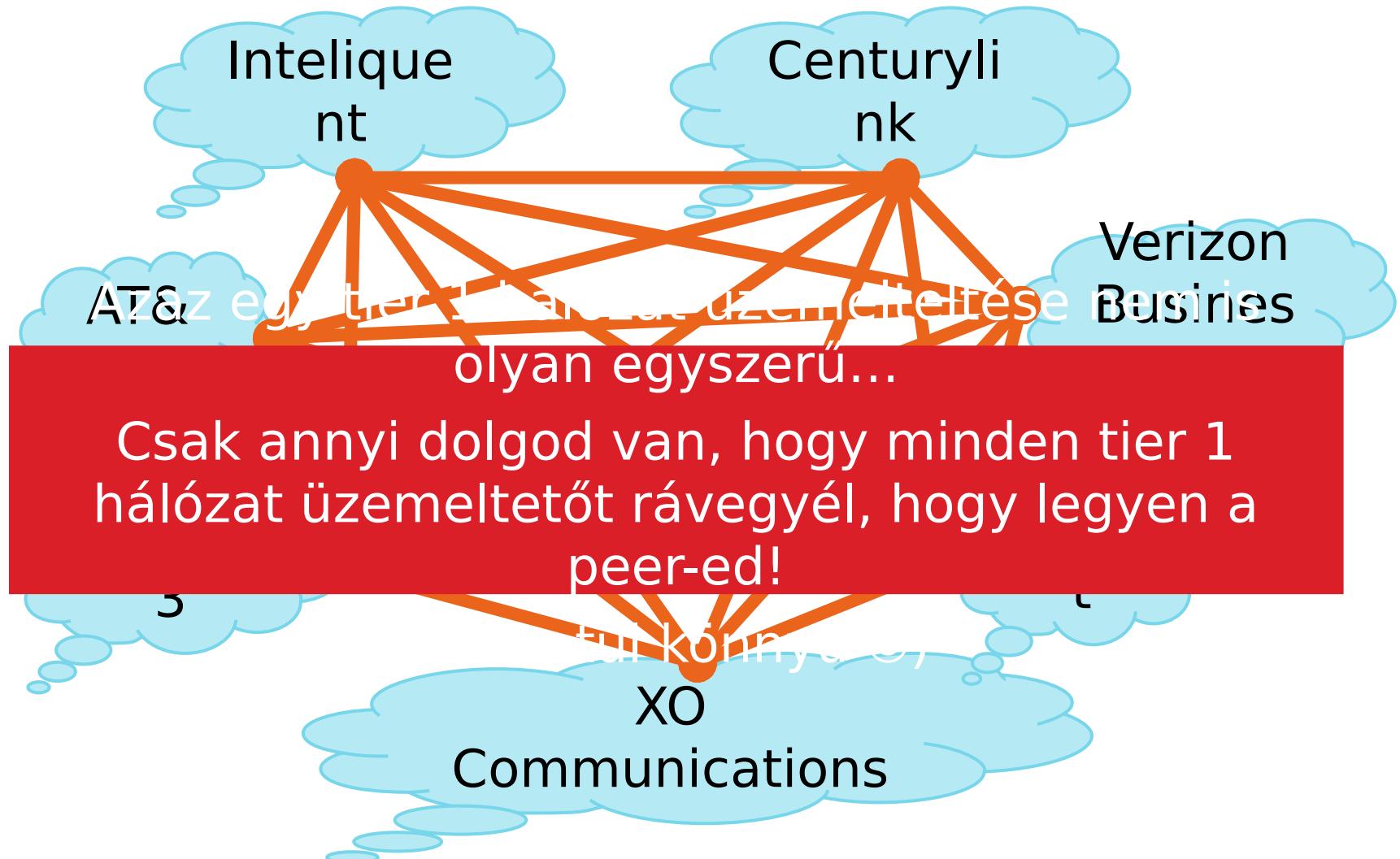
# Tier-1 ISP Peering

36



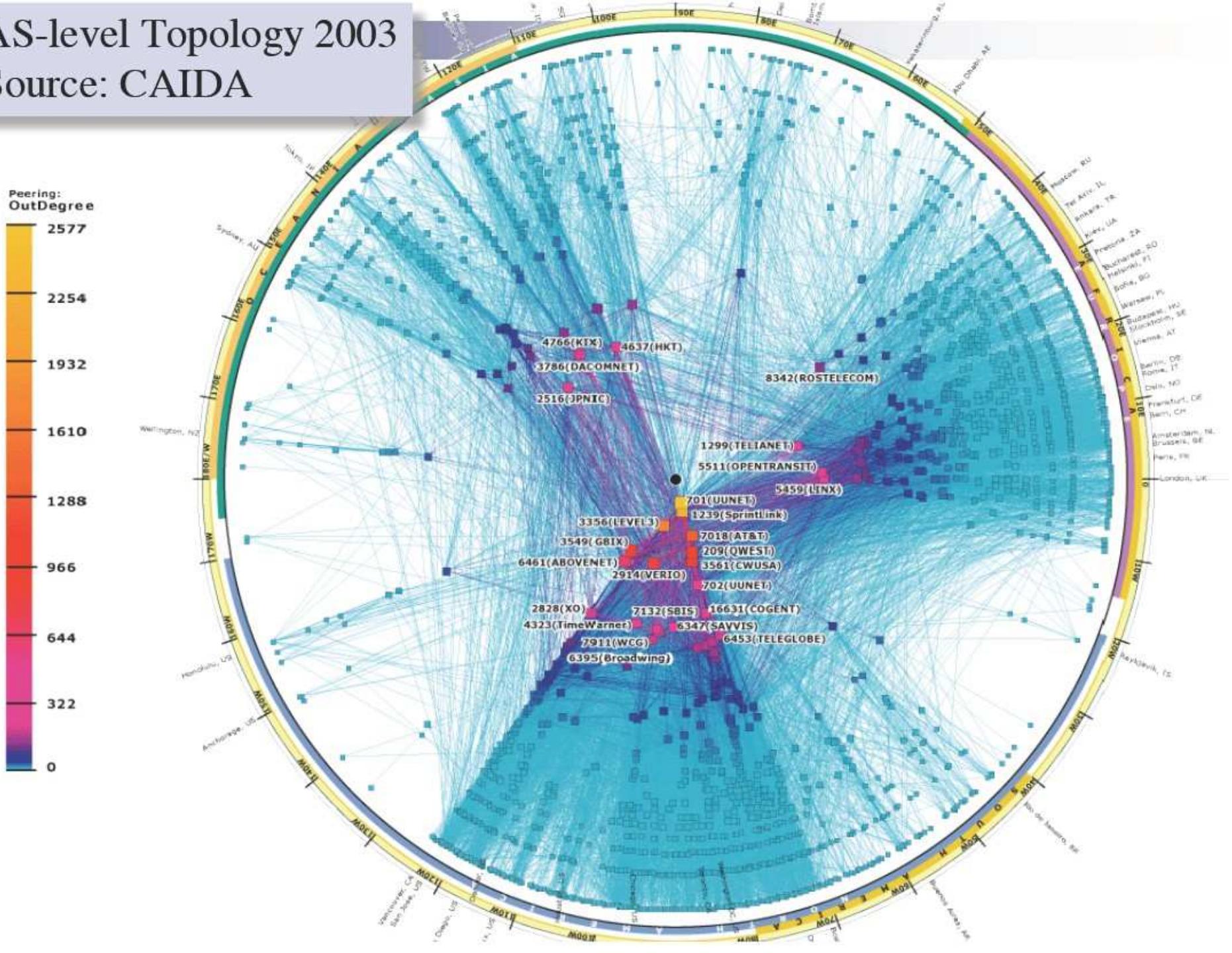
# Tier-1 ISP Peering

37



# AS-level Topology 2003

Source: CAIDA

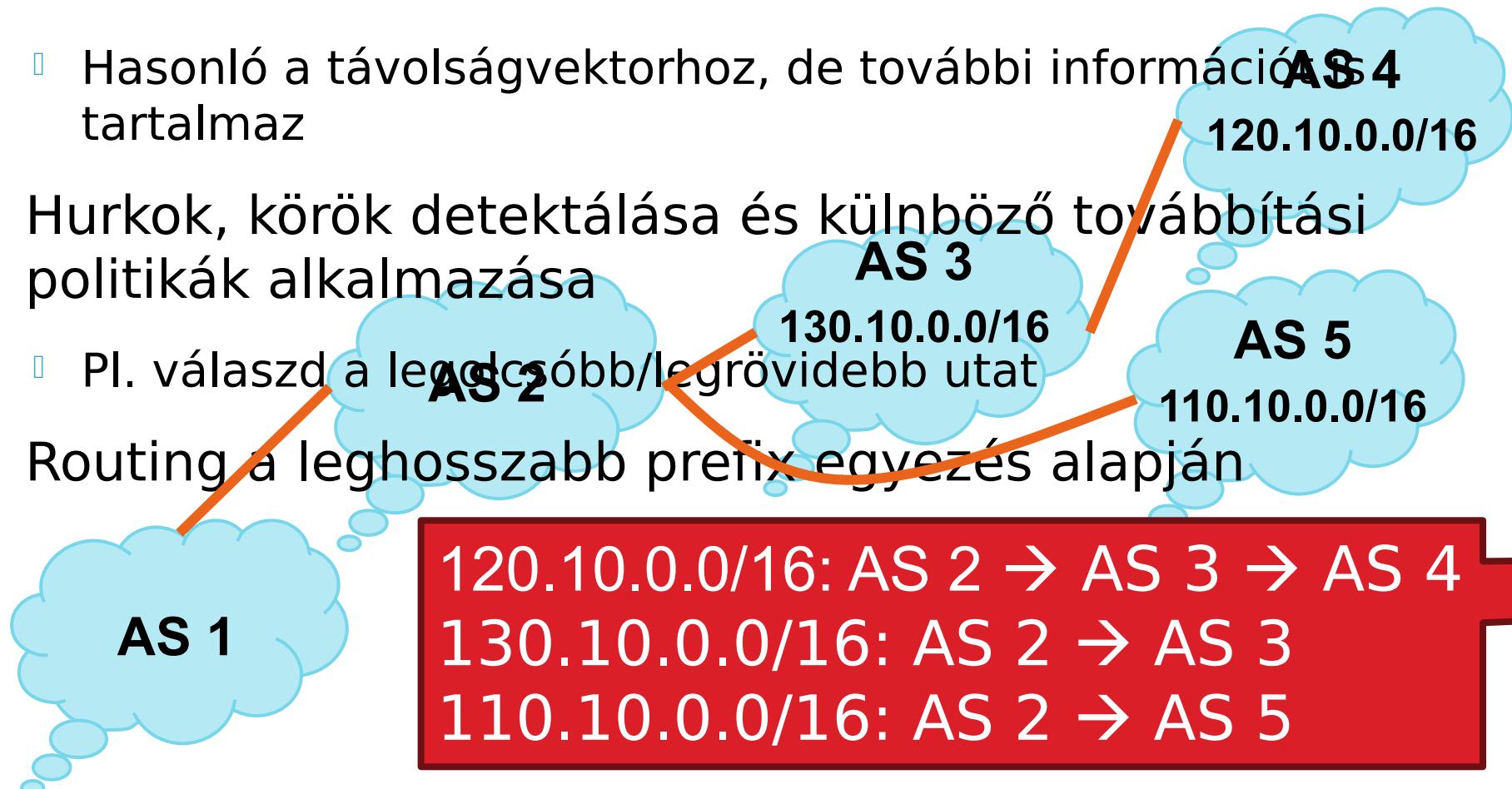


# Útvonalvektor protokoll

## Path Vector Protocol

39

- AS-útvonal: AS-ek sorozata melyeken áthalad az útvonal
  - Hasonló a távolságvektorhoz, de további információ tartalmaz
- Hurkok, körök detektálása és különböző továbbítási politikák alkalmazása
  - Pl. válaszd a legolcsóbb/legrövidebb utat
- Routing a leghosszabb prefix egyezés alapján



# Útvonalvektor protokoll

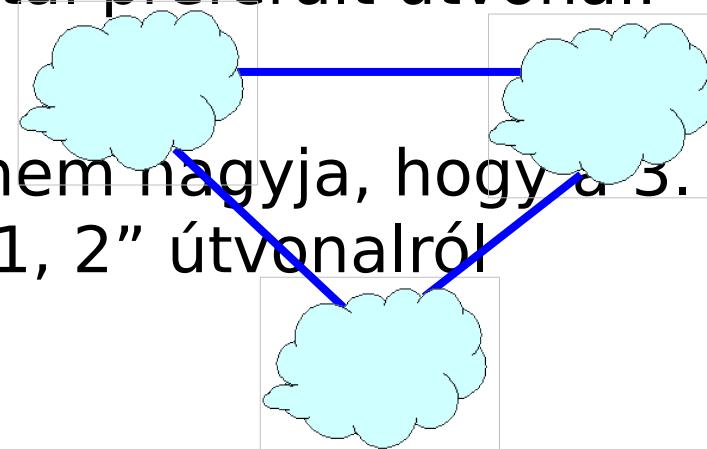
## Path Vector Protocol

- A távolságvektor protokoll kiterjesztése
  - Rugalmas továbbítási politikák
  - Megoldja a végtelenig számolás problémáját
  - Útvonalvektor: Célállomás, következő ugrás (nh), AS útvonal
- Ötlet: a teljes útvonalat meghirdeti



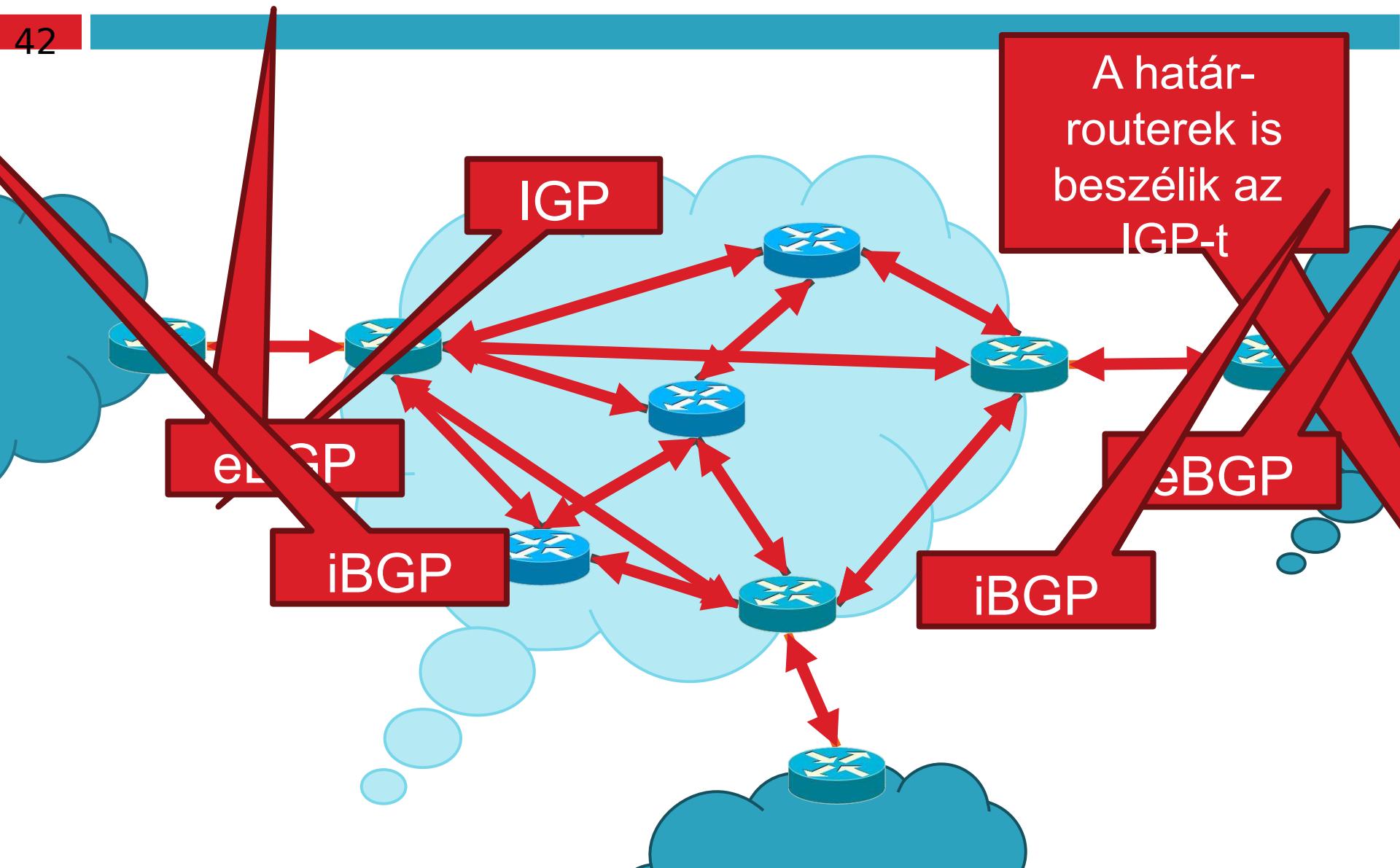
# Rugalmas forgalomirányítás

- minden állomás hely/saját útválasztási politikát alkalmaz
  - Útvonal kiválasztás: Melyik útvonalat használjuk?
  - Útvonal export: Melyik útvonalat hirdessük meg?
- Példák
  - A 2. állomás által preferált útvonal: "2, 3, 1" (nem a "2, 1")
  - Az 1. állomás nem hagyja, hogy a 3. állomás értesüljön az "1, 2" útvonalról



# BGP

42

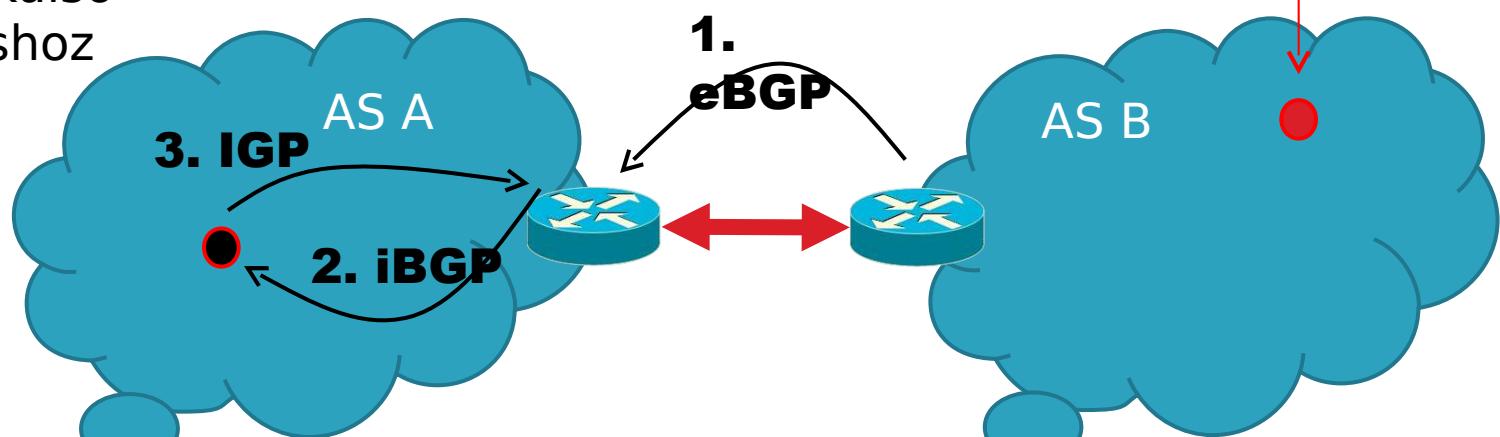


# IGB – iBGP – eBGP

43

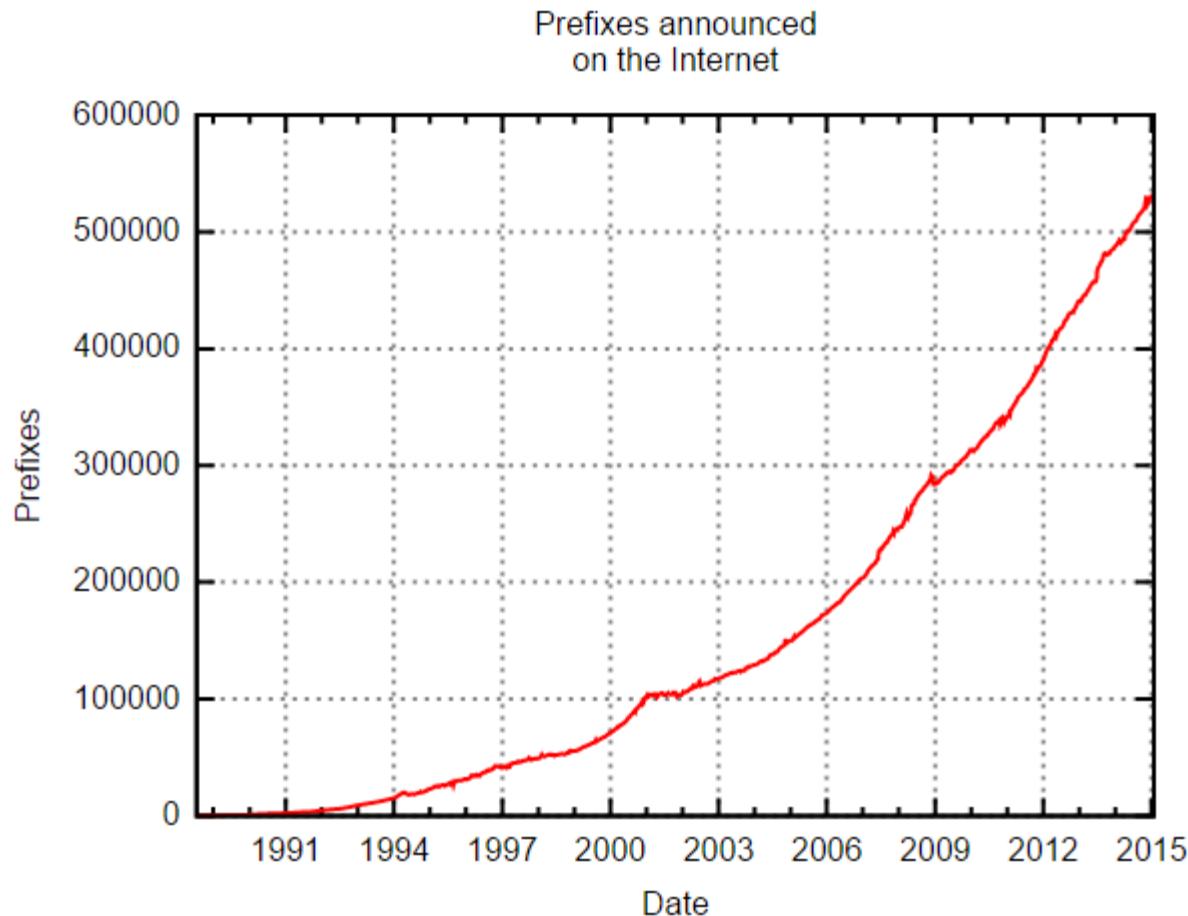
- eBGP: Routing információk cseréje autonóm rendszerek között
- IGP: útválasztás egy AS-en belül belső célállomáshoz
- iBGP: útválasztás egy AS-en belül egy külső célállomáshoz

- 1. eBGP – A megismeri az útvonal a célohoz, ehhez eBGP-t használunk
- 2. iBGP – A-ban levő router megtanulja a célohoz vezető utat az iBGP segítségével (a köv. ugrás a határ router)
- 3. IGP – IGP segítségével eljuttatja a csomagot az **Cél állomás**



# Forrás: wikipedia

44



További protokollok

# Internet Control Message Protocol

46

## Feladata

- Váratlan események jelentése

## Használat

- Többfélé *ICMP*-üzenetet definiáltak:

- Elérhetetlen cél;
- Időtúllépés;
- Paraméter probléma;
- Forráslefojtás;
- Visszhang kérés;
- Visszhang válasz;
- ...

# Internet Control Message Protocol

47

- *Elérhetetlen cél* esetén a csomag kézbesítése sikertelen volt.
  - **Esemény lehetséges oka:** Egy nem darabolható csomag továbbításának útvonalán egy „kis csomagos hálózat” van.
- *Időtúllépés* esetén az IP csomag élettartam mezője elérte a 0-át.
  - **Esemény lehetséges oka:** Torlódás miatt hurok alakult ki vagy a számláló értéke túl alacsony volt.
- *Paraméter probléma* esetén a fejrészben érvénytelen mezőt észleltünk.

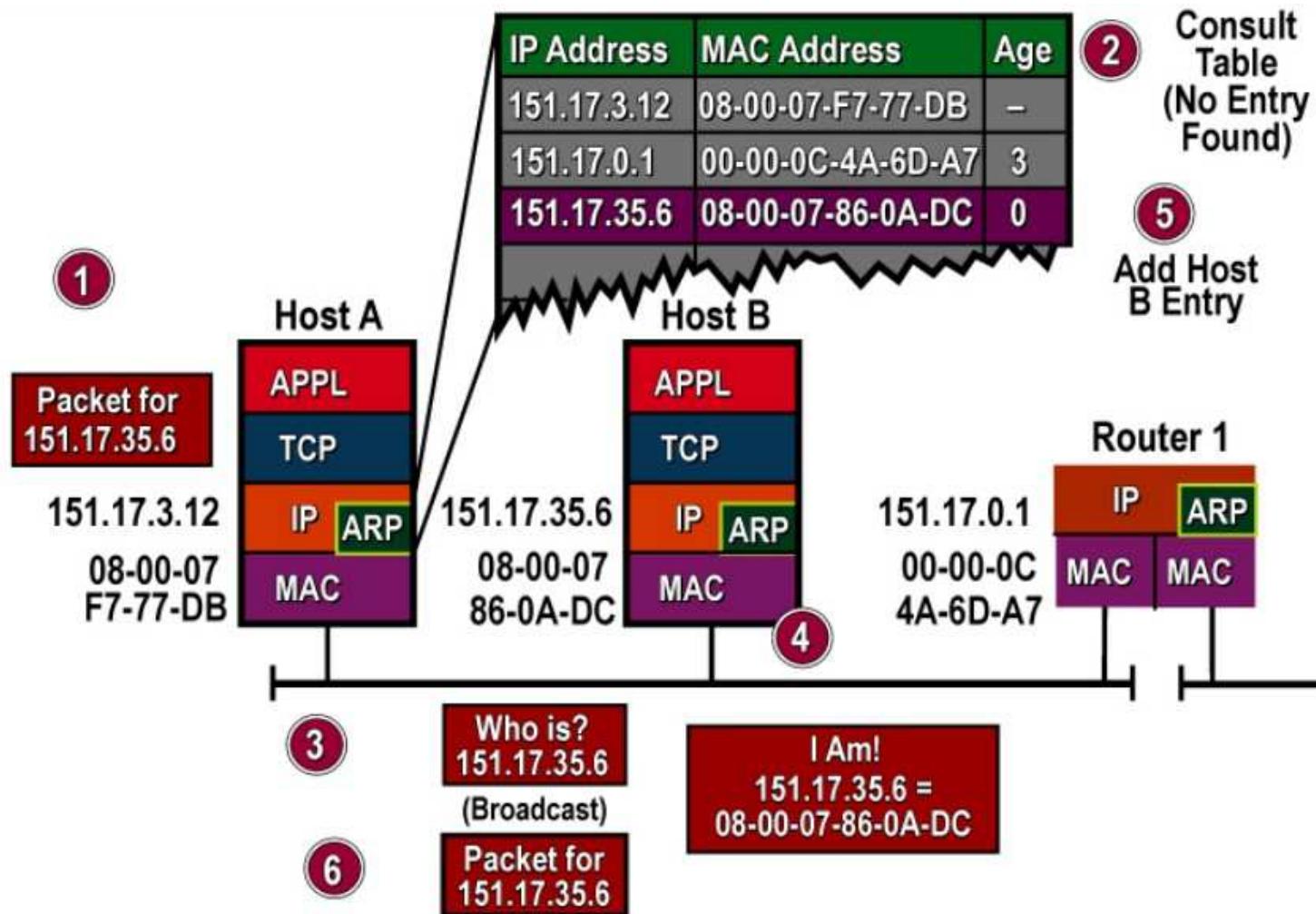
# Internet Control Message Protocol

48

- Forráslefojtás esetén lefojtó csomagot küldünk.
  - **Esemény hatása:** A fogadó állomásnak a forgalmazását lassítania kellett.
- Visszhang kérés esetén egy hálózati állomás jelenlétét lehet ellenőrizni.
  - **Esemény hatása:** A fogadónak vissza kell küldeni egy visszhang választ.
- Átirányítás esetén a csomag rosszul irányítottságát jelzik.
  - **Esemény kiváltó oka:** Router észleli, hogy a csomag nem az optimális útvonall.

