

Telekommunikációs Hálózatok

6. gyakorlat

Feladat 1 (Számológép)

- Készítsünk egy proxy szerveret
 - TCP-s számológép klienstől kapja az üzenetet. (korábbi gyakorlat)
 - UDP-s számológép szervernek küldi tovább
 - A szerver visszaküldi a proxynak, aki visszaküldi a kliensnek

Feladat 2 (Számológép 2)

- Egy számológép kliens az UDP szervertől kérje el a TCP-s szerver elérhetőségét!
 - Küldjön egy ,GET' üzenetet
- A kliens küldjön egy ,Hello Server' üzenetet a UDP szervernek, aki visszaküldi a TCP szerver elérését, ahova a számokat és az operátort fogja elküldeni.
- A TCP szerver legyen a korábbi számológép szerver

Feladat 3 (Web proxy)

- Készítsünk proxyt, ahol a kliens egy webböngésző, a szerver pedig egy webszerver.
- A proxy továbbítja a böngésző kérését a szervernek.
- Pl: `python netProxy.py ggombos.web.elte.hu 9000`
- Böngészőben: `localhost 9000`

Feladat 4

- Tekintsük a következő paritás-technikát. Tekintsük az n küldendő adatbitet, mint egy $k \times l$ bit mátrixot. Minden oszlophoz számoljunk ki egy paritás-bitet (odd parity) és egészítsük ki a mátrixot egy új sorral, mely ezeket a paritás-biteket tartalmazza. Küldjük el az adatokat soronként.

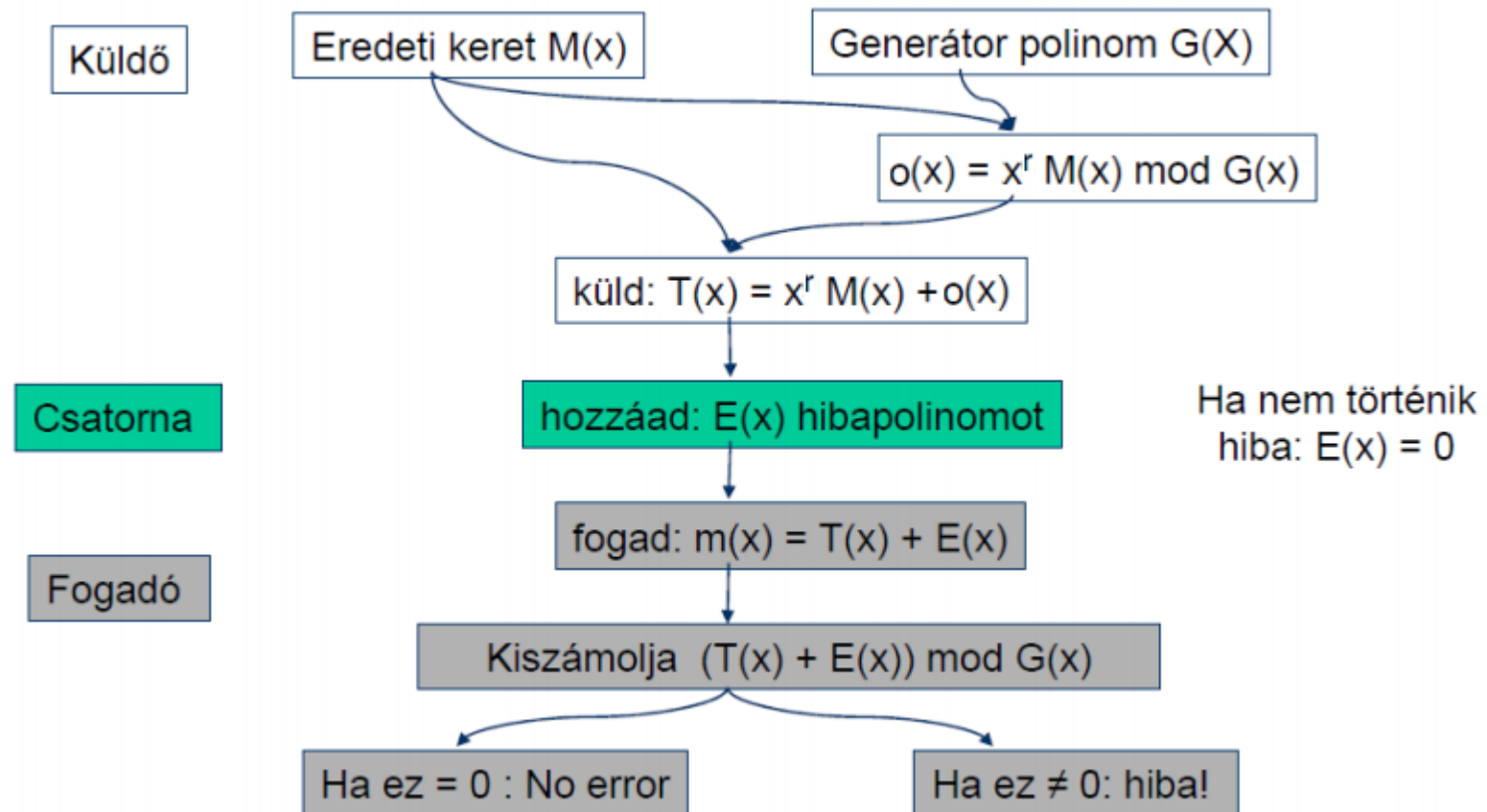
- Példa $k = 2, l = 3$ esetén:

