PUBLISH-SUBSCRIBER APPLICATION

Dokumentacija projekta iz predmeta

Industrijski komunikacioni protokoli u elektroenergetskim sistemima

školska 2019./2020. godina

Studenti

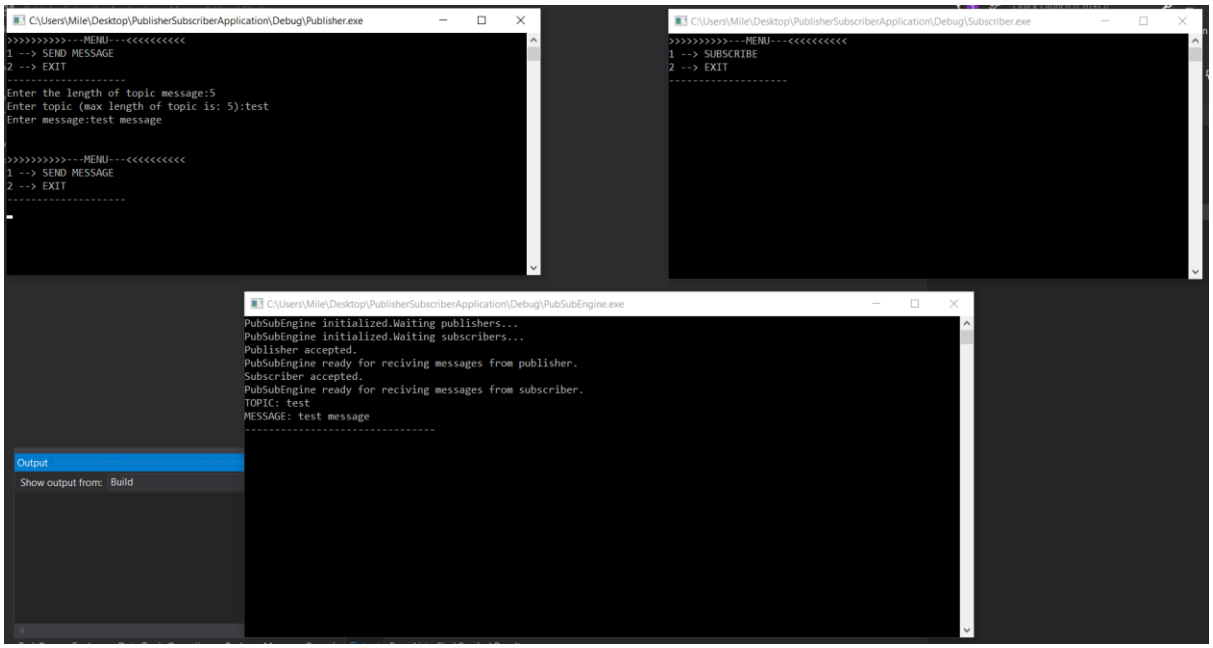
Mile Kajtez, PR 55/2016
Miloš Bakmaz, PR 46/2016

SADRŽAJ

1. UVOD—O APLIKACIJI

Publish-Subscriber Application je aplikacija koja služi za prenos podataka između više različitih terminala, kroz mrežu. Aplikacija ima tri učesnika: Publisher, Subscriber i Server. Cilj aplikacije je slanje poruka od strane Publisher-a i prihvat poruka na strani Subscriber-a. Publisher šalje ime teme i tekst o toj temi, dok Subsriber šalje ime teme na za koju želi da dobija poruke (pretplaćuje se na određenu temu), a prima tekstove vezane za temu na koju se pretplatio. Treća komponenta je Server koji je posrednik u komunikaciji između Publisher-a i Subsriber-a i koji obrađuje podatke koji su stigli sa obe strane i po potrebi ih šalje. Pored posredovanja u komunikaciji, cilj servera je da osigura neposrednu vezu između Publisher-a i Subsriber-a, kao i čuvanje svih podataka koji se razmenjuju.

