# Számítógépes Hálózatok
# 2012

## 3. Adatkapcsolati réteg –
## Hibafelismerés és javítás, Hamming távolság, blokk kódok

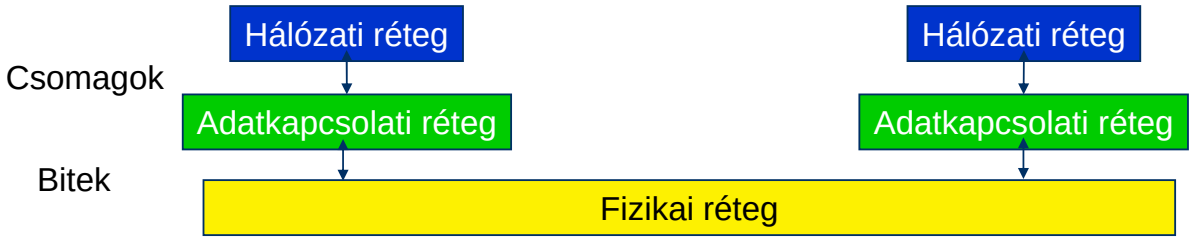


## Adatkapcsolati réteg (Data Link Layer)

- Az adatkapcsolati réteg feladatai:
  - Szolgáltatásokat rendelkezésre bocsátani a hálózati rétegnek
  - Keretek (frames)
  - Hibafelügyelet
  - Folyamfelügyelet
- Hibafelismerés és javítás
  - Hibajavító kódok
  - Hibafelismerő kódok
- Elemi adatkapcsolati protokollok
  - Simplex
  - „Stop-and-Wait“
  - „Noisy Channel“
- Csúszó ablak (sliding window)
  - 1-Bit-Sliding Window
  - „Go Back N“
  - „Selective Repeat“
- Protokoll-verifikáció
  - Véges automaták
  - Petri hálók

## Az adatkapcsolati réteg szolgáltatásai

- Az adatátkapcsolati réteg szituációja
  - a fizikai réteg biteket visz át
  - struktúra nélkül és esetleg hibásan
- A hálózati réteg az adatkapcsolati rétegtől a következőket várja el:
  - hibamentes átvitel
  - strukturált adatok átvitele
    - adatcsomagok vagy adatáram
  - zavarmentes adatfolyam






## Az adatkapcsolati réteg lehetséges szolgáltatásai

- Megbízható szolgáltatás?
  - A küldött és a fogadott csomagnak egyformának kell lenni
  - Minden elküldött csomagnak meg kell érkezni (valamikor)
  - A csomagoknak a megfelelő sorrendben kell megérkezni
  - **Hibafelügyelet** szükséges lehet
- Kapcsolat-orientált?
  - A pont-pont kapcsolat egy nagyobb összefüggésben van?
  - Kapcsolatnak foglalás szükséges?
- Csomagok vagy adatáram (bitáram)?

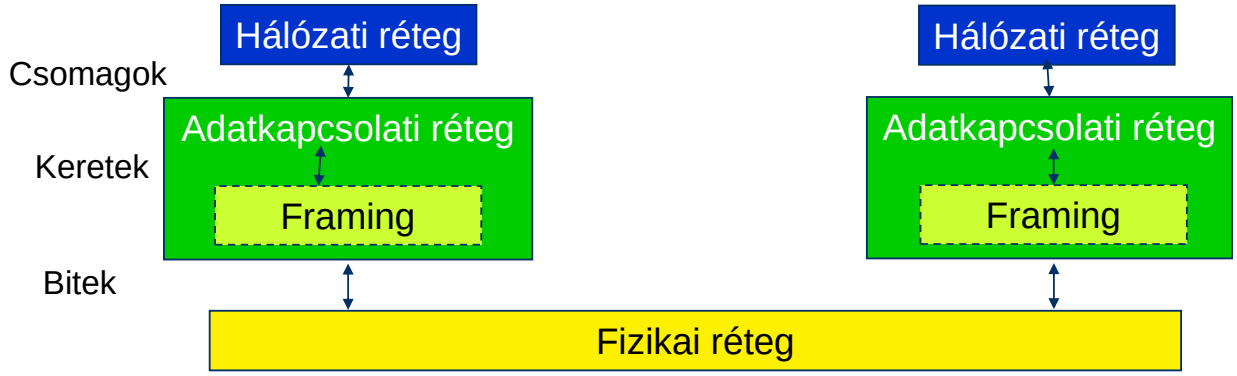
## Megkülönböztetés: szolgáltatás és implementáció