1	0	1
0	1	1
0	0	1

- Hogy viselkedik ez a módszer egyszerű bit-hibák és löketszerű (burst) bit-hibák esetén, ha $k = 3, l = 4$? Milyen hosszú lehet egy bitsorozat, melynek minden bitje hibás, hogy a hibázást meg tudjuk állapítani? (Löketszerű: egymás utáni bitek hibásan jönnek át)
- Egészítsük 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?

CRC hibajelző kód – emlékeztető

- Forrás: Dr. Lukovszki Tamás fóliái alapján



Példa CRC számításra – emlékeztető

- Keret ($M(x)$): 1101011011
- Generátor ($G(x)$): 10011
- Végezzük el a következő maradékos osztást: $\frac{11010110110000}{10011}$
- (A maradék lesz a CRC ellenőrzőösszeg)

$$\begin{array}{r}
 11010110110000 \text{ / } 10011 = 1100001010 \\
 \underline{10011} \\
 10011 \\
 \underline{10011} \\
 0000 \\
 10110 \\
 \underline{10011} \\
 010100 \\
 \underline{10011} \\
 01110
 \end{array}$$

maradék

Feladat 5

- Adva a $G(x) = x^4 + x^3 + x + 1$ generátor polinom.
- Számoljuk ki a
1100 1010 1110 1100 bemenethez a 4-bit CRC ellenőrzőösszeget!
- A fenti üzenet az átvitel során sérül, a vevő adatkapcsolati rétege az
1100 1010 1101 1010 0100 bitsorozatot kapja.
Történt-e olyan hiba az átvitel során, amit a generátor polinommal fel lehet ismerni? Ha nem, akkor ennek mi lehet az oka?

CRC, MD5, SHA1 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())
```

- SHA1

