

# Számítógép-hálózatok



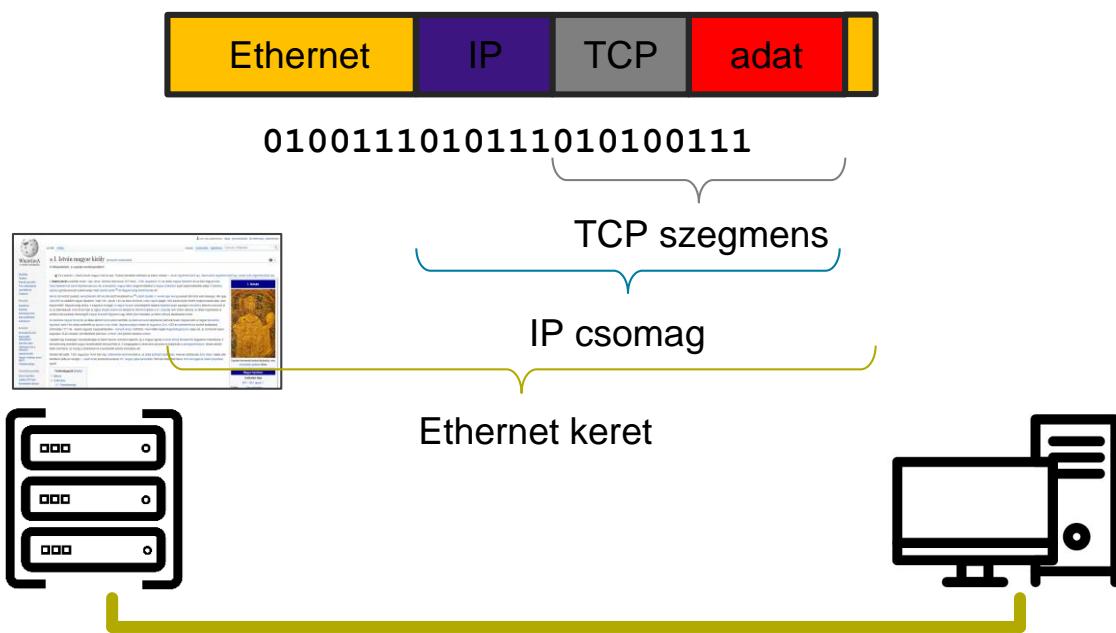
## 6. Hálózati réteg I. Útválasztás

# Tartalom

- Hálózati réteg feladatai
- Hálózati réteg szolgáltatásai
  - Összeköttetés nélküli
  - Összeköttetés-alapú
- Útválasztás és továbbítás
- Útválasztó algoritmusok
  - Legrövidebb útvonal alapján történő útválasztás
  - Elárasztás
  - Távolságvektor alapú útválasztás
  - Kapcsolatállapot-alapú útválasztás
  - Hierarchikus útválasztás
  - Broadcast, multicast, anycast
- Hálózatok összekapcsolása
- Példák: internetes útválasztó protokollok
- IPv4
- IPv6



# Ismétlés: réteges szerkezet



# A hálózati réteg feladata

- Csomag eljuttatása a forrástól a célig
  - Az adatkapcsolati réteg szolgáltatásait használja
- Esetleg több útválasztón és több hálózaton keresztül
- Ismernie kell a hálózat topológiáját
- Útvonal kiválasztása
  - Optimalizálás
  - Túlterhelés elkerülése

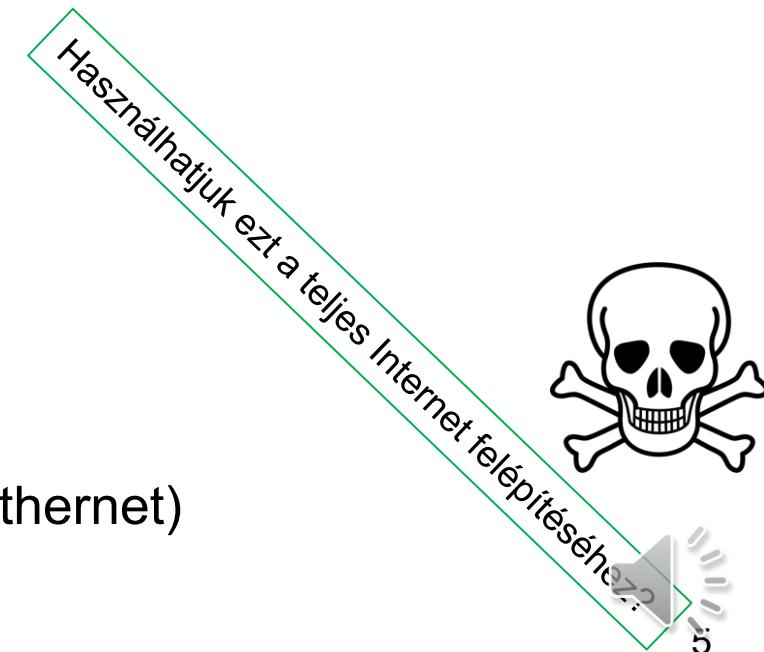


# Miért kell a hálózati réteg? (1)

- Az adatkapcsolati rétegben már tudunk kereteket küldeni két állomás között...
- A hálózati kapcsolók megtanulják a topológiát...
- Miért kell még egy réteg?

## • Kapcsoló (ismétlés):

- minden ismert címet tárol
- Tanulás: broadcast
- Körök elkerülése: feszítőfa
- Csak egyfajta közegre működik (pl. Ethernet)



# Miért kell a hálózati réteg? (2)

- Kapcsoló:
  - **Minden ismert címet tárol**
  - Tanulás: broadcast
  - Körök elkerülése: feszítőfa
  - Csak egyfajta közegre működik (pl. Ethernet)



Minden kapcsolónak az Internet MINDEN eszközének MAC-címét tárolnia kellene  
Hatalmas adatmennyiség



Hierarchikus címeket fogunk alkalmazni  
Analógia: postai hierarchikus címzés



# Miért kell a hálózati réteg? (3)

- Kapcsoló:
  - minden ismert címet tárol
  - **Tanulás: broadcast**
  - Körök elkerülése: feszítőfa
  - csak egyfajta közegre működik (pl. Ethernet)



Minden új cím esetén az Internet MINDEN géphez el kellene egy broadcast üzenetet juttatni



A hierarchikus címzés miatt csak a fő útvonalakat kell tudni  
Útvonalválasztók egymást segítik a tanulásban



# Miért kell a hálózati réteg? (4)

- Kapcsoló:

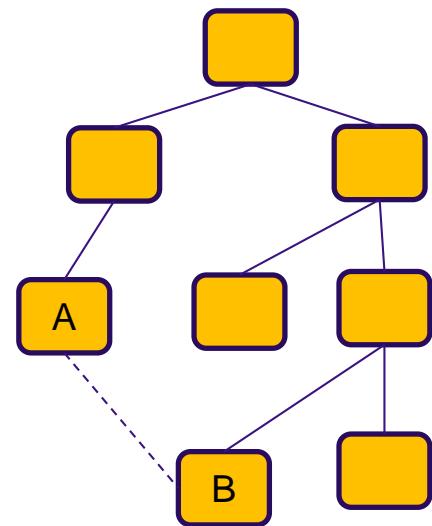
- minden ismert címet tárol
- Tanulás: broadcast
- **Körök elkerülése: feszítőfa**
- Csak egyfajta közegre működik (pl. Ethernet)



Egyes linkeket nem használunk  
Az útvonal sok esetben nem optimális



Az útvonalválasztók minden linket használnak



# Miért kell a hálózati réteg? (5)

- Kapcsoló:

- minden ismert címet tárol
- Tanulás: broadcast
- Körök elkerülése: feszítőfa
- **Csak egyfajta közegre működik (pl. Ethernet)**



A teljes Internetet egyetlen technológiára  
kellene építeni



Tetszőleges adatkapcsolati rétegek összekapcsolhatók



# A szállítási rétegnek nyújtott szolgáltatások

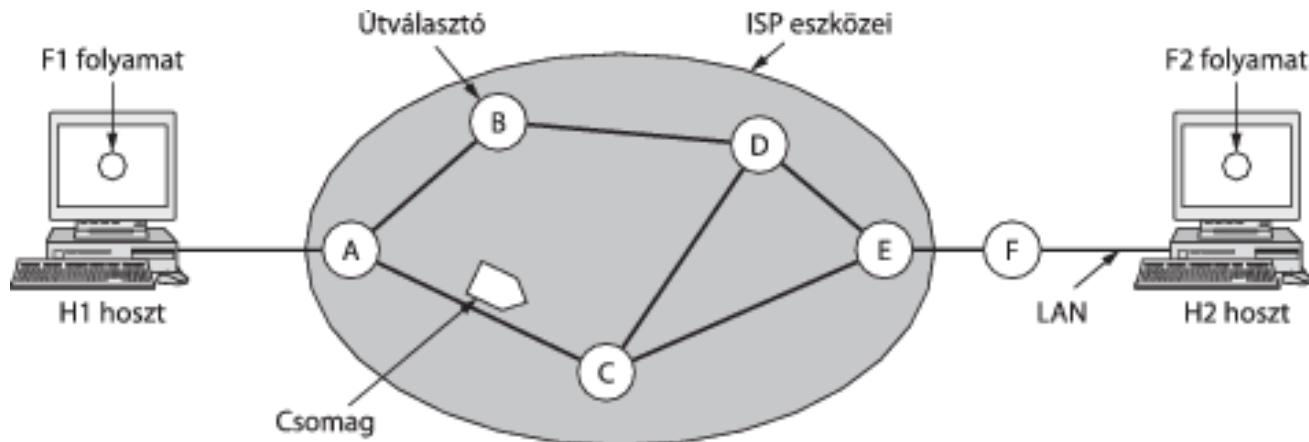
- Csomag eljuttatása a forrástól a célig
- Elvárások:
  - A szolgáltatásoknak függetleneknek kell lenniük az útválasztók kialakításától.
  - A szállítási réteg elől el kell takarni a jelenlevő útválasztók számát, típusát és topológiáját.
  - A szállítási réteg rendelkezésére bocsátott hálózati címeknek egységes számozási rendszert kell alkotniuk, még LAN-ok és WAN-ok esetén is.
- Módszer.
  - Tárol- és továbbít típusú csomagkapcsolás
- Lehetséges megoldások:
  - Összeköttetés nélküli szolgáltatás
  - Összeköttetés-alapú szolgáltatás



# Tárol- és továbbít típusú csomagkapcsolás

- Hoszt:
  - csomagot a szolgáltató felé pont-pont kapcsolaton keresztül küldik a legközelebbi útválasztóig
- Útválasztó tárolja a csomagot, míg be nem érkezik
  - Ellenőrzés (CRC)
- Útválasztó a következő útválasztónak küldi a csomagot
  - Amíg meg nem érkezik a címzett hoszthoz.

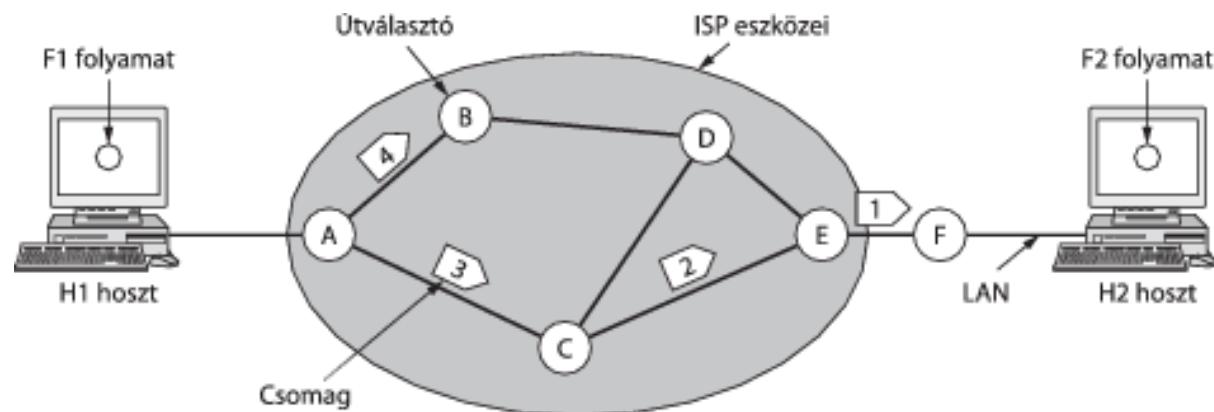
Tárol és továbbít



# Összeköttetés nélküli szolgáltatás



- Datagramalapú hálózat
- Nincs előre kiépített útvonal
- minden csomag egyedi úton halad
- Analógia: levél/távirat
- Példa: IP
- Forrás hoszt:
  - Csomag címzettnek
  - Cím: pl. IP cím
- Útválasztók:
  - Fogad/tárol/továbbít
  - Útválasztó táblázat alapján dönt
  - Ezt az útválasztó algoritmus tartja karban



A táblázata (kezdetben)	
Cél	Vonal
A	-
B	B
C	C
D	B
E	C
F	C

A táblázata (később)	
Cél	Vonal
A	-
B	B
C	C
D	B
E	B
F	B

C táblázata	
Cél	Vonal
A	A
B	A
C	-
D	E
E	E
F	E

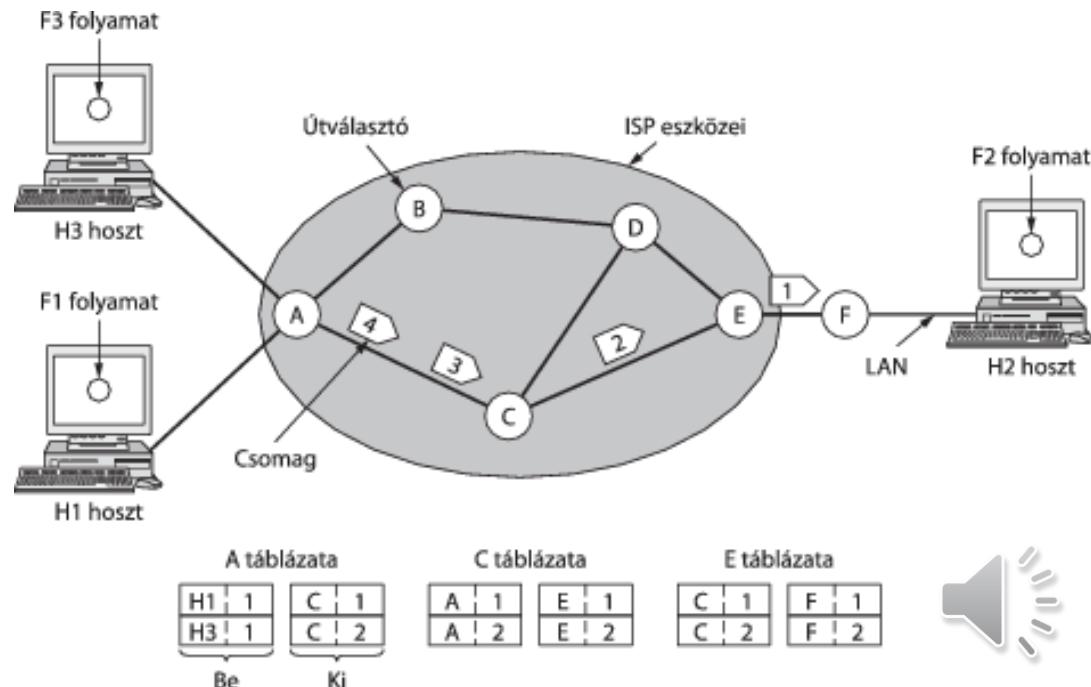
E táblázata	
Cél	Vonal
A	C
B	D
C	C
D	D
E	-
F	F



# Összeköttetés-alapú szolgáltatás (1)



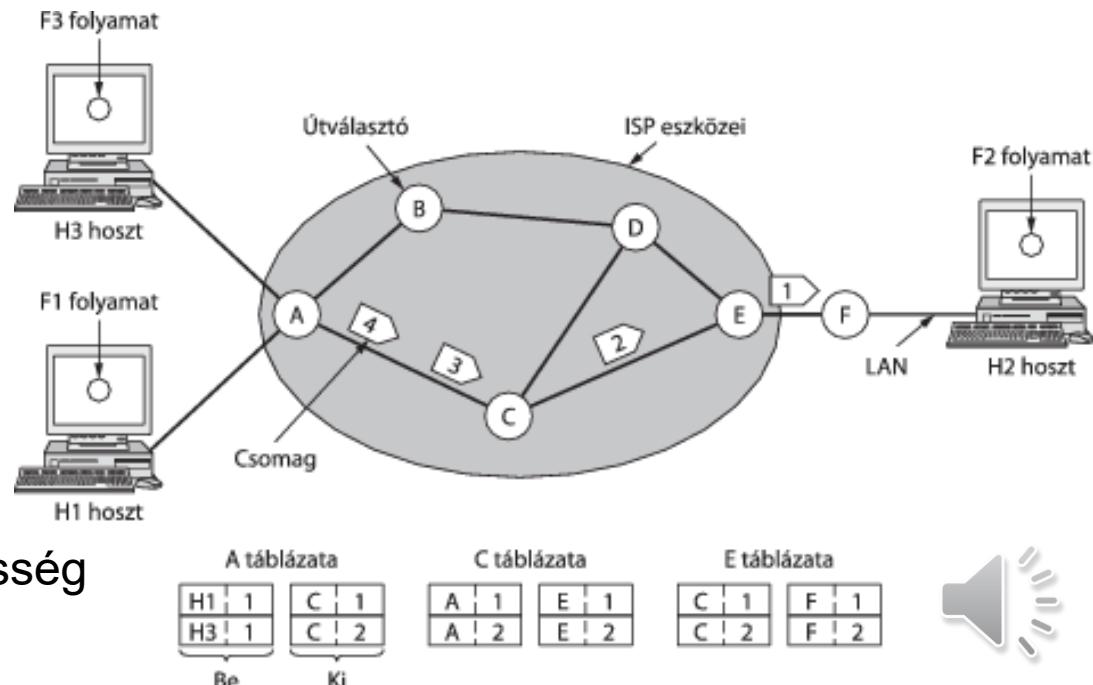
- Virtuálisáramkör-alapú hálózat
- Először útvonal kiépítése, végül lebontása
  - „virtuális áramkör”
  - Ennek azonosítóját tárolják az útválasztók (összeköttetés [ÖK] azonosító)
- minden csomag egy úton halad
- Végül az útvonal lebontása
- Analógia: telefon
- Példa: MPLS



# Összeköttetés-alapú szolgáltatás (2)

- Forrás hoszt H1 (felső sorok):
  - ÖK azonosító = 1, címzett: H2
- Forrás hoszt H3 (alsó sorok):
  - ÖK azonosító = 1, címzett: H2
- Útválasztó táblázat:
  - Bejövő vonal/azonosító
  - Kimenő vonal/azonosító
  - Azonosító változik!

- Miért áramkör?
  - Rögzített útvonal
- Miért virtuális?
  - Nincs lefoglalva sávszélesség



# Összehasonlítás

Kérdés	Datagramalapú hálózat	Virtuálisáramkör-alapú hálózat
Áramkör-felépítés	Nem szükséges 	Megkövetelt 
Címzés	Minden csomag tartalmazza a teljes forrás- és céldímet 	Minden csomag egy rövid virtuálisáramkör-számot tartalmaz 
Állapotinformáció	Az útválasztó nem tartalmaz állapotinformációt 	Minden virtuális áramkör összeköttetéseként helyet igényel az útválasztó táblázatában 
Útválasztás	Minden csomagot egyedileg irányítanak	Az útvonalat akkor választják ki, amikor a virtuális áramkör felépül; minden csomag ezt az útvonalat követi
Az útválasztó meghibásodásainak hatása	Semmi, eltekintve az összeomlás során elveszett csomagoktól 	Minden virtuális áramkör megszakad, amely a meghibásodott útválasztón keresztülhaladt 
Szolgáltatás-minőség	Bonyolult 	Könnyű, ha elég erőforrást lehet előre lefoglalni mindegyik virtuális áramkör számára 
Torlódáskezelés	Bonyolult 	Könnyű, ha elég erőforrást lehet előre lefoglalni mindegyik virtuális áramkör számára 



# Útválasztás és továbbítás

- Routing / forwarding
- Útválasztás
  - Hálózati réteg arról dönt, hogy egy beérkező csomagok merre menjenek tovább
  - Ehhez meg kell tanulni a hálózat aktuális topológiáját
  - A hálózati eszközök együtt, elosztott módon végzik
  - Eredmény:
    - útválasztó táblázatok feltöltése és karbantartása
  - Lassú folyamat
- Csomagtovábbítás
  - Amikor a csomag beérkezik, továbbítás a megfelelő irányba
  - A korábban megtanultak (táblázatok) alapján
  - Gyors



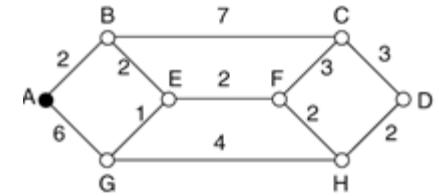
# Útválasztó algoritmusok

- Legrövidebb útvonal alapján történő útválasztás
- Elárasztás
- Távolságvektor alapú útválasztás
- Kapcsolatállapot-alapú útválasztás
- Hierarchikus útválasztás
- Broadcast, multicast, anycast



# Legrövidebb út algoritmus (1)

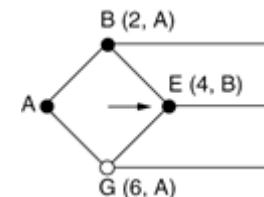
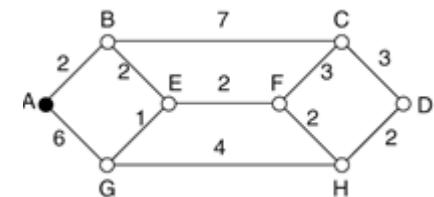
- (Shortest path)
- Gráf ábrázolás
  - Csomópont: állomás
  - Él: kapcsolat
  - Él költsége: a kapcsolat valamelyen költsége
    - Pl. távolság, késleltetési idő, ár, stb...
- Keressük két pont között a legkisebb költségű utat
- Dijkstra algoritmusa



Edsger Wybe Dijkstra

# Legrövidebb út algoritmus (2)

- Diskstra algoritmusa:
  - keressük egy adott csomóponttól (forrás) a legrövidebb távolságokat (a forrás legyen A)
- Címkézzük fel a csomópontokat:
  - Legrövidebb távolság a forrástól
  - A legrövidebb út előző csomópontja
  - Egy címke lehet végleges ( $\cdot$ ) vagy ideiglenes ( $\circ$ )
- Inicializálás:
  - Kezdetben minden címke:  $(\infty, -)$  és ideiglenes (kivéve a forrást)



# Legrövidebb út algoritmus (3)

Iteráció:

1. Legyen a forrás a **munkacsomópont**
2. **Címkézzük újra a munkacsomópont szomszédait**
  - Az új távolság alábbiak közül a kisebb:
    - régi címke távolság
    - munkacsomóponttól mért távolság + munkacsomópont távolsága a forrástól
  - A címke előző csomópontja:
    - marad a korábbi
    - munkacsomópont
3. Válasszuk ki a **legkisebb értékű ideiglenes csomópontot**
  - ezt véglegesítük
  - ez lesz az új munkacsomópont
4. Ismétlés a 2. lépéstől, amíg minden csomópont végleges nem lesz.



# Legrövidebb út algoritmus (4)

0. Kezdeti címkék:  $(\infty, -)$

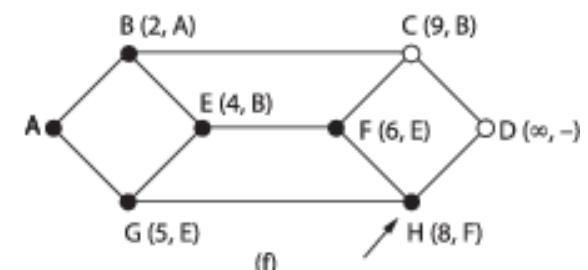
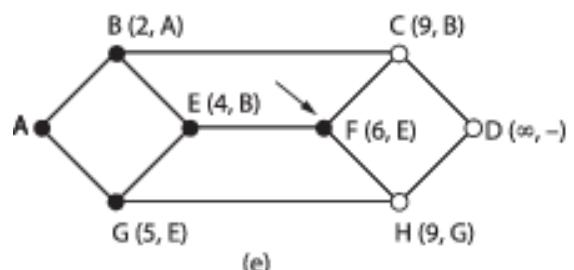
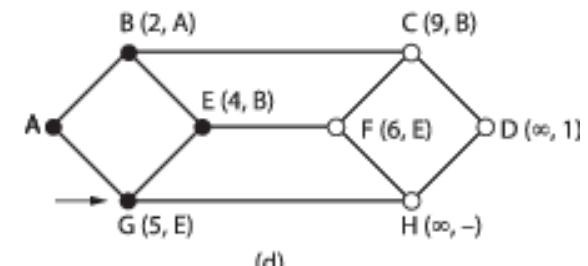
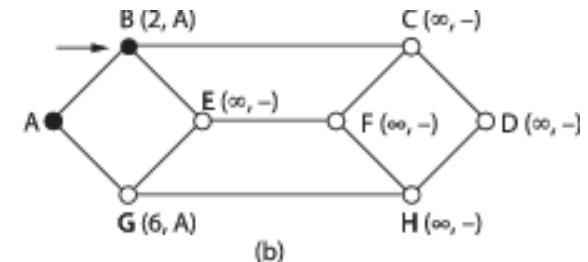
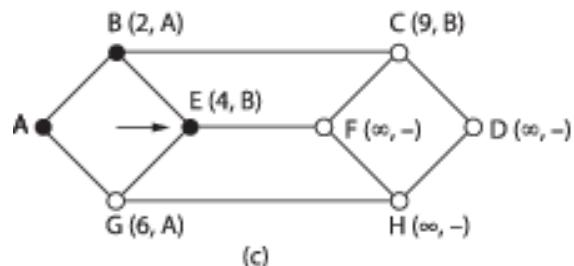
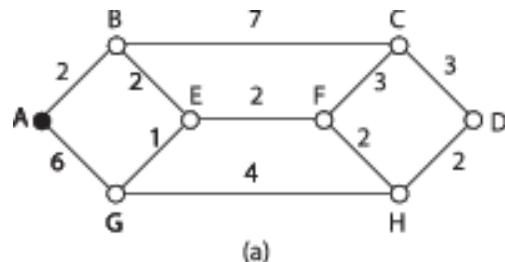
1. Legyen a forrás a MCS\*

2. MCS szomszédainak újracímkézése

3. Legkisebb értékű ideiglenes CSP\*\*

- véglegesítés
- ez lesz az új MCS

4. Ismétlés a 2. lépéstől.



\* munka csomópont

\*\* csomópont

Melyik a legrövidebb út A-ból F-be?

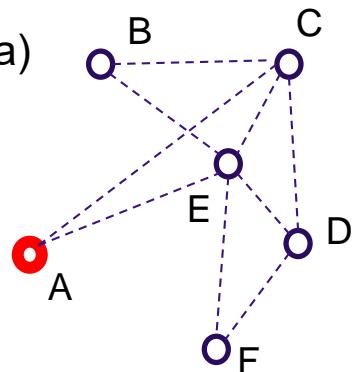
Visszafelé lépkedünk a címkézett gráfon:  $F \rightarrow E \rightarrow B \rightarrow A$

A legrövidebb út tehát: A-B-E-F



# Elárasztás

- (Flooding)
- Szabályok:
  - minden beérkező üzenetet továbbítunk minden szomszédnak
    - kivéve annak, akitől kaptuk
  - de csak egyszer!
    - azonosítani kell az üzeneteket (pl. sorszám+feladó azonosítója)
    - meg kell jegyezni a továbbküldött üzenetek azonosítóját
- Nagyon egyszerű, robosztus
- Nem takarékos
  - feleslegesen sok üzenet
  - egy állomás többször is megkaphat egy üzenetet
- Nagyon fontos és gyakori építőelem



A: AE, AC  
C: CB, CE, CD  
E: EA, EB, EC, ED, EF  
B: BC, BE  
D: DC, DE, DF  
F: FD, FE



# Távolságvektor alapú útválasztás (1)

- minden útválasztóban táblázat:
  - minden célig a legrövidebb távolság
  - minden célhoz a következő ugrás azonosítója
- A táblázatokat egymással (szomszédokkal) tudatják
- Frissíti saját táblázatát az alábbi források alapján:
  - A szomszéduktól kapott információ (táblázat)
  - Szomszéduktól mért „távolság” (ugrás, késleltetés)
  - Számítás:
    - Távolságom X-től =  $\min(S_i \text{ szomszédom távolsága X-től} + \text{távolságom } S_i\text{-től})$



# Távolságvektor alapú útválasztás (2)

- Példa: J táblázatának számítása
- J szomszédai: A,I,H,K
  - 4 táblázatot kap

- J szomszédaitól távolságát méri
  - Távolság a szomszédaitól:

8, 10, 12, 6

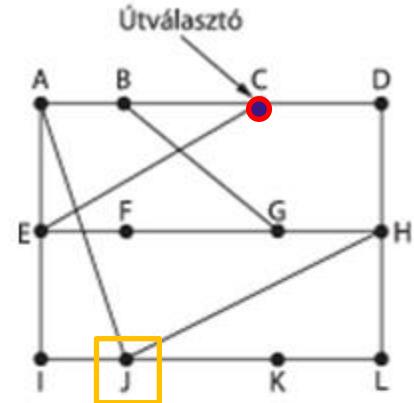
- Pl. távolság C-től:

$$\min(25+8, \textcolor{red}{18+10}, 19+12, 36+6)=28$$

A	I	H	K
---	---	---	---

- Ugrás C felé:

I-n keresztül



Útválasztók felé A	Új becsült késleltetés J-től				
	I	H	K	Vonal	
A 0	24	20	21	8 A	
B 12	36	31	28	20 A	
C 25	18	19	36	28 I	
D 40	27	8	24	20 H	
E 14	7	30	22	17 I	
F 23	20	19	40	30 I	
G 18	31	6	31	18 H	
H 17	20	0	19	12 H	
I 21	0	14	22	10 I	
J 9	11	7	10	0 -	
K 24	22	22	0	6 K	
L 29	33	9	9	15 K	
JA	JI	JH	JK	Új útválasztó tábla J-hez	
késleltetés = 8	késleltetés = 10	késleltetés = 12	késleltetés = 6	J négy szomszédjától kapott vektorok	

# Távolságvektor alapú útválasztás (3)

- Probléma: információ esetenként lassan terjed
  - Jó hír gyorsan terjed (van egy új csomópont / új link)
  - Rossz hír lassan terjed (egy csomópont/link megszűnt)
    - Végtelenig számolás problémája
  - Példa: egyszerű lineáris hálózat: A-B-C-D-E
    - minden távolság 1

A  
B  
C  
D  
E

•	•	•	•	•
1	•	•	•	•
1	2	•	•	•
1	2	3	•	•
1	2	3	4	•

(a)

A  
B  
C  
D  
E

•	•	•	•	•
1	2	3	4	•
3	2	3	4	•
3	4	3	4	•
5	4	5	4	•
5	6	5	6	•
7	6	7	6	•
7	8	7	8	•
⋮	⋮	⋮	⋮	⋮
•	•	•	•	•

(b)

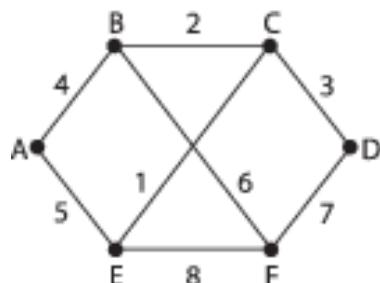
Kezdetben az ARPANET ezt használta.  
Felváltotta a kapcsolatállapot-alapú útválasztás.

- B, C, D és E csomópontok távolsága az A csomóponttól a csomópontok táblázatai szerint*
- A megjavul: jó hír terjedése gyors*
  - A elromlik: rossz hír terjedése lassú*

A gyakorlatban számos heurisztikát alkalmaznak a javításra.  
Egyik sem teljesen jó...

# Kapcsolatállapot-alapú útválasztás (1)

- A szomszédokkal való kapcsolatokat használja/cseréli (Link State Routing)
- minden útválasztó tennivalója (5 lépés):
  1. Felkutatni a szomszédait és megtudni a hálózati címeiket.
    - **HELLO üzenetek** minden linken
  2. Beállítani a távolság vagy a költség értékét a minden szomszédjáig a mért késleltetés alapján.
    - A **költség** a sávszélesség reciprokával arányos
    - **Mérés:** pl. ECHO körülfordulási ideje
  3. Összeállítani egy csomagot, amely a most megtudottakat tartalmazza.
    - Tartalmaz még sorszámot és kort is.



Kapcsolatállapot-csomagok					
A	B	C	D	E	F
Sorsz.	Sorsz.	Sorsz.	Sorsz.	Sorsz.	Sorsz.
B 4	A 4	B 2	C 3	C 5	B 6
E 5	C 2	D 3	F 7	A 1	D 7



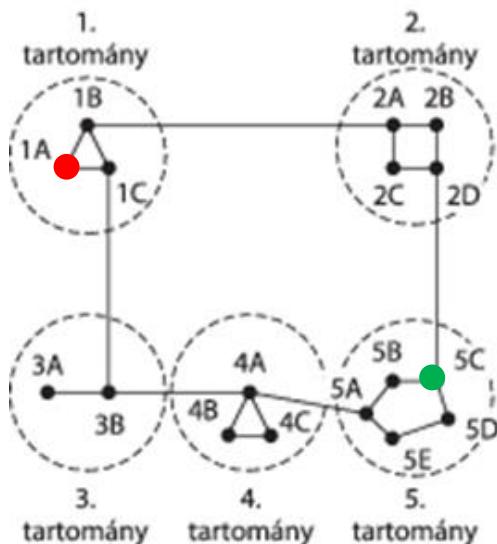
# Kapcsolatállapot-alapú útválasztás (2)

4. Elküldeni ezt a csomagot az összes többi útválasztónak.
    - Továbbfejlesztett **elárasztás**
    - A sorszám és kor mezők segítségével biztosítja az adatok konzisztenciáját
  5. Kiszámítani az összes többi útválasztóhoz vezető legrövidebb utat.
    - Kapott adatokból **összeállítja a teljes hálózat gráfját**
    - Ezen Diskstra algoritmust futtat ( minden útválasztó külön-külön)
- Sok rendszer használja (főleg belső hálózatokban):
    - **IS-IS** (Intermediate System to Intermediate System Intradomain Routing Protocol) – Közbenső rendszertől közbenső rendszerig történő körzetben belüli útválasztó protokoll
    - **OSPF** (Open Shortest Path First) – nyílt hozzáférésű, a legrövidebb utat előrevező protokoll
  - Kapcsolatállapot- és távolságvektor-alapú algoritmusok problémája:
    - Teljes hálózat topológiája kell hozzá
    - Nagy hálózatokra nem alkalmasak



# Hierarchikus útválasztás

- Alapötlet: a hálózatot régiókra kell osztani
- Útválasztás két lépésben:
  - Régiók között
  - Régiót belül
- Analógia:
  - Levéltovábbítás
  - Telefonközpontok



Szuboptimális lehet!  
Pl.: 5C

1A teljes táblázata

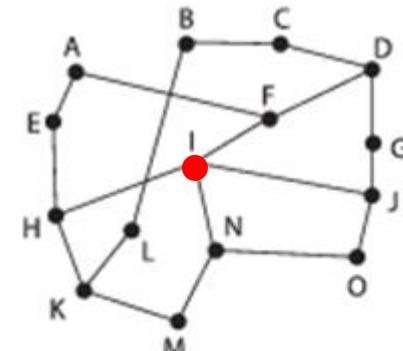
Cél	Vonal	Ugrás
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

1A hierarchikus táblázata

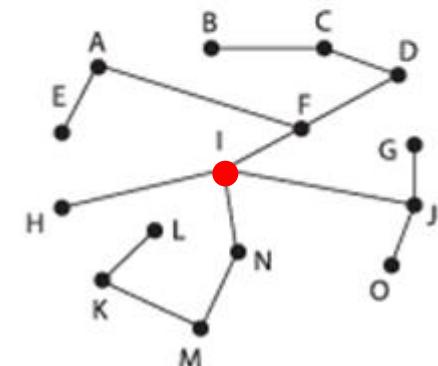
Cél	Vonal	Ugrás
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

# Adatszórás

- (Broadcasting)
- Cél:
  - a hálózat minden csomópontjához eljuttatni az üzenetet
- Legegyszerűbb megoldás:
  - elárasztás
- Egyéb módszerek
  - Feszítőfa alkalmazása



(a)



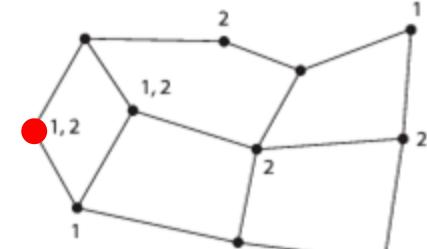
(b)

- (a) Hálózat  
(b) A piros csomópont feszítőfája

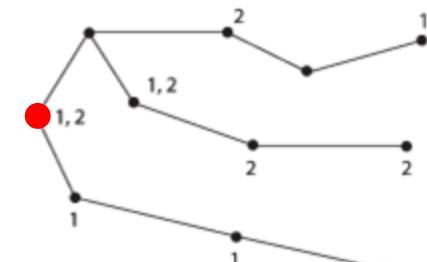


# Többesküldés

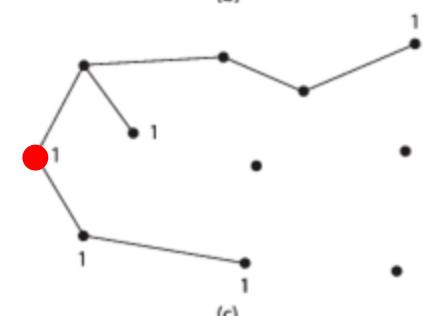
- (Multicasting)
- Cél:
  - a hálózat egy csoportjához eljuttatni az üzenetet
  - A csoportnak van egy közös címe
- Lehetséges megoldások
  - Elárasztás + vevőben szűrés
    - Jó, ha sűrű a hálózat
    - Pazarló, hiszen oda is eljut, ahová felesleges
  - Csonkolt feszítőfa alkalmazása
    - Feszítőfa építése a forrásból
    - Felesleges részek levágása



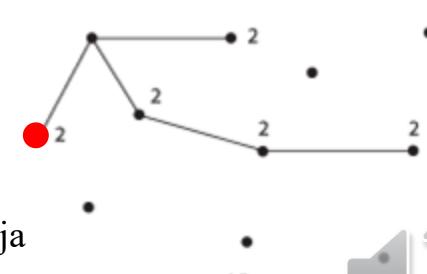
(a)



(b)



(c)



