

Számítógépes Hálózatok

3. Előadás: Fizikai réteg II.rész Adatkapcsolati réteg

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

Csatorna hozzáférés módszerei

Multiplexálás

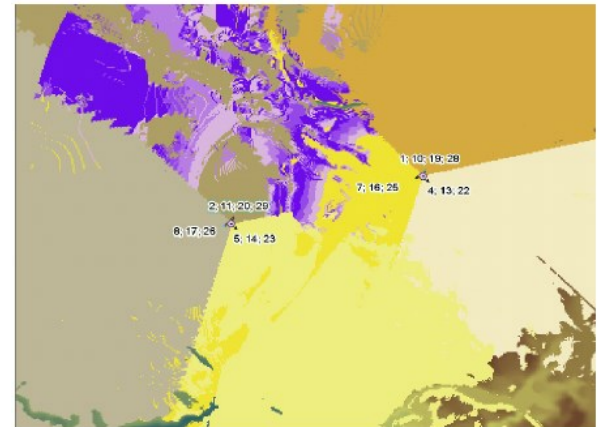
3

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

Térbeli multiplexálás

4

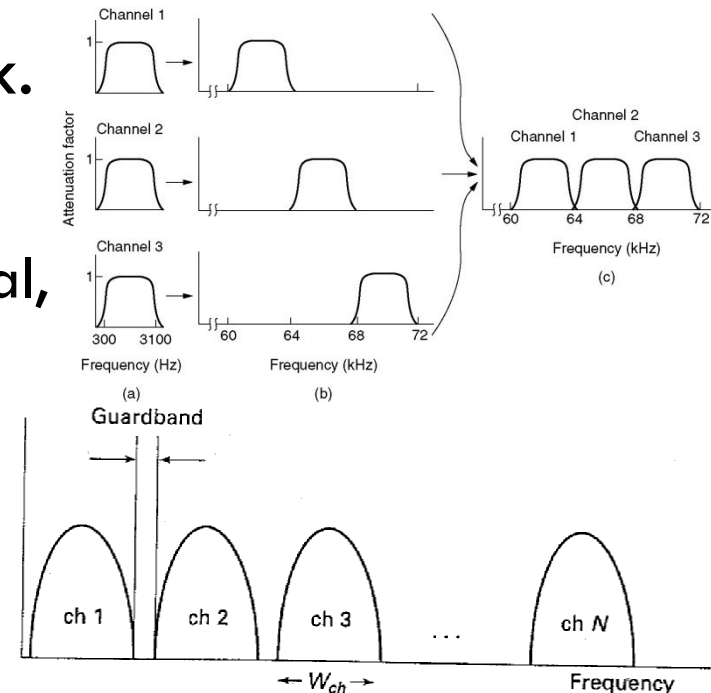
- Ez a legegyszerűbb multiplexálási módszer.
- Angolul **Space-Division Multiplexing**
- Vezetékes kommunikáció esetén minden egyes csatornához külön pont-pont vezeték tartozik.
- Vezeték nélküli kommunikáció esetén minden egyes csatornához külön antenna rendelődik.



Frekvencia multiplexálás

5

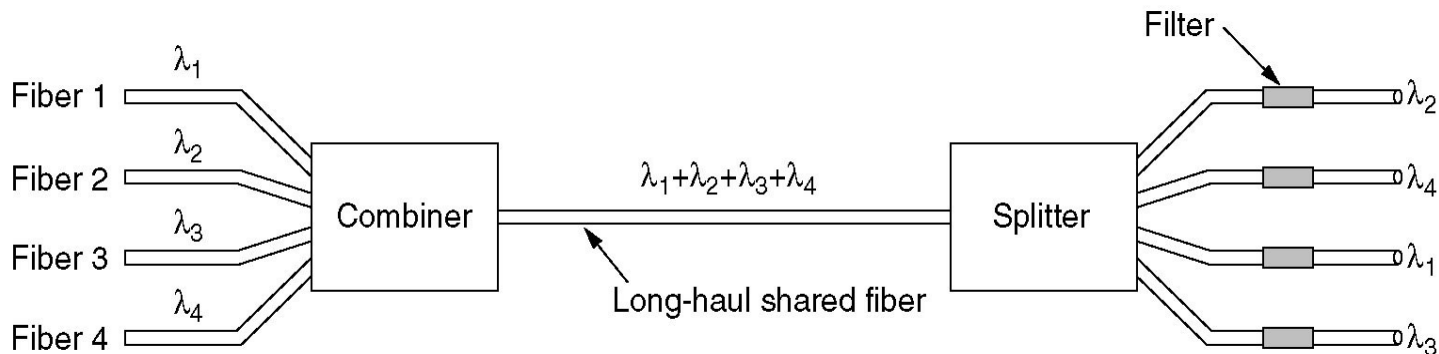
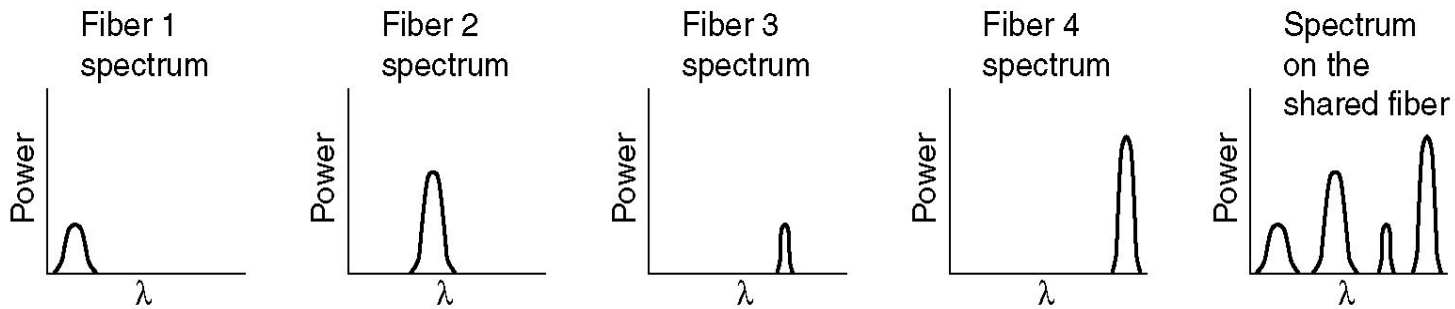
- Olyan módszertan, amelyben egy kommunikációs csatornán több szignál kombinációja adja az átvitelt.
- Minden szignálhoz más frekvencia tartozik.
- Angolul **Frequency-Division Multiplexing**
- Tipikusan analóg vonalon használják.
- Többféle megvalósítása van:
 - ▣ XOR a szignálokon véletlen bitsorozattal,
 - ▣ pseudo véletlen szám alapú választás



Hullámhossz multiplexálás

6

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

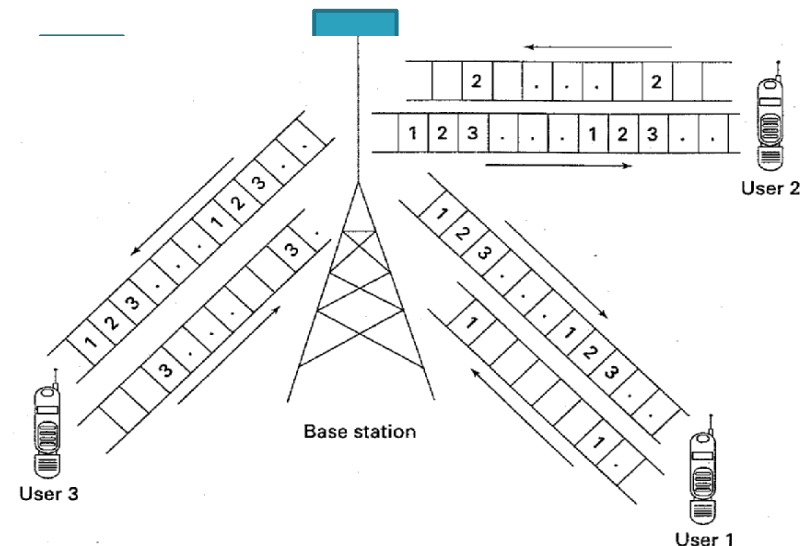


7

-
- ```

graph LR
 A[A] --> TDM[TDM]
 B[B] --> TDM
 C[C] --> TDM
 TDM --> Output[B C A B C A]

```



# Code Division Multiple Access 1/3

8

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

## ALGORITMUS

- minden bitidőt  $m$  darab rövid intervallumra osztunk, ezek a töredékek (angolul *chip*)
- minden állomáshoz egy  $m$  bites kód tartozik, úgynevezett töredéksorozat (angolul *chip sequence*)
- Ha 1-es bitet akar továbbítani egy állomás, akkor elküldi a saját töredéksorozatát.
- Ha 0-es bitet akar továbbítani egy állomás, akkor elküldi a saját töredéksorozatának egyes komplementjét.



# Code Division Multiple Access 2/3

9

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

# Code Division Multiple Access 3/3

10

- szinkron esetben a *Walsh* mátrix oszlopai vagy sorai egyszerű módon meghatároznak egy kölcsönösen ortogonális töredék sorozat halmazt

$$H(2^1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, H(2^2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix},$$

$$\forall k \in \mathbb{N} \wedge k \geq 2: H(2^k) = \begin{bmatrix} H(2^{k-1}) & H(2^{k-1}) \\ H(2^{k-1}) & -H(2^{k-1}) \end{bmatrix}$$

# Code Division Multiple Access példa

11

## A állomás

Chip kódja legyen  $(1, -1)$ .

