Számítógépes Hálózatok

5. gyakorlat

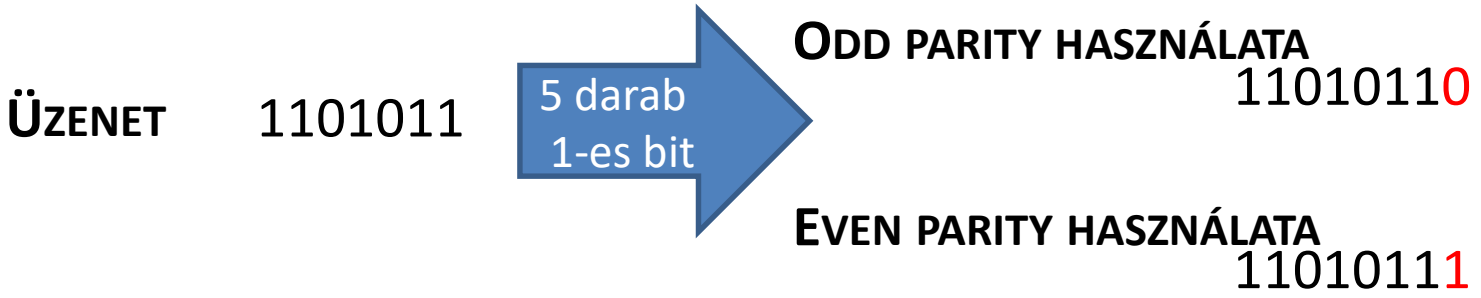
REDUNDANCIA, KÓDOLÁS

Redundancia

- Redundancia nélkül:
  - 2^m lehetséges üzenet írható le *m* biten
  - Ekkor minden hiba egy új helyes üzenetet eredményez → a hiba felismerése lehetetlen
- Emiatt egy keret felépítése:
  - *m* adat bit (üzenet bit)
  - *r* redundáns/ellenőrző bit (üzenetből számolt, új információt nem hordoz)
  - A teljes küldendő keret (kódszó) hossza: *n = m+r*.

Paritás bit használata

- A paritásbitet úgy választjuk meg, hogy ha a kódszóban levő 1-ek száma
  - **Odd parity** – páratlan, akkor 0 befűzése; egyébként 1-es befűzése
  - **Even parity** – páros, akkor 0 befűzése; egyébként 1-es befűzése



Hiba felügyelet Hamming távolsággal

- Hamming távolság: két azonos hosszúságú bitszóban a különböző bitek száma.
- Kiterjesztése azonos hosszúságú bitszavak S halmazára:

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

  - (S halmazt hívják kódkönyvnek vagy egyszerűen kódnak is.)
- *d* bit **hiba felismeréséhez** a megengedett (helyes) keretek halmazában legalább *d+1* Hamming távolság szükséges.
- *d* bit **hiba javításához** a megengedett (helyes) keretek halmazában legalább *2d+1* Hamming távolság szükséges
- 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)

Feladat

- Adott S kódkönyv: $S = [1000010,0011011,1011010,0011101]$
- Adjuk meg S Hamming távolságát ($d(S)$)!
- Adjuk meg S kód rátáját (R_S) és távolságát (δ_S)!
- Mit mondhatunk S hibafelismerő és javító képességéről? Igazoljuk az állításunkat!

Megoldás

	1000010	0011011	1011010	0011101
1000010	0	4	2	6
0011011	4	0	2	2
1011010	2	2	0	4
0011101	6	2	4	0

$\rightarrow d(S) = 2$

- $R_S = \frac{\log_2 |S|}{n} = \frac{\log_2 4}{7} = 0.2857$ és $\delta_S = \frac{2}{7} = 0.2857$
- Max. 1 bithiba ismerhető fel, de 0 javítható (mivel a $d(S) = 2$)

Feladat

Egyetlen paritásbit által nyújtottnál nagyobb biztonságot akarunk elérni, így olyan hibaészlelő sémát alkalmazunk, amelyben két paritásbit van: az egyik a páros, a másik a páratlan bitek ellenőrzésére.

- Mekkora e kód Hamming-távolsága?
- Mennyi egyszerű és milyen hosszú burst-ös hibát képes kezelni?

Megoldás

- A kód Hamming-távolsága 2, mivel a páros pozíciókban lévő paritás bit független a páratlan pozíciókban levőtől, külön-külön pedig könnyen látszik, hogy a H-táv 2 → 1 hibát tudunk jelezni.
- A burst-ös hibánál 3 hosszúságúnál még épp tudjuk jelezni, ha baj van, mivel így vagy a páros vagy a páratlan pozíciókra csak 1 hiba fog esni, azt pedig jelezni fogja a megfelelő paritás bit.

CRC, MD5

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 $x^rM(x)$ polinomot fogja reprezentálni.