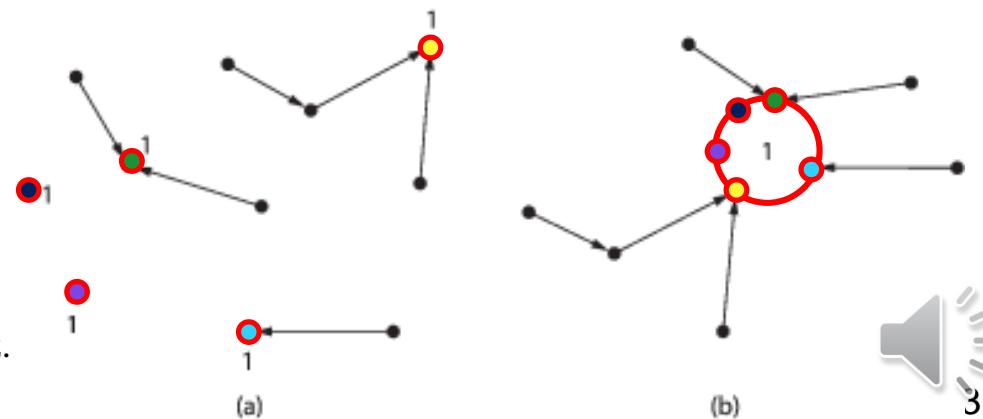
(d)

- (a) Hálózat
- (b) A piros csomópont útválasztó feszítőfája
- (c) Többesküldéses fa az 1-es csoporthoz
- (d) Többesküldéses fa a 2-es csoporthoz



# Bárkinek küldés

- (Anycasting)
- Cél:
  - a hálózat egy csoportján belül...
  - ... egy tetszőleges (legközelebbi) taghoz kell eljuttatni az üzenetet
  - A csoportnak van egy közös címe
- Megoldás:
  - minden csoportnak ugyanaz a címe
  - A távolság- vagy kapcsolatállapot-alapú útválasztás mindenkor a legközelebbi választja ki



- (a) Bárkinek küldéses útvonalak az 1-es csoporthoz.  
(b) Az útválasztó protokoll által látott topológia



# Hálózatok összekapcsolása (1)

- Rengeteg fajta hálózat létezik
  - Ezek sok mindenben különböznek
  - Az internethoz ezeket mégis össze kell tudni kapcsolni
  - Réteges szerkezet segít...
  - Eszközök:
    - Ismétlők (repeater)
    - Elosztók (hub)
    - Hidak (bridge)
    - Kapcsolók (switch)
    - Útválasztók (router)
- 32
- Fizikai réteg
- Adatkapcsolati réteg
- Hálózati réteg

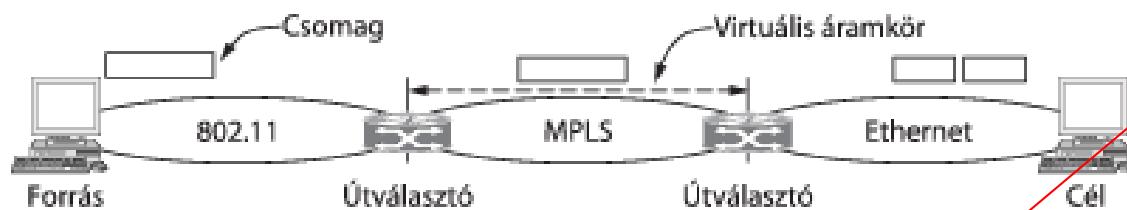
Tétel	Néhány lehetőség
Felkinált szolgáltatás	Összeköttetés nélküli vagy összeköttetés-alapú
Címzés	Különböző méretek, egyszintű vagy hierarchikus
Adatszórás	Van vagy nincs (ugyanez többesküldésre is)
Csomagmérét	Minden hálózat megszab egy legnagyobb értéket
Sorrendiségek	Sorrendhelyes és nem sorrendhelyes kézbesítés
Szolgáltatás minősége	Van vagy nincs; sokfajta létezik
Megbízhatóság	Különböző szintű csomagvesztés
Biztonság	Bizalmassági szabályok, titkosítás stb.
Paraméterek	Eltérő időzítések, folyam-meghatározások stb.
Számlázás	Összeköttetési idő alapján, csomagok száma alapján, bájtok száma alapján vagy seholgy

# Hálózatok összekapcsolása (2)

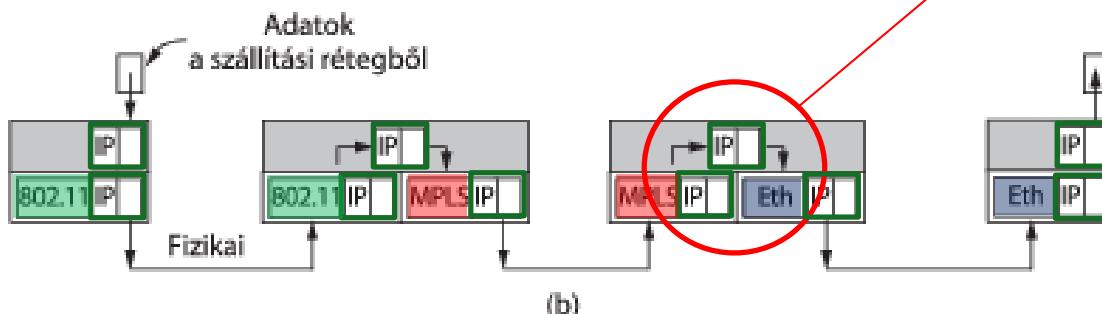
## Példa

- Különböző hálózatok (L1, L2):
  - WiFi
  - MPLS (virtuális áramkör-alapú)
  - Ethernet
- Közös hálózati protokoll (L3):
  - IP

1. Útválasztó fogadja a keretet
2. Eldobja az L2 keretet
3. L3 csomagból kiveszi a címet
4. Dönt a továbbítás irányáról
5. Becsomagolja a megfelelő L2 keretbe (esetleg darabol is)
6. Továbbítja az új keretet



(a)



(b)

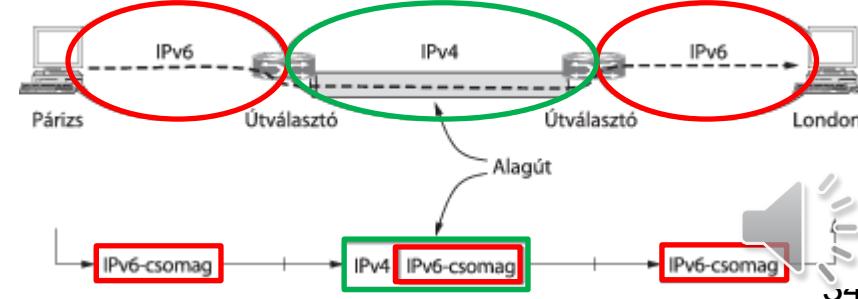
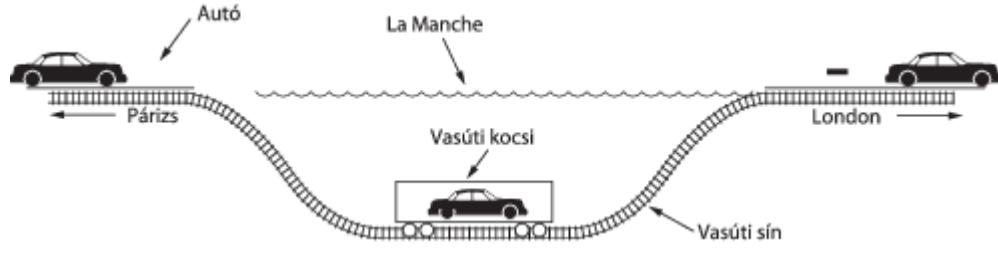
L1: fizikai réteg  
L2: adatkapcsolati réteg  
L3: hálózati réteg



# Hálózatok összekapcsolása (3)

## Alagút típusú átvitel

- Gyakori probléma:
- Forrás és célhosztok azonos típusú hálózaton vannak
  - Pl. egy cég két telephelye IPv6
- A közbülső hálózat más típusú
  - Pl. IPv4.
- Alagút (tunnel):
  - Becsomagoljuk a forrás csomagot a közbülső hálózati réteg csomagjába
  - Átvitel
  - Kicsomagolás
  - Az eredeti protokoll szerint halad tovább



# Csomagok tördelése (1)

*MTU: Maximum Transmission Unit*

- Csomagok mérete különböző
- Hálózatok maximális csomagmérete (MTU) is változik
- Mit tegyünk, ha nagy a csomag, de „keskeny” a hálózat
  - Csomagok tördelése
- Hol tördeljünk?
  - A feladó tördel MTU szerint
  - Az útvonalválasztók tördeljenek igény szerint

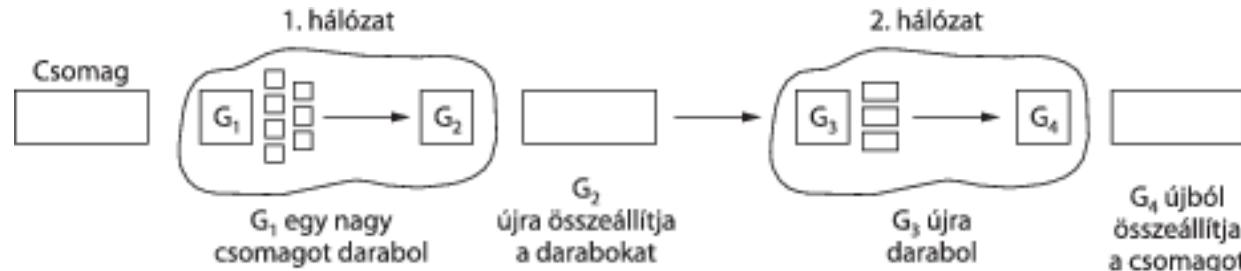
# Csomagok tördelése (2)

## Átlátszó darabolás:

- Minden hálózatban a belépéskor darabolás, kilépéskor összeállítás
- Sok munka az útválasztóknak

## Nem átlátszó darabolás:

- Minden hálózatban a belépéskor darabolás, de kilépéskor nincs összeállítás
- A csomagot a címzett állítja össze
- Egyszerű (pl. IP)



(a)



(b)

(a) Átlátszó darabolás. (b) Nem átlátszó darabolás



# Csomagok tördelése (3)

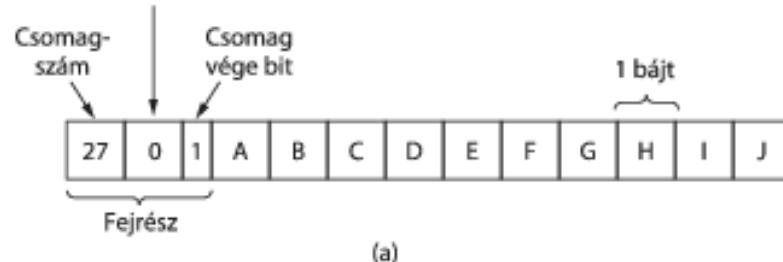
## Példa

Példa: 10 bájtos csomag

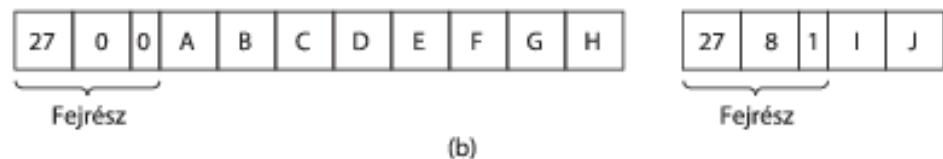
1. hálózatban MTU=8 bájt+ fejrész
2. hálózatban MTU=5 bájt + fejrész

Eredeti csomag (10 bájt+fejrész)

Ebben a csomagban az első bájt indexe



MTU: 8bájt + fejrész



MTU: 5bájt + fejrész



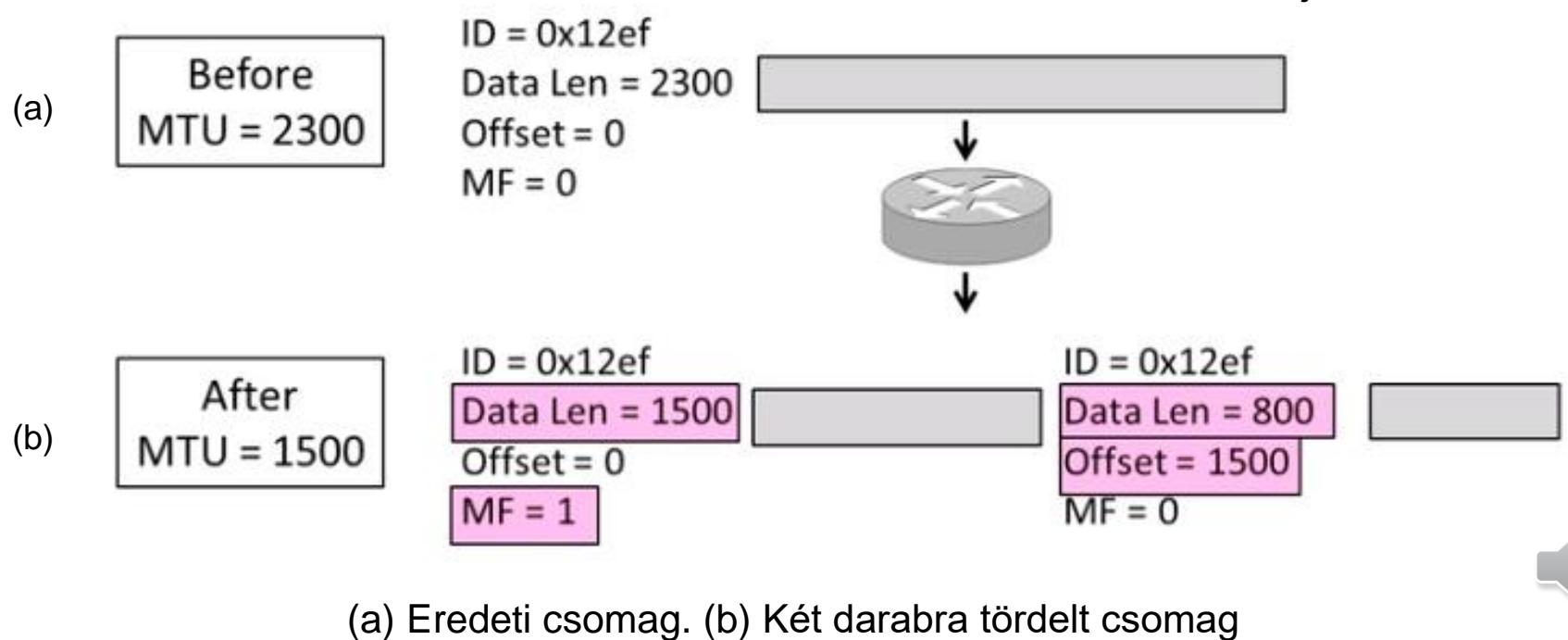
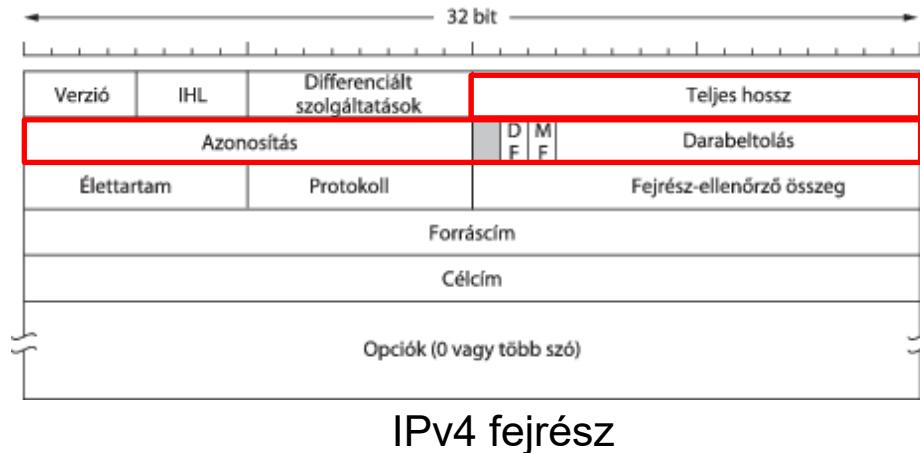
Darabolás, ha az elemi darabméretet 1 bájt.

- (a) Eredeti csomag, amely 10 adatbájtot tartalmaz.
- (b) A darabok egy olyan hálózaton való keresztülhaladás után, ahol a maximális csomagmáret 8 adatbájt + fejrész.
- (c) A darabok egy 5 adatbájt + fejrész csomagmáretű útválasztón való áthaladás után



# Csomagok tördelése (4)

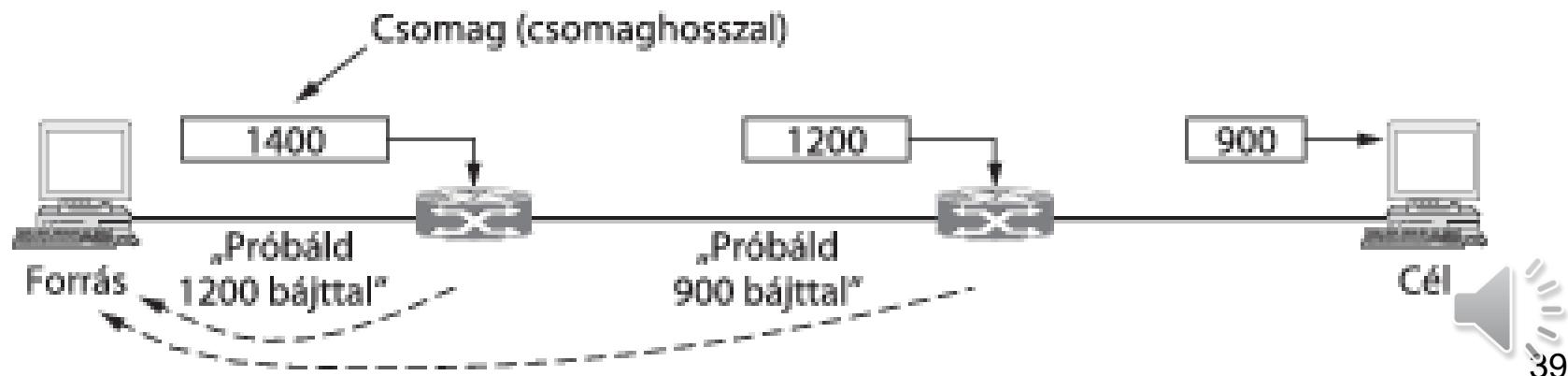
## Példa: IPv4 csomag tördelése



# Csomagok tördelése (5)

## Útvonal-MTU felderítése

- Az útvonalon tördelés működik, de...
  - Rontja hálózat teljesítményét
- Újabb trend:
  - Csak a feladó tördelhet, a routerek nem foglalkoznak vele
  - Ha nagy a csomag, router hibaüzenetet küld a feladónak
- Honnan tudja a feladó, mekkora csomagméret kell?
  - MTU-felderítés



# Példák

## Az Internet útválasztó protokolljai

- Belső átjáró protokoll
  - Cél: minél hatékonyabb útvonalkeresés saját hálózaton belül (**intradomain routing**)
  - **RIP** (Routing Information Protocol)
    - Távolságvektor-alapú
    - Végtelenig számolás problémája
  - **OSPF** (Open Shortest PathFirst)
    - Kapcsolatállapot-alapú
    - A leggyakoribb belső átjáró protokoll
  - **IS-IS** (Intermediate System to Intermediate System)
    - OSPF elődje
- Külső átjáró protokoll
  - Cél: AS-ek közötti útvonalkeresés
  - **BGP** (Border Gateway Protocol)



# BGP – Határátjáró protokoll (1)

## Működési körülmények és követelmények

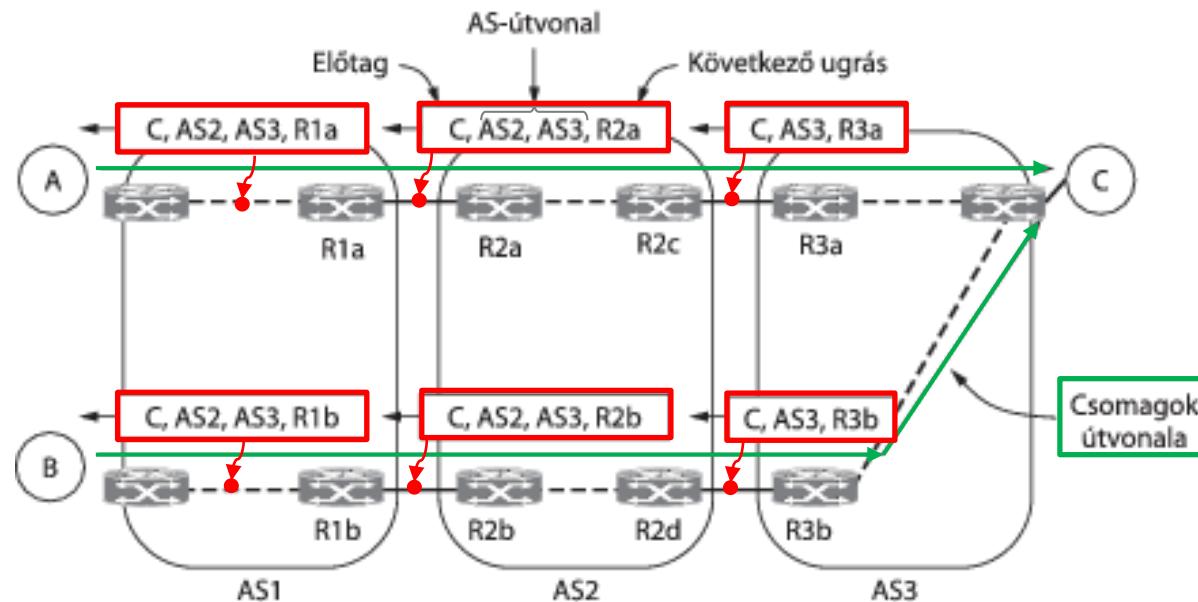
- BGP: Border Gateway Protocol
- A különféle **autonóm** hálózatok/rendszerek (**AS**) között használatos útválasztó protokoll
- Cél: hatékony útvonalkeresés + politikai szempontok (stratégiai) figyelembe vétele
- Az egyes AS-ek különféle **stratégiaikat** alkalmaznak
  - Felsőoktatási hálózaton nem lehet kereskedelmi forgalom
  - A Pentagonból nem lehet Irakon keresztül forgalom
  - Használd a TeliaSonera hálózatát a Verizon helyett, mert olcsóbb
  - Ne használd az AT&T hálózatát Ausztráliában, mert lassú
  - Az Apple-ből induló és oda érkező forgalom ne menjen át a Google hálózatán
- Lehetséges **szolgáltatások**:
  - **Tranzit**: átmenő forgalom az internet másik oldal felé (pl. ügyfél-ISP, ISP-ISP)
  - **Peering**: az egymás közötti forgalom ingyenes (ISP-ISP)
- Ilyen feltételek mellett kell globális együttműködést biztosítani: ez a BGP

USA példák



# BGP – külső átjáró protokoll (2)

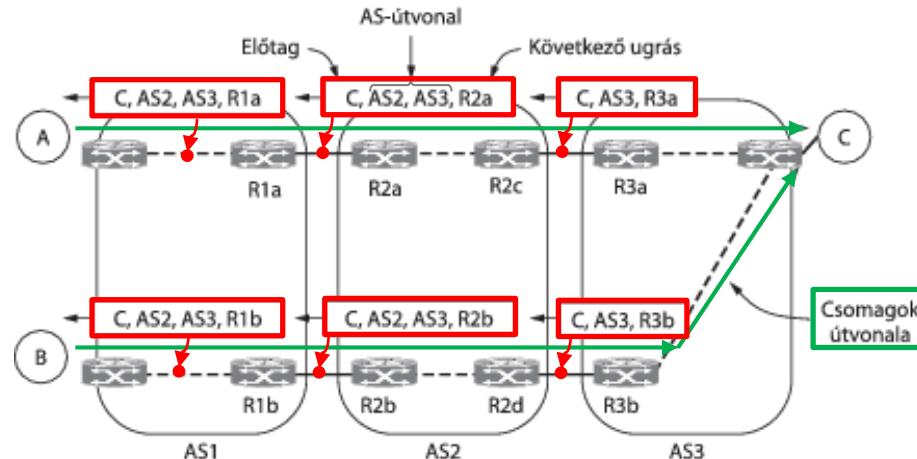
- BGP útvonal-vektort használ
  - Cél prefix – Útvonal (AS-ek sorozata) – következő ugrás
- A BGP az útvonal-vektorokat terjeszti a hálózatban az útválasztókon keresztül
  - Az AS határán az útválasztó a lista előtt teszi az AS számát, a következő ugrásnak pedig magát állítja be
  - A határ-útválasztók egymás útvonalait is megtanulják
- Útválasztás a kapott útvonal-vektorokat alapján történik



# BGP – külső átjáró protokoll (3)

Gyakori stratégiák:

1. Peering útvonalaknak előnye van a tranzittal szemben
2. Rövidebb útvonal jobb
3. AS-en belül a legolcsóbb (legrövidebb) útvonal használata
  - Ez a korai kilépés, vagy **forró krumpli útválasztás**
  - Pl. az A-C csomagok a felső útvonalon fognak haladni, a B-C csomagok pedig az alsón
4. A határon csak olyan útvonalat hirdet, amelyet a szomszéd AS felé támogat
  - Többöt kiszűri (akkor is, ha ilyen útvonal létezik)



# BGP – külső átjáró protokoll (4)

Hirdetések fajtái:

- CU: felhasználó
- TR: tranzit
- PE: peering

Példa:

Hirdetések

**CU:**  
AS2→AS1: (A,[AS2])

**TR:**  
AS1→AS2: (B,[AS1, AS3])  
AS1→AS2: (C,[AS1, AS4])

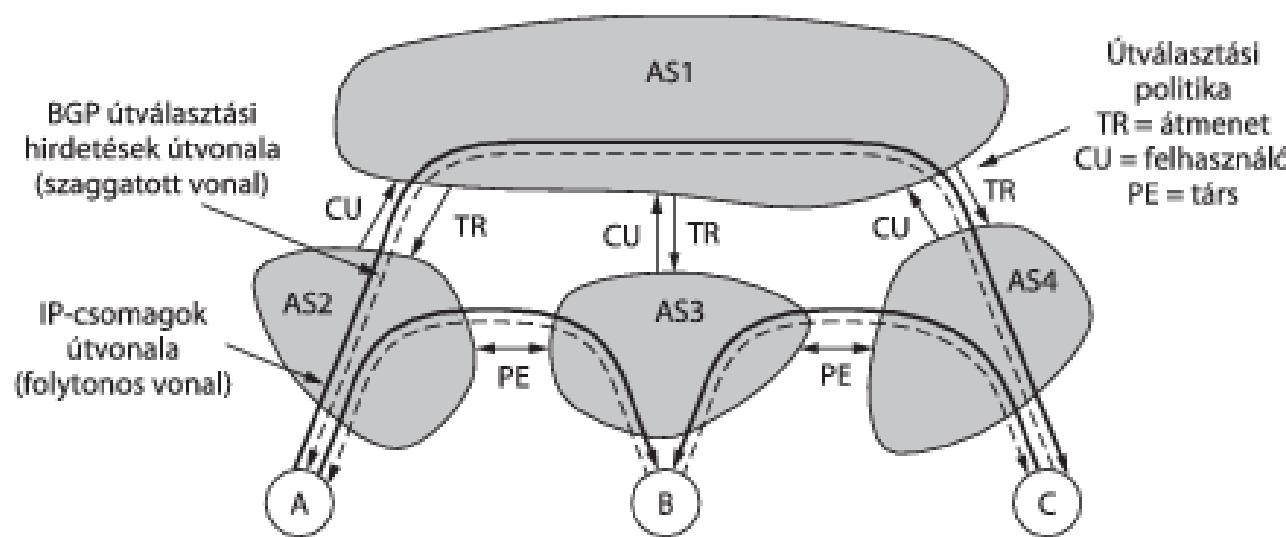
**PE:**  
AS3→AS2: (B,[AS3])  
AS2→AS3: (A,[AS2])

Csomagok továbbítása

A→C  
[AS1, AS4]  
A→B  
[AS1, AS3]  
[AS3]

Csak egy út van

Két út van,  
ez a jobb



# Számítógép-hálózatok



## 7. Hálózati réteg II. Az internetprotokoll

# Tartalom

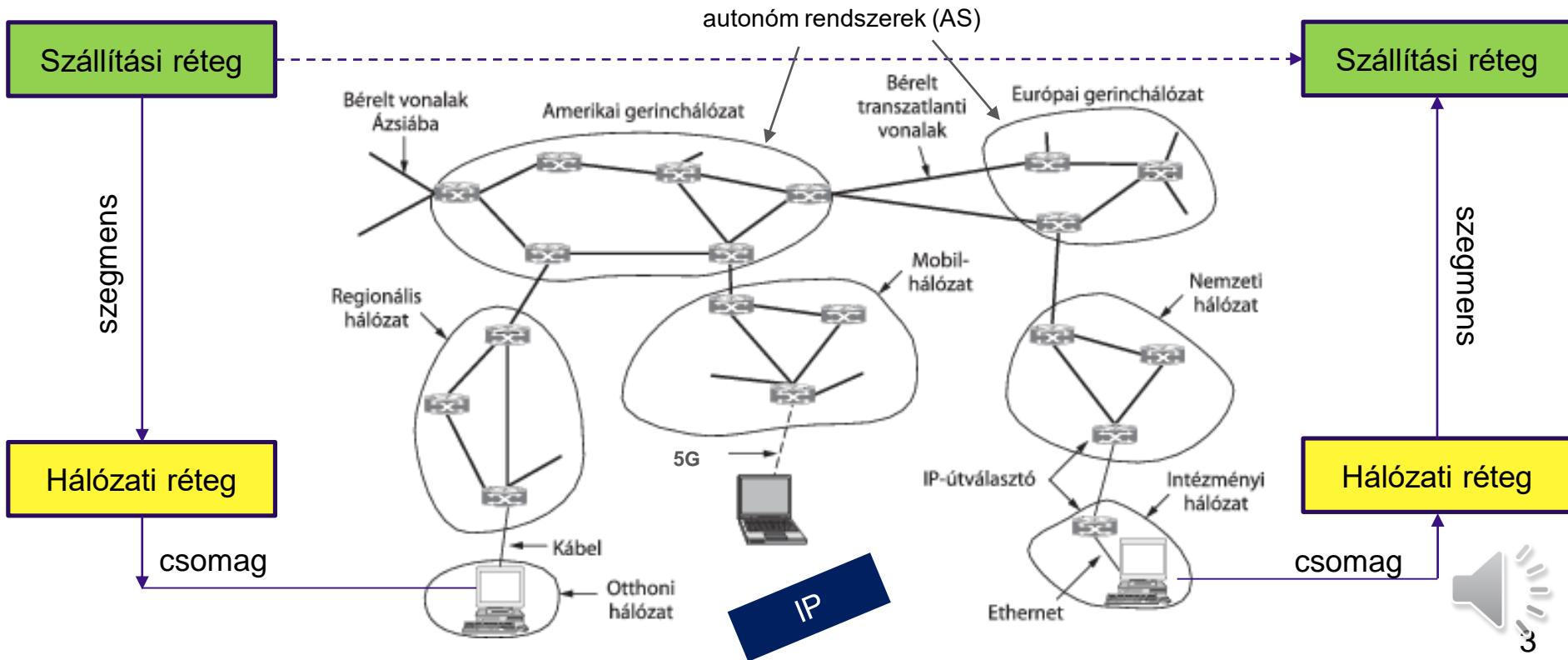
- Az IP protokoll 4-es verziója (IPv4)
  - Az IP protokoll 6-os verziója (IPv6)
  - Internet kontroll protokollok
- 
- Címzés  
Címek típusai  
Alhálózatok  
Útválasztás  
(NAT)



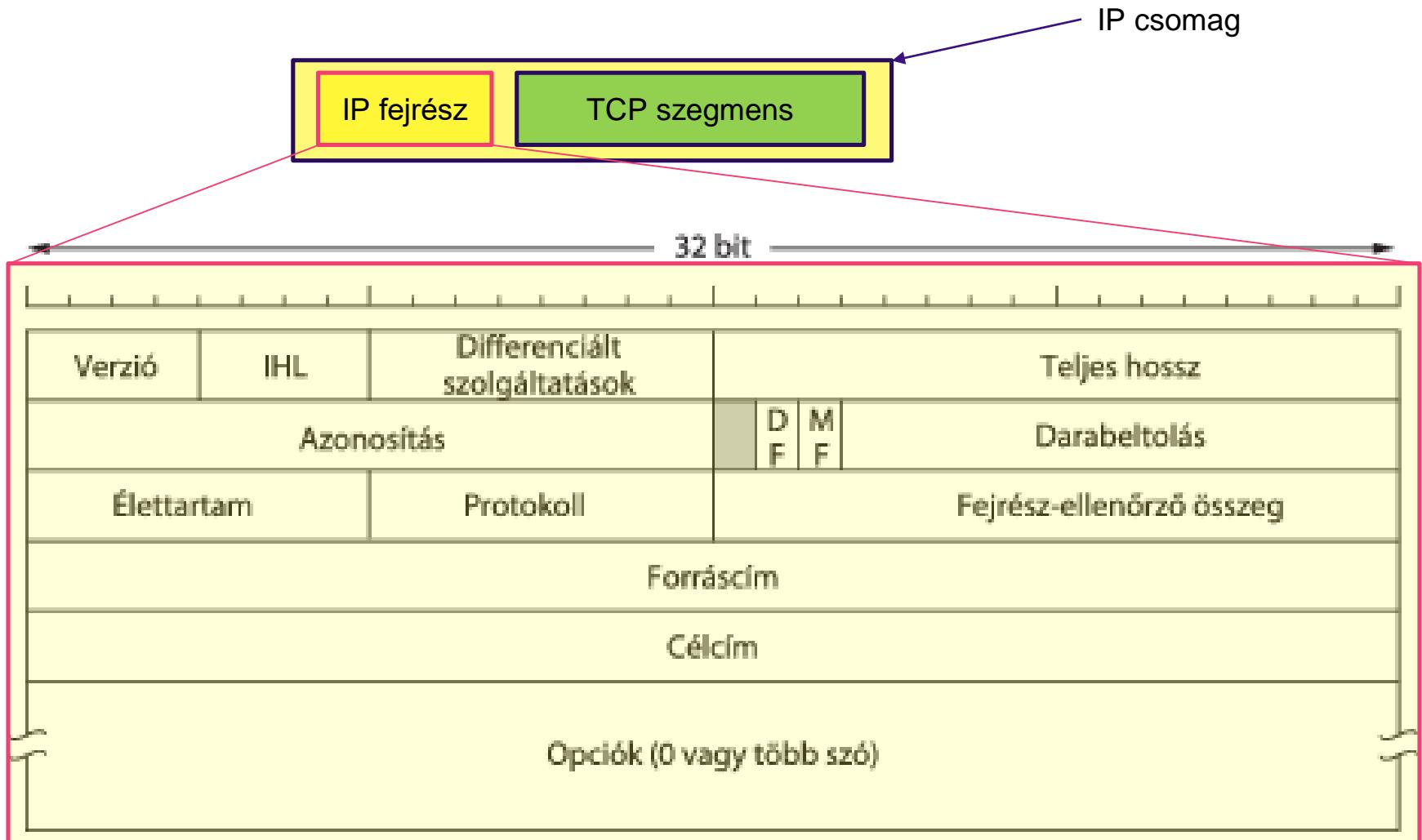
# Hálózati réteg az interneten

## Kialakítás

- Autonóm rendszerek összekapcsolt együttese
- Nincs meghatározott szerkezete
- Gerinchálózatok, regionális hálózatok, ISP-k, LAN-ok, ...



