Univerzitet u Novom Sadu Fakultet Tehničkih Nauka
Departman za Računarstvo i automatiku

Dušan Stojković RA 140/2019

ABSTRACT

Dokumentacija iz Osnova Računarskih mreža: Sistem poštanskih sandučića

1. Opis zadatka

Realizovati primer namenskog sistema poštanskih sandučića pomoću TCP protokola. Primer se sastoji iz serverske i klijentske strane. Klijent preko tastature može da zadaje sledeće komande:

- Login - Prijava korisnika.
- Logout - Odjava korisnika.
- Send - Slanje poruke u poštanski sandučić
- Check - Provera da li postoji poruka za prijavljenog u poštanskom sandučiću.
- Receive - Prijem poruke iz poštanskog sandučića.

Po prihvatanju zahteva za prijem poruke i nakon slanja iste, serverska strana tu poruku uklanja.

2. Realizacija u C++ programskom jeziku

Osnovna ideja koja leži iza odluke da relaizujemo sam projekat u C++ programskom jeziku je mogućnost upotrebe: naprednih struktura podataka, biblioteka za konkurentno programiranje i realizacija projekta u objektno orijentisanom stilu.

Projekat je poželjno podeliti u više faza izrade kako bi realizacija bila što jasnija. Na taj način znamo tačno gde smo stigli i možemo diskutovati promene u fazama koje nisu implementirane. Svakoj fazi ću dodeliti ime tako da se redni broj faze može videti na osnovu poglavlja.

Smatram za korisnim da se dalje dokumentovanje, u smislu same implenetacije, vodi na **github** jer bi ovde bilo suvišno, ovaj dokument služi da se u kratke crte definiše svaki deo projekta i funkcionalnosti.

2.1. Prevođenje jednostavne *clent-server* arhitekture u C++

Za ovaj deo procesa već raspolažemo C kodom sa labaratorijskih vežbi. Namena tog koda je da uz pomoć TCP protokola pošalje sa klijenta na server poruku koja će se na konzoli servera ispisati. Prvi korak je uspešno kompajlirati izvorni kod sa **g++** komandom.

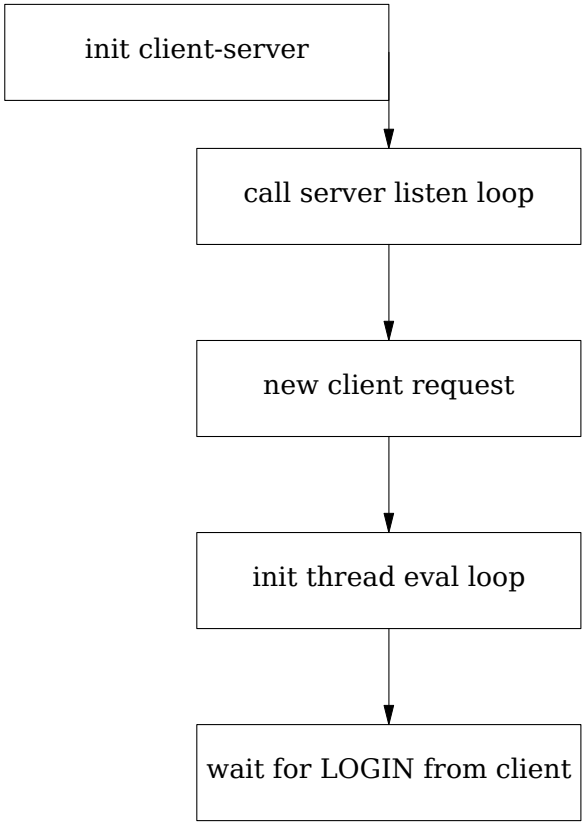
Pošto će se kasnije dodavati novi fajlovi u projekat, poželjno je kreirati **MAKEFILE** koji će kompajlirati ceo projekat *clent-server* arhitekture.

Poželjno će biti da se naprave zasebne klase koje u sebi sadrže metode *clent-server* za uspostavljanje i prekid veze TCP protokola.