- Példa
  - A hálózati réteg kapcsolatmentes és megbízható szolgáltatást követel
  - Az adatkapcsolati réteg **intern** kapcsolatorientált szolgáltatást használ hibakontrollal
- Más kombinációk lehetségesek



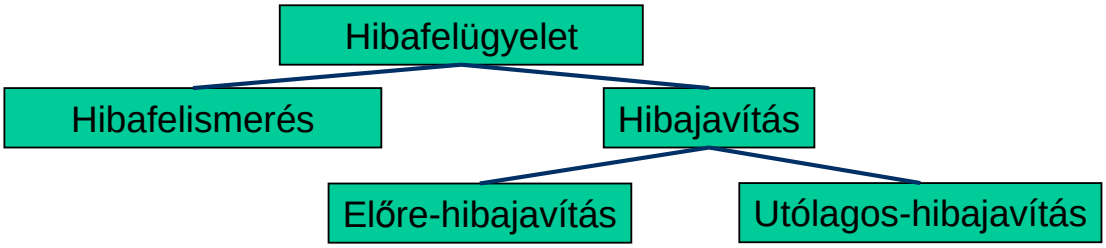
## Keretek (frames)

- A fizikai réteg bitáramát darabokra, u.n. **keret**ekre (**frames**) osztjuk
  - Szükséges a hibafelügyelethez
  - A keretek az adatkapcsolati réteg csomagjai
- Keretekre-osztás (fragmentálás és a fogadó oldalon defragmentálás) szükséges, ha a hálózati réteg csomagjai nagyobbak, mint a keretek



## Hibafelügyelet

- Minimálisan megkövetelt szolgáltatás az adatkapcsolati rétegtől
  - Keretek segítségével
- Hibafelismerés: van-e hibásan átvitt bit
- Hibajavítás: bithibák megtisztítása
  - Előre-hibajavítás (Forward Error Correction)
    - Redundáns kód használata, amely újraátvitel nélkül lehetővé teszi a hiba kijavítását
  - Útólagos-hibajavítás (Backward Error Correction)
    - A hiba felismerése után a hiba utólagos kommunikációval kerül kijavításra






## Kapcsolat felépítés

- Kapcsolatok használata
  - A kapcsolat állapotának felügyelete
    - Protokollok helyessége
  - Hibafelügyelet
    - Különböző hibafelügyeleti módszerek megbíznak a küldő és a fogadó közös kontextusában
- Kapcsolatok felépítése és befejezése
  - Virtuális kapcsolatok
    - A bit-áram interpretációja
  - Keretek által
  - Különösen fontos vezeték nélküli médiumok esetén
- A problémát a szállítói rétegnél átfogóan tárgyaljuk
  - L. OSI-modell ülés réteg

## Folyamfelügyelet

- Probléma: gyors küldő és lassú fogadó
  - A küldő túlárasztja a fogadó pufferét
  - Az átvitel sávszélességét elpazarolják az értelmetlen újraküldések (a hibafelismerés után)

Lassú fogadó


Gyors küldő

- Szükséges a keretküldési ráta hozzáigazítása a fogadóhoz



## Keretek (frames)

- Hol kezdődik egy keret és hol ér véget?

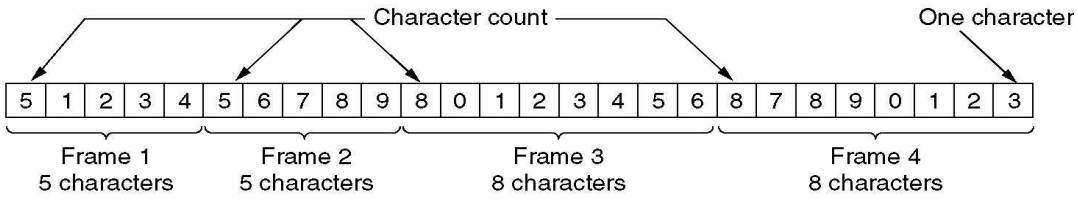
Átvitt bit-áram: 01100101011101011100101000101010101010101100010

keret kezdete? keret vége?