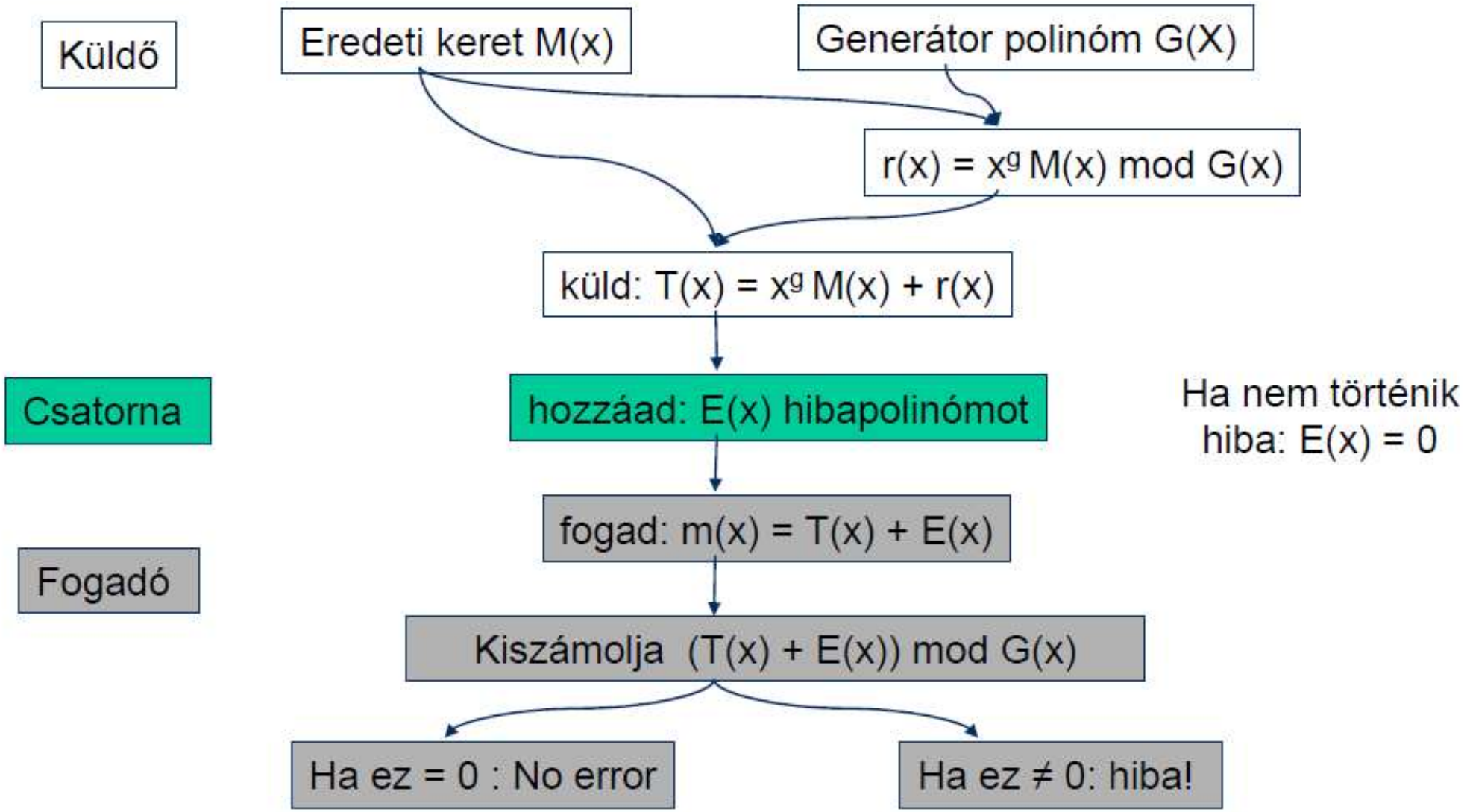
2. Osszuk el az $x^rM(x)$-hez tartozó bitsorozatot a G(x)-hez tartozó bitsorozattal modulo 2

3. Vonjuk ki a maradékot (mely mindig r vagy kevesebb bitet tartalmaz) az $x^rM(x)$-hez tartozó bitsorozatból. 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.

