

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

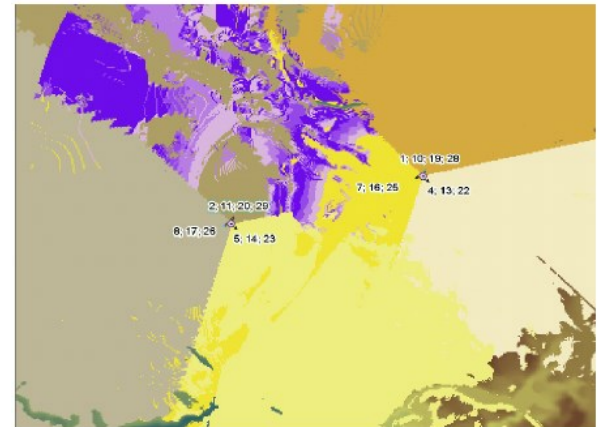
3

- Lehetővé teszi, hogy több jel egy időben 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

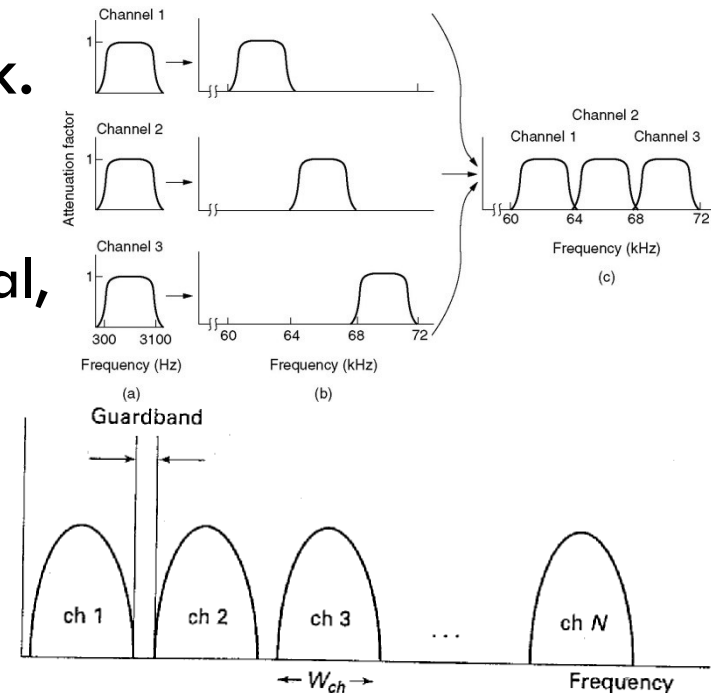
- Ez a legegyszerűbb multiplexálási módszer.
- Angolul **Space-Division Multiplexing**
- Vezetékes kommunikáció esetén minden egyes csatornához külön pont-pont vezeték tartozik.
- Vezeték nélküli kommunikáció esetén minden egyes csatornához külön antenna rendelődik.