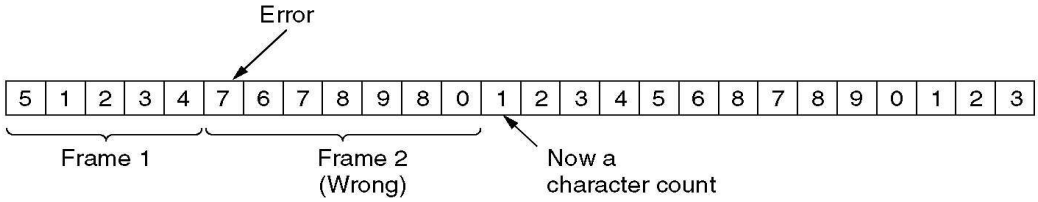
- Figyelem:
  - A fizikai réteg akkor is küldhet biteket, ha a küldő valójában semmit se küld
  - A fogadó
    - interpretálhatná a médium zaját
    - adhatná a 00000000.... sorozatot
      - adat vagy kontroll információ?

## Kerethatárok hosszinformációval?

- Ötlet: A keret fejlécében jelezni a bitek számát



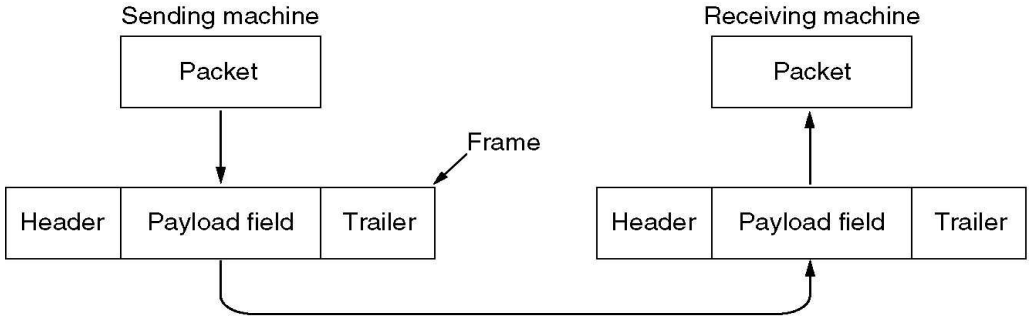

- Probléma: Mi történik, ha a keret hossza hibásan kerül átvitelre?
  - A fogadó elveszti az ütemet és új értelmetlen kereteket interpretál
- Változó keretméret hosszinformációval így nem jó koncepció






## Fejléc és lezáró (header and trailer)

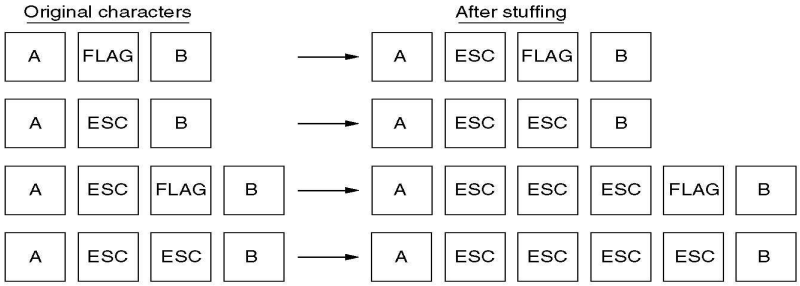
- Header és Trailer
  - legtöbbször a keret kezdetén használnak egy **Header**-t a végén pedig egy **Trailer**-t
  - jelzik a keret kezdetét és végét
  - kontrollinformációt hordoznak
    - Pl. küldő, fogadó, kerettipus, hibafelügyeleti információ



## Flag byte és byte beszúrás (byte stuffing)

- Speciális “Flag Byte”-ok jelzik a keret kezdetét és végét

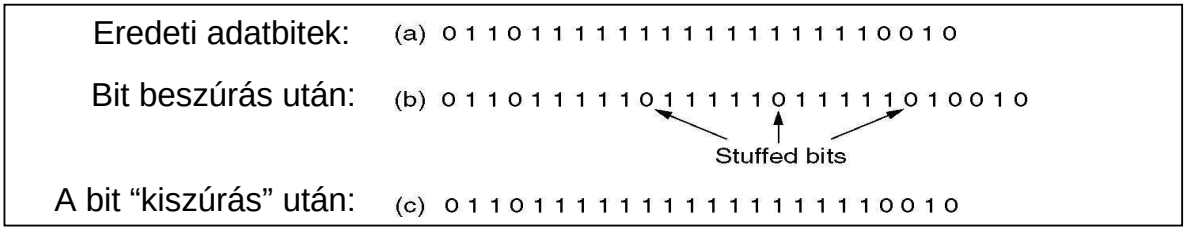

- **Byte beszúrás** (**byte stuffing**):
  - Ha a „flag-byte” a küldendő adatok között előfordul, akkor
    - mint adatbyte-ot egy másik speciális jellel (Escape) kell jelezni
  - Ha a másik speciális jel (Escape) a küldendő adatok között előfordul, azt is.