# Address Resolution Protocol

49



# Address Resolution Protocol

50

## Feladata

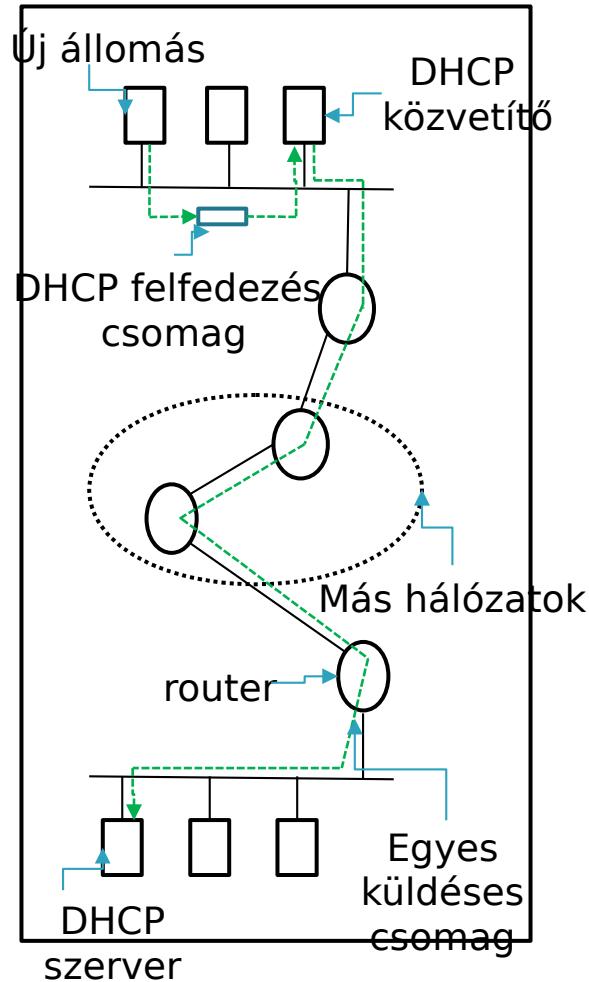
- Az IP cím megfeleltetése egy fizikai címnek.

## Hozzárendelés

- Adatszóró csomag kiküldése az *Ethernetre* „Ki-é a 192.60.34.12-es IP-cím?” kérdéssel az alhálózaton, és mindenegyes hoszt ellenőrzi, hogy övé-e a kérdéses IP-cím. Ha egyezik az IP a hoszt saját IP-jével, akkor a saját *Ethernet* címével válaszol. Erre szolgál az ARP.
- Opcionális javítási lehetőségek:
  - a fizikai cím IP hozzárendelések tárolása (*cache használata*);
  - Leképezések megváltoztathatósága (*időhatály bevezetése*);
- Mi történik távoli hálózaton lévő hoszt esetén?
  - A router is válaszoljon az ARP-re a hoszt alhálózatán. (*proxy*)

# Reverse Address Resolution Protocol

51



# Reverse Address Resolution Protocol

## Feladata

- A fizikai cím megfeleltetése egy IP címnek

## Hozzárendelés

- Az újonnan indított állomás adatszórással csomagot küld ki az *Ethernetre* „A 48-bites Ethernet-címem 14.04.05.18.01.25. Tudja valaki az IP címemet?” kérdéssel az alhálózaton. Az *RARP*-szerver pedig válaszol a megfelelő IP címmel, mikor meglátja a kérést
- Opcionális javítási lehetőségek:

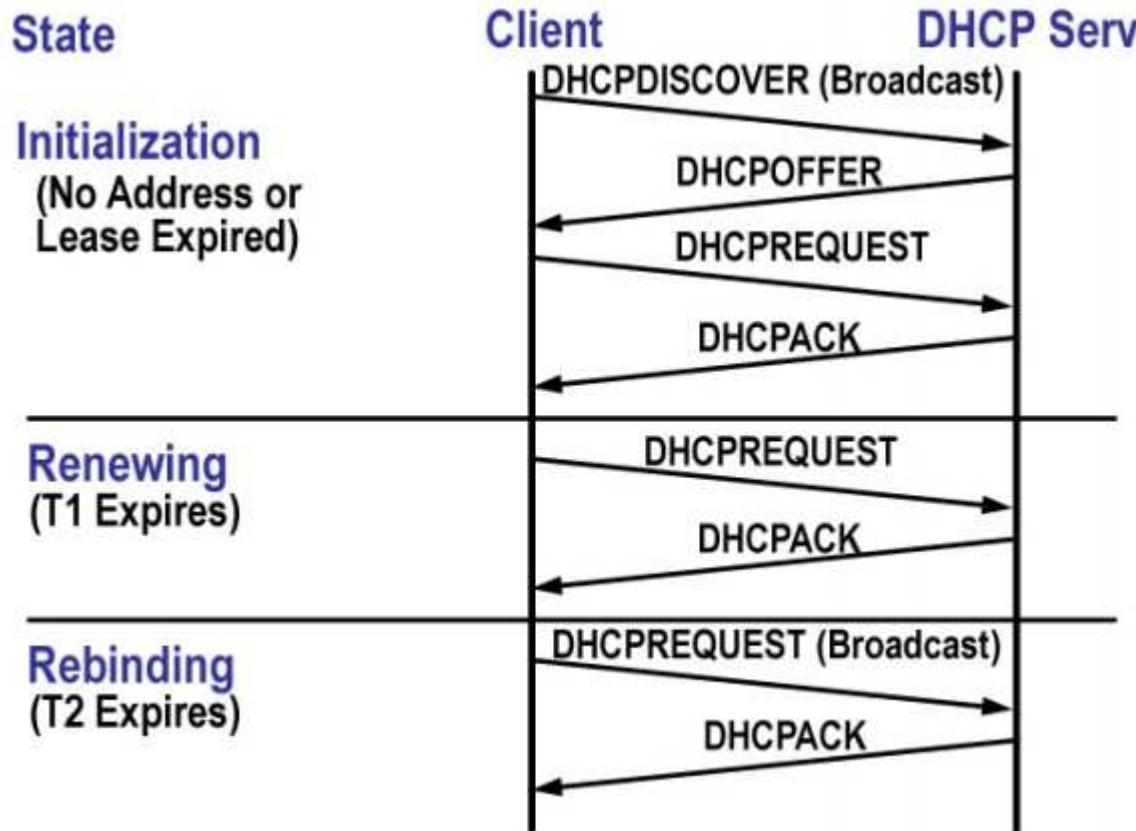
- *BOOTP* protokoll használata. UDP csomagok használata. Manuálisan kell a hozzárendelési táblázatot karbantartani. (statikus címkiosztás)

- *DHCP* protokoll használata. Itt is külön kiszolgáló osztja ki

# DHCP: DYNAMIC HOST CONFIGURATION

53

## PROTOCOL



# DHCP

54

- Lényegében ez már az **Alkalmazási réteg**
  - de logikailag ide tartozik
- Segítségével a hosztok automatikusan juthatnak hozzá a kommunikációjukhoz szükséges hálózati azonosítóhoz:
  - IP cím, hálózati maszk, alapértelmezett átjáró, stb.
- Eredetileg az RFC 1531 a BOOTP

# DHCP lehetőségei

55

- IP címek osztása MAC cím alapján DHCP szerverrel
  - Szükség esetén (a DHCP szerveren előre beállított módon) egyes kliensek számára azok MAC címéhez fix IP cím rendelhető
- IP címek osztása dinamikusan
  - A DHCP szerveren beállított tartományból „érkezési sorrendben” kapják a kliensek az IP címeket
  - Elegendő annyi IP cím, ahány gép egyidejűleg működik

# DHCP – Címek bérlese

56

- A DHCP szerver a klienseknek az IP-címeket bizonyos bérleti időtartamra (lease time) adja „bérbe”
  - Az időtartam hosszánál a szerver figyelembe veszi a kliens esetleges ilyen irányú kérését
  - Az időtartam hosszát a szerver beállításai korlátozzák
- A bérleti időtartam lejárta előtt a bérlet meghosszabbítható
- Az IP-cím explicit módon vissza is adható

# Virtuális magánhálózatok alapok

## □ Fő jellemzői

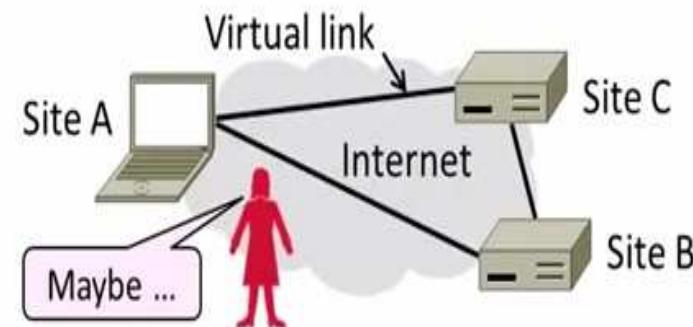
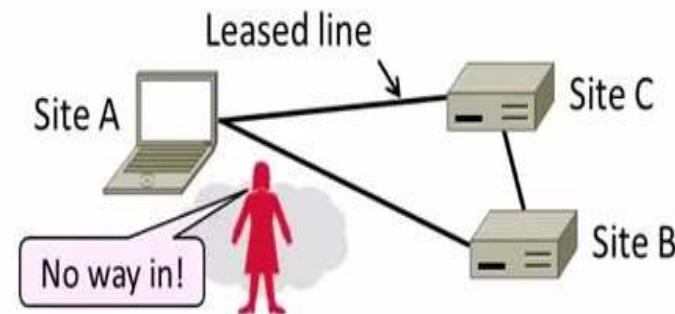
- Mint közeli hálózat fut az interneten keresztül.
- IPSEC-et használ az üzenetek titkosítására.

▫ Azaz informálisan megfogalmazva fizikailag távol lévő hosztok egy közös logikai egységet alkotnak.

- Például távollévő telephelyek rendszerei.

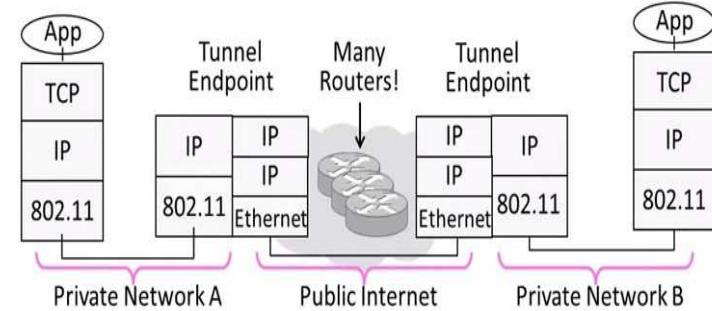
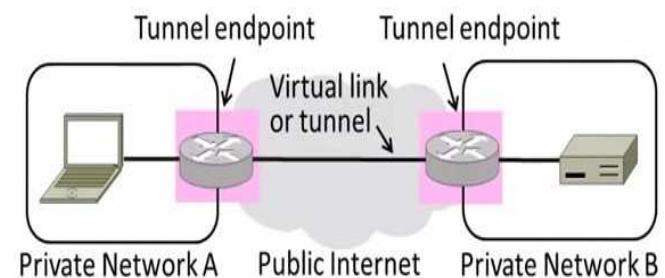
## □ Alapelv

- Bérelt vonalak helyett használjuk a publikusan hozzáférhető "Leased line"-t.



# Virtuális magánhálózatok alapok

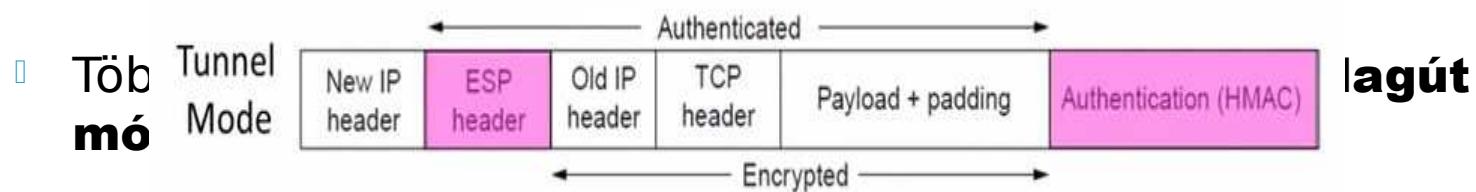
- A virtuális linkekkel alagutak képzésével valósítjuk meg.
- **Alagútak**
  - Egy magánhálózaton belül a hosztok egymásnak normál módon küldhetnek üzenetet.
  - Virtuális linken a végpontok beágyazzák a csomagokat.
    - IP az IP-be mechanizmus.
- Az alagutak képzése önmagában kevés a védelemhez. Mik a hiányosságok?
  - Bizalmasság, authentikáció
  - Egy támadó olvashat, küldhet



# Virtuális magánhálózatok alapok

## □ IPSEC

- Hosszú távú célja az IP réteg biztonságossá tétele.  
(bizalmasság, autentikáció)
- Műveletei:
  - Hoszt párok kommunikációjához kulcsokat állít be.
  - A kommunikáció kapcsolatorientáltabbá tétele.
  - Fejlécek és láblécek hozzáadása az IP csomagok védelme érdekében



# Számítógépes Hálózatok

**9. Előadás: VPN + Szállítói  
réteg**

Based on slides from

# Virtuális magánhálózatok alapok

## □ Fő jellemzői

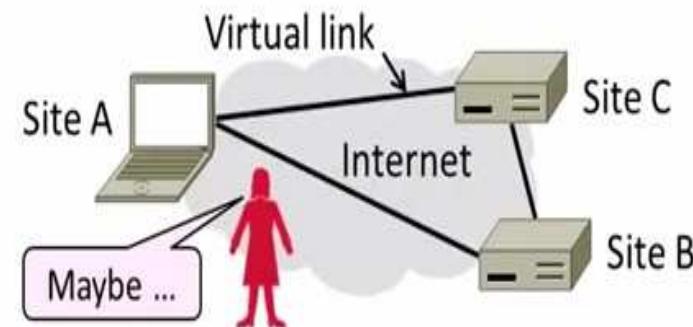
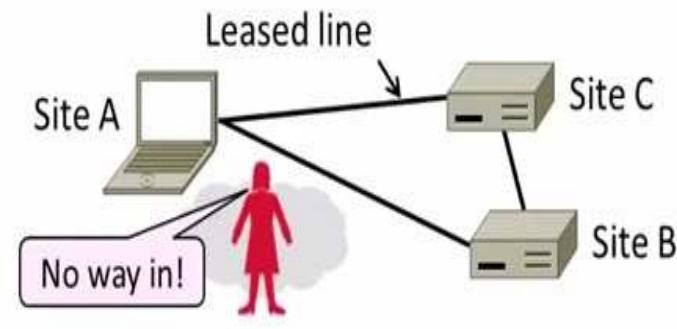
- Mint közeli hálózat fut az interneten keresztül.
- IPSEC-et használ az üzenetek titkosítására.

□ Azaz informálisan megfogalmazva fizikailag távol lévő hosztok egy közös logikai egységet alkotnak.

- Például távollévő telephelyek rendszerei.

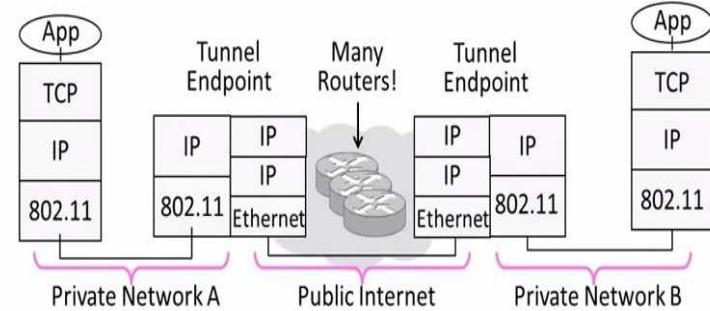
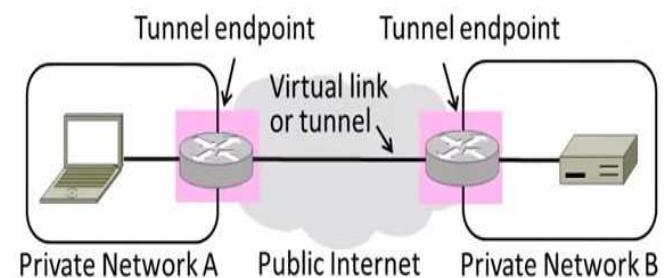
## □ Alapelv

- Bérelt vonalak helyett használjuk a publikusan hozzáférhető "Leased line"-t.



# Virtuális magánhálózatok alapok

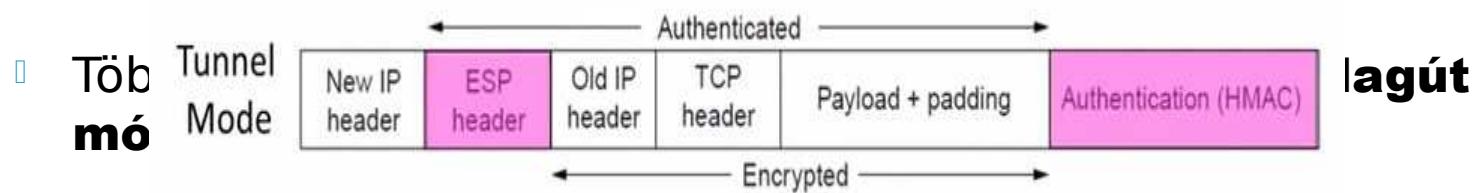
- A virtuális linkekkel alagutak képzésével valósítjuk meg.
- **Alagútak**
  - Egy magánhálózaton belül a hosztok egymásnak normál módon küldhetnek üzenetet.
  - Virtuális linken a végpontok beágyazzák a csomagokat.
    - IP az IP-be mechanizmus.
- Az alagutak képzése önmagában kevés a védelemhez. Mik a hiányosságok?
  - Bizalmasság, authentikáció
  - Egy támadó olvashat, küldhet



# Virtuális magánhálózatok alapok

## □ IPSEC

- Hosszú távú célja az IP réteg biztonságossá tétele.  
(bizalmasság, autentikáció)
- Műveletei:
  - Hoszt párok kommunikációjához kulcsokat állít be.
  - A kommunikáció kapcsolatorientáltabbá tétele.
  - Fejlécek és láblécek hozzáadása az IP csomagok védelme érdekében



# Szállítói réteg

5



- Feladat:
  - Adatfolyamok demultiplexálása
- További lehetséges feladatok:
  - Hosszú élettartamú kapcsolatok
  - Megbízható, sorrendhelyes csomag leszállítás
  - Hiba detektálás

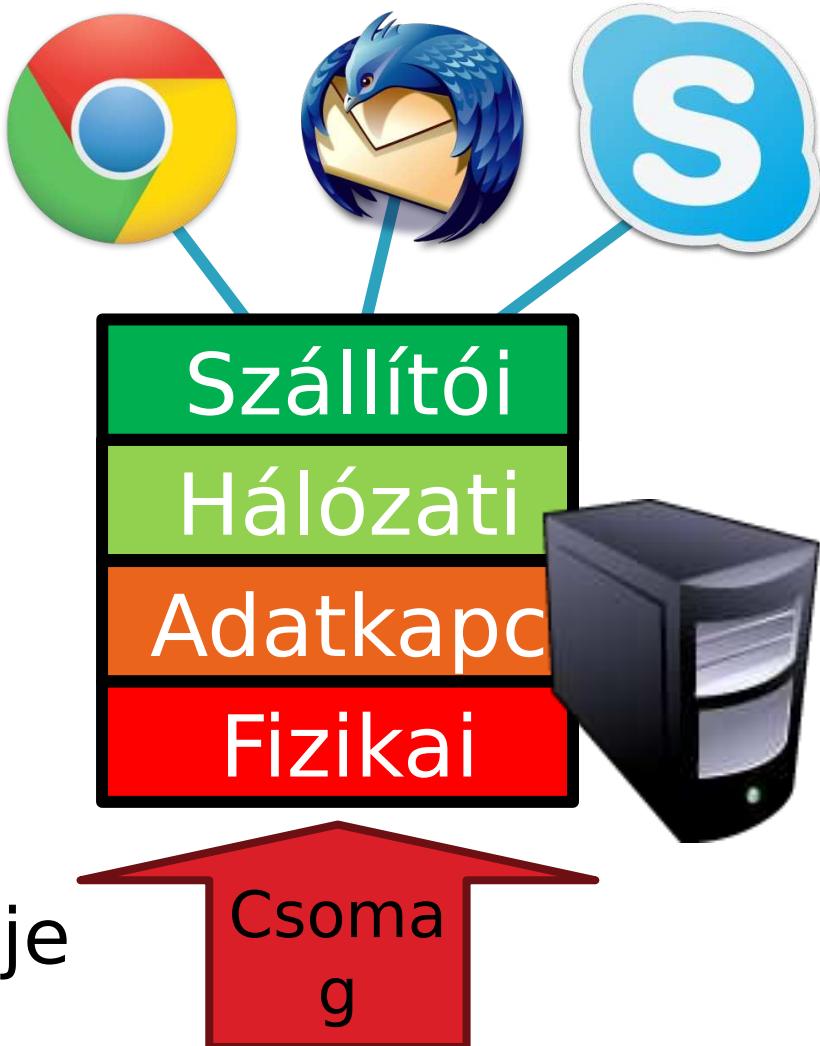
# Kivonat

- ❑ UDP
- ❑ TCP
- ❑ Torlódás vezérlés
- ❑ TCP evolúciója
- ❑ A TCP problémái

# Multiplexálás

7

- Datagram hálózat
  - Nincs áramkör kapcsolás
  - Nincs kapcsolat
- A kliensek számos alkalmazást futtathatnak egyidőben
  - Kinek szállítsuk le a csomagot?
- IP fejléc “protokoll” mezője
  - 8 bit = 256 konkurens



# Forgalom demultiplexálása

8

A szerver alkalmazások számos klienssel

Alkalmazás kommunikálhat

Szállítói

P1

P2

Hálózati

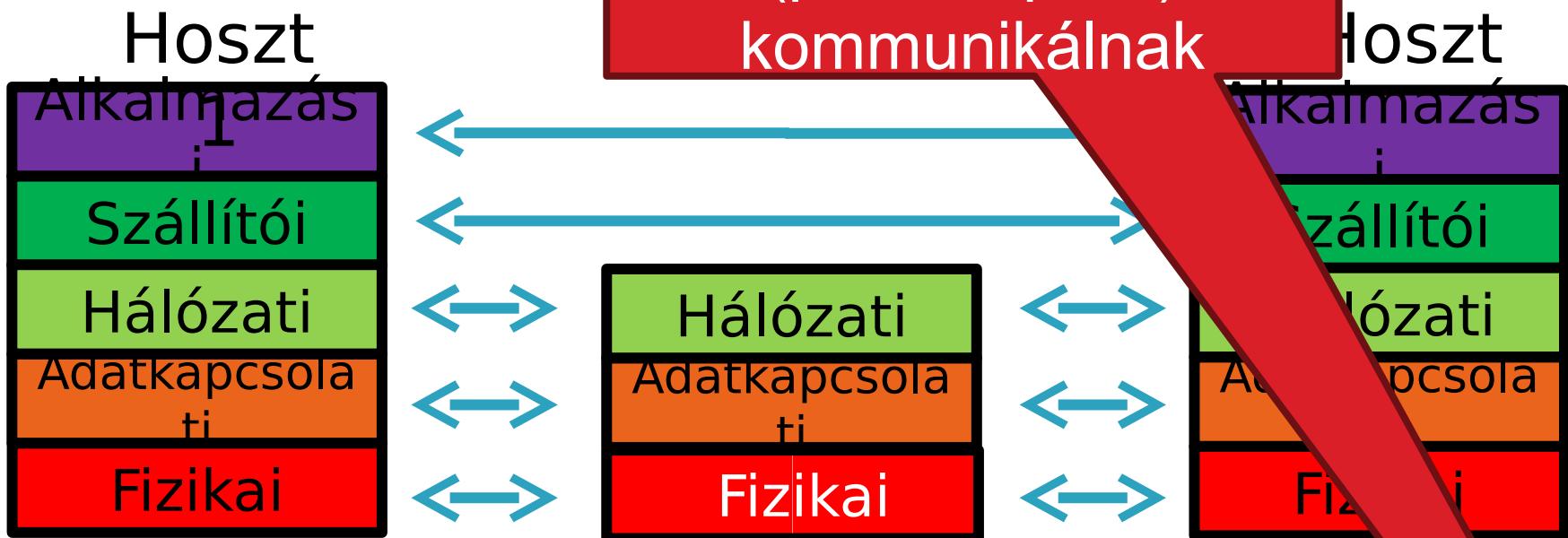
Egyedi port minden

Az alkalmazások mind ugyanazt a hálózatot használják

Végpontok azonosítása:  $\langle \text{src\_ip}, \text{src\_port}, \text{dest\_ip}, \text{dest\_port}, \text{proto} \rangle$