2.2. Konkurentno programiranje *clent-server*

Prvi korak ove faze je dodavanje podrške za konkurentno opsluživanje više klijenta na serveru. Ovde ja važno jasno definisati koje funkcije treba opsluživati konkurentno.

Dijagram koji za proizvoljnog klijenta uspostavlja vezu sa serverom je sledeći:



Blok dijagram petlje *server* funkcionalnosti u ovoj fazi

Jednostavna realizacija logovanja korisnika se može postići uz pomoć *.txt* fajla za to će biti dovoljno definisati funkcije čitanja svih korisnika **Registrovanih** na serveru (imati u vidu da je jedno od proširenja mogućnost registracije) i spakovati u listu sve korisnike pri inicijalizaciji *server* strane.

Petlja koja čita dolazne poruke i interpretira ih mora biti definisana delom u jednoj funkciji, ona čita dolazne poruke i čita im tip na osnovu tipa poziva se dalje neka od funkcija koja je deo specifikacije. Ova funkcija interpretira dolazne poruke na osnovu rečnika koji sadrži u sebi sve moguće tipove poruka i funkcije koje su asocirane sa tim porukama. Sam sadržaj poruke se dalje interpetira po specifikaciji pozivom očekivane funkcije.

Konstruisati objekat tipa klijent na serverskoj strani nakon poziva LOGIN komande, do tada mogu se smatrati klijenti povezani na server kao GUEST. Nakon LOGIN komande logično da poziv LOGOUT komande vraća status klijenta sa ulogovan na GUEST.

2.3. Realizacija prijema i slanja poruka

Kada je temelj veze između klijenta i servera uspostavljen, možemo se posvetiti implementaciji Send, Check i Receive funkcija. Server strana mora posedovati listu svih klijenata koji sadrže listu poruka koje su primljene od drugih klijenta. Sama metoda za primanje poruka na *server* strani smešta poruke u tu listu. Kada klijent1 pokrene Check funkciju, otvara se kanal preko koga se sve poruke poslate sa drugih klijnta na server prebrajaju i formatira se kratak opis tih poruka koji se prosleđuje do klijenta1.

Receive funkcija generalno može da se odradi na dva načina:

- Klijent1 traži od servera poruke po njegovom izboru tako što specifikuje neke parametre u RECIEVE funkciji. Primer je primi poruke samo od klijenta2.

- Klijent1 ne specifikuje ništa u RECIEVE funkciji tako da se proslede sve poruke namenjene na serveru do klijenta1.

Naravno, poruke koje se proslede sa servera se moraju i obrisati po specifikaciji.

Dodaću da treba proširiti skup poziva originalne specifikacije jer se naredbe moraju podeliti na *server* side i *client* side.

2.4. Dodatak A: Enkripcija

Ideja je sledeća: dodati clasu **Encrypt** koja bi svaku poruku koja se šalje na server enkriptovala nekom hash funkcijom (SHA256). Ovo je posebno komplikovano kada se sadržaj poruke enkriptuje ali zato je sasvim izvodljivo da se modifikuje LOGIN i LOGOUT tako što se šifre korisnika enkriptuju i dekriptuju na neki jednostavan način.

2.5. Dodatak B: Konkurentan *client*

Proširiti klijenta tako da postoje dve niti jedna za ispis na konzoli, a duga za upis na konzoli, tako da se resorsi ne bi blokirali. Ovo naravno po specifikaciji zadatka nema smisla, jer blokiranje neće da se desi nikad, ali ako se RECIEVE i SEND automatizuju dobili bismo pravi chat klijent nešto što jako želim napraviti.

2.6. Dodatak C: Formatirati poruke u XML zaglavljima

Ideja je jednostavna, serializovati poruke u xml tako što ćemo koristiti stringstream metode (vrlo verovatno postoje funkcije koje rade ovo za nas). Ovo ima dosta smisla raditi jer se time naš program može mnogo lakše skalirati. Ideja je da se sva bitna polja neke klase **Message** formatiraju u string koji liči na xml fajl pa takav string proslediti dalje.