## Keretek bit beszúrással (bit stuffing)

- Byte beszúrás egy byte-ot vesz elemi egységnek.
  Hasonló módszer működik a bitekkel is
- Flag bits és **bit beszúrás** (**bit stuffing**)
  - flag byte helyett egy bitsorozatot használunk, pl.: 01111110
  - bit beszúrás:
    - Ha a küldő öt 1-esből álló sorozatot küld, automatikusan beszúr a bit-áramba egy 0-t
      – a flag bit-ek kivételével
- A fogadó, ha öt 1-es után egy 0-t kap, törli a 0-t



## Keretek kód megsértés által

- Lehetséges egy játéktér a bitek szignálokká kódolásával a fizikai rétegen
  - ha nem használja fel az összes lehetséges kombinációt a kódoláshoz
  - Pl.: a Manchester-kód csak a mély/magas és a magas/mély átmenetet használja
- A kódolási szabályok megsértésével lehet a keret kezdetét és végét jelezni
  - Pl.: Manchester – hozzáadjuk a magas/magas-t vagy a mély/mély-t
    - A Manchester önütemezése veszélybe kerül?
- Egyszerű és robosztus módszer
  - Például az Ethernet használja
  - Költség? A sávszélesség hatékony felhasználása?

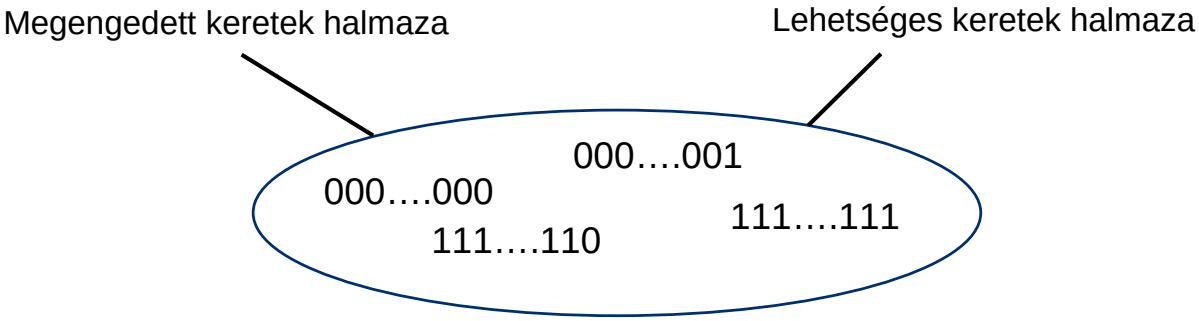


## Hibafelügyelet

- Feladatok:
  - Hibák felismerése (hibás bitek) egy keretben
  - Hibák javítása egy keretben
- Ezen feladatok minden kombinációja előfordul
  - Felismerés javítás nélkül
    - Keret törlése további értesítés nélkül (drop a frame)
    - Magasabb rétegeknek kell a problémát kezelni
  - Javítás felismerés nélkül
    - Lehető legjobban megtisztítja a bithibákat, de esetleg még maradnak bithibák
    - Van értelme, ha a felhasználás a hibát tolerálni tudja
      - Pl.: Hangátvitel
    - Alapvetően létjogosult, mert mindig marad egy pozitív hiba valószínűség

## Redundancia

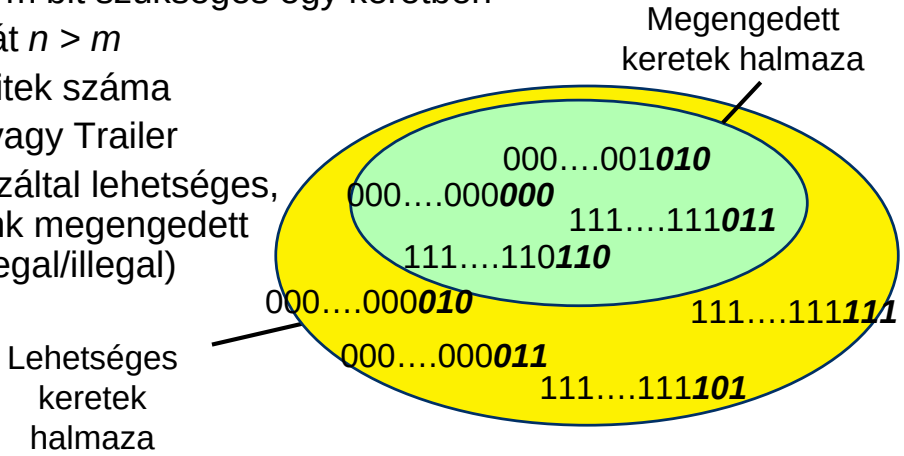
- Redundancia előfeltétele a hibafelügyeletnek
- Redundancia nélkül
  - egy m hosszúságú keret $2^m$ féle lehetséges adatot reprezentálhat,
  - ezek mindegyike megengedett
- Egy hibás bit egy új adattartalmat eredményez






