

Industrijski komunikacioni protokoli u EE sistemima

Projekat 9

Đorđe Kojić PR8/2019

Stefan Olabina PR11/2019

1.Uvod

1.1 Opis problema

Potrebno je napraviti klijent-server aplikaciju u C programskom jeziku koja omogućava klijent-server komunikaciju sa ciljem da obezbedi „Pub-Sub“ način rada i komunikacije. Ovo podrazumeva da publisher(1) objavljuje određeni tekst na datu temu, a subscriber(2) koji bi u ovom slučaju predstavljao klijentsku aplikaciju i imao mogućnost prijavljivanja na datu temu posredstvom neke serverske(3) aplikacije i mehanizmima koju ta aplikacija nudi.

Ovi mehanizmi bi podrazumevali rukovanje sa podacima koji pristižu sa publisher-a i subscriber-a i njihovom adekvatnom obradom u cilju uspostavljanja učinkovite komunikacije. Takođe aplikacija bi trebala da podrži rad sa većim brojem klijenata odnosno subscriber-a koji bi mogli da se prijavljuju na određene teme.

1.2 Ciljevi zadatka

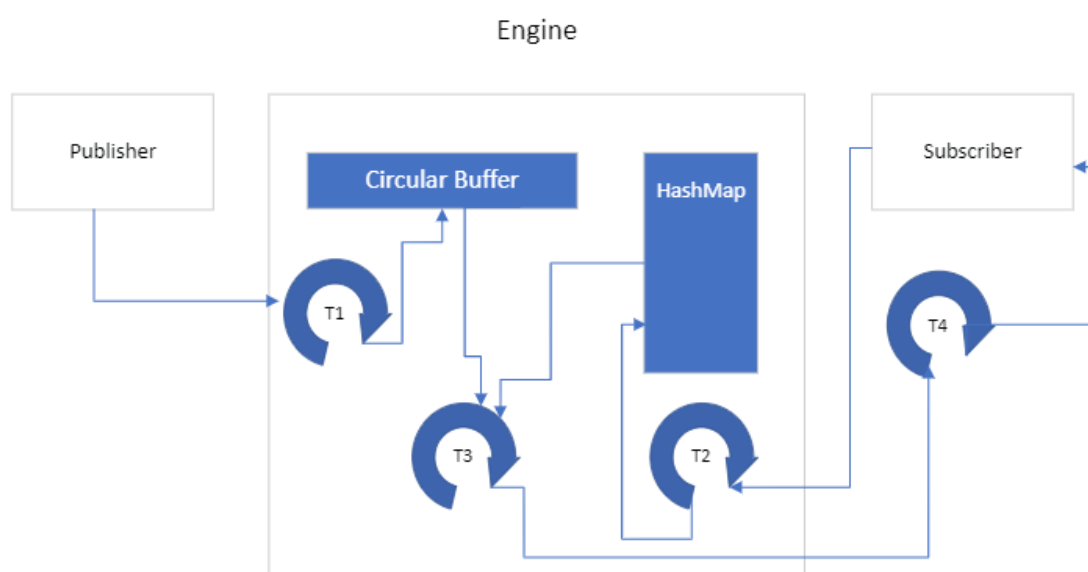
Kako bismo ispunili potrebne zahteve opisane u poglavlju 1.1 neophodno je obezbediti određene funkcionalnosti za sve 3 aplikacije. Naime najpre je neophodno definisati strukturu datog sistema i napraviti određeni prototip rešenja. Pobrnuti se da sve metode potrebne za realizaciju ovih aplikacija budu dokumentovane i izdvojene u posebne .cpp(source) i .h(header) fajlove, takođe potrebno je voditi računa o racionalnoj upotrebi memorijskih i procesorskih resursa kao i rukovanju određenim ugrađenim mehanizmima poput kritičkih sekcija i niti...

Potrebno je komunikaciju između gore pomenutih komponenti realizovati uz pomoć protokola koji se nalaze na transportnom sloju protokola ISO/OSI modela takozvanih UDP i TCP protokola.

U zavisnosti od potreba komunikacije koristiće se jedan, odnosno drugi protokol.

2. Dizajn rešenja

Za realizaciju potrebnog rešenja poželjno bi bilo najpre implementirati model servera odnosno serverske aplikacije, koja će biti zadužena za posrednu komunikaciju između publisher-a i subscriber-a. Naime server ima zadatak da primi podatke od publisher-a koji između ostalog oduhvataju tekstualni sadržaj na datu temu kao i samu temu, koja će se skladištiti u strukturi podataka kružnog buffer-a. Sa druge strane subscriber će imati mogućnost prijavljivanja na datu temu te će on svoje podatke koji obuhvataju informacije na koju temu se on prijavio proslediti serveru koji će te podatke skladištiti u HashMapu.



Slika 1

Na *Slika 1* prikazan je dizajn aplikacije te se iz priloženog može videti da server takođe mora da obezbedi dvosmernu komunikaciju sa subscriber-om, odnosno neophodno je da nakon gore opisanog postupka server pošalje odgovor subscriber-u koji je prijavljen na datu temu u vidu tekstualnog sadržaja pristiglog od publisher-a.

2.1 Engine(server)

Kao što je prethodno navedeno i kao što se može videti sa Slika 1, server će biti zadužen za komunikaciju sa publisher-om i subscriber-om.

Kako bi se komunikacija uspešno odvijala bez štetnog preplitanja i narušavanja konzistencije same aplikacije na serversoj strani će figurisati 3 niti koje će biti zadužene za komunikaciju sa preostale 2 aplikacije.

- Na osnovu slike možemo uočiti niti T1(PubTCP), T2(SubTCP) i T3(SubUDP).