# Réteg modellek újragondolva

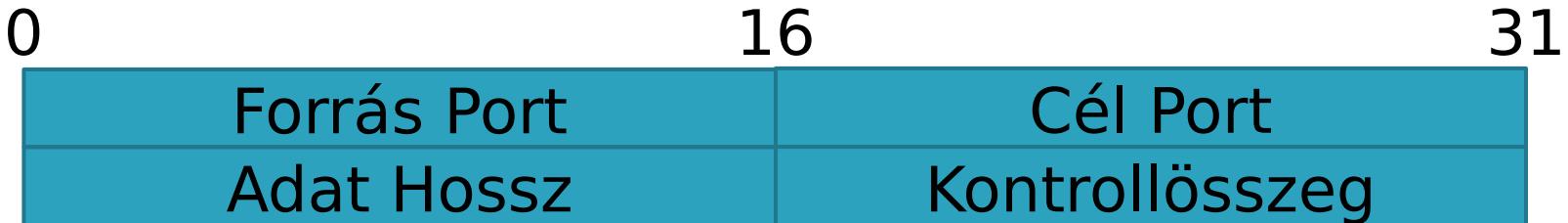
9



- A legalacsonyabb szintű végpont-végpont protokoll
  - A szállítói réteg fejlécei csak a forrás és cél végpontok olvassák

# User Datagram Protocol (UDP)

10



- 8 bájtos UDP fejléc
- Egyszerű, kapcsolatnélküli átvitel
  - C socketek: SOCK\_DGRAM
- Port számok teszik lehetővé a demultiplexálást
  - 16 bit = 65535 lehetséges port
  - 0 port nem engedélyezett

# UDP felhasználások

11

- A TCP után vezették be
  - Miért?
- Nem minden alkalmazásnak megfelelő a TCP
- UDP felett egyedi protokollok valósíthatók meg
  - Megbízhatóság? Helyes sorrend?
  - Folyam vezérlés? Torlódás vezérlés?
- Példák
  - RTMP, real-time média streamelés (pl. hang, video)

# Transmission Control Protocol

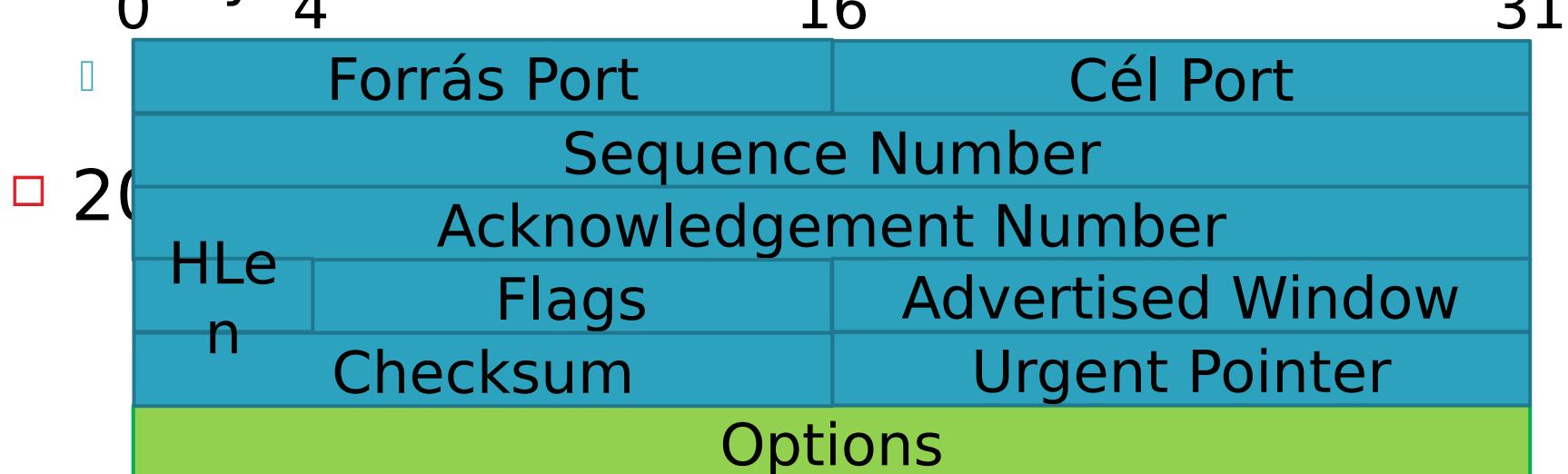
12

- Megbízható, sorrend helyes, két irányú bájt folyamok

- Port számok a demultiplexáláshoz

- Kapcsolat alapú

- Folyam vezérlés



# Kapcsolat felépítés

13

- Miért van szükség kapcsolat felépítésre?
  - Állapot kialakítása minden két végponton
  - Legfontosabb állapot: sorszámok/sequence numbers
    - Az elküldött bájtok számának nyilvántartása
    - Véletlenszerű kezdeti érték
- Fontos TCP flag-ek/jelölő bitek (1 bites)
  - SYN - szinkronizációs, kapcsolat felépítéshez
  - ACK - fogadott adat nyugtázása
  - FIN - vége kapcsolat lezárásához

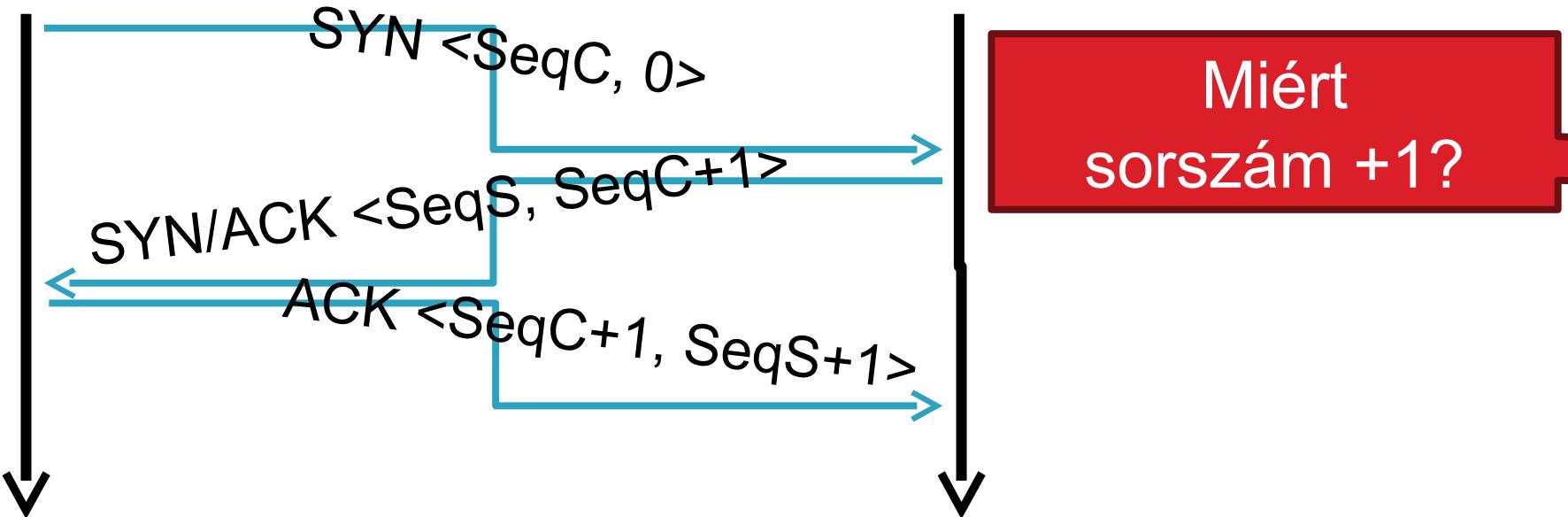
# Three Way Handshake

## Három-utas kézfogás

14

Kliens

Szerver



- Mindkét oldalon:
  - Másik fél értesítése a kezdő sorszámról
  - A másik fél kezdő sorszámának nyugtázása

# Kapcsolat felépítés problémája

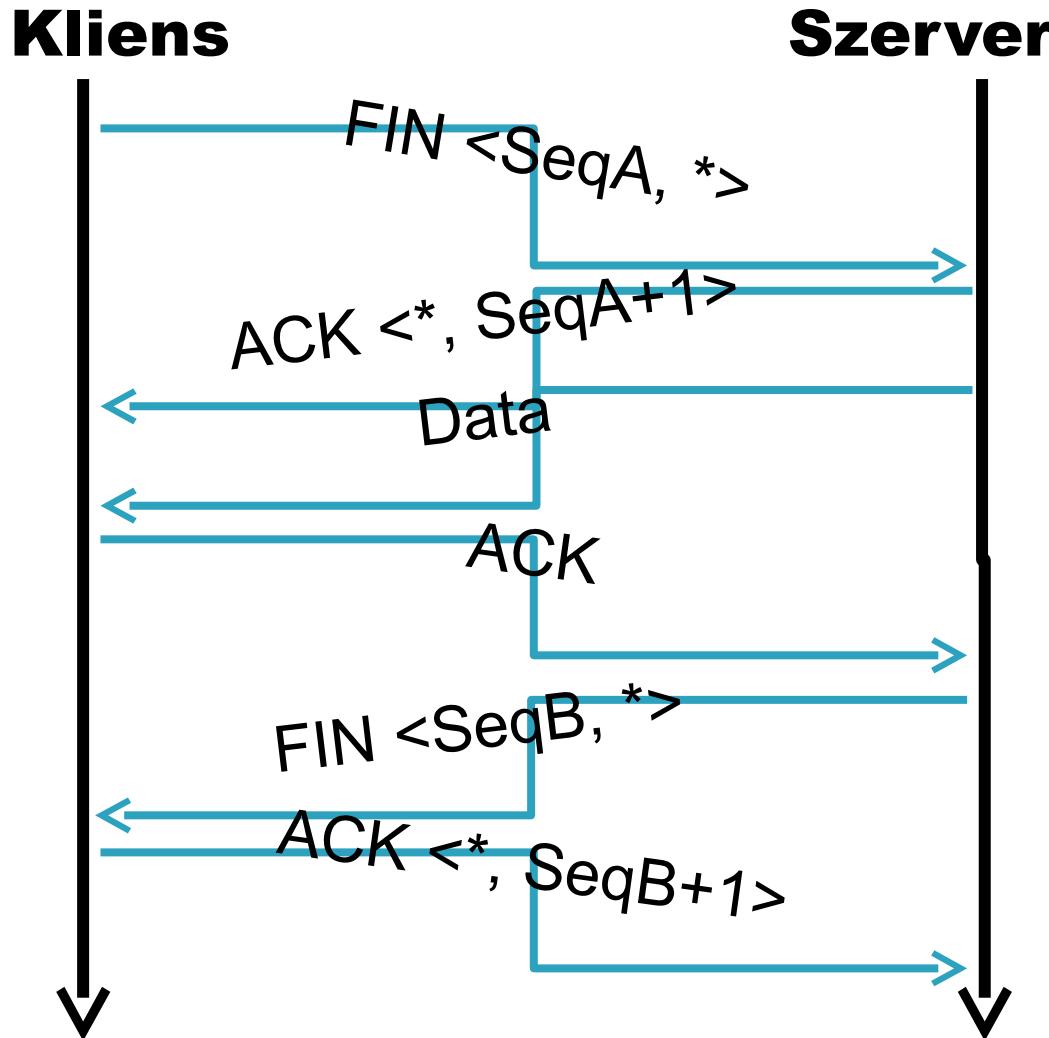
15

- Kapcsolódási zűrzavar
  - Azonos hoszt kapcsolatainak egyértelműsítése
  - Véletlenszerű sorszámmal - biztonság
- Forrás hamisítás
  - Kevin Mitnick
  - Jó random szám generátor kell hozzá!
- Kapcsolat állapotának kezelése
  - minden SYN állapotot foglal a szerveren
  - SYN flood = denial of service (DoS) támadás

# Kapcsolat lezárása

16

- Mindkét oldal kezdeményezheti a kapcsolat bontását
- A másik oldal még folytathatja a küldést
  - Félig nyitott kapcsolat
  - *shutdown()*
- Az utolsó FIN nyugatáza



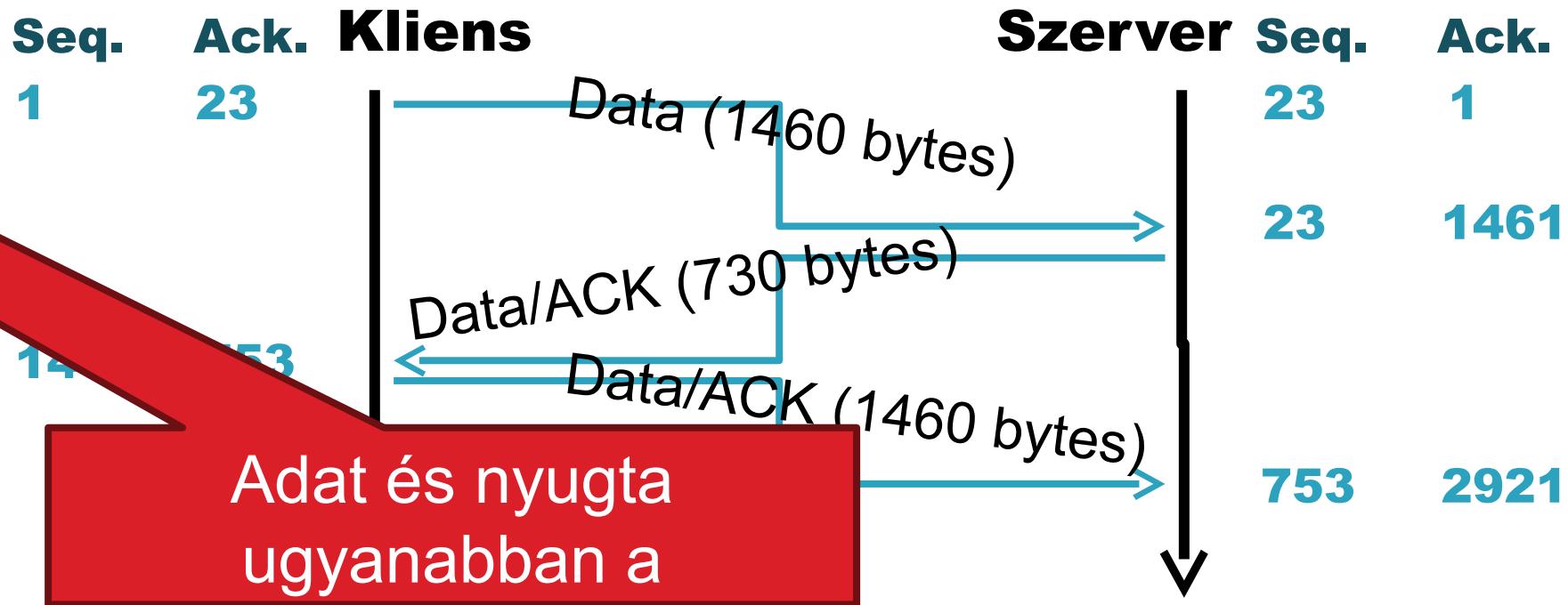
# Sorszámok tere

17

- A TCP egy absztrakt bájt folyamot valósít meg
  - A folyam minden bájtja számozott
  - 32-bites érték, körbefordul egy idő után
  - Kezdetben, véletlen érték a kapcsolat felépítésénél.
- A bájt folyamot szegmensekre bontjuk (TCP csomag)
  - A méretét behatárolja a Maximum Segment Size (MSS) 8 Segment 8 Segment 9 Segment 10 13450 14950 16050 17550

# Kétirányú kapcsolat

18



- Mindkét fél küldhet és fogadhat adatot
  - Különböző sorszámok a két irányba

# Folyam vezérlés

19

- Probléma: Hány csomagot tud a küldő átvinni?
  - Túl sok csomag túlterhelheti a fogadót
  - A fogadó oldali puffer-méret változhat a kapcsolat során
- Megoldás: csúszóablak
  - A fogadó elküldi a küldőnek a pufferének méretét
  - Ezt nevezzük meghirdetett ablaknak: **advertised window**
  - Egy  $n$  ablakmérethez, a küldő  $n$  bájtot küldhet el ACK fogadása nélkül

# Folyam vezérlés - csúszóablak

20

Packet Sent

Src. Port	Dest. Port
Sequence Number	
Acknowledgement Number	
HL	Flags
Checksum	Window Urgent Pointer

Pufferelni kell a  
nyugtáig

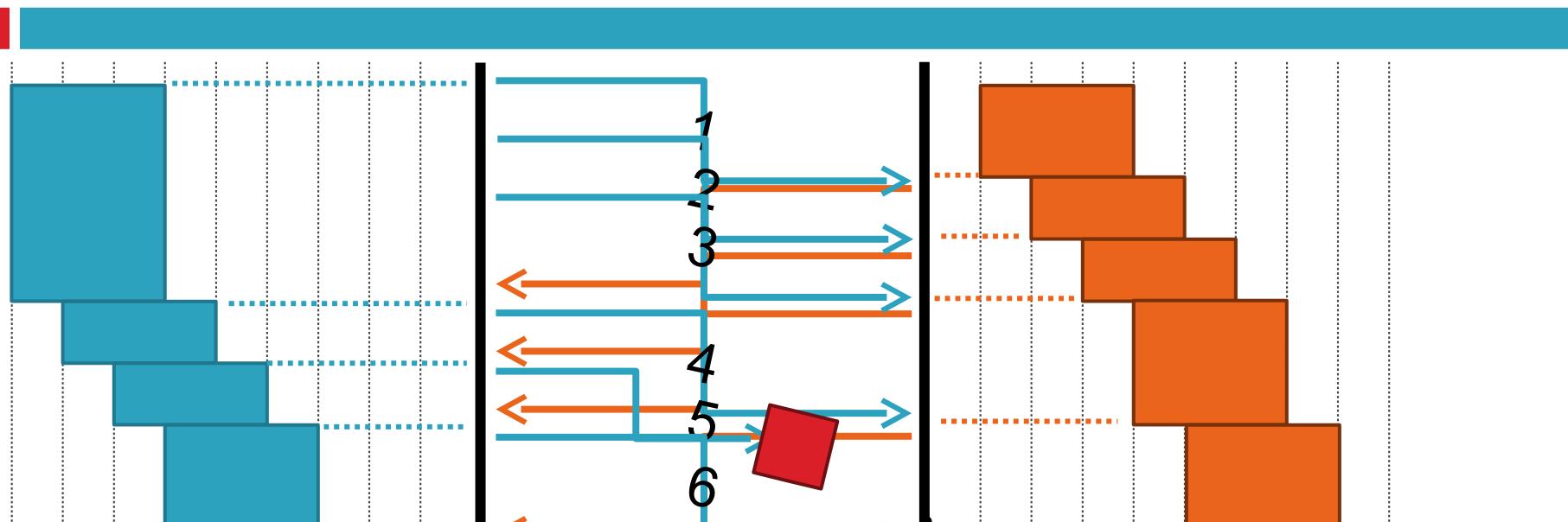
Packet Received

Src. Port	Dest. Port
Sequence Number	
Acknowledgement Number	
HL	Flags
Checksum	Window Urgent Pointer



# Csúszóablak példa

21



## A TCP ACK ütemezett

- Rövid RTT → gyors ACK → az ablak gyorsan léptethető



# Megfigyelések

22

- Átvitel arányos ~  $w/RTT$ 
  - $w$ : küldési ablakméret
  - RTT: körülfordulási idő
- A küldőnek pufferelni kell a nem nyugtáztott csomagokat a lehetséges újraküldések miatt
- A fogadó elfogadhat nem sorrendben érkező csomagokat, de csak amíg az elfér a pufferben

# Mit nyugtázhat a fogadó?

23

1. minden egyes csomagot
2. Használhat *kumulált nyugtát*, ahol egy n sorszámú nyugta minden  $k < n$  sorszámú csomagot nyugtáz
3. Használhat *negatív nyugtát (NACK)*, megjelölve, hogy mely csomag nem érkezett meg
4. Használhat *szelektív nyugtát (SACK)*, jelezve, hogy mely csomagok érkeztek meg, akár nem megfelelő sorrendben

# Sorszámok

24

- 32 bites, unsigned
  - Miért ilyen nagy?
- A csúszó-ablakhoz szükséges...
  - $|sorszámok\ tere| > 2 * |Küldő\ ablak\ mérete|$
  - $2^{32} > 2 * 2^{16}$
- Elkörülözött csomagok kivédése
  - IP csomagok esetén a maximális élettartam (MSL) of 120 mp
    - Azaz egy csomag 2 percig bolyonghat egy hálózatban

# Buta ablak szindróma

25

- Mi van, ha az ablak mérete nagyon kicsi?
  - Sok, apró csomag. A fejlécek dominálják az
    - Header Data
    - Header Data
    - Header Data
    - Header Data
- Lényegében olyan, mintha bájtonként küldenénk az üzenetet...
  1. `for (int x = 0; x < strlen(data); ++x)`
  2. `write(socket, data + x, 1);`

# Nagle algoritmusa

26

1. Ha az ablak  $\geq$  MSS és az elérhető adat  $\geq$  MSS:  
Küldjük el az adatot
2. Különben ha van nem nyuhatáztott adat::  
Várakoztassuk az adatot egyszer, ha nyugtát nem kapunk  
Küldjünk egy nem teljes csomagot, ha
3. Különben: küldjük az adatot

⚠️ Probléma: Nagle algoritmusa késlelteti az átvitelt

# Hiba detektálás

27

- A kontrollösszeg detektálja a hibás csomagokat
  - I Az IP, TCP fejlécből és az adatból számoljuk
- A sorszámok segítenek a sorrendhelyes átvitelben
  - I Duplikátumok eldobása
  - I Helytelen sorrendben érkező csomagok sorba rendezése vagy eldobása
  - I Hiányzó sorszámok elveszett csomagot jeleznek
- A küldő oldalon: elveszett csomagok

# Retransmission Time Outs (RTO)

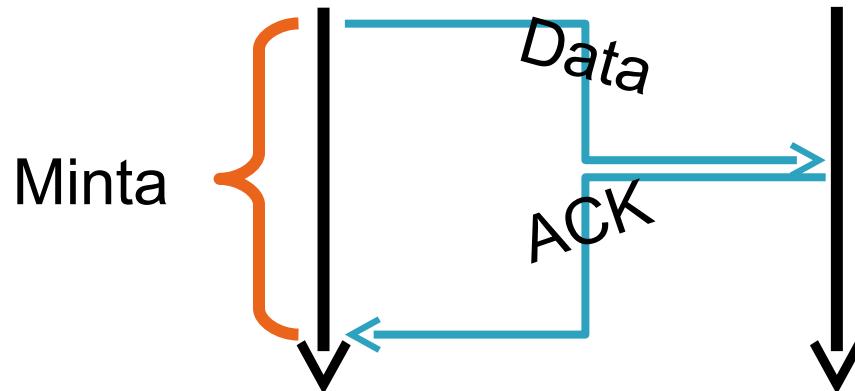
28

Problémá: Időtúllépés RTT-hez kapcsolása



# Round Trip Time becslés

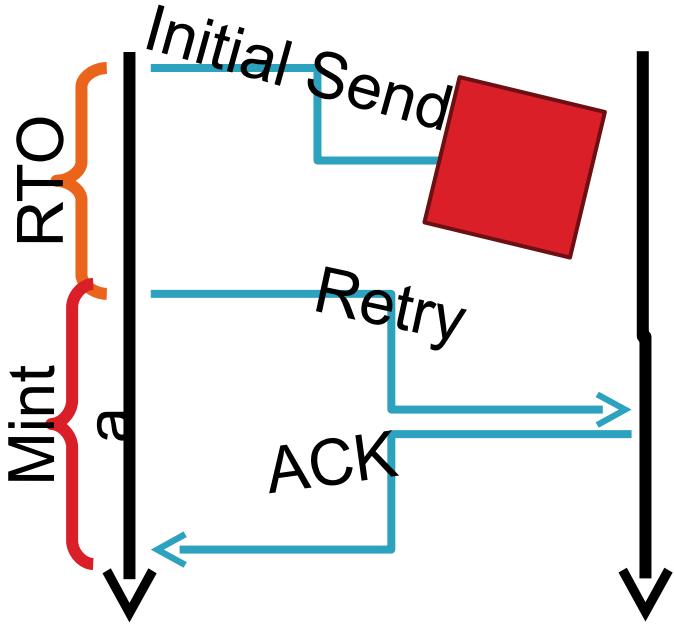
29



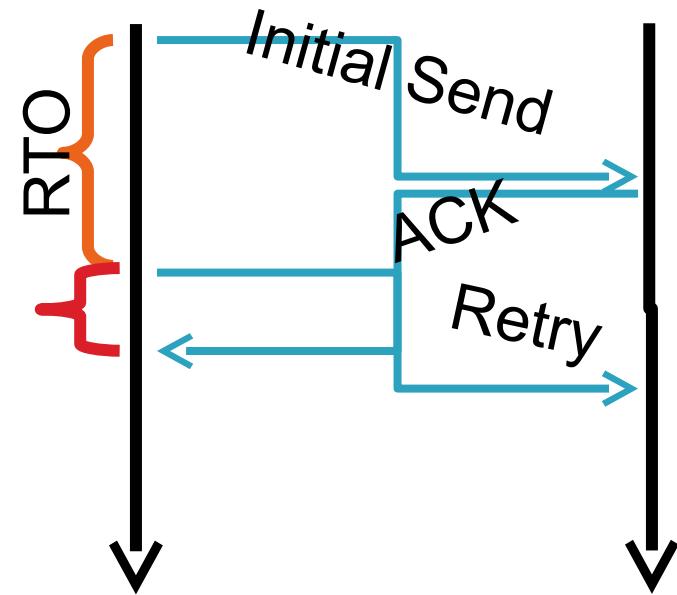
- Az eredeti TCP RTT becslője:
  - RTT becslése mozgó átlaggal
  - $\text{new\_rtt} = \alpha (\text{old\_rtt}) + (1 - \alpha)(\text{new\_sample})$
  - Javasolt  $\alpha$ : 0.8-0.9 (0.875 a legtöbb TCP esetén)
- $\text{RTO} = 2 * \text{new\_rtt}$  (a TCP konzervatív)

# Az RTT minta félre is értelmezhető

30



Minta?