# Az IP protokoll 4-es változata (1) (IPv4)

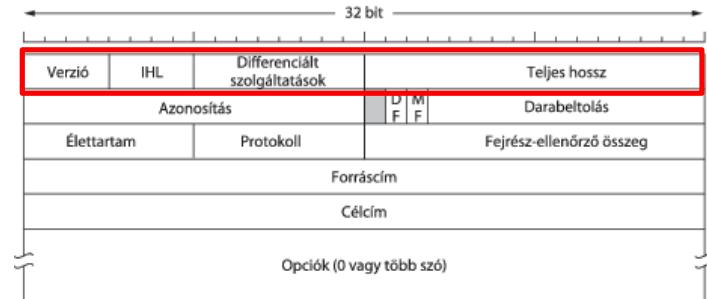


Az IPv4-(internetprotokoll) fejrész

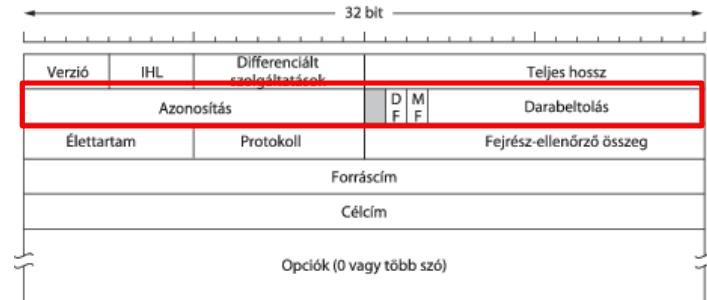


# Az IP protokoll 4-es változata (2) (IPv4)

- Verzió: 4
- IHL:
  - fejrész hossza 32 bites szavakban
- Differenciált szolgáltatások:
  - 6 bit: szolgáltatási osztályok (pl. gyorsított, biztosított)
  - 2 bit: explicit torlódásértesítés
- Teljes hossz:
  - Fejrész + adatrész, max. 65535 bájt



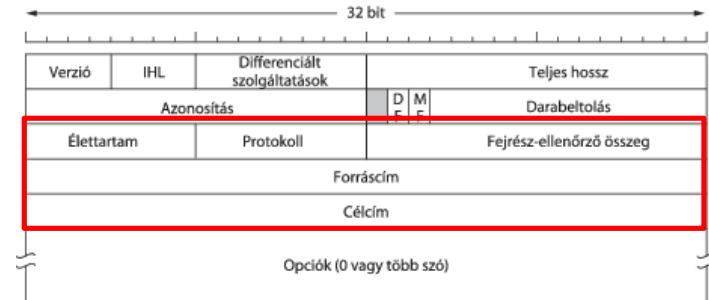
# Az IP protokoll 4-es változata (3) (IPv4)



- Azonosítás:
  - Darabolás esetén a datagramot azonosítja
  - Azonos datagramhoz tartozó darabok azonosítója ugyanaz
- Nem használt bit
- DF (Don't fragment):
  - Darabolás tiltása
  - Hasznos az MTU (legnagyobb átvihető adategység) felderítésére
- MF (More fragments):
  - minden darabban 1, kivéve az utolsót
- Darabeltolás:
  - Darabolás esetén a jelen darab pozíciója a datagramban



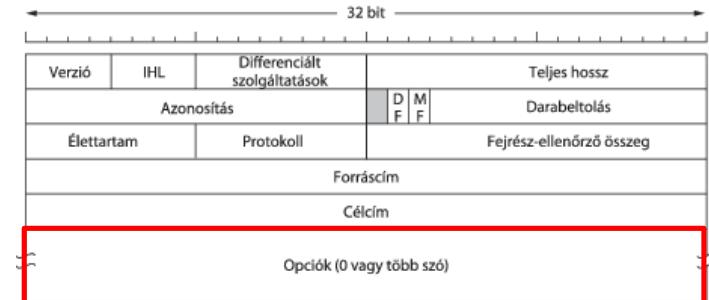
# Az IP protokoll 4-es változata (4) (IPv4)



- Élettartam:
  - Ugrásokat számolja, minden ugrásnál értéke csökken eggyel
  - Ha nulla, a csomagot el kel dobni
  - Megelőzi, hogy hiba esetén a csomagok végtelen ideig kószáljanak
- Protokoll:
  - Melyik szállítási protokollnak kell adni a csomagot?
  - Pl. TCP, UDP
- Fejrész ellenőrző összeg:
  - 16 bites ellenőrző összeg
  - minden ugrásnál számítani kell 😞 (hiszen az élettartam változik)
- Forráscím, célcím:
  - 32 bites IPv4 címek



# Az IP protokoll 4-es változata (5) (IPv4)



- Opciók:
  - Azonosító, hossz, adat
  - Ritkán használt funkciók számára
    - nem kell állandó helyet a fejrészben fenntartani
  - Ritkán használják

Opció	Leírás
Biztonság	Meghatározza, mennyire titkos a datagram
Szigorú formás általi útválasztás	Megadja a teljes követendő utat
Laza forrás általi útválasztás	Felsorolja a felkeresendő útválasztókat
Útvonal feljegyzése	Felszólítás, hogy minden útválasztó füsse hozzá az IP-címét
Időbélyeg	Felszólítás, hogy minden útválasztó füsse hozzá az IP-címét és az időbélyegét



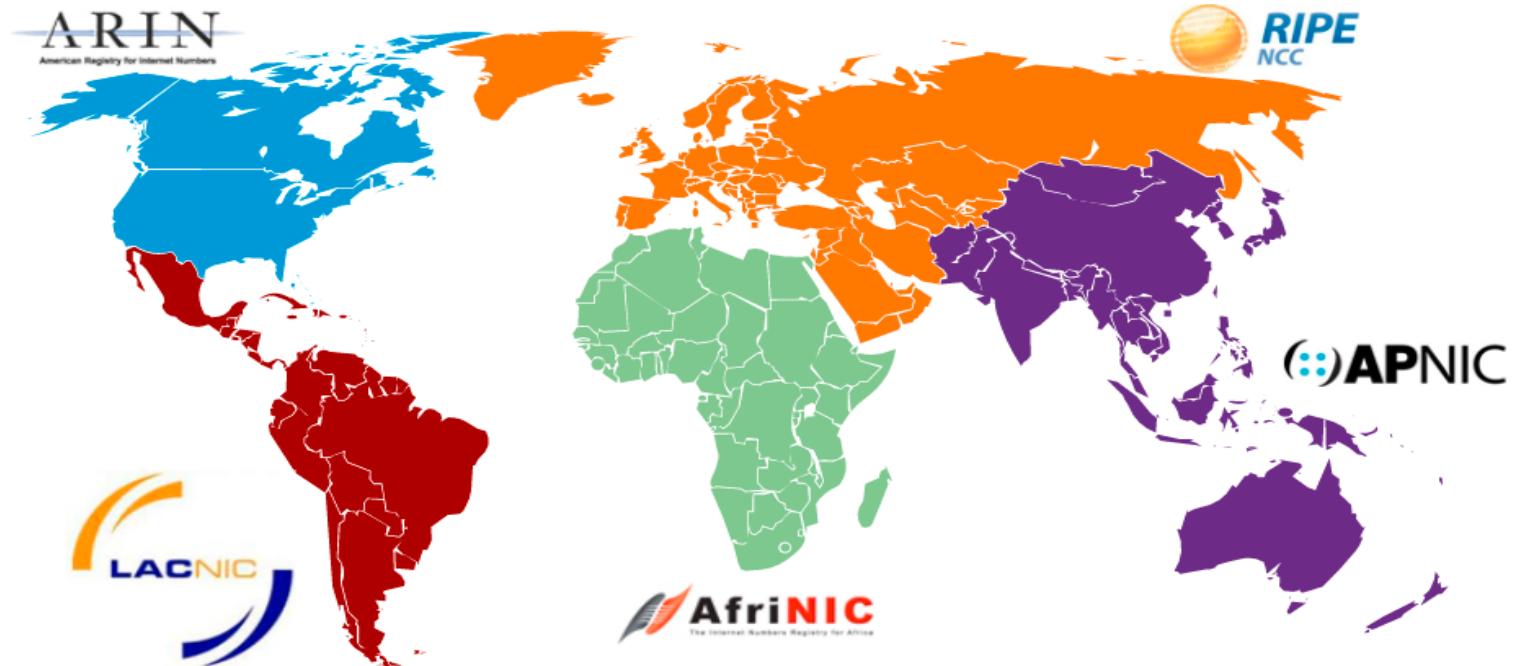
# IP-címzés

- Címek kiosztása
- Előtagok (prefix)
- Alhálózatok
- CIDR—Osztály nélküli körzetek közötti útválasztás
- Osztályalapú címzés (legacy)
- NAT—Hálózati címfordítás



# IPv4 címek kiosztása

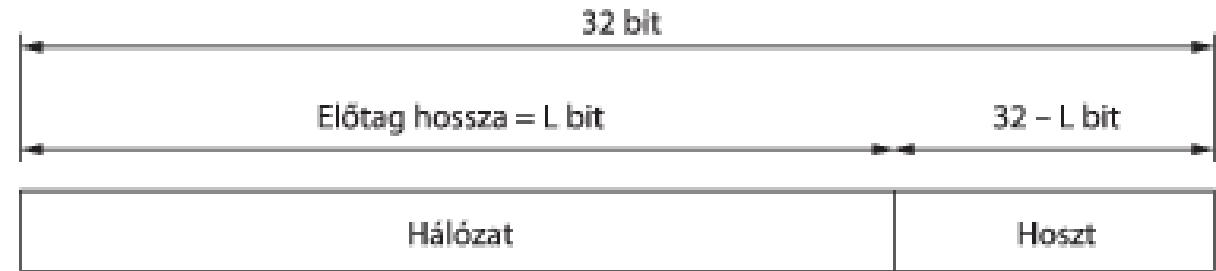
- Központi szervezet: IANA (Internet Assigned Numbers Authority)
- 5 regionális szervezet



- AfriNIC (African Network Information Centre) - Africa Region
- APNIC (Asia Pacific Network Information Centre) - Asia/Pacific Region
- ARIN (American Registry for Internet Numbers) - North America Region
- LACNIC (Regional Latin-American and Caribbean IP Address Registry) - Latin America and some Caribbean Islands
- RIPE NCC (Réseaux IP Européens Network Coordination Centre) - Europe, the Middle East, and Central Asia

# IPv4 címek felépítése

- 32 bites
- Jelölés:
  - Pontokkal elválasztott decimális jelölés. Pl. 193.224.41.159
- Hálózati rész
  - előtag (prefix)
    - A hálózat folyamatos címblokkja
- Állomás (hoszt) rész
- Alhálózati maszk:
  - Egyesek a hálózati rész helyén, nullák az állomás részen
  - Pl.: 255.255.255.0
- Prefix hossz jelölés: egyesek száma az alhálózati maszkban
  - Pl. /24, /25, /8, stb.



# IPv4 címek felépítése

## Példa 1.

- Decimális: 123.45.67.89
- Bináris: 01111011 00101101 01000011 01011001
- Bináris alhálózati maszk: 11111111 11111111 11111111 00000000
- Alhálózati maszk: 255.255.255.0
- Prefix hossz: /24
- Röviden: 123.45.67.89/24
- Hálózat címe: 123.45.67.0
- Első host cím: 123.45.67.1
- Utolsó host cím: 123.45.67.254
- Hálózati broadcast cím: 123.45.67.255
- Hány állomás lehet a hálózaton?  $2^8 - 2 = 254$



# IPv4 címek felépítése

## Példa 2.

- Decimális: 123.45.67.89
- Bináris: 01111011 00101101 01000011 01011001
- Bináris alhálózati maszk: 11111111 11111111 11111111 11000000
- Alhálózati maszk: 255.255.255.192
- Prefix hossz: /26
- Röviden: 123.45.67.89/26
- Hálózat címe: 123.45.67.64
- Első host cím: 123.45.67.65
- Utolsó host cím: 123.45.67.126
- Hálózati broadcast cím: 123.45.67.127
- Hány állomás lehet a hálózaton?  $2^6 - 2 = 62$



# IPv4 címek felépítése

## Példa 3.

- Decimális: 123.45.67.89
- Bináris: 01111011 00101101 01000011 01011001
- Bináris alhálózati maszk: 11111111 11110000 00000000 00000000
- Alhálózati maszk: 255.240.0.0
- Prefix hossz: /12
- Röviden: 123.45.67.89/12
- Hálózat címe: 123.32.0.0
- Első host cím: 123.32.0.1
- Utolsó host cím: 123.47.255.254
- Hálózati broadcast cím: 123.47.255.255
- Hány állomás lehet a hálózaton?  $2^{20} - 2 = 1.048.574$



# IPv4 címek felépítése

## Példa 4.

- Decimális: 123.45.67.89
- Bináris: 01111011 00101101 01000011 01011001
- Bináris alhálózati maszk: 11111111 00000000 00000000 00000000
- Alhálózati maszk: 255.0.0.0
- Prefix hossz: /8
- Röviden: 123.45.67.89/8
- Hálózat címe: 123.0.0.0
- Első host cím: 123.0.0.1
- Utolsó host cím: 123.255.255.254
- Hálózati broadcast cím: 123.255.255.255
- Hány állomás lehet a hálózaton?  $2^{24} - 2 = 16.777.214$



# IPv4 címek felépítése

## Példa 5.

- Decimális: 123.45.67.89
- Bináris: 01111011 00101101 01000011 01011001
- Bináris alhálózati maszk: 11111111 11111111 00000000 00000000
- Alhálózati maszk:
- Prefix hossz:
- Röviden:
- Hálózat címe:
- Első host cím:
- Utolsó host cím:
- Hálózati broadcast cím:
- Hány állomás lehet a hálózaton?



# IPv4 címek felépítése

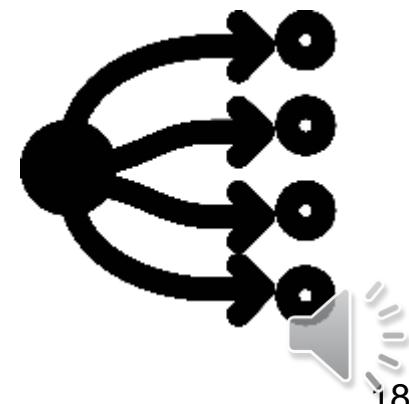
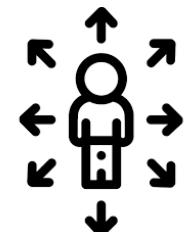
## Példa 6.

- Decimális:
- Bináris:
- Bináris alhálózati maszk:
- Alhálózati maszk:
- Prefix hossz:
- Röviden: **193.44.73.12/26**
- Hálózat címe:
- Első host cím:
- Utolsó host cím:
- Hálózati broadcast cím:
- Hány állomás lehet a hálózaton?



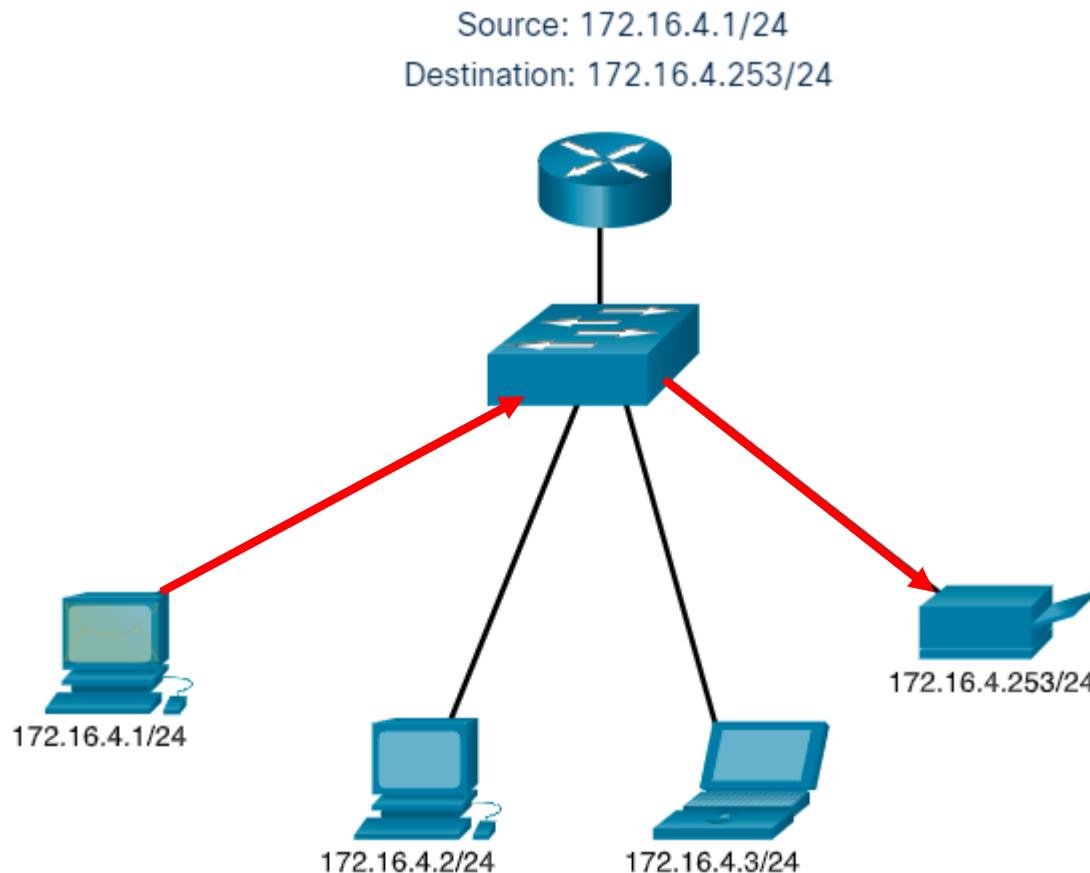
# IPv4 címek típusai (1)

- Unicast (egyesküldés)
  - Egyetlen címzett van
  - PI: 192.168.1.18
- Broadcast (üzenetszórás)
  - mindenki címzett az adott hálózaton
  - A lokális hálózaton: 255.255.255.255
  - Célzott broadcast: távoli hálózati cím + csupa 1 bit
    - PI. 192.168.1.255 (/24)
    - Ezt általában tiltják
- Multicast (többesküldés)
  - Egy forrásból egy csoportba küldhető üzenet
  - Fel kell „iratkozni” a csoportba
  - 224.0.0.0 és 239.255.255.255 közötti tartomány van fenntartva multicast címekre



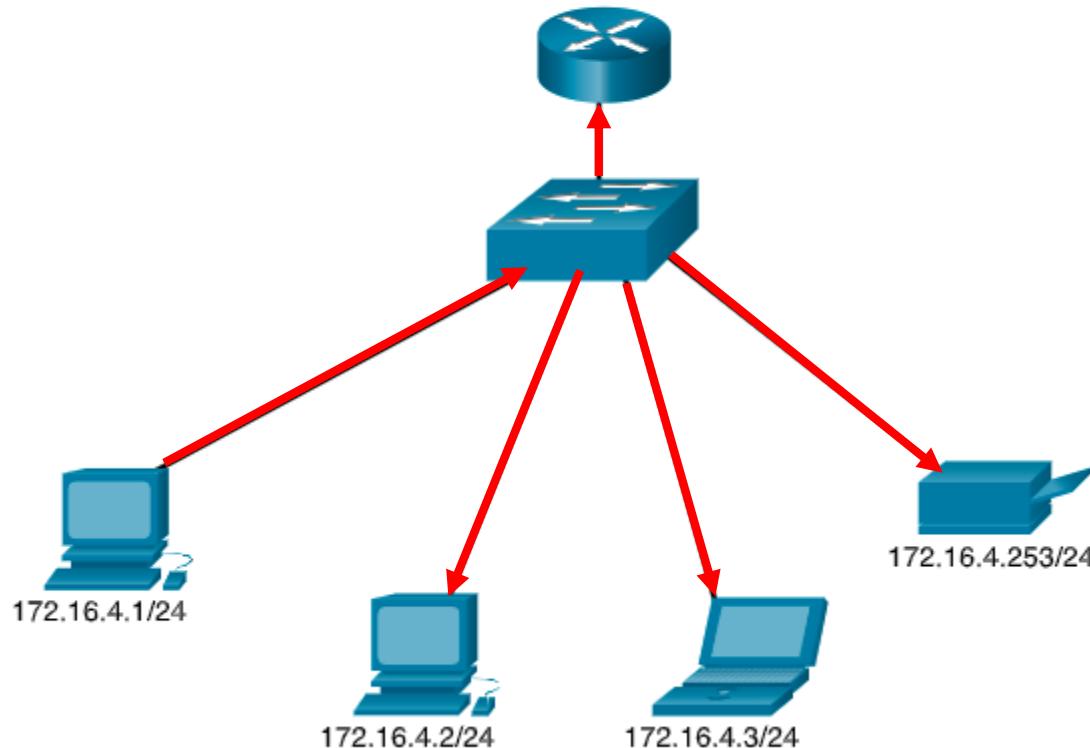
# IPv4 címek típusai (2)

## Unicast



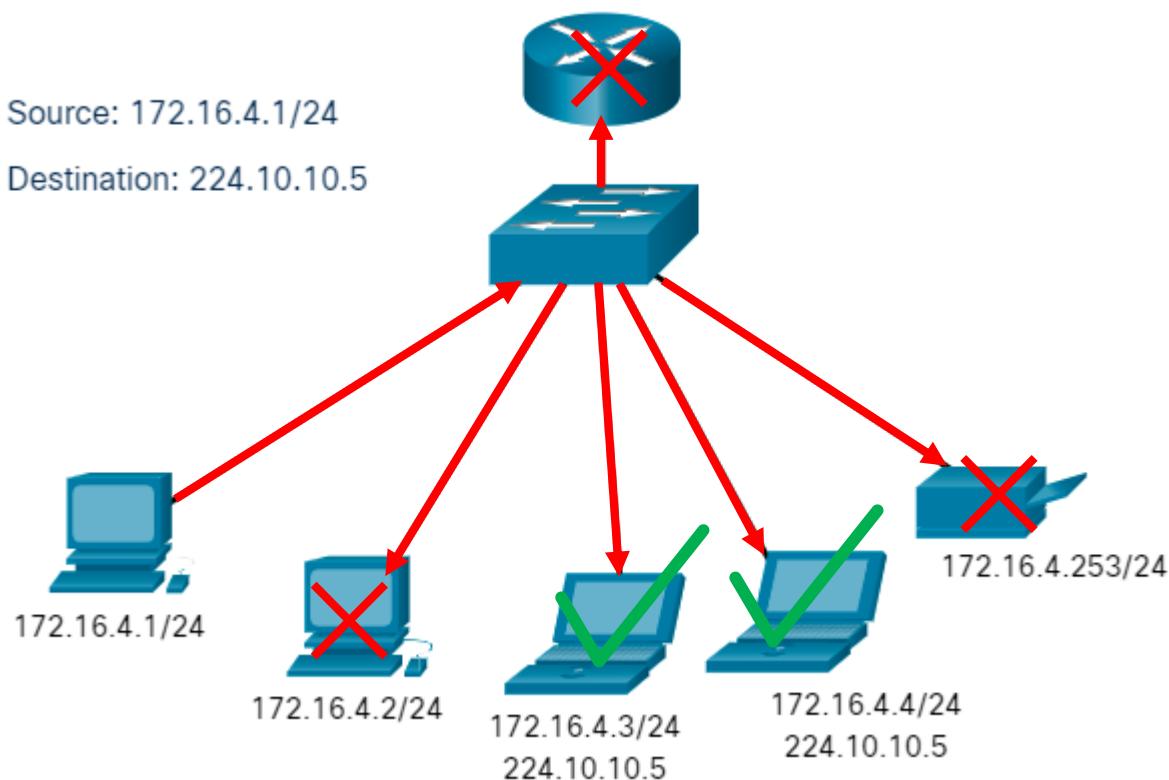
# IPv4 címek típusai (3) Broadcast

Limited Broadcast  
Source: 172.16.4.1/24  
Destination: 255.255.255.255



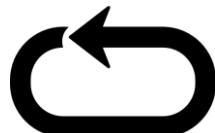
# IPv4 címek típusai (3)

## Multicast



# Speciális IPv4 címek

- Privát IP címek
  - Hálózaton belüli címzésre
    - 10.0.0.0/8 (10.0.0.0 - 10.255.255.255)
    - 172.16.0.0/12 (172.16.0.0 - 172.31.255.255)
    - 192.168.0.0/16 (192.168.0.0 - 192.168.255.255)
  - Hálózatok között ilyenkor NAT kell
- Loopback címek
  - 127.0.0.0 /8 (127.0.0.1 - 127.255.255.254)
  - Címzett: a küldő hoszt
  - Tesztelésre használjuk
- Link-local (autokonfigurációs) címek
  - 169.254.0.0 /16 (169.254.0.1 - 169.254.255.254)
  - Automatic Private IP Addressing (APIPA)
  - Pl. Windows DHCP kliens használja, ha nincs elérhető DHCP szerver



# Alhálózatok (1)

- Egy rendelkezésre álló címblokk további részekre (alhálózatokra) bontható
- Pl. Egy egyetemnek rendelkezésére áll egy /16 címtér: 128.208.0.0/16
  - Ez 16 bites host címet jelent  $\rightarrow 2^{16} - 2 = 65534$  darab host
  - Ezt 3 tanszék részére kell felosztani (3 alhálózat):
    - Számítástudományi Tanszék (SZT) : /17 (címek fele)
    - Villamosmérnöki Tanszék (VT) : /18 (címek negyede)
    - Bölcsészettudományi Tanszék (BT) : /19 (címek nyolcada)
    - Nincs kiosztva (tartalék): címek nyolcada

Számítástudományi Tanszék:	10000000	11010000	1 xxxxxx	xxxxxxxx	32768-2 db cím
(Maradék:	10000000	11010000	0 xxxxxx	xxxxxxxx)	32768-2 db cím
Villamosmérnöki Tanszék:	10000000	11010000	00 xxxxxx	xxxxxxxx	16384-2 db cím
(Maradék:	10000000	11010000	01 xxxxxx	xxxxxxxx)	16384-2 db cím
Bölcsészettudományi Tanszék:	10000000	11010000	011 xxxx	xxxxxxxx	8192-2 db cím
(Tartalék:	10000000	11010000	010 xxxx	xxxxxxxx)	8192-2 db cím

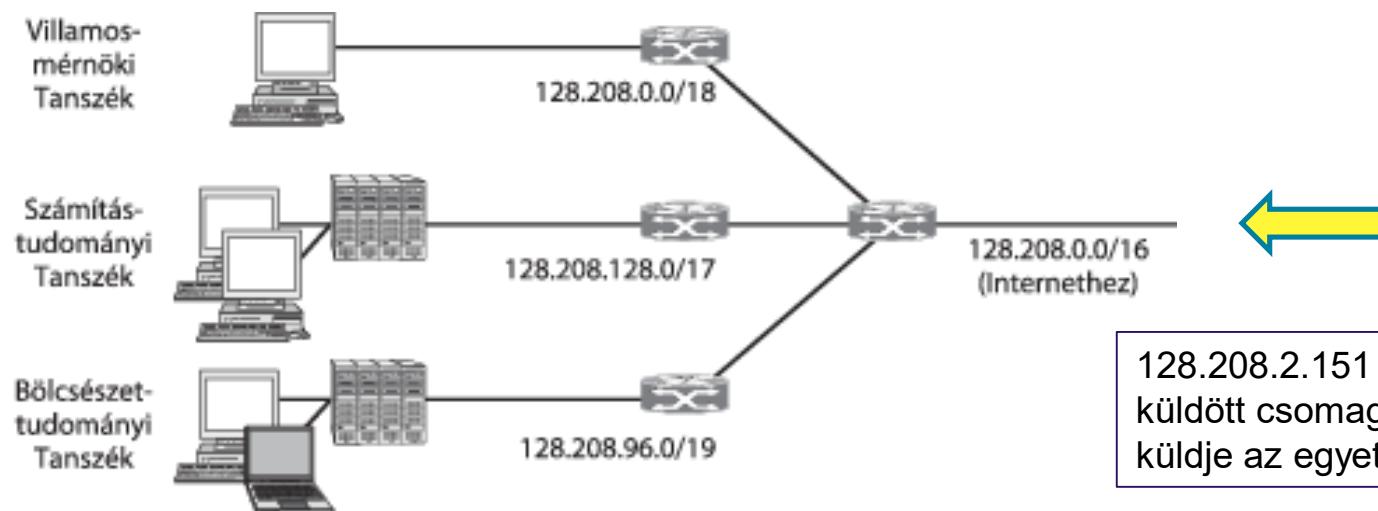


Az alhálózatok megkülönböztetését végző biteket a hoszt címbitekből „kölcsönöztük”



# Alhálózatok (2)

Számítástudományi Tanszék:	10000000	11010000	1 xxxxxx	xxxxxxxx	32768-2 db cím
Alhálózati maszk:	11111111	11111111	10000000	00000000	
Alhálózati cím:	<b>128.208.128.0/17</b>				
Villamosmérnöki Tanszék:	10000000	11010000	00 xxxxxx	xxxxxxxx	16384-2 db cím
Alhálózati maszk:	11111111	11111111	11000000	00000000	
Alhálózati cím:	<b>128.208.0.0/18</b>				
Bölcsészettudományi Tanszék:	10000000	11010000	011 xxxx	xxxxxxxx	8192-2 db cím
Alhálózati maszk:	11111111	11111111	11100000	00000000	
Alhálózati cím:	<b>128.208.96.0/19</b>				



# Alhálózatok (3)

Számítástudományi Tanszék: 10000000 11010000 1|xxxxxx xxxxxxxx 32768-2 db cím

Alhálózati maszk: 11111111 11111111 10000000 00000000

Alhálózati cím: 192.208.128.0/17

Villamosmérnöki Tanszék:

Alhálózati maszk:

Alhálózati cím:

Bölcsészettudományi Tanszék:

Alhálózati maszk:

Alhálózati cím:

SZT (192.208.128.0/17)?

IP cím: 128.208.2.151

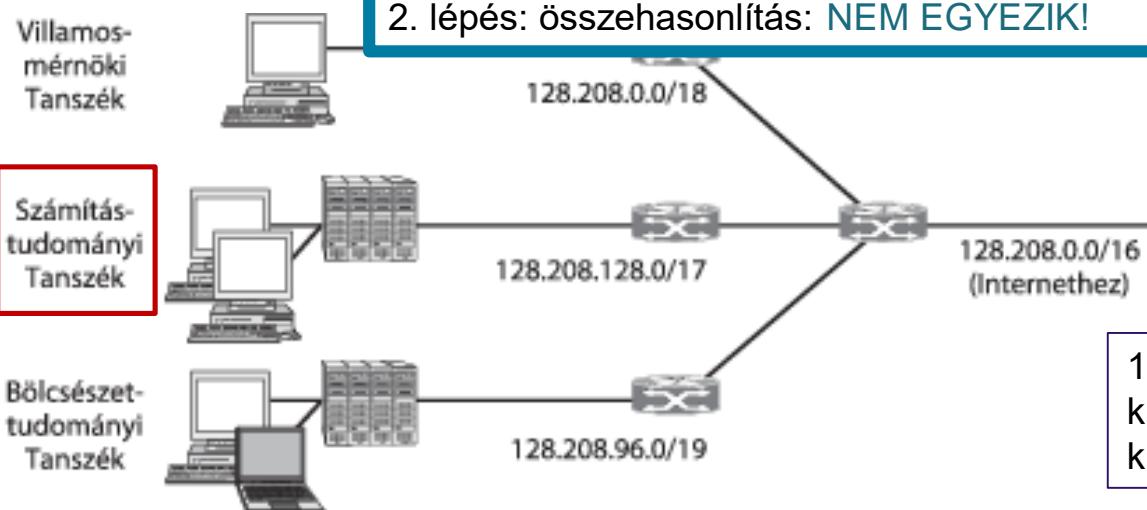
IP cím bináris: 10000000 11010000 00000010 10010111

Maszk (17 bit): 11111111 11111111 10000000 00000000

1. lépés: maszkolás: 10000000 11010000 00000000 00000000

Hálózati cím: 10000000 11010000 10000000 00000000

2. lépés: összehasonlítás: NEM EGYEZIK! ^



Számítástudományi Tanszék

128.208.2.151 IP címre  
küldött csomagot hová  
küldjük?



# Alhálózatok (4)

BT (192.208.96.0/19)?

Számítástudományi Tanszék:

10

IP cím: 128.208.2.151

Alhálózati maszk:

11

IP cím bináris: 10000000 11010000 00000010 10010111

Alhálózati cím:

19

Maszk (19 bit): 11111111 11111111 11100000 00000000

Villamosmérnöki Tanszék:

10

1. lépés: maszkolás: 10000000 11010000 00000000 00000000

Alhálózati maszk:

11

Hálózati cím: 10000000 11010000 01100000 00000000

Alhálózati cím:

19

2. lépés: összehasonlítás: NEM EGYEZIK! ^

Bölcsészettudományi Tanszék: 10000000 11010000 011|xxxxx xxxxxxxx 8192-2 db cím

Alhálózati maszk:

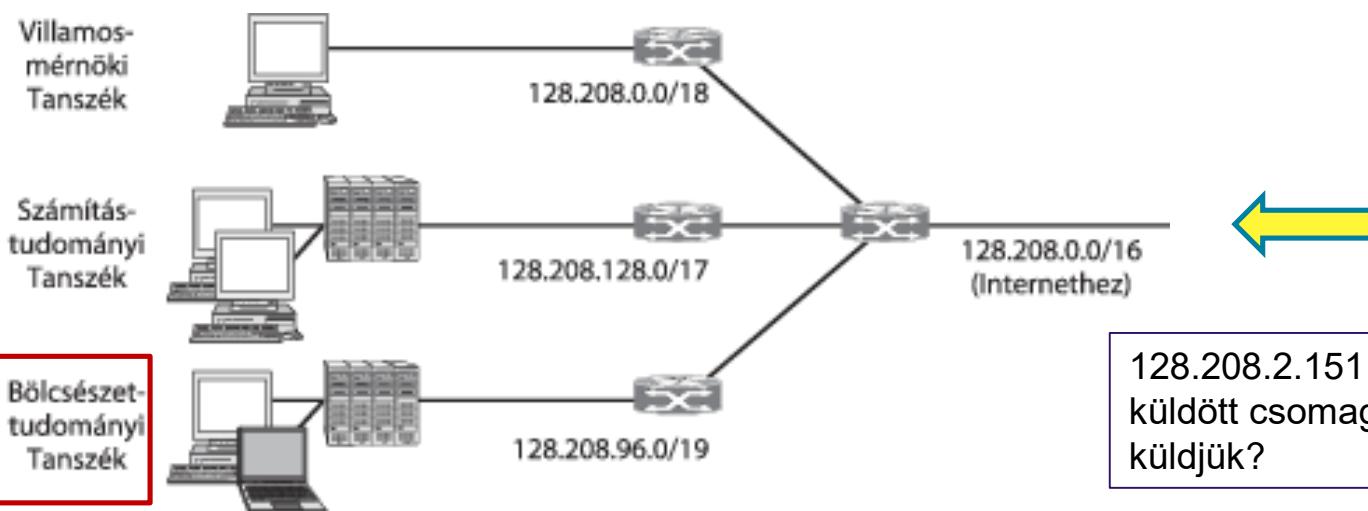
11111111

11111111 11100000

00000000

Alhálózati cím:

192.208.96.0/19



128.208.2.151 IP címre  
küldött csomagot hová  
küldjük?



# Alhálózatok (5)

VT (192.208.0.0/18)?

Számítástudományi Tanszék: 10

Alhálózati maszk: 11

Alhálózati cím: 192.208.128.0/17

Villamosmérnöki Tanszék: 10000000 11010000 00|xxxxxx xxxxxxxx

Alhálózati maszk: 11111111 11111111 11000000 00000000

Alhálózati cím: 192.208.0.0/18

Bölcsészettudományi Tanszék: 10000000 11010000 011|xxxxx xxxx|xxxxx

Alhálózati maszk: 11111111 11111111 11100000 00000000

Alhálózati cím: 192.208.96.0/19

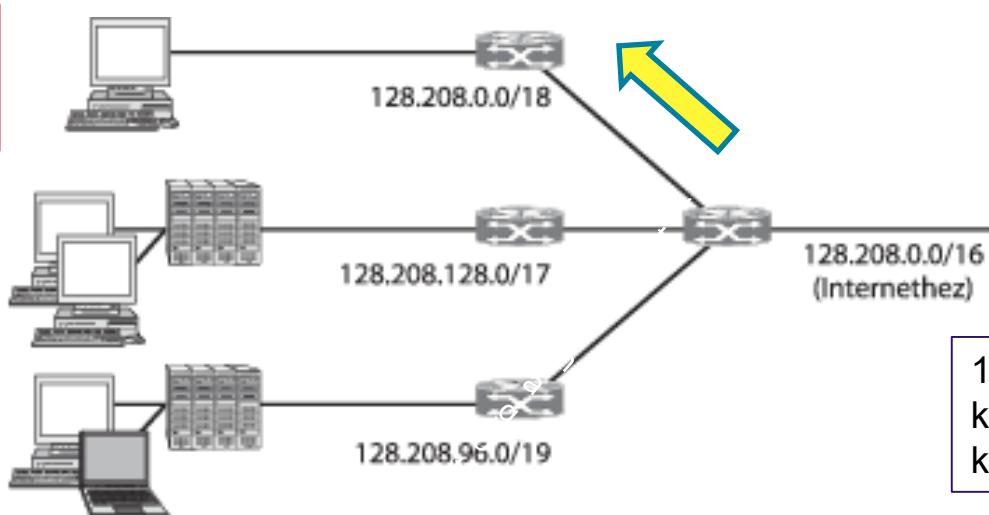
IP cím:	128.208.2.151	10000000	11010000	00000010	10010111
IP cím bináris:	10000000 11010000 00000010 10010111				
Maszk (18 bit):	11111111 11111111 11000000 00000000				
1. lépés: maszkolás:	10000000 11010000 00000000 00000000				
Hálózati cím:	10000000 11010000 00000000 00000000				
2. lépés: összehasonlítás:	EGYEZIK!				



Villamos-  
mérnöki  
Tanszék

Számítá-  
studományi  
Tanszék

Bölcsézet-  
tudományi  
Tanszék



128.208.2.151 IP címre  
küldött csomagot hová  
küldjük?



# Alhálózatok (6)

## Gyakorlás

Számítsuk ki minden egyik alhálózaton az első és utolsó érvényes host-címet és a broadcast címet!

Számítástudományi Tanszék: **10000000 11010000 1|xxxxxx xxxxxxxx 32768-2 db cím**

Alhálózati maszk: **11111111 11111111 1**

Alhálózati cím: **192.208.128.0/17**

Villamosmérnöki Tanszék: **10000000 11010000 00|xxxxx xxxxxxxx 16384-2 db cím**

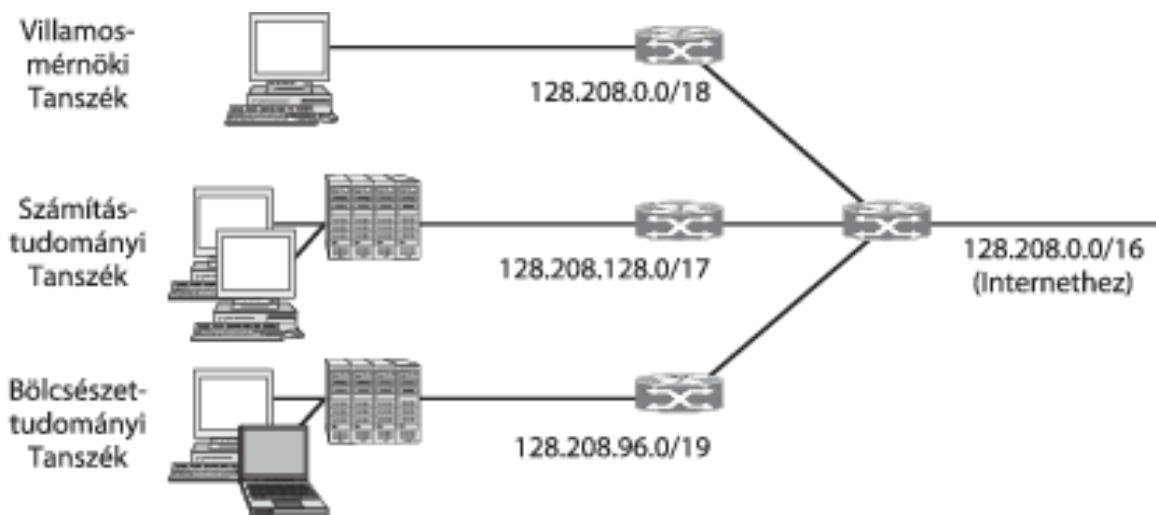
Alhálózati maszk: **11111111 11111111 11**

Alhálózati cím: **192.208.0.0/18**

Bölcsészettudományi Tanszék: **10000000 11010000 011|xxxxx xxxxxxxx 8192-2 db cím**

Alhálózati maszk: **11111111 11111111 111**

Alhálózati cím: **192.208.96.0/19**



# Útválasztás (1)

- Mit kell tárolni egy hálózati útválasztónak, ha az egyetem címeit akarja kezelni?

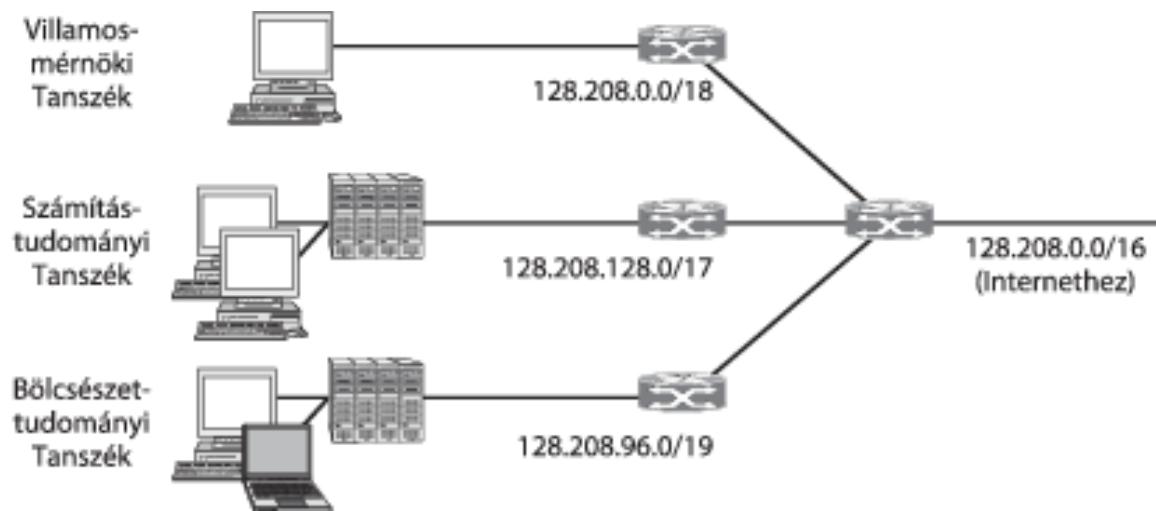
## A. megoldás

Minden tanszék címét tárolja:  
128.208.0.0/18      Seattle  
128.208.128.0/17      Seattle  
128.208.96.0/19      Seattle



## B. megoldás

Az egyetem egyetlen címét tárolja:  
128.208.0.0/16      Seattle



# Útválasztás (2)

- Mit kell tárolni egy New York-i hálózati útválasztónak, ha az alábbi egyetemek címeit akarja kezelni?