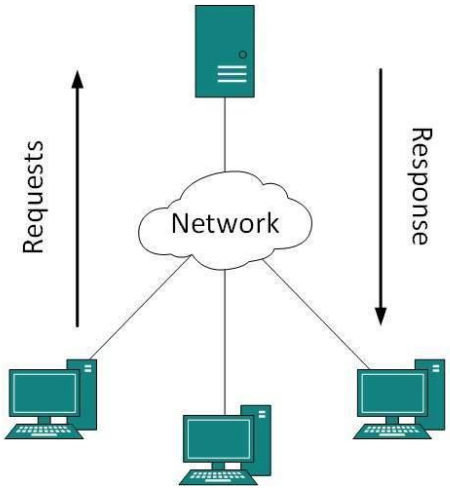
Slika 1.1 predstavlja izgled same aplikacije u režimu rada (na jednom računaru,sa jednim Publisher-om i jedni Subscriber-om).



Slika 1.1 Publisher-Subsriber Application.

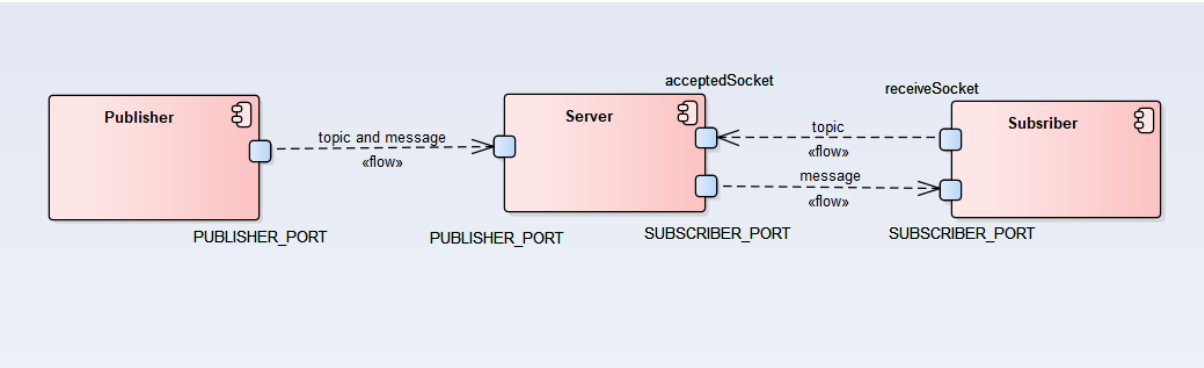
2. DIZAJN APLIKACIJE

Model koji je zastupljen u samom dizajnu aplikacije je Client-Server model (Slika 2.1) čije su karakteristike da centralni deo predstavlja server koji vrši sva izračunavanja i logiku, dok klijenti (u ovom slučaju Publisher i Subscriber) koriste usluge servera. Sama komunikacija između komponenti je implementirana uz pomoć TCP protokola (protokol transportnog sloja), što znači da aplikacija ima siguran prenos podataka kroz mrežu, kao i zanavljanje poruka u slučaju neuspešnog slanja. Takođe, aplikacija može da se koristi na jednom ili više računara.



Slika 2.1 Client-Server model.

Slika 2.2 predstavlja dijagram komponenti na kome je prikazan odnos svih komponenti aplikacije.



Slika 2.2 Dijagram komponenti aplikacije

Pri startovanju aplikacije, prvo je potrebno pokrenuti server (**PubSubEngine**), koji će potom da čeka klijente na određnim portovima. Zatim se pokreće prozivoljan broj Publisher-a i Subscriber-a, koji Server-u šalju zahtev za uspostavu veze. Nakon što uspostave vezu sa serverom klijenti imaju određene opcije i to:

- **Publisher** ima opciju slanja topic-a i poruke ("**SEND MESSAGE**") i opciju njegovog gašenja ("**EXIT**") (Slika 2.3).
- **Subscriber** ima opciju slanja topic-a na koji želi da se pretplati ("**SUBCRIBE**") i opciju njegovog gašenja ("**EXIT**") (Slika 2.4).

Server pri pokretanju ispiše poruke da je spreman za prihvatanje klijenata ("**PubSubEngine initialized.Waiting publishers...**" i "**PubSubEngine initialized.Waiting subscribers...**"). Ove dve poruke potvrđuju da se server pravilno pokrenuo. Kada se klijent poveže na server, ispiše da se klijent uspešno povezao (npr: "**Publisher accepted.**") (Slika 2.5).