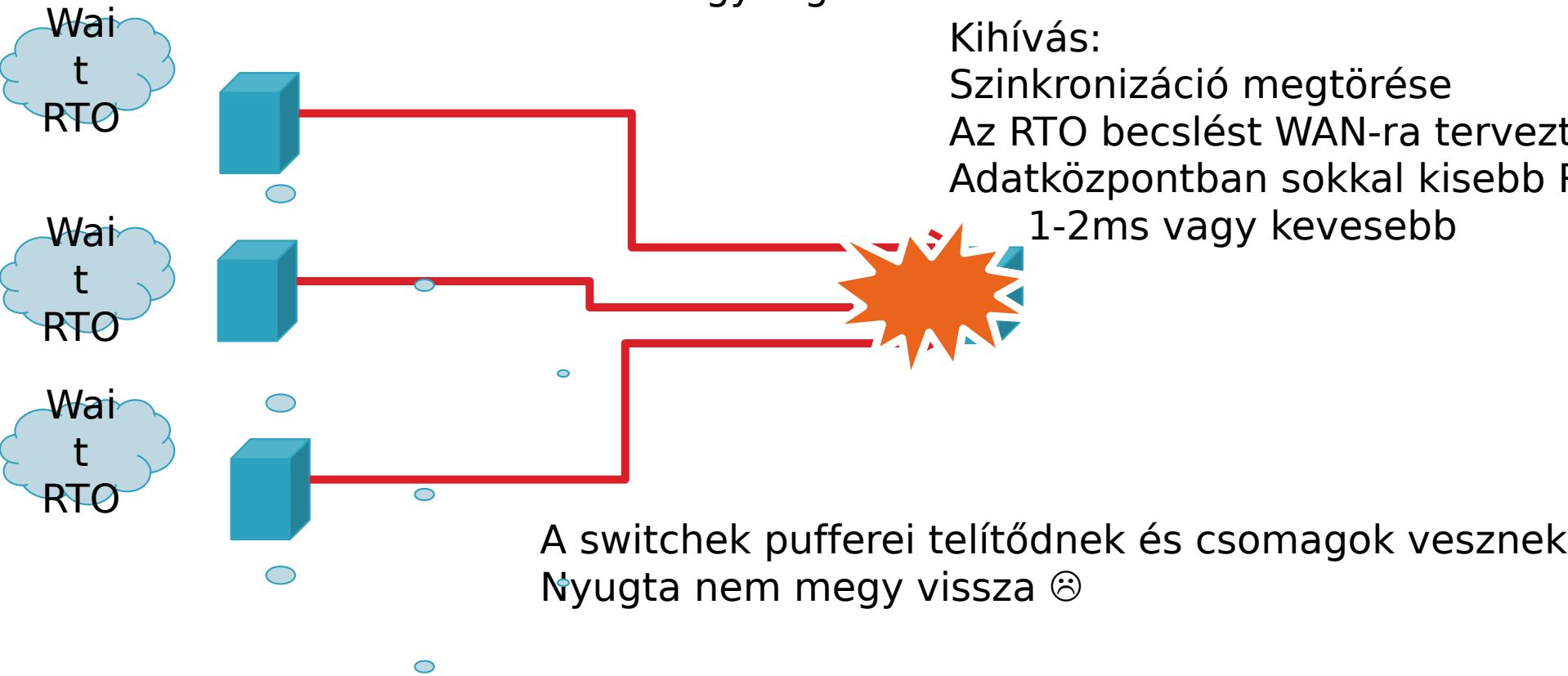
- **Karn algoritmusa:** dobjuk el azokat a mintákat, melyek egy csomag újraküldéséből származnak

# RTO adatközpontokban???

31

- TCP Incast probléma - pl. Hadoop, Map Reduce, HDFS, GFS

Sok szimultán küldő egy fogadóhoz



# Mi az a torlódás?

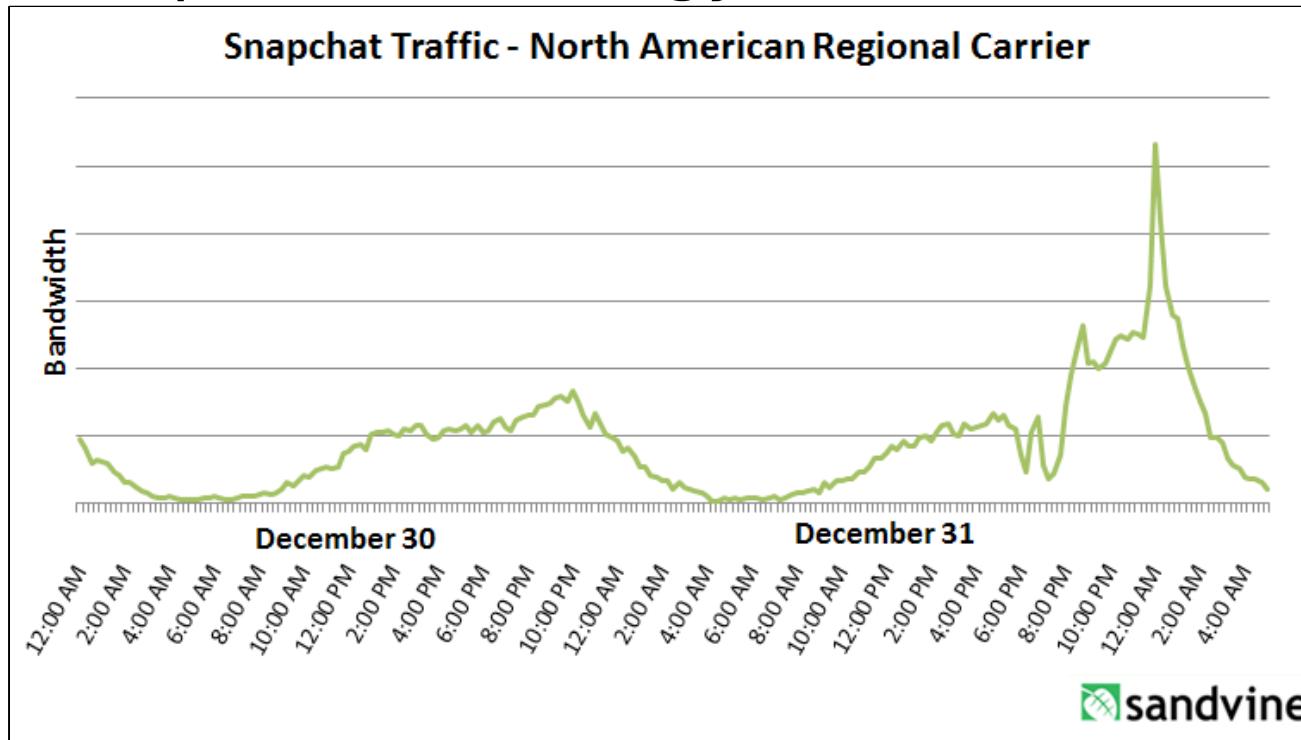
32

- A hálózat terhelése nagyobb, mint a kapacitása
  - A kapacitás nem egyenletes a hálózatban
    - Modem vs. Cellular vs. Cable vs. Fiber Optics
  - Számos folyam verseng a sávszélességért
    - otthoni kábel modem vs. corporate datacenter
  - A terhelés időben nem egyenletes
    - Vasárnap este 10:00 = BitTorrent Game of Thrones

# Mi az a torlódás?

33

- A hálózat terhelése nagyobb, mint a kapacitása
- A kapacitás nem egyenletes a hálózatban



# Miért rossz a torlódás?

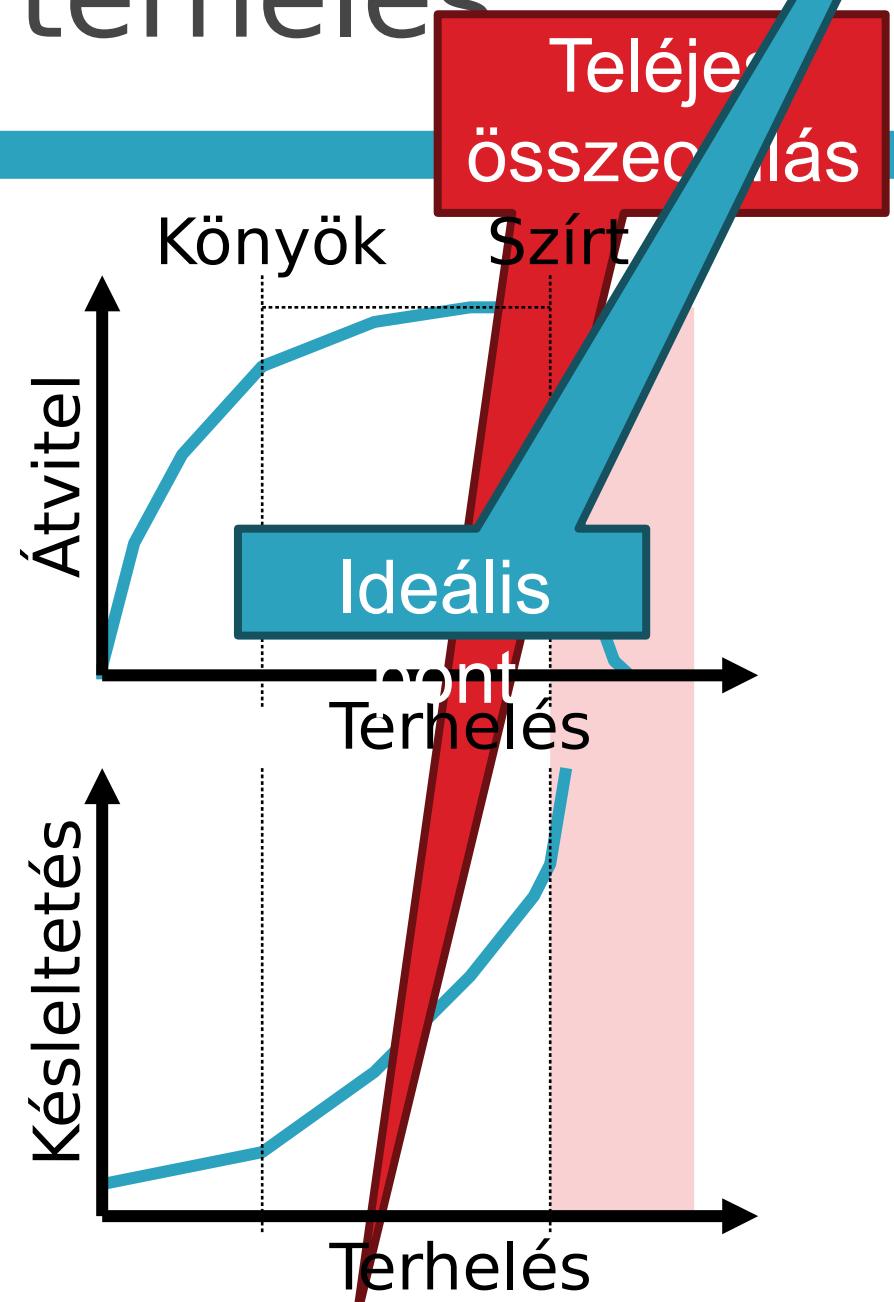
34

- Csomagvesztést eredményez
  - A routerek véges memóriával (puffer) rendelkeznek
  - Önhasonló Internet forgalom, nincs puffer, amiben ne okozna csomagvesztést
  - Ahogy a routerek puffere elkezd telítődni, csomagokat kezd eldobni... (RED)
- Gyakorlati következmények
  - A routerek sorai telítődnek, megnövekedett késleltetés

# Megnövekedett terhelés

35

- Könyök („knee”)- a pont, ami után
  - Az átvitel szinte alig nő
  - Késleltetés viszont gyorsan emelkedik
- Egy egyszerű sorban (M/M/1)
  - Késleltetés =  $1/(1 - \text{utilization})$
- Szűrő („cliff”)- a pont

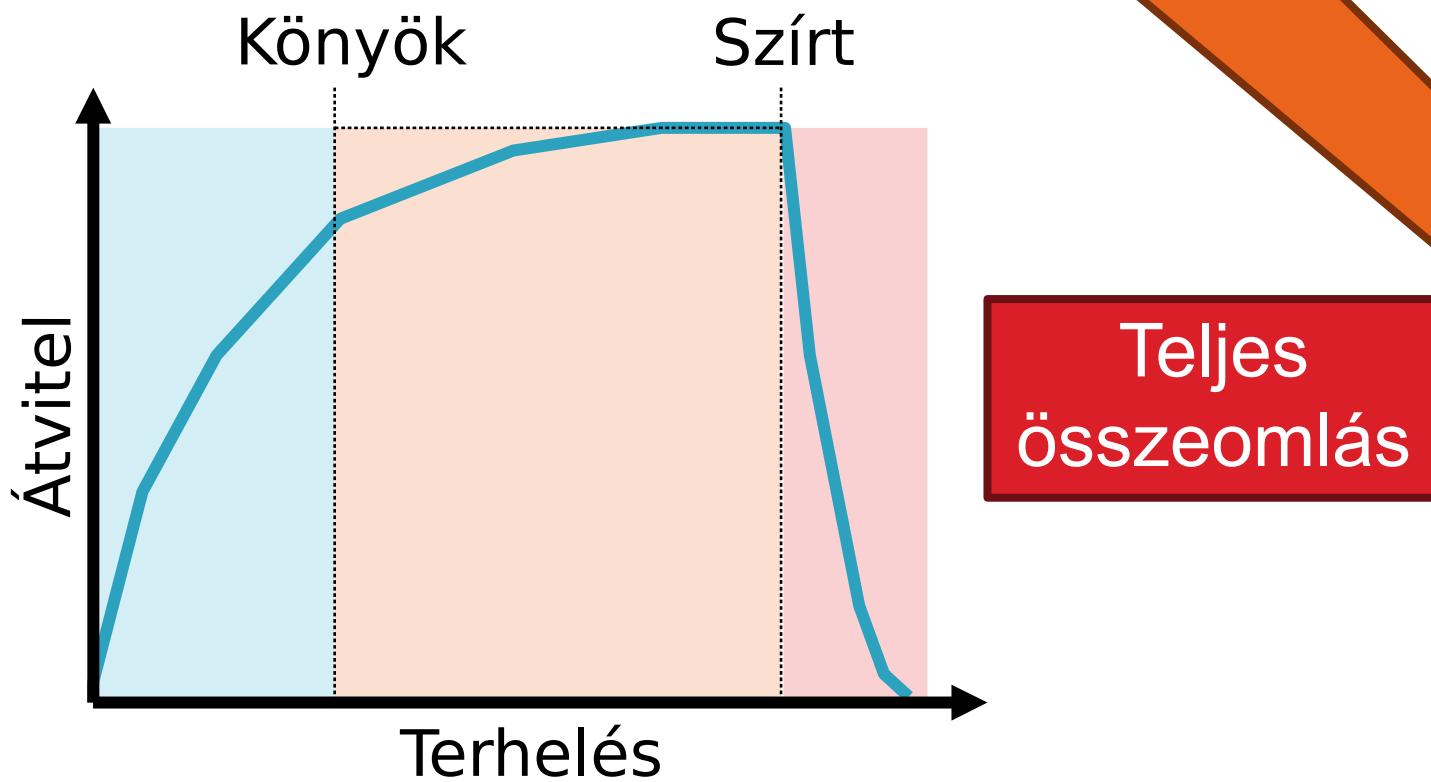


# Torlódás vezérlés vs torlódás elkerülés

36

Torlódás elkerülés:  
Maradj a könyök bal

Torlódás vezérlés  
Maradj a szírt bal



# Számítógépes Hálózatok

**10. Előadás: Szállítói réteg 2**

Based on slides from

# Szállítói réteg

2



- Feladat:
  - Adatfolyamok demultiplexálása
- További lehetséges feladatok:
  - Hosszú élettartamú kapcsolatok
  - Megbízható, sorrendhelyes csomag leszállítás
  - Hiba detektálás

# Kivonat

- ❑ UDP
- ❑ TCP
- ❑ Torlódás vezérlés
- ❑ TCP evolúciója
- ❑ A TCP problémái

# Mi az a torlódás?

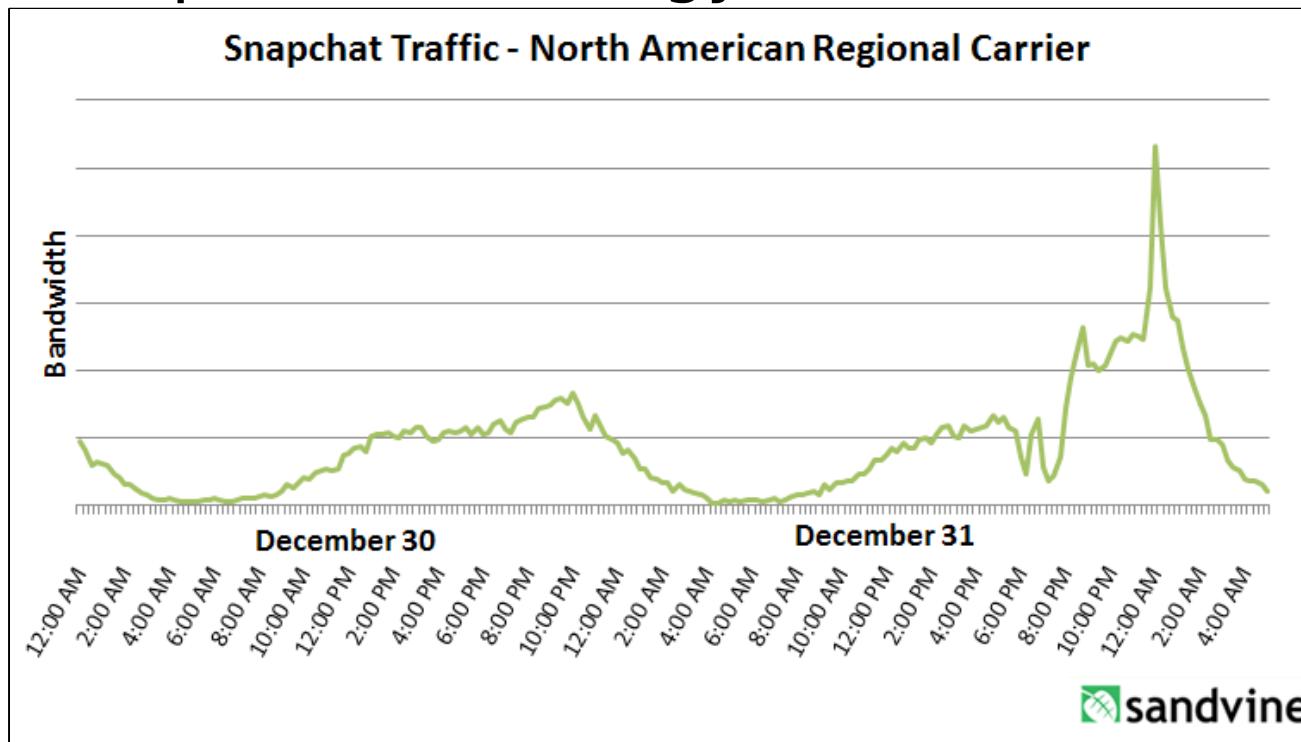
4

- A hálózat terhelése nagyobb, mint a kapacitása
  - A kapacitás nem egyenletes a hálózatban
    - Modem vs. Cellular vs. Cable vs. Fiber Optics
  - Számos folyam verseng a sávszélességért
    - otthoni kábel modem vs. corporate datacenter
  - A terhelés időben nem egyenletes
    - Vasárnap este 10:00 = BitTorrent Game of Thrones

# Mi az a torlódás?

5

- A hálózat terhelése nagyobb, mint a kapacitása
- A kapacitás nem egyenletes a hálózatban



otics  
égért  
center  
of Thrones

# Miért rossz a torlódás?

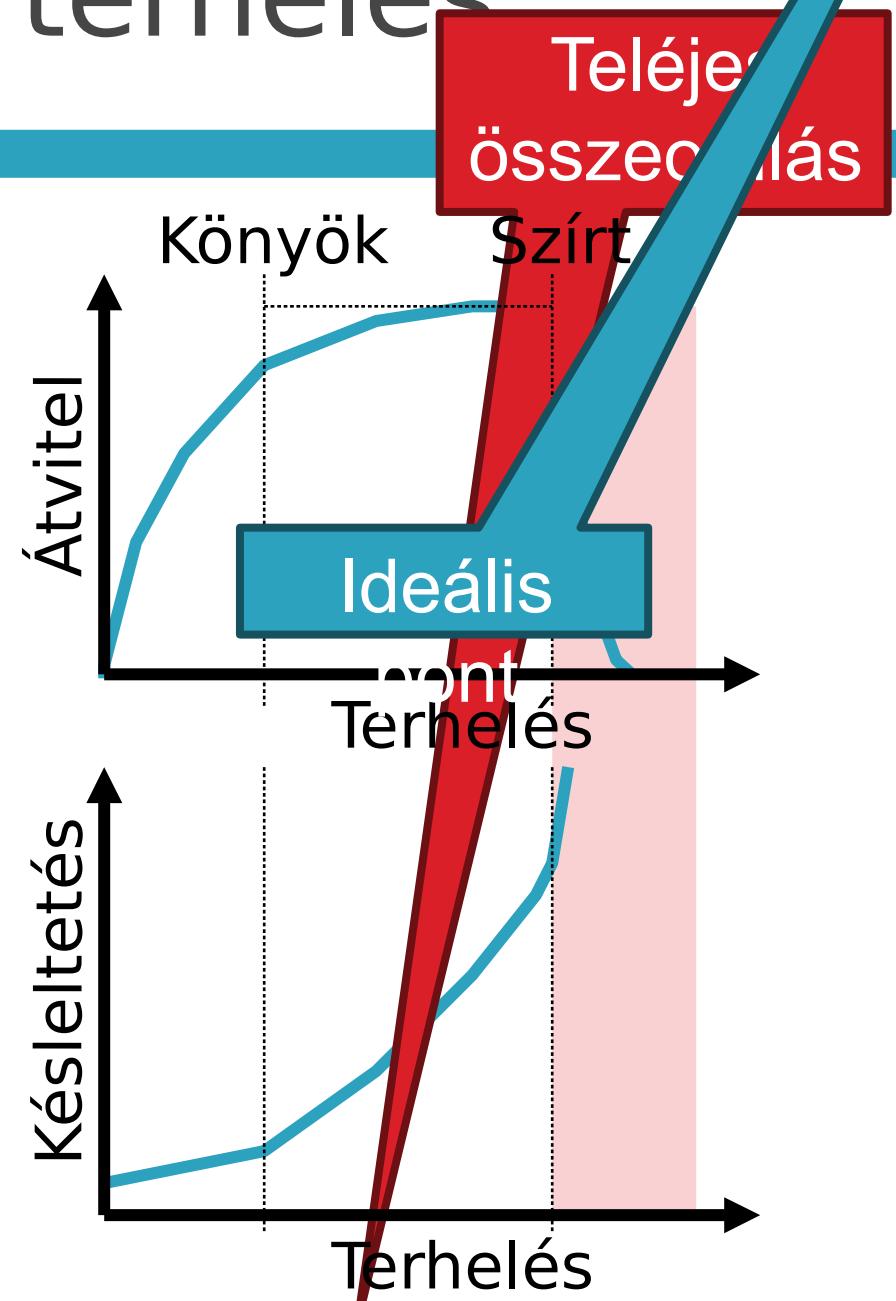
6

- Csomagvesztést eredményez
  - A routerek véges memóriával (puffer) rendelkeznek
  - Önhasonló Internet forgalom, nincs puffer, amiben ne okozna csomagvesztést
  - Ahogy a routerek puffere elkezd telítődni, csomagokat kezd eldobni... (RED)
- Gyakorlati következmények
  - A routerek sorai telítődnek, megnövekedett késleltetés

# Megnövekedett terhelés

7

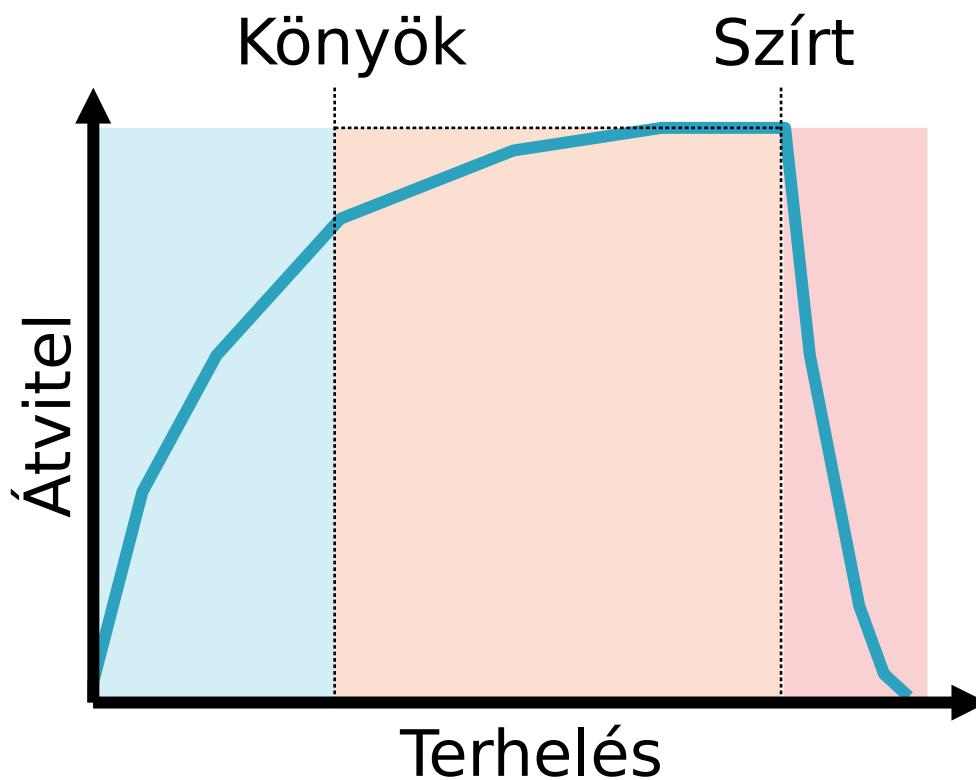
- Könyök („knee”)- a pont, ami után
  - Az átvitel szinte alig nő
  - Késleltetés viszont gyorsan emelkedik
- Egy egyszerű sorban (M/M/1)
  - Késleltetés =  $1/(1 - \text{utilization})$
- Szűrő („cliff”)- a pont



# Torlódás vezérlés vs torlódás elkerülés

Torlódás elkerülés:  
Maradj a könyök bal

Torlódás vezérlés  
Maradj a szírt bal



Teljes  
összeomlás

# Advertised Window

## 9 Meghirdetett ablak,

Megoldja-e a törökös problémáját a TCP útirányításban esetén a meghirdetett ablak használata?