Átvitelre szánt adat legyen

1011

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


## B állomás

Chip kódja legyen  $(1, 1)$ .

Átvitelre szánt adat legyen

0011

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


$$((1 + (-1), (-1) + (-1)), ((-1) + (-1), 1 + (-1)), (1 + 1, (-1) + 1), (1 + 1, (-1) + 1)) = (0, -2, -2, 0, 2, 0, 2, 0)$$

# Code Division Multiple Access példa

12

$$((1+(-1), (-1)+(-1)), ((-1)+(-1), 1+(-1)), (1+1, (-1)+1), (1+1, (-1)+1)) = ((0, -2), (-2, 0), (2, 0), (2, 0))$$

## Vevő 1

Ismeri B chip kódját:  $(1, 1)$ .

1. Visszakódolás az ismert kóddal:  
 $((0, -2) * (1, 1), (-2, 0) * (1, 1), (2, 0) * (1, 1), (2, 0) * (1, 1))$
2. Kapott  $(-2, -2, 2, 2)$  eredmény értelmezése:  
 $(-, -, +, +)$ , azaz 0011 volt az üzenet B-től.

## Vevő 2

Ismeri A chip kódját:  $(1, -1)$ .

1. Visszakódolás az ismert kóddal:  
 $((0, -2) * (1, -1), (-2, 0) * (1, -1), (2, 0) * (1, -1), (2, 0) * (1, -1))$
2. Kapott  $(2, -2, 2, 2)$  eredmény értelmezése:  
 $(+, -, +, +)$ , azaz 1011 volt az üzenet A-tól.

# Médium többszörös használata

## összefoglalás

13

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

ADATKAPCSOLATI RÉTEG



# Adatkapcsolati réteg

15



- Szolgáltatás
  - ▣ Adatok keretekre tördelése: határok a csomagok között
  - ▣ Közeghozzáférés vezérlés (MAC)
  - ▣ Per-hop megbízhatóság és folyamvezérlés
- Interfész
  - ▣ Keret küldése két közös médiumra kötött eszköz között
- Protokoll
  - ▣ Fizikai címezés (pl. MAC address, IB address)
- Példák: Ethernet, Wifi, InfiniBand

# Adatkapcsolati réteg

16



## □ Funkciók:

- ▣ Adat blokkok (**keretek/frames**) küldése eszközök között
- ▣ A fizikai közeghez való hozzáférés szabályozása

## □ Legfőbb kihívások:

- ▣ Hogyan **keretezzük** az adatokat?
- ▣ Hogyan ismerjük fel a **hibát**?
- ▣ Hogyan vezéreljük a **közeghozzáférést (MAC)**?
- ▣ Hogyan oldjuk fel vagy előzzük meg az **ütközési** helyzeteket?



# Adatkapcsolati réteg

## Feladatai

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

# Keret képzés / Keretezés / Framing

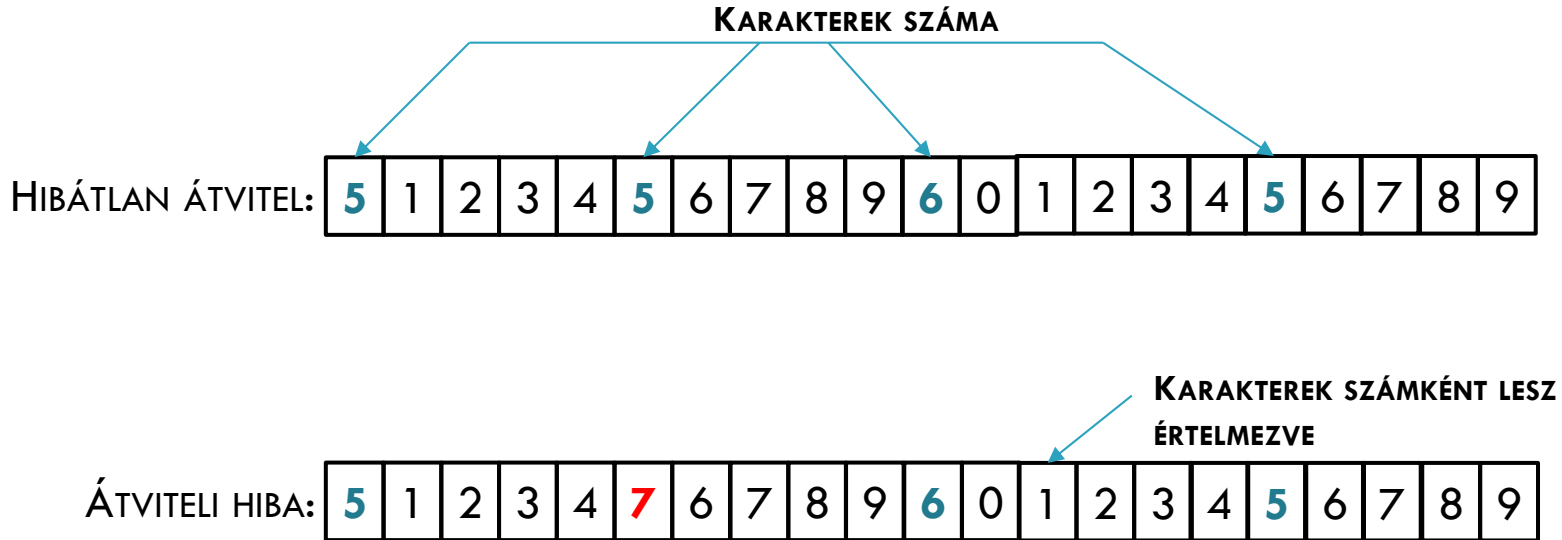
# Keret képzés/Keretezés/Framing

19

- A bitek kódolását a fizikai réteg határozza meg
- A következő lépés az adatblokkok „kódolása”
  - ▣ Csomag-kapcsolt hálózatok
    - Minden csomag útvonal (routing) információt is tartalmaz
    - Az adatakat ismernünk kell a fejlécek olvasásához
  - ▣ a fizikai réteg nem garantál hibamentességet, az adatkapcsolati réteg feladata a hibajelzés illetve a szükség szerint javítás
    - Megoldás: keretekre tördelése a bitfolyamnak, és ellenőrző összegek számítása
  - ▣ a keretezés nem egyszerű feladat, mivel megbízható időzítésre nem nagyon van lehetőség
- Keret képzés fajtái
  - ▣ Bájt alapú protokollok
  - ▣ Bit alapú protokollok
  - ▣ Óra alapú protokollok

# Bájt alapú: Karakterszámlálás

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



# Bájt alapú: Bájt beszúrás (Byte Stuffing)

21



- Egy speciális **FLAG** bájt (jelölő bájt) jelzi az adat keret elejét és végét
  - ▣ Korábban két speciális bájtot használtak: egyet a keret elejéhez és egyet a végéhez
- Probléma: Mi van, ha a **FLAG** szerepel az adat bájtok között is?
  - ▣ Szúrjunk be egy speciális **ESC** (Escape) bájtot az „adat” **FLAG** elé
  - ▣ Mi van ha **ESC** is szerepel az adatban?
    - Szúrjunk be egy újabb **ESC** bájtot elé.
  - ▣ Hasonlóan a C stringeknél látottakhoz:
    - `printf(“You must \”escape\” quotes in strings”);`
    - `printf(“You must \\escape\\ forward slashes as well”);`
- Pont-pont alapú protokollok használják: modem, DSL, cellular, ...

# Bájt beszúrás példa

KERETEZENDŐ ADAT

|   |   |   |   |   |         |       |
|---|---|---|---|---|---------|-------|
| H | E | L | L | O | [SPACE] | [ESC] |
|---|---|---|---|---|---------|-------|



BÁJT BESZÚRÁS

KERETEZETT ADAT

|        |   |   |   |   |   |         |       |       |        |
|--------|---|---|---|---|---|---------|-------|-------|--------|
| [FLAG] | H | E | L | L | O | [SPACE] | [ESC] | [ESC] | [FLAG] |
|--------|---|---|---|---|---|---------|-------|-------|--------|

# Bit alapú: Bit beszúrás (Bit stuffing)

23

01111110

Adat

01111110

- Minden keret speciális bitmintával kezdődik és végződik (hasonlóan a bájt beszúráshoz)
  - ▣ A kezdő és záró bitsorozat ugyanaz
  - ▣ Például: 01111110 a High-level Data Link Protocol (HDLC) esetén
- A Küldő az adatban előforduló minden 11111 részsorozat elé 0 bitet szúr be
  - ▣ Ezt nevezzük bit beszúrásnak
- A Fogadó miután az 11111 részsorozattal találkozik a fogadott adatban:
  - ▣ 111110 → eltávolítja a 0-t (mivel ez a beszúrási eredménye volt)
  - ▣ 111111 → ekkor még egy bitet olvas
    - 1111110 → keret vége
    - 1111111 → ez hiba, hisz ilyen nem állhat elő a küldő oldalon. Eldobjuk a keretet!
- Hátránya: legrosszabb esetben 20% teljesítmény csökkenés
- Mi történik ha a záró bitminta meghibásodik?