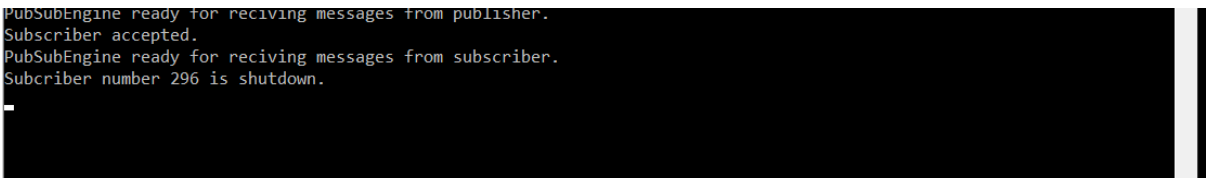
Slika 2.3 Početak rada Publisher-a

Slika 2.4 Početak rada Subscriber-a

Slika 2.5 Početak rada Server-a

U daljem radu aplikacije, Publisher šalje topic-e i poruke serveru, dok se Subsriber pretplaćuje na topic-e. Kada se Subscriber pretplati na postojeći topic, server mu na svakih 5 sekundi šalje poruke koje su vezane za topic na koji je pretplaćen.

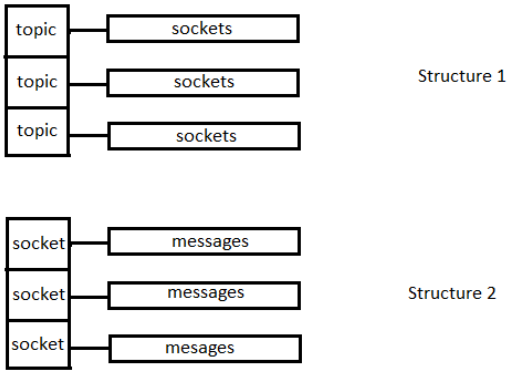
Ukoliko se jedan od klijenata isključi, na serveru će biti ispisan koji klijent se isključio. (Slika 2.6)



Slika 2.5 Rad servera u slučaju gašenja određnog klijenta

2.1. SMEŠTANJE I ČUVANJE PODATAKA NA SERVERU

Jedina kompeoneta koja čuva i obrađuje podatke je server. Podaci se čuvaju u dve odvojene strukture. Sama struktura predstavlja jednostruku spregnutu listu gde svaki element liste pokazuje na queue (koji je takođe predstavljen sa jednostrukom spregnutom listom). U pravoj strukturi imamo listu topic-a, gde svaki topic ima listu socket-a tj. Subscriber-a koji su se pretplatili na njega. U dugoj strukturi imamo listu socket-a tj. Subscriber-a gde svaki Subscriber ima queue poruka koje server treba da mu pošalje. Slika 2.6 predstavlja skicu struktura koje se koriste.



Slika 2.6 Skica korišćenjih struktura

Razlozi korišćenja ovog tipa strukture su sledeći:

1) **Lako smeštanje podataka** – dodavanje novog elementa uvek ide na kraj liste
2) **Laka paralelizacija slanja poruka na Subscriber** – zbog toga što svaki socket ima sve poruke koje mu trebaju biti poslate, samo slanje se svodi na nalaženje socket-a određenog Subscriber-a u strukturi 2 i iščitavanje svih poruka iz queue-a nađenog socket-a
3) **Laka pretraga**
4) **Lako brisanje elemenata iz queue-a poruka** (uvek uzimamo prvu poruku – **FIFO**)

Mane korišenja ovog tipa strukture:

1) **Dupliranje poruka u strukturi 2** – svaki socket (Subscriber) ima sve poruke koje server treba da mu pošalje, međutim ukoliko je više Subscriber-a prijavljeno na jedan topic, svaki će imati sve poruke sa tog topic-a, što znači da su poruke duplirane onoliko puta koliko je socket-a prijavljeno na taj topic. **Optimizacija ovog rešenja** bi bila uvođenje pomoćne strukture (queue) koja bi imala sve poruke jednog topic-a i odakle bi svaka poruka iz queue-a bila slata svim Subscriber-ima koji su pretplaćeni na topic. Kada se sve poruke iz queue-a pošalju svim Subscriber-ima, queue se briše.
2) **Korišćenje niti (threads) umesto procesa** – korišćenjem niti dobijamo konkurentan rad u aplikaciji. Niti se bore za resurse i ukoliko je jedna nit zauzeta resurse, ostale čekaju oslobađanje resursa, što usporava rad aplikacije. **Ukoliko bi se upotrebili procesi**, svaki proces bi imao svoju memoriju i ne bi bilo čekanja na izvršenje. Jedino čekanje bi bilo čekanje da se svi procesi izvrše (join). Korišćenje procesa bi dodatno ubrzalo rad aplikacije.