Egyetem	Első cím	Utolsó cím	Hány hosszú?	Előtag
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Nem használt)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Cambridge:

11000010 00011000 00000000 00000000

Edinburgh:

11000010 00011000 00001000 00000000

Nem használt:

11000010 00011000 00001100 00000000

Oxford:

11000010 00011000 00010000 00000000

csoportosított előtag



194.24.0.0/19



# Útválasztás (3)

- Mit kell tárolni egy New York-i hálózati útválasztónak, ha az alábbi egyetemek címeit akarja kezelni?

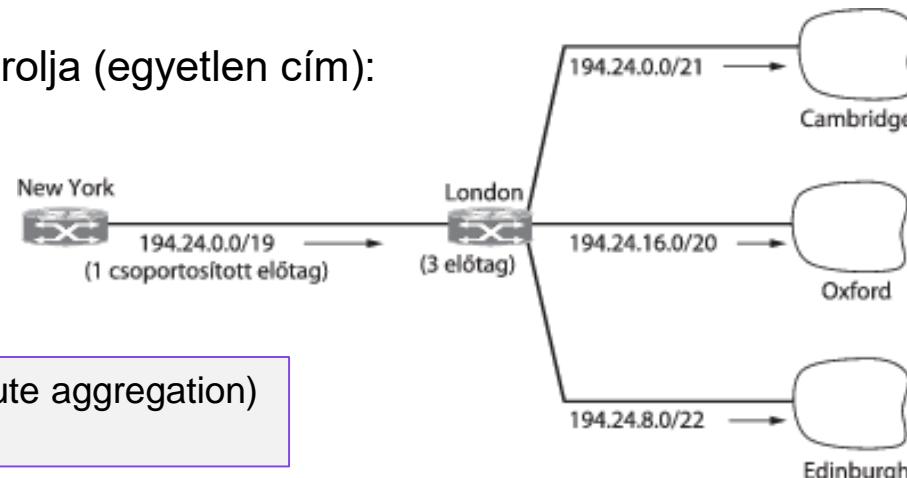
Egyetem	Első cím	Utolsó cím	Hány hoszt?	Előtag
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Nem használt)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

194.24.0.0/19

- Csak a **csoportosított előtagot** tárolja (egyetlen cím):

194.24.0.0/19

London



Ez a módszer az **útvonal-csoportosítás** (route aggregation)  
Bővebb előtag: **szuperhálózat**



# Útválasztás (4)

- Mit kell tárolni egy New York-i hálózati útválasztónak, ha a szabad címtartományt San Francisco kapja meg?
- A 194.24.0.0/19 San Francisco-t is tartalmazza. Ezt nem szabad Londonba küldeni!

## A. megoldás

Mind a 4 tartományt tárolja:

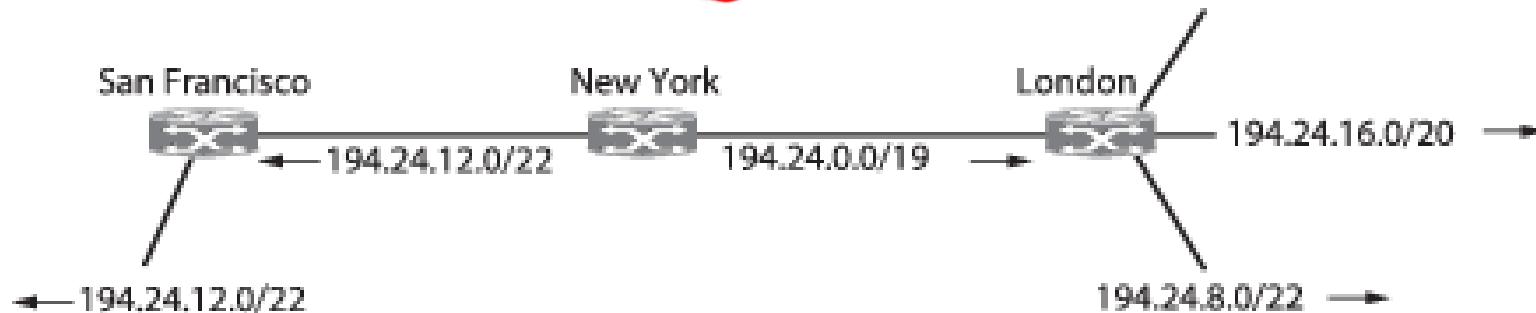
194.24.0.0/21	London
194.24.16.0/20	London
194.24.8.0/22	London
194.24.12.0/22	San Francisco



## B. megoldás

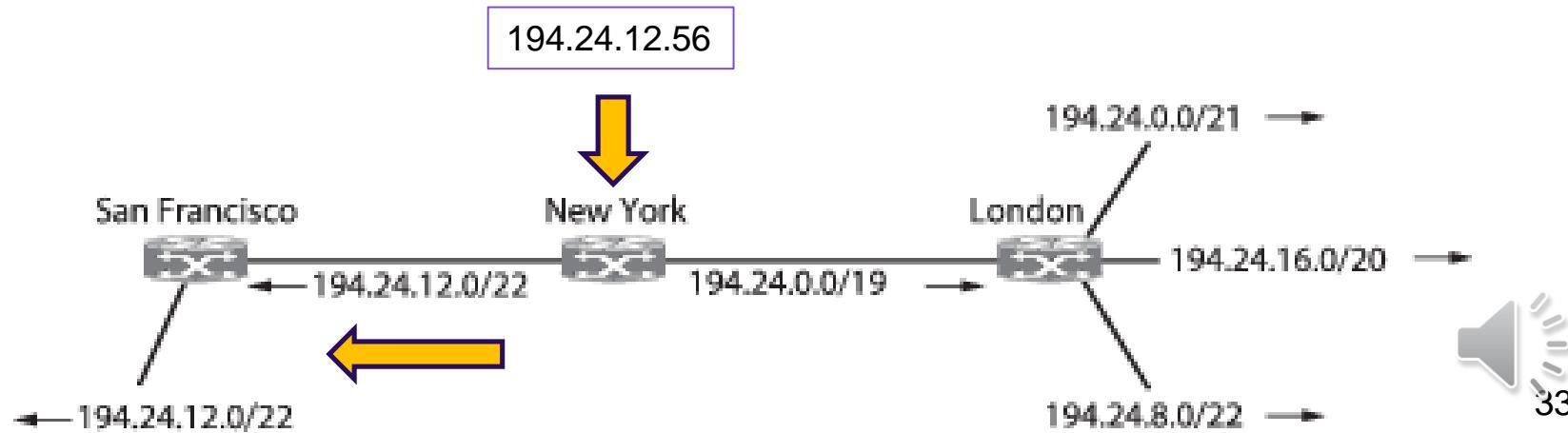
A csoportosított címet és a kivételt tárolja:

194.24.0.0/19	London
194.24.12.0/22	San Francisco



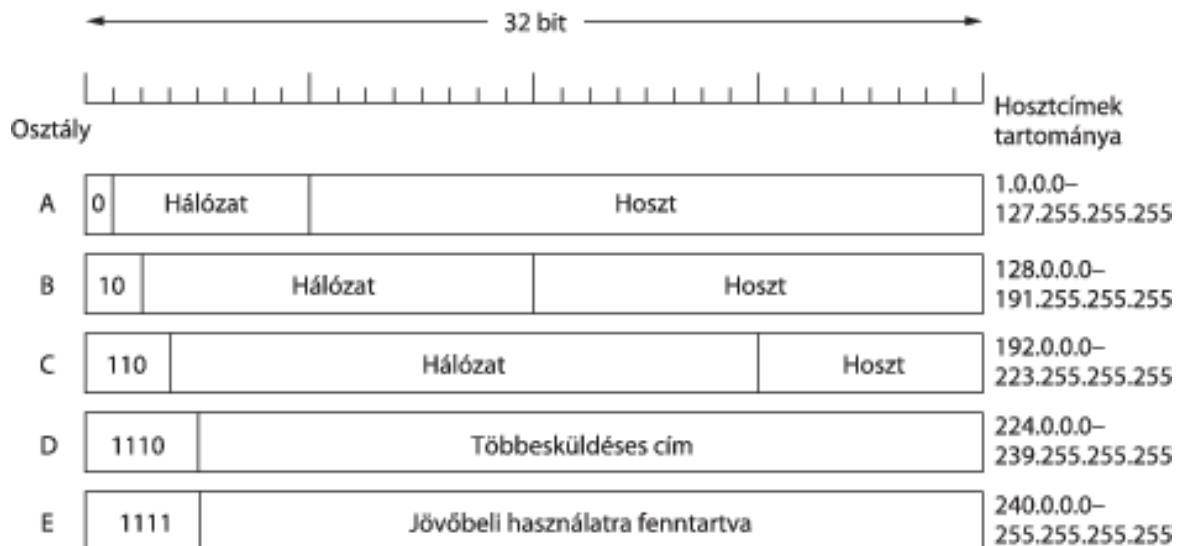
# Útválasztás (5)

- **Leghosszabb egyező előtag** (longest prefix) alapú útválasztás
- Elvi működés:
  - Csomag beérkezik, a címe A
  - Útválasztó megnézi, hogy az A cím melyik bejegyzésére illeszkedik
  - Ha több ilyen bejegyzés is van, akkor kiválasztja azt, amelyik a leghosszabb előtaggal bír
    - Pl. ha egy /22 és egy /19 bejegyzésre is illeszkedik, akkor a /22 bejegyzést használja
  - A bejegyzésnek megfelelő irányba küldi tovább a csomagot



# Útválasztás (5)

- Nincsenek előre meghatározott előtag-méretek
  - Pl. a /20, /21, /22 előtagokból a router készített egy /19 egyesített előtagot
- Ezt nevezzük osztály nélküli körzetek közötti útválasztásnak
  - **CIDR** - Classless InterDomain Routing
- Korábban alkalmaztak fix osztályokat (osztályalapú címzés):

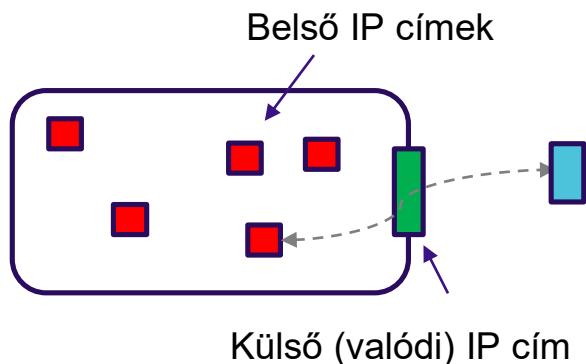


# NAT – hálózati címfordítás (1)

- Kevés az IPv4 cím
- A szolgáltató csak 1 címet ad az ügyfélnek
- Az ügyfélnek sok címre lenne szüksége
- Megoldás: NAT (Network Address Translation)
  - Hálózaton belül egyedi címek
  - Privát címtartományból:
    - 10.0.0.0 – 10.255.255.255/8 (16 777 216 hoszt)
    - 172.16.0.0 – 172.31.255.255/12 (1 048 576 hoszt)
    - 192.168.0.0 – 192.168.255.255/16 (65 536 hoszt)
- Csak korlátozásokkal működik
  - TCP, UDP

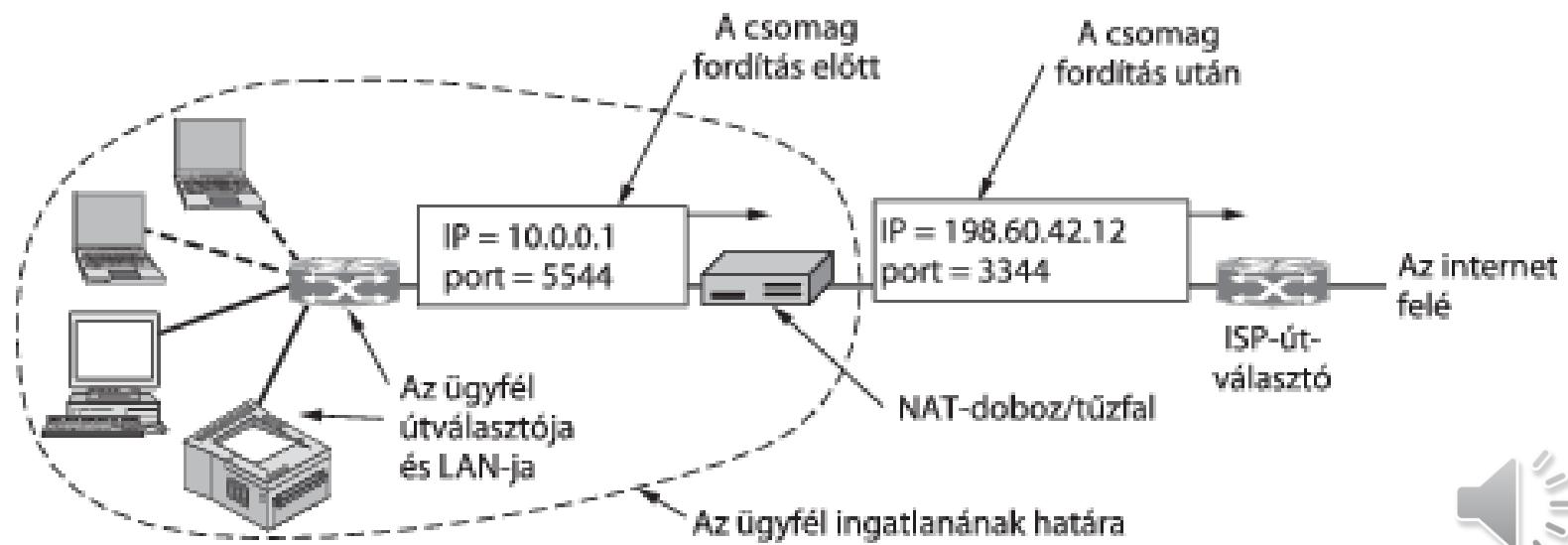
# NAT – hálózati címfordítás (2)

- NAT doboz fordít a belső és külső címek között
- TCP: IP cím és port cím (pl: 123.45.67.89:9876)
  - Analógia: telefonközpont: IP cím, port: belső mellék
- NAT fordítótábla: fiktív forrás port – belső IP cím – (valódi) forrás port
  - PI: 3344 – 10.0.0.1 – 5544
- Kifelé menő üzenet: **B:Pb → S:Ps**
  - B: belső forrás IP, Pb: forrás port, S: cél (szerver) IP, Ps: szerver port
  - Új fiktív port: Px
  - Cím csere: belső IP helyett külső IP (**C**), forrás port helyett fiktív port (**C:Px** → **S:Ps**)
  - Táblába bejegyzés: **Px – B – Pb**
- Befelé jövő üzenet: **S:Ps → C:Px**
  - Táblában keresés: **Px** bejegyzése: **B – Pb**
  - Cím csere a tábla bejegyzése szerint: **S:Ps → B:Pb**



# NAT – hálózati címfordítás (3)

- Külső IP cím: 198.60.42.12
- Kifelé menő üzenet: 10.0.0.1:5544 → 220.43.12.4:7788
  - Új fiktív port: 3344
  - Cím csere: 198.60.42.12:3344 → 220.43.12.4:7788
  - Táblába bejegyzés: 3344 – 10.0.0.1 – 5544
- Befelé jövő üzenet: 220.43.12.4:7788 → 198.60.42.12:3344
  - Táblában keresés: 3344 – 10.0.0.1 – 5544
  - Cím csere: 220.43.12.4:7788 → 10.0.0.1:5544



# NAT – hálózati címfordítás (4)

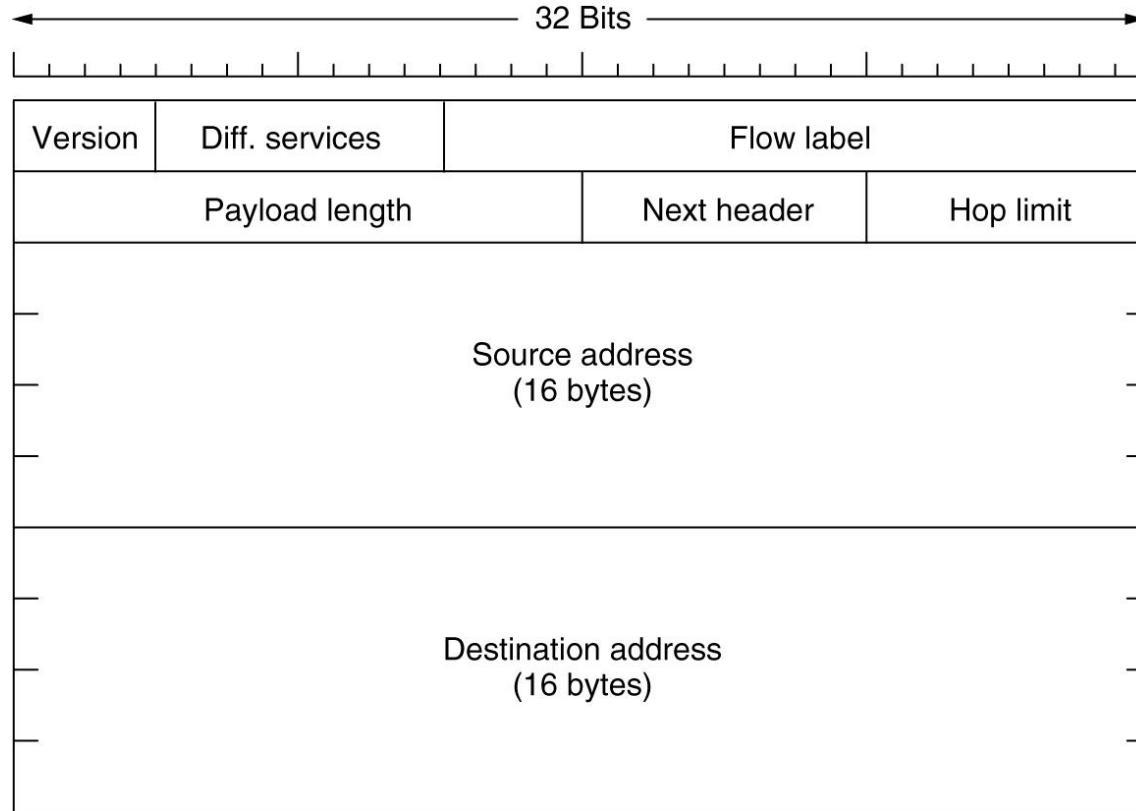
Problémák:

- IP címek nem globálisan egyediek
  - pl. sok 10.0.0.1 cím a hálózaton
- Nem tud bárki bárkinek küldeni
  - NAT mögül csak kezdeményezni lehet
  - További trükkök kellenek ennek áthidalására
- Az összeköttetés nélküli internetbe összeköttetés-alapú kapcsolatot kever
  - NAT tábla sérülése: összeomlás
- Összemossa a 3. és 4. réteget
  - portszámok: 4. réteg
- TCP-n és UDP-n kívül más szállítási protokollok is vannak...

# IPv6

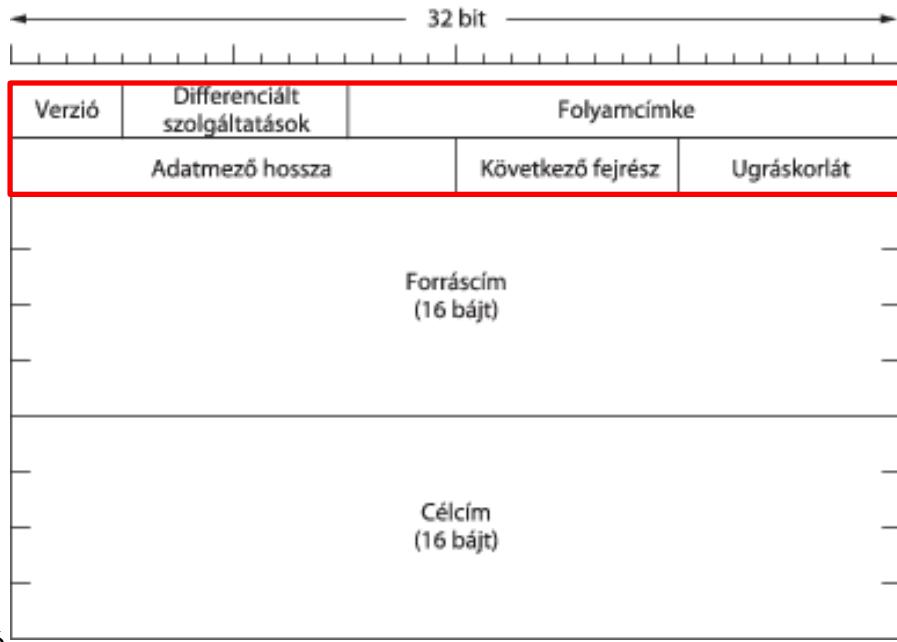
- Fő különbségek IPv4 és IPv6 között:
  - Sokkal több IP cím van (32 bites helyett 128 bites cím)
  - Fejrész egyszerűbb lett (13 mezőből 7 lett)
  - Opciók jobb támogatása
    - Szükségszerű a rövidebb fejrész miatt
    - Gyorsabb csomagfeldolgozást tesz lehetővé
  - Biztonság javítása
  - Szolgáltatásminőség nagyobb hangsúlyt kapott
    - Multimédia

# The Main IPv6 Header



The IPv6 fixed header (required)

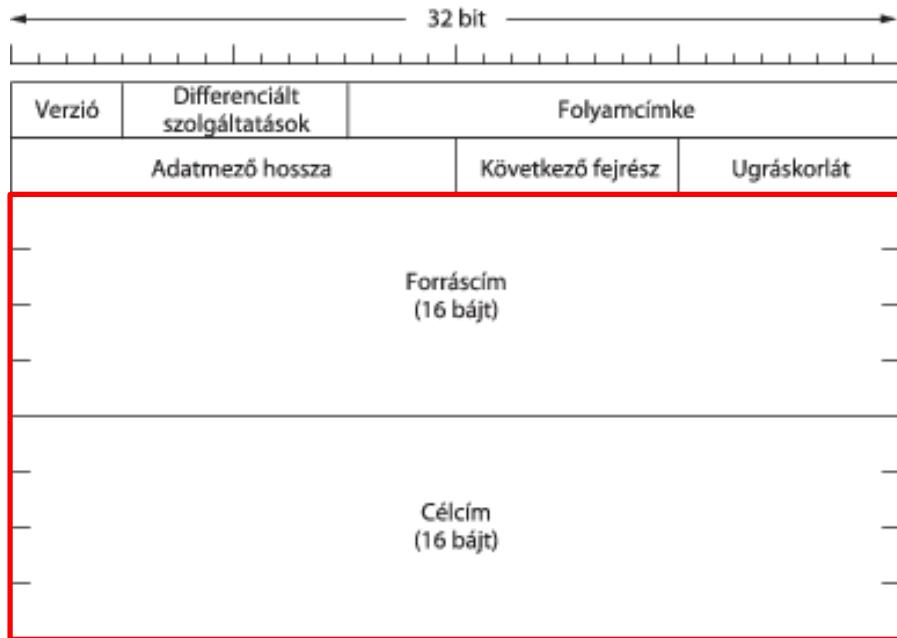




# A fő IPv6 fejrész

- Verzió: 6
- Differenciált szolgáltatások:
  - hasonlóan az IPv4-hez.
  - torlódásjelzés, prioritás
- Folyamcímke:
  - Virtuális-áramkör alapú megközelítést tesz lehetővé
  - Folyamot előre fel lehet állítani, ez az azonosítója
  - minden útválasztó kikeresi táblázatából, hogy a címke milyen különleges elbánást igényel
- Adatmező hossza:
  - Fejrész utáni méret
- Következő fejrész:
  - Jelzi, hogy van-e következő opcionális fejrész, és milyen típusú
  - Ha ez az utolsó IP fejrész, akkor itt jelzi, hogy melyik szállítási protokollnak kell a csomagot adni (TCP, UDP)
- Ugráskorlát:
  - Mint az IPv4 élettartam





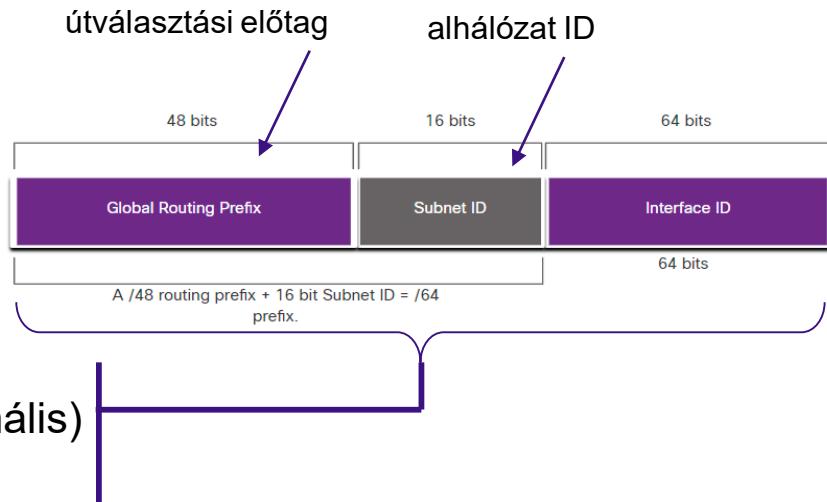
# IPv6 címek

- Forrás cím, célcím
  - 128 bit (16 bájt)
  - Formátum:
    - 8 csoport
    - Kettősponttal elválasztva
    - Mindegyik csoportban 4-4- hexadecimális számjegy (hextet)
    - Pl.:8000:0000:0000:0000:0123:4567:89AB:CDEF
  - Egyszerűsítések a jelölésben:
    - Csoporton belül a bevezető 0-k elhagyhatók
      - Pl.: 0123 → 123
    - Csupa nulla csoportok (egy vagy több) két kettősponttal helyettesíthető
      - csak egyszer egy címben
      - Pl.: 8000::123:4567:89AB:CDEF
  - IPv4 címek írásmódja:
    - ::192.31.20.46



# IPv6 címek típusai

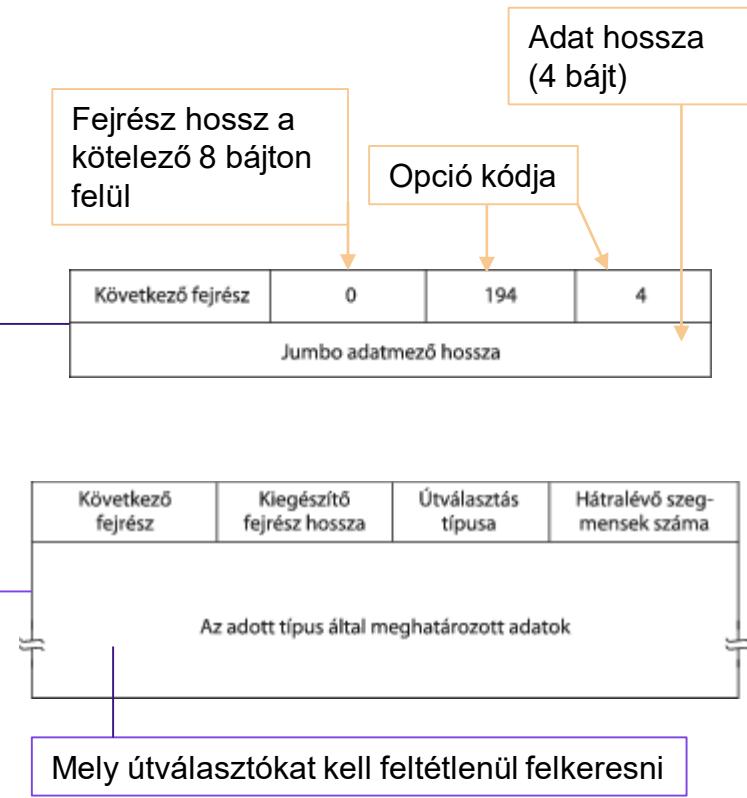
- Unicast
  - Egyetlen címzett
  - Globális (Global Unicast Address - GUA):
    - mint IPv4, globálisan egyedi. (Opcionális)
    - 2000::/3 (első 3 bit: 001)
  - Lokális (Link-Local Address - LLA):
    - csak helyi hálózatra alkalmazzuk. (Kötelező)
    - fe80::/10
  - Localhost:
    - ::1/128
- Multicast
  - Több címzett, prefix: ff00::/8
  - Pl.: ff02::1 – minden hoszt, ff02::2 – minden router (azonos adatkapcsolaton)
- Anycast
  - Unicast üzenet (egy cím), de a címzett bármely lehet a lehetséges címzettek közül (általában a legközelebbi)
- Broadcast
  - mindenki címzett: IPv6-ban nincs. Helyette Multicast.



# Kiegészítő IPv6 fejrészek

- Következő fejréssz mező jelzi
- Opcionálisak, minden a fő fejréssz után állnak
- Jelenleg 6 ilyen van

Kiegészítő fejréssz	Leírás
Ugrás opciók	Különféle információ az útválasztók számára
Címzetti opciók	További információ a címzett számára
Útválasztás opció	Laza lista a felkeresendő útválasztókról
Darabolás opció	A datagramdarabok kezelése
Hitelesítés opció	Az adó személyazonosságának ellenőrzése
Titkositott biztonsági adatmező	Információ a titkositott tartalomról



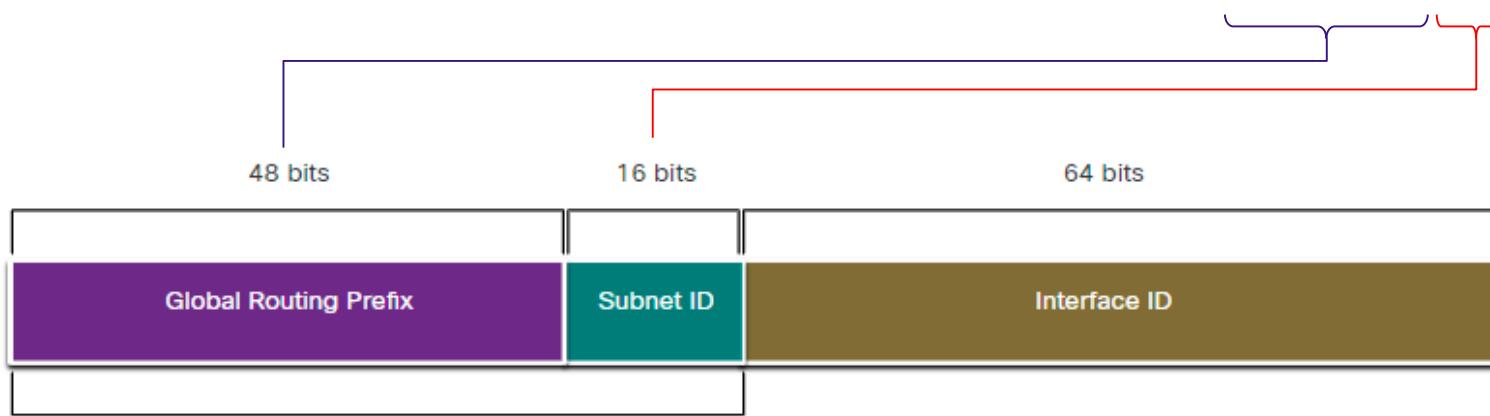
Darabolás: hasonló IPv4-hez, de csak a feladó darabolhat, útválasztó nem



# IPv6 alhálózatok

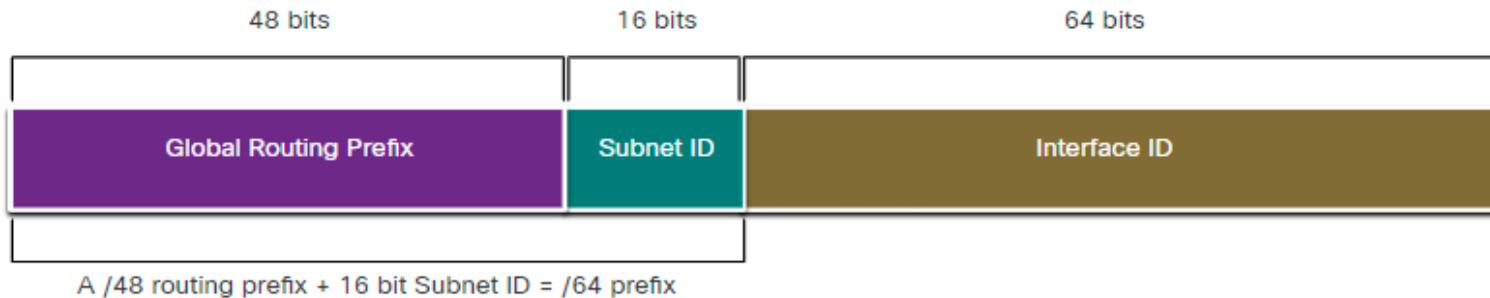
- IPv4 alhálózatok:
  - Takarékosan kellett bánni a rendelkezésre álló címekkel
  - Az alhálózati címbiteket a hoszt címbitekből „kölcsönöztük”
- IPv6 alhálózatok:
  - Van elég bit (128)
  - Hoszt címbitek: 64 bit
  - Prefix: összesen 64 bit
    - Ebből a **16 alsó bit** az **alhálózatokat** különbözteti meg
    - A 48 felső bit a **globális útvonalválasztási előtag**

```
2001:db8:acad:0000::/64
2001:db8:acad:0001::/64
2001:db8:acad:0002::/64
2001:db8:acad:0003::/64
2001:db8:acad:0004::/64
2001:db8:acad:0005::/64
2001:db8:acad:0006::/64
2001:db8:acad:0007::/64
2001:db8:acad:0008::/64
2001:db8:acad:0009::/64
2001:db8:acad:000a::/64
2001:db8:acad:000b::/64
2001:db8:acad:000c::/64
Subnets 13 – 65,534 not shown
2001:db8:acad:ffff::/64
```



# Útválasztás IPv6

- Az alapelvek hasonlóak, mint az IPv4 CIDR esetén
- Most 128 bites címek



# Az internet vezérlőprotokolljai

IPv4

- ICMP — Internet Control Message Protocol
  - Internetes vezérlőüzenet protokoll

Valami gond van...

- ARP — Address Resolution Protocol
  - Címfeloldási protokoll

Mi a címzett fizikai címe?

- DHCP — Dynamic Host Configuration Protocol
  - Dinamikus hosztkonfigurációs protokoll

Mi a logikai címem?

IPv6

- ICMPv6
  - IPv6 verzió, a fenti 3 funkció egyben



# ICMP

- Internetes vezérlőüzenet protokoll (Internet Control Message Protocol)
- Váratlan események jelzésére, tesztelésre

Üzenet típusa	Leírás	
Cél elérhetetlen	A csomagot nem lehetett kézbesíteni	<ul style="list-style-type: none"><li>• Nem található a cél</li><li>• DF miatt elakadt egy kicsomagos hálózatban</li></ul>
Időtúllépés	Az Élettartam mező elérte a 0-t	
Paraméter probléma	Érvénytelen fejrész mező	<ul style="list-style-type: none"><li>• A <b>tracert</b> ezt használja:</li><li>• élettartam: 1, 2, 3, ...</li></ul>
Forráslefojtás	Lefojtócsomag	
Átirányítás	Egy útválasztót tanít meg a földrajzra	
Visszhang kérés és visszhang válasz	Annak ellenőrzése, hogy egy gép életben van-e	
Időbényeg kérés/válasz	Ugyanaz, mint a visszhang kérés, csak időbényeggel	<ul style="list-style-type: none"><li>• A ping ezt használja</li></ul>
Útválasztó hirdetés/kérelmezés	Egy közelű útválasztó megtalálása	

A legfőbb ICMP üzenettípusok



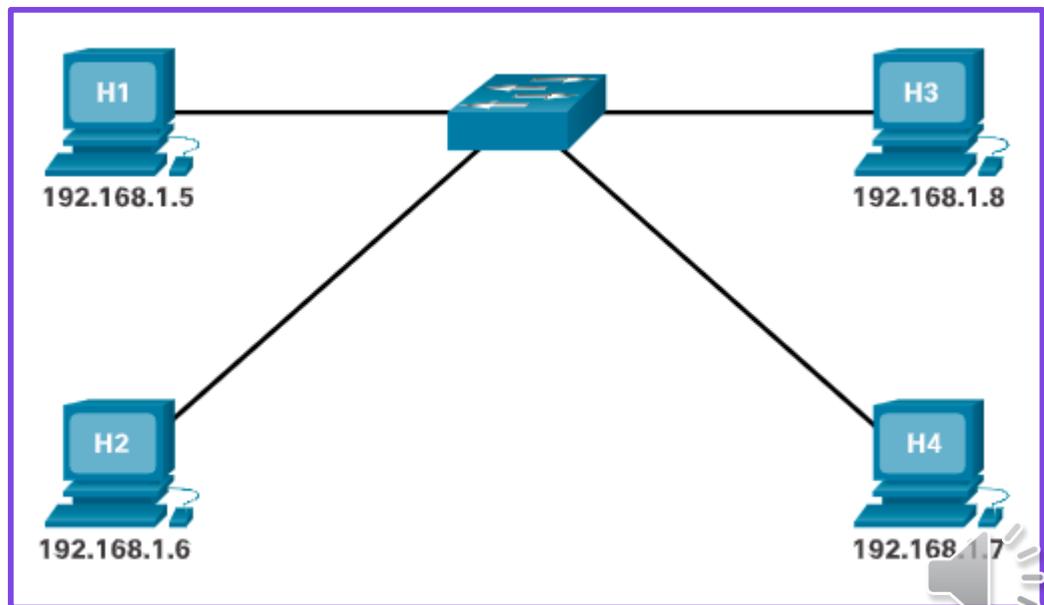
# ARP (1)

- Címfeloldási protokoll (Address Resolution Protocol)
- Probléma:
  - Hálózati réteg logikai (IP) címeket használ
  - Hálózati réteg az adatkapcsolati réteg szolgáltatásait használja
  - De az adatkapcsolati réteg fizikai (MAC) címeket használ
  - Honnan tudjuk, hogy melyik fizikai címhez melyik logikai cím tartozik?
- Megoldás:



# ARP (2)

1. **ARP REQUEST:** Broadcast keret küldése az adatkapcsolati rétegben:
    - Helló mindenki (FF-FF-FF-FF-FF-FF). Kié ez az IP-cím?
    - A fizikai címem AAA (ide kérem a választ).
  2. **ARP REPLY:** Válasz a feladónak (unicast):
    - Helló AAA. Enyém ez a cím. A fizikai címem BBB.
  3. Üzenet küldése:
    - Ethernet keret felépítés  
(forrás: AAA, cél: BBB)
    - Üzenet elküldése
  4. Üzenet vétele:
    - Ethernet keretet veszi a BBB című állomás
    - Keretet leveszi, a csomagot átadja az hálózati rétegnek.
- } ARP



# ARP (3)

- ARP használata:
  - Hálózaton belüli cím: másik hoszt fizikai címe
  - Hálózaton kívüli cím: átjáró fizikai címe

- Optimalizálás:

- **ARP táblázat**

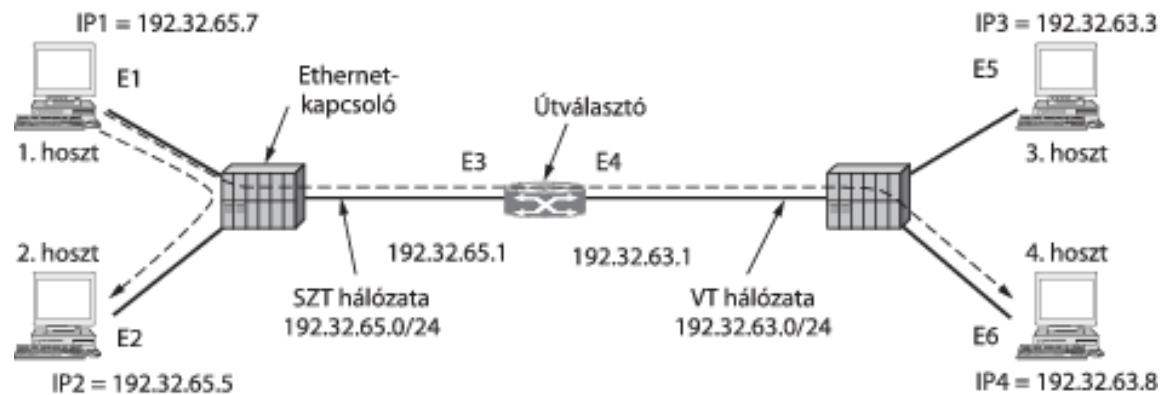
- Ismert címeket tartalmazza:

- IP - MAC

- Idővel „lejár”

- Működés:

- Küldés előtt ellenőrzi a táblát
    - Ha van bejegyzés, használja
    - Ha nincs, ARP REQUEST
    - ARP REPLY eredményét beírja

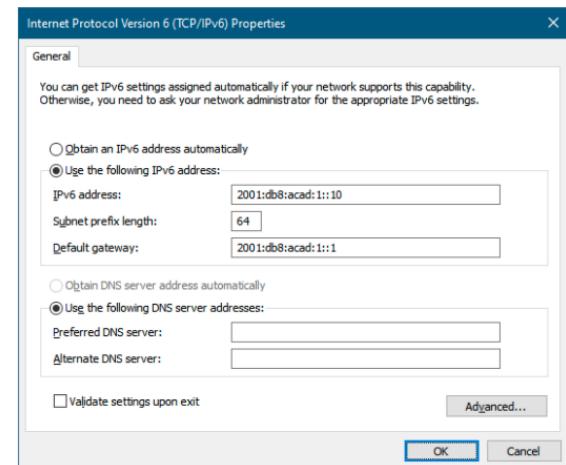


Keret	Forrás IP	Forrás Ethernet	Címzett IP	Címzett Ethernet
1-es hosztól a 2-es hoszt felé az Szt hálózatán	IP1	E1	IP2	E2
1-es hosztól a 4-es hoszt felé az Szt hálózatán	IP1	E1	IP4	E3
1-es hosztól a 4-es hoszt felé a VT hálózatán	IP1	E4	IP4	E6



# DHCP (1)

- Dinamikus hosztkonfigurációs protokoll (Dynamic Host Configuration Protocol)
- Probléma:
  - Mi az IP címem?
- Megoldás:
  - Kézzel konfiguráljuk
  - Automatikusan osztjuk ki: DHCP
- DHCP:
  - IP címeket „lízingel”: meghatározott időre
  - Egyéb paraméterek:
    - Hálózati maszk
    - Alapértelmezett átjáró
    - DNS szerver...