Frekvencia multiplexálás

5

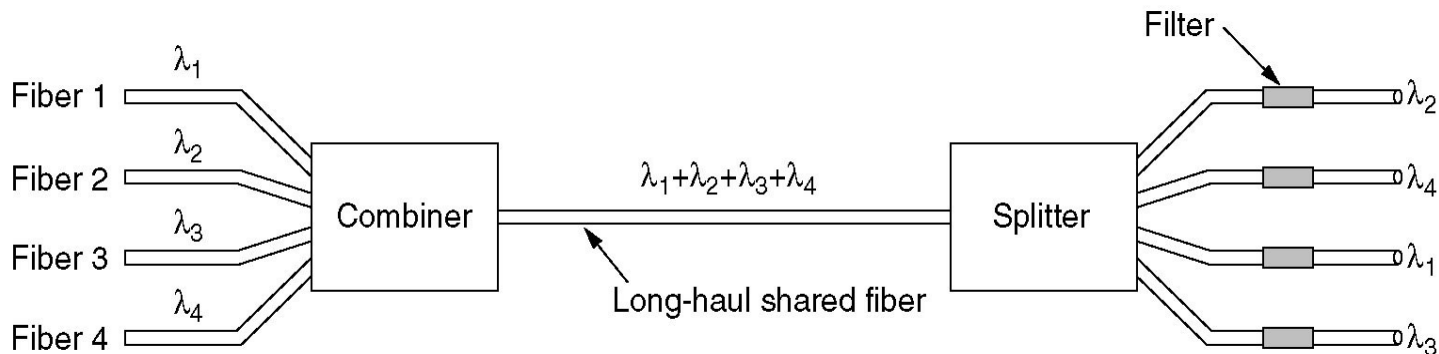
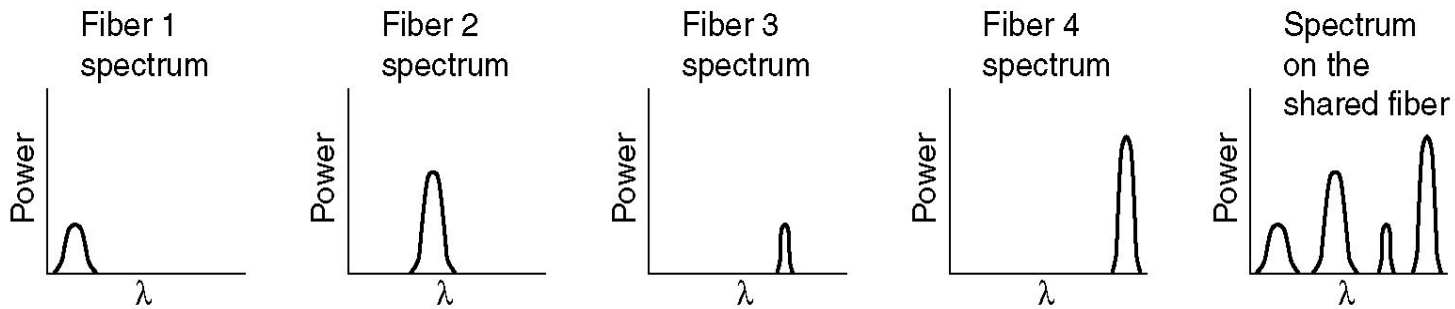
- Olyan módszertan, amelyben egy kommunikációs csatornán több szignál kombinációja adja az átvitelt.
- Minden szignálhoz más frekvencia tartozik.
- Angolul **Frequency-Division Multiplexing**
- Tipikusan analóg vonalon használják.
- Többféle megvalósítása van:
 - ▣ XOR a szignálokon véletlen bitsorozattal,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



Hullámhossz multiplexálás

6

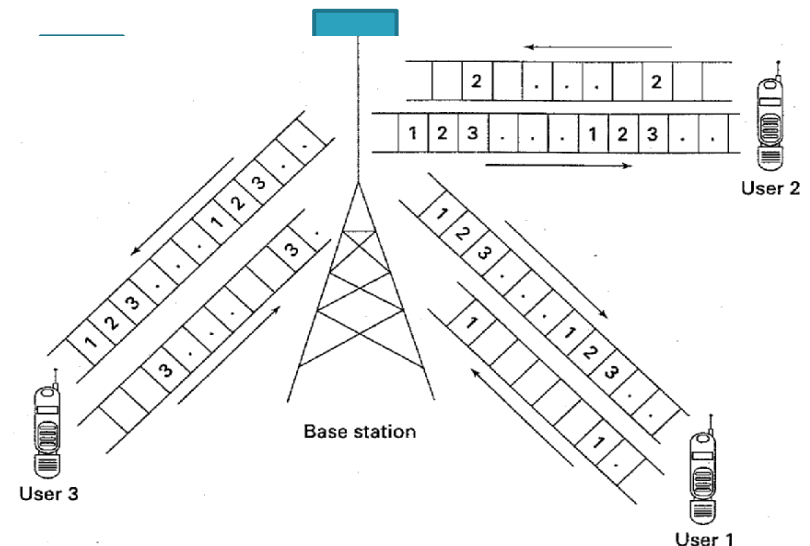
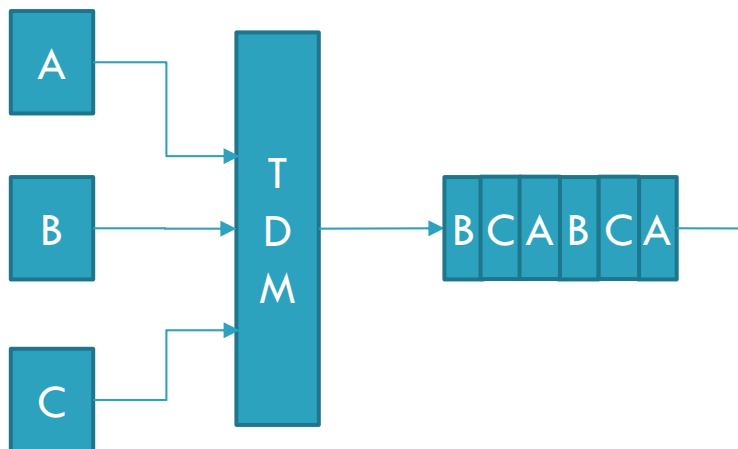
- Optikai kábeleknel alkalmazzák.
- Angolul **Wavelength-Division Multiplexing**



