Számítógépes Hálózatok

5. gyakorlat

Feladat - Chat

- Készítsünk egy chat alkalmazást, amelyen a chat szerverhez csatlakozott kliensek képes beszélni egymással!

- A szerver szerepe, hogy a kliensektől jövő üzenetet minden más kliensnek továbbítja névvel együtt: [<név>] <üzenet> ; pl. [Józsi] Kék az ég!

- A kliensek a szervertől jövő üzeneteket kiírják a képernyőre.

Házi feladat – 1 pont

- Készítsünk egy barkóba alkalmazást. A szerver legyen képes kiszolgálni több klienst. A szerver válasszon egy egész számot 1..100 között véletlenszerűen. A kliensek próbálják kitalálni a számot.
- A kliens üzenete egy összehasonlító operátor: <, >, = és egy egész szám, melyek jelentése: kisebb-e, nagyobb-e, mint az egész szám, illetve rákérdez a számra. A kérdésekre a szerver Igen/Nem/Nyertél/Kiestél/Vége üzenetekkel tud válaszolni. A Nyertél és Kiestél válaszok csak a rákérdezés (=) esetén lehetségesek.
- Ha egy kliens kitalálta a számot, akkor a szerver minden újabb kliens üzenetre az „Vége” üzenetet küldi, amire a kliensek kilépnek. A szerver addig nem választ új számot, amíg minden kliens ki nem lépett.
- • Nyertél, Kiestél és Vége üzenet fogadása esetén a kliens bontja a kapcsolatot és terminál. Igen/Nem esetén folytatja a kérdezgetést.
- A kommunikációhoz TCP-t használjunk!
- Folytatás a következő oldalon!

Házi feladat – 1 pont

- A kliens logaritmikus keresés segítségével találja ki a gondolt számot. A kliens tudja, hogy milyen intervallumból választott a szerver.
- AZAZ a kliens NE a standard inputról dolgozzon.
- Minden kérdés küldése előtt véletlenszerűen várjon 1-5 mp-et. Ezzel több kliens tesztelése is lehetséges lesz.
- Folytatás a következő oldalon!

Házi feladat – 1 pont

- Üzenet formátum:
  - Klienstől: bináris formában **egy db karakter, 32 bites egész szám**
    A karakter lehet: <: kisebb-e, >: nagyobb-e, =: egyenlő-e
  - Szervertől: ugyanaz a bináris formátum, de a számnak nincs szerepe (bármi lehet)
    A karakter lehet: I: Igen, N: Nem, K: Kiestél, Y: Nyertél, V: Vége
- Fájlnevek és parancssori argumentumok:
- Szerver: **server.py** <bind_address> <bind_port> # A bindolás során használt pár
- Kliens: **client.py** <server_address> <server_port> # A szerver elérhetősége

- Beadási határidő: **BEAD rendszerben**

Kódolások, Hamming-távolság

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?

Feladat

Tekintsük a következő paritás-technikát. Tekintsük az n küldendő adatbitet, mint egy k×l bit-mátrixot. Minden oszlophoz számoljon ki egy paritás-bitet (pl. odd parity) és egészítse ki a mátrixot egy új sorral, mely ezeket a paritás-biteket tartalmazza. Küldje el az adatokat soronként.

Példa k = 2, l = 3 esetén (odd-parity):