NEM

- Ez az ablak csak a fogadót védi a túlterheléstől
- Egy kellően gyors fogadó kimaxolhatja ezt az ablakot
  - Mi van, ha a hálózat lassabb, mint a fogadó?
  - Mi van, ha vannak konkurens folyamok is?
- Következmények

# Általános megoldások

10

- Ne csinálunk semmit, küldjük a csomagokat megkülönböztetés nélkül
  - Nagy csomagvesztés, jósolhatatlan teljesítmény
  - Teljes összeomláshoz vezethet
- Erőforrás foglalás
  - Folyamokhoz előre sávszélességet allokálunk
  - Csomagküldés előtt egy tárgyalási szakaszra is szükség van
  - Hálózati támogatás kell hozzá
- Dinamikus beállítás

# TCP Torlódásvezérlés

11

- minden TCP kapcsolat rendelkezik egy ablakkal
  - A nem-nyugtázott csomagok számát vezérli
- Küldési ráta ~ window/RTT
- Ötlet: ablak méretének változtatása a küldési ráta vezérléséhez
- Vezessünk be egy **torlódási ablakot** (**congestion window**) a küldő oldalon
  - Torlódás vezérlés egy küldő oldali probléma
  - Jelölése: cwnd

# Két fő komponens

12

## 1. Torlódás detektálás

- Eldobott csomag egy megbízható jel
  - Késleltetés alapú megoldások - nehéz és kockázatos
- Hogyan detektáljuk a csomag eldobását? Nyugtával
  - Időkorlát lejár ACK fogadása nélkül
  - Számos duplikált ACK jön be sorban (később lesz róla szó)

## 2. Ráta beállító algoritmus

# Ráta vezérlés

13

- Tudjuk, hogy a TCP ACK ütemezett
  - Torlódás = késleltetés = hosszú várakozás a nyugták között
  - Nincs torlódás = alacsony késleltetés = gyors ACK
- Alapvető algoritmus
  - ACK fogadása esetén: növeljük a cwnd ablakot
    - Adat leszállítva, valószínűleg gyorsabban is küldhetünk
    - *cwnd* növekedése arányos az RTT-vel

# Torlódás vezérlés megvalósítása

14

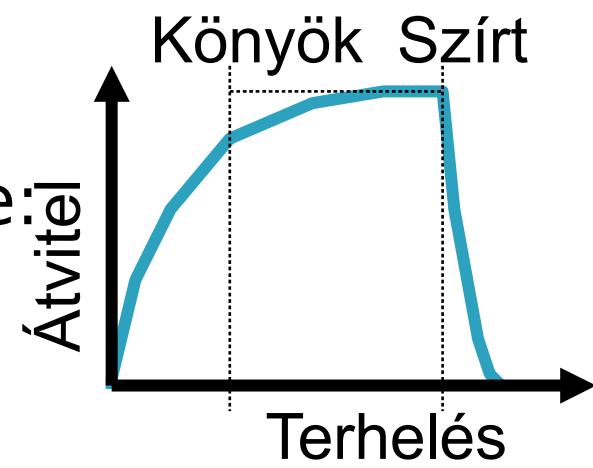
- Három változót kell nyilvántartani:
  - $cwnd$ : torlódási ablak
  - $adv\_wnd$ : a fogadó meghirdetett ablaka
  - $ssthresh$ : vágási érték (a  $cwnd$  frissítésére használjuk)
- Küldésnél használjuk:  $wnd = \min(cwnd, adv\_wnd)$
- A torlódás vezérlés két fázisa:
  1. Lassú indulás („Slow start”) ( $cwnd < ssthresh$ )
    - Az új bottleneck (legszűkebb) sávszélessége

14

# Lassú indulás - Slow Start

15

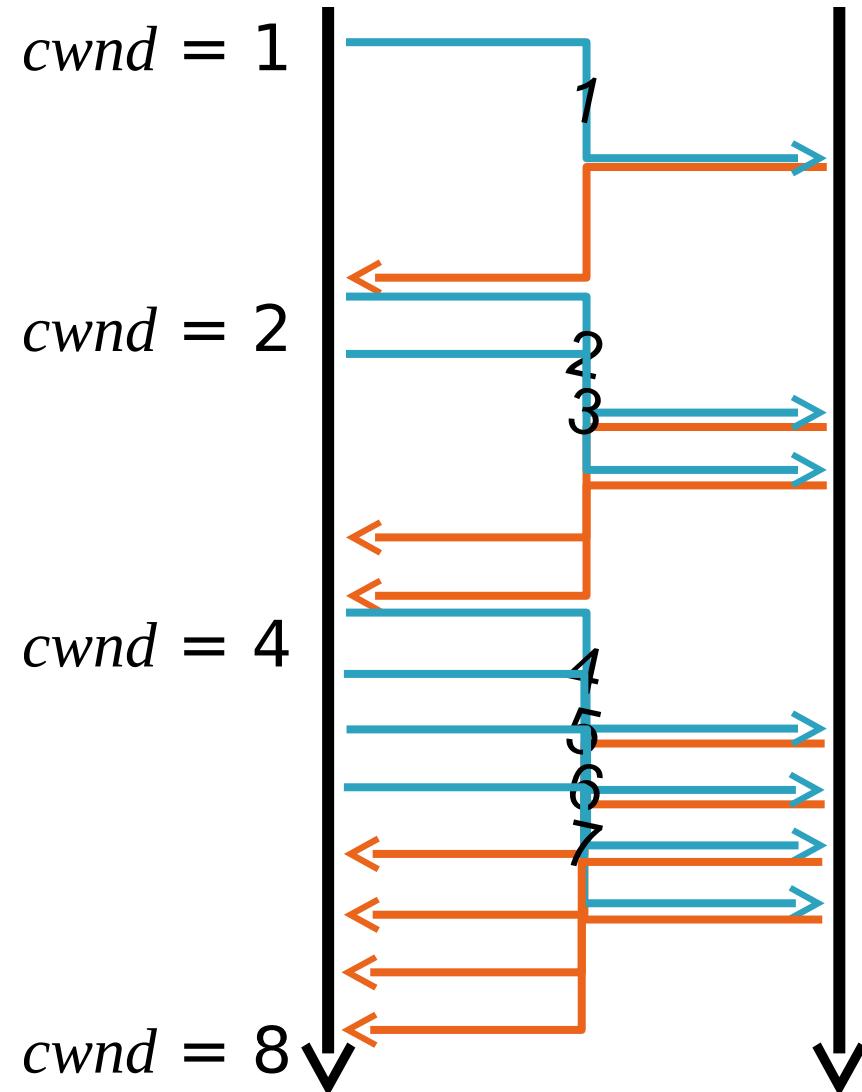
- Cél, hogy gyorsan elérjük a könyök pontot
- Egy kapcsolat kezdetén (vagy újraindításakor)
  - $cwnd = 1$
  - $ssthresh = adv\_wnd$
  - minden nyugtázott szegmensre:  $cwnd++$
- Egészen addig amíg
  - El nem érjük az  $ssthresh$  értéket



# Slow Start példa

16

- $cwnd$  gyorsan nő
- Lelassul, amikor...
  - $cwnd >= ssthresh$
  - Vagy csomagvesztés történik



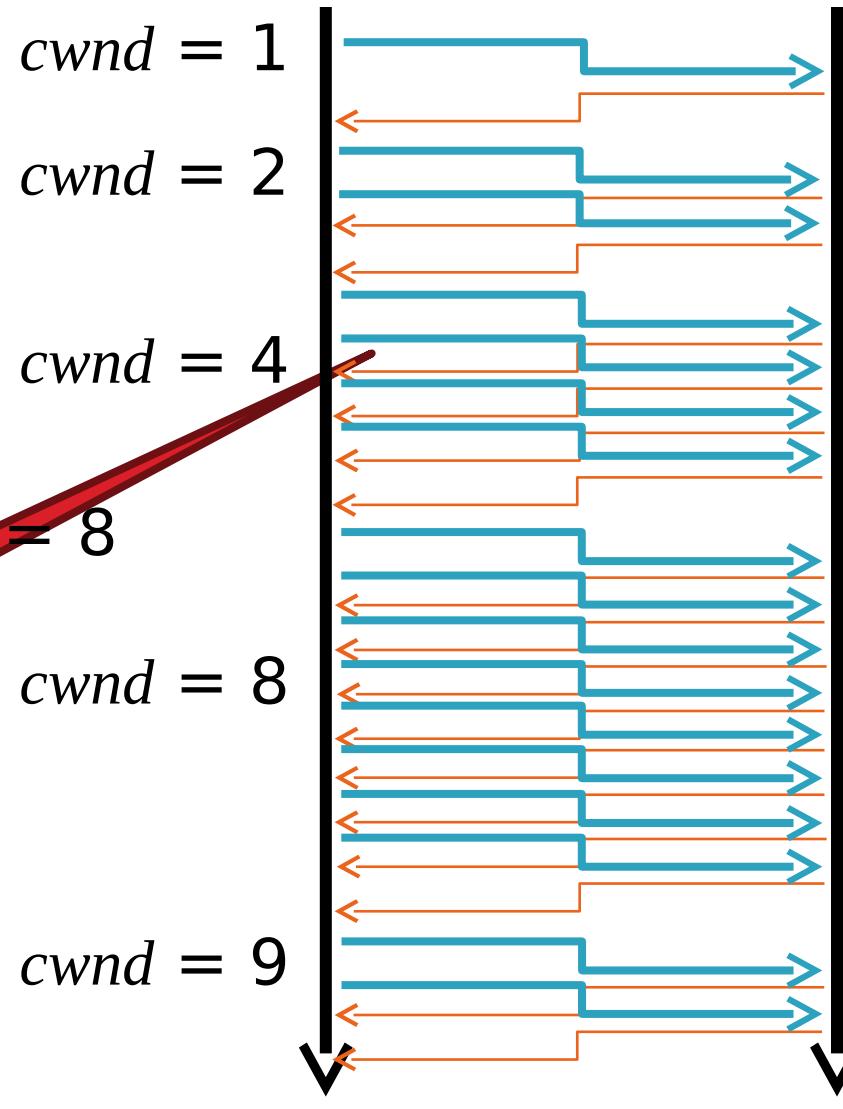
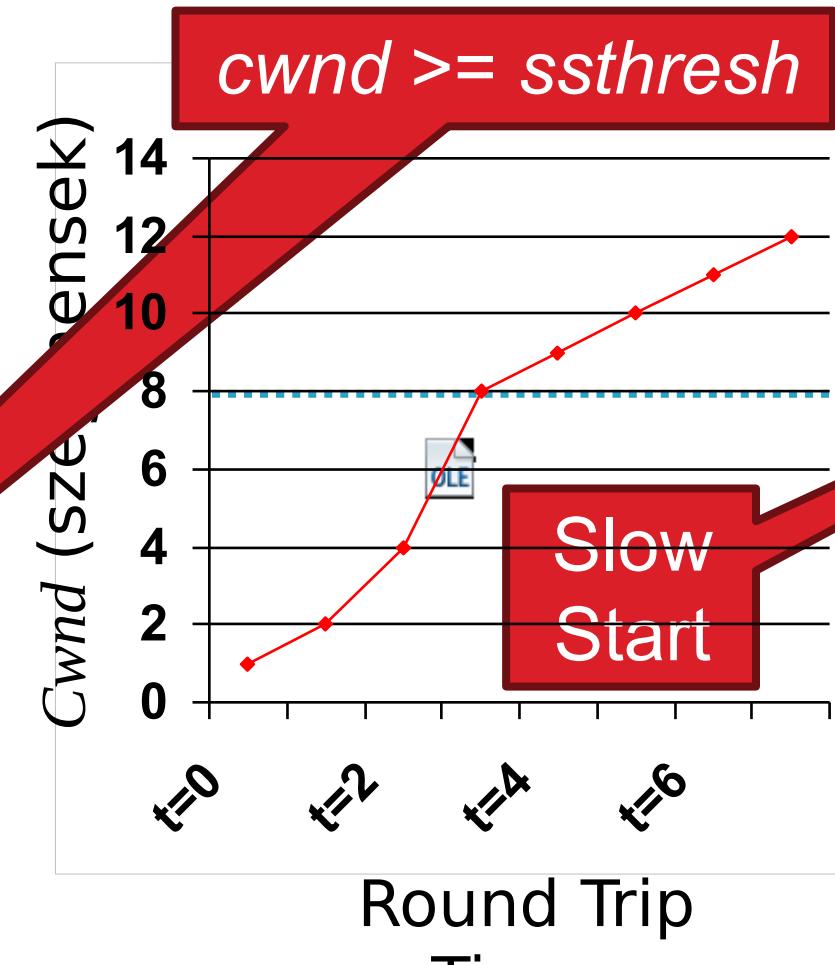
# Torlódás elkerülés

17

- Additive Increase Multiplicative Decrease (AIMD) mód
- $ssthresh$  valójában egy alsóbecslés a könyök pontra
- **Ha**  $cwnd >= ssthresh$  **akkor**  
Minden nyugtázott szegmens alkalmával növeljük a  $cwnd$  értékét ( $1/cwnd$ )-vel (azaz  $cwnd += 1/cwnd$ ).
- Azaz a  $cwnd$  eggyel nő, ha minden csomag nyugtázva lett.

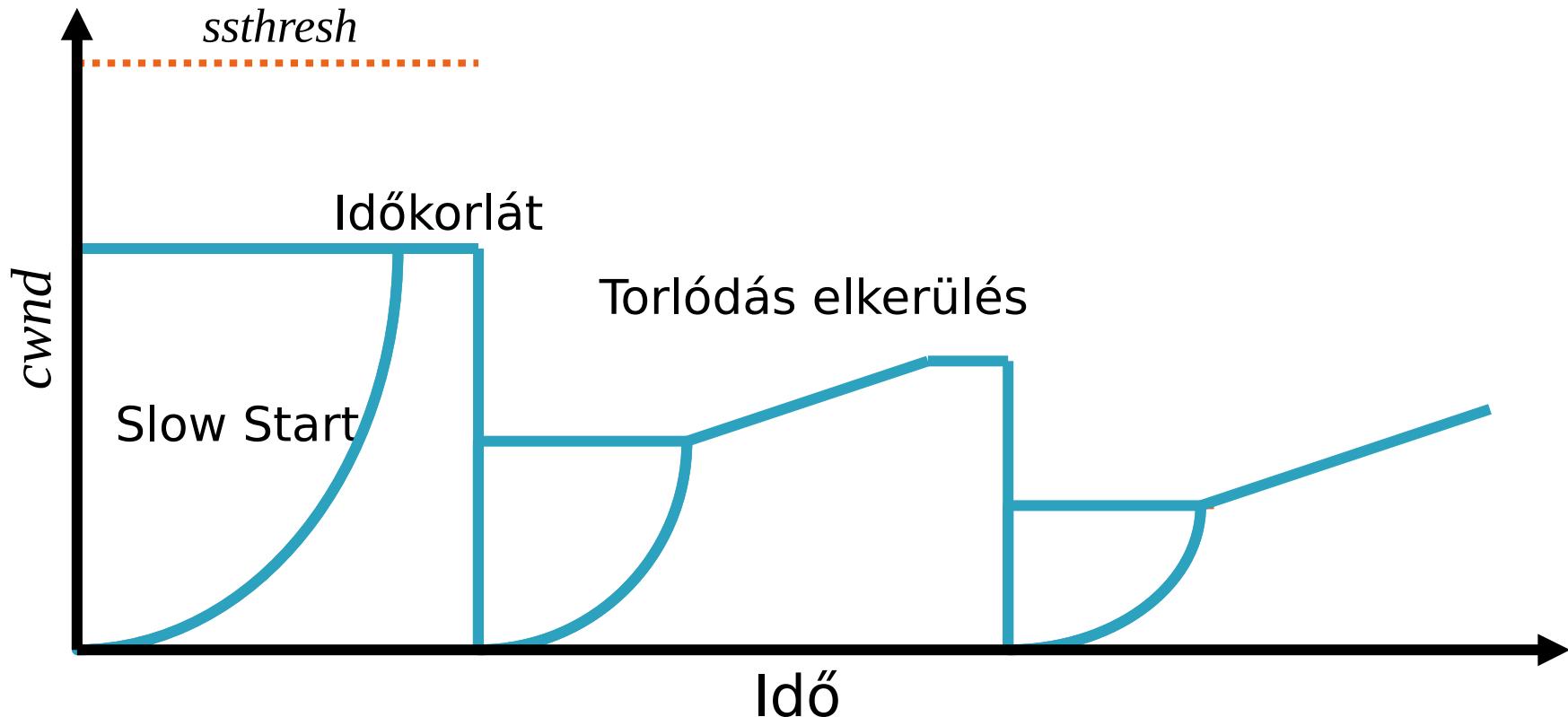
# Torlódás elkerülés példa

18



# A teljes kép - TCP Tahoe (az eredeti TCP)

19



# Összefoglalás - TCP

## jellemzői

20

„A *TCP* egy kapcsolatorientált megbízható szolgáltatás kétirányú bájt-folyamokhoz.”

### Kapcsolatorientált

- Két résztvevő, ahol egy résztvevőt egy *IP-cím* és egy *port* azonosít.
- A kapcsolat egyértelműen azonosított a résztvevő párral.
- Nincs se *multi-*, se *broadcast* üzenetküldés.
- A kapcsolatot fel kell építeni és le kell bontani.
- Egy kapcsolat a lezárásáig aktív.

# Összefoglalás - TCP

## jellemzői

21

„A TCP egy kapcsolatorientált megbízható szolgáltatás kétirányú bájt-folyamokhoz.”

### Megbízhatóság

- minden csomag megérkezése nyugtázsra kerül.
- A nem nyugtázott adatcsomagokat újraküldik.
- A fejléchez és a csomaghoz ellenőrzőösszeg van rendelve.
- A csomagokat számozza, és a fogadónál sorba rendezésre kerülnek a csomagok a sorszámaik alapján.
- Duplikátumokat törli.

# Összefoglalás - TCP

## jellemzői

22

„A TCP egy kapcsolatorientált megbízható szolgáltatás kétirányú bájt-folyamokhoz.”

### Kétirányú bájtfolyam

- Az adatok két egymással ellentétes irányú bájt-sorozatként kerülnek átvitelre.
- A tartalom nem interpretálódik.
- Az adatcsomagok időbeli viselkedése megváltozhat: átvitel sebessége növekedhet, csökkenhet, más késés, más sorrendben is megérkezhetnek.
- Megpróbálja az adatcsomagokat időben egymáshoz közel kiszállítani.
- Megpróbálja az átviteli közeget hatékonyan használni.

# A TCP evolúciója

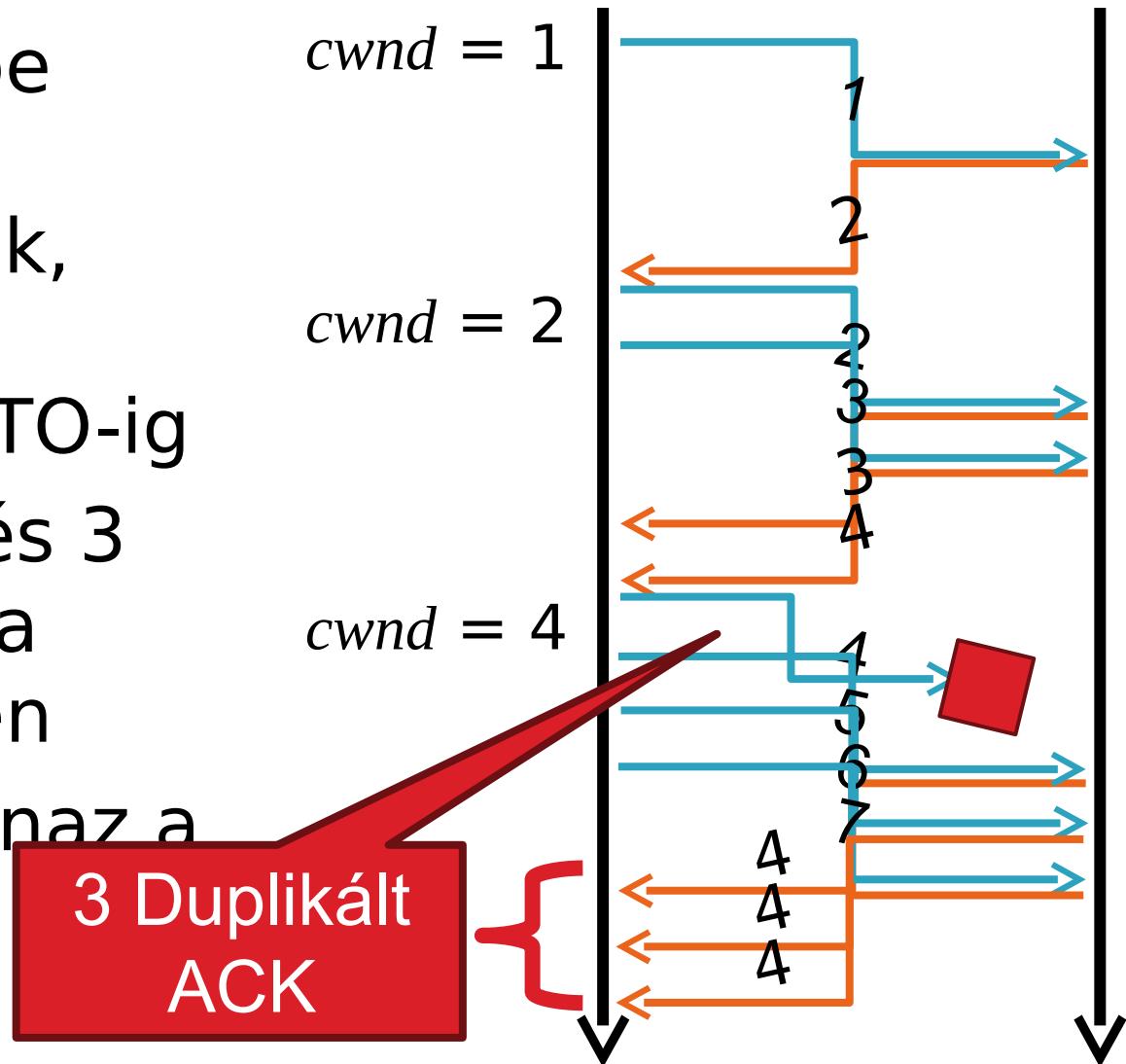
23

- Az eddigi megoldások a TCP Tahoe működéshez tartoztak
  - Eredeti TCP
- A TCP-t 1974-ben találták fel!
  - Napjainkba számos változata létezik
- Kezdeti népszerű változat: TCP Reno
  - Tahoe lehetőségei, plusz...
  - Gyors újraküldés (Fast retransmit)
    - 3 duplikált ACK? -> újraküldés (ne várunk az RTO-ra)

# TCP Reno: Gyors újraküldés

24

- Probléma: Tahoe esetén ha egy csomag elveszik, akkor hosszú a várakozás az RTO-ig
- Reno: újraküldés 3 duplikált nyugta fogadása esetén
- Duplikált: ugyanaz a sorszám
- Explicit jele a csomag elvesztésére

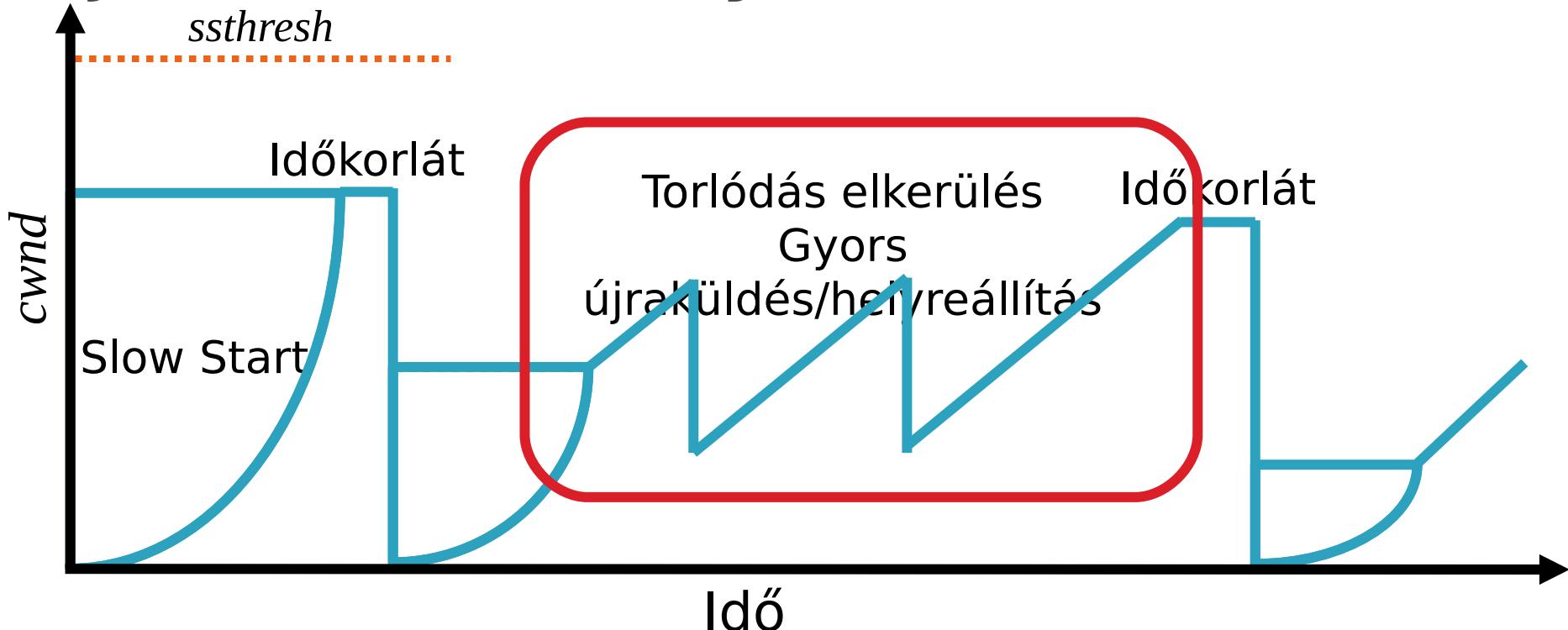


# TCP Reno: Gyors helyreállítás

25

- Gyors újraküldés után módosítjuk a torlódási ablakot:
  - $cwnd := cwnd/2$  (*valójában ez a Multiplicative Decrease*)
  - $ssthresh :=$  az új  $cwnd$
  - Azaz nem álltjuk vissza az eredeti 1-re a  $cwnd$ -t!!!
  - Ezzel elkerüljük a felesleges slow start fázisokat!
  - Elkerüljük a költséges időkorlátokat
- Azonban ha az RTO lejár, továbbra is  $cwnd =$

# Példa: Gyors újraküldés/helyreállítás



- Stabil állapotban, a cwnd az optimális ablakméret körül oszcillál
- TCP minden csomagdobásokat kényszerít

# Számos TCP változat...

27

- Tahoe: az eredeti
  - Slow start és AIMD
  - Dinamikus RTO, RTT becsléssel
- Reno:
  - fast retransmit (3 dupACKs)
  - fast recovery ( $cwnd = cwnd/2$  vesztés esetén)
- NewReno: javított gyors újraküldés
  - minden egyes duplikált ACK újraküldést vált ki
  - Probléma: >3 hibás sorrendben fogadott

# TCP a valóságban

28

- Mi a legnépszerűbb variáns napjainkban?
  - Probléma: TCP rosszul teljesít nagy késleltetés-sávszélesség szorzattal rendelkező hálózatokban (a modern Internet ilyen)
  - Compound TCP (Windows)
    - Reno alapú
    - Két torlódási ablak: késleltetés alapú és vesztés alapú
    - Azaz egy összetett torlódás vezérlést alkalmaz
  - TCP CUBIC (Linux)
    - Eszerint a TCP CUBIC (Piecewise Linear Connection Control)

# Nagy késleltetés-sávszélesség szorzat

29

□ Probléma: A TCP nem teljesít jól ha

(Delay-bandwidth product)

- A hálózat kapacitása (sávszélessége) nagy
- A késleltetés (RTT) nagy
- Vagy ezek szorzata nagy
  - $b * d =$  maximális szállítás alatt levő adatmennyiség
  - Ezt nevezzük késleltetés-sávszélesség szorzatnak
- Miért teljesít ekkor gyengén a TCP?
  - A slow start és az additive increase csak lassan konvergál
  - A TCP ACK ütemezett (azaz csak minden ACK)