2.2. PRIMER RADA APLIKACIJE

U ovom poglavlju biće opisan jedan primer rada aplikacije (sa jednim Publisher-om i Subscriber-om). Primer rada se sastoji iz sledećih koraka:

1) Pokretanje aplikacije
2) Publisher šalje topic i poruku na server
3) Smeštanje podataka u strukturu 1
4) Subscriber šalje topic na server
5) Smeštanje socket-a (Subscriber-a) u strukturu 1 i 2
6) Publisher ponovo šalje topic i poruku na server
7) Server prima novu poruku od Publisher-a
8) Server šalje poruku Subscriber-u
9) Subscriber prima poruku i ispisuje je

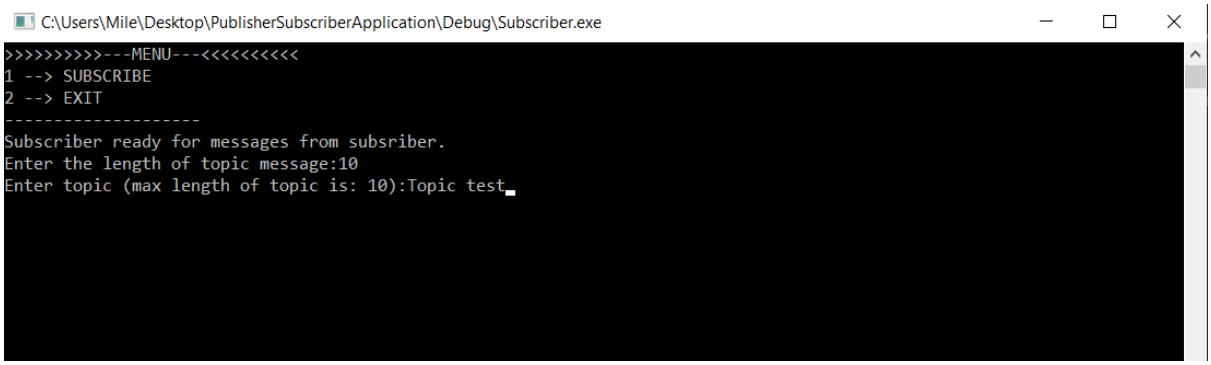
1. Pokretanje aplikacije: Kao što je prethodno i navedeno, pokrene se server, a zatim Publisher i Subscriber.

2. Publisher šalje topic i poruku na server: Slika 2.7 predatvlja unete podatke na strani Publisher-a. Pritiskon na tatser **ENTER** podaci se šalju na server.

Slika 2.6 Unos i slanje podataka na Publisher-u

3. Smeštanje podataka u strukturu 1: Kada su podaci sa Publihser-a stigli na server, topic se smesta u strukturu 1 ukoliko ne postoji. U tom slučaju poruka se ne prosleđuje u strukturu 2, jer u tom trenutku sigurno niko još od Subscriber-a nije pretplaćen na novi topic. Ukoliko topic već postoji, proverava se koji Subscriber-i su pretplaćeni na topic i svima njima se prosleđuje poruka tj. poruka se upisuje strukturu 2 svugde gde imamo pretplaćenog Subscriber-a tj. socket.

4. Subscriber šalje topic na server: Trenutno na server postoji jedan topic (“Test topic”) i ukoliko Subscriber ukuca ovaj topic i pošalje je server, biće pretplaćen na topic i stizaće mu sve poruke koje se odnose na taj topic (Slika 2.7).



Slika 2.7 Unos i slanje topic na Subscriber-u

5. Smeštanje socket-a (Subscriber-a) u strukturu 1 i 2: Kada stigne poruka sa Subscriber-a, server obrađuje poruku i ukoliko se prijavio novi Subscriber, smešta ga u strukturu 2. Zatim u strukturi 1 proverava da li postoji topic koji je primljen i ako postoji Subscriber će biti ubačen u red Subscriber-a koji su pretplaćeni na taj topic. Ukoliko topic ne postoji, Subscriber se neće nigde pretplatiti tj. neće biti ubačen u queue.