# Példa bit beszúrásra



AZ ÁTVITELRE SZÁNT BITSOROZAT BITBESZÚRÁS ELŐTT → 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

AZ ÁTVITELRE SZÁNT BITSOROZAT BITBESZÚRÁS  
UTÁN (FEJLÉC NÉLKÜL) → 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

BESZÚRT BITEK

A VEVŐNÉL MEGJELENŐ ÜZENET A REDUNDÁNS BITEK  
ELTÁVOLÍTÁSA UTÁN → 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0



# Óra alapú keretezés: SONET

## □ Synchronous Optical Network

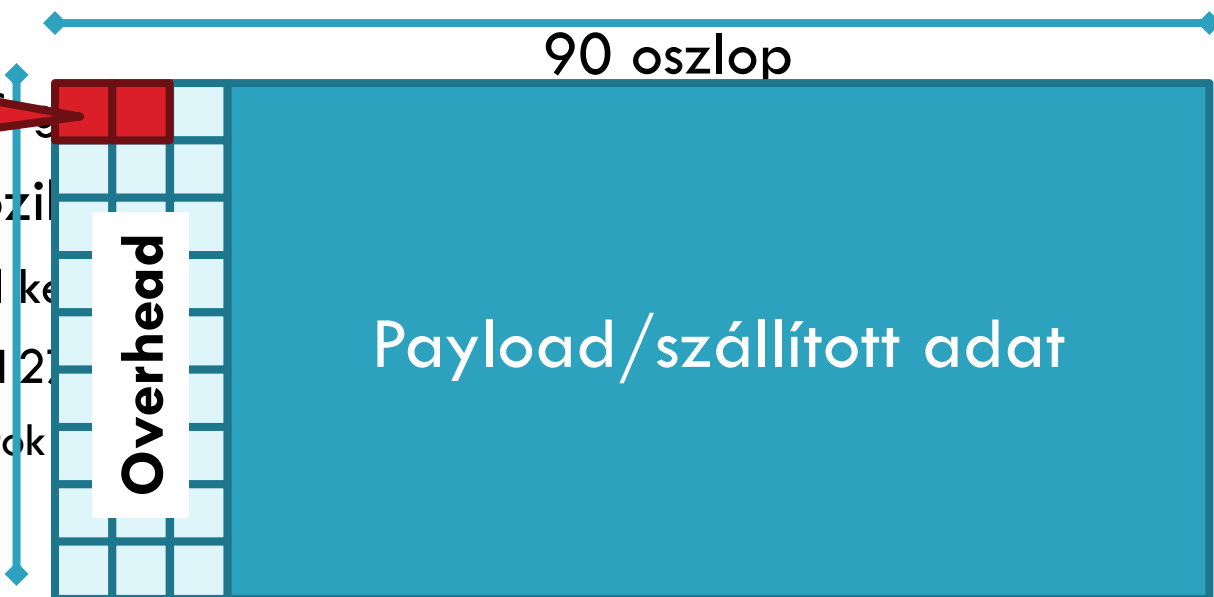
- Nagyon gyors optikai kábelben való átvitel
- STS- $n$ , e.g. STS-1: 51.84 Mbps, STS-768: 36.7 Gbps

## □ Az STS-1 keretei rögzített mérettel rendelkeznek

- $9 \times 90 = 810$  bájt  $\rightarrow$  810 bájt fogadása után újabb keret-kezdő mintázat keresése

**Speciális kezdő  
mintázat**

- A bitek NRZ kódolással kerülnek átvitelre
- Payload egy speciális 125 ns-es időtartamra van korlátozva
- A hosszú 0 és 1 sorozatok



# Hiba felügyelet

# Zaj kezelése

27

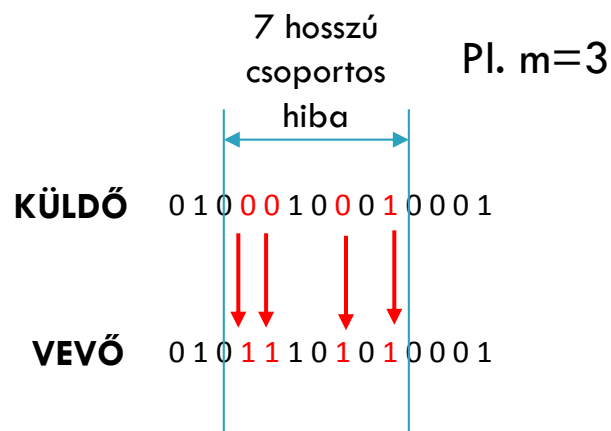
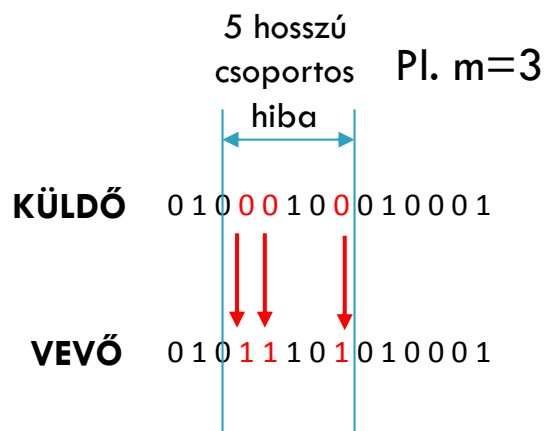
- A fizikai világ eredendően zajos
  - ▣ Interferencia az elektromos kábelek között
  - ▣ Áthallás a rádiós átvitelek között, mikrosütő, ...
  - ▣ Napviharok
- Hogyan detektáljuk a bithibákat az átvitelben?
- Hogyan állítsuk helyre a hibát?

# Bithibák definíciók és példák

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

|       |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|
| KÜLDŐ | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
|       |   |   |   |   |   |   |   |   |
| VEVŐ  | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

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



# Naiv hibadetektálás

29

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


# Paritás Bit

30

- Ötlet: egy extra bitet adunk a bitsorozathoz úgy, hogy az egyesek száma végül **páros** legyen
  - ▣ Példa: 7-bites ASCII karakterek + 1 paritásbit

0101001 1 1101001 0 1011110 1 0001110 1 0110100 1

10



- 1-bit hiba detektálható
- néha 2-bit hiba is detektálható
- Nem megbízható burstös hibák esetén

# Hiba vezérlés

## □ Stratégiák

### ▣ Hiba javító kódok

- Előre hibajavítás
- Forward Error Correction (FEC)
- kevésbé megbízható csatornákon célszerűbb

### ▣ Hiba detektálás és újraküldés

- Automatic Repeat Request (ARQ)
- megbízható csatornákon olcsóbb

# Hiba vezérlés

## □ Célok

### ▣ Hiba detektálás

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

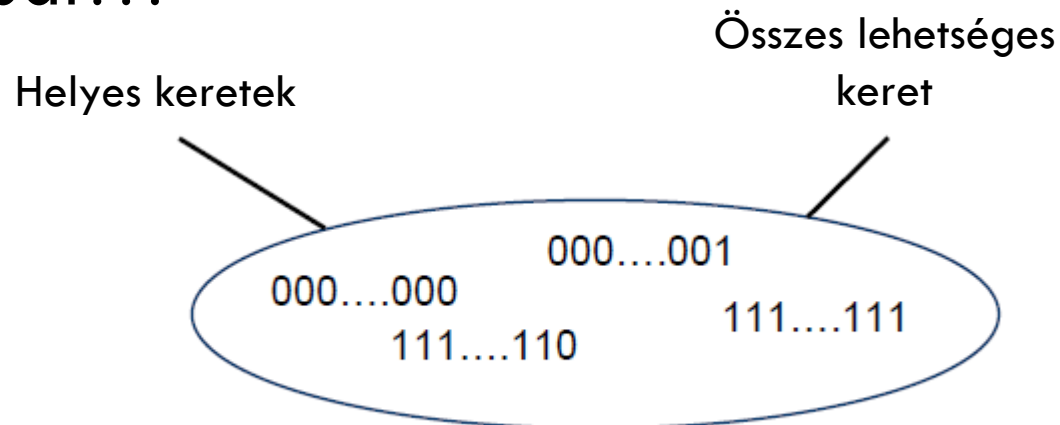
### ▣ Hiba javítás

- Hiba detektálás nélkül
  - ▣ Pl. hangátvitel



# Redundancia

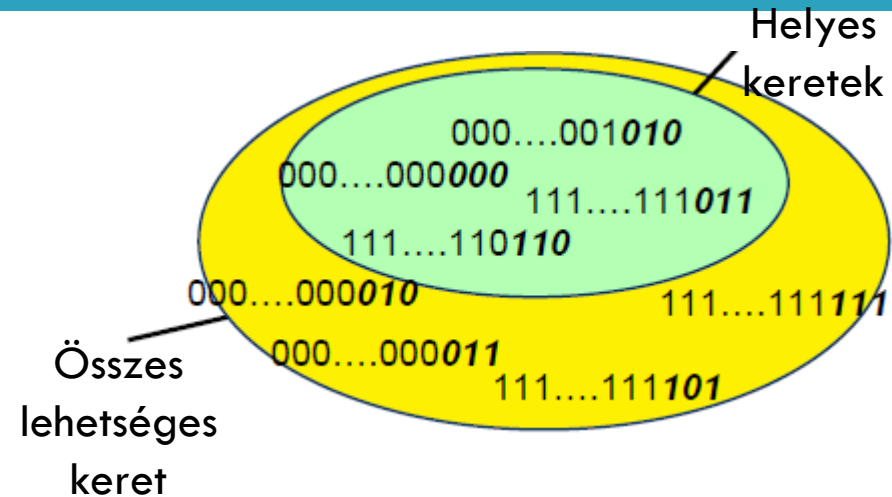
- ❑ Redundancia szükséges a hiba vezérléshez
- ❑ Redundancia nélkül
  - ▣  $2^m$  lehetséges üzenet írható le  $m$  biten
  - ▣ Mindegyik helyes (legal) üzenet és fontos adatot tartalmazhat
  - ▣ Ekkor minden hiba egy új helyes (legal) üzenetet eredményez
    - A hiba felismerése lehetetlen
- ❑ Hogyan ismerjük fel a hibát???