DHCP:  
Valójában egy alkalmazás UDP felett



# DHCP (2)

IP cím kérése:

- DHCP DISCOVER: Broadcast kérés
  - Helló, van itt egy DHCP szerver? Kérek egy IP címet.
- DHCP OFFER: Unicast válasz
  - Hello, itt a DHCP szerver. Mit szólnál ehhez a címhez?
- DHCP REQUEST: Broadcast kérés
  - Köszönöm, kérem ezt a címet
- DHCP ACK: Unicast válasz
  - Rendben van. 12 órára a tiéd.



# DHCP (3)

IP cím frissítése (lízing lejárta előtt):

- DHCP REQUEST: Broadcast kérés
  - Szeretném újra ezt a címet
- DHCP ACK: Unicast válasz
  - Rendben van. 1 órára a tiéd.

DHCP információk (ipconfig /all)

```
IPv4 Address . . . . . : 192.168.31.209(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained . . . . . : Wednesday, February 24, 2021 9:00:07 AM
Lease Expires . . . . . : Wednesday, February 24, 2021 9:00:06 PM
Default Gateway . . . . . : 192.168.31.1
DHCP Server . . . . . : 192.168.31.1
```



# ICMPv6

- ICMP-hez hasonló funkciók
    - Hiba, visszhang, időtúllépés, ...
  - **Neighbor Discovery protocol (ND)**
    - Neighbor **Solicitation** (szomszéd megszólítása)
    - Neighbor **Advertisement** (szomszéd hirdetés)
    - Router **Solicitation** (útválasztó megszólítása)
    - Router **Advertisement** (útválasztó hirdetés)
    - Redirect Message (üzenet átirányítás)
- } Mi a szomszéd fizikai címe?  
Mint ARP...
- } IPv6 cím infók  
DHCP helyett.



# ICMPv6 szomszédság felderítés - NS és NA

IPv6-ban hasonló a működés az ARP-hoz:

1. Szeretnék egy IPv6 címre üzenetet küldeni.
2. Tudom-e a MAC címét?
  - Neighbor Cache: mint ARP tábla
  - Ha van benne info, akkor használjuk (GOTO 5), ha nincs, akkor GOTO 3

A multicast MAC cím tartalmazza az IPv6 cím egy részét  
Csak néhány (tipikusan 1) állomás veszi.  
Nem kell broadcast! ☺

3. **NEIGHBOR SOLICITATION (NS): Multicast keret az adatkapcsolati rétegben:**
  - Helló szomszédok, akinek ehhez hasonló IPv6 címe van. Kié ez az IP cím?
  - A fizikai címem AAA (ide kérem a választ).
4. **NEIGHBOR ADVERTISEMENT (NA): Válasz a feladónak (unicast)**
  - Helló AAA. Enyém ez a cím. A fizikai címem BBB.
5. Üzenet küldése
6. Üzenet vétele
  - Neighbor cache frissítése

Neighbor (szomszéd) cache ~ ARP tábla  
IPv6 cím – MAC cím



# ICMPv6 RS és RA

IPv6 DHCP-szerű funkciója a **globális IP-cím (GUA)** előállítására

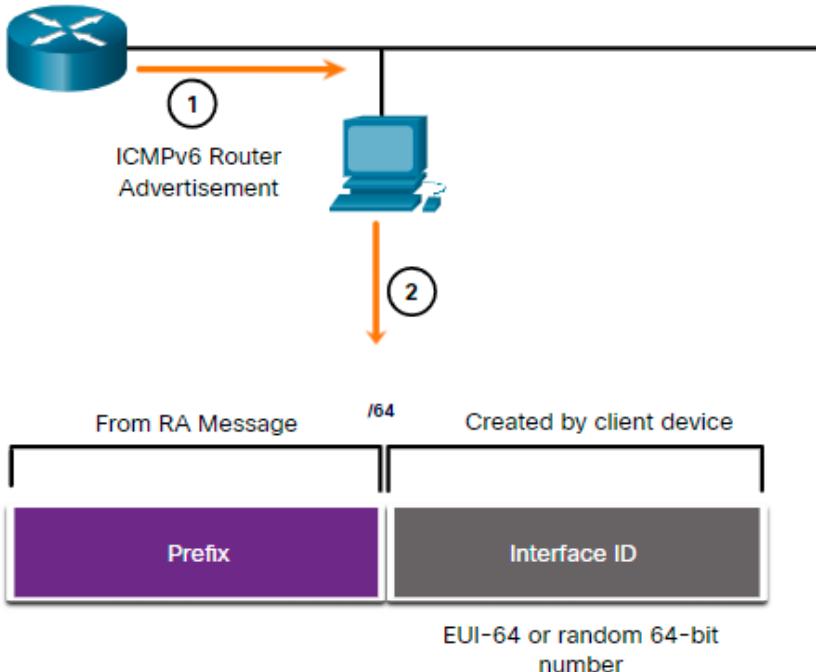
- **ROUTER SOLOCITATION (RS):** Multicast üzenet minden útválasztónak
  - Hello útválasztók (ff02::2), cím információra van szükségem
- **ROUTER ADVERTISEMENT (RA):** Multicast üzenet minden állomásnak
  - Helló minden hoszt (ff02::1), küldöm az információt:
    - Hálózati prefix és a prefix hossza
    - Alapértelmezett átjáró címe
    - DNS címe
    - [DHCP szerver címe]
  - De mi legyen a hoszt cím (a prefix már megvan)?
    - Csinálj magadnak egyet (SLAAC)
    - Küldök egy DHCP szerver címet, konzultálj vele

RA üzeneteket kérés nélkül is periodikusan küldik



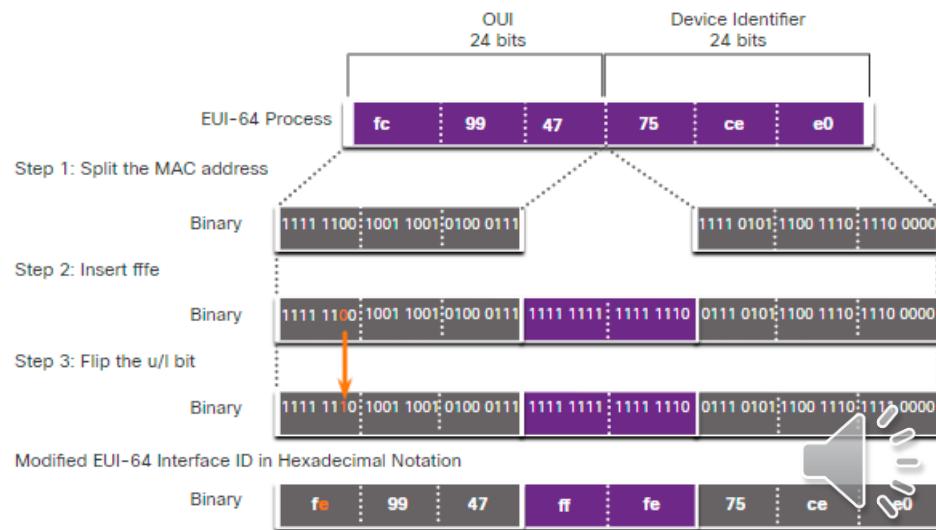
# Állapotmentes cím autokonfiguráció (SLAAC)

- SLAAC (Stateless Address Autoconfiguration)
- RA megküldte a prefixet (64 bit)
- Hiányzik még a hoszt cím (64 bit)
  - Véletlenszám (Windows 10)
  - EUI-64 (Cisco routerek)



## EUI-64

- A MAC címből készül
  - Kiegészítés (48bit → 64 bit)
  - 1 bit invertálása



# Számítógép-hálózatok



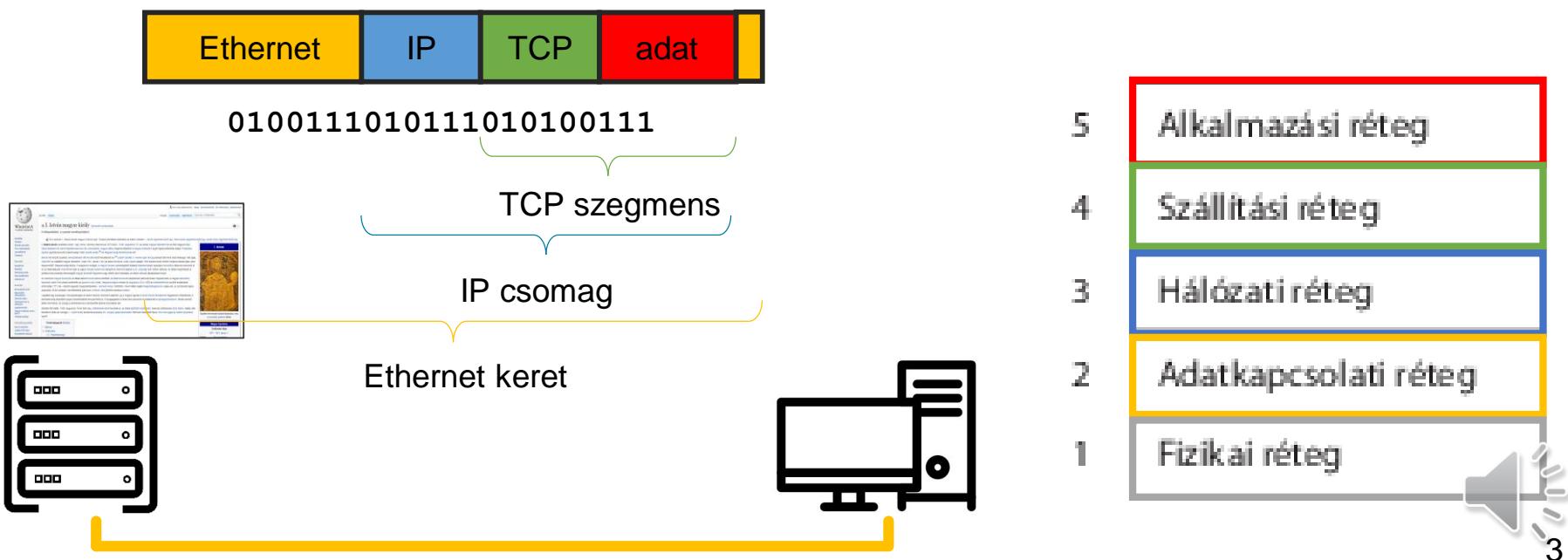
## 8. Szállítási réteg I. A szállítási protokollok elemei

# Tartalom

- A szállítási szolgáltatás
- Szolgáltatási (csatlakozó) primitívek
- A szállítási protokoll elemei
  - Címzés
  - Összeköttetés létesítése és bontása
  - Hibakezelés és forgalomszabályozás
  - Nyalábolás (multiplexelés)
- Szállítási protokollok az Interneten
  - UDP
  - TCP

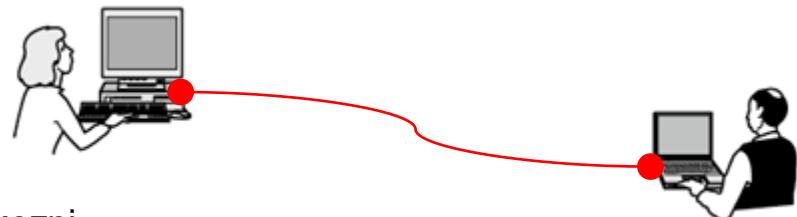


# Ismétlés: réteges szerkezet

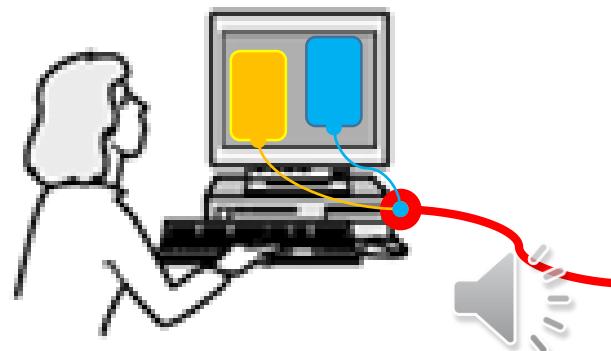


# A szállítási szolgáltatás (1)

- A hálózati réteg:
  - Két végpont közötti csomagtovábbítás
    - Datagram
    - Virtuális áramkör
  - Nem megbízható
  - Nagyrészt útválasztókon fut
    - Hiba esetén a felhasználó nem tud beavatkozni



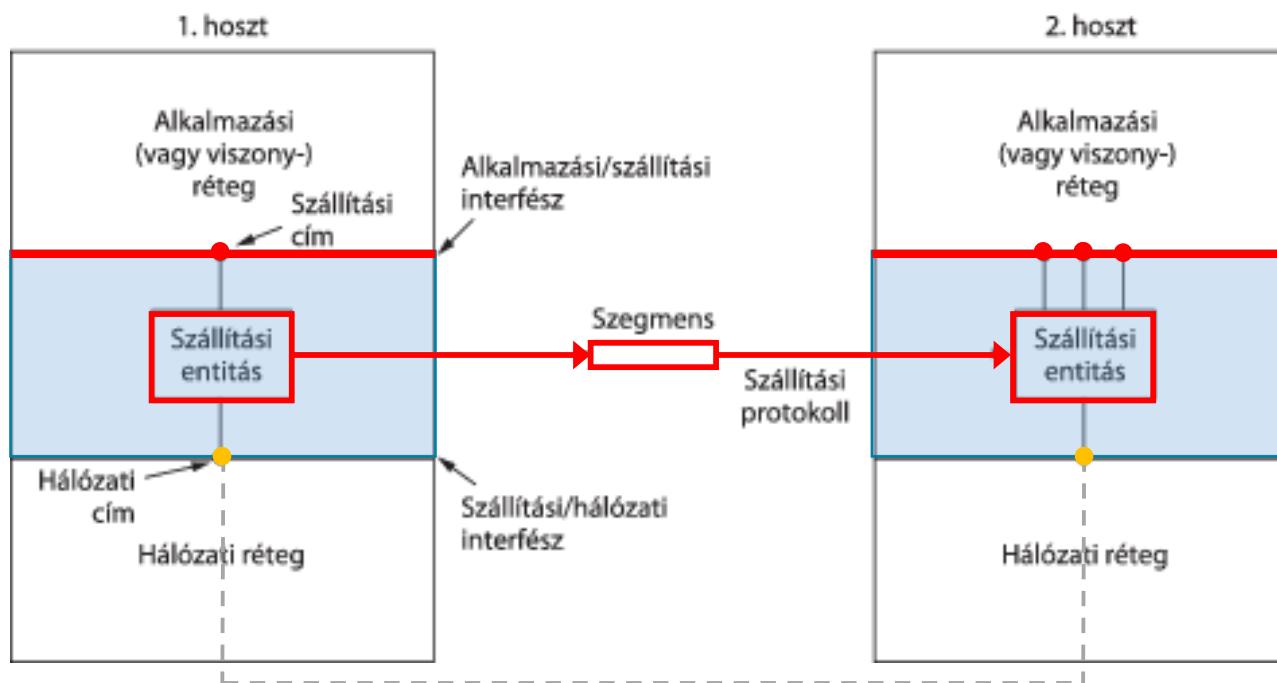
- Szállítási réteg:
  - Forrásgép egy **folyamatától** a célgép egy **folyamatáig**
  - **Megbízható** szolgáltatás
  - Használja a hálózati réteg szolgáltatásait
  - A felhasználó gépen fut
    - Teljes felhasználói kontroll
    - Társentitások tudnak egyeztetni
      - Pl.: megérkezett? Ha nem, újraküldés.
    - Így megbízhatóbb tud lenni, mint az alatta lévő réteg



# A szállítási szolgáltatás (2)

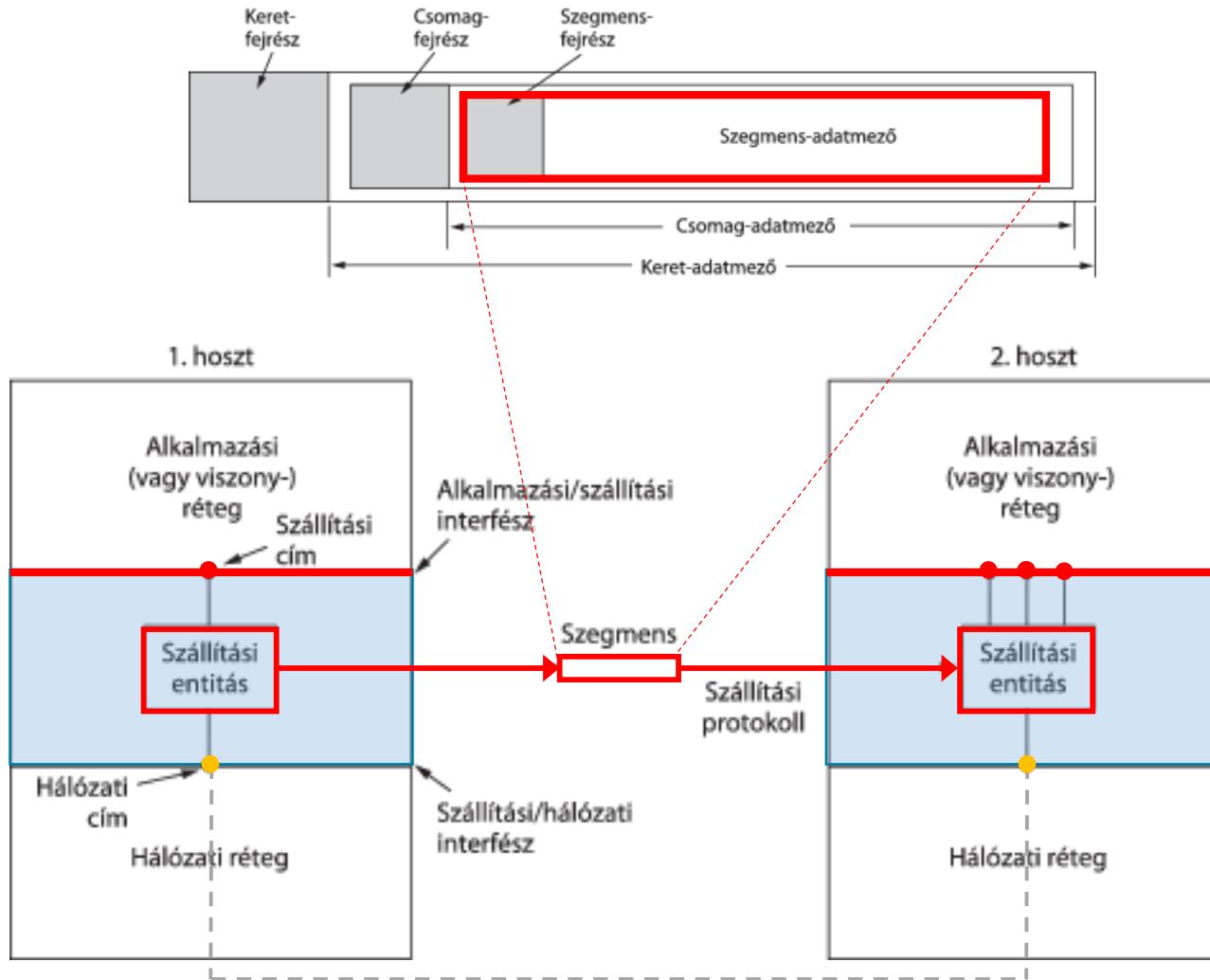
- A felsőbb rétegeknek nyújtott szolgáltatások
  - Összeköttetés alapú
  - Összeköttetés nélküli
- Végrehajtó szoftverelem: **szállítási entitás**
- **Szállítási cím**
  - port + hálózati cím+ protokoll azonosító
- Interfész: szállítási **szolgáltatási primitívek**

Primitív	Jelentése
SOCKET	Új kommunikációs végpont létrehozása
BIND	Helyi cím hozzárendelése a csatlakozóhoz
LISTEN	Összeköttetés-elfogadási szándék bejelentése, várakozási sor hosszának megadása
ACCEPT	Bejövő összeköttetés passzív létesítése
CONNECT	Aktív próbálkozás összeköttetés létesítésére
SEND	Adatküldés az összeköttetésen keresztül
RECEIVE	Adatfogadás az összeköttetésről
CLOSE	Összeköttetés bontása



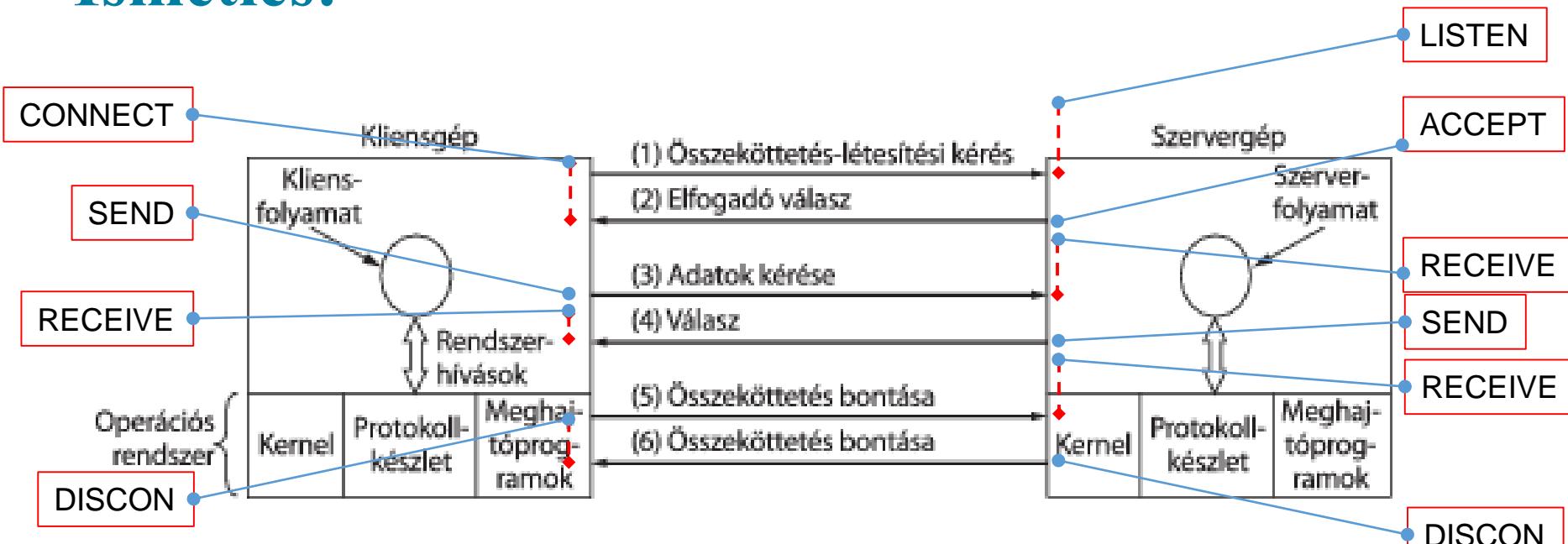
# A szállítási szolgáltatás (3)

- Szegmens



# Berkeley csatlakozó primitívek (1)

## Ismétlés:



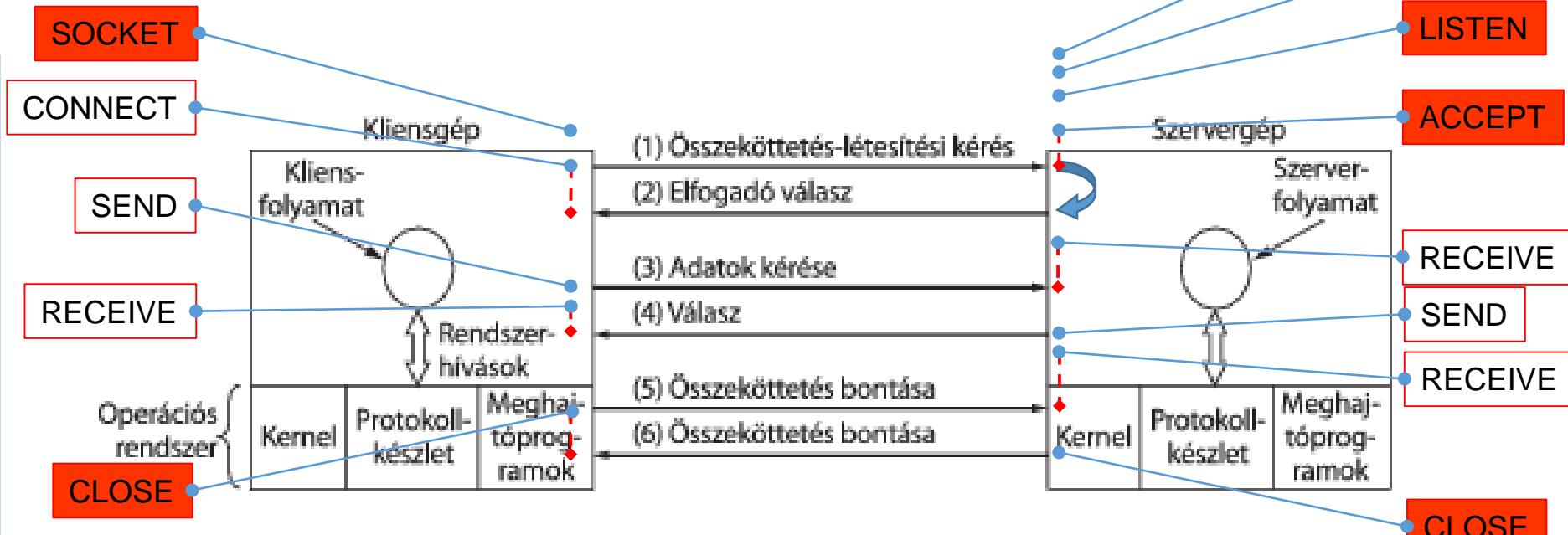
id  

Primitiv	Jelent��se
LISTEN	Blokolt v��rakoz��s bej��v�� kapcsolatfelv��telre
CONNECT	��sszek��ttet��s l��trehoz��sa egy v��rakoz�� t��rsentit��ssal
ACCEPT	Bej��v�� kapcsolat elfogad��sa a t��rsentit��st��l
RECEIVE	Blokolt v��rakoz��s bej��v�� ��zenetre
SEND	��zenet k��ld��se a t��rsentit��snaik
DISCONNECT	��sszek��ttet��s bont��sa



# Berkeley csatlakozó primitívek (2)

## Konkrét megvalósítás



Primitív	Jelentése
SOCKET	Új kommunikációs végpont létrehozása
BIND	Helyi cím hozzárendelése a csatlakozóhoz
LISTEN	Összeköttetés-elfogadási szándék bejelentése, várakozási sor hosszának megadása
ACCEPT	Bejövő összeköttetés passzív létesítése
CONNECT	Aktív próbálkozás összeköttetés létesítésére
SEND	Adatküldés az összeköttetésen keresztül
RECEIVE	Adatfogadás azzal a összeköttetésről
CLOSE	Összeköttetés bontása



# A szállítási protokollok elemei

- Címzés
- Összeköttetés létesítése
  - Probléma: Késleltetett/duplikált csomagok
  - Megoldás: háromutas kézfogás
- Összeköttetés bontása
- Hibakezelés és forgalomszabályozás
- Nyalábolás (multiplexelés)



# Címzés (1)

- **Folyamattól folyamatig**

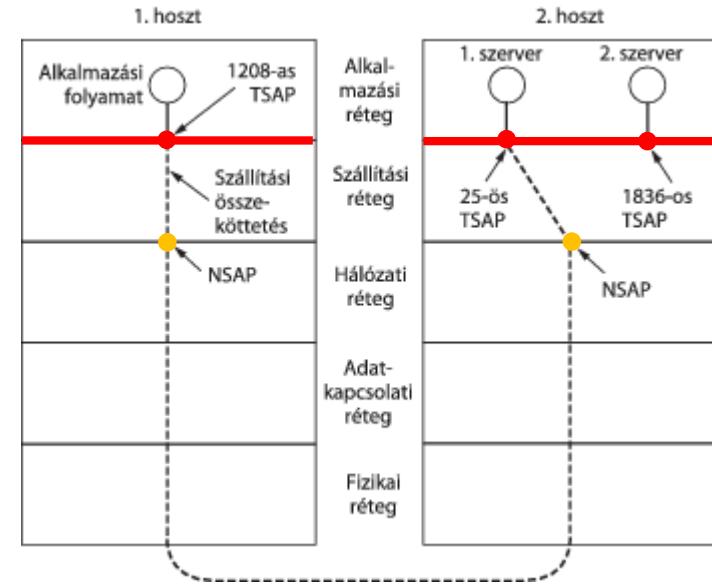
- **Cím:**

- Szállítási réteg:

- **TSAP** (Transport Service Access Point)
    - Interneten **portnak** nevezzük

- Hálózati réteg:

- **NSAP** (Network Service Access Point)
    - Interneten **IP-címnak** nevezzük

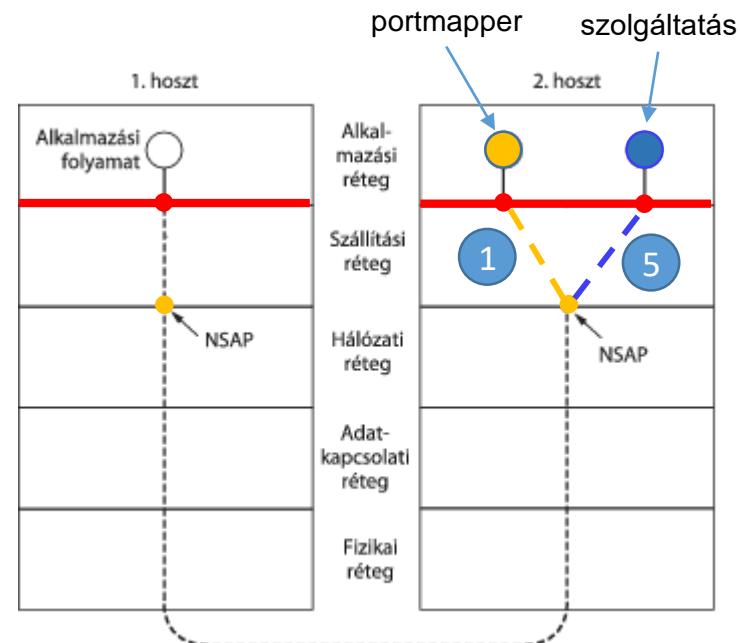


1. A 2. hoszt levelezőszervere a 25-ös TSAP-hoz csatlakozik (pl. SOCKET, BIND, LISTEN)
2. Az 1. hoszt alkalmazása 1208-as TSAP-ra csatlakozik és elküld egy CONNECT kérést
  - Forrás: 1. hoszt (NSAP), 1208-as TSAP
  - Cél: 2. hoszt (NSAP), 25-ös TSAP
3. 1. hoszt alkalmazása elküldi a levelet
4. Levelezőszerver nyugtáz
5. ÖK lebontása



# Címzés (2)

- Honnan tudja az 1. hoszt a szolgáltatás TSAP címét a 2. hoszon?
- Megoldások:
  - Állandó cím**, amit egy adatbázis tárol
    - Pl.: levelezés – 25
    - Unix: /etc/services
  - Portszolgáltató (portmapper)**
    - Speciális folyamat
    - Címe ismert/állandó
    - ÖK létesítése **portszolgáltatóval (PSZ)**
    - Kérés küldése: hol van a **szolgáltatás**?
    - PSZ visszaküldi a **szolgáltatás** portját
    - ÖK bontása **PSZ**-val
    - ÖK létesítése a **szolgáltatás**sal...



ftp	21/tcp	
fsp	21/udp	fspd
ssh	22/tcp	
telnet	23/tcp	
smtp	25/tcp	mail
time	37/tcp	timserver
time	37/udp	timserver

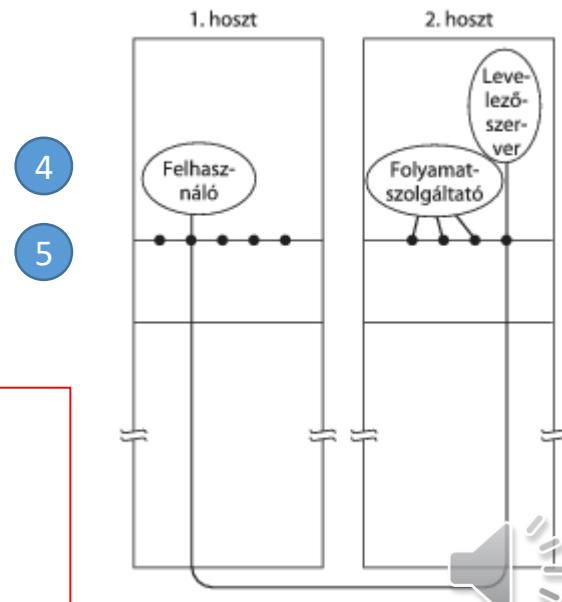
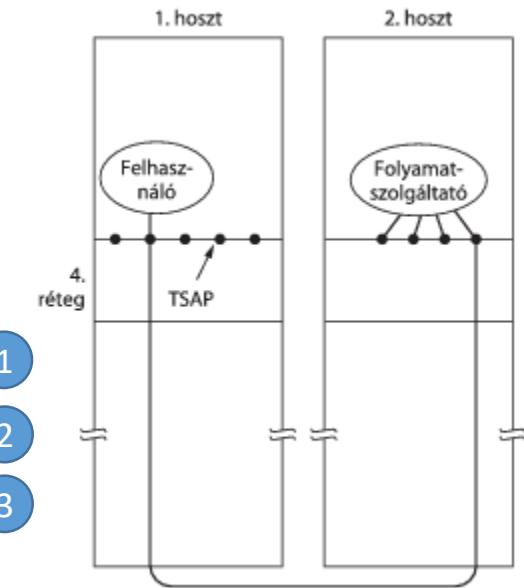
/etc/services

sunrpc	111/tcp	portmapper
sunrpc	111/udp	portmapper
rpc2portmap	369/udp	



# Címzés (3)

- Probléma:
  - minden futó szolgáltatás erőforrást foglal
  - lehet, hogy nagyon ritkán használják
- Megoldás:
  - **kezdeti összeköttetés protokoll**
  - (initial connection protocol)
  - **Folyamatszerver** (process server):
    - több portot is figyel
    - ha kérés érkezik, akkor
      - elindítja a megfelelő folyamatot
      - átadja neki az ÖK-t
    - pl. unix: inetd



1. Az 1. hoszt folyamata levelet akar küldeni a 2. hoszt 25-ös portján keresztül
2. A 2. hoszon nem fut a levelező szerver, csak a folyamatszerver (FSZ)
3. A kapcsolatot a FSZ építi ki a 25-ös porton
4. A FSZ elindítja a levelező szervert és átadja neki a kapcsolatot
5. FSZ tovább figyel a többi porton

# Címzés (4)

- A címzés tehát három elemből áll:
  - Protokoll (pl. TCP, UDP)
  - IP cím (pl. IPv4 vagy IPv6)
  - Port száma (pl. 16 bites egész)
- Szerverek gyakran „ismert” portokon vannak („Well Known Ports”)
  - 1024 alatti porok
  - Pl.:
    - SSH (TCP, UDP): 22
    - FTP (TCP): 20, 21
    - SMTP (TCP): 25
    - HTTP (TCP): 80
    - HTTPS (TCP): 443
- A kliensek általában ideiglenes portokra csatlakoznak
  - Az operációs rendszer választja
  - Véletlenszerű



# Összeköttetés létesítése (1)

- Naiv megközelítés:
  - → CONNECTION REQUEST
  - ← CONNECTION ACCEPT
- Probléma:
  - Elveszett csomagok
  - Kettőzött csomagok
  - Késve érkező csomagok
- A protokollnak MINDIG jól kell működnie

# Összeköttetés létesítése (2)

## Duplikált csomagok kezelése

- **Sorszámozzuk a csomagokat**

- A sorszámok egy tartományban mozognak, utána átfordulnak
  - Pl. 32 bites egész számok
- Ha az adási sebesség korlátos, akkor egy  $T$  ideig biztosan nem ismétlődhetnek a sorszámok
  - Ha mégis, akkor az egy kóbor/késő csomag, amit már megismételtünk



Elküldött csomagok: 0 1 2 3 4 5 6 7 8 9 ... 99 0 1 2 3 4 5

Vett csomagok: 0 1 2 3 4 5 6 **6** **5** 7 8 9 ... 99 0 1 2 3 4 5 **6**

Ezeket eldobjuk

- De mi van a nagyon sokat késő csomagokkal ( $> T$ )?