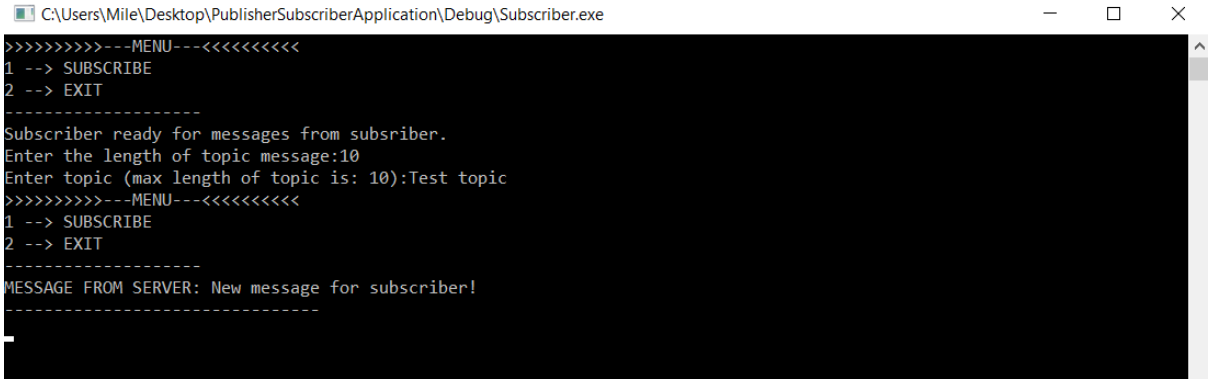
6. Publisher ponovo šalje topic i poruku na server: Trenutno na server nemamo ni jednu poruku koja se odnosi na topic “Topic test”, a Subscriber čeka poruke. sada Publisher topic “Test topic” i novu poruku (Slika 2.8).

Slika 2.7 Publisher šalje novu poruku na server za isti topic

7. Server prima novu poruku od Publisher-a: Pošto je primljen topic koji već postoji u listi topic-a, za svaki od Subscriber-a koji su pretplaćeni na taj topic se dodaje poruka u strukturi 2.

8. Server šalje poruku Subscriber-u: Pošto sada postoji poruka poruka u strukturi 2, ona se može poslati Subscriber-u. Poruka se prvo šalje, a zatim se briše iz queue-a. Uzimanje poruka iz queue-a se radi po FIFO principu. Poruke se šalju na svakih 5 sekundi, sve dok se queue poruka ne isprazni.

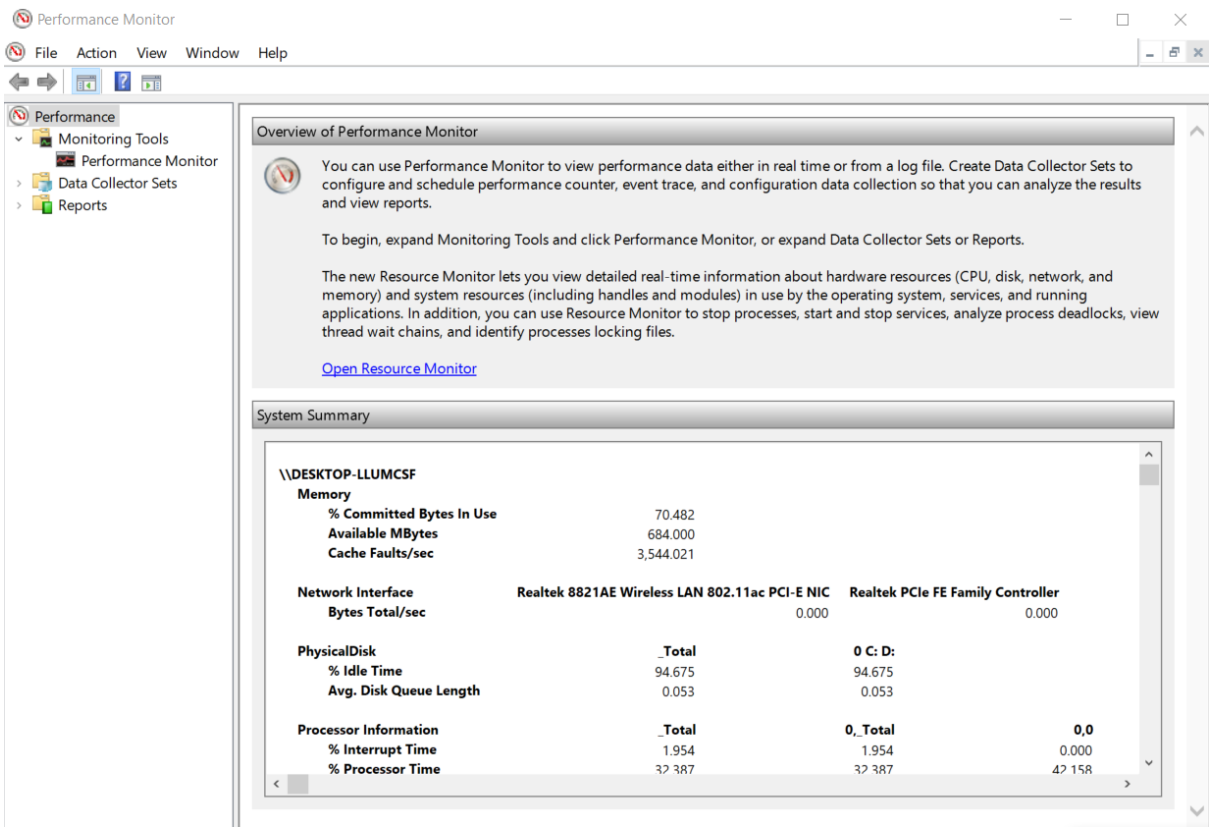
9. Subscriber prima poruku: Slika 2.8 prikazuje poruku koju je server poslao Subscriber-u.



