Telekommunikációs Hálózatok

6. gyakorlat

ZH időpont!!!

- **HÉTFŐI CSOPORTNAK: november 13.**
- **KEDDI CSOPORTOKNAK: november 7.**
- zárthelyi, a gyakorlat helyén és idejében

Feladat 1

- Készítsünk egy ***kliens***-***proxy***-***szerver*** alkalmazást, ahol:
  - a ***szerver*** egy TCP szerver,
  - a ***proxy*** a ***szerver*** irányába egy TCP kliens, a ***kliens*** irányába egy TCP szerver,
  - a ***kliens*** egy TCP kliens a ***proxy*** irányába
- Folyamat:
  - a ***kliens*** küldje a ‚Hello Server' üzenetet a ***proxy***nak,
  - amely küldje tovább azt a ***szerver***nek,
  - amely válaszolja vissza a ‚Hello Kliens' üzenetet a ***proxy***nak,
  - amely küldje tovább azt a ***kliens***nek

Feladat 2: Egyszerű TCP proxy

Készítsünk egy egyszerű TCP alapú proxyt (átjátszó). A proxy a kliensek felé szerverként látszik, azaz a kliensek csatlakozhatnak hozzá. A proxy a csatlakozás után kapcsolatot nyit egy szerver felé (parancssori argumentum), majd minden a klienstől jövő kérést továbbítja a szerver felé és a szervertől jövő válaszokat pedig a kliens felé.

Pl: python netProxy_gyak6_f2.py szalaigj.web.elte.hu 9000

Webböngészőbe írjuk be: localhost:9000

Feladat 3A

- Készítsünk a számológéphez egy proxy-t (select-tel, több kliens is lehet), ami a klienstől kapott TCP kéréseket az UDP serverhez küldi, majd az eredmény a proxyn keresztül vissza a kliensnek.

Feladat 3B

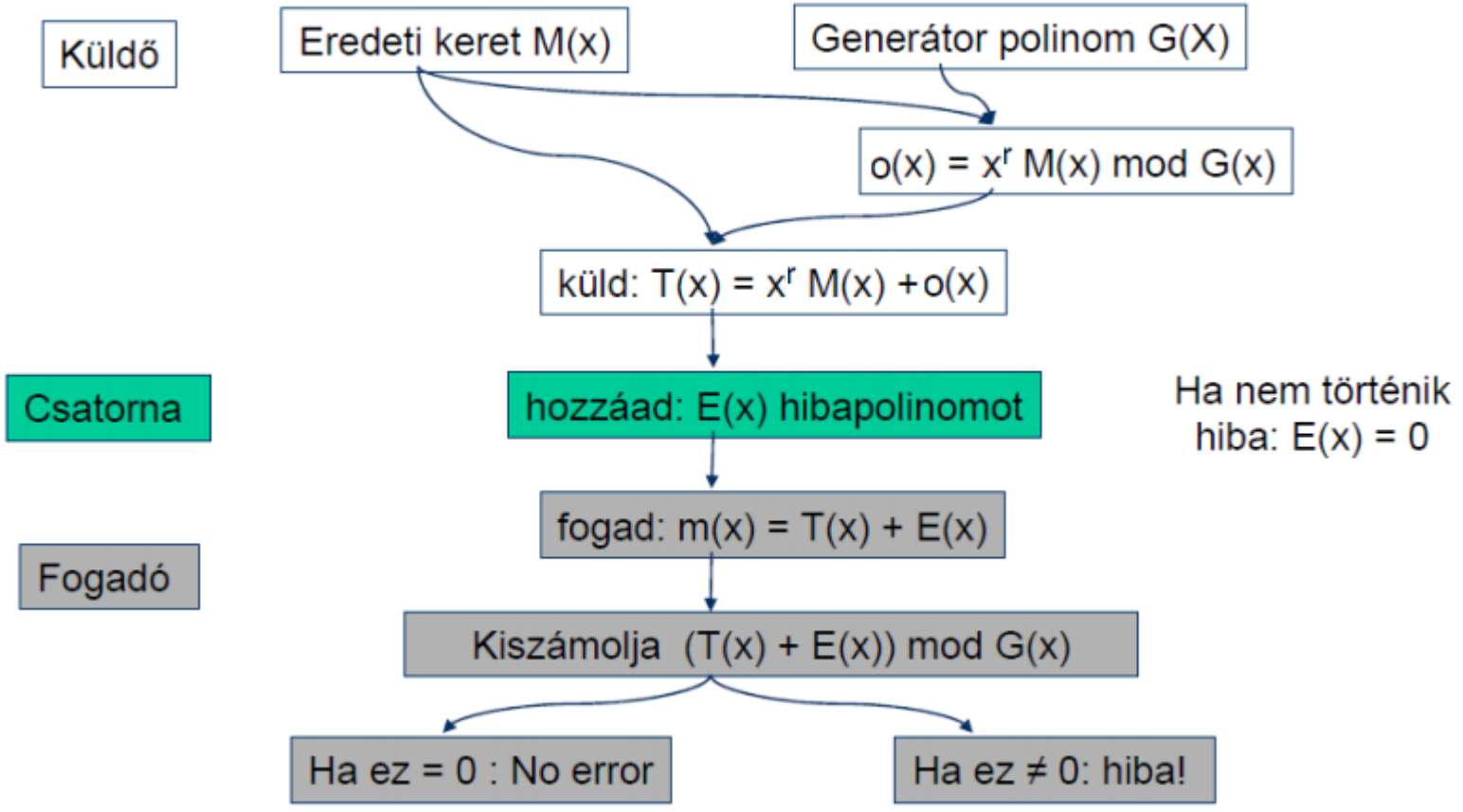
- Egy számológép kliens az UDP szervertől kérje el a TCP-s szerver elérhetőségét!
  - Küldjön egy ‚GET' üzenetet
- A kliens küldjön egy ‚Hello Server' üzenetet a UDP szervernek, aki visszaküldi a TCP szerver elérését, ahova a számokat és az operátort fogja elküldeni.
- A TCP szerver legyen a korábbi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