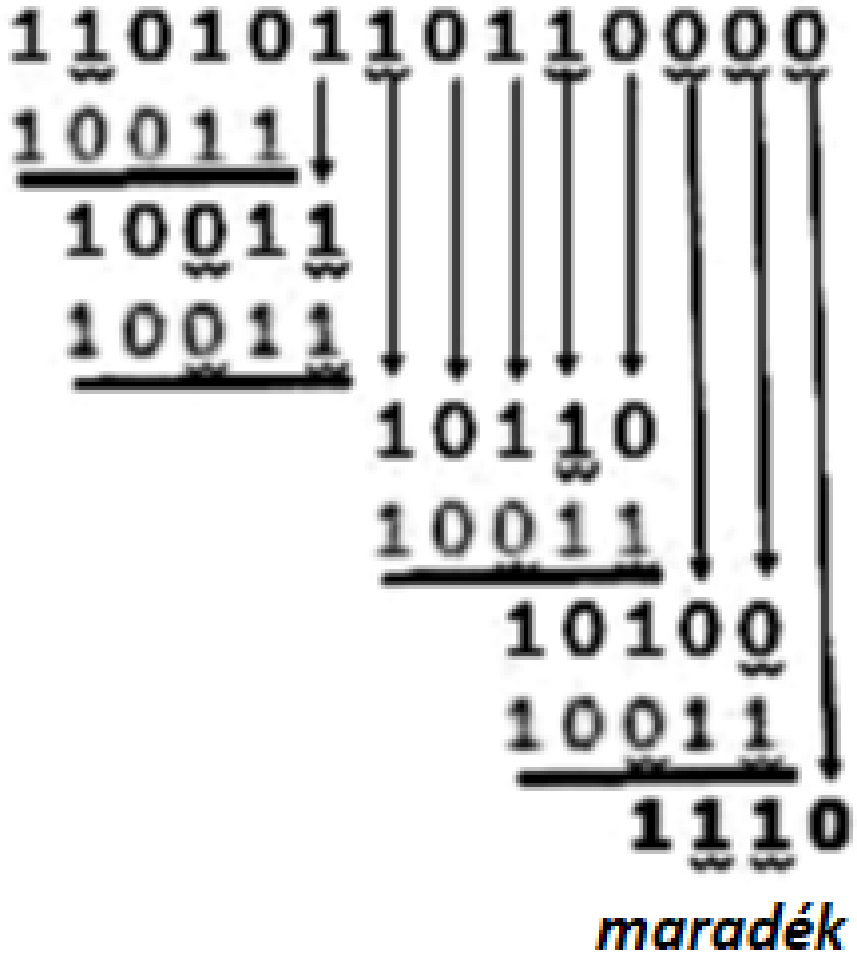
- Ha az osztási maradék, amit R(x) jelöl, nem nulla, akkor hiba történt

CRC


Küldő
Eredeti keret M(x)
Generátor polinóm G(X)
r(x) = x^g M(x) mod G(x)
küld: T(x) = x^g M(x) + r(x)
Csatorna
hozzáad: E(x) hibapolinómot
Ha nem történik hiba: E(x) = 0
fogad: m(x) = T(x) + E(x)
Fogadó
Kiszámolja (T(x) + E(x)) mod G(x)
Ha ez = 0 : No error
Ha ez ≠ 0: hiba!

CRC példa

- Keret: 1101011011
- Generátor: 10011 -> (x^4+x+1)
  4 fokú polinom
- Kiegészített keret:
  11010110110000
- Osztás binárisan: XOR művelet
- Maradék: 1110
- A továbbítandó üzenet:
  11010110111110



CRC példa

- Az osztásban 11010110110000 az $x^{13}+x^{12}+x^{10}+x^8+x^7+x^5+x^4$ polinomot reprezentálja. 10011 pedig az x^4+x+1 polinomot.
- Ebből (11010110110000) * (1110) = 11010110111110 ami az elküldendő keretünk lesz.
- (11010110111110) mod 10011 = 0
- Amennyiben hozzáadtunk volna egy E(x) hibapolinomot (pl. $E(x) = x^2+x$ =110), akkor a maradék nem nulla (a példában 11) lenne, így tudnánk, hogy meghibásodott a keret.

CRC, MD5 pythonban

- CRC

```
import binascii, zlib

test_string = "Fekete retek rettenetes".encode('utf-8')

print(hex(binascii.crc32(bytearray(test_string))))
print(hex(zlib.crc32(test_string)))
```

- MD5

```
import hashlib

test_string = "Fekete retek rettenetes".encode('utf-8')

m = hashlib.md5()
m.update(test_string)
print(m.hexdigest())
```

FÁJLÁTVITEL (SZÖVEG ÉS KÉP)

Fájl átvitel

- fájl bináris megnyitása

```
with open („input.txt”, „rb”) as f:
        …
```

- read(x) – x bytes

```
        …
        f.read(128)    #128 byte-ot fog beolvasni
```

„When size is omitted or negative, the entire contents of the file will be read and returned; it’s your problem if the file is twice as large as your machine’s memory. „ - python.org

Feladat - Bináris fájl átvitel

- Juttassunk át egy szöveget tartalamzó fájlt (input.txt) a kilensről a szerverre bináris módon. (Ez most lokálisan egy fájl másolásnak fog megfelelni.) Az új fájl neve legyen output.txt.

- Egészítsük ki a kódot úgy, hogy egy bármilyen fájlformátumot (pl. png kép) is át tudjunk küldeni. A kliens paramétere az eredeti fájl, a szerver paramétere az új létrejövő fájl neve legyen.

Órai feladat - Fájl átvitel ellenőrzése

- Küldjünk át binárisan egy fájlt a parancssori argumentumról a kliensről a szerverre.
- A kliens számoljon ki a fájlból egy md5-ös kódot, majd azt is küldje át a szervernek.
- A szerver fogadja a kódot is és a kapott fájlra ő is számoljon ki egy md5-ös kódot.
- Majd a szerver vesse össze a két md5-ös kódot, ha megegyezik, akkor írjaki stdout-ra, hogy "OK", ha nem akkor pedig "HIBA".

VÉGE