Slika 2.8 Subscriber prima poruku sa servera

3. TESTIRANJE

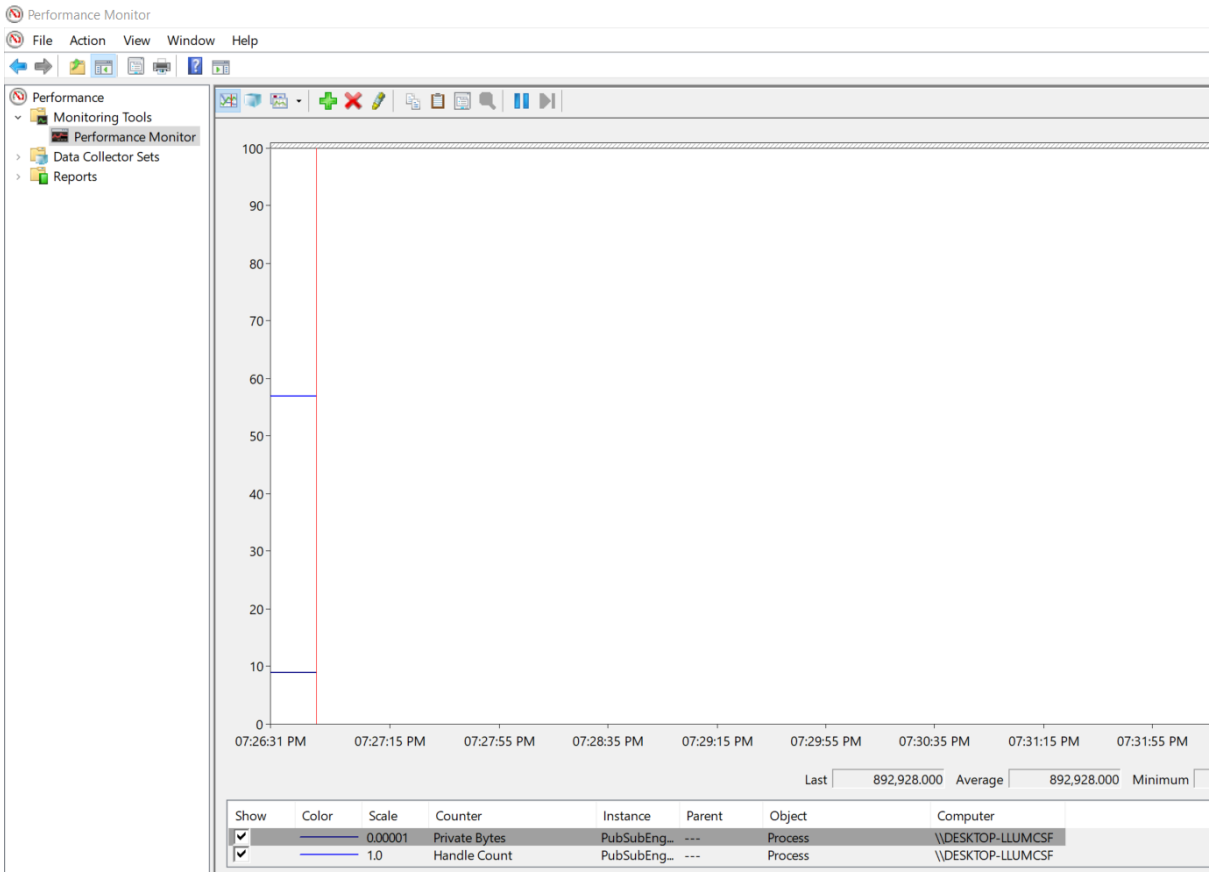
Testiranje aplikacje (stress test) je odrađeno uz pomoć "**Performanse Monitor**" alata (Slika 2.9). Testiran je sam rad aplikacije u tri tačke. Prvo testiranje aplikacije se odnosi na rad aplikacije pri samom pokretanju, drugi test u toku samog rada, dok se poslednje odnosi na trenutak kada aplikacija prestaje sa radom. Testiranje se radi nad serversom aplikacijom **PubSubEngine**.