Időbeli multiplexálás

7

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



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 sáv szélesség 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

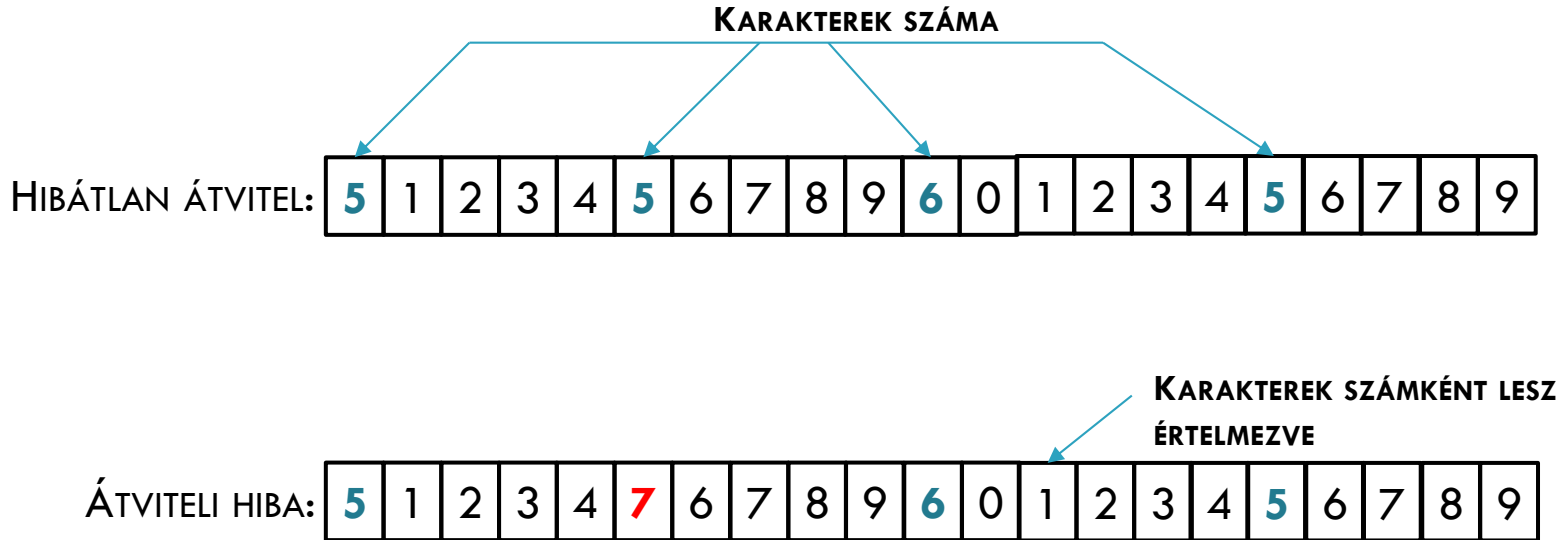
Keret képzés/Keretezés/Framing

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

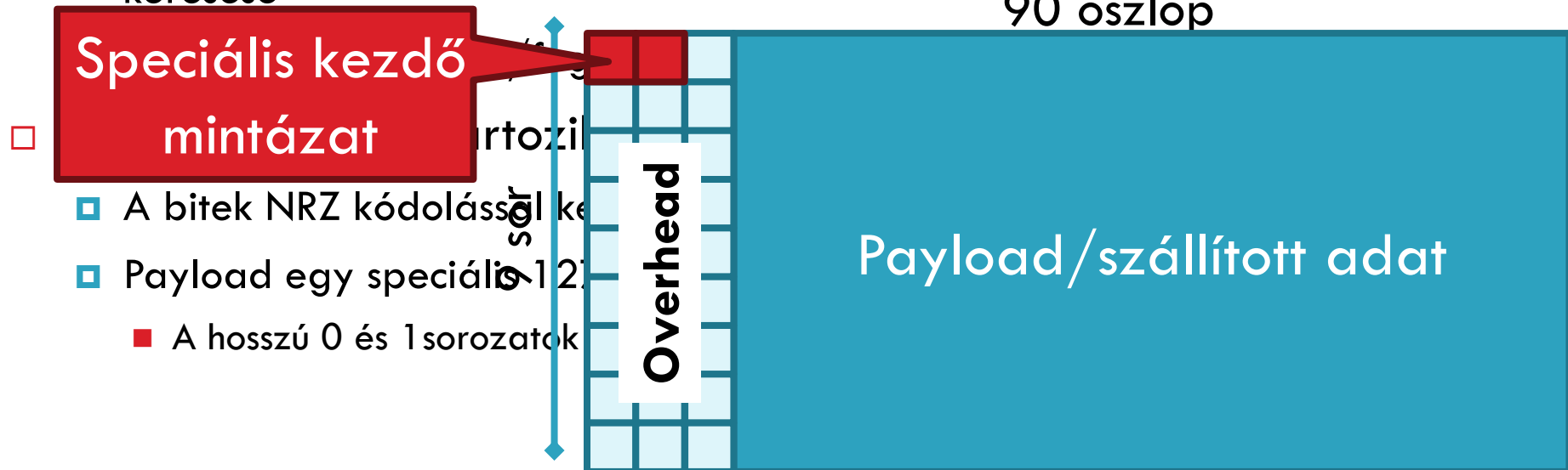
Óra alapú keretezés: SONET

