

Számítógépes Hálózatok GY – 6.hét

Laki Sándor

ELTE-Ericsson Kommunikációs Hálózatok Laboratórium

ELTE IK - Információs Rendszerek Tanszék

lakis@elte.hu

<http://lakis.web.elte.hu>

Teszt – 10 kérdés 10 perc

canvas.elte.hu

Kód:

Házi feladat

Házi feladat

netcopy alkalmazás 4 pont

Készíts 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 komponents/programot kell elkészíteni:

1. Checksum szerver: (fájl azonosító, checksum hossz, checksum, lejárati idő (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.

Beadás: **BE-AD rendszeren keresztül**

Határidő: **2019.04.07. 23:59**

Checksum szerver

- Beszúr üzenet
 - Formátum: szöveges
 - Felépítése: BE | <fájl azon.> | <érvényesség másodpercben> | <checksum hossza bájtzámban> | <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 bájtzámban> | <checksum bájtjai>
Péda: 12 | abcdefabcdef
 - Ha nincs checksum, akkor ezt küldi: 0 |
- Futtatás
 - `.\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.

Netcopy kliens

- 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:

- `.\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

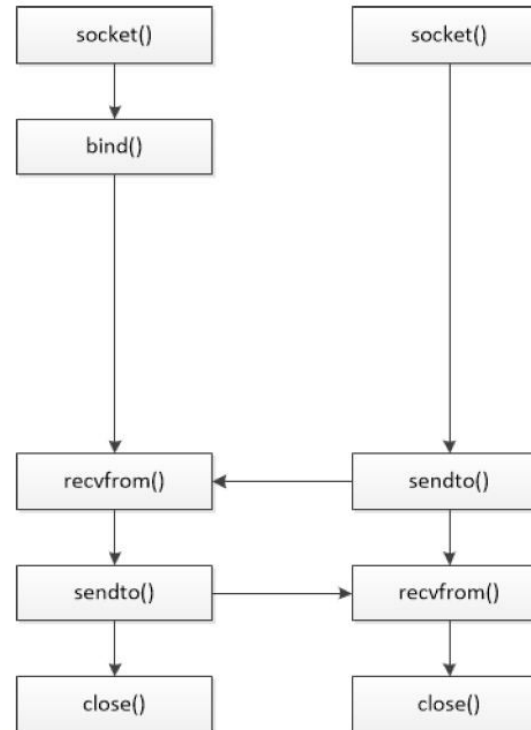
Netcopy szerver

- 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:
 - `.\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

Python socket - UDP

UDP

UDP Server Client



UDP

- socket

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

- recvfrom()

```
data, address = sock.recvfrom(4096)
```

- sendto()

```
sent = sock.sendto(data, address)
```

Feladat 1

Készítsünk egy kliens-szerver alkalmazást, amely UDP protokollt használ. A kliens küldje a „Hello Server” üzenetet a szervernek, amely válaszolja a „Hello Kliens” üzenetet.

Feladat 2 - Számológép UDP felett

Készítsünk egy szerver-kliens alkalmazást, ahol a kliens elküld 2 számot és egy operátort a szervernek, amely kiszámolja és visszaküldi az eredményt. A kliens üzenete legyen struktúra. Használjunk UDP protokollt!

Feladat 3 – fájlátvitel UDP felett

Fájlátvitel megvalósítása úgy, hogy a fájl letöltése UDP felett legyen megoldava. Készüljünk fel arra, hogy az átvitel során csomagvesztés, vagy sorrend csere is történhet! Az UDP szerver portját szabadon definiálhatjuk!

A hibakezeléshez egy javaslat:

Max. 1000 bájtontként UDP csomagokban elkezdjük átküldeni a fájl tartalmát. Minden csomag egy pár bájtos fejléccel indul, amiben jelezzük, hogy az utolsó darab-e, amit átküldtünk, továbbá egy másik mező jelzi a byteoffset-et a fájl elejétől. Működés:

- Ha a kliens kapott egy adatcsomagot, akkor egy nyugtacsomagot küld vissza.
- A nyugtacsomag fogadása után a szerver, küldi a következő adatcsomagot.
- Ha nem jön nyugta, akkor T idő után újraküldi a korábbi adatcsomagot. (pl. $T=200\text{ms}$)
- Ha nyugta veszik el, akkor a vevő az offset alapján el tudja dönteni, hogy egy új adatcsomag, vagy egy korábbi duplikátuma érkezett-e.
- Ha az utolsó csomag is megérkezett, akkor a kliens nyugtázza azt is és lezárja a fájlba írást. A szerver az utolsó nyugta után befejezi az átvitelt.

Vége

Köszönöm a figyelmet!