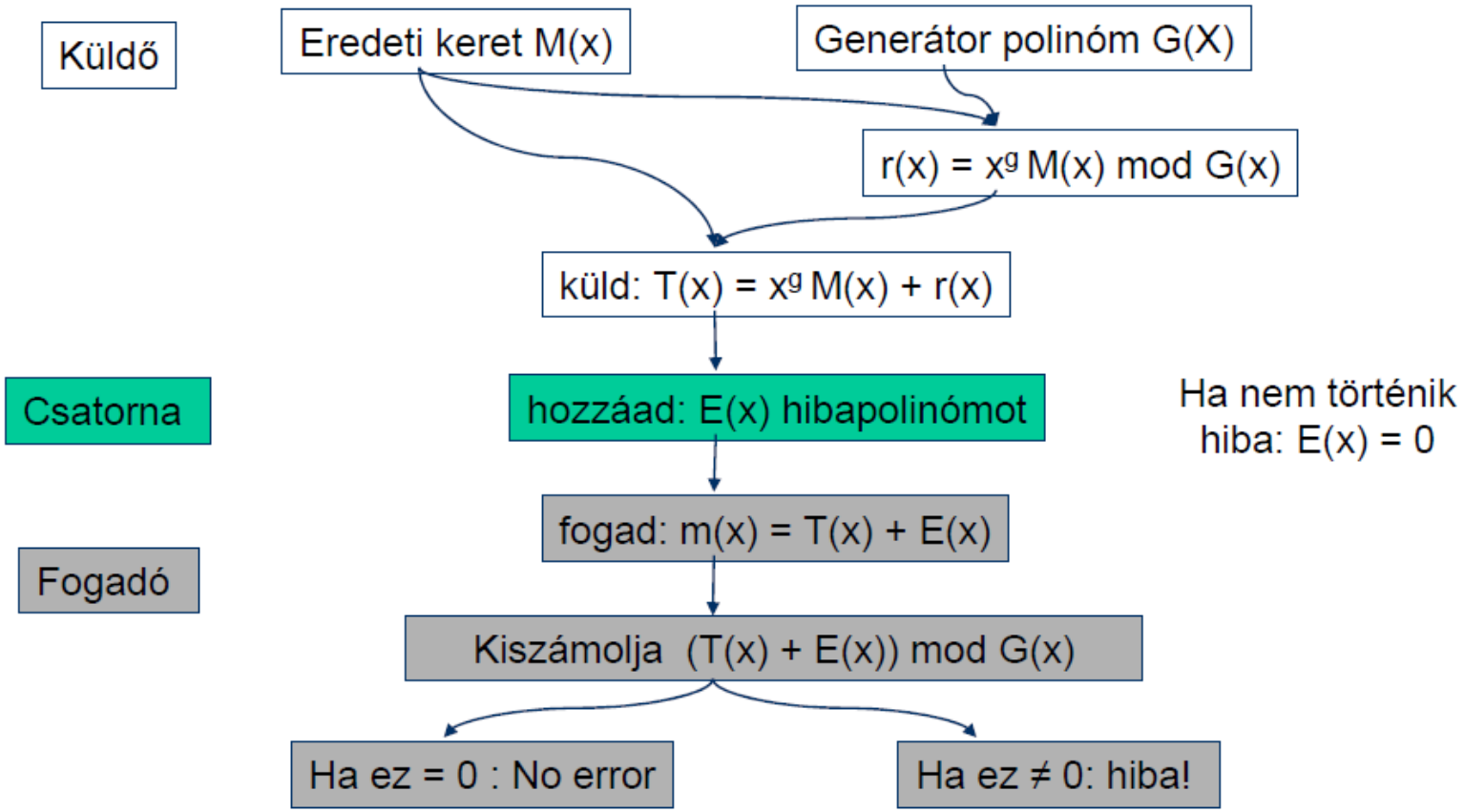
1 0 1
0 1 1
0 0 1

a) Adjon egy példát k = 3, l = 4 esetén.

b) Hogy viselkedik ez a módszer egyszerű bit-hibák és löketszerű bit-hibák esetén? Milyen hosszú lehet egy bitsorozat, melynek minden bitje hibás, hogy a hibát felismerjük?

c) Egészítse ki a mátrixot egy új oszloppal is, amely minden sorhoz paritás-bitet tartalmaz (két dimenziós paritás technika). Hogyan használható ez a módszer 1-bithiba javítására? Mi a helyzet több bithibával és löketszerű-hibákkal.

Feladat


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!

Feladat

Számolja ki a `0101.1011.1101.0010` inputhoz a 3-bit CRC kontrollösszeget, ha a generátor polinom $x^3 + x^2 + 1$. Adjon egy olyan inputot, amely 1-gyel kezdődik és ugyanezt a kontrollösszeget eredményezi.

Feladat

Az előbbi üzenetet a fogadó meg kapta. Ellenőrizzük, történt-e hiba:
Elöbbi üzenet: `0101.1011.1101.0010`

```
A -       0101.1011.1101.0010 101

B -       0101.1101.1101.0010 101

C -       0101.1101.0110.0110 101
```

A generátor polinom $x^3 + x^2 + 1$.

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())
```

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

VÉGE