```
import hashlib
test_string= "Fekete retek rettenetes".encode('utf-8')
m = hashlib.sha1()
m.update(test_string)
print(m.hexdigest())
```

Házi feladat

netcopy alkalmazás

Készítsen egy netcopy kliens/szerver alkalmazást, mely egy fájl átvitelét és az átvitt adat ellenőrzését teszi lehetővé CRC vagy MD5 ellenőrzőösszeg segítségével! A feladat során három komponenst/programot kell elkészíteni:

1. Checksum szerver: (fájl azonosító, checksum hossz, checksum, lejárát (mp-ben)) négyesek tárolását és lekérdezését teszi lehetővé. A protokoll részletei a következő oldalon.
2. Netcopy kliens: egy parancssori argumentumban megadott fájlt átküld a szervernek. Az átvitel során/végén kiszámol egy md5 checksumot a fájlra, majd ezt feltölti fájl azonosítóval együtt a Checksum szerverre. A lejárati idő 60 mp. A fájl azonosító egy egész szám, amit szintén parancssori argumentumban kell megadni.
3. Netcopy szerver: Vár, hogy egy kliens csatlakozzon. Csatlakozás után fogadja az átvitt bájtokat és azokat elhelyezi a parancssori argumentumban megadott fájlba. A végén lekéri a Checksum szervertől a fájl azonosítóhoz tartozó md5 checksumot és ellenőrzi az átvitt fájl helyességét, melynek eredményét stdoutputra is kiírja. A fájl azonosító itt is parancssori argumentum kell legyen.

Leadás: A program leadása a TMS rendszeren **.zip** formátumban, amiben egy **checksum_srv.py**, egy **netcopy_cli.py** és egy **netcopy_srv.py** szerepeljen!

Beadási határidő: **TMS rendszerben**

Checksum szerver - TCP

- **Beszúr üzenet**
 - Formátum: szöveges
 - Felépítése: BE|<fájl azon.>|<érvényesség másodpercben>|<checksum hossza bájtszámban>|<checksum bájtjai>
 - A „|” delimiter karakter
 - Példa: BE|1237671|60|12|abcdefabcdef
 - Ez esetben: a fájlazon: 1237671, 60mp az érvényességi idő, 12 bájt a checksum, abcdefabcdef maga a checksum
 - Válasz üzenet: OK
- **Kivesz üzenet**
 - Formátum: szöveges
 - Felépítése: KI|<fájl azon.>
 - A „|” delimiter karakter
 - Példa: KI|1237671
 - Azaz kérjük az 1237671 fájl azonosítóhoz tartozó checksum-ot
 - Válasz üzenet: <checksum hossza bájtszámban>|<checksum bájtjai>
Péda: 12|abcdefabcdef
 - Ha nincs checksum, akkor ezt küldi: 0|
- **Futtatás**
 - `python checksum_srv.py <ip> <port>`
 - <ip> - pl. localhost a szerver címe bindolásnál
 - <port> - ezen a porton lesz elérhető
 - A szerver végtelen ciklusban fut és egyszerre több klienst is ki tud szolgálni. A kommunikáció TCP, csak a fenti üzeneteket kezeli.
 - Lejárat utáni checksumok törlődnek, de elég, ha csak a következő kérésnél ellenőrzi.

Netcopy kliens – TCP alapú

- Működés:
 - Csatlakozik a szerverhez, aminek a címét portját parancssori argumentumban kapja meg.
 - Fájl bájtjainak sorfolytonos átvitele a szervernek.
 - A Checksum szerverrel az ott leírt módon kommunikál.
 - A fájl átvitele és a checksum elhelyezése után bontja a kapcsolatot és terminál.
- Futtatás:
 - `python netcopy_cli.py <srv_ip> <srv_port> <chsum_srv_ip> <chsum_srv_port> <fájl azon> <fájlnév elérési úttal>`
 - <fájl azon>: egész szám
 - <srv_ip> <srv_port>: a netcopy szerver elérhetősége
 - <chsum_srv_ip> <chsum_srv_port>: a Checksum szerver elérhetősége

Netcopy szerver – TCP alapú

- Működés:
 - Bindolja a socketet a parancssori argumentumban megadott címre.
 - Vár egy kliensre.
 - Ha acceptálta, akkor fogadja a fájl bájtjait sorfolytonosan és kiírja a parancssori argumentumban megadott fájlba.
 - Fájlvége jel olvasása esetén lezárja a kapcsolatot és utána ellenőrzi a fájlt a Checksum szerverrel.
 - A Checksum szerverrel az ott leírt módon kommunikál.
 - Hiba esetén a stdout-ra ki kell írni: CSUM CORRUPTED
 - Helyes átvitel esetén az stdout-ra ki kell írni: CSUM OK
 - Fájl fogadása és ellenőrzése után terminál a program.
- Futtatás:
 - `python netcopy_srv.py <srv_ip> <srv_port> <chsum_srv_ip> <chsum_srv_port> <fájl azon> <fájlnév elérési úttal>`
 - <fájl azon>: egész szám ua. mint a kliensnél – ez alapján kéri le a szervertől a checksumot
 - <srv_ip> <srv_port>: a netcopy szerver elérhetősége – bindolásnál kell
 - <chsum_srv_ip> <chsum_srv_port>: a Checksum szerver elérhetősége
 - <fájlnév> : ide írja a kapott bájtokat

VÉGE
KÖSZÖNÖM A FIGYELMET!