# Redundancia

## □ Egy keret felépítése:

- ▣ m adat bit (ez az üzenet)
- ▣ r redundáns/ellenőrző bit
  - Az üzenetből számolt, új információt nem hordoz
- ▣ A teljes keret hossza:  $n = m + r$



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

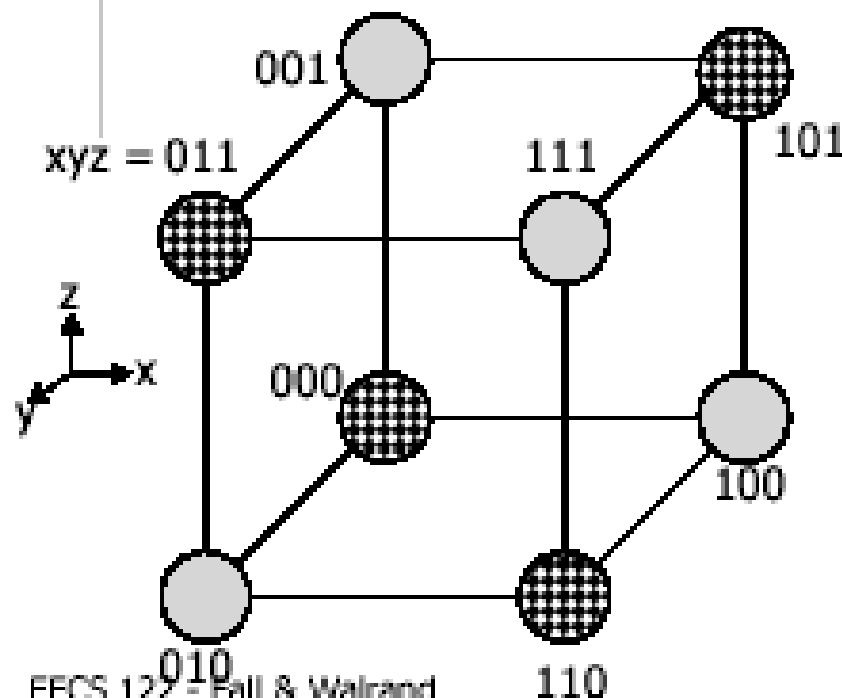
# Error Control Codes

## How Codes Work: Words and Codewords

◆ Code = subset of possible words: Codewords

◆ Example:

$n = 3$  bits  $\Rightarrow$  8 words; codewords: subset



Words:

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

Code:

000, 011, 101, 110

Send only codewords

# Elméleti alapok

- Tegyük fel, hogy a keret  $m$  bitet tartalmaz. (üzenet bitek)
- A redundáns bitek száma legyen  $r$ . (ellenőrző bitek)
- A küldendő keret tehát  $n=m+r$  bit hosszú. (kódszó)

## Hamming távolság

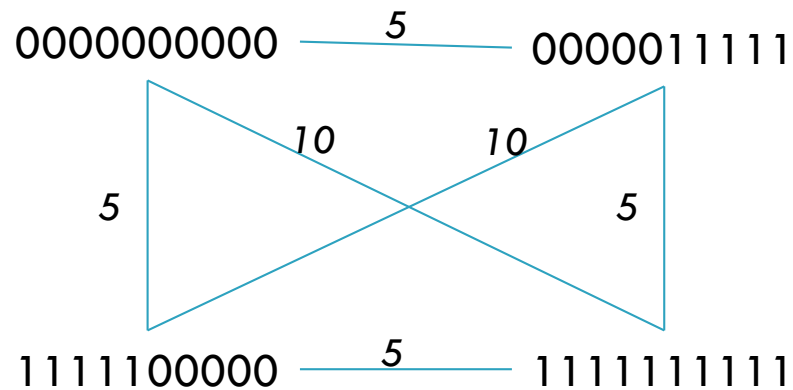
- Az olyan bitpozíciók számát, amelyeken a két kódszóban különböző bitek állnak, a két kódszó Hamming távolságának nevezzük.
  - ▣ Jelölés:  $d(x,y)$
- Legyen  $S$  egyenlő hosszú bitszavak halmaza, ekkor  $S$  Hamming távolsága az alábbi:

$$d(S) := \min_{x,y \in S \wedge x \neq y} d(x,y)$$

- ▣ Jelölés:  $d(S)$
- A Hamming távolság egy metrika.

# Példa Hamming távolságra

- Legyen  $S = \{0000000000, 0000011111, 1111100000, 1111111111\}$ .
- Mi lesz a halmaz Hamming távolsága?
  - ▣  $d(S) = 5$



# Hamming távolság használata

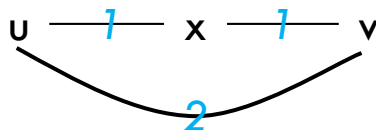
$S$  halmaz legyen a megengedett azonos hosszú kódszavak halmaza.

**$d(S)=1$  esetén**

- nincs hibafelismerés
- megengedett kódszóból megengedett kódszó állhat elő 1 bit megváltoztatásával

**$d(S)=2$  esetén**

- ha az  $u$  kódszóhoz létezik olyan  $x$  megengedett kódszó, amelyre  $d(u, x) = 1$ , akkor hiba történt.
- Feltéve, hogy az  $u$  és  $v$  megengedett kódszavak távolsága minimális, akkor a következő összefüggésnek teljesülnie kell:  
 $2 = d(u, v) \leq d(u, x) + d(x, v)$ .
- Azaz egy bithiba felismerhető, de nem javítható.



# Hamming korlát bináris kódkönyvre 1/3

## TÉTEL

Minden  $C \subseteq \{0,1\}^n$  kód, ahol  $d(C) = k$  ( $\in \mathbb{N}_+$ ). Akkor teljesül az alábbi összefüggés:

$$|C| \sum_{i=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{n}{i} \leq 2^n$$

## BIZONYÍTÁS

1. Hány olyan bitszó létezik, amely egy tetszőleges  $x \in C$  kódszótól pontosan  $i \in \mathbb{N}_+$  távolságra helyezkedik el?

■ Pontosan  $\binom{n}{i}$  lehetőség van.

2. Hány olyan bitszó létezik, amely egy tetszőleges  $x \in C$  kódszótól legfeljebb  $\lfloor \frac{k-1}{2} \rfloor$  távolságra helyezkedik el?

■ Pontosan  $\sum_{i=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{n}{i}$  lehetőség van.

# Hamming korlát bináris kódkönyvre 2/3

3. Lássuk be, hogy egy tetszőleges  $x \in \{0,1\}^n$  bitszóhoz legfeljebb egy legális  $u \in C$  kódszó létezhet, amelyre  $d(x, u) \leq \frac{k-1}{2}$  teljesül.
- ▣ Indirekt tegyük fel, hogy létezhet két legális kódszó is a  $C$  kódkönyvben, jelölje őket  $u_1$  és  $u_2$ . Ekkor viszont az alábbi két feltétel együttesen teljesül:

$$d(x, u_1) \leq \frac{k-1}{2} \text{ és } d(x, u_2) \leq \frac{k-1}{2}$$

- ▣ Mi a két kódszó távolsága?

$$d(u_2, u_1) \leq d(u_2, x) + d(x, u_1) \leq \frac{k-1}{2} + \frac{k-1}{2} = k-1$$

- ▣ Ez viszont ellentmond annak hogy a kódkönyv Hamming távolsága  $k$ , azaz az indirekt feltevésünk volt hibás. Vagyis tetszőleges bitszóhoz legfeljebb egy legális kódszó létezhet, amely a kódkönyv minimális távolságának felénél közelebb van a bitszóhoz.

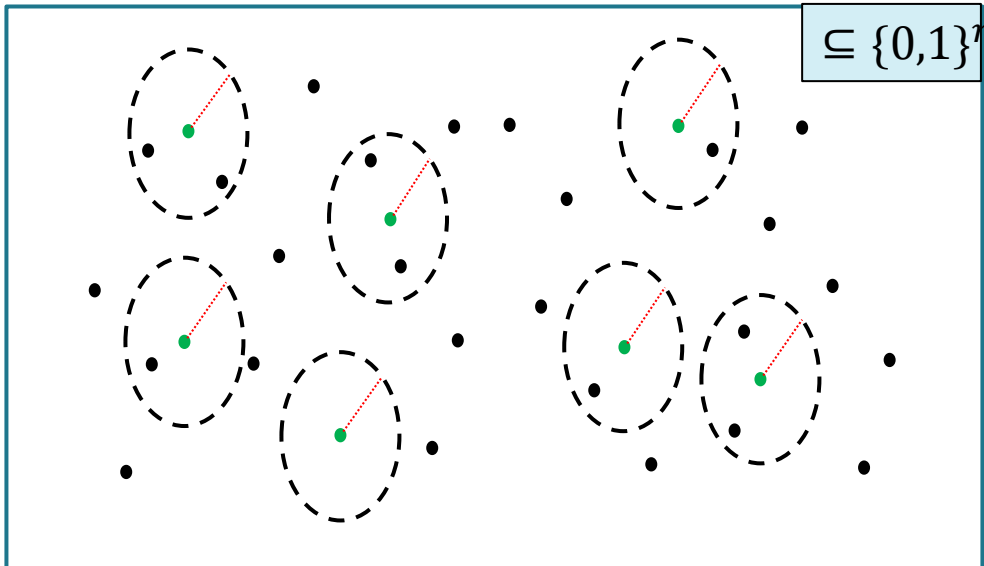


# Hamming korlát bináris kódkönyvre 3/3

4. A kódszavak  $\frac{k-1}{2}$  sugarú környezetében található bitszavak egymással diszjunkt halmazainak uniója legfeljebb az  $n$ -hosszú bitszavak halmazát adhatja ki. Vagyis formálisan:

$$|C| \sum_{i=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{n}{i} \leq 2^n$$

□



## JELMAGYARÁZAT

- Kódszó
- Bitszó, amely nem kódszó

# Hibafelismerés és javítás Hamming távolsággal

## Hibafelismerés

- $d$  bit hiba felismeréséhez a megengedett keretek halmazában legalább  $d+1$  Hamming távolság szükséges.