- Hasznos lehet a null ('\x00') karakter eltűntetése:

```
hostname = b'localhost\x00\x00\x00\x00\x00\x00'
hostname = hostname.replace(b'\x00', b'')
```

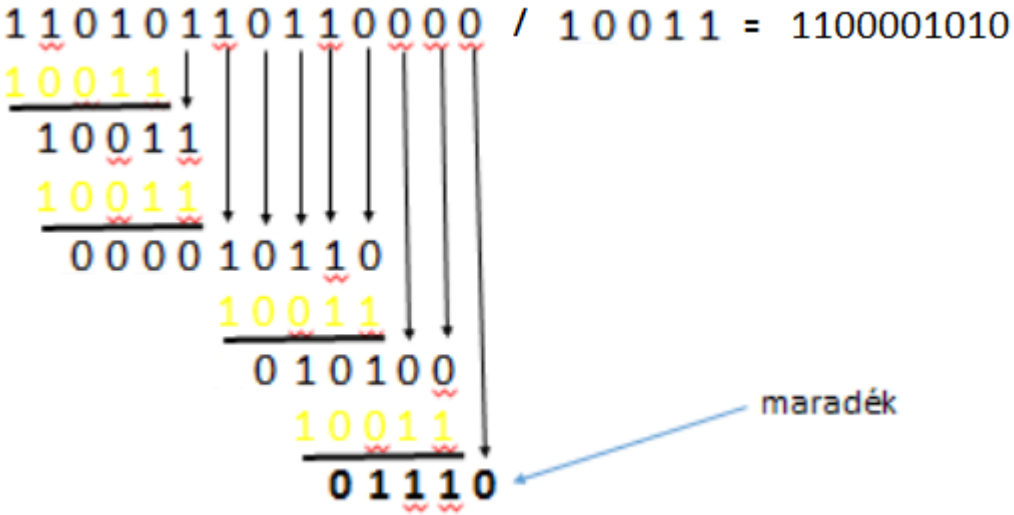
CRC hibajelző kód – emlékeztető

- 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)



Feladat 4

- 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?

Feladat 4 megoldása

- Mivel a generátor polinom foka 4, ezért négy 0-t írunk a bemenet végéhez. A $G(x)$ bináris alakban: 11011 lesz, tehát a $\frac{1100\ 1010\ 1110\ 1100\ 0000}{11011}$ maradékos osztást kell elvégeznünk:

```
1100101011011000000   /   11011
-----------------------------
1100101011011000000
11011
-----------------------------
 0010010111011000000
   11011
-----------------------------
  1001111011000000
   11011
-----------------------------
  100011011000000
   11011
-----------------------------
  10101011000000
   11011
-----------------------------
  1110011000000
   11011
-----------------------------
  011111000000
   11011
-----------------------------
  0100000000
   11011
-----------------------------
  10110000
   11011
-----------------------------
  1101000
   11011
-----------------------------
  000100   →   0100 a CRC ellenőrzőösszeg
```

Feladat 4 megoldása

- Az előbbi számításnál az jött ki, hogy 1100 1010 1110 1100 0100 lenne az a bitsorozat, amelyet a vevő kapna. Ha ebből kivonjuk az alfeladatban megadott sorozatot, az alábbi eredmény jön ki:

$$\begin{array}{rr} & 11001010111011000100 \\ - & 11001010110110100100 \\ \hline & 00000000001101100000 \end{array}$$

- Tehát a két bitsorozat pontosan a generátor polinom többszörösével tér egymástól, amely tehát a hiba polinom. Ezt pedig nem lehet felismerni.

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, lejárat (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!