INDUSTRIJSKI KOMUNIKACIONI PROTOKOLI U ELEKTROENERGETSKIM SISTEMIMA

Publisher – Subscriber Project

Predmetni asistent: Bojan Jelačić

Radili: Mihajlo Džever i Vladimir Hristov

# UVOD

Pre razvoja interneta i komunikacionih mreža generalno, predominatni način informisanja su bile dnevne novine. Novine su se kupovale na ulici ili su bile dostavljane na adresu čitaoca, ako se on, prethodno, pretplatio na njih.

Razvoj interneta i društvenih mreža uslovio je dnevnu štampu da pređe iz fizičkog u digitalni format, kako bi zadržala svoju bazu čitalaca i kreairala profit na alternativan način, kroz reklame i ekskluzivan sadržaj koji se plaća.

Prelazak u digitalni format je zahtevao stvaranje platforme (softvera) koja će omogućiti čitaocima isto ono što im je omogućavala dnevna štampa, čitanje članaka njima od interesa. Takođe, platforma treba da zadovoljava i potrebe pisca članka, odnosno, da omogući pisanje, pregled i isporučavanje istog čitaocima.

Jedan od bitnih ciljeva jeste sprečavanje spamovanja čitaoca sa člancima koji mu nisu od interesa, tj. omogućuvanje kontrole korisniku kroz filtraciju sadržaja pomoću tema na koje je čitaoc pretplaćen.

# DIZAJN

Platforma je realizovana kroz dve konzolne aplikacije, koje međusobno komuniciraju koristeći TCP (Transmission Control Protocol). Server, zadužen za prijem članaka od pisaca i njihovo ispručivanje čitaocima, i klijenta, koji može biti ili pisac ili čitaoc.

## Server

Server je logički podeljen na 4 celine:

1. Dio zadužen za inicijalizaciju i gašenje servera.
2. Dio zadužen za komunikaciju sa piscima.
3. Dio zadužen za komunikaciju sa čitaocima.
4. Dio zadužen za međusobnu interakciju delova zaduženih za komunikaciju sa klijentima.

### Dio zadužen za inicijalizaciju i gašenje servera

Implementiran je u main funkciji programa. Zadužen je za pokretanje ostalih logičkih delova servera. Prilikom zatvaranja programa, obaveštava ostale delove o kraju rada, čeka da se oni uspešno ugase i gasi aplikaciju.

### Dio zadužen za komunikaciju sa piscima

Otvara neblokirajući soket na portu 20000 na kojem očekuje uspostavu konekcije od strane klijentske aplikacije u svojstvu pisca. Prilikom uspostave konekcije, stvara se novi soket preko kojeg se odvija sva komunikacija sa klijentom i isti se smešta u dinamički alociranu listu soketa, u kojoj se nalaze svi konektovani pisci.

Poruke koje, dio zadužen za komunikaciju sa piscima, prima su u formatu članka. Članak se sastoji iz tri dela: ime autora članka, naziva teme i teksta članka. Klijentima je omogućeno slanje članaka na 5 tema: igre, tehnologija, mimovi, slavne ličnosti i sport.

Za svakog klijenta u listi, vrši se provera da li je poslao članak i ako jeste smešta ga u odgovarajući red na osnovu teme članka.

Na kraju rada programa, zadužen je da zatvori konekcije sa svim konektovanim klijentima i da izbriše listu soketa, odnosno da oslobodi svu zauzetu memoriju.

### Dio zadužen za komunikaciju sa čitaocima

Otvara neblokirajući soket na portu 20000 na kojem očekuje uspostavu konekcije od strane klijentske aplikacije u svojstvu čitaoca. Po uspostavi konekcije, stvara novi soket preko kog se odvija dalja komunikacija, a soket smešta u rečnik pretplaćenih korisnika, odvojenih po temi.

Komunikacija je dvosmerna. Korisnici šalju teme na koje se pretplaćuju, a server vraća članke koji pripadaju temama na koje je korisnik pretplaćen.

Prilikom gašenja, ovaj deo je zadužen za prekid komunikacije sa čitaocima.

### Dio zadužen za međusobnu interakciju delova zaduženih za komunikaciju sa klijentima

Interakcija između delova odgovornih za komunikaciju sa klijentima omogućena je upotrebom rečnika, semafora i kritičnih sekcija.

Server koristi rečnike u koje se podaci upisuju nakon što stigne poruka od klijenata i iz kojih se podaci čitaju prilikom slanja poruka ka klijentu.