Kao što im nazivi govore svaka nit će biti zadužena za komunikaciju sa odgovarajućom komponentom uz pomoć naznačenih protokola.

- ❖ T1(PubTCP) – nit koja je zadužena za prenos tekstualnog sadržaja kao i naziv odgovarajuće teme, od publisher-a do servera, gde ona smešta podatke u kružni buffer. Komunikacija je realizovana preko TCP protokola. Strukture podataka koje su korišćene za relizaciju ovih funkcionalnosti su: **circular_buffer** i struktura **topic_and_text**, podaci koji pristižu od strane publisher-a se smeštaju u **topic_and_text**, a zatim se spremaju u kružni buffer, za obezbeđivanje konzistencije se koristi kritična sekcija, čime sprečavamo štetno preplitanje i eventualne bug-ove.

- ❖ T2(SubTCP) – nit koja je zadužena za prijem informacija odnosno podataka na koju temu se subscriber prijavio kao i informaciju o njegovom UDP portu, koji će biti neophodan kako bismo imali informaciju koji je subscriber prijavljen na određenu temu. Upravo taj UDP port će biti neophodan za realizaciju dvosmerne komunikacije subscriber-a i servera. Strukture podataka koje se koriste su: **port_and_topic** i **hash_map** čija je adresa prosleđena prilikom kreiranja niti. Poslati podaci se smeštaju u hash_map-u. Protokol po kojem se odvija komunikacija je TCP.
- ❖ T3(SubUDP) – omogućava obradu podataka i njihov prenos dok subscriber-a. Naime data nit izvršava funkcionalnosti koje omogućavaju pronalazak UDP portova onih subscibera koji su se prijavili baš na određenu temu i omogućava njihov prenos do subscriber-a koji će upotrebom niti T4 prihvatiti i ispisati na konzolu sadržaj teksta na datu temu koji je poslat od publisher-a. Strukture podataka: **hash_map**, **circular_buffer** i **topic_and_text**.
Upoređivanjem tema iz kružnog buffera i hash mape lako je utvrđeno koji subscriberi su prijavljeni na određenu temu, zatim su iz hash mape uzeti UDP portovi na koje će biti poslat tekst o datoj temi.

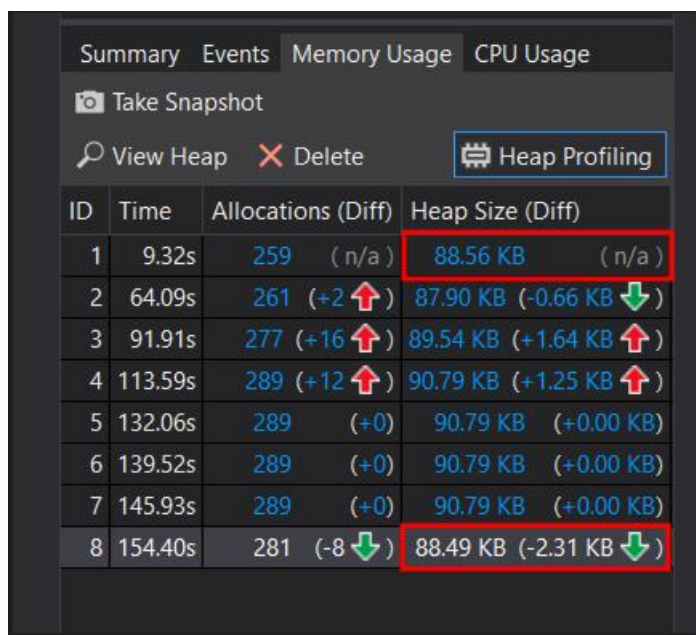
2.2 Publisher

Publisher je komponenta koja je zadužena za slanje tekstualnog sadržaja o datoj temi. On će komunicirati samo sa serverom gde će se njegovi podaci smeštati u kružni buffer. Za njegovu realizaciju koristili smo strukturu podataka **TCPMessage** u koju smo smeštali temu i sam tekst. Za komunikaciju sa serverom je korišćen TCP protokol te uz pomoć **send** funkcije podaci su prosleđeni serverskoj strani.

2.3 Subscriber

Subscriber je komponenta kojoj je omogućeno prijavljivanje na određene teme. Ona komunicira sa serverom preko TCP i UDP protokola. Prilikom slanja teme koristi TCP, a tekst o prijavljenim temama čeka na UDP portu koji je predhodno generisan i prosleđen serveru. UDP socket se pokreće na zasebnoj niti koja izvršava funkcionalnost čekanja i prihvatanja poruka pristiglih od servera.

3. Zaključak



ID	Time	Allocations (Diff)	Heap Size (Diff)
1	9.32s	259 (n/a)	88.56 KB (n/a)
2	64.09s	261 (+2 ↑)	87.90 KB (-0.66 KB ↓)
3	91.91s	277 (+16 ↑)	89.54 KB (+1.64 KB ↑)
4	113.59s	289 (+12 ↑)	90.79 KB (+1.25 KB ↑)
5	132.06s	289 (+0)	90.79 KB (+0.00 KB)
6	139.52s	289 (+0)	90.79 KB (+0.00 KB)
7	145.93s	289 (+0)	90.79 KB (+0.00 KB)
8	154.40s	281 (-8 ↓)	88.49 KB (-2.31 KB ↓)

Zaključno iz priložene slike možemo videti da je količina memorije koja je zauzeta u velikoj meri i vraćena sistemu, te se može govoriti o dobrom rukovanju memorijskim resursima.

Poštovana su pravila prilikom dinamičkog alociranja i otpuštanja memorije.