Ministerul Educației, Culturii și Cercetării Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică Departamentul Ingineria Software și Automatică

RAPORT

la Lucrarea de Laborator Nr. 4

Disciplina: Programarea în Rețea

TEMA: Aplicație Client-Server TCP

A elaborat: st. gr. TI-172, Movileanu Vladislav

A verificat: asistent universitar, Buldumac Oleg

Chișinău 2020

Sarcina lucrării:

Să se creeze o aplicație Client-Server TCP utilizând Sockets API

Pentru nota 8, 9 si 10:

- Să proiectați și elaborați propriul protocol

Atenție:

- Fiecare client trebuie procesat de către server într-un fir de execuție aparte
- Nu se admite aplicații simple de genul Echo Client-Server
- Aplicația poate fi GUI sau consolă.
- Pentru cei ce doresc 8, 9 și 10, ca exemplu luați protocoalele HTTP, SMTP unde clientul și serverul discută printr-un set de reguli bine definite. De exemplu, pentru o aplicație de tip chat pentru ca clientul să fie identificat de către server acesta trebuie să transmită un mesaj de tipul: HELLO-REQUEST după care serverul va răspunde cu HELLO-APPROVE. Mai detailat accesați ultimele 3 referințe de la sfîrșitul acestui fișier.

Raspunsuri la întrebări de control:

- Ce este un protocol orientat pe conexiune?
  Un protocol orientat pe conexiune (COP) este un protocol de rețea utilizat pentru a stabili o sesiune de comunicare a datelor în care dispozitivele finale folosesc protocoale preliminare pentru a stabili conexiuni end-to-end, apoi fluxul de date ulterioare este livrat în modul de transfer secvențial.
- Ce tipuri de aplicații beneficiază în general de utilizarea protocolului TCP?
  TCP este utilizat de multe aplicații web, inclusiv World Wide Web (WWW), e-mail, Protocol de transfer de fișiere, Secure Shell și partajare de fișiere peer-to-peer.
- Cum TCP garantează că datele vor fi transmise cu succes?
  Pentru a garanta livrarea datelor, TCP implementeză scheme de retransmisie pentru datele care pot fi pierdute sau deteriorate.
- Diferența dintre blocking si non-blocking sockets?
  În modul blocking, apelurile API socket, recv, send, connect (doar TCP) și accept (doar TCP) vor bloca la nesfârșit până când acțiunea solicitată a fost efectuată. În modul non-blocking, aceste funcții revin imediat. select se va bloca până când socket-ul este gata.
- Diferența dintre blocking multithreaded și non-blocking single thread socket.
  Un non-blocking single thread socket permite operarea I/O pe un canal fără a bloca

procesele care îl utilizează. Aceasta înseamnă că putem folosi un singur fir pentru a gestiona mai multe conexiuni concurente și pentru a obține o operație de I/O „asincronă de înaltă performanță”.

- Cum are loc procesul TCP Three Way Handshake?
  1) Se stabilește o conexiune între server și client;
  2) Serverul primește pachetul SYN de la nodul client;
  3) Nodul client primește SYN/ACK de la server și răspunde cu un pachet ACK;

- Numiti cele 4 apeluri de sistem necesare pentru a crea un server TCP ?
  1) Folosind create(), Creăm un TCP socket;
  2) Folosind bind(), legăm socket-ul la adresa serverului;
  3) Folosind listen(), introducem socket-ul serverului într-un mod pasiv, apoi acesta așteaptă ca clientul să se apropie de server pentru a realiza o conexiune;
  4) Folosind accept(), conexiunea este stabilită între client și server, suntem gata pentru transferul de date.

- Ce se întîmplă cînd apelați mai întîi **connect()** apoi **bind()**?
  Clientul nu se va conecta la server, fiindcă socketul nu a fost asociat cu o adresă locală.

- Diferența dintre protocoalele fără stare și cele cu stare. Cărui tip îi aparține HTTP?
  Aceste două tipuri de protocoale se disting după cerința ca software-ul server sau serverul să stocheze informații despre stare sau sesiune. ***HTTP*** este un protocol cu stare.

- Ați avea vreodată nevoie să implementați un timeout într-un client sau server care utilizează TCP?
  Da, atunci când un server necesită prea mult timp pentru a răspunde la o solicitare de date făcută de pe un alt dispozitiv.

- Într-o conexiune TCP, clientul sau serverul trimite mai întâi datele?
  În dependență de arhitectura serverului, dar cel mai des clientul trimite mai întâi datele.

- Care este adresa de loopback IPv4 și care este rolul ei?
  O adresă de loopback IPv4(127.0.0.1) este folosită ca mijloc de validare a faptului că placa de rețea fizică conectată local funcționează corect și stiva TCP / IP este instalată.

- De unde știe un sistem de operare ce aplicație este responsabilă pentru un pachet primit din rețea?
  Port source și Port destination sunt utilizate pentru a afla ce aplicație este responsabilă pentru un pachet primit din rețea.

- Datele primite prin **recv()** au întotdeauna aceeași dimensiune cu datele trimise cu **send()**?
  Datele primite prin **recv()** nu au întotdeauna aceeași dimensiune cu datele trimise cu **send()**.

- Este acceptabil să închei executia programului dacă este detectată o eroare de rețea?
  Da!

- Puteți îmbunătăți performanța aplicației prin dezactivarea algoritmului Nagle?
  Dezactivarea algoritmului Nagle nu este foarte eficientă atunci când efectuăm o comunicare unidirecțională. În comunicarea bidirecțională, dezactivarea algoritmului Nagle poate îmbunătăți debitul, deoarece beneficiile eliminării întârzierilor se pot acumula, datorită faptului că fiecare nod poate trimite răspunsurile mai devreme.

- Ce instrumente listează socket-urile TCP deschise în sistemele de operare Windows și Linux?
  Windows - https://www.howtogeek.com/howto/28609/how-can-i-tell-what-is-listening-on-a-tcpip-port-in-windows/
  Linux/MacOS - netsat (https://www.tecmint.com/find-open-ports-in-linux/)

- Tehnici de sincronizare a firelor de execuții.
  Câteva tehnici simple și de bază (https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzahw/rzahwsynco.htm)

Descrierea Aplicației:

Am utilizat limbajul ***GO(golang)***, pentru a crea un chat-server.
Utilizând limbajul ***Swift,*** am realizat o aplicație de tip ***Messenger.***
Am utilizat ***fire de execuție***, ***socket***, ***regex***.
Pentru această aplicație am utilizat API-ul IMGUR, pentru a procesa pozele utilizatorilor.
Am creat 2 protocoale de tipul ***Client-Server*** și 2 protocoale de tipul ***Server-Client***.
Aplicația permite utilizatorului să se alăture unui Chat-Room, utilizând o poză de pe dispozitiv și un nume de utilizator, pentru a comunica cu alți participanți.