Uloga semafora je da onemoguće slanje članaka ukoliko nijedan pisac nije objavio članak na tu temu. Za svaku temu postoji jedan semafor.

Kritične sekcije sprečavaju istovremene izmene redova članaka, lista soketa čitaoca i promenljive koja se koristi za obustavljanje rada servera (stopWork). Za svaku temu, postoje dve kritične sekcije, jedna za red i jedna za listu. Za promenljivu stopWork postoji jedna kritična sekcija.

## Klijent

Klijentska aplikacija ima 2 režima rada koji korisnik bira prilikom paljenja aplikacije:

1. Pisac
2. Čitalac

### Pisac

U režimu „Pisac“ prvo se od korisnika zahteva da unese korisničko ime, nakon čega mu se prikazuje meni i on može da odabere da objavi članak ili da obustavi korišćenje aplikacije. Ako odabere da objavi članak, bira jednu od pet dostupnih tema. Nakon toga, unosi proizvoljnu poruku koja se prosleđuje na server i vraća se nazad na meni.

### Čitalac

U režimu „Čitalac“ korisniku se prilazuje meni sa tri mogućnosti gde treća opcija omogućava prekidanje rada programa.

Prva mu omogućava da se pretplati na temu po njegovom izboru. Nakon što korisnik odabere željenu temu, ona se prosleđuje na server i smešta se u lokalnu listu koja služi za sprečavanje korisnika da se više puta pretplati na istu temu.

Druga opcija dozvoljava korisniku da izlista sve primljene članke. Korisnik sve vreme prihvata članke od servera i oni se smeštaju u drugu lokalnu listu koja se štampa izborom druge opcije. Na konzoli će biti ispisano svaki put kada korisnik primi novi članak, nezavisno od menija u kom se trenutno nalazi. Za svaki članak biće ispisana tema, poruka i autor. Nakon izlistavanja članaka, oni se brišu a lista se prazni.

### Obustavljanje rada aplikacije

Izborom poslednje opcije u oba režima, zatvaranje aplikacije, gasi se tred za prijem poruka, oslobađa se dinamički alocirana memorija i zatvara se soket. Ako u bilo kom trenutku klijentska aplikacija primi informaciju da je server ugašen, korisnik će biti obavešten o tome. Nakon povratka u glavni meni(ako se korisnik već ne nalazi u njemu) bilo koja opcija pored izlistavanja članaka u režimu čitaoca, dovešće do gašenja aplikacije.

# STRUKTURE PODATAKA

## Rečnik

Ova struktura podataka je bila pogodna za čuvanje drugih struktura kojima su svi elementi bili povezani sa zajedničkom osobinom. U slučaju ovog projekta, to je bila tema. Na serveru postoje 2 rečnika.

U jednom se čuvaju redovi članaka koje su objavili pisci. Kada pisac objavi članak, proverava se njegova tema. Ona istovremeno predstavlja i ključ rečnika, odnosno članak će biti smešten u red koji se nalazi u paru ključ-vrednost sa ključem koji je jednak temi članka.

Drugi rečnik sadrži liste soketa. Kada se čitalac pretplaćuje na određenu temu, šalje poruku serveru koja sadrži odabranu temu. Koristeći dobijenu informaciju kao ključ, soket čitaoca se dodaje u na listu soketa koja predstavlja vrednost para ključ-vrednost.

U oba rečnika može biti maksimalano pet parova ključ-vrednost. Dalja upotreba vrednosti rečnika biće obrazložena u naredna dva poglavlja.

## Red

Red je pogodan kada treba pristupati isključivo prvom i poslednjem elementu i kada nam ne treba čuvanje u strukturi nakon nakon što je element strukture iskorišćen.

U projektu se artikli čuvaju u redovima. Po prijemu, artikli se smeštaju na kraj odgovarajućeg red i taj proces se odvija sve dok se minimalno jedan čitalac ne pretplati na temu kojoj odgovaraju elementi u redu. Tada se, jedan po jedan, preuzimaju sa početka reda i prosledjuju svim korisnicima koji su pretplaćeni na tu temu, a novi artikli se konstantno dodaju na kraj. Kritičnim sekcijama je ograničeno da artika ne može istovremeno da se dodaje u red i da se preuzima iz reda.

## Lista

Lista je pogodna za dugotrajno čuvanje podataka i višestruku manipulaciju nad njima. Za razliku od reda, iskorišćeni podaci mogu da ostanu u listi i da se kasnije koriste opet. Takođe, nije ograničena na dodavanje elemenata na kraj i preuzimanje sa početka, nego se i jedna i druga operacija se mogu izvršavati na proizvoljnim pozicijama.