□ 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



Hiba felügyelet

Zaj kezelése

27

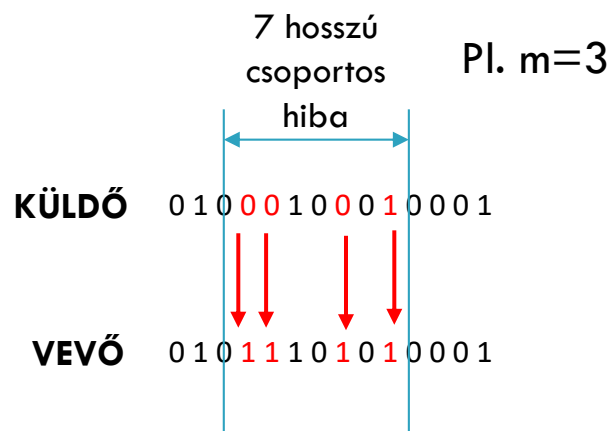
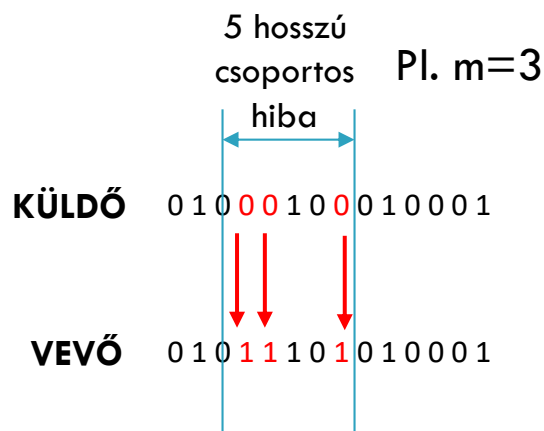
- A fizikai világ eredendően zajos
 - ▣ Interferencia az elektromos kábelek között
 - ▣ Áthallás a rádiós átvitelek között, mikrosütő, ...
 - ▣ Napviharok
- Hogyan detektáljuk a bithibákat az átvitelben?
- Hogyan állítsuk helyre a hibát?