Slika 3.1 Performanse Monitor Manager

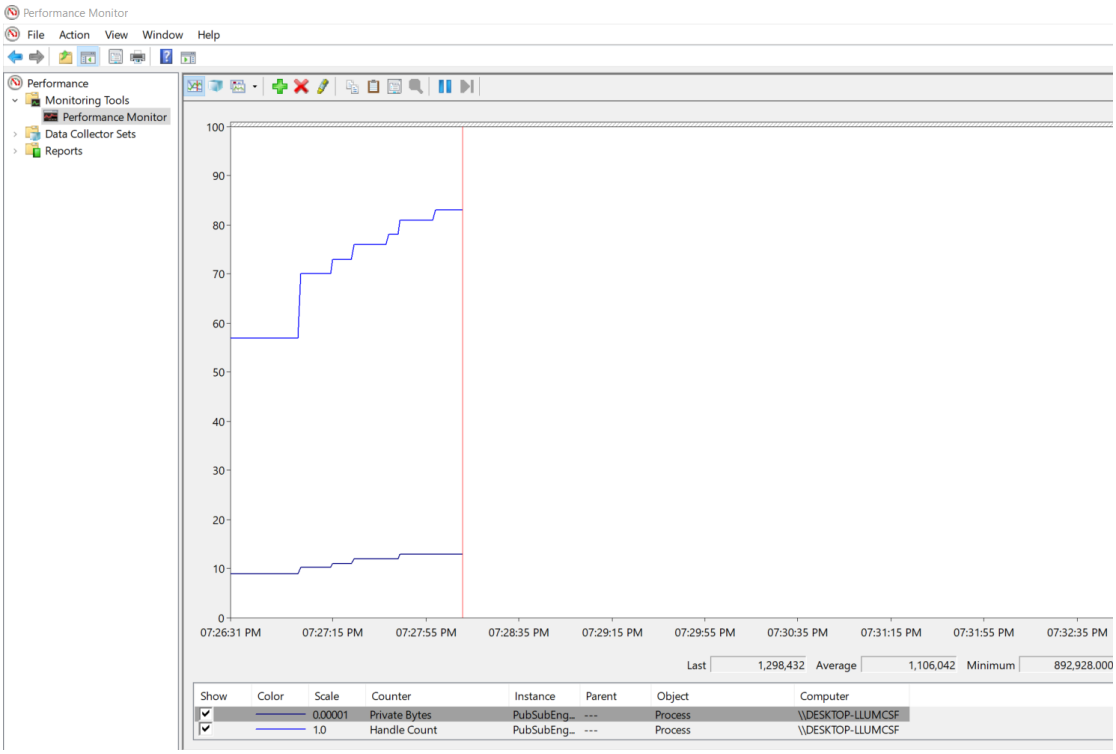
Nakon pokretanja aplikacije potrebno je u Performanse Monitor manager dodati proces koji koji predstavlja PubSubEngine aplikaciju (**Monitoring Tools -> Performance Monitor**). Nakon odabira procesa, potrebno je izabrati brojače koji će biti prikazivani pri radu aplikacije. Brojači izabrani za ovaj test su **HandleCount** (meri zauzetost handle-ova) i **PrivateByte** (meri zauzetost same memorije).