## Hibajavítás

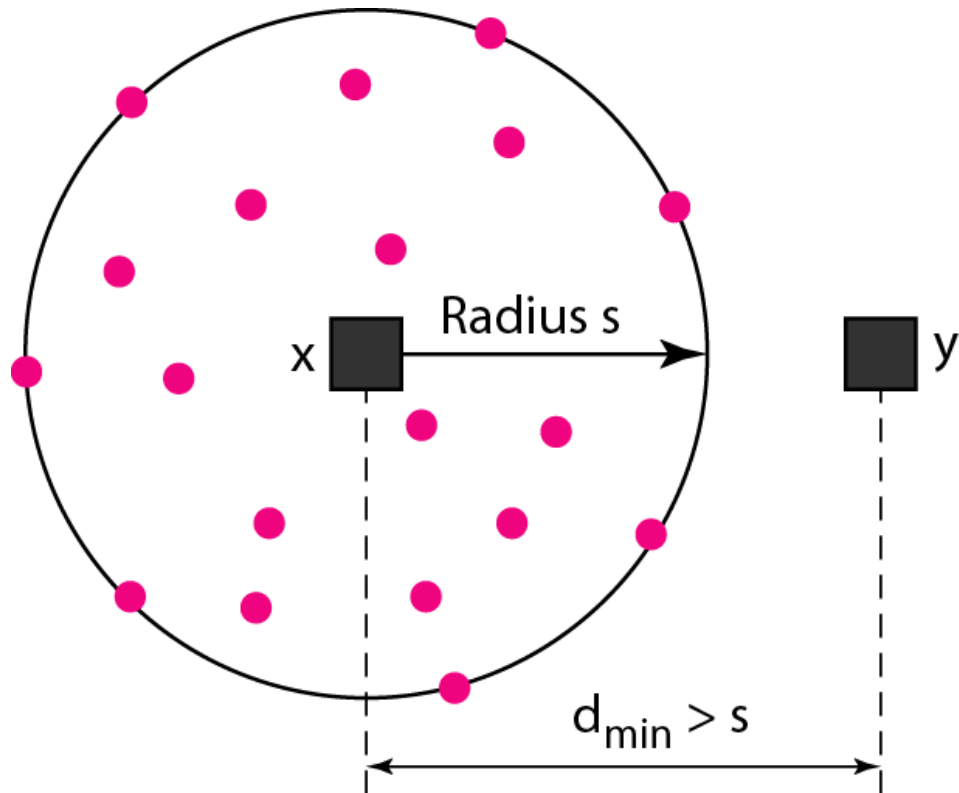
- $d$  bit hiba javításához a megengedett keretek halmazában legalább  $2d+1$  Hamming távolság szükséges

## Definíciók

- Egy  $S \subseteq \{0,1\}^n$  kód rátája  $R_S = \frac{\log_2 |S|}{n}$ . (a hatékonyságot karakterizálja)
- Egy  $S \subseteq \{0,1\}^n$  kód távolsága  $\delta_S = \frac{d(S)}{n}$ . (a hibakezelési lehetőségeket karakterizálja)
- A jó kódoknak a rátája és a távolsága is nagy.

# Hiba felismerés

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



## Legend



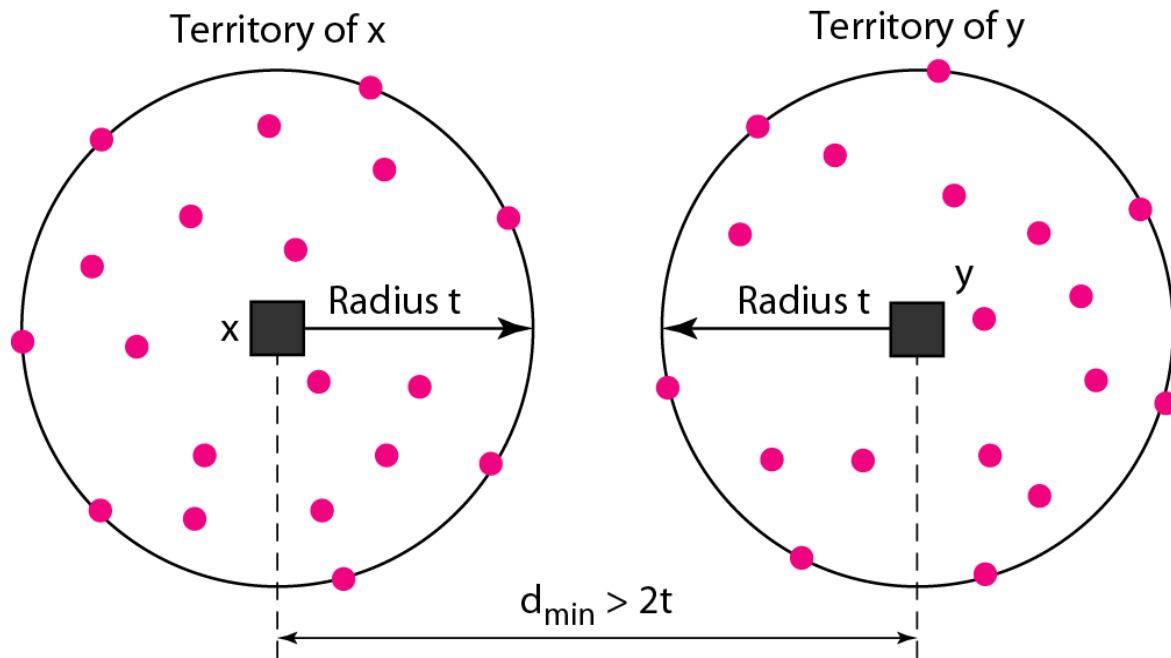
Any valid codeword



Any corrupted codeword  
with 0 to  $s$  errors

# Hiba javítás

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

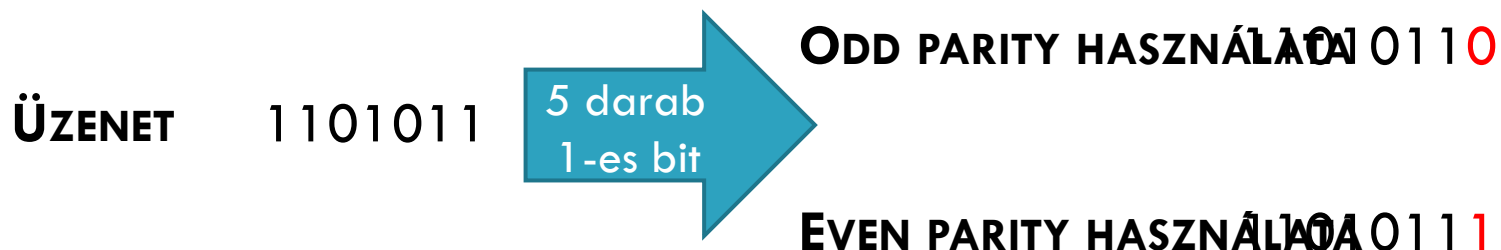


## Legend

- Any valid codeword
- Any corrupted codeword with 1 to  $t$  errors

# Újra a paritás bit használata 1/4

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



# Paritás bit használata 2/4

## Egy paritást használó módszer (*Hamming*)

- a kódszó bitjeit számozzuk meg 1-gyel kezdődően;
- 2 egészhatvány sorszámú pozíciói lesznek az ellenőrző bitek, azaz 1,2,4,8,16,...;
- a maradék helyeket az üzenet bitjeivel töltjük fel;
- mindegyik ellenőrző bit a bitek valamilyen csoportjának a paritását állítja be párosra (vagy páratlanra)
- egy bit számos paritásszámítási csoportba tartozhat:
  - $k$  pozíciót írjuk fel kettő hatványok összegeként, a felbontásban szereplő ellenőrző pozíciók ellenőrzik a  $k$ -adik pozíciót
  - Példa:  $k=13$ -ra  $k=1+4+8$ , azaz az első, a negyedik illetve a nyolcadik ellenőrző bit fogja ellenőrizni

# Paritás bit használata - példa 3/4

- Az *ASCII* kód 7 biten ábrázolja a karaktereket
- A példában *EVEN PARITY*-t használunk

## ÜZENET BITEK KÓDSZÓBAN LÉVŐ POZÍCIÓNAK FELBONTÁSAI

- $3 = 1 + 2$
- $5 = 1 + 4$
- $6 = 2 + 4$
- $7 = 1 + 2 + 4$
- $9 = 1 + 8$
- $10 = 2 + 8$
- $11 = 1 + 2 + 8$

| ASCII karakter | ASCII decimális | Üzenet forrás bitjei | Az előállt kódszavak |
|----------------|-----------------|----------------------|----------------------|
| E              | 69              | 1000101              | 10100000101          |
| L              | 76              | 1001100              | 10110011100          |
| T              | 84              | 1010100              | 00110101100          |
| E              | 69              | 1000101              | 10100000101          |
|                | 32              | 0100000              | 10001100000          |
| I              | 73              | 1001001              | 11110011001          |
| K              | 75              | 1001011              | 00110010011          |

# Paritás bit használata 4/4

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

FOGADOTT E KARAKTER 10100100101

Számláló != 0

SZÁMLÁLÓ = 2 + 4

FOGADOTT L KARAKTER 11110011100

Számláló != 0

SZÁMLÁLÓ = 2



# Hamming kódok csoportos hibákra

- a kódszavak  $k$  hosszú sorozatát mátrixba rendezzük, minden sorba egy kódszót
  - ▣ az adatot kódszavanként továbbítjuk (bitek szerint balról jobbra)
- csoportos hiba javításához oszloponként továbbítjuk az adatot (oszlopok szerint balról jobbra)
  - ▣ a vevő újra felépíti a mátrixot az oszlopokból
  - ▣  $k$  hosszú csoportoshibát javítani tudunk, mivel maximum egy érintett bit van soronként
  - ▣  $kr$  ellenőrző bit  $km$  méretű adatahoz; maximum  $k$  hosszú csoportos hiba védhető ki

# Hibajelző kódok

# Hibajelző kódok

## Polinom-kód, avagy ciklikus redundancia (CRC kód)

- Tekintsük a bitsorozatokat  $\mathbb{Z}_2$  feletti polinomok reprezentációinak.