# Célok

30

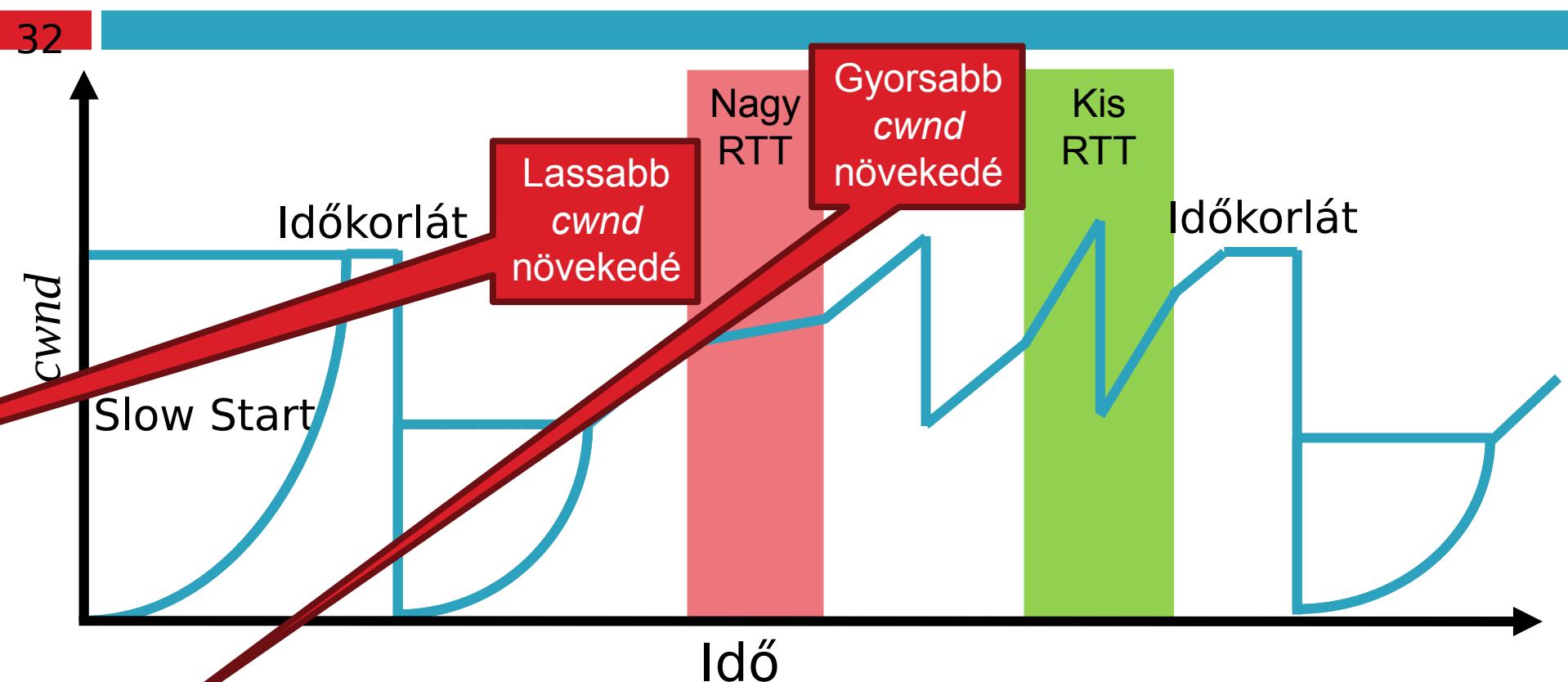
- A TCP ablak gyorsabb növelése
  - A slow start és az additive increase túl lassú, ha nagy a sávszélesség
  - Sokkal gyorsabb konvergencia kell
- Fairség biztosítása más TCP változatokkal szemben
  - Az ablak növelése nem lehet túl agresszív
- Javított RTT fairség
  - A TCP Tahoe/Reno folyamok nem adnak fair erőforrás-megosztást nagyon eltérő RTT-k

# Compound TCP

31

- Alap TCP implementáció Windows rendszereken
- Ötlet: osszuk a *torlódási ablakot* két különálló ablakba
  - Hagyományos, vesztés alapú ablak
  - Új, késleltetés alapú ablak
- $wnd = \min(cwnd + dwnd, adv\_wnd)$ 
  - *cwnd*-t az AIMD vezérli AIMD
  - *dwnd* a késleltetés alapú ablak

# Compound TCP példa



- Agresszívan reagál az RTT változására
- Előnyök: Gyors felfutás, sokkal fairebb viselkedés más folyamokkal szemben eltérő RTT esetén

# TCP CUBIC

33

- Alap TCP implementáció Linux rendszereken
- Az AIMD helyettesítése egy „köbös” (CUBIC) függvénnel

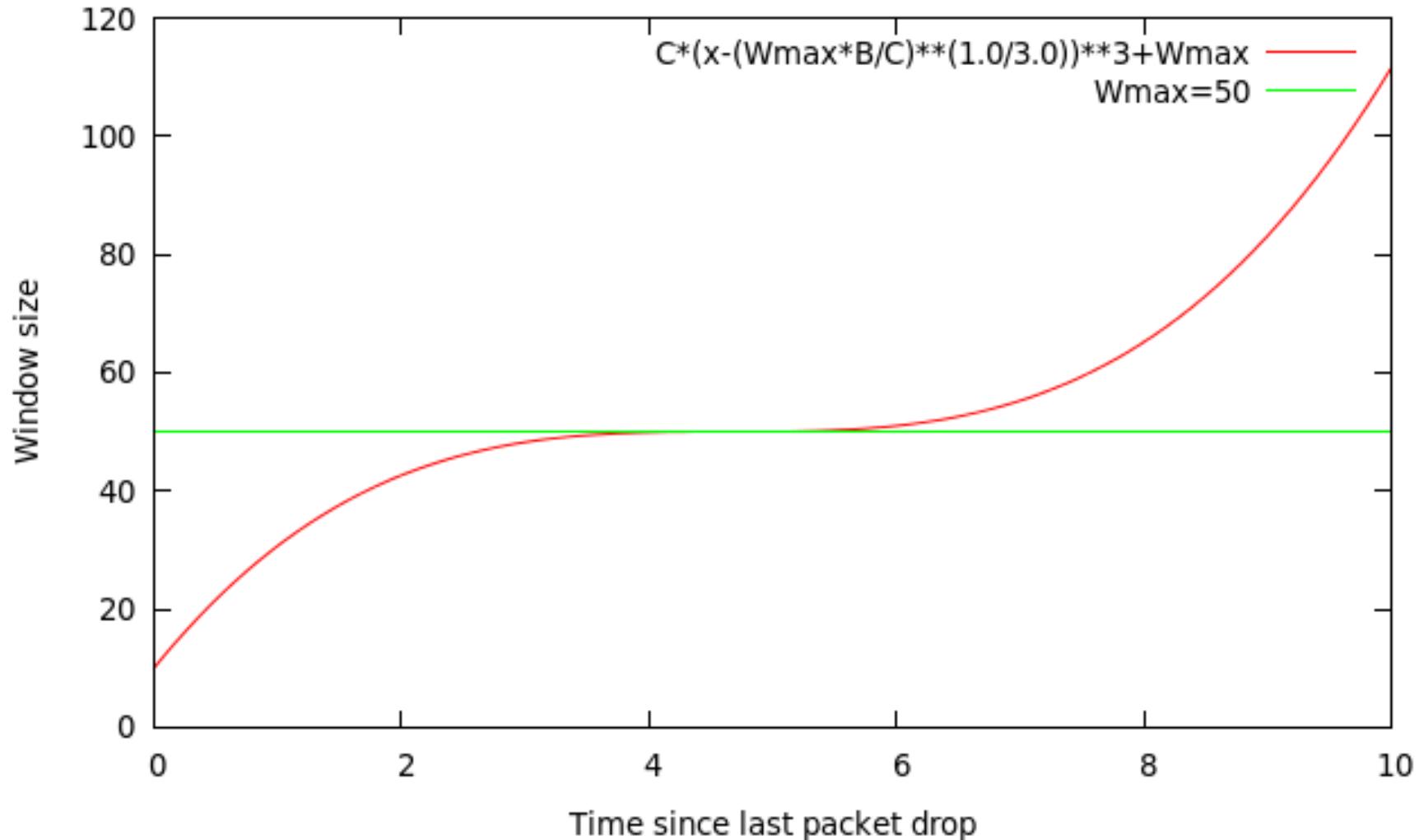
$$W_{cubic} = C(T - K)^3 + W_{max} \quad (1)$$

C is a scaling constant, and  $K = \sqrt[3]{\frac{W_{max}\beta}{C}}$

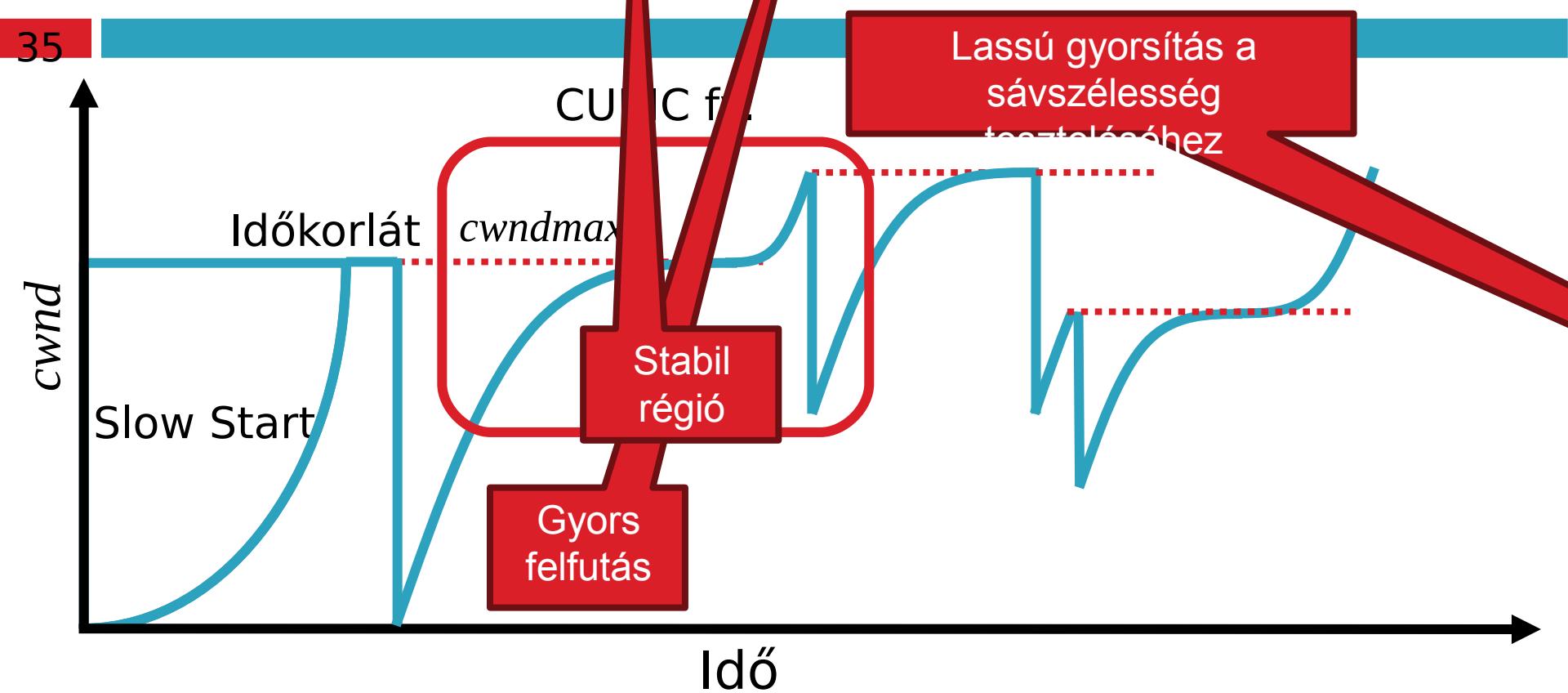
- B → egy konstans a multiplicative increase fázishoz
- T → eltelt idő a legutóbbi csomagvesztés óta
- W\_max → cwnd a legutolsó csomagvesztés idején

# TCP CUBIC

34



# TCP CUBIC példa



- Kevésbé pazarolja a sávszélességet a gyors felfutások miatt
- A stabil régió és a lassú gyorsítás segít a fairség biztosításában

# Problémák a TCP-vel

36

- Az Internetes forgalom jelentős része TCP
- Azonban számos probléma okozója is egyben
  - Gyenge teljesítmény kis folyamok esetén
  - Gyenge teljesítmény wireless hálózatokban
  - DoS támadási felület

# Kis folyamok (flows)

37

- Probléma: kis folyamok esetén torz viselkedés
  - 1 RTT szükséges a kapcsolat felépítésére (SYN, SYN/ACK)
    - pazarló
  - $cwnd$  mindenkorban 1-gyel indul
    - Nincs lehetőség felgyorsulni a kevés adat miatt
- Az Internetes forgalom nagy része kis folyam
  - Többnyire HTTP átvitel, <100KB

# Wireless hálózatok

# Szolgáltatás megtagadása

## Denial of Service (DoS)

39

- Probléma: a TCP kapcsolatok állapottal rendelkeznek
  - A SYN csomagok erőforrásokat foglalnak az szerveren
  - Az állapot legalább néhány percig fennmarad (RTO)
- SYN flood: elég sok SYN csomag küldése a szervernek ahhoz, hogy elfogyjon a memória és összeomoljon a kernel
- Megoldás: SYN cookie-k

# Kitekintés

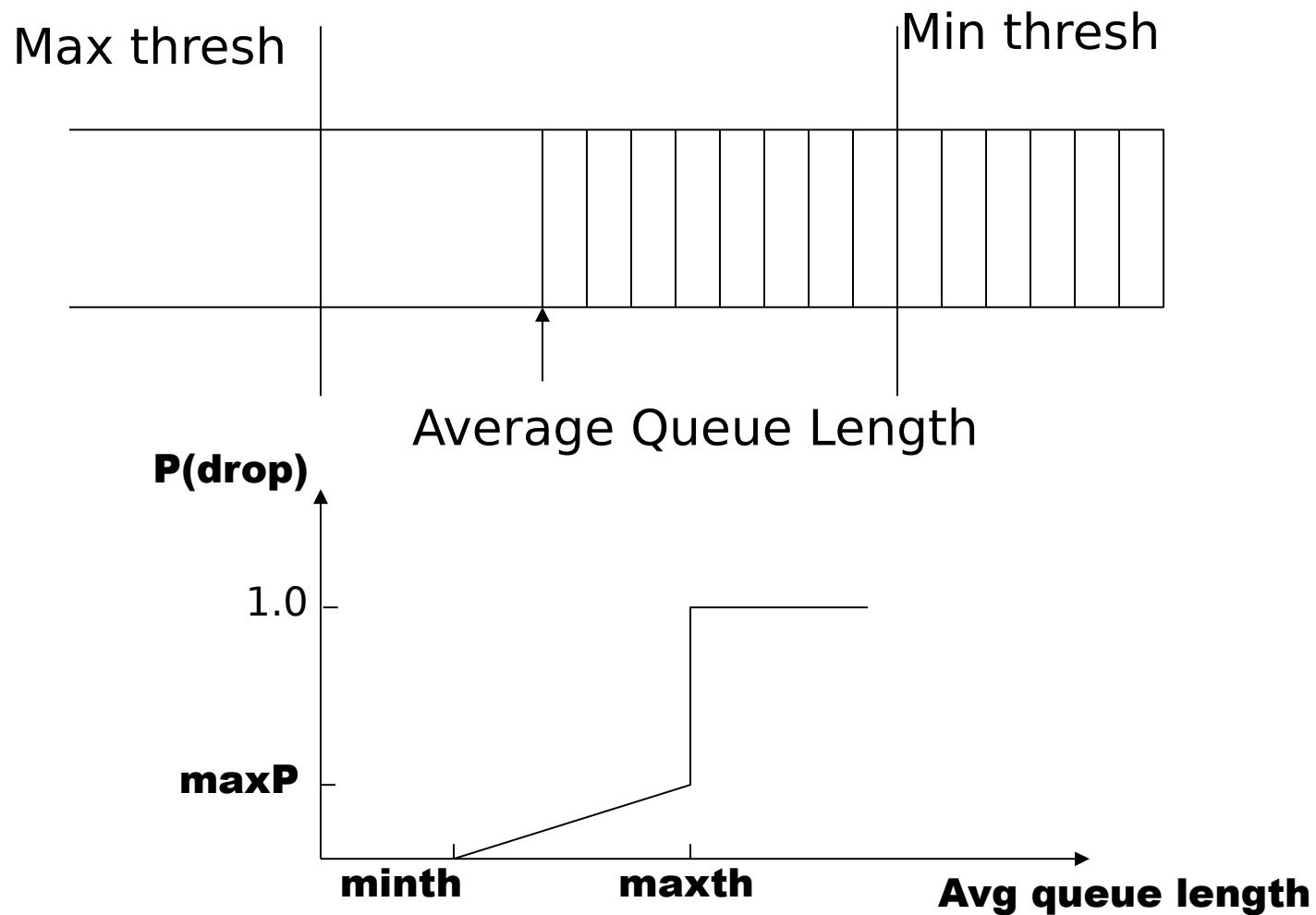
# Typical Internet Queuing

- FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet
- FIFO (first-in-first-out)
  - Implies single class of traffic
- Drop-tail
  - Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
  - FIFO: scheduling discipline
  - Drop-tail: queueing discipline

# RED Algorithm

- Maintain running average of queue length
- If  $\text{avgq} < \text{minth}$  do nothing
  - Low queuing, send packets through
- If  $\text{avgq} > \text{maxth}$ , drop packet
  - Protection from misbehaving sources
- Else mark packet in a manner proportional to queue length
  - Notify sources of incipient congestion
  - E.g. by ECN IP field or dropping packets with a given probability

# RED Operation



# RED Algorithm

- Maintain running average of queue length
- For each packet arrival
  - Calculate average queue size (avg)
  - If  $\text{minth} \leq \text{avgq} < \text{maxth}$ 
    - Calculate probability  $P_a$
    - With probability  $P_a$ 
      - Mark the arriving packet: drop or set-up ECN
    - Else if  $\text{maxth} \leq \text{avg}$ 
      - Mark the arriving packet: drop, ECN

# Data Center TCP: DCTCP

# Generality of Partition/Aggregate

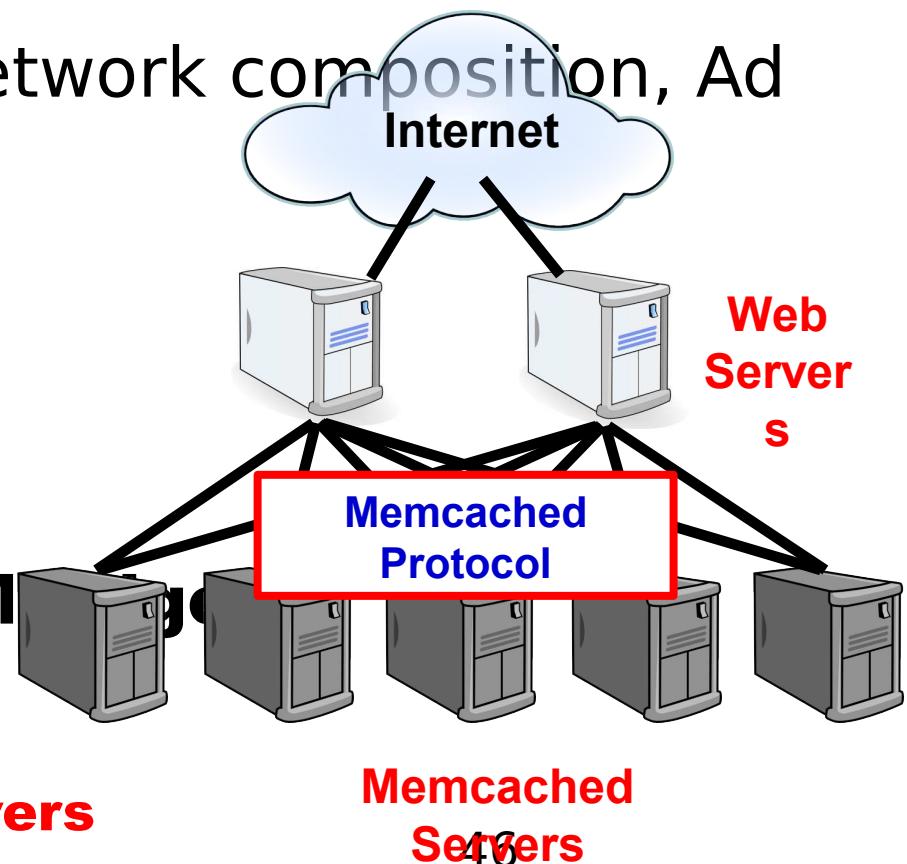
- The foundation for many large-scale web applications.

- Web search, Social network composition, Ad selection, etc.

- Example: **Facebook**

## Partition/Aggregate ~ Memcached

- Aggregators: **Web Servers**
  - Workers: **Memcached Servers**



Memcached  
Servers  
40

# Workloads

47

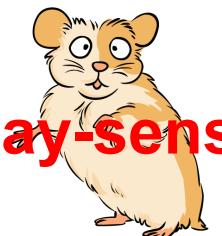
- Partition/Aggregate  
**(Query)**

→ Delay-sensitive



- Short messages [50KB-1MB]  
**(Coordination, Control state)**

→ Delay-sensitive



- Large flows [1MB-50MB]  
**(Data update)**

→ Throughput-sensitive



# Impairments

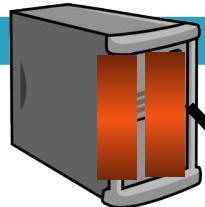
48

- Incast
- Queue Buildup
- Buffer Pressure

# Incast

49

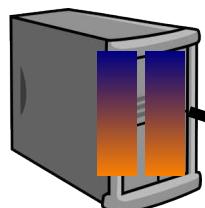
**Worker  
1**



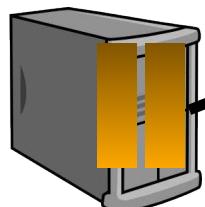
- Synchronized mice collide.

➤ **Caused by  
Partition/Aggregate.**

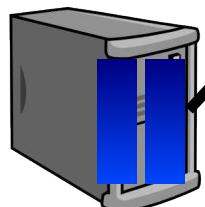
**Worker  
2**



**Worker  
3**

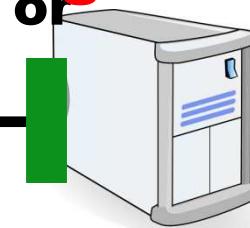


**Worker  
4**



← **TCP  
timeout**

Aggregate  
or

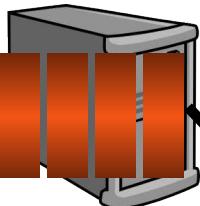


$RTT_{min} = 300 \text{ ms}$



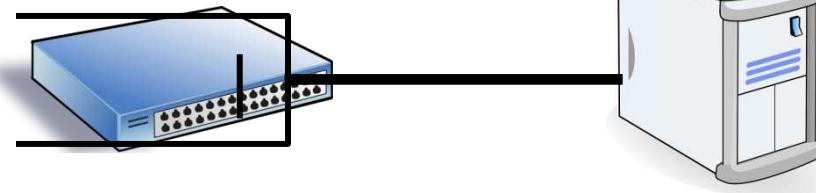
# Queue Buildup

Sender 1

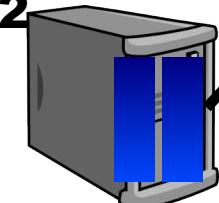


- Big flows buildup queues.
  - **Increased latency for short flows.**

Receiver



Sender 2



- Measurements in Bing cluster
  - **For 90% packets: RTT < 1ms**
  - **For 10% packets: 1ms <**

# Data Center Transport Requirements

51

## 1. **High Burst Tolerance**

- Incast due to Partition/Aggregate is common.

## 2. **Low Latency**

- Short flows, queries

## 3. **High Throughput**

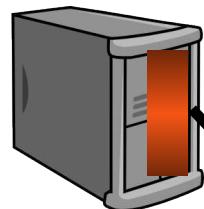
- Continuous data updates, large file transfers

**The challenge is to achieve these three together.**

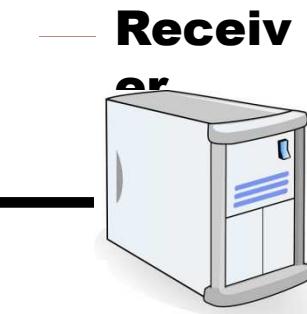
# DCTCP: The TCP/ECN Control Loop

Sender 1

**ECN = Explicit Congestion Notification**



**ECN Mark (1  
bit)**



Sender

2

# DCTCP: Two Key Ideas

18

- React in proportion to the **extent** of congestion, not its **presence**.

✓ Reduces **variance** in sending rates, lowering queuing

ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by <b>50%</b>	Cut window by <b>40%</b>
0 0 0 0 0 0 0 0 1	Cut window by <b>50%</b>	Cut window by <b>5%</b>

# Data Center TCP Algorithm

19

## Switch side:

- Mark packets when **Queue Length**

**> K.**

## Sender side:

- Maintain running average of **fraction** of packets marked ( $\alpha$ ).



$$F = \frac{\# \text{ of marked ACKs}}{\text{Total } \# \text{ of ACKs}} \quad \text{In each RTT:} \quad \alpha \leftarrow (1 - g)\alpha + gF$$

$$Cwnd \leftarrow (1 - \frac{\alpha}{2})Cwnd$$

## Adaptive window decreases:

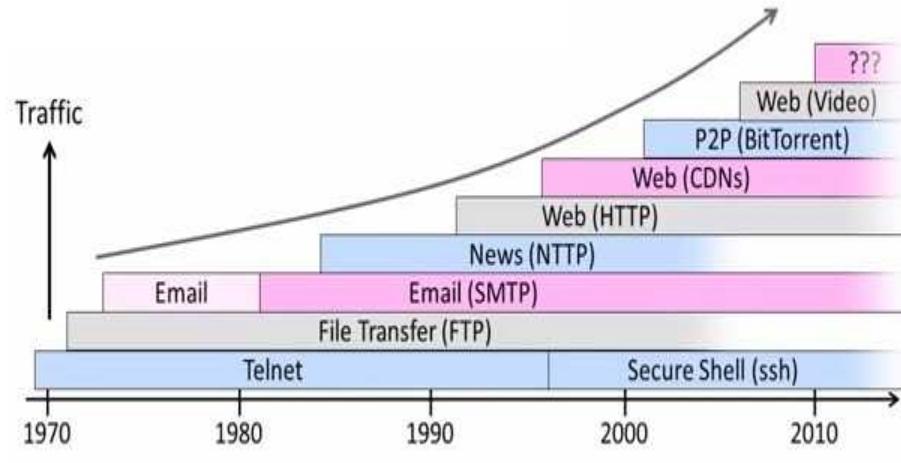
- Note: decrease factor between 1 and 2.

# Számítógépes Hálózatok

**12-13. Előadás:  
Alkalmazási réteg  
DNS, HTTP, CDNs**

Based on slides from

# Internetes alkalmazások evolúciója



Forrás: [1]

- Folyamatosan változik, növekszik...
  - [www.evolutionoftheweb.com](http://www.evolutionoftheweb.com)

3

DNS

# „8. réteg” (A szénalapú 4 csomópontok)

- Ha...
  - Fel szeretnél hívni valakit, akkor el kell kérned a telefonszámát
    - Nem hívhatod csak úgy “P I S T Á T”
  - Levelet küldenél valakinek, akkor szükséged van a címére
- Mi a helyzet az Internettel?
  - Ha el akarod érni a Google-t, szükséges annak IP címe
  - Tudja valaki a Google IP címét???