Bithibák definíciók és példák

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

| | | | | | | | | |
|-------|---|---|---|---|---|---|---|---|
| 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

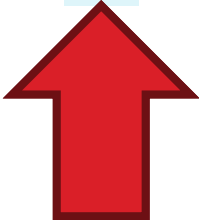
Paritás Bit

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

Hiba vezérlés

□ Stratégiák

▣ Hiba javító kódok

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

▣ Hiba detektálás és újraküldés

- Automatic Repeat Request (ARQ)
- megbízható csatornákon olcsóbb

Hiba vezérlés

□ Célok

▣ Hiba detektálás

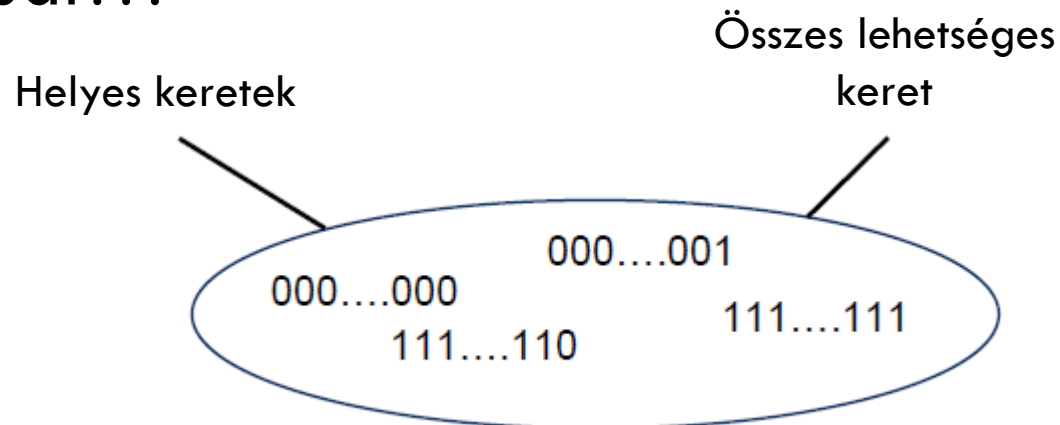
- javítással
 - ▣ Forward error correction
- Javítás nélkül -> pl. eldobjuk a keretet
 - ▣ Utólagos hibajavítás
 - ▣ A hibás keret újraküldése

▣ Hiba javítás

- Hiba detektálás nélkül
 - ▣ Pl. hangátvitel

Redundancia

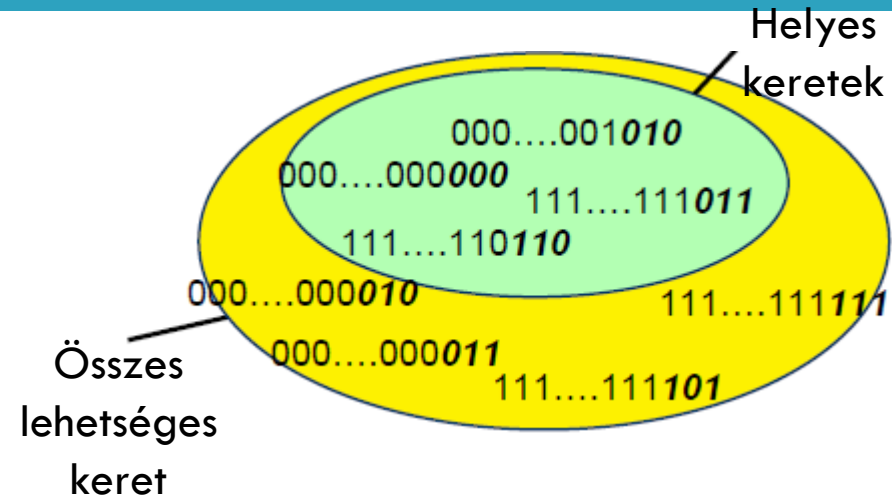
- ❑ Redundancia szükséges a hiba vezérléshez
- ❑ Redundancia nélkül
 - ▣ 2^m lehetséges üzenet írható le m biten
 - ▣ Mindegyik helyes (legal) üzenet és fontos adatot tartalmazhat
 - ▣ Ekkor minden hiba egy új helyes (legal) üzenetet eredményez
 - A hiba felismerése lehetetlen
- ❑ Hogyan ismerjük fel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