## Polinom ábrázolása $\mathbb{Z}_2$ felett

$$p(x) = \sum_{i=0}^n a_i x^i = a_n x^n + \dots + a_1 x^1 + a_0 x^0, \text{ ahol } a_i \in \{0,1\}$$

- A számítás *mod 2* történik. (összeadás, kivonás, szorzás, osztás)
- reprezentálható az együtthatók  $n+1$ -es vektorával, azaz  $(a_n, \dots, a_1, a_0)$
- Például az ASCII „b” karakter kódja 01100010, aminek megfelelő polinom hatod fokú polinom
$$p(x) = 1 * x^6 + 1 * x^5 + 0 * x^4 + 0 * x^3 + 0 * x^2 + 1 * x^1 + 0 * x^0$$
- Az összeadás és a kivonás gyakorlati szempontból a logikai KIZÁRÓ VAGY művelettel azonosak.

$$\begin{array}{r} 11110000 \\ - 10100110 \\ \hline 01010110 \end{array}$$

$$\begin{array}{r} 10011011 \\ + 11001010 \\ \hline 01010001 \end{array}$$

# CRC

Definiáljuk a  $G(x)$  generátor polinomot ( $G$  foka  $r$ ), amelyet a küldő és a vevő egyaránt ismer.

## Algoritmus

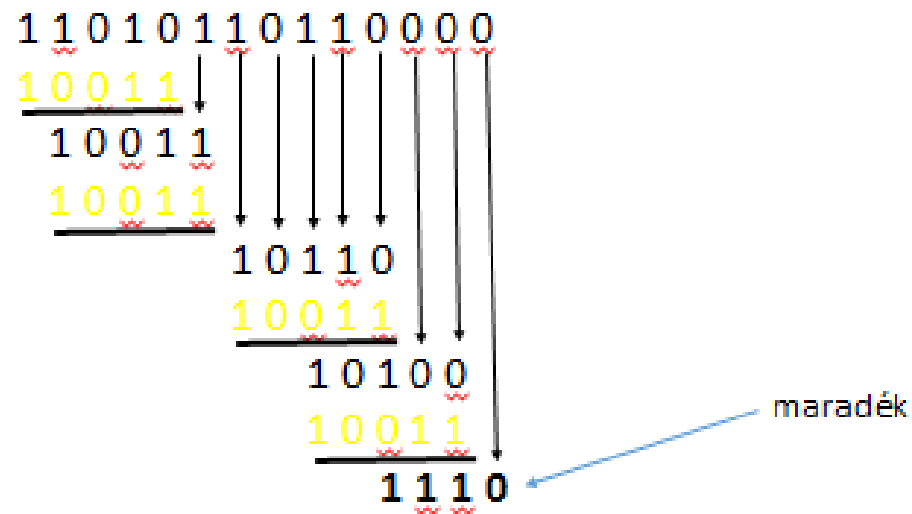
1. Legyen  $G(x)$  foka  $r$ . Fűzzünk  $r$  darab 0 bitet a keret alacsony helyi értékű végéhez, így az  $m+r$  bitet fog tartalmazni és az  $x^r M(x)$  polinomot fogja reprezentálni.
2. Osszuk el az  $x^r M(x)$  tartozó bitsorozatot a  $G(x)$ -hez tartozó bitsorozattal modulo 2.
3. Vonjuk ki a maradékot (mely mindig  $r$  vagy kevesebb bitet tartalmaz) az  $x^r M(x)$ -hez tartozó bitsorozatból moduló 2-es kivonással. Az eredmény az ellenőrző összeggel ellátott, továbbítandó keret. Jelölje a továbbítandó keretnek megfelelő a polinomot  $T(x)$ .
4. A vevő a  $T(x) + E(x)$  polinomnak megfelelő sorozatot kapja, ahol  $E(x)$  a hiba polinom. Ezt elosztja  $G(x)$  generátor polinommal.
  - Ha az osztási maradék, amit  $R(x)$  jelöl, nem nulla, akkor hiba történt.

# Példa CRC számításra

**Keret:** 1101011011

**Generátor:** 10011

**A továbbítandó üzenet:** 11010110111110



# CRC áttekintés

- A  $G(x)$  többszöröseinek megfelelő bithibákat nem ismerjük fel, azaz, ha  $\exists j \in \mathbb{N}: E(x) = x^j G(x)$ .
- $G(x)$  legmagasabb illetve legalacsonyabb fokú tagjának együtthatója mindig 1.

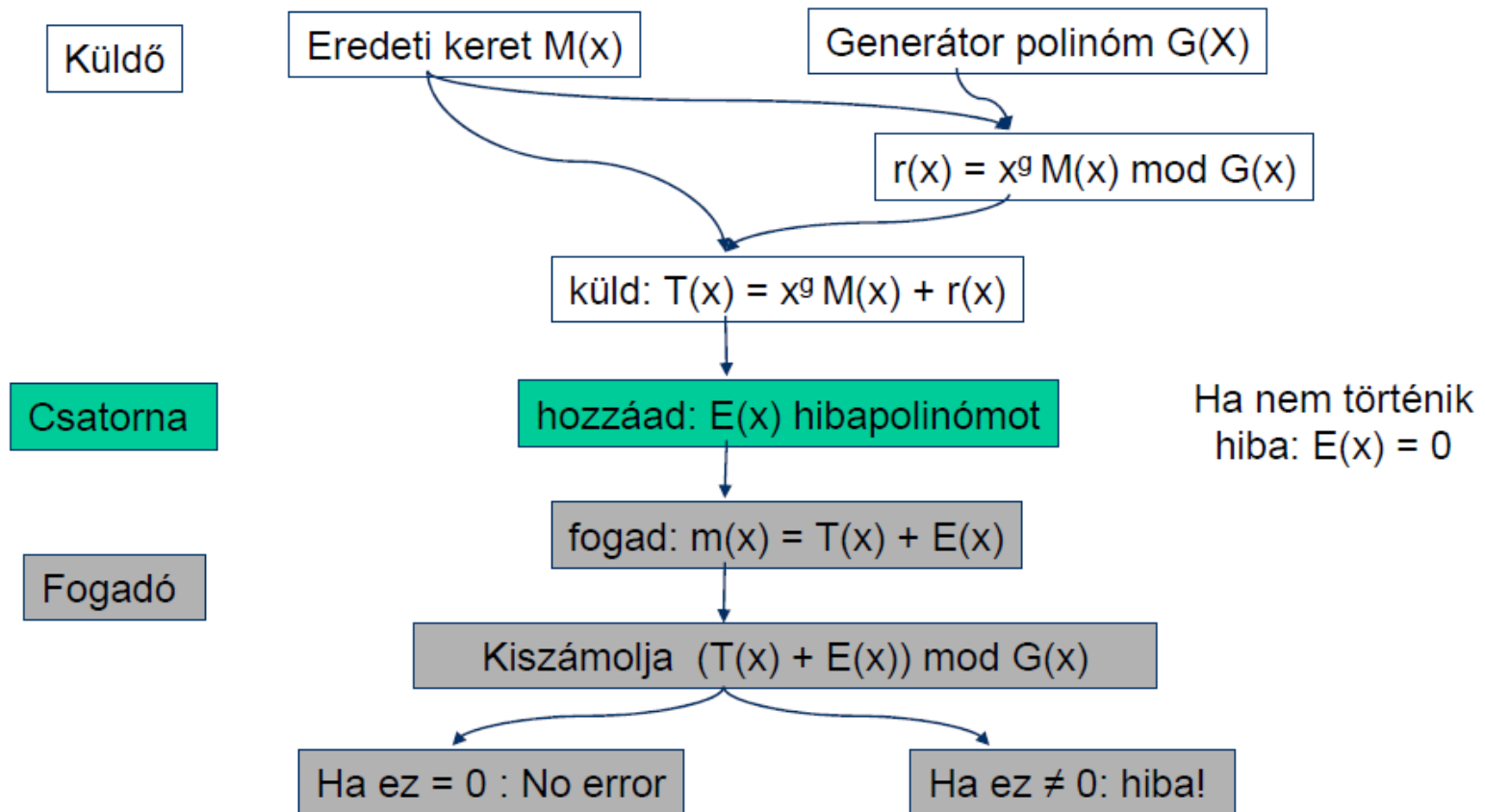
## Hiba események

- $E(x) = x^i$ , azaz  $i$  a hibás bit sorszáma, mivel  $G(x)$  kettő vagy több tagból áll, ezért minden egybites hibát jelezni tud.
- $E(x) = x^i + x^j = x^j (x^{i-j} + 1)$  ( $i > j$ ), azaz két izolált egybites hiba esetén.
  - ▣  $G(x)$  ne legyen osztható  $x$ -szel;
  - ▣  $G(x)$  ne legyen osztható  $(x^k + 1)$  –gyel semmilyen maximális kerethossznál kisebb  $k$ -ra. (Pl.  $x^{15} + x^{14} + 1$ )
- Ha  $E(x)$  páratlan számú tagot tartalmaz, akkor nem lehet  $x+1$  többszöröse. Azaz, ha  $G(x)$  az  $x+1$  többszöröse, akkor minden páratlan számú hiba felismerhető
- Egy  $r$  ellenőrző bittel ellátott polinom-kód minden legfeljebb  $r$  hosszúságú csoportos hibát jelezni tud

# CRC áttekintés

55

□ Forrás: Dr. Lukovszki Tamás fóliái



# CRC a gyakorlatban

- IEEE 802 által használt polinom az

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

- Néhány jó tulajdonságai a fenti polinomnak:

1. minden legfeljebb 32 bites hibacsomót képes jelezni,
2. minden páratlan számú bitet érintő hibacsomót tud jelezni.