U **prvoj fazi** testiranja na grafiku (Slika 2.10) vidimo početnu zauzetost handle-ova i memorije. Primećuje se da na početku nema povećanja korišćenja ni memorije ni handle-ova zbog trenutnog ustaljenog rada aplikacije.



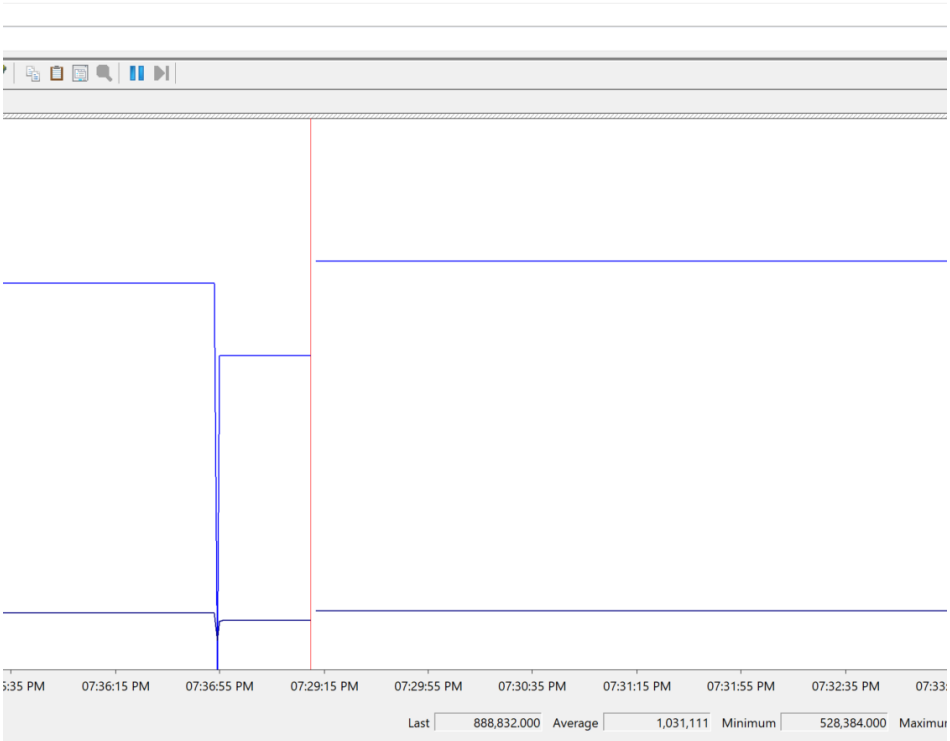
Slika 3.2 Faza 1 - testiranje

Druga faza testiranja se radi pri najvećem opterećenju u radu aplikacije. U ovom preiodu rada, zauzetost memorije i handle-ova je najveća (Slika 3.3). Konkretno trenutak testiranja je trenutak u kome Server šalje poruke Subscriber-u i potom ih briše iz reda poruka.



Slika 3.2 Faza 2 - testiranje

Treća faza testiranja predstavlja trenutak kada se serverska aplikacija ugasi. Tada se svi zauzeti handle-ovi i memorija oslobađaju (Slika 3.3).



Slika 3.2 Faza 3 - testiranje

4. ZAKLJUČAK

Prikazano testiranje pokazuje da sama aplikacija u nekim delovima rada koristi više, a u nekim manje memorije. Definitivno, najkritičniji trenutak u radu aplikacije, u kome se i zauzima najviše resursa, je trenutak slanja poruka Subscriber-ima. Optimizacija ovog dela aplikacije je navedena u delu 2.1. „Dupliranje poruka u strukturi 2“.

Arhitektura aplikacije (Client-Server) predstavlja jedanu od prednosti jer klijenti ne moraju da imaju posebnu logiku, već samo šalju podatke na server, koji vrši sve proračune i po potrebi im vraća odgovore.

U delu delu 2.2 – „Primer rada aplikacije“ je prikazan rad aplikacije sa tri komponente (Server i po jedan Publisher i Subscriber). Aplikacija može da radi i ukoliko imamo veći broj klijenata (i Publisher-a i Subscriber-a). U tom slučaju će se i zauzimati više resursa, prevenstveno zbog više niti koje opslužuju klijente, kao i većeg broja poruka koje će trebati da se klijentima šalju.