## Redundancia

- Alapötlet:
  - Bizonyos lehetséges m bit hosszú adatok nem megengedettek
  - Ekkor $2^m$ különböző adat ábrázolásához
    - több mint $2^m$ lehetséges keret szükséges
    - tehát több mint m bit szükséges egy keretben
  - A keret hossza tehát $n > m$
  - $r=n-m$ redundáns bitek száma
    - Pl. Header és/vagy Trailer
- Hibafelügyeletet csak azáltal lehetséges, hogy megkülönböztetünk megengedett és nem megengedett (legal/illegal) kereteket



## Legegyszerűbb redundancia: paritás bit

- Egy egyszerű szabály egy redundáns bit hozzáadására (azaz, n=m+1): a paritás
  - **Odd parity**
    - 0-t fűzünk be, ha az 1-es bitek száma páratlan, egyébként 1-est
  - **Even parity**
    - 0-t fűzünk be, ha az 1-es bitek száma páros, egyébként 1-est
- Példa:
  - Eredeti üzenet redundancia nélkül: 01101011001
  - Odd parity: 011010110011***1***
  - Even parity: 011010110011***0***

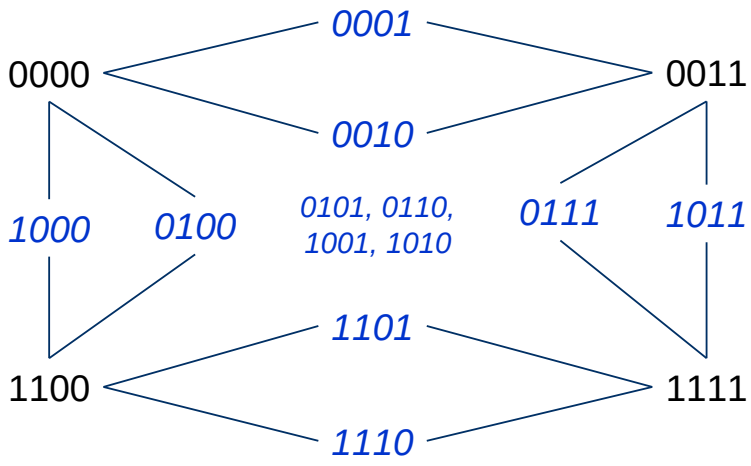


## Nem megengedett keretek haszna

- A küldő csak megengedett kereteket küld
- A fizikai rétegben a bitek meghibásodhatnak
- Remény:
  - Megengedett keretek meghibásodása mindig nem megengedett keret eredményez
  - sohasem egy másik megengedett keretet
- Szükséges feltétel:
  - A fizikai rétegben csak legfeljebb bizonyos számú bit változhat meg
    - pl. $\leq$ k bit per keret
  - A megengedett keretek elegendően különbözőek ahhoz, hogy ezt a keret-hiba rátát felismerje

## A keret megváltozása bit hiba által

- Tegyük fel, hogy a következő keretek megengedettek: 0000, 0011, 1100, 1111




Az élek olyan keret párokat kötnek össze, melyek csak egy bitben különböznek

uvxy – megengedett *abcd* – nem megengedett

Egy egyszerű bithiba nem tud egy megengedett keretet egy másik megengedett keretté változtatni!