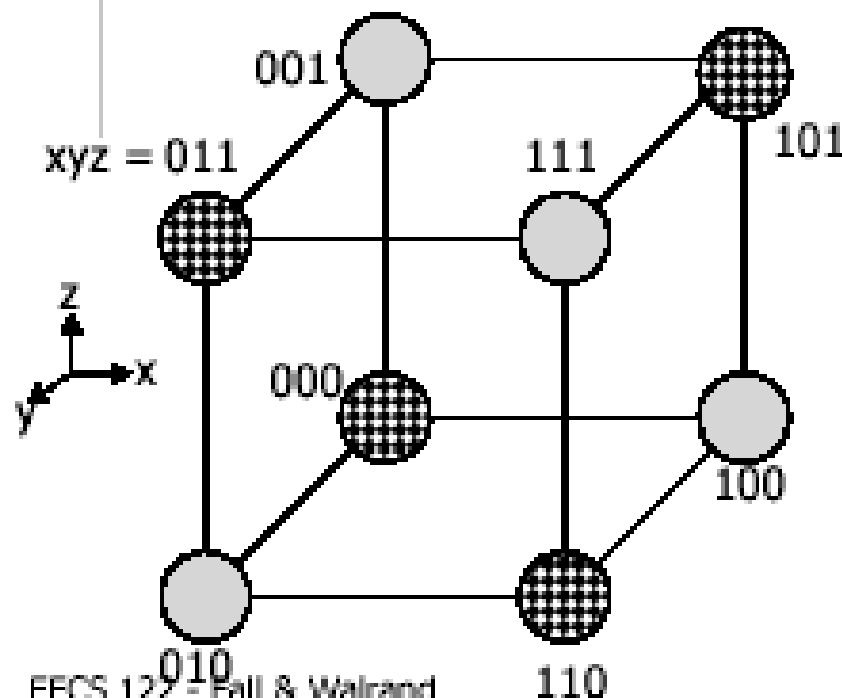
Error Control Codes

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

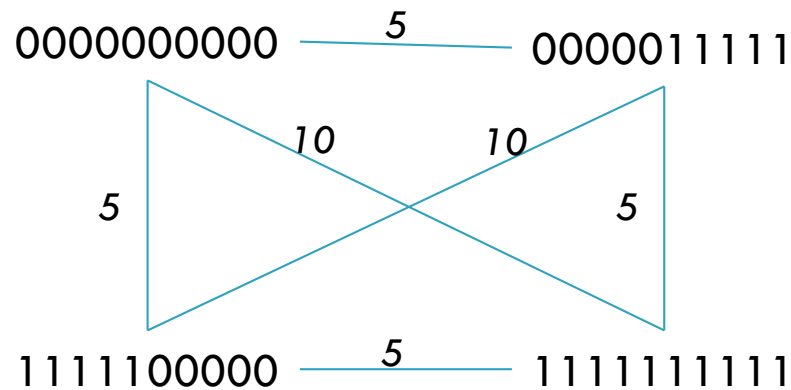
- Az olyan bitpozíciók számát, amelyeken a két kódszóban különböző bitek állnak, a két kódszó Hamming távolságának nevezzük.
 - ▣ Jelölés: $d(x,y)$
- Legyen S egyenlő hosszú bitszavak halmaza, ekkor S Hamming távolsága az alábbi:

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

- ▣ 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

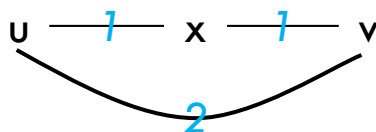
S halmaz legyen a megengedett azonos hosszú kódszavak halmaza.

$d(S)=1$ 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?