- Ezeket ki kell irtani
- Módszerek:

- **Ugrásszámláló alkalmazása** (ha túl sok ugráson át bolyong, akkor eldobjuk)
- Időcímke alkalmazása (ha túl hosszú ideig bolyong, akkor eldobjuk)
  - Bonyolult (szinkronizált órák kellenek)
  - Helyette az ugrásszámlálót használjuk



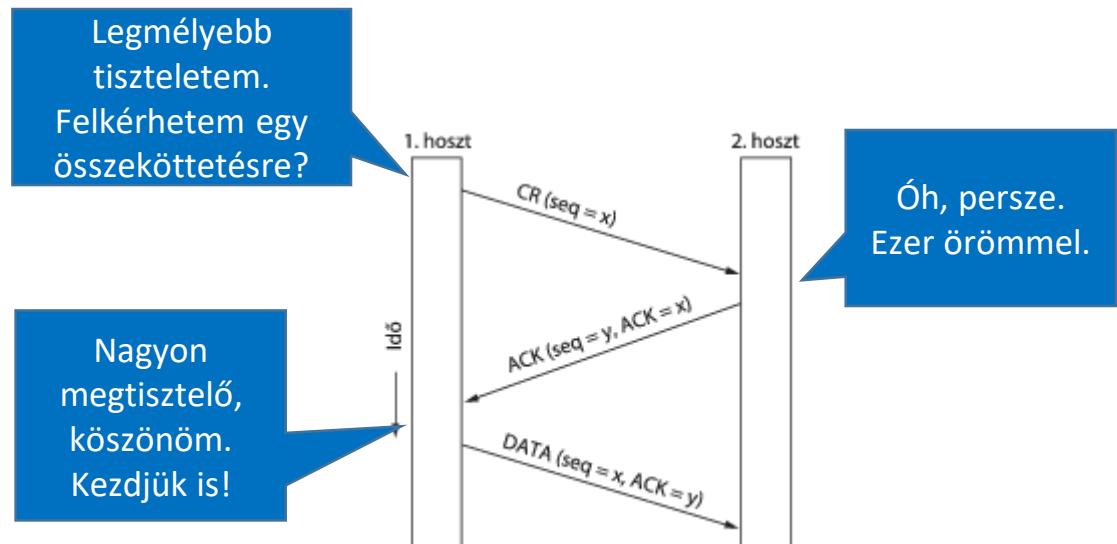
Itt baj van...  
Ilyen sokáig nem  
kóborolhat csomag!



# Összeköttetés létesítése (3)

## Háromutas kézfogás

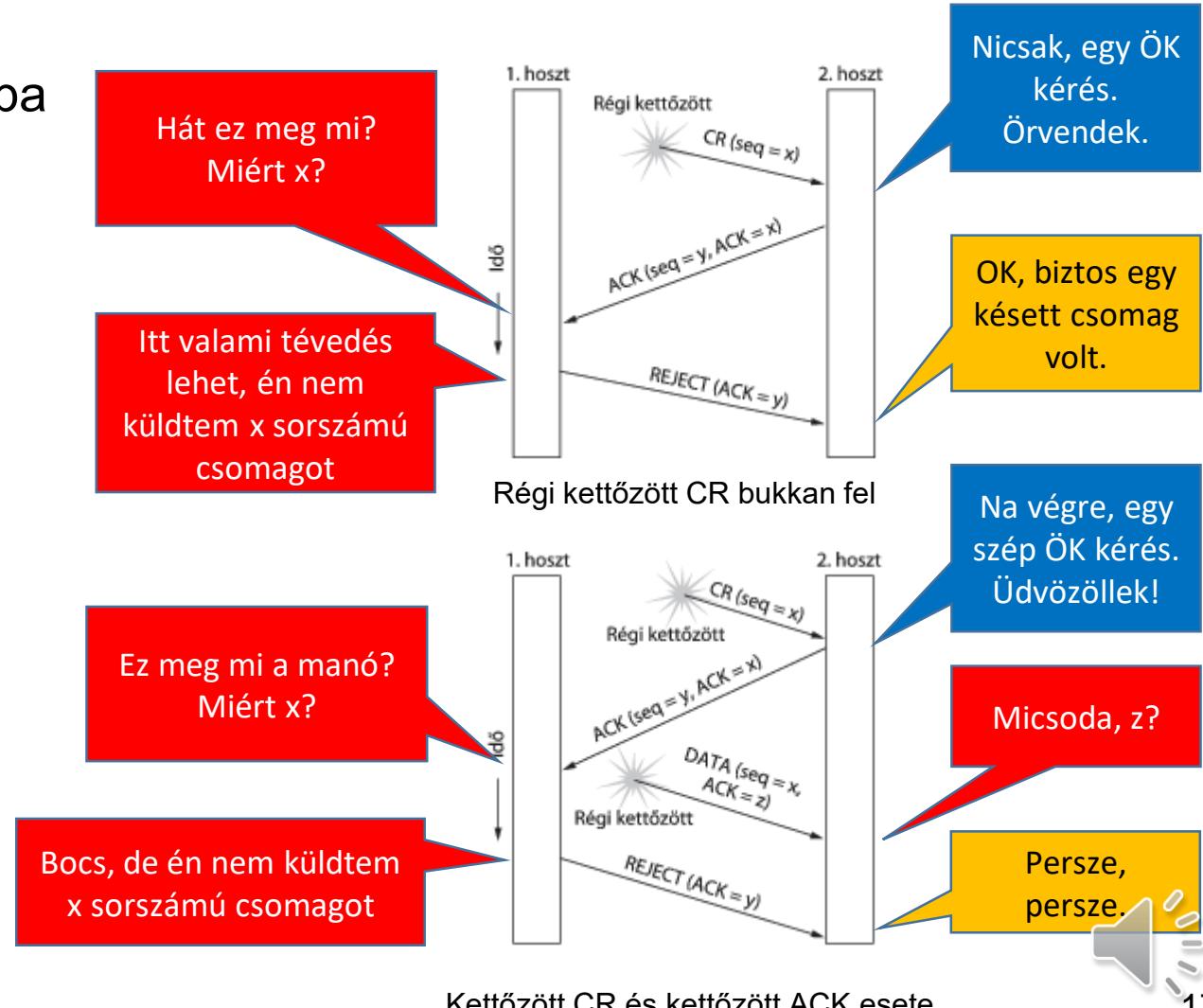
- (Three-way handshake)
- **CR:**
  - Connection request
  - Saját sorszám:  $x$
- **ACK:**
  - Saját sorszám:  $y$
  - Nyugtázza  $x$ -et
- **DATA:**
  - Nyugtázza  $y$ -t
- A gyakorlatban
  - a kezdeti sorszám ( $x, y$ ) álvéletlen szám
  - Biztonsági okból:
    - Ne legyen megjósolható, így támadható



# Összeköttetés létesítése (4)

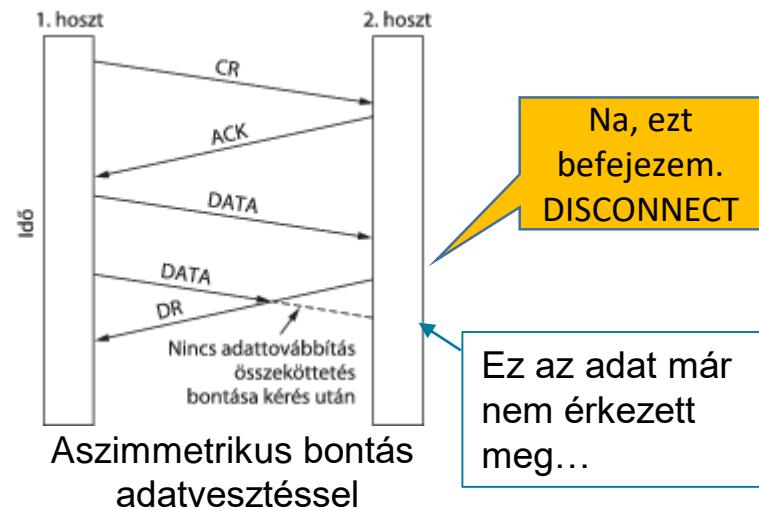
## Háromutas kézfogás hibatűrése

- Néhány potenciális hiba
  - Régi kettőzött CR
  - Kettőzött CR és ACK



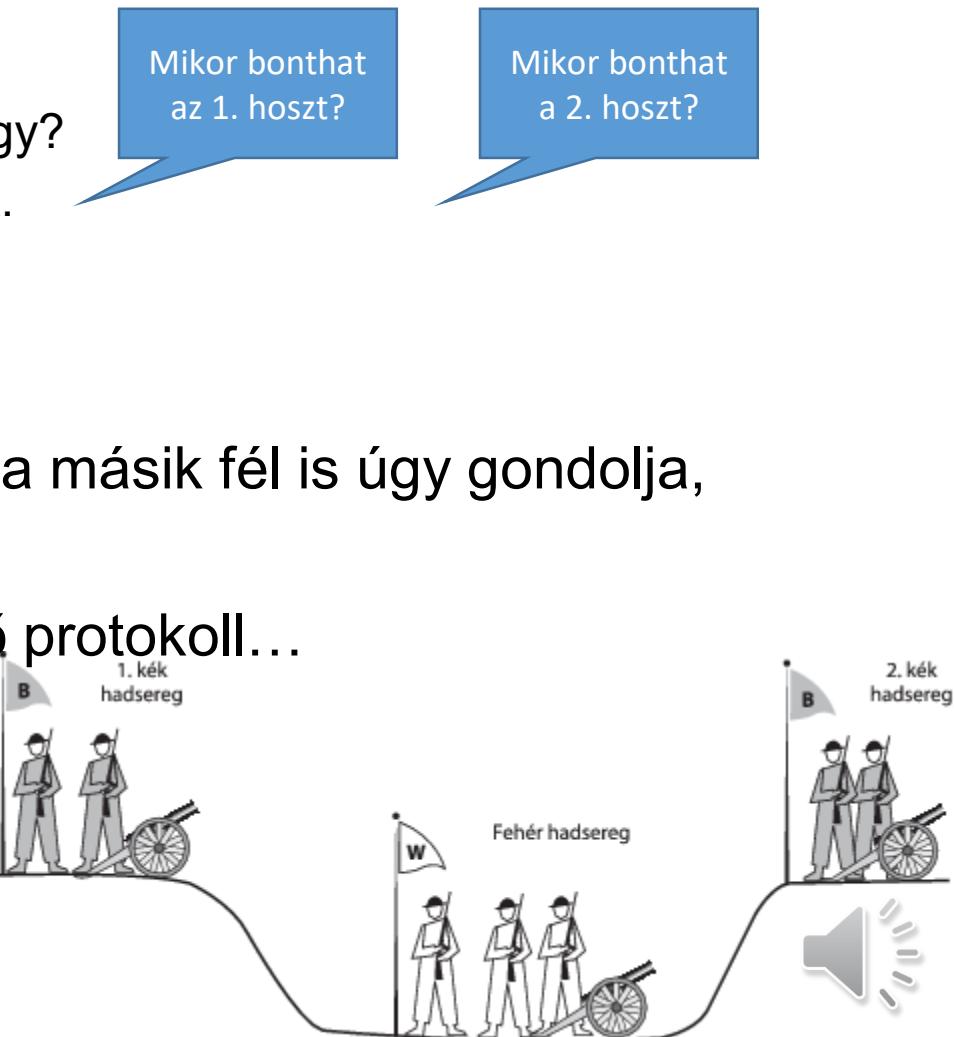
# Az összeköttetés bontása (1)

- Aszimmetrikus bontás
  - Mint telefon esetén
  - Az egyik fél bont és az ÖK megszakad
  - Ez adatvesztéssel járhat
- Szimmetrikus bontás
  - Mindkét irányt külön kezeljük
  - Mindkét irányt függetlenül bontjuk le
  - Ha az egyik irányt lebontottuk, a másikban még küldhetünk adatot



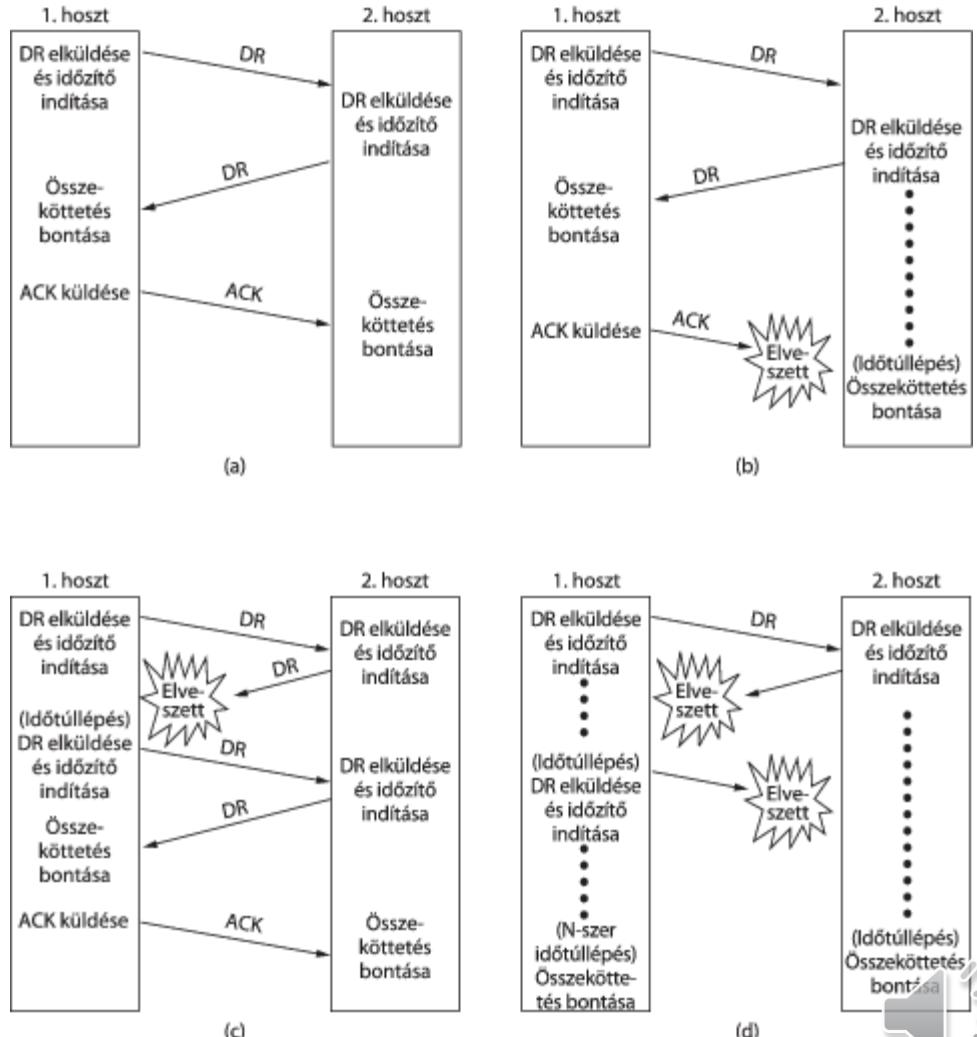
# Az összeköttetés bontása (2)

- Mehetünk-e biztosra? Pl.:
  - 1. $\rightarrow$ 2. Én befejezem. Te készen vagy?
  - 1. $\leftarrow$ 2. Én is befejezem. Bonthatunk.
  - 1. $\rightarrow$ 2. OK, bontsunk.
- Biztosak lehetünk-e abban, hogy a másik fél is úgy gondolja, hogy bonthat?
- Sajnos nincs tökéletesen működő protokoll...
  - Ez a „két hadsereg” problémája
- Helyette: „elég jó” módszer:
  - Háromutas kézfogás + időzítő



# Az összeköttetés bontása (3)

- DR: DISCONNECT REQUEST
  - Egyirányú bontás jelzése
- a. Normál működés
- b. Elveszett ACK.
  - 2. hoszt időtúllépéssel bont
- c. Elveszett válasz DR
  - 1. hoszt ismételt DR-t küld
- d. Elveszett válasz és minden további üzenet
  - Mindkét hoszt időtúllépéssel bont



# Hibakezelés és forgalomszabályozás (1)

- Hibakezelés:
  - Adatok megfelelő biztonsággal (hibátlanul) érkezzenek meg
- Forgalomszabályozás:
  - Az adó és vevő sebességét szinkronizálja
  - Egy gyors adó ne töltön túl egy lassú vevőt
- Az adatkapcsolati réteg is hasonló problémákkal küzd
  - Megismert eszközök:
    - Hibajelzés: CRC vagy ellenőrző összeg
    - Automatikus újraküldés keret sorszáma alapján: Ha nincs ACK, akkor újraküldjük
      - Megáll-és-vár protokoll
      - Csővezetékes megoldások
  - Ezen megoldásokat fogjuk itt is alkalmazni (alkalmas módosításokkal)



# Hibakezelés és forgalomszabályozás (2)

- Miért kell ismét ezzel foglalkozni ebben a rétegben is?
  - Az adatkapcsolati rétegben ugrásonként ellenőrzünk.  
De mi történik, ha pl. egy router belsejében sérül az adat?
  - mindenéppen szükséges a **végponttól végpontig** ellenőrzés
- Más a cél és a mérték is:
  - Adatkapcsolati réteg:
    - Sávszélesség-késleltetés szorzat alacsony (pl. WiFi) → kicsi pufferméret
    - Sok esetben nagyon kicsi a hibaarány (pl. optikai vagy Ethernet), így nincs is újraküldés
  - Szállítási réteg:
    - Nagy sávszélesség-késleltetés szorzat → nagy pufferméret. Ezt okosan kell kezelni.
    - Mindig van újraküldés, kijavítja az összes, esetleg még előforduló hibát.



# Hibakezelés és forgalomszabályozás (3)

## Ismétlés

	Adó	Vevő	
Megáll-és-vár	<p>Egy üzenetet elküld, majd vár a nyugtára</p> <p>Csak a nyugta vétele után küldi a következő üzenetet</p> <p>Időtúllépés esetén újraküldés (adó ablak mérete <b>1</b>)</p>	<p>Csak a sorban következő <b>egyetlen</b> üzenetet fogadja el és nyugtázza</p> <p>Többi vett üzenetet eldobja (vevő ablak mérete <b>1</b>)</p>	
N-visszalépéses csővezeték	<p>Maximum <b>N</b> üzenet küldhet nyugta nélkül</p> <p>Időtúllépés esetén újraküldés az elveszett üzenettől kezdve (adó ablak mérete <b>N</b>)</p>	<p>Csak a sorban következő <b>egyetlen</b> üzenetet fogadja el és nyugtázza</p> <p>Többi vett üzenetet eldobja (vevő ablak mérete <b>1</b>)</p>	
Szelektív ismétléses csővezeték	<p>Maximum <b>N</b> üzenet küldhet nyugta nélkül</p> <p>Minden üzenetre időzítő indul</p> <p>Időtúllépés esetén egy üzenet újraküldése</p> <p>NACK esetén üzenet újraküldése</p> <p>(adó ablak mérete <b>N</b>)</p>	<p>Maximum <b>M</b> üzenetet tud tárolni (ha több jön, azt eldobja)</p> <p>A sorban megérkezett üzeneteket nyugtázza (kummulatív nyugta)</p> <p>Hiányzó üzeneteket szelektív NACK-val jelezheti (vevő ablak mérete <b>M</b>)</p>	

# Hibakezelés és forgalomszabályozás (4)

- Csúszóablakos protokoll:

- Egyesíti a korábbi eszközöket, de
- **változó méretű ablakokat (pufferméret) használ**
- Különválasztja a
  - nyugtázást és a
  - pufferkezelést

- Egyszerű példa:

- A ad
- B vesz
- Sorszám 4 bites (0-15)

- B minden üzenete tartalmaz:

- nyugta (sorszámmal)
- rendelkezésre álló pufferméret  
(vevő oldali ablakmérettel)

- Valóságban szimmetrikus!

- A is nyugtázza B üzeneteit
- A is küld ablakmérétet B-nek



TCP

A: 0 1 2 3 4 5 6 7 ...

A	Üzenet	B	Megjegyzés
1	→ <8 puffer kérése >	→	A 8 puffert szeretne
2	← <nyug = 15, puff = 4 >	←	B csak 0-3 sorszámu üzeneteket engedélyezi
3	→ <sorsz = 0, adat = m0 >	→	A-nak 3 puffere maradt
4	→ <sorsz = 1, adat = m1 >	→	A-nak 2 puffere maradt
5	→ <sorsz = 2, adat = m2 >	...	Az üzenet elveszett, de A azt hiszi, hogy már csak egy puffer maradt
6	← <nyug = 1, puff = 3 >	←	B nyugtázza 0 és 1 sorszámuakat, 2-4-et engedélyezi
7	→ <sorsz = 3, adat = m3 >	→	A-nak 1 puffere maradt
8	→ <sorsz = 4, adat = m4 >	→	A-nak nincs több puffere, le kell állnia
9	→ <sorsz = 2, adat = m2 >	→	A időzítése lejár és újraküldi a 2-es üzenetet
10	← <nyug = 4, puff = 0 >	←	Minden nyugta megérkezett, de A még mindig blokkolt
11	← <nyug = 4, puff = 1 >	←	A most küldheti az 5-ös üzenetet
12	← <nyug = 4, puff = 2 >	←	B valahol talált egy újabb puffert
13	→ <sorsz = 5, adat = m5 >	→	A-nak 1 puffere maradt
14	→ <sorsz = 6, adat = m6 >	→	A most ismét blokkolódik
15	← <nyug = 6, puff = 0 >	←	A továbbra sem küldhet
16	... <nyug = 6, puff = 4 >	←	Potenciális holtpont



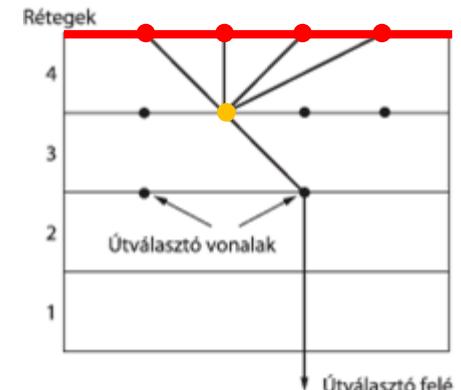
# Nyalábolás (multiplexelés)

- Nyalábolás:

- Több szállítási összeköttetés ugyanazt a hálózati címet használja
- Több **TSAP** → egy **NSAP**
- Ez a tipikus, hiszen általában 1 hálózati cím tartozik egy géphez
- Pl. a TCP ezt használja

- Fordított nyalábolás

- Egy szállítási összeköttetés több hálózati címet (interfész) is használ
- Egy **TSAP** → több **NSAP**
- Ok: sávszélesség megnő
- k db. hálózatai összeköttetés: k-szoros sávszélesség
- Pl. SCTP (Stream Control Transmission Protocol) is ilyen megoldást használ



# Forgalomszabályozás és torlódáskezelés

- Forgalomszabályozás (flow control)

- Adó és vevő sebességének illesztése
  - Kezelés:

- Vevő információt küld az állapotáról (ablakméret)
    - Adó ennek megfelelően ad (elküldött csomagok száma)
    - Csúszóablakos protokoll



- Torlódáskezelés (congestion control)

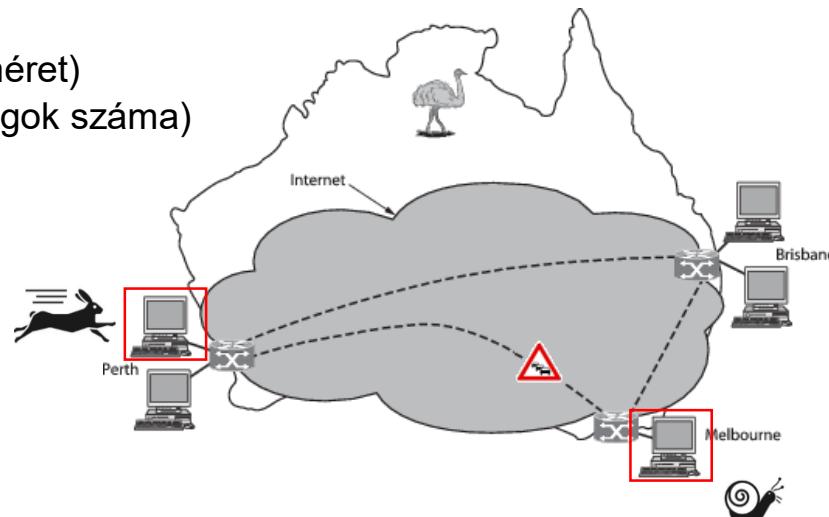
- Ha

- túl sok gép
    - túl gyorsan
    - túl sok

- csomagot küld a hálózatba, akkor torlódás keletkezik.
    - A torlódás az útválasztókban jön létre, de a végpontokon kezeljük.

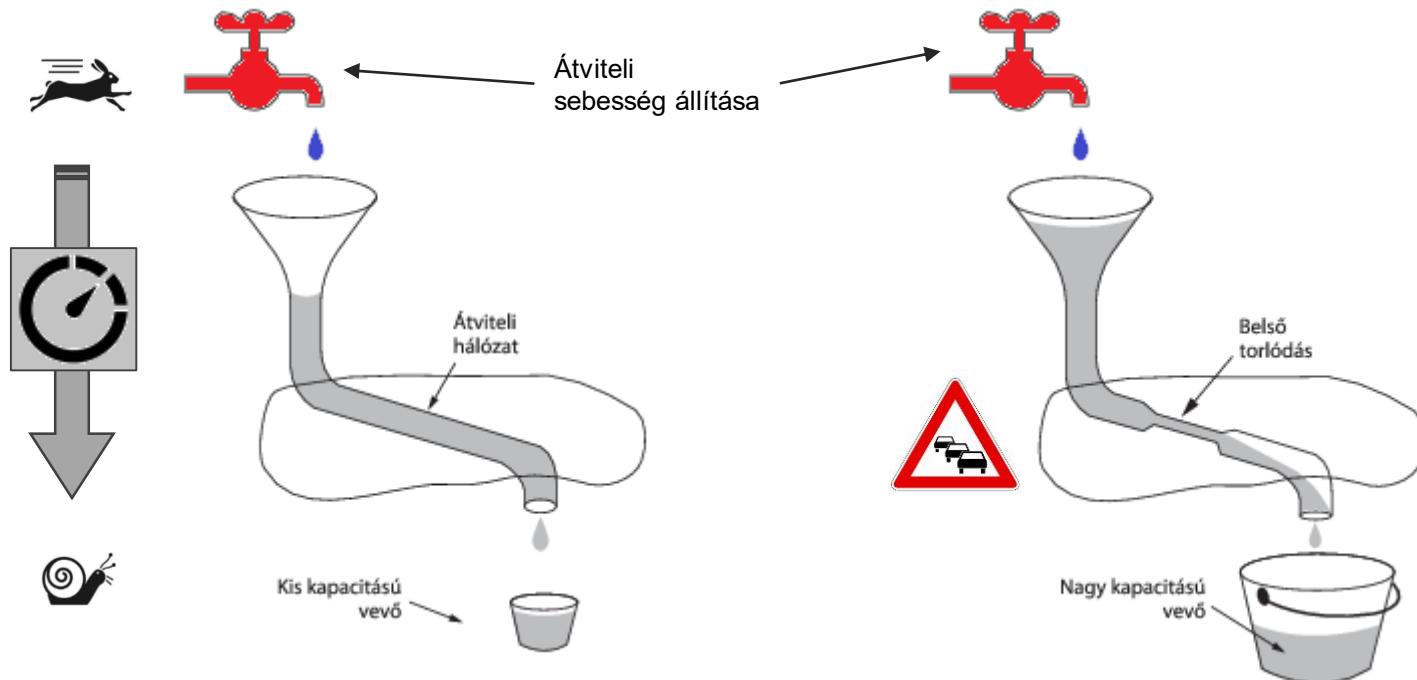
- Kezelés:

- Érzékeljük a torlódást
    - Ennek megfelelően a hálózatba küldött csomagok számának szabályozása



# Küldési sebesség szabályozása

- Beavatkozás módja:
  - Átviteli sebesség szabályozása



Gyors hálózat, kis kapacitású vevő:  
forgalomszabályozás

Lassú hálózat, nagy kapacitású vevő:  
torlódáskezelés

# Torlódáskezelés (1)

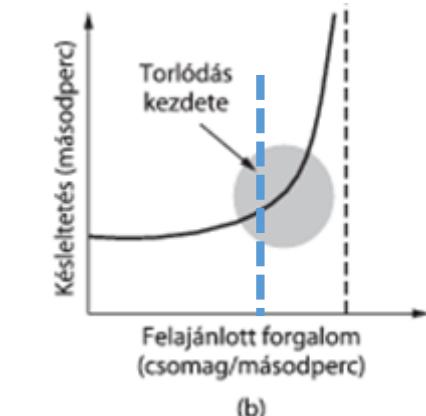
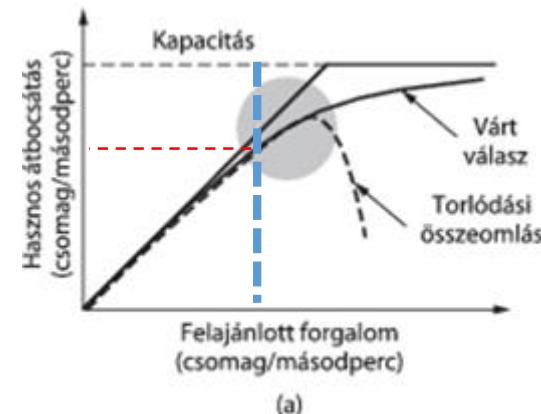
- Cél:
  - **Torlódás elkerülése**
  - **Hatókony:** a teljes rendelkezésre álló sávszélességet ki tudja használni
  - **Sávszélesség megfelelő elosztása** a hálózatot használó entitások között
  - **Igazságosan** osztja el a sávszélességet
  - Gyorsan képes **követni** a forgalmi igényeket (**konvergencia**)



# Torlódáskezelés (2)

## Hatékonyság

- Felajánlott forgalom (offered load)
- Hasznos átbocsátóképesség (goodput)
  - Egy darabig a felajánlott forgalommal együtt nő, majd:
  - Torlódás:
    - Goodput lassul, majd visszaesik
    - Késleltetés nő
- Legjobb teljesítőképesség:
  - A sávszélességet a késleltetés növekedéséig foglaljuk le

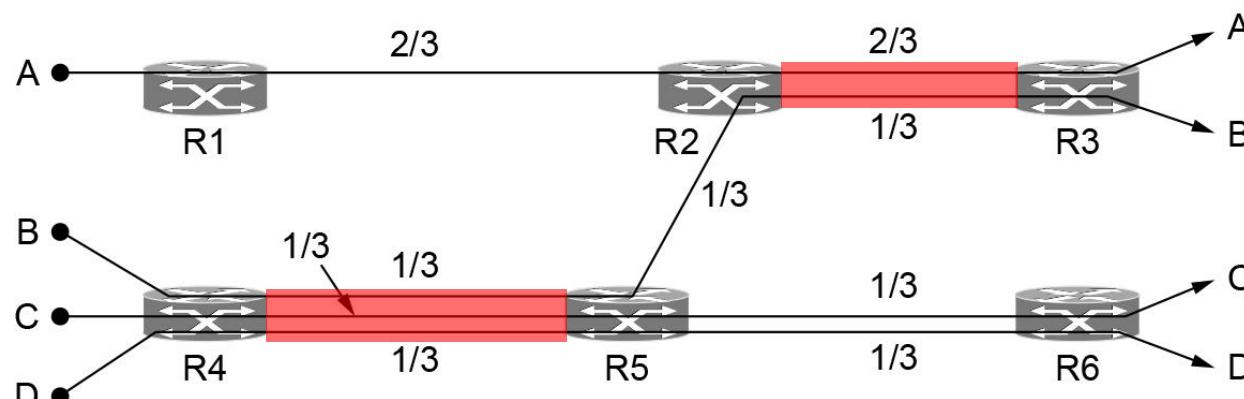


- (a) Hasznos átbocsátás a felajánlott forgalom függvényében  
(b) Késleltetés a felajánlott forgalom függvényében

# Torlódáskezelés (3)

## Sávszélesség igazságos elosztása

- Maximum-minimum igazságosság
  - Egy kiosztás **maximum-minimum igazságos**, ha egy folyamnak kiosztott sávszélesség nem növelhető anélkül, hogy egy másik, kisebb vagy azonos kiosztott sávszélességű folyam sávszélességét ne csökkentenénk.
  - „Jómódú nem gazdagodhat tovább a szegényebbek rovására, de szegény gazdagodhat jómódú rovására.”