## **Peterson és Brown (1961)**

- Szerkeszthető egy egyszerű, léptető regiszteres áramkör az ellenőrző összeg hardverben történő kiszámítására és ellenőrzésére.



# Forgalomszabályozás

# Forgalomszabályozás

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

## Megoldási lehetőségek

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

# Elemi adatkapcsolati protokollok

## Feltevések

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

## Kommunikációs fajták

- *szimplex kommunikáció* – a kommunikáció pusztán egy irányba lehetséges
- *fél-duplex kommunikáció* – mindkét irányba folyhat kommunikáció, de egyszerre csak egy irány lehet aktív.
- *duplex kommunikáció* – mindkét irányba folyhat kommunikáció szimultán módon

# Korlátozás nélküli szimplex protokoll

a legegyszerűbb protokoll („utópia”)

## A környezet

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

## A protokoll

- résztvevők: *küldő* és *vevő*;
- nincs sem sorszámozás, sem nyugta;
- küldő végtelen ciklusban küldi kifelé a kereteket folyamatosan;
- a vevő kezdetben várakozik az első keret megérkezésére, keret érkezésekor a hardver puffer tartalmát változóba teszi és az adatrészt továbbküldi a hálózati rétegnek

# Szimplex megáll-és-vár protokoll (stop-and-wait protocol)

## A környezet

- mind az adó, mind a vevő hálózati rétegei mindig készen állnak;
- A vevőnek  $\Delta t$  időre van szüksége a bejövő keret feldolgozására (nincs puffereles és sorban állás sem);
- Az adatkapcsolati rétegek közötti kommunikációs csatorna sosem rontja vagy veszíti el a kereteket;

## A protokoll

- résztvevők: *küldő* és *vevő*;
- küldő egyesével küldi kereteket és addig nem küld újat, még nem kap nyugtát a vevőtől;
- a vevő kezdetben várakozik az első keret megérkezésére, keret érkezésekor a hardver puffer tartalmát változóba teszi és az adatrészt továbbküldi a hálózati rétegnek, végül nyugtázza a keretet

**Következmény:** fél-duplex csatorna kell.

# Szimplex protokoll zajos csatornához 1 / 2

## A környezet

- mind az adó, mind a vevő hálózati rétegei mindig készen állnak;
- A vevőnek  $\Delta t$  időre van szüksége a bejövő keret feldolgozására (nincs pufferek és sorban állás sem);
- Az adatkapcsolati rétegek közötti kommunikációs csatorna hibázhat (keret megsérülése vagy elvesztése);

## A protokoll

- résztvevők: *küldő* és *vevő*;
- küldő egyesével küldi kereteket és addig nem küld újat, még nem kap nyugtát a vevőtől egy megadott határidőn belül, ha a határidő lejár, akkor ismételt elküldi az aktuális keretet;
- a vevő kezdetben várakozik az első keret megérkezésére, keret érkezésekor a hardver puffer tartalmát változóba teszi, leellenőrzi a kontroll összeget,
  - ha nincs hiba, az adatrészt továbbküldi a hálózati rétegnek, végül nyugtázza a keretet;
  - Ha hiba van, akkor eldobja a keretet és nem nyugtáz.

**Következmény:** duplikátumok lehetnek.

# Szimplex protokoll zajos csatornához 2/2

- **Megoldás:** sorszámok használata
- Mennyi sorszámra lesz szükség?  $\{0,1\}$  elegendő

## A protokoll (ARQ) – Alternáló-bit protokoll

- résztvevők: *küldő* és *vevő*;
- küldő egyesével küldi a sorszámmal ellátott kereteket (kezdetben 0-s sorszámmal) és addig nem küld újat, még nem kap nyugtát a vevőtől egy megadott határidőn belül:
  - ha a nyugta megérkezik a határidőn belül, akkor lépteti a sorszámot *mod 2* és küldi a következő sorszámmal ellátott keretet;
  - ha a határidő lejár, akkor ismételten elküldi az aktuális sorsszámmal ellátott keretet;
- a vevő kezdetben várakozik az első keret megérkezésére 0-s sorszámmal, keret érkezésekor a hardver puffer tartalmát változóba teszi, leellenőrzi a kontroll összeget és a sorszámot
  - ha nincs hiba, az adatrészt továbbküldi a hálózati rétegnek, végül nyugtázza a keretet és lépteti a sorszámát *mod 2*;
  - ha hiba van, akkor eldobja a keretet és nem nyugtáz.

# Csúszó-ablak protokollok 1 / 2

## ALAPOK (ÁLTALÁNOS)

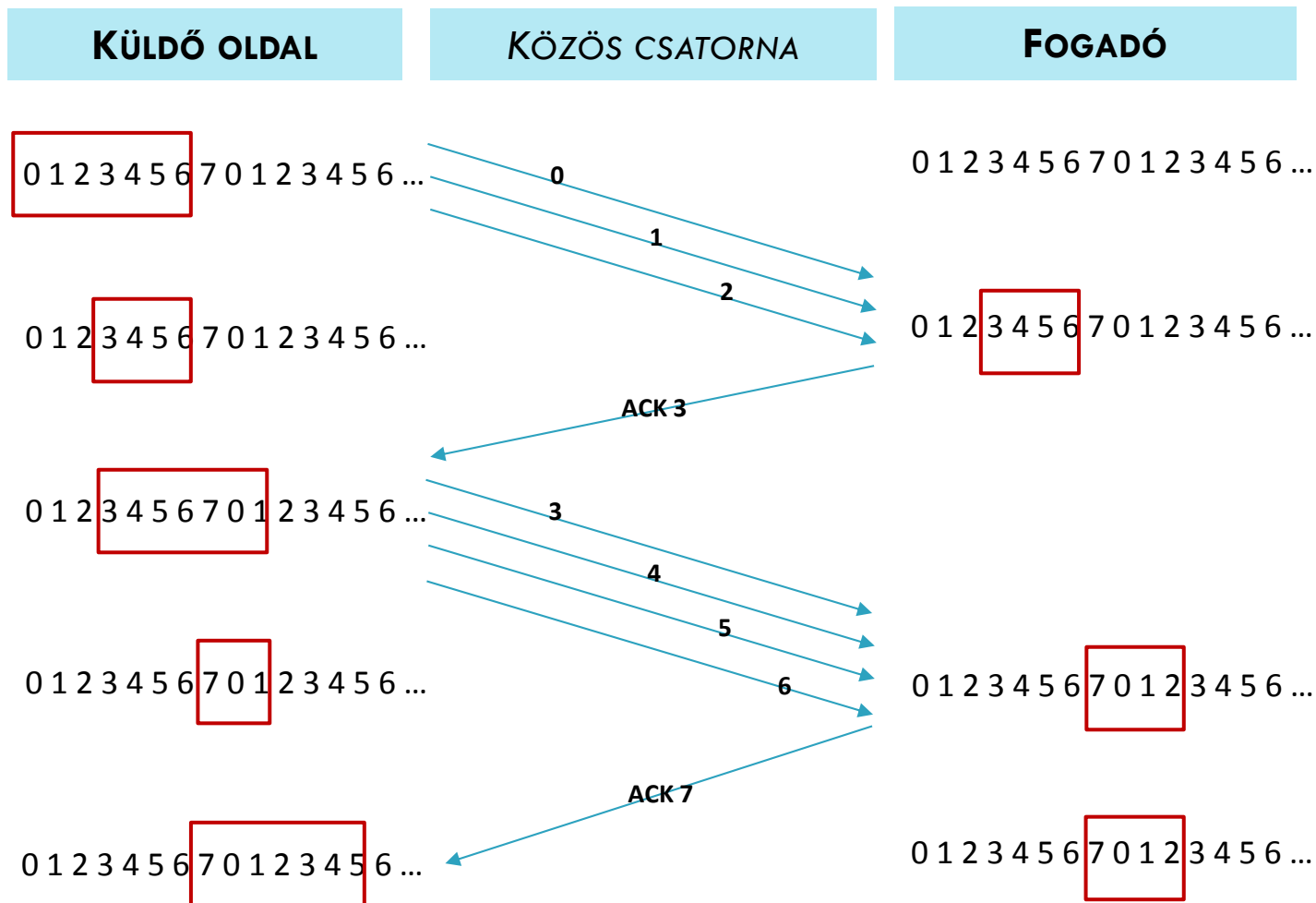
- Egy adott időpontban egyszerre több keret is átviteli állapotban lehet.
- A fogadó  $n$  keretnek megfelelő méretű puffert allokál.
- A küldőnek legfeljebb  $n$ , azaz ablak méretnyi, nyugtázatlan keretet küldése engedélyezett.
- A keret sorozatbeli pozíciója adja a keret címkéjét. (sorozatszám)

## ALAPOK (FOGADÓ)

- A keret nyugtázója tartalmazza a következőnek várt keret sorozatszámát.
  - ▣ *kumulatív nyugta* – Olyan nyugta, amely több keretet nyugtáz egyszerre. Például, ha a 2,3 és 4 kereteket is fogadnánk, akkor a nyugtát 5 sorszám tartalommal küldenénk, amely nyugtázza mind a három keretet.
- A hibás kereteket el kell dobni.
- A nem megengedett sorozatszámmal érkező kereteket el kell dobni.