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

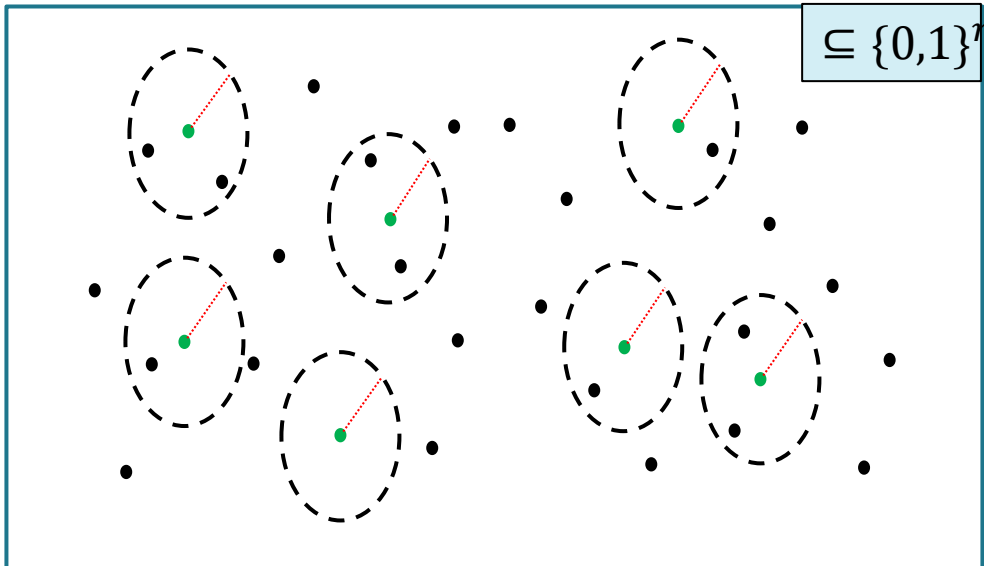
- ▣ Ez viszont ellentmond annak hogy a kódkönyv Hamming távolsága k , azaz az indirekt feltevésünk volt hibás. Vagyis tetszőleges bitszóhoz legfeljebb egy legális kódszó létezhet, amely a kódkönyv minimális távolságának felénél közelebb van a bitszóhoz.

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

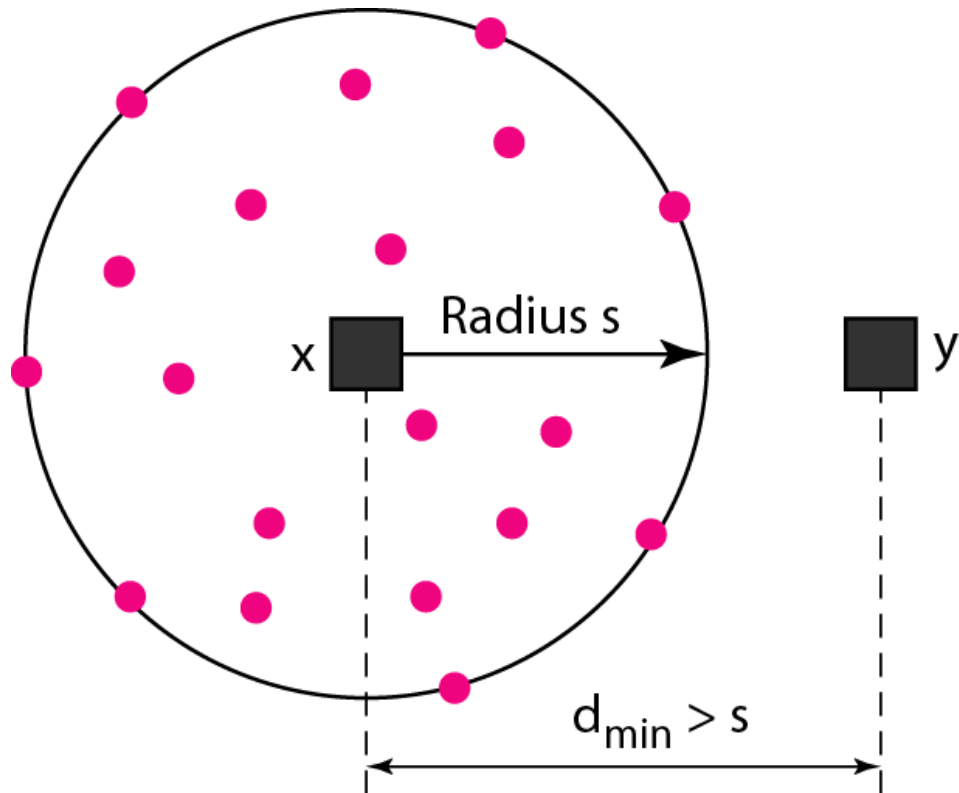
- d bit hiba javításához a megengedett keretek halmazában legalább $2d+1$ Hamming távolság szükséges