# Internetes nevek és címek

5

- Címek, pl. 129.10.117.100
  - Számítógépek által használt címkék a gépek azonosítására
  - A hálózat szerkezetét tükrözi
- Nevek, pl. www.northeastern.edu
  - Ember számára értelmes címkék a gépeknek
  - A szervezeti struktúrát tükrözi
- Hogyan képezzünk az egyikről a másikra?
  - Domain Name System (DNS)

# Réges régen...

6

- A DNS előtt minden név-IP leképezés egy *hosts.txt*-ben volt
  - /etc/hosts - Linuxon
  - C:\Windows\System32\drivers\etc\hosts - Windowson
- Központosított, manuális rendszer
  - A változásokat emailben kellett beküldeni a SRI-nek
    - SRI=Stanford Research Institute
  - A gépek periodikus időközönként letöltötték (FTP) a *hosts.txt* fájlt

# A DNS felé

7

- Végül a *hosts.txt* alapú rendszer szétesett
  - Nem skálázható, SRI nem bírt a terheléssel/igényekkel
  - Nehéz volt a nevek egyediségének biztosítása
    - Pl. MIT
      - Massachusetts Institute of Technology?
      - Melbourne Institute of Technology?
  - Számos gép rendelkezett nem naprakész *hosts.txt*-vel
- Ez vezetett a DNS megszületéséhez...

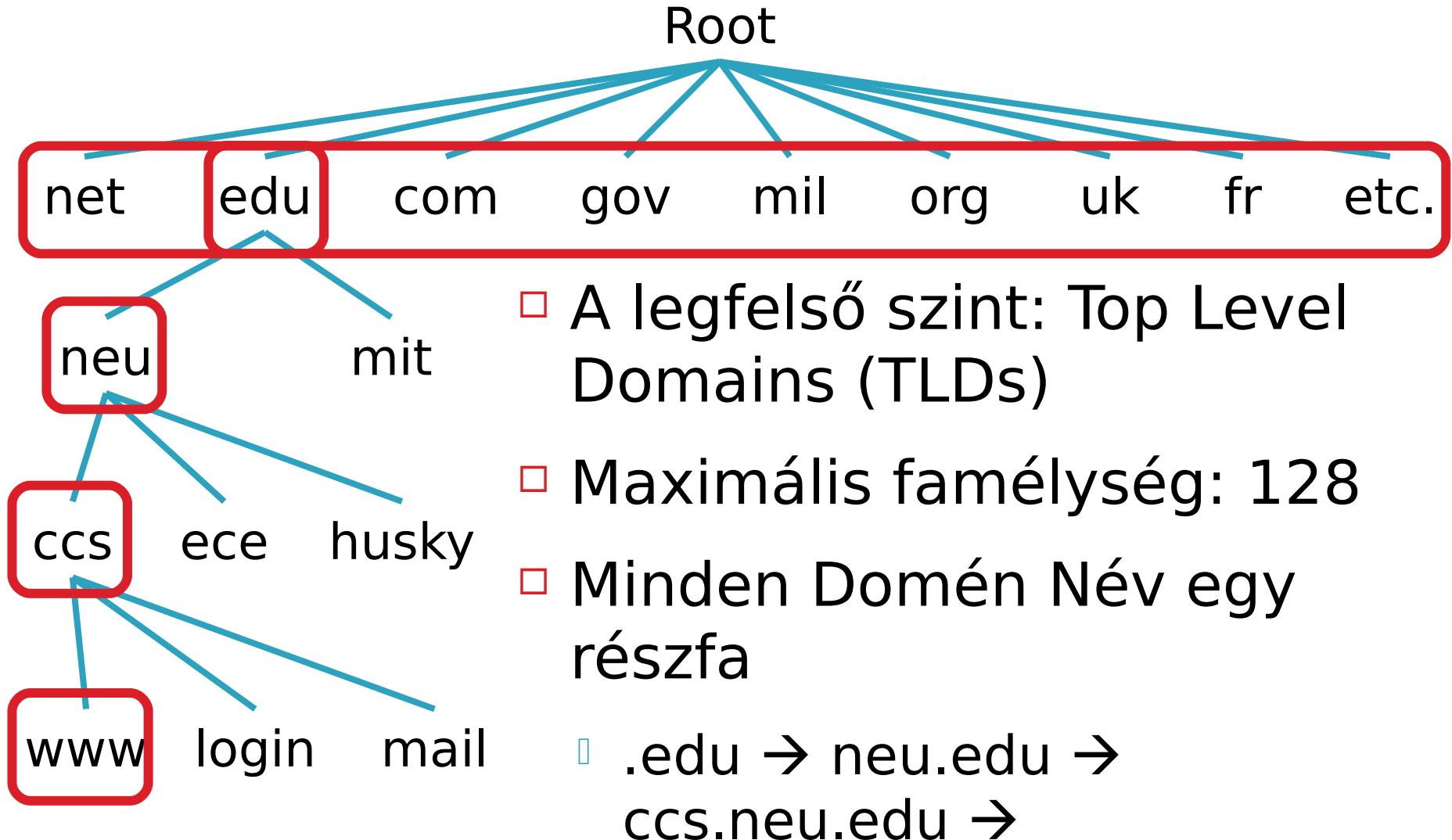
# DNS általánosságban

8

- Domain Name System
- Elosztott adatbázis
  - Nem központosított
- Egyszerű kliens-szerver architektúra
  - UDP 53-as port, vannak TCP implementációk is
  - Rövid kérések - rövid válaszok; kérés-válasz típusú kommunikáció
- Hierarchikus névtér
  - Szemben a hosts.txt alapú flat megoldással

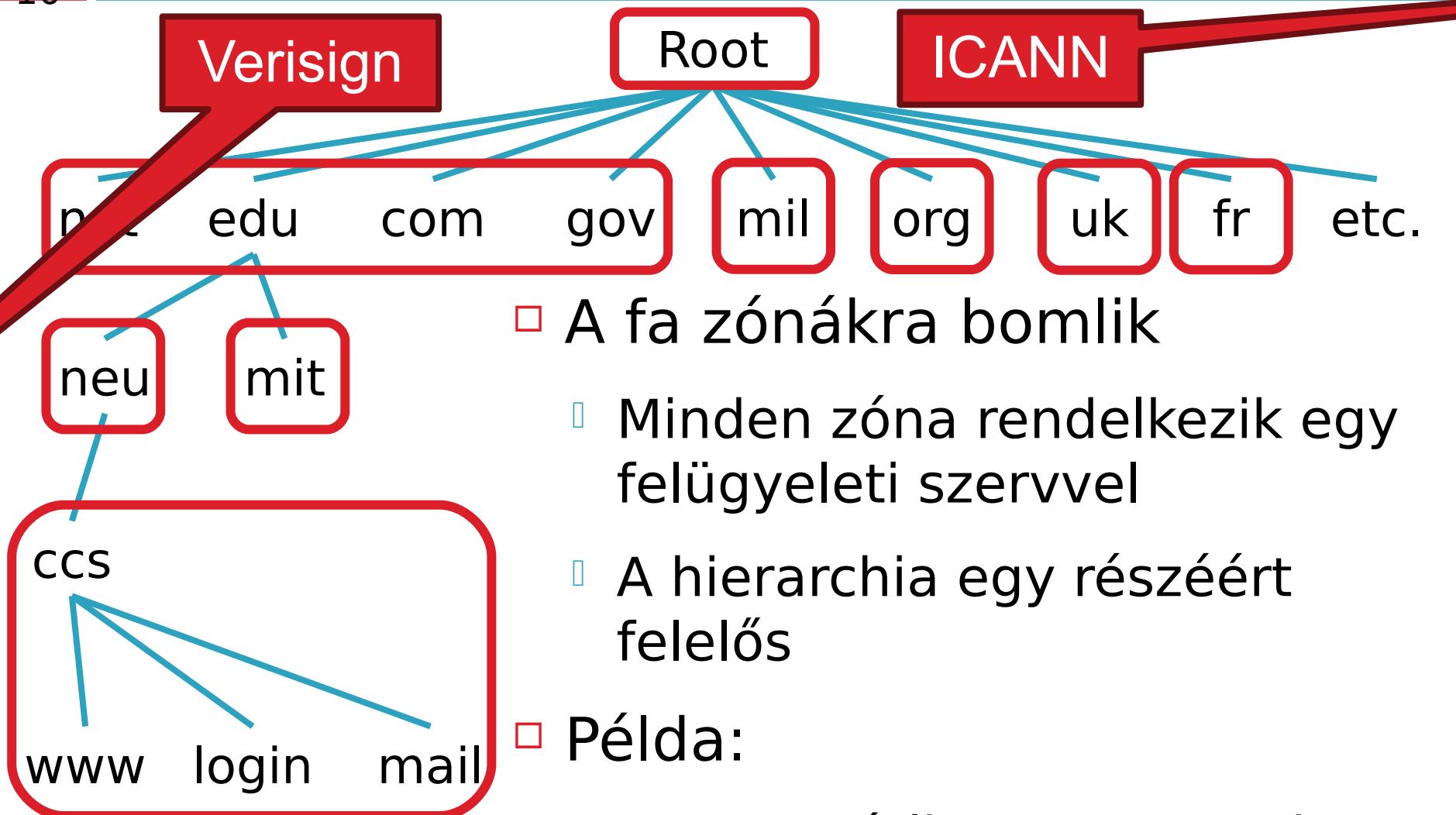
# Név hierarchia

9



# Hierarchikus adminisztráció

10



# Szerver hierarchia

11

- Egy DNS szerver funkciói:
  - A hierarchia egy részét felügyeli
    - Nem szükséges minden DNS nevet tárolnia
  - A zónájához tartozó összes hoszt és domén rekordjainak tárolása
    - Másolatok lehetnek a robosztusság növelés végett
  - Ismeri a root szerverek címét
    - Ismeretlen nevek feloldása miatt kell
- A root szerverek minden TLD-t ismernek
  - Azaz innen indulva fel lehet tárni

# *Top Level Domains*

- Internet Corp. Assigned Names and Numbers (1998)
- 22+ **általános TLDs** létezik
  - klasszikusok: *.com, .edu, .gov, .mil, .org, .net*
  - később keletkeztek: *.aero, .museum, .xxx*
- ~250 TLDs a különböző **ország kódoknak**
  - Két betű (mint például *.au, .hu*), 2010-től plusz nemzetközi karakterek (például *kínai*)
  - Több elüzletisedett, például a *.tv* (*Tuvalu*)
  - Példa domén hack-ekre: *instagr.am* (Örményország), *goo.gl* (Grönland)

# Root Name Servers

13

## □ A Root Zone Fájlért felelős

- Listát vezet a TLD-kről és arról, hogy ki felügyeli őket.
- ~272KB a fájl mérete
- Pl. bejegyzése:

com. 172800 IN NS a.gtld-servers.net.

com. 172800 IN NS b.gtld-servers.net.

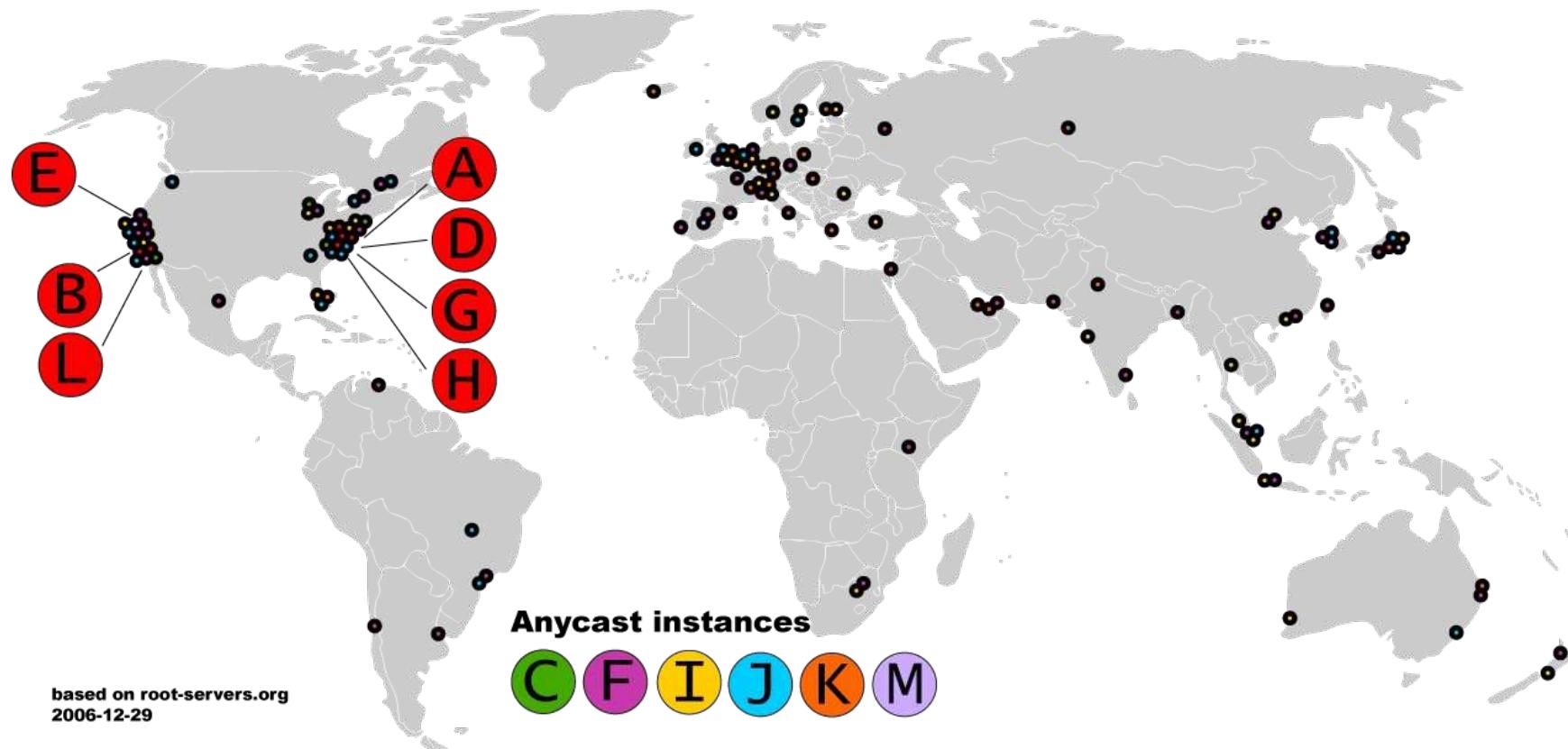
com. 172800 IN NS c.gtld-servers.net.

## □ Az ICANN adminisztrálja

- 13 root szerver címekkel: A→M

# Map of the Roots

14



# Lokális névszerverek

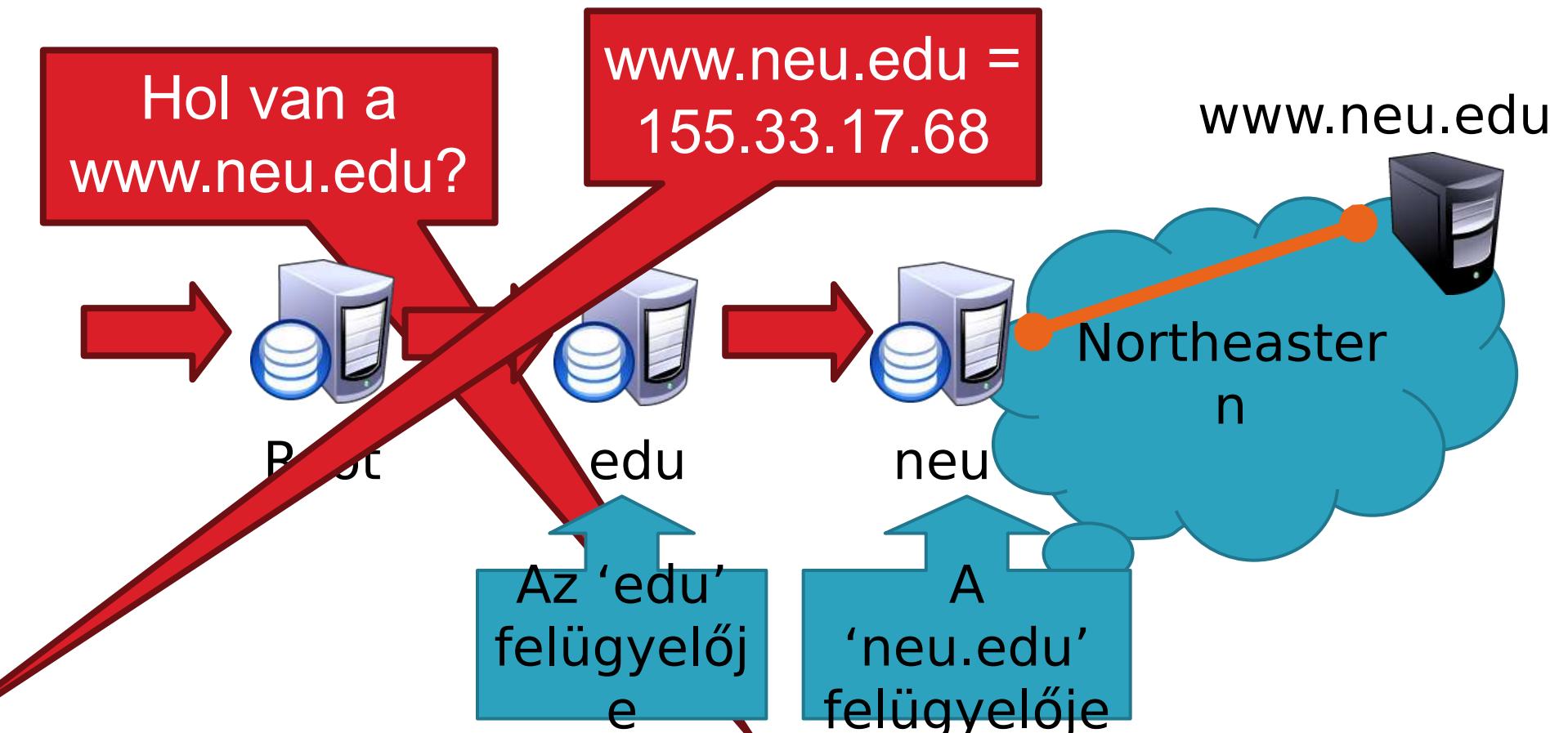
15



- minden ISP/cég rendelkezik egy lokális, default névszerverrel
- Gyakran a DHCP konfigurálja f
- A DNS lekérdezések a lokális névszervernél kezdődnek

# Authoratív Névszerverek

16



- név → IP leképezéseket tárolja egy adott hoszthoz

# Egyszerű doménnév feloldás

17

- minden hoszt ismer egy lokális DNS szervet
  - minden kérést ennek küld
- Ha a lokális DNS szerver tud válaszolni, akkor kész...
  1. A lokális szerver a felügyelő szerver az adott névhez
  2. A lokális szerver cache-ében van rekord a keresett névhez
- Különben menjünk végig a teljes hierarchián felülről lefelé egészen a keresett név felülről lefelé egészen a keresett név

# Lekérdezések

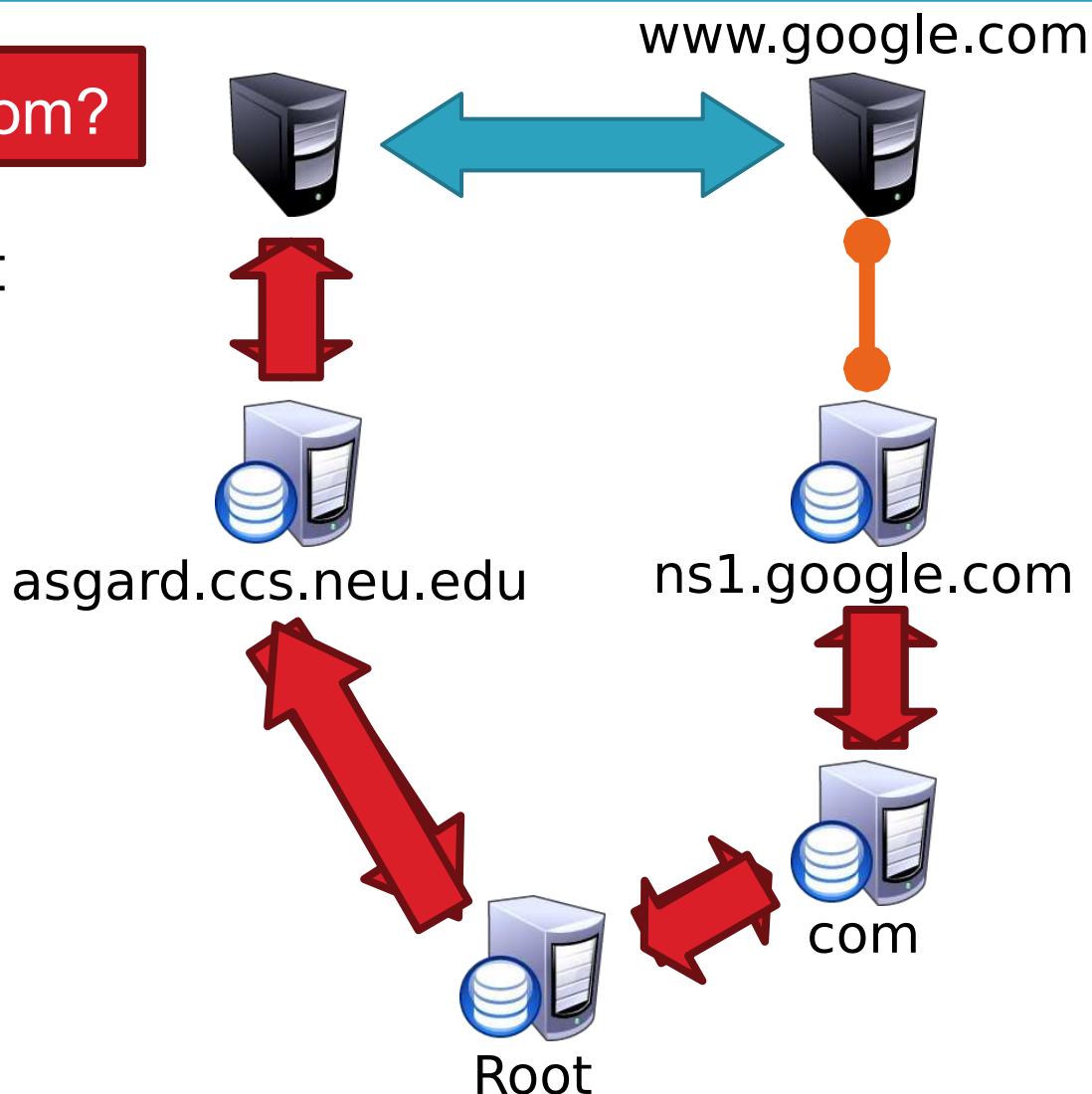
- A lekérdezésnek két fajtája van:
  - *Rekurzív lekérdezés* – Ha a névszerver végzi el a névfeloldást, és tér vissza a válasszal.
  - *Iteratív lekérdezés* – Ha a névszerver adja vissza a választ vagy legalább azt, hogy kitől kapható meg a következő válasz.
- Melyik a jobb?
  - *Rekurzív jellemzői*
    - Lehetővé teszi a szervernek a kliens terhelés kihelyezését a kezelhetőségért.
    - Lehetővé teszi a szervernek, hogy a kliensek egy csoportja felett végezzen *cachelést*, a jobb teljesítményért.
  - *Iteratív jellemzői*
    - Válasz után nem kell semmit tenni a kéréssel a névszervernek.

# Rekurzív DNS lekérdezés

19

Hol van a www.google.com?

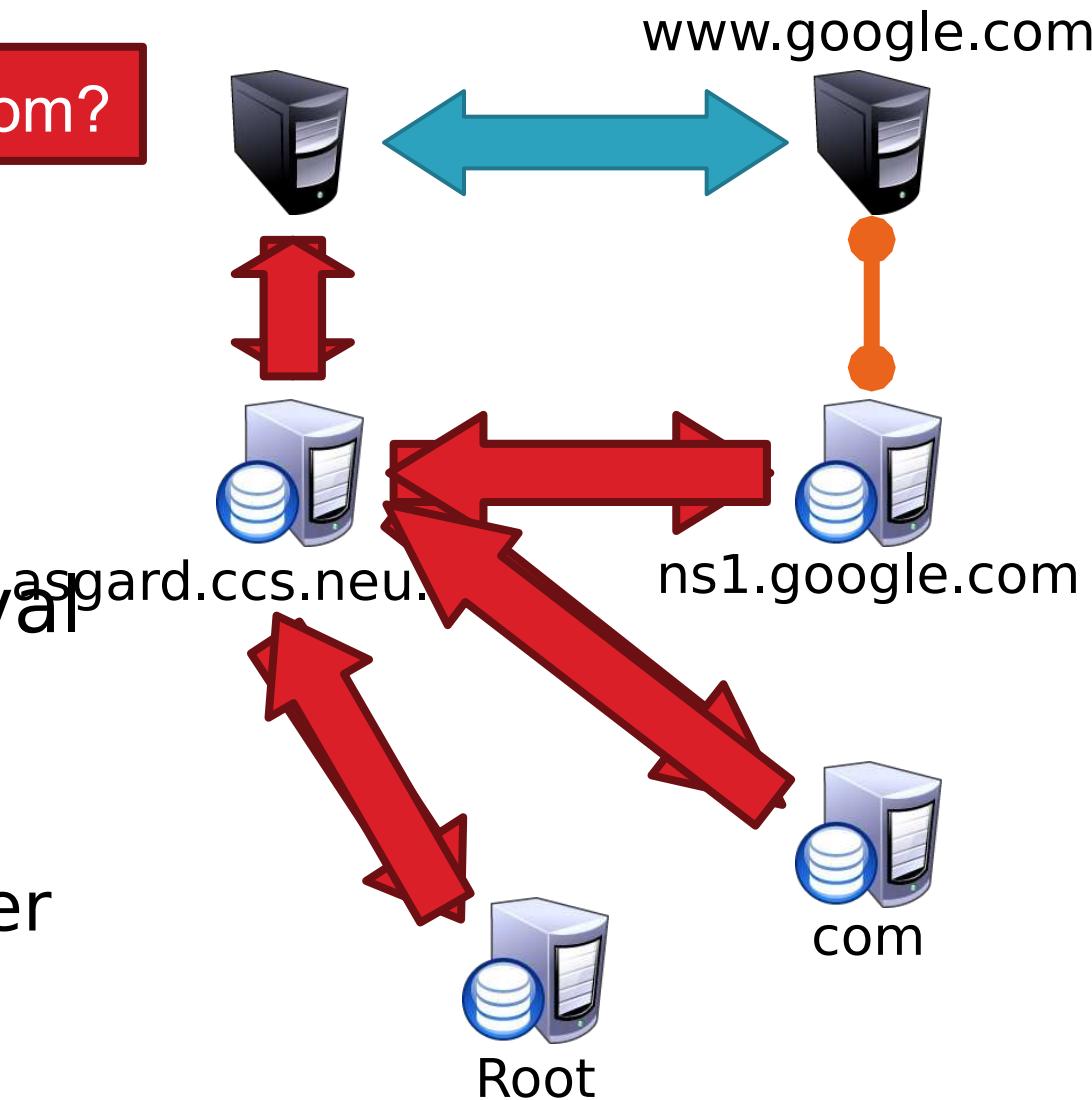
- A lokális szerver terhet rak a kérdezett névszerverre (pl. root)
- Honnan tudja a kérdezett, hogy kinek továbbítsa a választ?
  - Random ID a DNS lekérdezésben



# Iteratív DNS lekérdezés

20

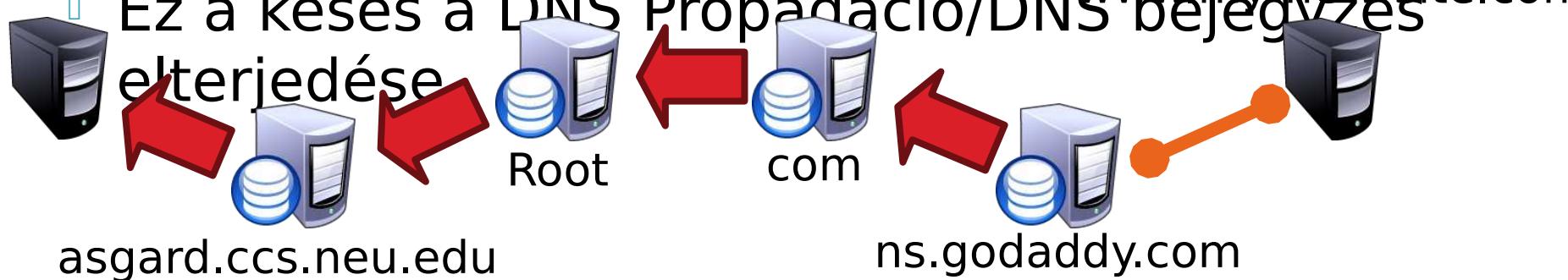
Hol van a www.google.com?