# Példa 3-bites csúszó-ablak protokollra



# Csúszó-ablak protokollok 2/2

## JELLEMZŐK (ÁLTALÁNOS)

- A küldő nyilvántartja a küldhető sorozatszámok halmazát. (*adási ablak*)
- A fogadó nyilvántartja a fogadható sorozatszámok halmazát. (*vételi ablak*)
- A sorozatszámok halmaza minden esetben véges.
  - ▣  $K$  bites mező esetén:  $[0..2^K - 1]$ .
- A adási ablak minden küldéssel szűkül, illetve nő egy nyugta érkezésével.

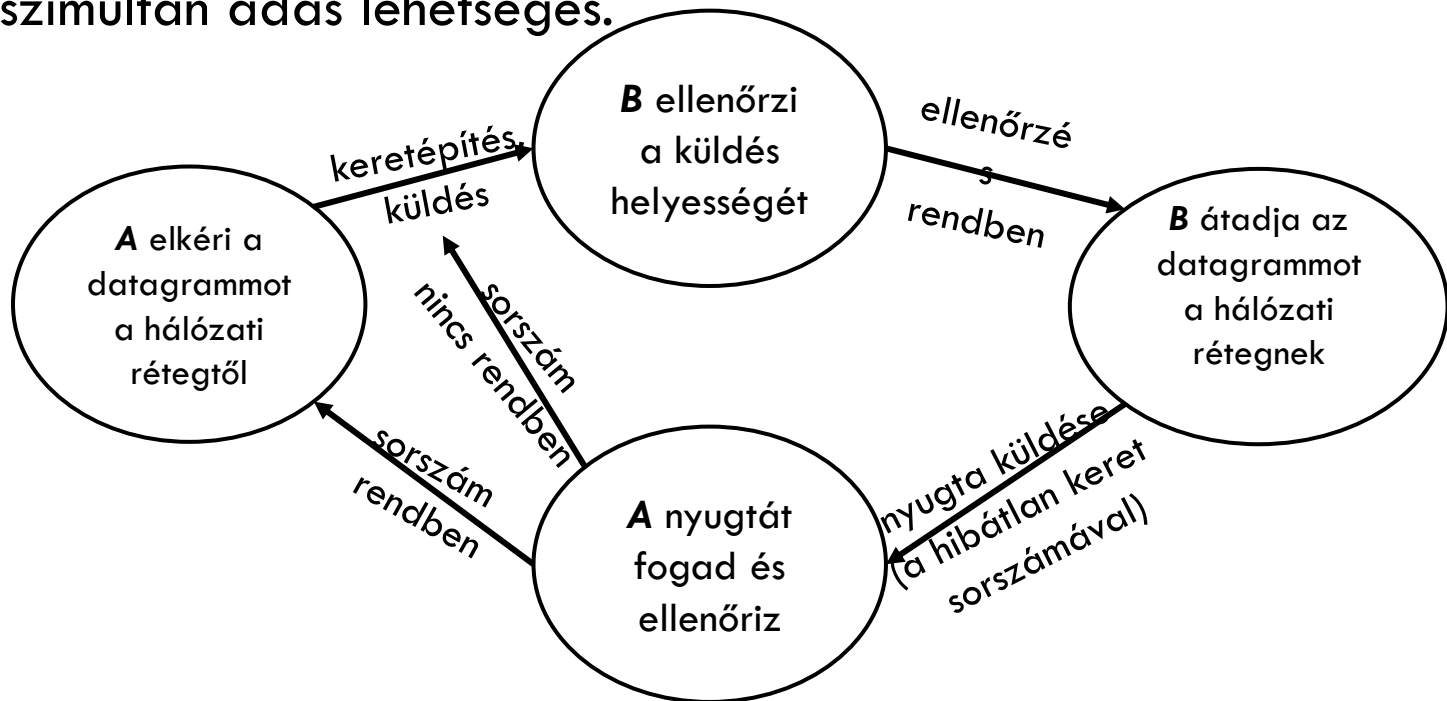
## JELLEMZŐK (GYAKORLATI ALKALMAZÁS ESETÉN)

- gyakorlatban kétirányú adatfolyamot kell kezelni (*duplex csatorna*)
  - ▣ két különböző szimplex csatorna használata (*két áramkör használata*)
  - ▣ egy csatorna használata (*egy áramkör használata*)
    - **piggybacking módszer**– a kimenő nyugtákat késleltetjük, hogy rá tudjuk akasztani a következő kimenő adatkeretre (*ack mező használata*);

# Egybites csúszó-ablak protokoll állapotátmenetei

## KÖRNYEZET

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



# Pipelining

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

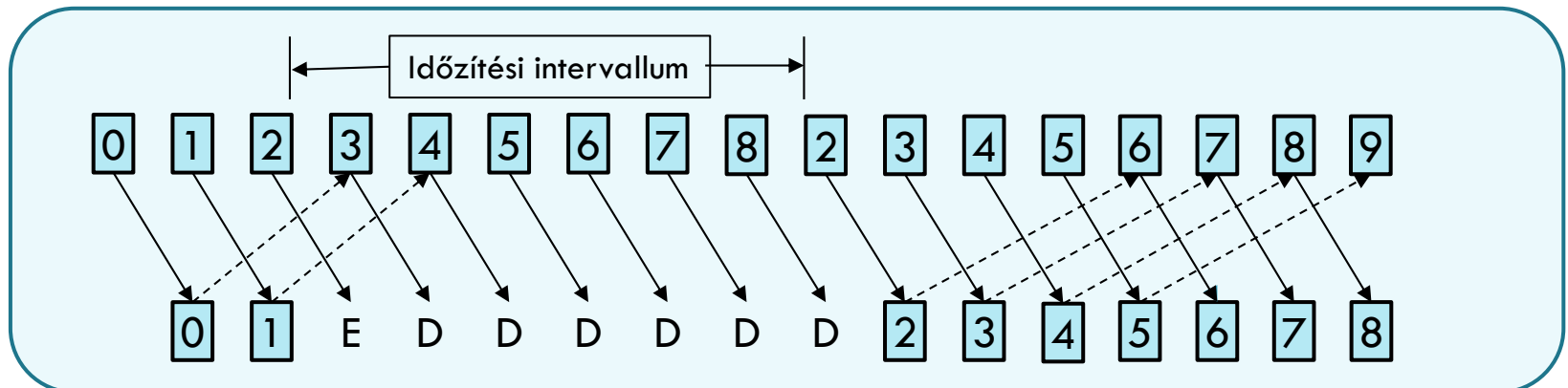
# „visszalépés N-nel” stratégia

## Stratégia lényege

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

## Következmények

- Egy méretű vételi ablakot feltételezünk.
- Nagy sávszélességet pazarolhat el, ha nagy a hibaarány.



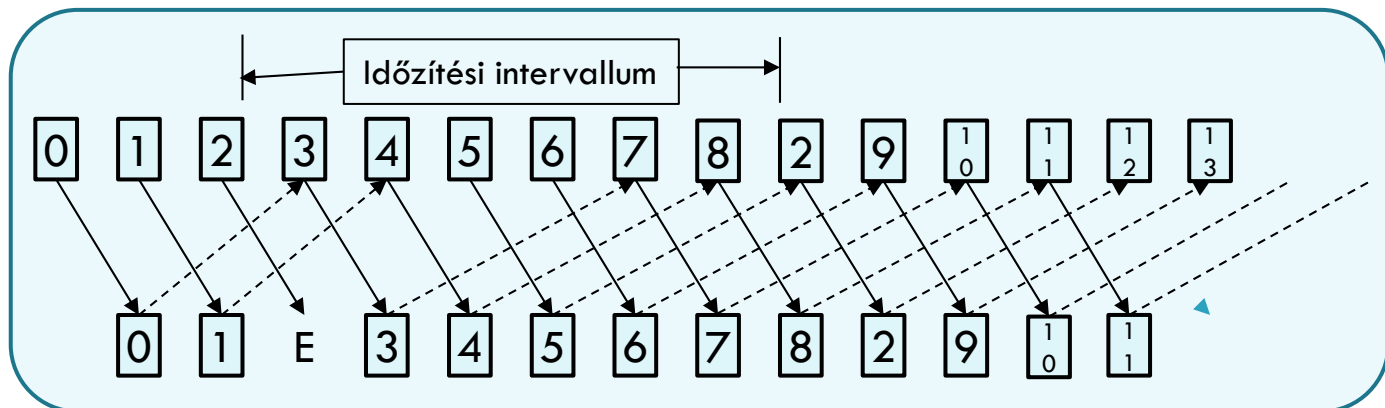
# „szelektív ismétlés” stratégia

## Stratégia lényege

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

## Következmények

- Javíthat a hatékonyságon a negatív nyugta használata. (NAK)
- Egynél nagyobb méretű vételi ablakot feltételezünk.
- Nagy memória igény, ha nagy vételi ablak esetén.



Köszönöm a figyelmet!