INDUSTRIJSKI KOMUNIKACIONI PROTOKOLI U ELEKTRO-ENERGETSKIM SISTEMIMA

Projekat br 40

Ognjen Sekulić PR29/2019
Aleksandar Ušljebrka PR26/2019

UVOD

1. Opis problema

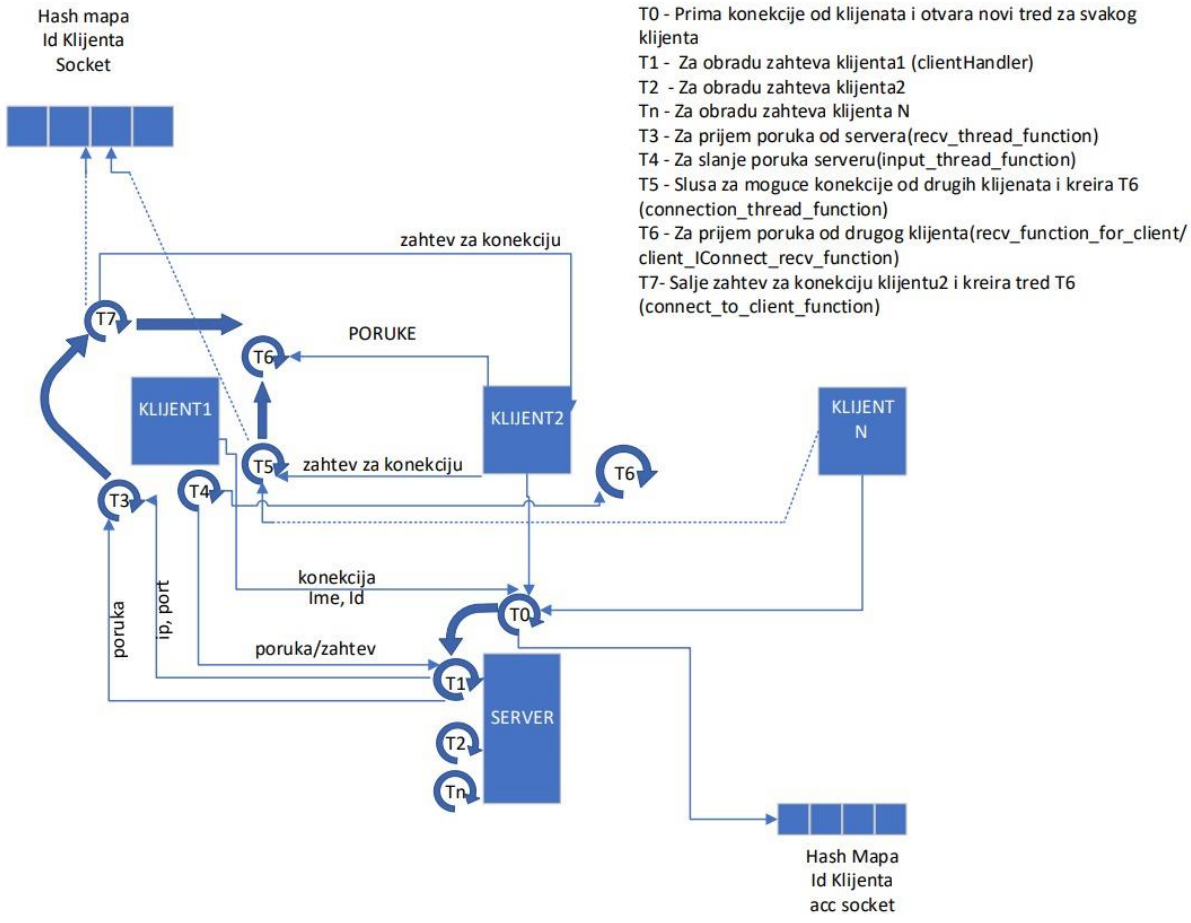
Potrebno je razviti rešenje za komunikaciju klijenata. Sistem se sastoji iz dve osnovne komponente: server i neodređen broj klijenata.
Server treba da omogući korisnicima da se registruju pri čemu oni navode svoj ID, odnosno ime. Takođe, treba da se omogući direktna i komunikacija kroz server. Direktna komunikacija znači da se klijenti povežu posle čega mogu da razmenjuju poruke. Potrebno je implementirati tako da je moguća komunikacija sa više klijenata. Ako je izabrana komunikacija kroz server, klijent serveru pošalje poruku posle čega je on prosledi odgovarajućem klijentu.
Klijenti pokreću komunikaciju sa drugim klijentom tako što šalju serveru zahtev za komunikaciju navodeći ime klijenta sa kim žele da pričaju.

2. Ciljevi zadatka

Kako bi se ispunili zahtevi i da bi se rešio gore pomenuti problem potrebno je uspešno implementirati klijentsku i serversku aplikaciju sa određenim

funkcionalnostima i uspostaviti komunikaciju između njih putem TCP protokola. Pre svega potrebno je napraviti dizajn sistema koji će služiti kao osnov za dalji rad. Takođe je potrebno definisati strukturu, elemente i osnovne funkcionalnosti samog sistema. Same aplikacije je poželjno što je više moguće razdvojiti na .h i .cpp fajlove radi lakšeg upravljanja aplikacijom i bolje preglednosti. Uz sve to trebalo bi voditi računa o upotrebi i oslobađanju memorijskih i procesorski resursa, kao i mehanizmima za sinhronizaciju pristupa.