- A szerver mindenkor a következő kérdezendő névszerver adataival tér vissza
  - “I don’t know this name, but this other server might”

# DNS bejegyzés elterjedése

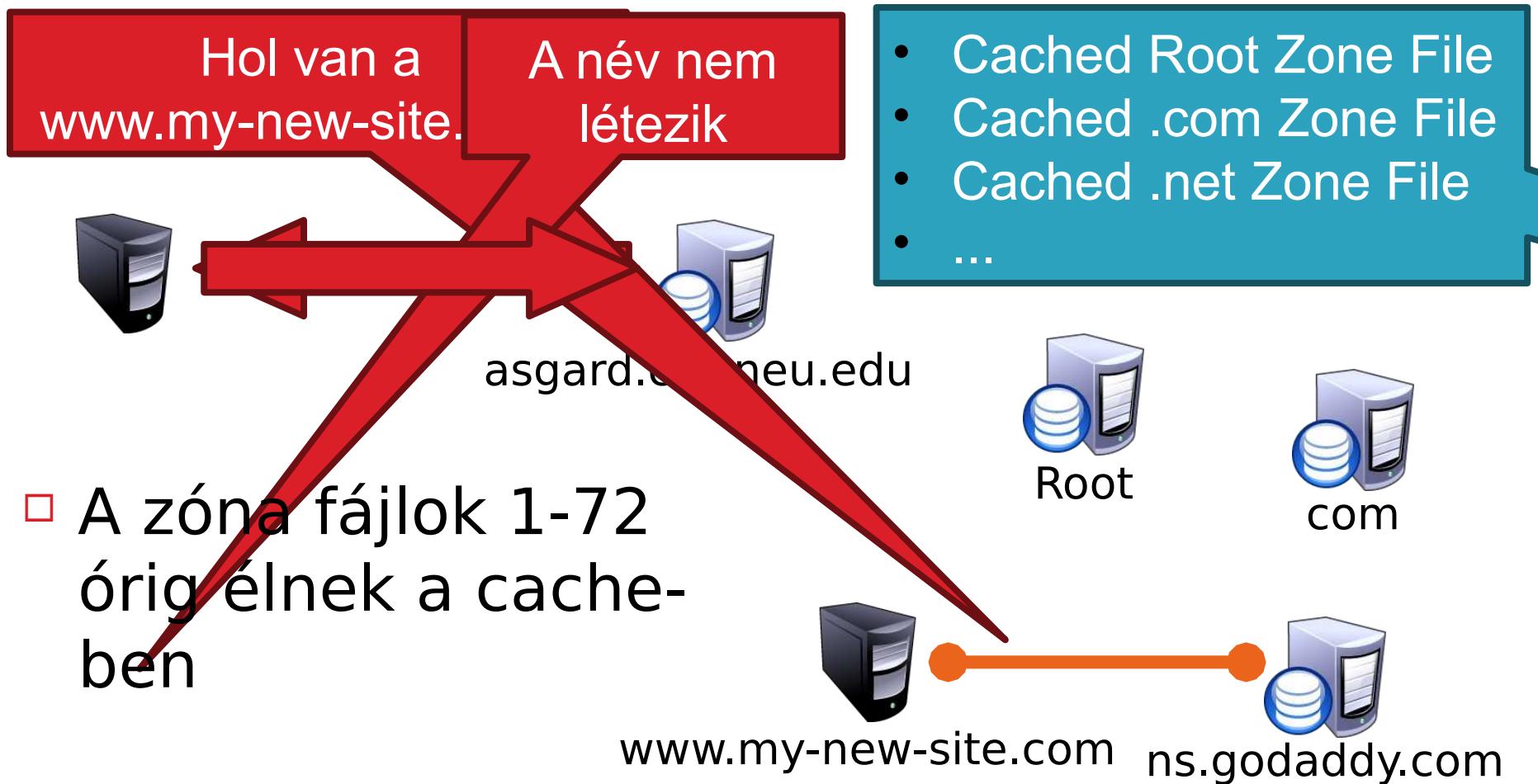
21

- Van-e a teremben olyan, aki vásárolt már domén nevet?
    - Észrevettétek-e, hogy kb. 72 óra kell ahhoz, hogy elérhető legyen a bejegyzés után?
    - Ez a késés a DNS Propagáció/DNS bejegyzés elterjedése
  - Miért nem sikerül ez egy új DNS név esetén?
- 

# Cachelés VS frissesség

22

- DNS elterjedés késését a cache okozza



# DNS Erőforrás rekordok (Resource Records)

23

- A DNS lekérdezéseknek két mezőjük van
  - name és type
- Az erőforrás rekord válasz egy DNS lekérdezésre
  - Négy mezőből áll: (name, value, type, TTL)
  - Egy lekérdezésre adott válaszban több rekord is szerpelhet
- Mit jelent a name és a value mező?
  - Ez a lekérdezés típusától (type) függ

# DNS lekérdezés típusok

24

- Type = A / AAAA
  - Name = domén név
  - Value = IP cím
  - A = IPv4, AAAA = IPv6

Query Name:  
www.ccs.neu.edu  
Type: A

Resp. Name:  
www.ccs.neu.edu  
Value: 129.10.116.81

- Type = NS
  - Name = rész domén
  - Value = a rész doménhez tartozó DNS szerver neve

Query Name: ccs.neu.edu  
Type: NS

Resp. Name: ccs.neu.edu  
Value: 129.10.116.51

# DNS lekérdezés típusok

25

- ❑ Type = CNAME

- ❑ Name = domén név
  - ❑ Value = kanonikus név

- ❑ Alias nevek használatához
  - ❑ CDN használja

- ❑ Type = MX

- ❑ Name = emailben

Query Name: foo.mysite.com  
Type: CNAME

Resp. Name: foo.mysite.com  
Value: bar.mysite.com

Query Name: ccs.neu.edu  
Type: MX

Resp. Name: ccs.neu.edu  
Value:  
ember.ccs.neu.edu

# Fordított lekérdezés (PTR rekord)

- Mi a helyzet az IP→név leképezéssel?
- Külön hierarchia tárolja ezeket a leképezéseket
  - Gyökér pont: in-addr.arpa és ip6.arpa
- DNS rekord típusa ([type](#)): PTR
  - Name = IP cím
  - Value = domén név
- Nincs garancia arra, hogy minden IP címre működik

Query    Name: 129.10.116.51  
            Type: PTR

Resp.    Name: 129.10.116.51  
            Value: ccs.neu.edu

# DNS as Indirection Service

27

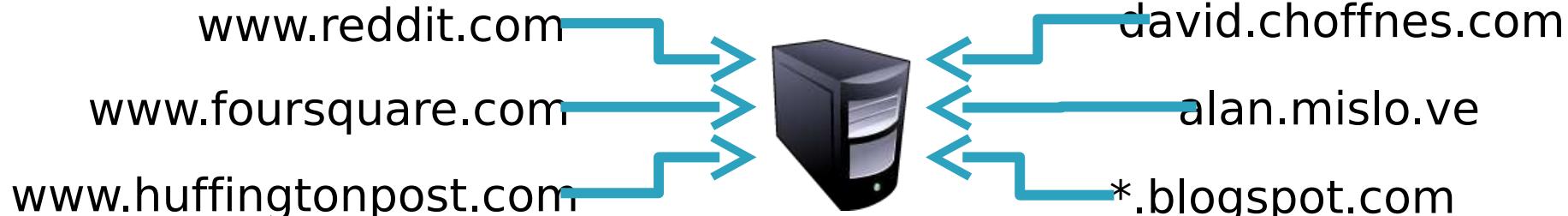
- DNS számos lehetőséget biztosít
  - Nem csak a gépekre való hivatkozást könnyíti meg!
- Egy gép IP címének lecserélése is triviális
  - Pl. a web szervert átköltöztetjük egy új hosztra
  - Csak a DNS rekord bejegyzést kell megváltoztatni!

# Aliasing/Kanonikus nevek és Load Balancing/Terhelés

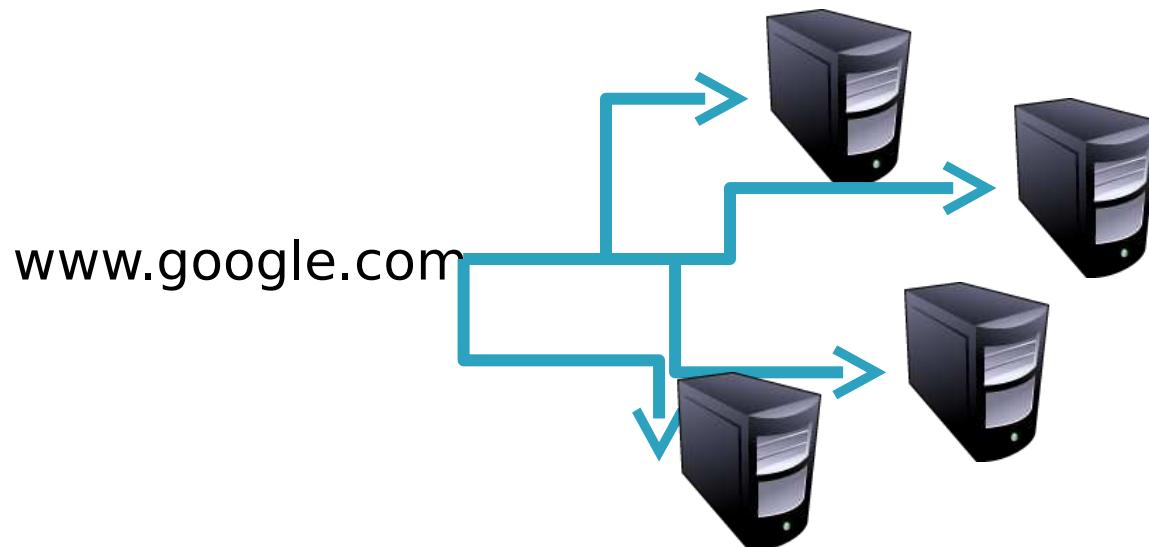
28

- Egy gépnak számos alias neve lehet

elosztás

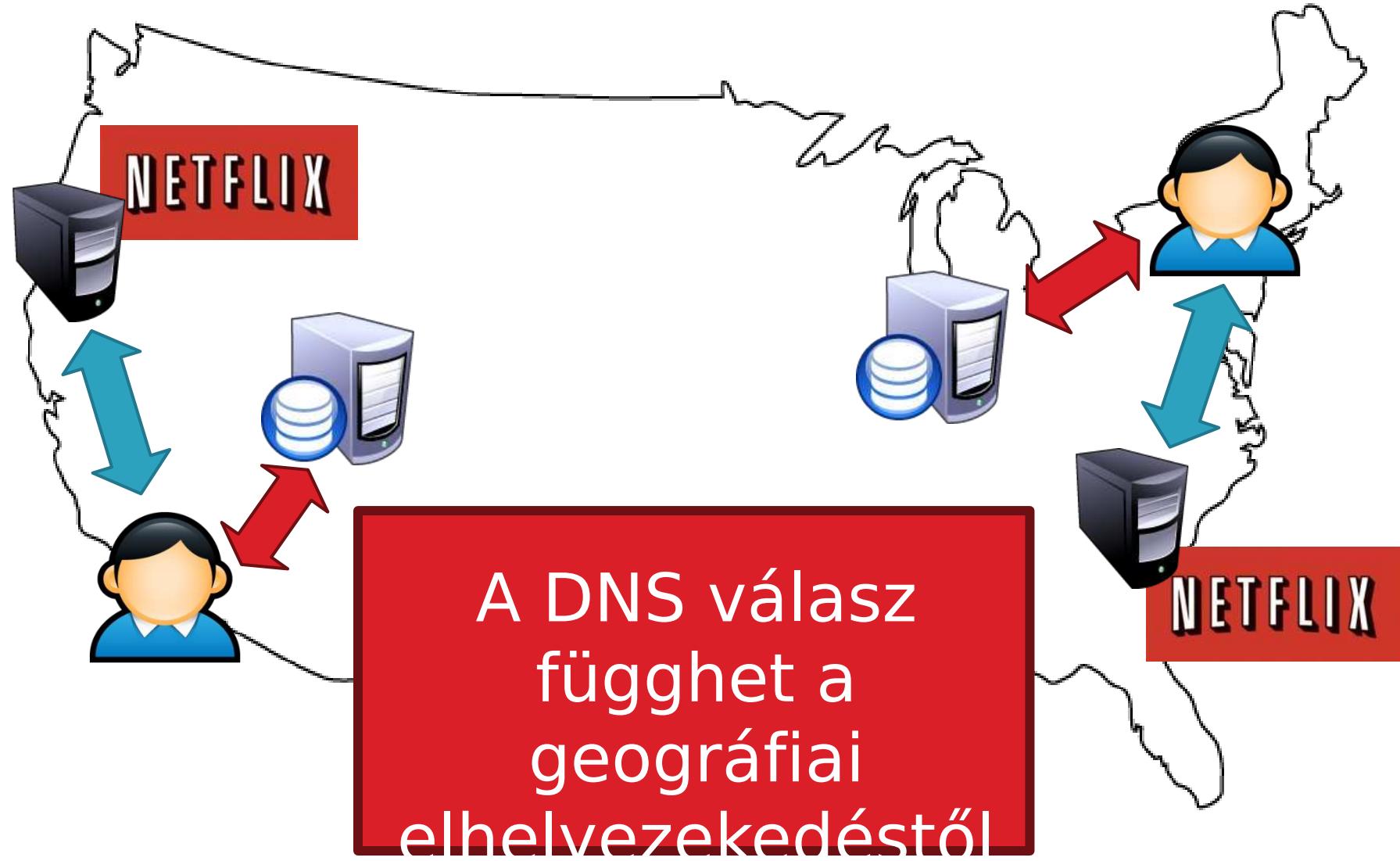


- Egy domén névhez számos IP cím tartozhat



# Content Delivery Networks

29



# A DNS fontossága

30

- DNS nélkül...
  - Hogyan találjunk meg egy weboldalt?
- Példa: a mailszervered azonosít
  - Email címet adunk meg weboldalakra való feliratkozásnál
  - Mi van, ha valaki eltéríti a DNS bejegyzést a mailszerveredhez?
- DNS a bizalom forrása a weben
  - Amikor a felhasználó begépeli a [www.bankofamerica.com](http://www.bankofamerica.com) címet, azt várja, hogy

# Denial Of Service (DoS)

31

- DNS szerverek túlterhelése lekérdezésekkel, amíg össze nem omlanak
- 2002 Október: masszív DDoS támadás a root szerverek ellen
  - Mi volt a hatása?
  - ... a felhasználók észre se vették
  - Root zónák mindenhol cache-elve vannak
- Célzottabb támadás hatékonyabb lenne
  - Lokális DNS szerver → elérhetetlen DNS
  - Authoritative szerver → elérhetetlen domain

# DNS Hijacking (eltérítés)

32

- OS vagy browser megfertőzése (virus/trojan)
  - Pl. Számos trójai megváltoztatja a bejegyzéseket a /etc/hosts fájlnan
    - \*.bankofamerica.com → evilbank.com
- Man-in-the-middle
  - 
- Válasz hamisítás (spoofing)
  - Kérések lehallgatása
  - Válaszok megversenyeztetése

D

Hol van a  
bankofamerica.com?

123.45.67.89

33



Honnan tudjuk, hogy a  
név→IP leképezés helyes?

bankofAmerica

Hol van a  
bankofamerica.com?

66.66.66.93

23.45.67.89



dns.evil.co



66.66.66.93

D

W

Hol van a

bisonina

Hol van a  
bankofamerica.com?

www.google.com =  
74.125.131.26

34



dns.neu.edu



ns1.google.com



- Amíg TTL lejár, a BofA összes szolgáltatója, amit a dns.neu.edu-nak küld, hamis/fertőzött válasszal téged.
- Sokkal rosszabb, mint a spoofing/man-in-the-middle

# Hogyan éri el a támadó a fertőzött bejegyzés tárolását?

35

- 1. Azt mondjuk a feloldónak, hogy az áldozathoz tartozó NS a támadó IP-jén érhető el

Áltó lekérdezés:

main.attacker.example IN A

■ Válasz:

- Válasz: (nincs válasz a lekérdezésre), de

A támadó azt mondja, „hogy a doménem authoratív szervere a ns.target.example és mellesleg itt van az IP-je”...

# Hogyan éri el a támadó a fertőzött bejegyzés tárolását?

NS rekord átirányítása a támadó névébe

Előző lekérdezés:

ns.attacker.example IN A

Válasz (nincs válasz)

Autoritás válasz:

Tárolás: 3600 IN NS ns.attacker.example.

A támadó nem releváns információt szűr be, melyet a szerver el fog tárolni a cacheben...

# Megoldás: DNSSEC

37

- Kritikus rekordokat kriptografikus aláírással látjuk el
  - A feloldó ellenőrzi az
- Két új erőforrástípus
  - Type = DNSKEY
    - Name = Zóna domén
    - Value = A zóna publikus kulcsa
  - Type = RRSIG
    - Name = (type, name) páros, pl. a lekérdezés maga
    - Value = A lekérdezés eredményének kriptografikus

Bizalmi hierarchiát hoz létre a zónák között

Megoldás az eltérítést esetén általában a

# DNSSEC 2

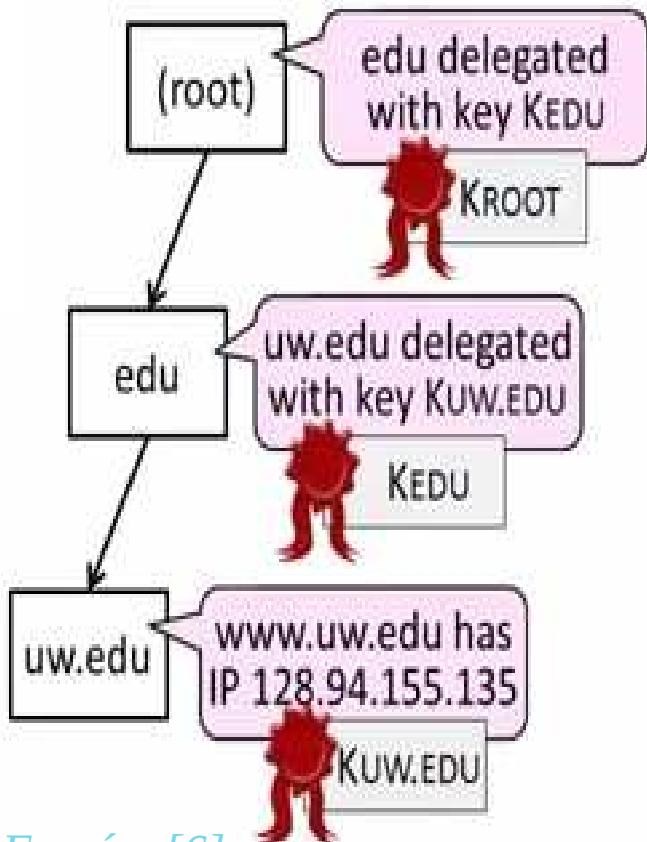
38

- Új rekordtípusokat vezettek be:
  - RRSIG a rekordok digitális aláírására.
  - DNSKEY publikus kulcsok használatához a validáció során.
  - DS publikus kulcsok használatához a delegációhoz.
  - NSEC/NSEC3 a létezés hitelesített visszautasítása.
- Első változat 1997-ben jelent meg. 2005-ben újraírták.
- Ez a fejlesztés viszont a kliensek és szerverek frissítését vonja maga után. (2010 - „Root” szerverek frissítése.)
- A kliensek a szokott módon kérdezik le a DNS-t, és később ellenőrzik a válasz hitelességét.
- A *Trust Anchor* a publikus kulcsok gyökere. (DNS kliens konfigurációjának a része)
- A biztonság leköveti a DNS hierarchiát.

# DNSSEC 3

39

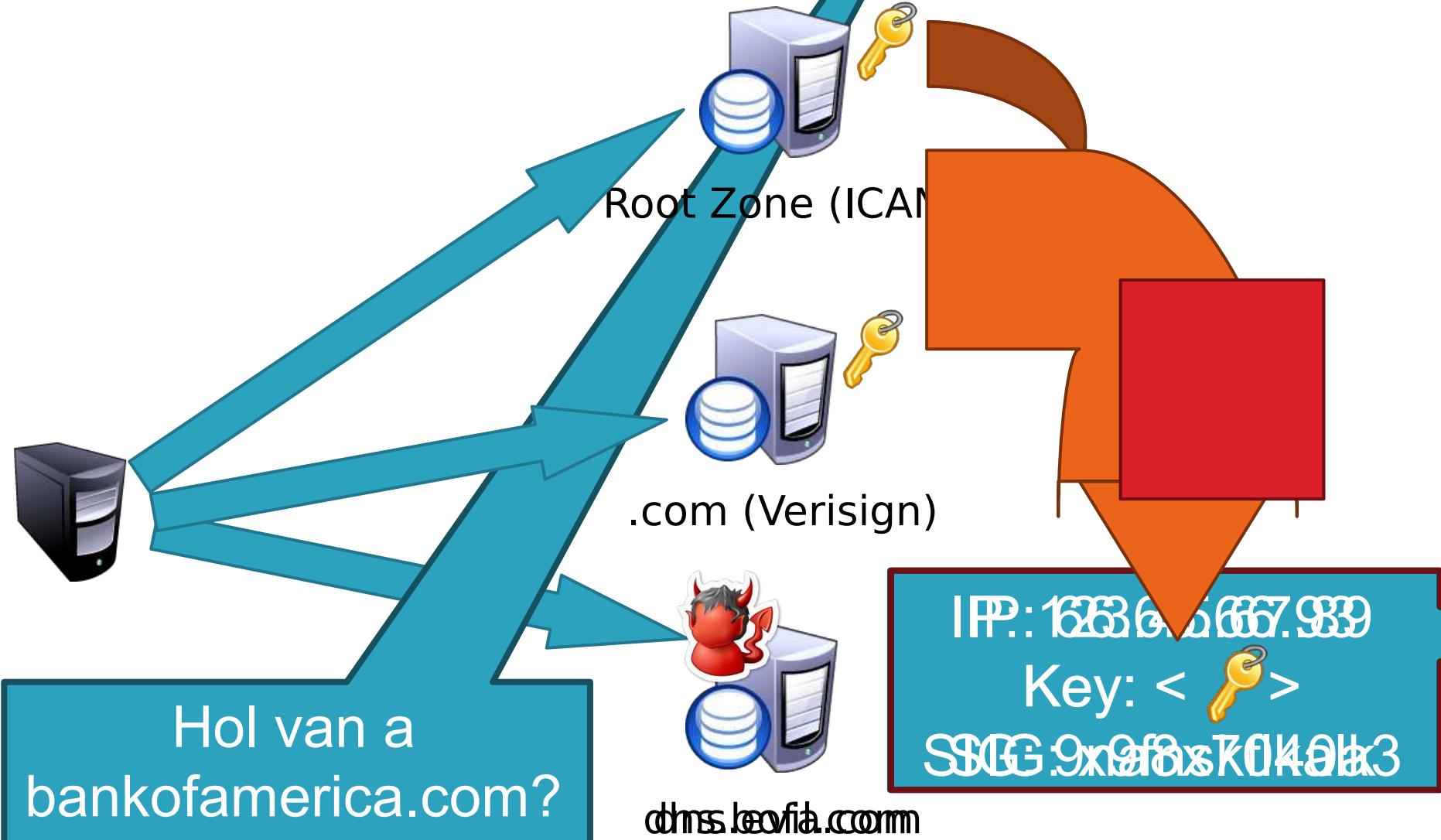
- Példa a www.uw.edu lekérdezése egy klienssel.
- Kliens hitelesíti a választ:
  1. Kroot egy *Trust Anchor*.
  2. Kroot-ot használja a Kedu ellenőrzésére.
  3. Kedu-t használja a Kuw.edu ellenőrzésére.
  4. Kuw.edu-t használja a IP cím ellenőrzésére.



Forrás: [6]

# DNSSEC Bizalmi hierarchia

40



# Does DNSSEC Solve all our problems?

41

- No.
- DNS still vulnerable to reflection attacks + injected responses

# DNS Reflection

42

- Very big incident in 2012
  - (<http://blog.cloudflare.com/65gbps-ddos-no-problem/>)
  - 65 Gbps DDoS
  - Would need to compromise 65,000 machines each with 1 Mbps uplink
    - How was this attack possible?
- Use DNS reflection to amplify a Botnet attack.
- Key weak link: Open DNS resolvers will answer queries for anyone  
<http://openresolverproject.org/>

# So how does this work?

43

- Remember: DNS is UDP
- No handshaking between endpoints
- I can send a DNS query with a forged IP address and the response will go to that IP address
  - **Secret sauce:** a small request that can elicit a large response
  - E.g., query for zone files, or DNSSEC records (both large record types).
- Botnet hosts spoof DNS queries with victim's

# DNS amplification illustrated

44



Src: Victim  
Dst: Open Resolver  
DNS ...



Sometimes the DNS  
resolver network thinks it  
is under attack by the  
bots infected by botnet  
victim!!

Victim

DNS ...