

Számítógépes Hálózatok

6. gyakorlat

Elérhetőségek

- honlap: <http://szalaigj.web.elte.hu/>
- email: szalaigindl@inf.elte.hu
- szoba: 2.507 (déli tömb)

Óra eleji kisZH

- Elérés:
 - <https://oktnb16.inf.elte.hu>



Connect to the TAO platform

Login

Password

[Guest access](#)

[Log in](#)

Gyakorlat tematika

- Bájtbeszűrés, Bitbeszűrés
- Hamming-távolság, Hamming-kód
- Órai / házi feladat

Keretezés – emlékeztető

- Adatkapcsolati réteg egyik legfontosabb feladata: a hibák jelzése és javítása
- Emiatt is szükség van a bitfolyam keretekre tördelésére
- Keret képzés fajtái:
 - Bájt alapú protokollok
 - Bit alapú protokollok
 - Óra alapú protokollok

Bájt alapú: bájt beszúrás – emlékeztető

- Egy speciális **FLAG** bájt (jelölő bájt) jelzi az adat keret elejét és végét
- Ha **FLAG** szerepel az adatok között is, akkor speciális **ESC** bájt beszúrása elé
- Ha **ESC** is szerepel az adatok között, akkor az elé még egy **ESC** bájt beszúrása

Feladat 1

- A bájt-beszúrási módszer esetén, hogy kerül átvitelre a következő adat:

| | | | | | | | | |
|---|---|---|------|------|-----|---|------|---|
| X | Y | Z | FLAG | FLAG | ESC | A | FLAG | B |
|---|---|---|------|------|-----|---|------|---|

- Mi az oka, hogy ennél a módszernél a keretek végére is kerülni kell **FLAG** bájtoknak? Nem lehetne összevonni az első keret végét jelző **FLAG** bájtot a második keret elejét jelző **FLAG** bájttal?

Feladat 1 megoldása

| | | | | | | | | | | | | | | |
|------|---|---|---|-----|------|-----|------|-----|-----|---|-----|------|---|------|
| FLAG | X | Y | Z | ESC | FLAG | ESC | FLAG | ESC | ESC | A | ESC | FLAG | B | FLAG |
|------|---|---|---|-----|------|-----|------|-----|-----|---|-----|------|---|------|

Bit alapú: bit beszúrás – emlékeztető

- A bájt beszúrásos módszer hátránya, hogy meg vagyunk kötve a 8-bit (vagy annak többszöröse) szerinti karakter kódolásra.
- A bit beszúrásnál egy speciális bitminta jelzi az adat keret elejét és végét (pl.: 01111110)
- A küldő az adatban előforduló minden 1111 részsorozat után 0 bitet szúr be
- Ezen a módon az elválasztó minta (01111110) *elvileg* nem fordulhat elő az adatot kódoló részben véletlenül

Feladat 2

- Az "A B C A" karaktereket szeretnénk átvinni bit beszúrással. Tegyük fel, hogy az alábbi karakter kódolás adott (a 4 bitenkénti elválasztás csak az olvashatóság miatt):

| | |
|---|-----------|
| A | 1110 1111 |
| B | 1100 0011 |
| C | 1111 0111 |

- A keret elejét és végét jelző bitminta (flag bájt): 01111110.
- Hogy fog kinézni az üzenet a módszer alkalmazása után?

Feladat 2 megoldása

- Az adat binárisan (a 4 bitenkénti elválasztás csak az olvashatóság miatt):
- 1110 1111 1100 0011 1111 0111 1110 1111
- Kódolás után:
- 0111 1110 1110 1111 10100 0011 11101 0111
11010 1111 0111 1110

Redundancia – emlékeztető

- Redundancia nélkül:
 - 2^m lehetséges üzenet írható le m biten
 - Ekkor minden hiba egy új helyes üzenetet eredményez \rightarrow 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: $n = m+r$.

Hiba felügyelet Hamming távolsággal – emlékeztető

- Hamming távolság: két azonos hosszúságú bitszóban a különböző bitek száma.
- Kiterjesztése azonos hosszúságú bitszavak S halmazára: $d(S) := \min_{x,y \in S \wedge x \neq y} d(x,y)$
 - (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 3

- 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!