DIZAJN



U priloženoj slici možemo videti na koji način je zamišljen rad zadatog sistema i na koji način komponente funkcionišu i komuniciraju .

KOMPONENTE

- SERVER – Server je komponenta koja je zadužena za registraciju klijenata, tj prikuplja njihove podatke i skladišti ih u hash-mapu. Hash mapa je izabrana kao odgovarajuća struktura podataka iz razloga što imamo par „ključ-vrednost“ i u slučaju velikog broja klijenata ova struktura se pokazuje kao bolja u odnosu na ostale(npr. lista,niz..). U glavnoj niti serverska aplikacija osluškuje konekcije i pri prihvatanju konekcije pravi novu nit za konektovanog klijenta (ClientHandler) koja je zadužena za obradu zahteva tog klijenta. Ta nit nakon što upiše klijenta u hash mapu, očekuje zahteve od klijenta i u zavisnosti od primljenog zahteva šalje adekvatan odgovor (ili grešku, npr. ukoliko ne postoji traženi klijent itd..). Smatra se da je jedna nit po klijentu dovoljna jer od servera se odmah očekuje odgovor nakon primljenog zahteva pa samim tim nema potrebe za razdvajanjem u send i recv thread. Postoje dva tipa zahteva:

-za direktnu komunikaciju
-za prosleđivanje poruke preko servera
U slučaju prvog zahteva server pretražuje hash mapu i ukoliko pronađe traženog klijenta vraća njegov port i ip kao odgovor, uz napomenu da se ip izvlači iz klijentskog soketa, a listen port je sadržan u poruci za registraciju svakog klijenta.
U drugom slučaju server primljenu poruku prosleđuje klijentu, čije ime je navedeno u zahtevu, nakon što ga pronađe u hash mapi.
Potrebno je navesti da server prepoznaje zahtev na osnovu flag-a na početku poruke koji označava da li je klijent zatražio direktnu komunikaciju ili prosleđivanje. Uz sve to vodi se računa o kritičnim sekcijama, pristupu hash-mapi, i pravilnom upravljanju i oslobađanju memorije.

- KLIJENT – klijent je aplikacija koja se prvobitno registruje na server kako bi mogla da nastavi sa daljim radom. Pri registraciji navodi ime i listen port na kom osluškuje potencijalne konekcije. U slučaju da već postoji klijent sa tim imenom, server mu vraća grešku i ponovo se otvara forma za registraciju. Nakon uspešne registracije klijent kreira 3 glavna threada(console, recvServer i listen).
  - Console thread je zadužen za slanje poruka, tako što na osnovu klijentskog unosa izvlači soket traženog klijenta iz hash mape, ili koristi serverski

soket ukoliko je klijent uneo da želi da komunicira sa serverom(naravno ukoliko postoji klijent sa tim imenom) nakon čega klijent unosi poruku za slanje i to se šalje traženom klijentu. Forma za unos pri slanju zahteva serveru je nešto drugačija, gde klijent treba da unese tip zahteva, ime klijenta i poruku(samo u slučaju da ne zahteva direktnu komunikaciju), naravno sve je odrađeno sa proverama klijentskog unosa koji se obaveštava usled nepravilnog unosa, tako da se on navede na ispravan unos.

-RecvServer thread se koristi za primanje i ispis odgovora, na naše zahteve, od strane servera. U slučaju pristigle poruke od drugog klijenta ona se samo ispisuje, dok u slučaju dobijenog porta i ip adrese otvara se novi connection thread zadužen za uspostavu konekcije sa kljentom čije smo podatke dobili. Nakon što uspostavi konekciju sa klijentom, upisuje ga u svoju hash mapu i otvara recv thread za tog klijenta, koji ispisuje pristigle poruke od tog klijenta i u slučaju prekida konekcije klijent se briše iz hash mape i taj thread se završava.

- Listen thread je zadužen za osluškivanje konekcija od drugih klijenata i nakon prihvatanja(u while petlji) otvara se recv thread za tog klijenta koji takođe prima poruke i ispisuje ih. Listen se radi na portu koji je klijent uneo pri registraciji.

ZAKLJUČAK

U obe aplikacije, odrađen je gracefully shutdown koji zatvara preostale handle-ove, kao i oslobađa objekte iz hash mape. Brišu se kritične sekcije, zatvaraju se svi soketi i radi se clen-up tako da curenja memorije nema i svi alocirani resursi su oslobođeni. Glavne while petlje u threadovima rade dok god se promenljiva „isWorking“ ne postavi na false, što se kod klijenta radi unošenjem „STOP“ u konzolu gde je to naznačeno, a na serverskoj strani unosom bilo kog karaktera.