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



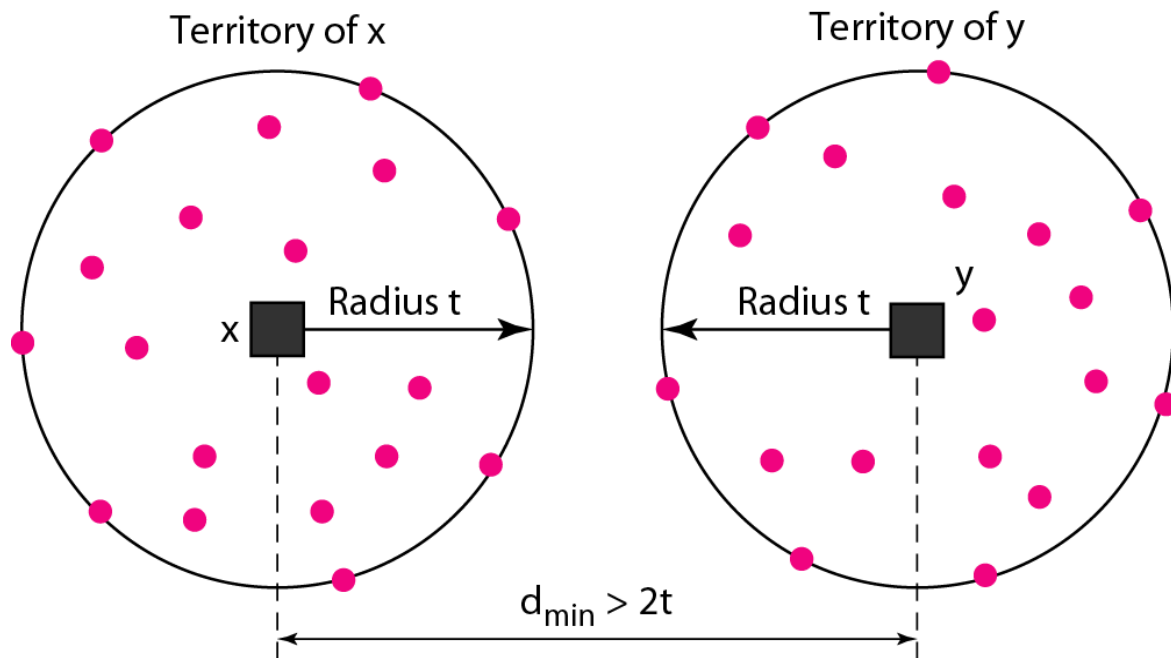
Any valid codeword



Any corrupted codeword
with 0 to s errors

Hiba javítás

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

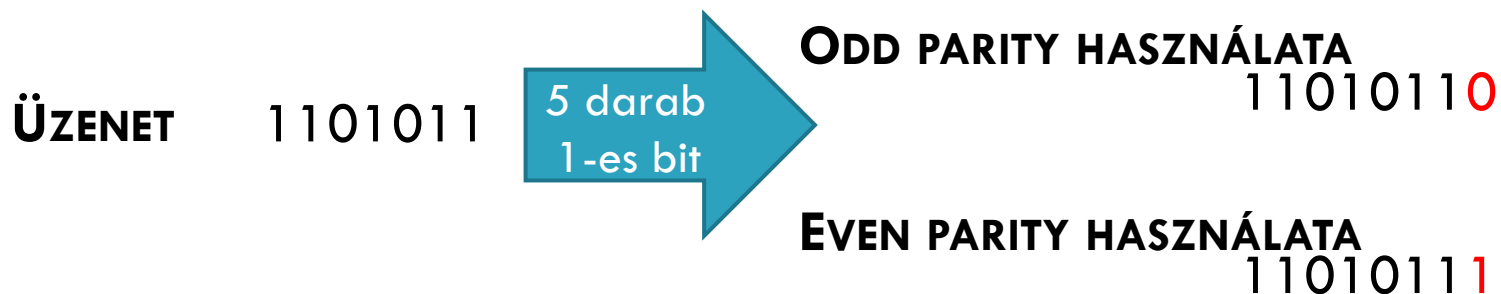


Legend

- 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

Köszönöm a figyelmet!