## Hamming távolság

- Az előző példában két megengedett üzenet “távolsága” egymástól legalább két bit volt
- Definíció:

  Legyen $x=x_1,\ldots, x_n$ és $y=y_1,\ldots, y_n$ két üzenet

  A **Hamming távolság d(x,y) = az 1-es bitek száma (x XOR y)-ban**
- Intuitíven: azon pozíciók száma, amelyeken x és y különbözik
- A Hamming távolság egy metrika:
  - Nem-negatív,
  - idempotens,
  - szimetrikus,
  - háromszögegyenlőtlenség

Példa:
x=0011010111
y=0110100101
x XOR y=0101110010

d(x,y) = 5

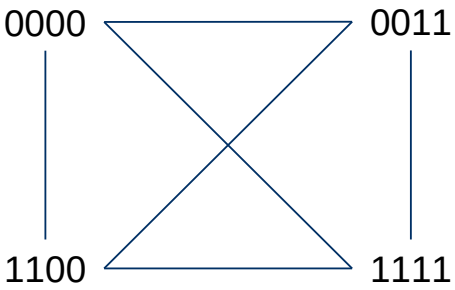
## Üzenethalmazok Hamming távolsága

- Legyen S (egyenlő hosszú) bit-sztringek halmaza. **S Hamming távolsága**:

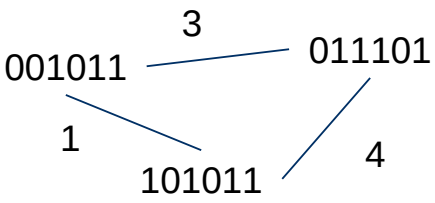
$$d(S) = \min_{x,y \in S, x \neq y} d(x,y)$$

Azaz a legkisebb távolság két különböző S-beli bit-sztring között

Példa:




Minden távolság $\geq 2$




Egy távolság 1!



## Hibafelismerés és javítás Hamming távolsággal

1. eset d(S)=1
   - Nincs hibafelismerés
   - Egy megengedett keretből 1 bit megváltozásával másik megengedett keret állhat elő
2. eset d(S) = 2
   - Ekkor minden x,y $\in$ S: d(x,y) $\geq$ 2
   - Így minden u, melyre d(x,u) = 1, nem megengedett,
     - mint ahogy minden u, melyre d(y,u)=1, se

x —(1 bit különbség)— u —(1 bit különbség)— y

   - 1-bit-hiba
     - mindig felismerhető
     - de nem javítható

## Hibafelismerés és javítás Hamming távolsággal

3. eset d(S) = 3

- Ekkor minden x,y $\in$ S : d(x,y) $\geq$ 3
- Minden u, melyre d(x,u) = 1, nem megengedett és d(y,u) > 1

x —(1 bit különbség)— u —(1 bit különbség)— v —(1 bit különbség)— y

- Ha a fogadóhoz u érkezik, a következő esetek lehetségesek:
  - x került átvitelre és 1 bit hibával érkezett
  - y került átvitelre és 2 bit hibával érkezett
  - valami más került átvitelre és legalább 2 bit hibával érkezett
- Tehát a valószínűbb, hogy x került átvitelre, mint az hogy y



## Hibafelismerés és javítás Hamming távolsággal

- Ahhoz, hogy d bit hibát felismerjünk, a megengedett keretek halmazában legalább d+1 Hamming távolság szükséges

- Ahhoz, hogy d bit hibát javíthassunk, a megengedett keretek halmazában legalább 2d+1 Hamming távolság szükséges

## Kód-könyvek, kódok

- A megengedett keretek S $\subseteq \{0,1\}^n$ halmazát **kód-könyv**nek vagy egyszerűen **kód**nak nevezzük.
  - Definíció: Egy S **kód $R_S$ rátája**: $R_S = \frac{\log |S|}{n}$
    - A ráta karakterizálja a kód hatékonyságát
  - Definíció: Egy S **kód $\delta_S$ távolsága**: $\delta_S = \frac{d(S)}{n}$
    - A távolság karakterizálja a hibajavítási vagy hibafelismerési lehetőségeket
- Jó kódoknak a rátája és a távolsága is nagy
  - trade-offs



## Blokk kódok

- Blokk kódok k bites eredeti adatot n kódolt bitben kódolnak
  - n-k bitet adunk hozzá
  - Bináris blokk kódok legfeljebb t hibát tudnak felismerni egy n hosszú kód-szóban (keretben) k eredeti bittel, ahol (Gilbert-Varshamov-korlát):

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i}$$

  - Ez egy elméleti felső korlát
    - Nem minden t,k és n esetén ismert vagy lehetséges ilyen kód
- Példák
  - Bose Chaudhuri Hocquenghem (BCH) kódok
    - Véges testek (Galois-testek) polinomjain alapulnak
  - Reed Solomon kódok
    - Nem bináris BCH kódok speciális esete