Korišćene su za čuvanje više tipova podataka, u obe konzolne aplikacije. Na klijentskoj aplikaciji liste se koriste za čuvanje tema na koje se korinsik pretplatio i za čuvanje članaka koji pristižu sa servera, koji pripadaju temama koje se čuvaju u listi tema. Po ispisu članaka, svi se brišu i oslobađa se memorija koju je lista zauzimala.

Na serverskoj aplikaciji, u listama se čuvaju klijentski soketi. Kada server prepozna novog korisnika, njegov soket se, u slučaju pisaca, smešta u zasebnu listu. Nakon toga, članci koje pisci objavljuju na server stižu preko soketa koji se nalaze u toj listi. U slučaju čitalaca soketi se smeštaju u liste koje se nalaze u jednom od rečnika. Kako postoji posebna lista za svaku temu, kada stigne novi članak od čitaoca, ako odgovarajuća lista nije prazna, server na svaki soket prosleđuje primljeni članak. Ukoliko dođe do greške prilikom slanja ili klijent ugasi svoju aplikaciju, soket se izbacuje iz liste.

# REZULTATI TESTIRANJA

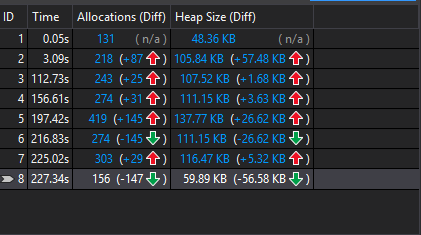
Na projektu su sprovedena tri testa:

1. Podizanje 3 pisca i 3 čitaoca. Čitaoci trebaju da budu pretplaćeni na sledeće načine: 1 na 5 tema, 1 na 3 teme i 1 na 1 temu. Pisci na svakih 100 ms šalju 500 nasumičnih poruka.
2. Podizanje 6 pisaca i 4 čitaoca. Čitaoci trebaju da budu pretplaćeni na sledeće načine: 1 na 5 tema, 1 na 3 teme, 1 na 2 teme i 1 na 1 temu. Pisci na svakih 100 ms šalju 1000 nasumičnih poruka.
3. Podizanje 9 pisaca i 4 čitaoca. Čitaoci trebaju da budu pretplaćeni na sledeće načine: 1 na 5 tema, 1 na 3 teme, 1 na 2 teme i 1 na 1 temu. Pisci na svakih 100 ms šalju 1500 nasumičnih poruka.

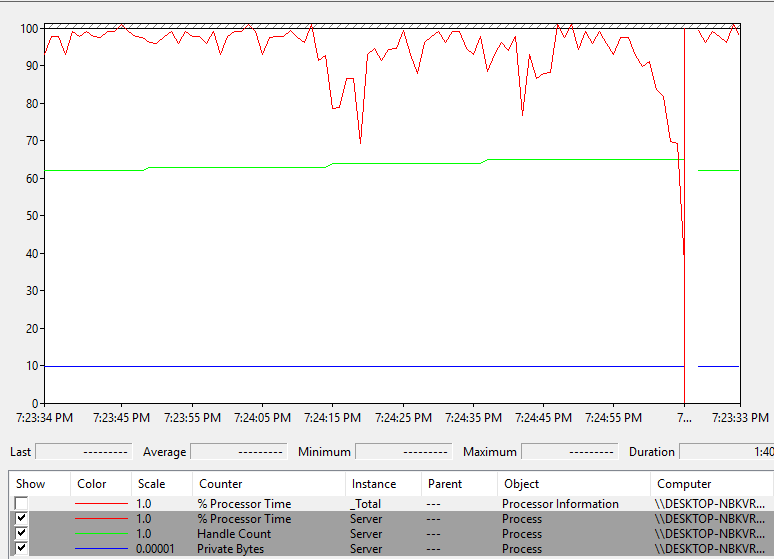
U svim testovima korišćena je standardna klijentska aplikacija za čitaoce i standardna serverska aplikacija. Za pisce je korišćena modifikovana verzija klijentske aplikacije koja nakon paljenja omogućava korisniku da odabere koliko poruka (500, 1000, 1500) želi da šalje na svakih 100 ms. Kada korisnik odabere broj poruka, generiše se onoliko članaka koliko je korisnik odabrao. Generisane poruke se potom šalju na server, odakle se prosleđuju čitaocima.

Server je podizan na jednom računaru dok su klijenti pokretani na drugom. Računari su bili povezani preko lokalne mreže.