Feladat 3 megoldása

| | 1000010 | 0011011 | 1011010 | 0011101 | |
|---------|---------|---------|---------|---------|--------------|
| 1000010 | 0 | 4 | 2 | 6 | |
| 0011011 | 4 | 0 | 2 | 2 | → $d(S) = 2$ |
| 1011010 | 2 | 2 | 0 | 4 | |
| 0011101 | 6 | 2 | 4 | 0 | |

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

Órai / házi feladat

Bit / Byte beszúrás és hibajavítás Hamming kóddal

- Adott kódkönyv, amit a kliens és a server is ismer
- A kliens szkript argumentumai:
 - Első: protokoll (bit vagy byte)
 - Második: keretezésre szánt üzenet
 - Az üzenetben lehessen jelölni, hogy az egyik adatkarakter hibásan legyen átküldve
- A kliens csatlakozása után küldje el a használt protokollt a servernek
- A kliens készítse el a keretezést (byte/bitbeszúrás alapján), és küldje el a keretezett üzenetet a servernek
 - A keretezett üzenet legyen '0' és '1' karakterekből álló egyszerű sztring

Órai / házi feladat

Bit / Byte beszúrás és hibajavítás Hamming kóddal

- A keretben található byte-okat a szerver a kétkönyv alapján oldja fel
 - Előtte írja ki a keretezett üzenetet a kimenetre
 - Ha ismeretlen a kódszó, akkor a Hamming-távolság alapján a legközelebbit értelmezzük
 - Ha több ilyen is van, akkor jelezzük a hibát (nem kell visszaküldeni, csak kiíratni)
 - A végén írja ki az eredeti üzenetet

Órai / házi feladat

Bit / Byte beszúrás és hibajavítás Hamming kóddal

- Kódkönyv:

```
codebook = {'FLAG': '01111110', 'A': '10000001', 'B':  
'00001110', 'C': '10110011', 'D': '11100110', 'ESC':  
'01110000'}
```

- Hibás bitsorozat 'C'-re, pl.:

- 1 hiba: 10010011

- 2 hiba: 10000011

- A hiba a bit/bájt beszúrás alkalmazása előtt történjen meg

- A tesztelhetőség miatt van erre szükség

Órai / házi feladat

Bit / Byte beszúrás és hibajavítás Hamming kóddal

- Ennek a házi feladatnak a pontozása (kivételesen):
 - Mindenképpen működnie kell a kliens-szerver kommunikációnak, továbbá:
 - Bit beszúrás megfelelően működik (keretezéssel, dekódolással együtt) → 1 pont
 - Bájt beszúrás megfelelően működik (keretezéssel, dekódolással együtt) → 1 pont
 - Hibás bit sorozat kezelése, javítása Hamming-távolsággal → 1 pont
- Tehát **1 extra pontot** lehet szerezni erre a házira, ha valaki mindegyik részét megfelelően implementálja
 - (mivel egy jól megoldott házira alapvetően 2 pont jár)

Órai / házi feladat

Bit / Byte beszúrás és hibajavítás Hamming kóddal

- Segítség: a bit hibák alkalmazása egy karakterre:

```
import random
...
def applyError(character, errorCnt):
    encodedCh = codebook[character]
    errorPositions = random.sample(range(len(encodedCh)), errorCnt)
    encChL = list(encodedCh)
    for pos in errorPositions:
        encChL[pos] = '0' if encChL[pos]=='1' else '1' # bit swapping
    encodedCh = ''.join(encChL)
    return encodedCh
```

Órai / házi feladat

Bit / Byte beszúrás és hibajavítás Hamming kóddal

- Kliens hívására példák:

- `haziBitByteHammingClient.py bit 'A,B,C,D'`
- `haziBitByteHammingClient.py bit 'A,B,FLAG,C,D,FLAG'`
- `haziBitByteHammingClient.py byte 'A,B,ESC,FLAG,C,D'`
- `haziBitByteHammingClient.py byte 'A,B,ESC,ESC,C,D'`
- `haziBitByteHammingClient.py byte 'FLAG,A,B,ESC,FLAG,C_1_HIBA,D,FLAG'`
- `haziBitByteHammingClient.py bit 'FLAG,A,B,ESC,FLAG,C,D_2_HIBA,FLAG'`

VÉGE
KÖSZÖNÖM A FIGYELMET!