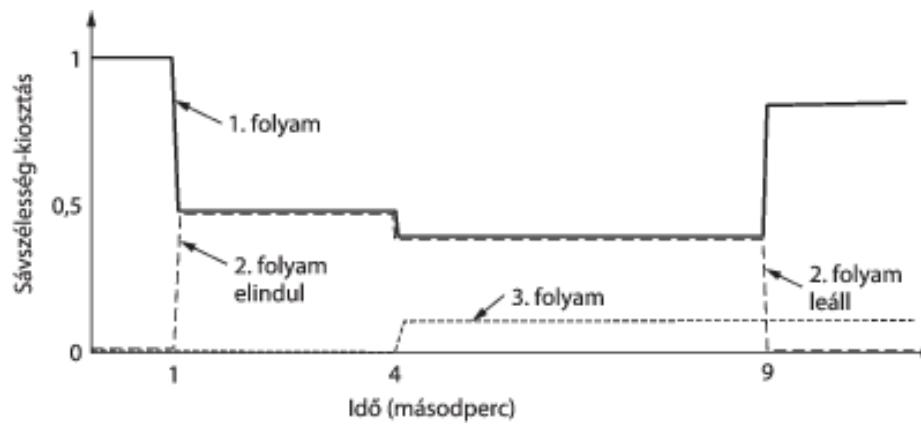


Maximum-minimum sávszélesség-kiosztás négy folyamra  
A példában minden adatkapcsolat egységnyi sávszélességű

# Torlódáskezelés (4)

## Konvergencia

- Gyorsan konvergáljon egy igazságos és hatékony sávszélesség-kiosztás felé
- Az igények gyakran változnak



# Torlódáskezelés (5)

## Visszacsatolás (érzékelés)

- Hogyan észleljük a torlódást? Milyen visszajelzést alkalmazzunk?
  - Explicit/implicit
    - Explicit: a torlódásról egyértelmű jelzést adunk (pl. ECN)
    - Implicit: a torlódást egyéb jelekből „sejtjük” (pl. körülfordulási idő növekedése)
  - Pontos/pontatlan
    - Pontos: útválasztók a küldési sebességet pontosan meghatározzák a küldő számára (pl. XCP)
    - Pontatlan: a küldési sebességet megpróbáljuk beszabályozni (de nem tudjuk az elvárt értéket)
      - Ha van torlódás: sebesség csökkentése
      - Ha nincs torlódás: sebesség növelése

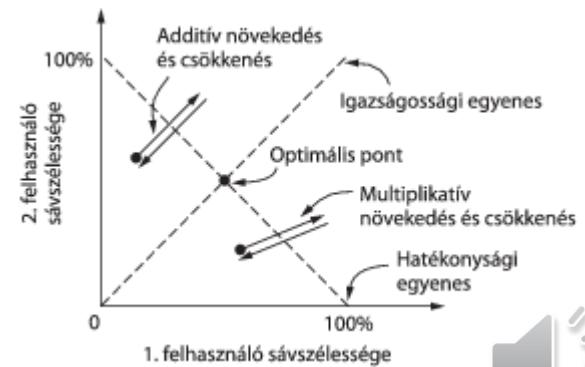
} Szabályozási törvény

Protokoll	Jelzés	Explicit?	Pontos?
XCP	Az előírt sebesség	Igen	Igen
TCP ECN-nel	Figyelmeztetés torlódásra	Igen	Nem
FAST TCP	Végpontok közötti késleltetés	Nem	Igen
Compound TCP	Csomagvesztés és végpontok közötti késleltetés	Nem	Igen
CUBIC TCP	Csomagvesztés	Nem	Nem
TCP	Csomagvesztés	Nem	Nem

# Torlódáskezelés (6)

## Szabályozási törvények

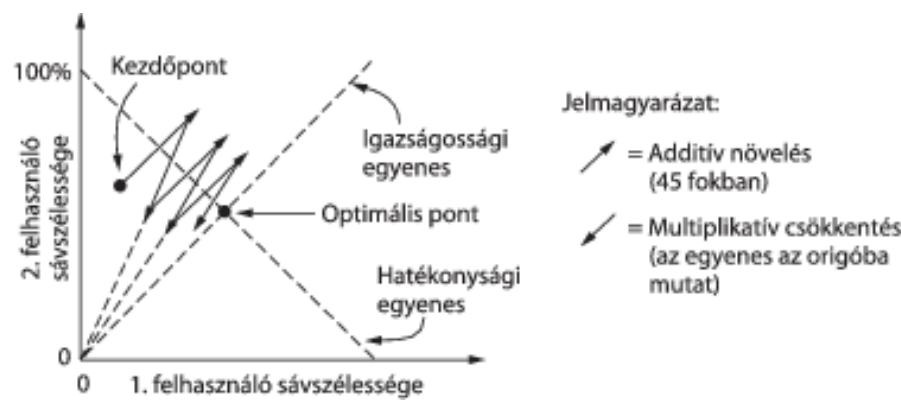
- Alaptörvény („alkotmány”):
  - Ha van torlódás: sebesség csökkentése
  - Ha nincs torlódás: sebesség növelése
- De mi módon csökkentsük/növeljük a sebességet?
  - Additív: a sebességet egy állandó mennyiséggel változtatjuk (pl. +1)
    - Igazságossági egyenessel párhuzamosan mozog
  - Multiplikatív: a sebességet egy állandó mennyiséggel szorozzuk (pl.  $\times 0.5$ )
    - Origóból induló egyenes mentén mozog
- Igazságossági egyenes
  - Ennek mentén igazságos az elosztás
- Hatékonyiségi egyenes
  - Ennek mentén hatékony az elosztás
  - (itt használjuk ki a teljes sávszélességet)



# Torlódáskezelés (7)

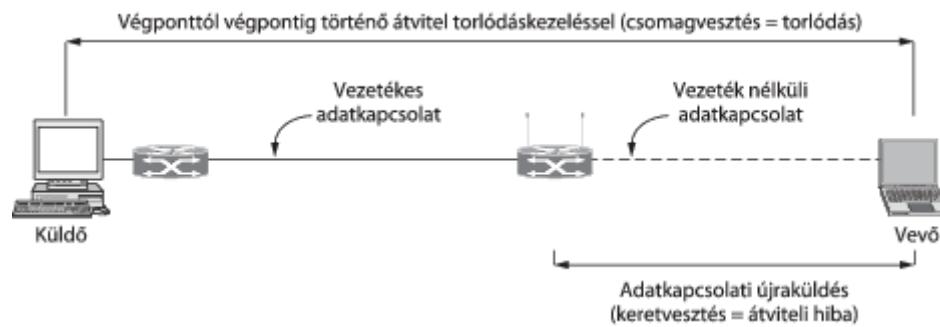
## AIMD szabályozási törvény

- AIMD: Additive Increase Multiplicative Decrease
  - A növelés legyen additív
  - A csökkentés legyen multiplikatív
- Az optimális pontba konvergál:
  - igazságos és
  - hatékony



# Torlódáskezelés vezeték nélküli hálózatokban

- A csomagvesztést a szállítási réteg arra használja, hogy a torlódást jelezze (pl. TCP)
- De a vezeték nélküli hálózatokban mindenkorban nagyon sok csomag elveszik
  - Ez nem a torlódás miatt van (átviteli hiba a rossz adatkapcsolat miatt)
  - Nem fog ez bezavarni a szállítási réteg érzékelésébe? Nem fogja tévesen torlódásként értelmezni?
- Megoldás:
  - A vezeték nélküli adatkapcsolati réteg érzékeli a keretvesztést
    - Azonnal javít: újraküld néhány ms-on belül
  - Az átviteli hiba gyorsan javításra kerül, a szállítási réteg nem veszi észre
    - A szállítási rétegen az időzítők a ms ... s nagyságrendben vannak



# Számítógép-hálózatok



## 9. Szállítási réteg II. UDP és TCP

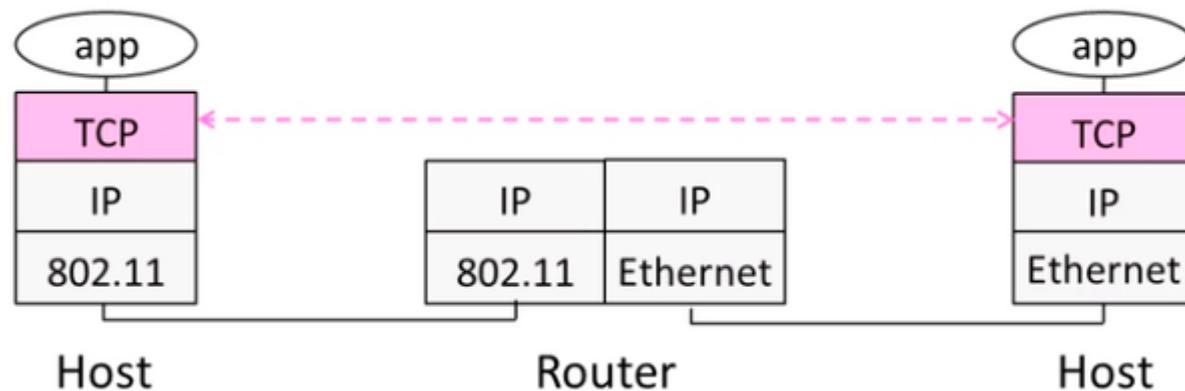
# Tartalom

- Szolgáltatások az Internet szállítási rétegében
- UDP
- TCP



# Szolgáltatások az Internet szállítási rétegében (1)

- Kapcsolat két alkalmazás között



- Lehet üzenet (datagramm) vagy bájtfolyam (stream)
- Lehet megbízható vagy nem megbízható
- Internet: a lehetséges 4 változatból 2 létezik: UDP, TCP

	Nem megbízható	Megbízható
Üzenet	Datagramm (UDP)	x
Bájtfolyam	x	Stream (TCP)



# Szolgáltatások az Internet szállítási rétegében (2)

- Összehasonlítás: TCP / UDP

TCP (bájtfolyam)	UDP (datagramm)
Összeköttetés alapú	Összeköttetés nélküli (datagramm)
A bájtok megbízható módon, sorrendben, 1x kerülnek átvitelre	Üzenet elveszhet, duplikálódhat, sorrendjük keveredhet
Tetszőleges hosszúságú lehet	Korlátos hosszúságú üzenet
Forgalomszabályozás hangolja az adót a vevőhöz	Adó adhat a vevő állapotától függetlenül
Torlódáskezelés hangolja az adót a hálózathoz	Adó adhat a hálózat állapotától függetlenül

Nagyon sok új szolgáltatás

Kb. annyit tud, mint egy IP csomag



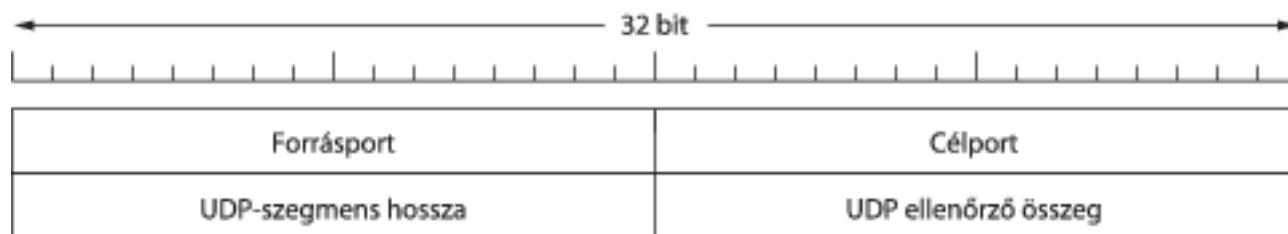
# Az internet szállítási protokolljai: az UDP

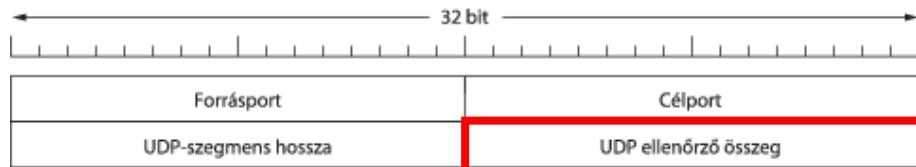
- (UDP: User Datagram Protocol)
- UDP felépítése és használata
- Alkalmazási példák:
  - Távoli eljáráshívás
  - Valós idejű szállítási protokollok
    - RTP
    - RTCP
    - Lejátszás puffereléssel és jitter-szabályozással



# Az UDP fejrész (1)

- Az UDP szegmenseket használ
- Szegmens fejrész: 8 bájt
  - Végpontok azonosítására szolgál
- Forrásport, célport
  - Portok azonosítói
- UDP-szegmens hossza:
  - Fejrész és adat együttes hossza
  - Max: 65515 bájt





## Az UDP fejrész (2)

- UDP ellenőrző összeg

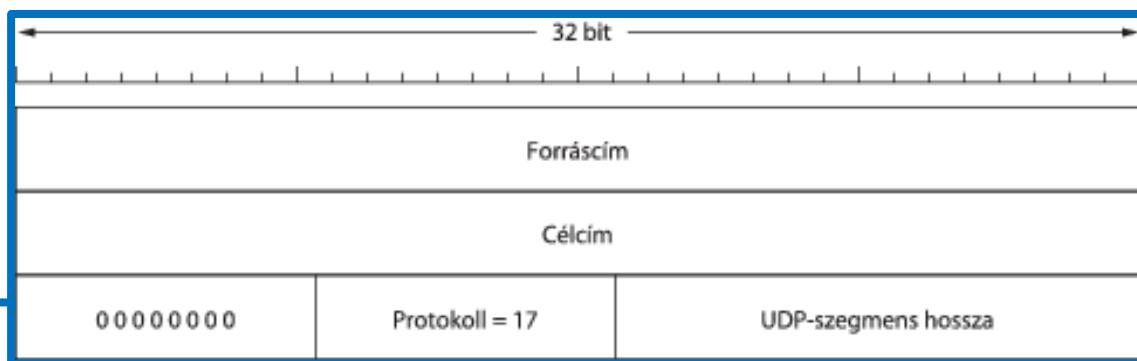
- Nem kötelező
- Számítása
  - Fejrész
  - Adat
  - IP-álfejrész

- IP-álfejrész:

- Forrás és cél IP címek
- Protokoll azonosító
- Szegmens hossz

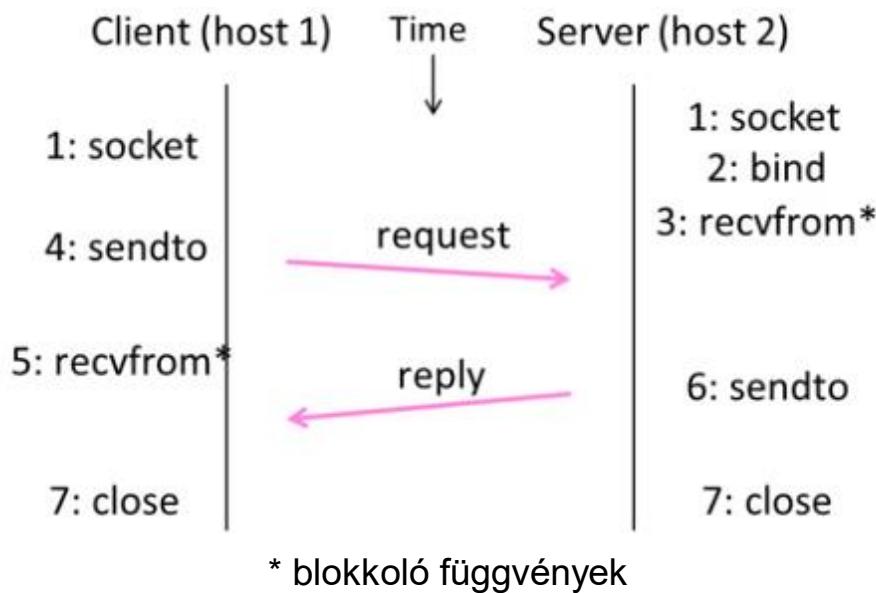
- Célja:

- Esetlegesen tévesen kézbesített csomagok azonosítása



# Az UDP használata

- Csak 2 üzenet halad át a hálózaton
- Nincs ...
  - torlódáskezelés
  - forgalomszabályozás
  - nyugta
  - újraküldés hibás vétel esetén
- Ha jobb szolgáltatásminőség kell:
  - Az alkalmazásnak kell erről gondoskodni
  - (pl. időzítő + újraküldés)



# Távoli eljáráshívás (1)

- Célja:
  - Úgy viselkedjen, mintha hagyományos (helyi) függvényhívás lenne
- Kliens csonk (stub)
  - Úgy viselkedik, mint egy helyi függvényhívás
  - Elvégzi a hálózati feladatokat:
    - Paramétereket üzenetbe pakolja (marshaling)
    - Elküldi az üzenetet
    - Veszi a választ
    - Visszatér az eredménnyel a hívóhoz
- Szerver csonk
  - Veszi az üzenetet
  - A paramétereket kicsomagolja (demarshaling)
  - Meghívja a függvényt
  - Az eredményeket visszaküldi



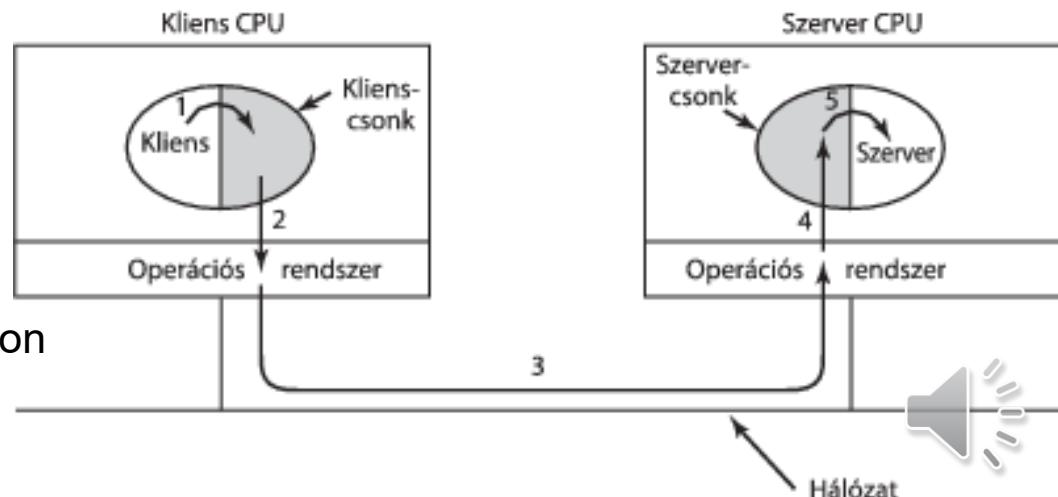
# Távoli eljáráshívás (2)

- A hívás lépései:

1. Kliens meghívja a klienscsonkot
    - Ez egy egyszerű helyi eljáráshívás
  2. Klienscsonk...
    - a paramétereket bepakolja egy üzenetbe és
    - elküldi az üzenetet a szervernek (UDP)
  3. Üzenet a hálózaton megérkezik a szerverhez
  4. Szervercsonk megkapja az üzenetet
  5. Szervercsonk...
    - kipakolja a paramétereket és
    - meghívja a szervereljárást
- **Visszafelé:**
    - Az eredmény fordított útvonalon jut vissza a klienshez (UDP)

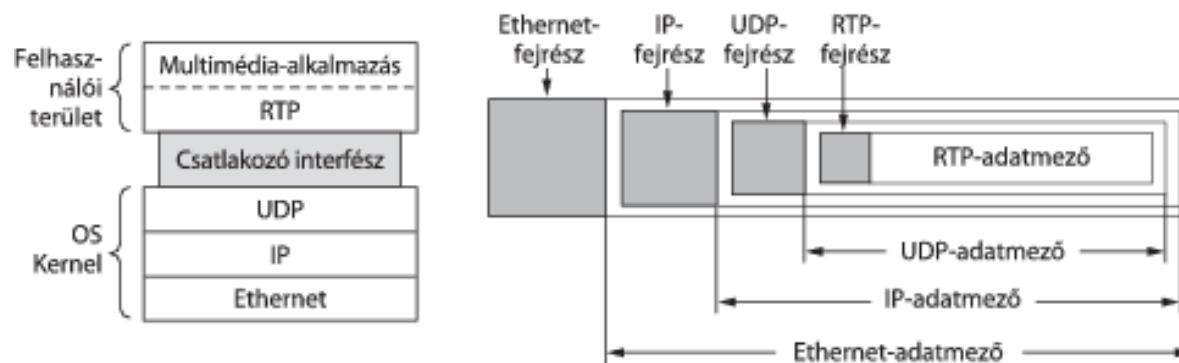
Elveszett csomagok érzékelésére:

- Időzítő a kliensben
  - Időzítő lejártakor: kérés újraküldése
- Potenciális probléma:
- Mellékhatással járó hívások



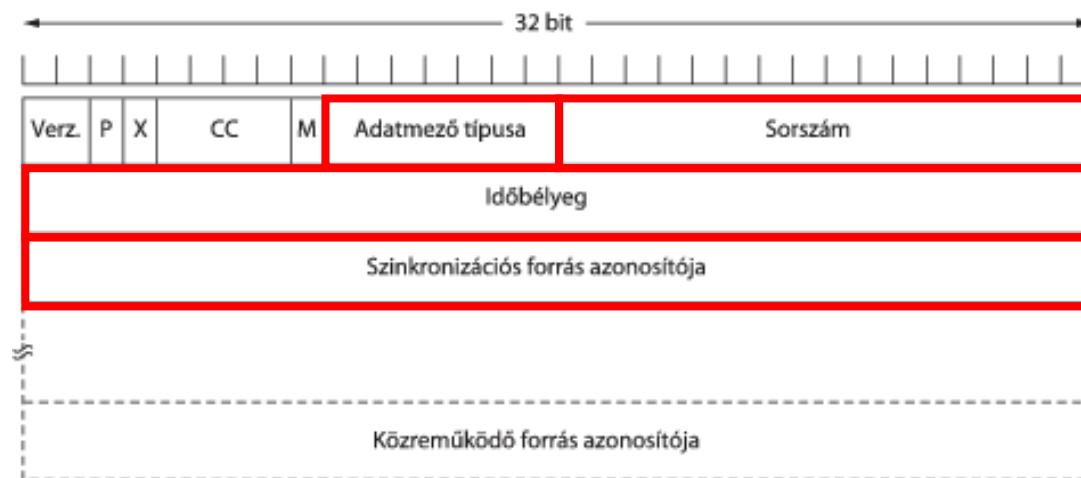
# Valós idejű szállítási protokollok (1)

- Valós idejű multimédiás alkalmazások
  - Internetes telefon
  - Hálózati zeneszolgáltatás
  - Hálózati videoszolgáltatás
  - Videkonferencia
- RTP: Real Time Transfer Protokol
  - Szerver több forrást is használhat (pl. video, audio különféle nyelveken)
  - Adatokat RTP blokkokba kódolják
  - RTP blokkokat UDP-n keresztül továbbítják



# Valós idejű szállítási protokollok (2)

- Fejrész főbb mezői:
  - Sorszám:
    - 1-től egyesével nő, az elveszett csomagok detektálását segíti
  - Adatmező típusa:
    - A használt kódolási algoritmust azonosítja (pl. mp3)
  - Időbélyeg:
    - A csomag első mintájának ideje (a folyam kezdete óta)
  - Szinkronizációs forrás
    - A folyam azonosítására szolgál



# Valós idejű szállítási protokollok (3)

- RTCP: Real-Time Transport Control Protocol
- Mintákat nem szállít
- Feladata:
  1. Visszacsatolást kezeli:
    - Hálózat tulajdonságairól a szervernek (késleltetés, jitter, sávszélesség, torlódás)
    - Ez alapján képes folyamatosan állítani a minőséget (pl. kódolás, felbontás)
  2. Szinkronizációt biztosít
    - A különféle folyamok óráit egymáshoz szinkronizálja
  3. Forrás-információkat szállít
    - Pl. a beszélő neve



# Valós idejű szállítási protokollok (4)

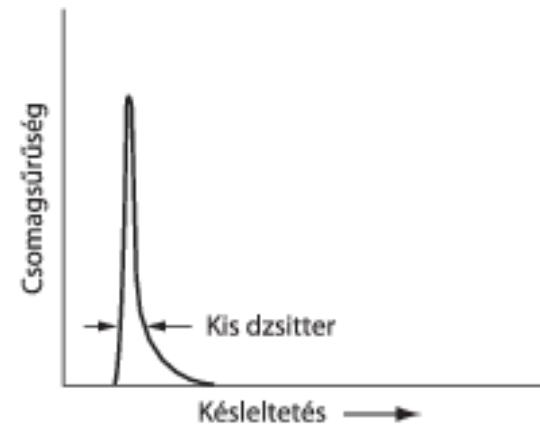
- Átvitel paraméterei:

- Késleltetés

- Sok alkalmazásban nincs lényegi hatása (pl. zenehallgatás)
    - Egyes alkalmazásokban kis értéken kell tartani (pl. telefon)

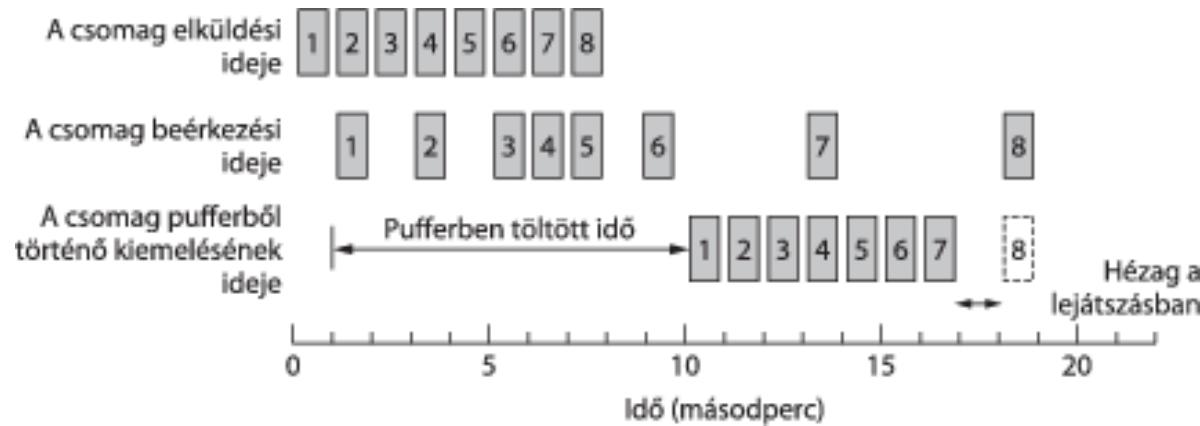
- Késleltetés ingadozása (jitter)

- Nagyon zavaró! Kompenzálni kell a vevőben!

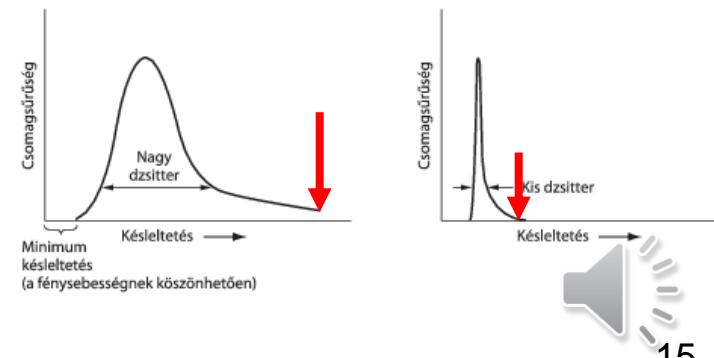


# Valós idejű szállítási protokollok (5)

- Jitter csökkentése:
  - Pufferelés a vevőben



- Lejátszási pont (playback point):
  - Mennyit kell várnia vevőnek a lejátszás indításáig
  - Pl. a csomagok 99%-a megérkezzen
- Lejátszási pont változtatása:
  - Pl. beszédszünetben



# Az internet szállítási protokolljai: a TCP

- A TCP: bevezetés
- A TCP szolgáltatási modellje
- A TCP protokoll
- A TCP szegmens fejrész
- A TCP szolgáltatás felépítése
- A TCP szolgáltatás lebontása
- A TCP összeköttetés-kezelésének modellje
- A TCP csúszóablak
- A TCP időzítéskezelése
- A TCP torlódáskezelése



# A TCP: bevezetés

- (TCP: Transmission Control Protocol)
- Végpontok közötti **megbízható bájtfolyam** biztosítása
- **Alkalmazkodik** az összekapcsolt hálózatok tulajdonságaihoz
- **Ellenálló** sokféle meghibásodással szemben
- Küldő:
  - Folyamból darabok (max 64kB) készítése
    - Gyakorlatban gyakran max. 1460B (Ethernet keret miatt)
  - A darabokból TCP keretek
  - IP datagramokban küldés
- Vevő:
  - TCP datagramból eredeti bájtfolyam visszaállítása
    - Esetleges megváltozott sorrend visszaállítása



# A TCP szolgáltatási modellje (1)

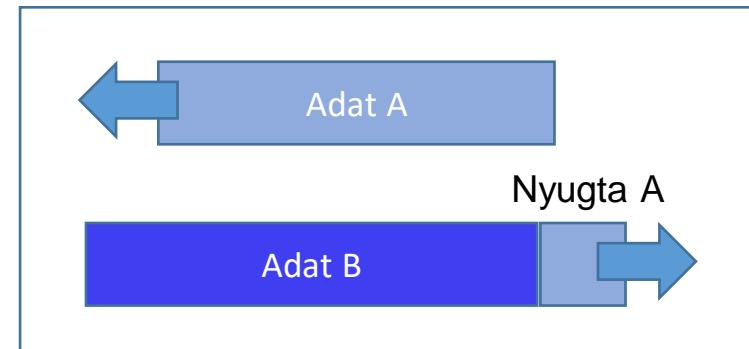
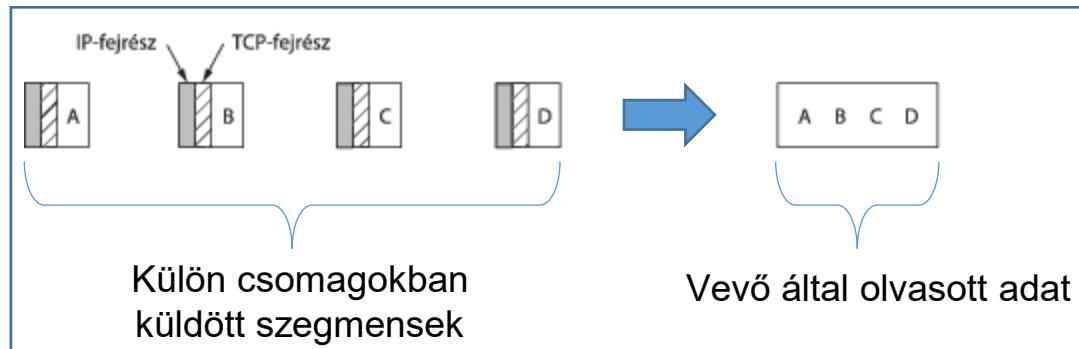
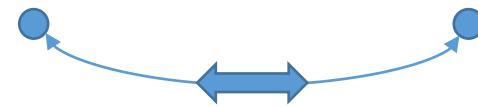
- Küldő és fogadó létrehoz egy **csatlakozót (socket)**
- A **socket címe** három elemből áll:
  - **IP cím** (pl. IPv4 vagy IPv6) ←NSAP
  - **Port száma** (16 bites egész) ←TSAP
  - **Protokoll** (pl. TCP, UDP)
- Szerverek:
  - „**Ismert**” portokat használnak („Well Known Ports”)
  - 1024 alatti portok
  - Csak privilegizált felhasználóknak
  - minden porthoz egy **démon** tartozik (pl. ftp démon vagy http démon)
  - Gyakran egyetlen felügyelő démont használunk (pl. **inetd** – Internet Daemon – a Linux alatt)
    - inetd figyel több portot is
    - szükség esetén elindítja a megfelelő démont
    - nem kell egyszerre sok démonnak feleslegesen futnia
- A kliensek:
  - Általában ideiglenes portokra csatlakoznak
  - Az operációs rendszer választja
  - Véletlenszerű

Port	Protokoll	Alkalmazás
20, 21	FTP	Állományátvitel
22	SSH	Távoli bejelentkezés, Telnet helyett
25	SMTP	E-levelezés
80	HTTP	Világháló (web)
110	POP-3	Távoli e-levél hozzáférés
143	IMAP	Távoli e-levél hozzáférés
443	HTTPS	Biztonságos világháló (HTTP SSL/TLS felett)
543	RTSP	Médialejátszó-vezérlés
631	IPP	Nyomtatómegosztás



# A TCP szolgáltatási modellje (2)

- A TCP (full-)duplex és kétpontos
  - Nincs adatszórás vagy többesküldés!
- Megbízható bájtfolyam
  - Nincs üzenet, nincs üzenethatár
  - A szegmensek kontroll infót is szállítanak (pl. ACK) →piggyback



- Az elküldendő adatot szabad belátása szerinti időben küldi
  - Lehet pufferelni, hogy nagyobb szegmens összegyűljön
  - PUSH bit: kérheti a TCP-t, hogy ne késleltessen (pl. online játék)



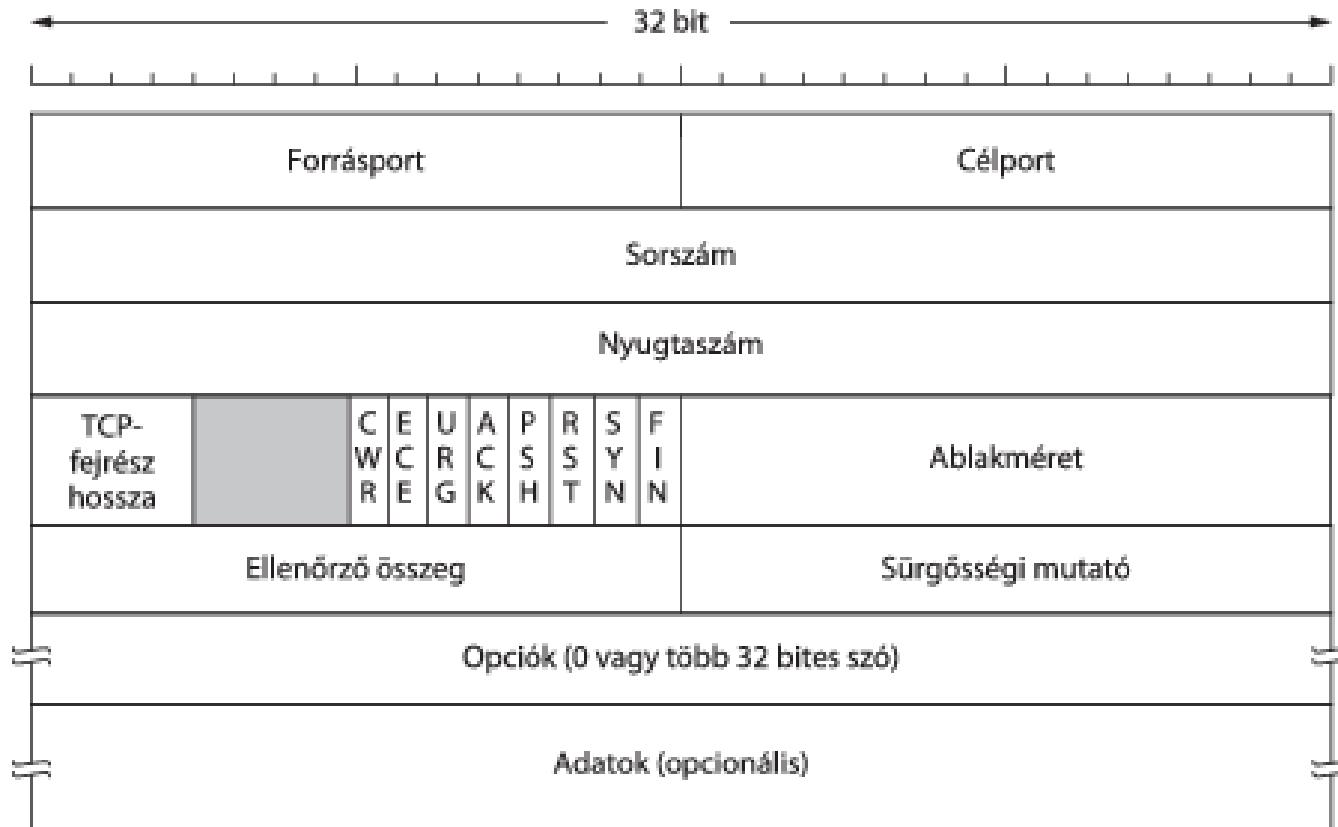
# A TCP protokoll

- minden bájt rendelkezik egy 32 bites sorszámmal
- TCP szegmens
  - Fejrész: 20bájtos kötelező rész + opciók
    - A szegmenst a TCP szoftvere állítja össze
    - Rögtön továbbít VAGY összegyűjthet több írásból származó darabot is
  - Méret korlát:
    - IP adatmező
    - MTU (legnagyobb átvihető adategység)
      - Pl. Ethernet: 1500B
  - Megoldások:
    - Darabolás a hálózatban
      - rontja a teljesítőképességet
      - már nem gyakran használt
    - Útvonal MTU meghatározás
- Csúszóablakos protokoll dinamikus ablakmérettel
  - minden szegmens nyugtázott, hiba esetén újraküldés

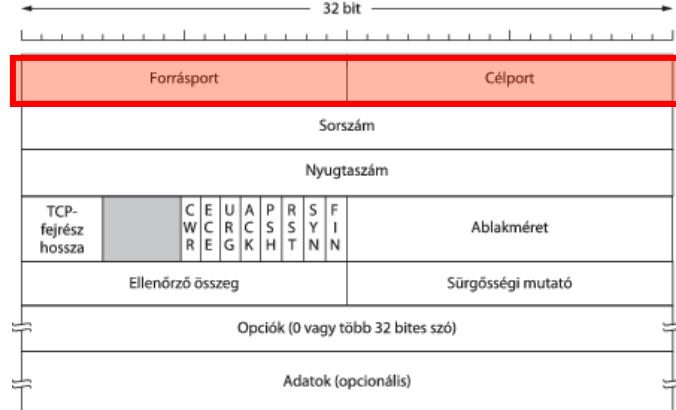


# A TCP-szegmens fejrésze (1)

- 20 bájt (fix fejrész) + opciók + adat
- Adat mérete max.  $65535 - 20 - 20 = 65495$  bájt (IP fejrész: 20B, TCP fejrész: 20B)

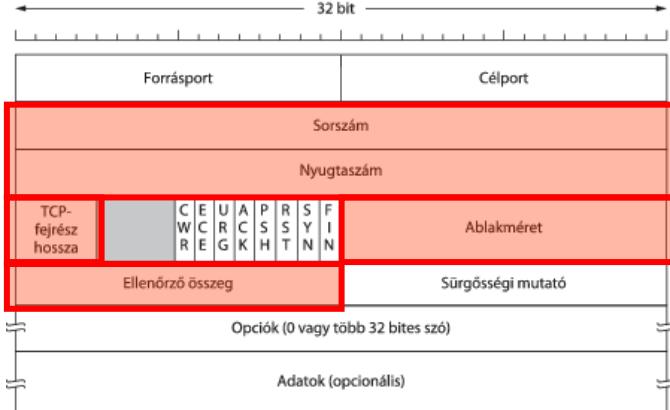


# A TCP-szegmens fejrésze (2)



- Forrásport, célport
  - Helyi végpontok azonosítói
- Végpontot azonosítja:
  - IP cím
  - Port száma
- Az összeköttetést azonosító 5-ös (5-adat, 5-tuple):
  - Protokoll (TCP)
  - Küldő IP címe
  - Küldő port száma
  - Vevő IP címe
  - Vevő port száma

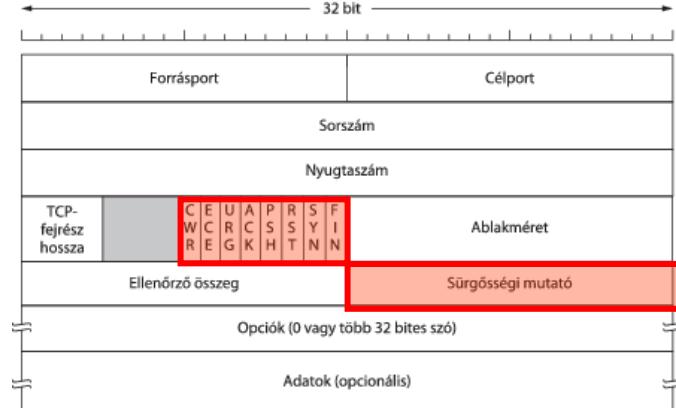
# A TCP-szegmens fejrésze (3)



- Sorszám (SEQ)
  - A korábban elküldött bájtok száma
- Nyugtaszám (ACK)
  - Halmozott nyugta
  - Jelzi, hogy a nyugtázott bájtig az összes adat hiánytalanul megérkezett
  - A nyugta valójában a várt bájt indexét tartalmazza (rendben vett bájt sorszáma + 1)
- TCP fejrész hossza
  - A fejrész hossza 32 bites szavakban mérve
  - Opciók miatt szükséges
- Ablakméret (WIN)
  - A vevőben rendelkezésre álló puffer mérete
- Ellenőrző összeg
  - Hasonlóan UDP-hez
  - Fejrész, adat, álfejrész épségét ellenőri



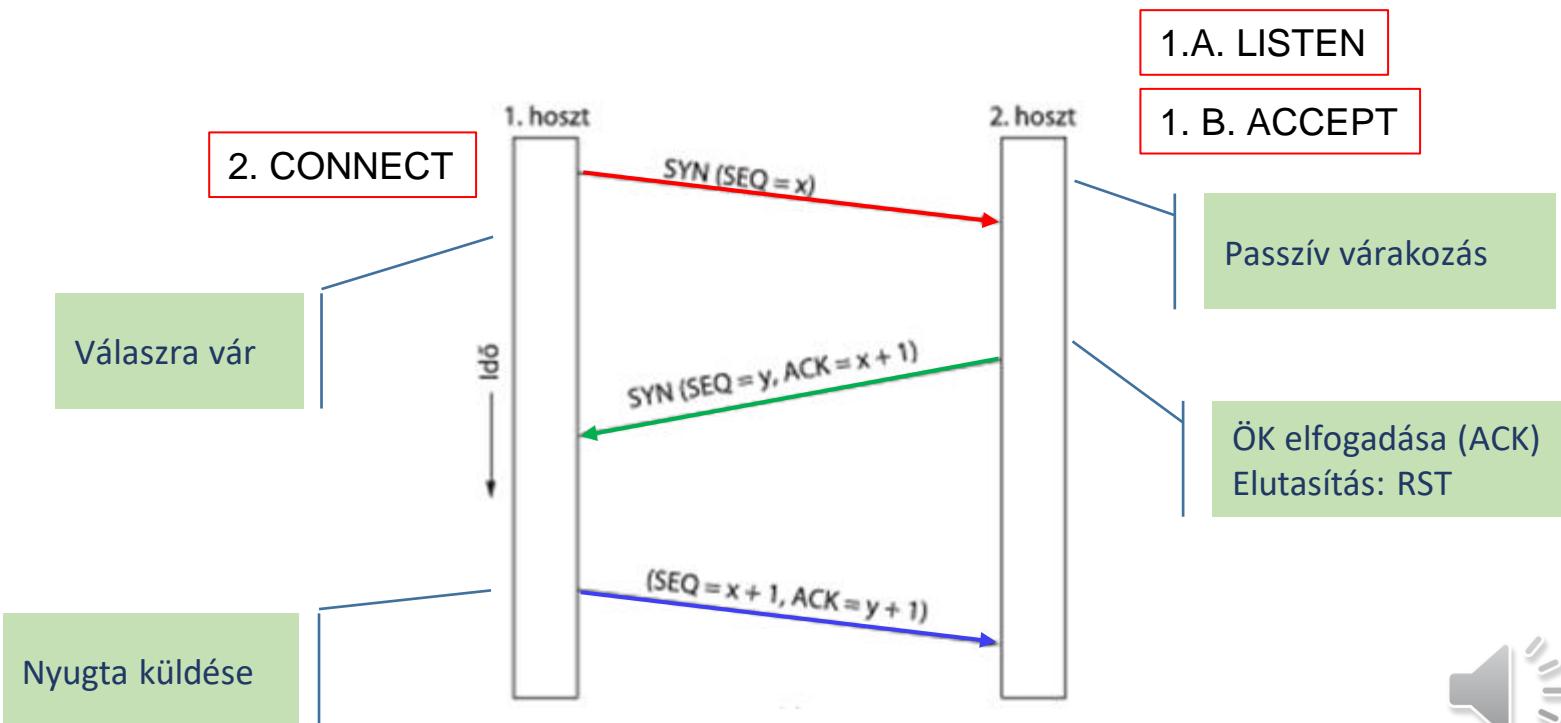
# A TCP-szegmens fejrésze (4)