## Rezultati prvog testa

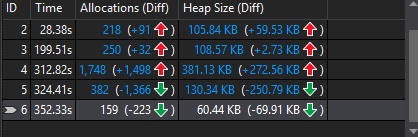


Hip profil 1

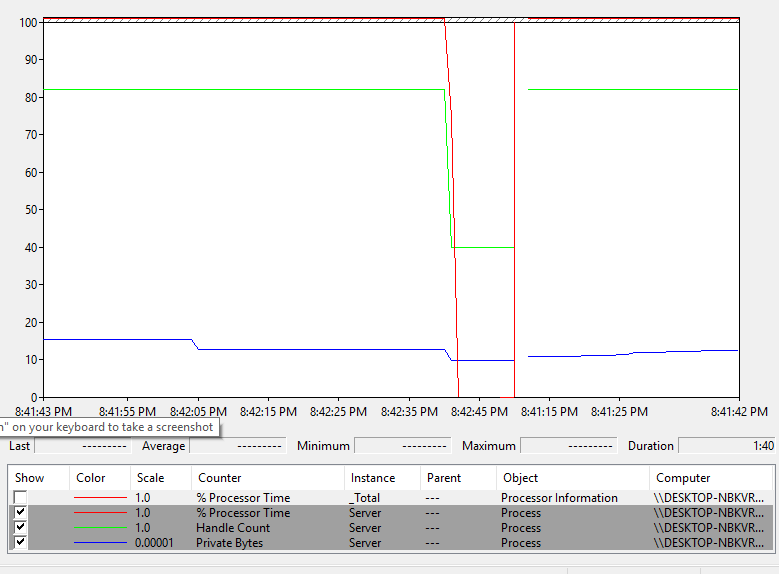


Monitor performansi 1 - Nakon konektovanja čitaoca

## Rezultati drugog testa

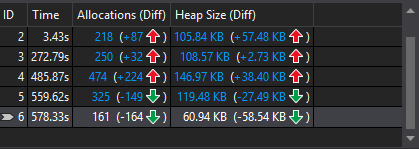


Hip profil 2

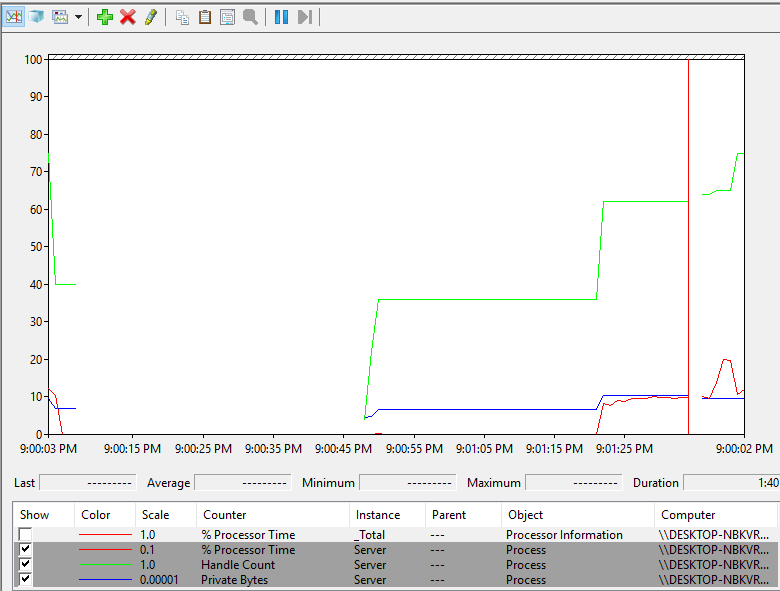


Monitor performansi 2 - Gašenje aplikacije

## Rezultati treceg testa



Hip profil 3



Monitor performansi 3 - Početal rada servera

# ZAKLJUČAK

Iz priloženih rezultata možemo da uočimo da je program dobro optimizovan. Uprkos velikom broju poruka koje potiču od velikog broja pisaca, server funkcioniše bez ikakvih grešaka. Svi članci stižu do čitaoca, sva alocirana memorija se uspešno oslobađa, a svi tredovi se uspešno gase. Server radi pod mnogo većim opterećenjem u poređenju sa redovnom upotrebom.

## Potencijalno unapređenje

Server bi dodatno mogao da se ubrza razdvajanjem ako bi se razdvojio deo za komunikaciju sa piscima na dva različita treda gde bi jedan bi zadužen za očekivanje novih konekcija dok bi drugi bio odgovoran za smeštanje primljenih članaka u redove.