- CWR, ECE: Torlódás jelzése
  - Vevő ECE bittel jelez: torlódás van a hálózaton
  - Adó CWR bittel jelez vissza, hogy a forgalmat (a torlódási ablaka méretét) lecsökkentette
- URG (URGENT), Sürgősségi mutató
  - URG=1, ha a *Sürgősségi mutatót* használja
  - *Sürgősségi mutató*: a sürgős adat helye a jelenlegi sorszámhöz képest
  - Ritkán használt
- ACK: Jelzi a nyugtaszám érvényességét
- PSH (PUSH): késedelem nélküli továbbítás kérése
  - Ne legyen pufferelés
- RST (RESET): Összeköttetés helyreállítás (valami baj történt)
- SYN: kapcsolat kiépítés
  - Jelzi a CONNECTION REQUEST és CONNECTION ACCEPTED üzeneteket
- FIN: kapcsolat bontása
  - Jelzi, hogy a küldőnek nincs több adata

# A TCP-összeköttetés létesítése

- Háromutas kézfogással
  - →SYN=1, ACK=0
  - ←SYN=1, ACK=1
  - →SYN=0, ACK=1



# A TCP kapcsolat bontása

- Mindkét fél bontja a belőle induló kapcsolatot (irányt)
  - FIN jelzi, hogy nem kíván több adatot küldeni

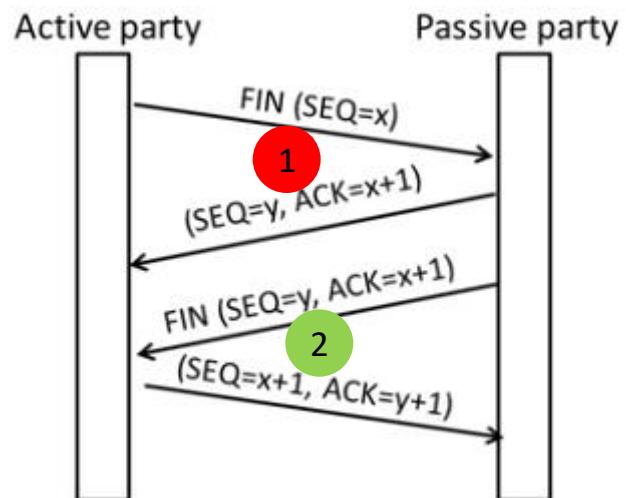
→FIN: részéről nincs több adat

←ACK: OK, vettetem

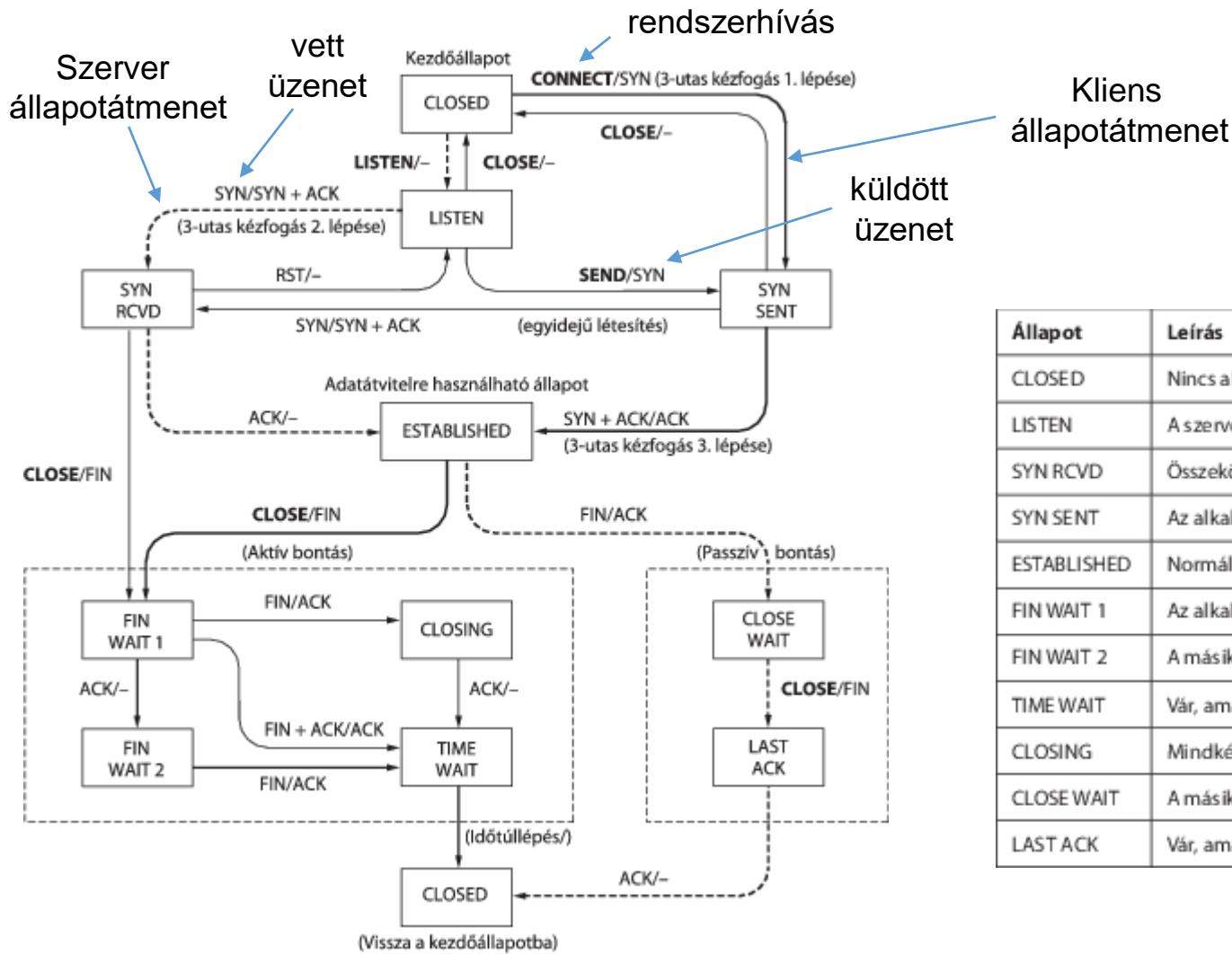
←FIN: Részéről sincs több adat

→ACK: OK, vettetem

- Időzítők használata:
  - Ha FIN-re nem jön válasz, bontja az ÖK-t



# A TCP összeköttetés-kezelésének modellje



Állapot	Leírás
CLOSED	Nincs aktív vagy függő összeköttetés
LISTEN	A szerver egy hívás beérkezésére vár
SYN RCVD	Összeköttetés-kérés érkezett, ACK-ra vár
SYN SENT	Az alkalmazás összeköttetés-létesítést kezdeményezett
ESTABLISHED	Normális adatátviteli állapot
FIN WAIT 1	Az alkalmazás bejelentette, hogy végzett teendőivel
FIN WAIT 2	A másik fél beleegyezett az összeköttetés bontásába
TIME WAIT	Vár, amíg az összes csomag ki nem hal
CLOSING	Mindkét fél egyszerre próbálta bontani az összeköttetést
CLOSE WAIT	A másik fél bontást kezdeményezett
LAST ACK	Vár, amíg az összes csomag ki nem hal

A TCP összeköttetést kezelő állapotgép

(vastag vonal: normál működés. Szaggatott vonal: szerver, folytonos vonal: kliens)

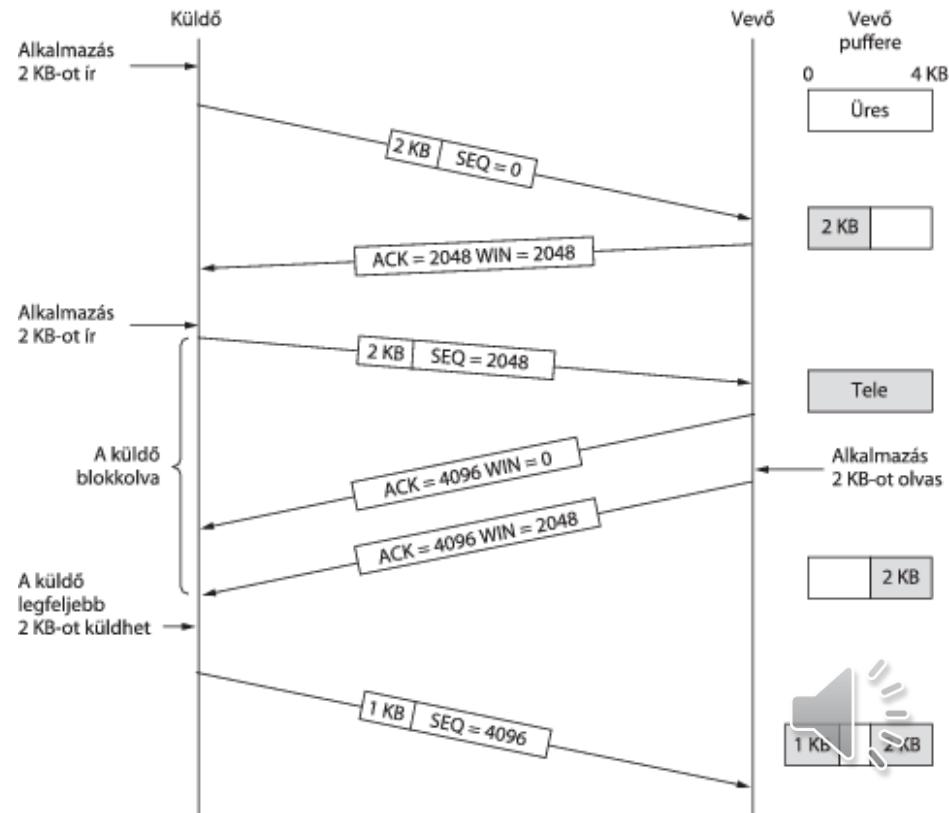


# A TCP csúszóablak

- Különválik az ACK és a vevő pufferméretének kezelése
- ACK: halmozott nyugta
  - A nyugtázott bájtig az összes adat hiánytalanul megérkezett
  - (ACK a következő – várt – sorszámot tartalmazza)
- WIN: vevőben rendelkezésre álló ablakméret = maximális küldhető adatmennyiség
- SEQ: sorszám
  - Az eddig elküldött bájtok száma
  - (ez a csomag nem számít bele)

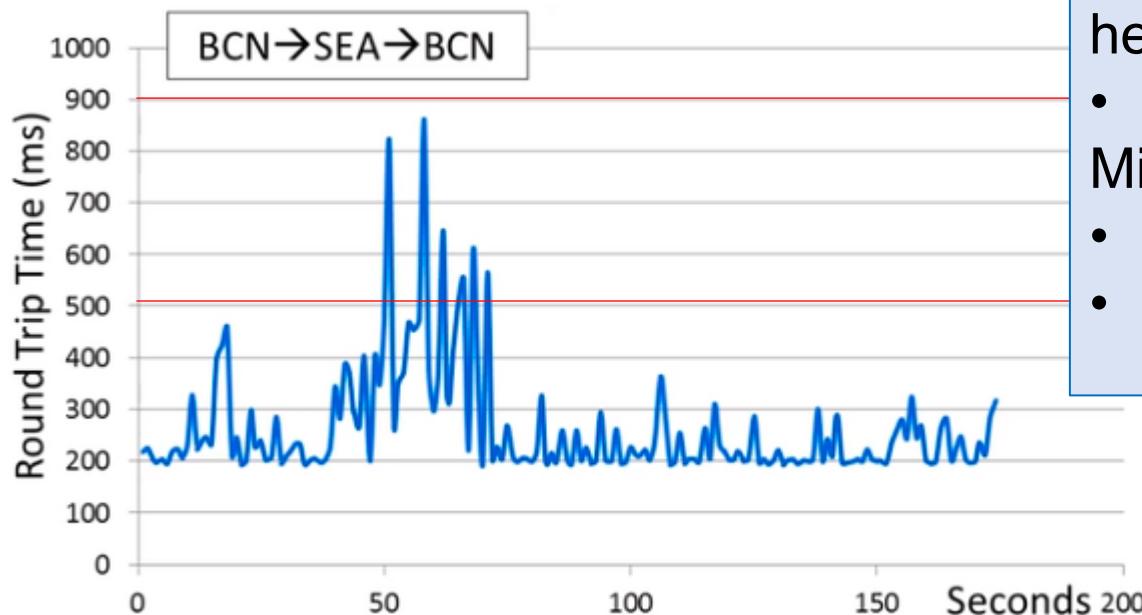
Q: A példában kezdetben honnan tudta az adó, hogy mennyi a vevő ablakmárete?

A: az ÖK létesítésekor megkapta (SYN, SYN/ACK, ACK)



# TCP időzítéskezelése (1)

- Ha a szegmensre nem jön az időzítő lejárta előtt válasz, akkor újraküldjük
- De mekkora legyen az időzítő?
  - 500ms: túl kicsi
  - 900ms: túl nagy



Jobb ötlet:  
Változtassuk az időzítő értékét az aktuális helyzetnek megfelelően:

- adaptív időzítő

Mit használjuk?

- átlag
- szórás



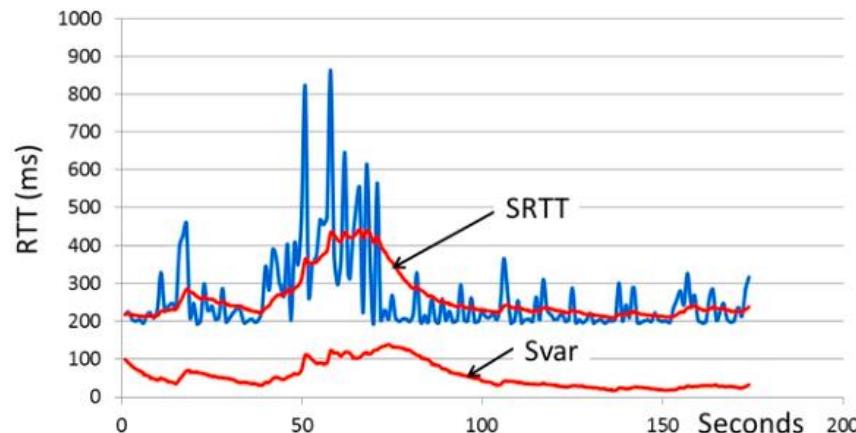
# TCP időzítéskezelése (2)

- Becsüljük meg a késleltetési idő átlagát és szórását
  - Exponenciális átlagolás
  - RTT: mért körülfordulási idő (Round Trip Time)
  - SRTT: simított körülfordulási idő (Smoothed Round Trip Time)
  - sVAR: az RTT simított „varianciája”

$$SRTT_{\text{új}} = \alpha \cdot SRTT_{\text{régi}} + (1 - \alpha)RTT$$

$$sVAR_{\text{új}} = \beta \cdot sVAR_{\text{régi}} + (1 - \beta)|SRTT_{\text{új}} - RTT|$$

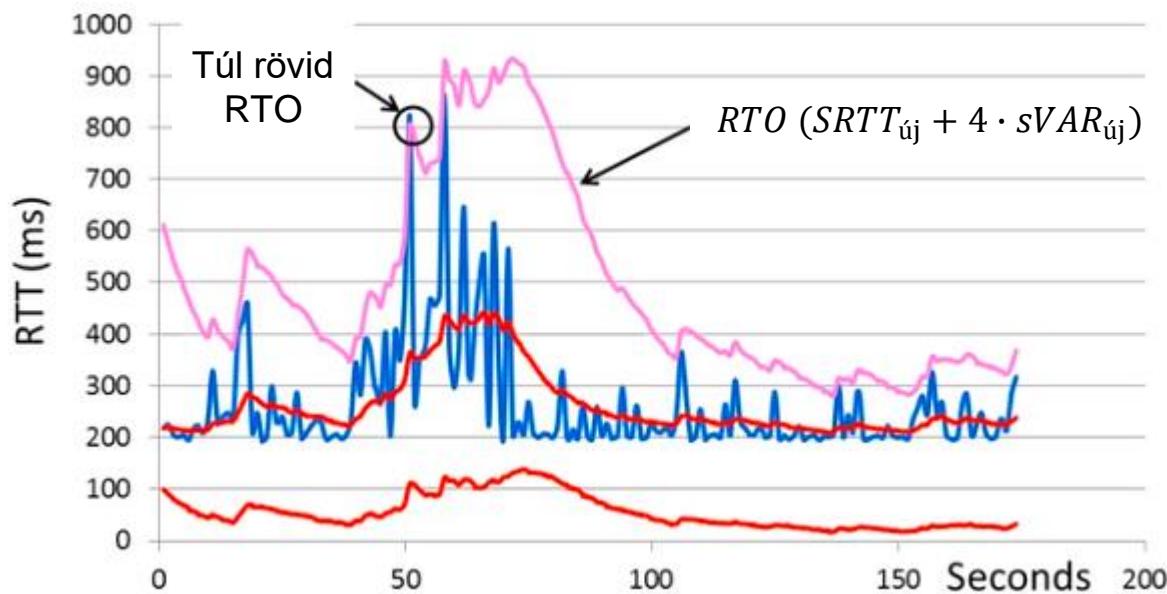
$$\alpha = 7/8$$
$$\beta = 3/4$$



# TCP időzítéskezelése (3)

- Az ismétlési időzítő: RTO (Retransmission TimeOut)
- Az RTO értéke legyen:

$$RTO = SRTT_{\text{új}} + 4 \cdot sVAR_{\text{új}}$$



# TCP időzítéskezelése (4)

- A szegmens valószínűleg elveszett, ha
  - nem jött rá nyugta RTO időn belül
    - Ehhez ki kell várni az RTO időt.
    - Lehetne gyorsabban is detektálni az elveszett csomagot?  
Igen... **3 nyugtamásolat szabály**
  - **legalább 3 ismételt nyugta** (nyugtamásolat) jön egymás után



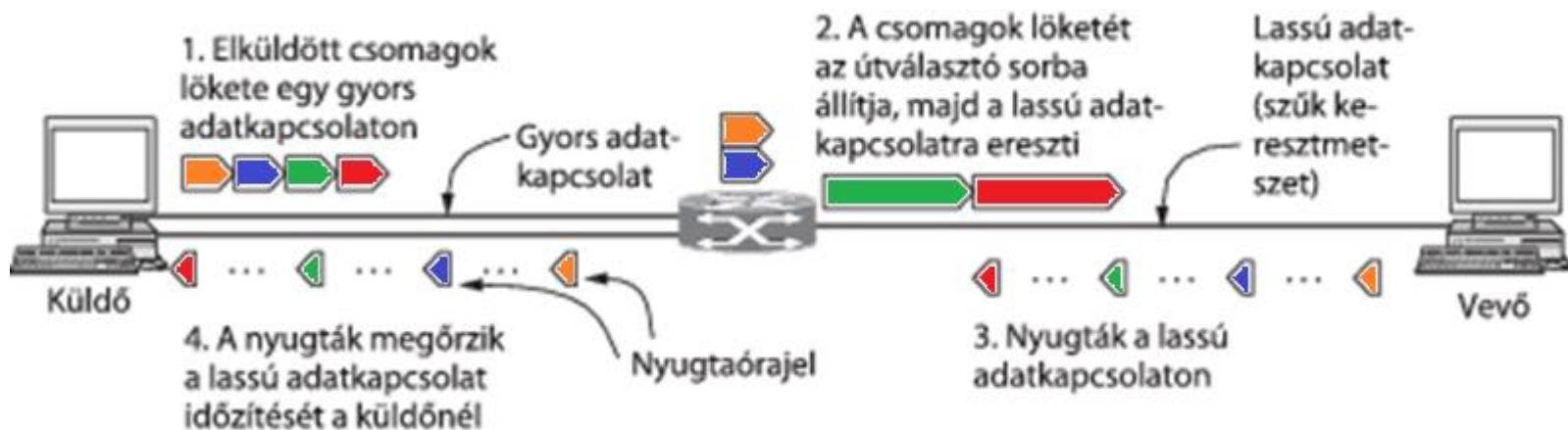
# A TCP torlódáskezelése (1)

- Ha a hálózati terhelés túl nagy
  - Csomagok feltorlódnak az útválasztókban
  - Csomagok késnek, elvesznek
- **TCP: torlódási ablakot használ**
  - Mennyi adat lehet a hálózaton egyszerre?
  - Hasonló a forgalomszabályozási ablakhoz
    - A TCP együtt használja a két ablakot
    - Amelyik kisebb, annak megfelelő mennyiségű adatot küld ki
- Használjuk majd
  - az **AIMD szabályt** a torlódási ablak méretének állítására,
  - **nyugtaórajelet** a „sima” kimenőforgalomért,
  - a „**lassú kezdés**” **algoritmust** a gyors indításért ☺,
  - **gyors újraküldést** (3 nyugtamásolat) a hibák gyors javítására,
  - **gyors helyreállítást** a hibák utáni gyors felépüléshez (MD).



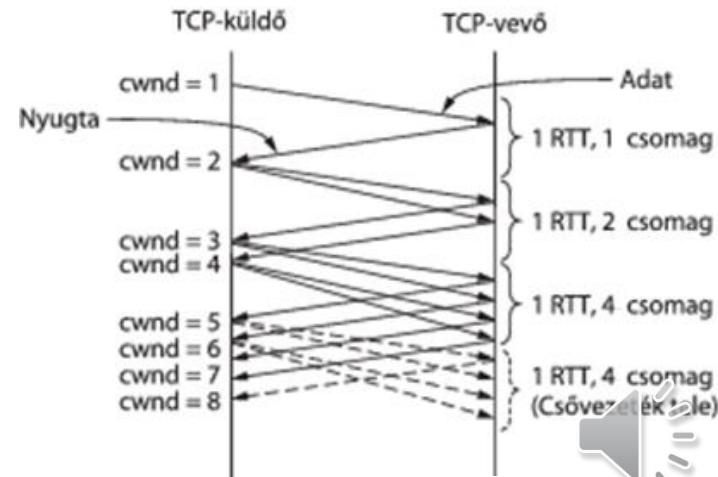
# A TCP torlódáskezelése (2)

- Egy hálózat sebességét a leglassabb szakasza határozza meg
- Egy gyors löketre válaszul **visszaérkező nyugták sebessége** azt mutatja meg, hogy milyen gyorsan lehet a csomagokat a hálózat leglassabb részén átvinni
- Ez a **nyugtaórajel** (ack clock)
- Az adó ennek megfelelő sebességgel ad
  - Elkerüli a felesleges sorbanállást az útvonalválasztókon



# A TCP torlódáskezelése (3)

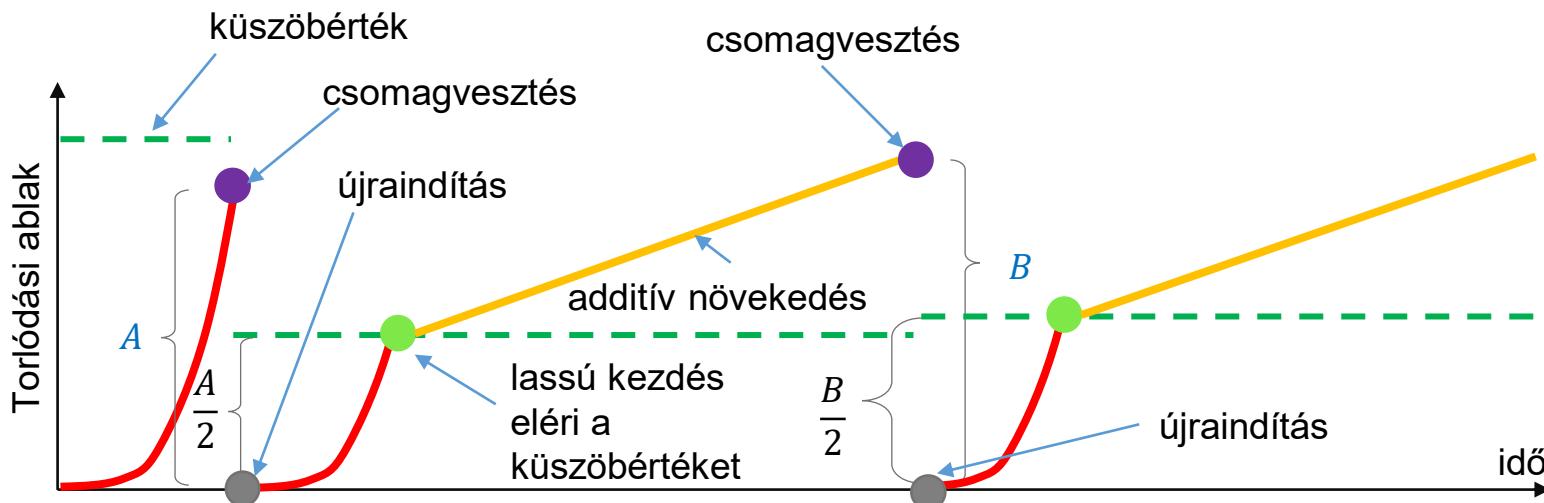
- Hogy állítsuk be a torlódási ablakot?
  - Ha nincs torlódás, növeljük (AI)
  - Ha torlódás van, csökkentsük (MD)
- A kezdeti munkapont elérése AI-val lassú
  - Helyette:
    - indításkor exponenciálisan növekvő ablakméretet használunk
    - 1, 2, 4, 8, 16, 32, ...
    - Ez lesz a „lassú kezdés” algoritmus (Slow Start)
  - Lassú kezdés működése:
    - minden nyugta vételénél:
      - ablakmáret eggyel nő
      - új csomagot küld a régi helyett
      - új csomagot küld az új üres helyre
    - Úton lévő csomagok száma:
      - RTT alatt duplázódik



\*RTT: körülfordulási idő

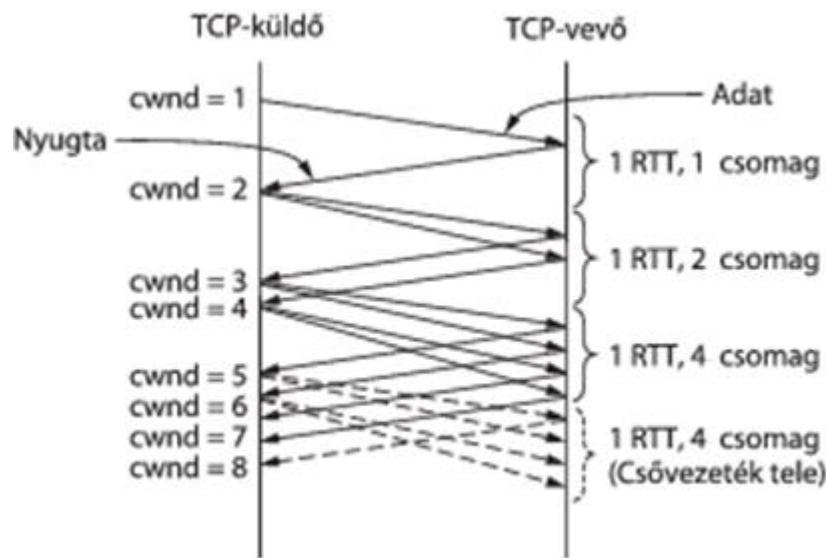
# A TCP torlódáskezelése (4)

- A lassú kezdésnek van egy küszöbértéke, ennél magasabbra a lassú kezdés nem megy
  - A küszöbérték kezdetben magas, pl. a forgalomszabályozási ablak mérete lehet
- Ha a lassú kezdésnél torlódást tapasztalunk (csomagvesztés), akkor
  - visszavesszük a küszöbértéket a jelenlegi torlódási ablak felére és
  - újraindítjuk a lassú kezdést
- Ha a lassú kezdés eléri a küszöbértéket, akkor
  - átkapcsolunk additív növekedésre (AI)
- Ha az additív növekedésnél torlódást tapasztalunk (csomagvesztés), akkor
  - visszavesszük a küszöbértéket a jelenlegi torlódási ablak felére és
  - újraindítjuk a lassú kezdést



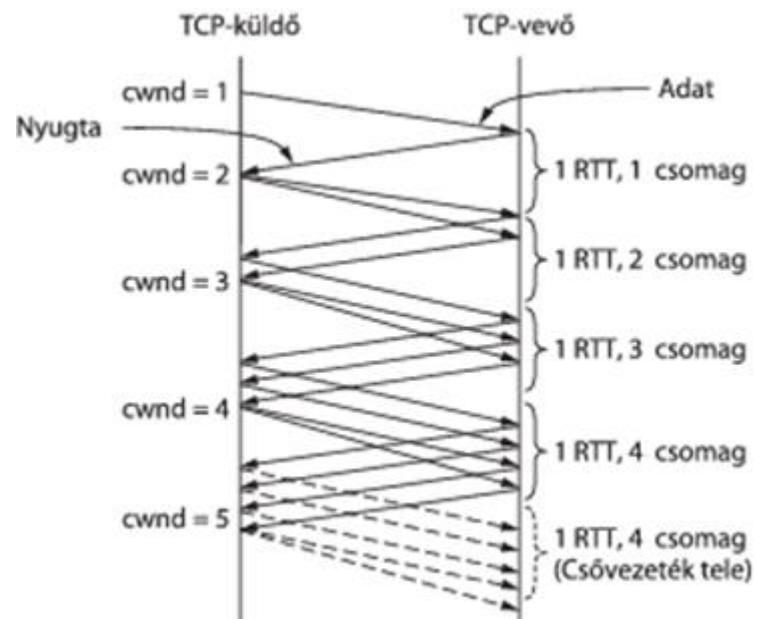
# A TCP torlódáskezelése (5)

Lassú kezdés



Körülfordulásonként a csővezetékben lévő csomagok mennyisége **duplázódik**

Additív növelés

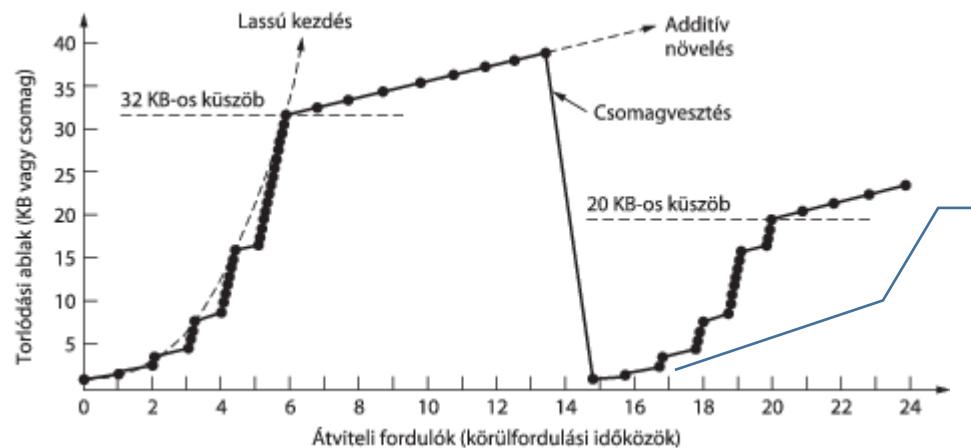


Körülfordulásonként a csővezetékben lévő csomagok **eggyel nő**



# A TCP torlódáskezelése (6)

- Ezt a módszert alkalmazták a TCP Tahoe-ban
- Torlódás/csomagvesztés érzékelése:
  - Időtúllépés vagy
  - 3 nyugtamásolat
    - ekkor a hiányzó csomagokat azonnal újraküldjük
    - Ez a **gyors újraküldés** (fast retransmission)

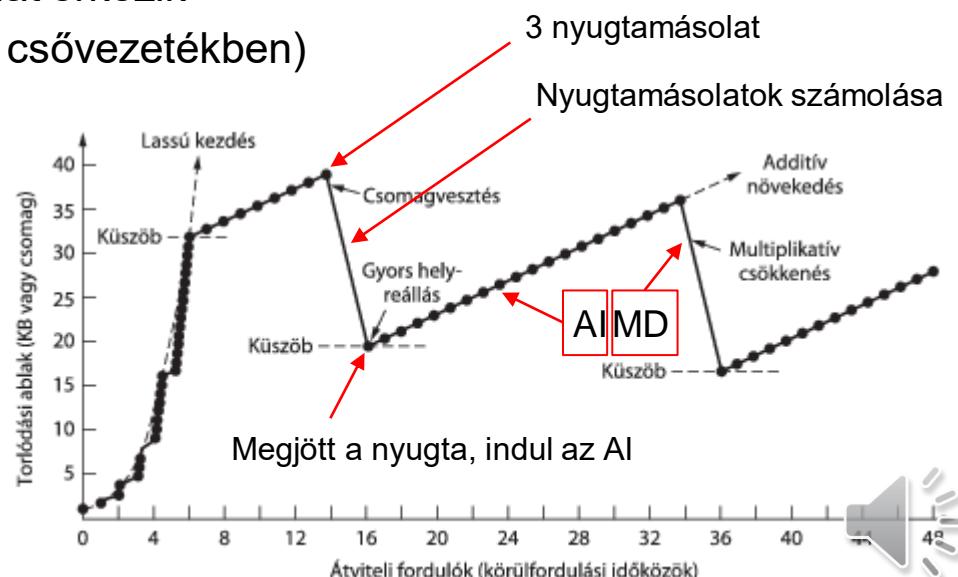


Vajon mindenkor újra kell indítani?  
Nincs okosabb megoldás?

# A TCP torlódáskezelése (7)

Továbbfejlesztés (TCP Reno):

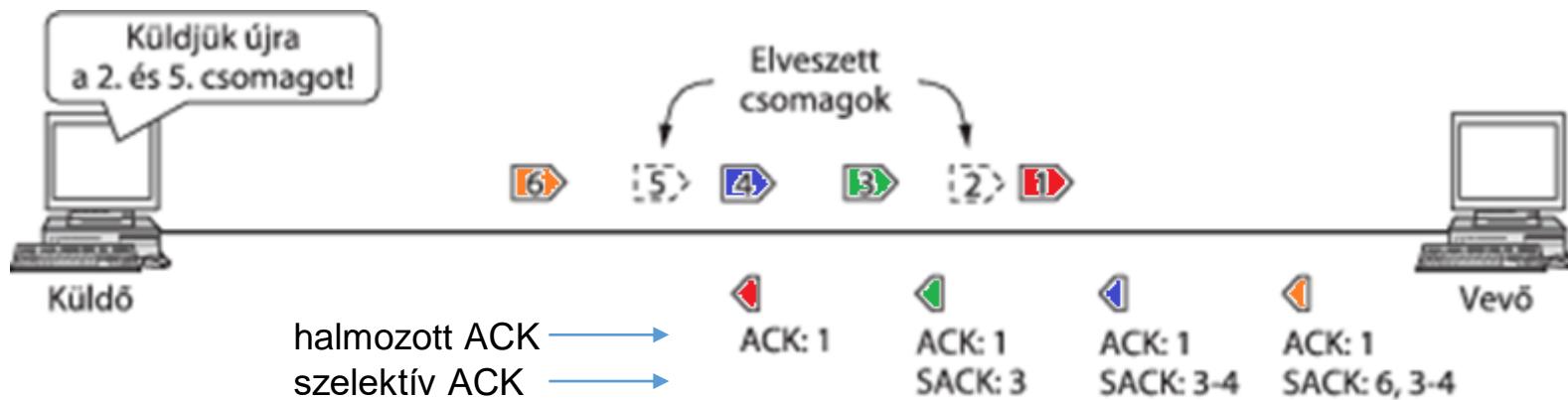
- Ha nyugtamásolatokkal csomagvesztést érzékelünk, az ablakméret felére lépünk vissza (nem pedig 1-re). Ez lesz az MD az AIMD-ben!
- Megoldás: **gyors helyreállítás** (fast recovery)
  - 3 csomagmásolat érkezik. Jelenlegi torlódási ablak mérete: T
  - Számoljuk a beérkező nyugtamásolatokat, de küldés felfüggesztve ( minden nyugtamásolat azt jelzi, hogy egy csomag időközben sikeresen megérkezett)
  - Addig várunk, amíg  $T/2$  nyugtamásolat érkezik (ekkor éppen  $T/2$  üzenet marad a csővezetékben)
  - ezután minden nyugtamásolatra 1 új üzenetet küldünk
  - Ha megjön a nyugta:
    - **gyors helyreállítás vége**
    - (nyugtamásolatok nem jönnek)
  - Additív növeléssel folytatjuk
    - $T/2$  lesz az új ablakméret
    - Küldés indul AI szabállyal



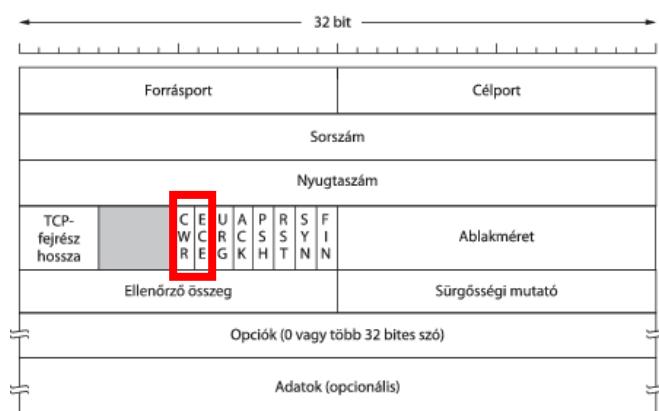
# A TCP torlódáskezelése (8)

Egyéb megoldások:

- Szelektív nyugtázás (SACK)
  - Kummulatív (halmozott) ACK mindig van
  - Szelektív ACK opcionális (széles körben támogatott)



- Explicit torlódásjelzés (ECN – Explicit Congestion Notification)
  - IP útválasztó a csomagban beállítja a torlódás bitet (lásd Differenciált szolgáltatások)
  - A vevő a válasz TCP fejrészben beállítja az **ECE** bitet (ECN Echo)
  - A küldő az **ECE** hatására úgy reagál, mintha csomagvesztés lenne
  - A küldő **CWR** bitet beállítja a következő csomagban (Congestion Window Reduced)



# Összefoglalás

- TCP-ben alkalmazott „trükkök”
  - Ablakozás/csővezeték:
    - használjuk ki a hálózat kapacitását, amennyire csak lehet
    - egyszerre több csomag is lehet úton
  - Forgalomszabályozás:
    - Illesszük az adót a vevőhöz (gyors adó ne árassza el a lassabb vevőt)
    - A vevő szabályozza az adó által elküldhető csomagmennyiséget
      - **forgalomszabályozási ablak** segítségével
  - Torlódáskezelés:
    - a hálózatban keletkezett torlódások kezelése:
      - észlelés (csomagvesztés, explicit torlódásérzékelés, késleltetési idő)
      - az adási sebesség állítása
        - „lassú kezdés”
        - AIMD:
          - **torlódási ablak** additív növelése (+1), amíg csak lehet
          - multiplikatív csökkentése torlódás esetén (/2)
        - Szelektív ACK

