Факултет техничких наука

Универзитет у Новом Саду

Replication

Industrijski komunikacioni protokoli u infrastrukturnim sistemima

Primenjeno softversko inženjerstvo

NOVI SAD, 2024.

Autori: ***Broj indeksa:***

Bojan Kuljić *PR43/2020*
Arsenije Knežević *PR52/2020*

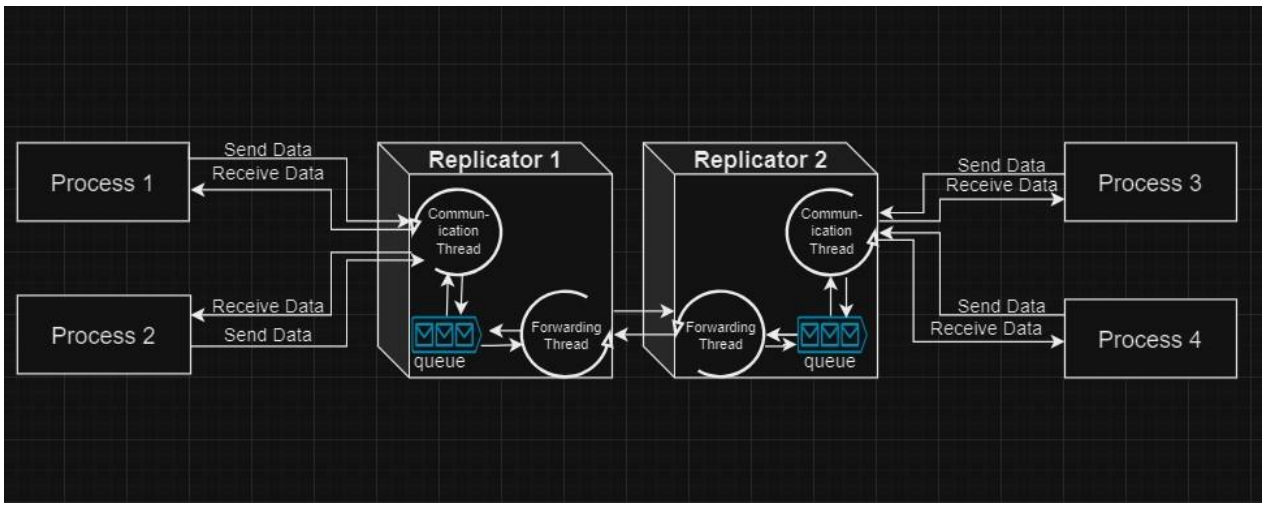
Uvod

*Projekat implementira servis za replikaciju podataka izmedju klijenata. Može da postoji više **klijenata** i 2 **replikatora** koji su zaduženi za replikaciju i komunikaciju sa klijentima. Replikator opslužuje jednu grupu korisnika i šalje drugom replikatoru. Proces replikacije se odvija na 2 načina, sinhrono i asinhrono. Sinhrona replikacija je implementirana tako da odmah po slanju podataka na jedan od izabranih replikatora, oni se repliciraju na drugi. Dok kod asinhrone replikacije podaci se ispisuju samo na izabranom replikatoru, a da bi se izvršila replikacija neophodno je da klijent zatrazi od servisa izvršenje metode „Replicate“.*

*Glavni problem predstavlja način implementacije replikatora u smislu da poruke trebaju da se **pamte** na istom i **prosledjuju** narednom.*

*Ciljevi zadatka su realizacija komunikacije klijenata na što **efikasniji** i **jeftiniji** način, kao i izbegavanje dobro poznatog bug-a tzv. „**curenje memorije**“.*

Dizajn



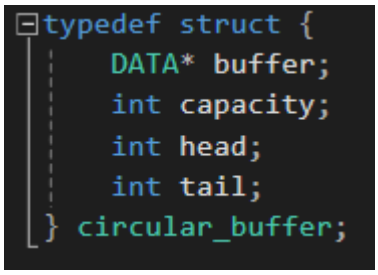
Dijagram komponenti

*Replikatori sadrže thread-ove od kojih je jedan zadužen za replikaciju sa susednim replikatorom, a ostali za komunikaciju sa odgovorajućim procesima, to jeste za slanje i primanje podataka. Replikator, takodje, sadrži **kružni bafer(**na dijagramu označen sa queue**)** u koji smešta podatke koji pristižu i prosledjuje ih susednom replikatoru. Pored toga, u replikatoru se nalazi lista u koju se uvezuju svi procesi koji su se na njega registrovali. Takođe postoji glavni Thread koji je zadužen za registrovanje novih korisnika.*

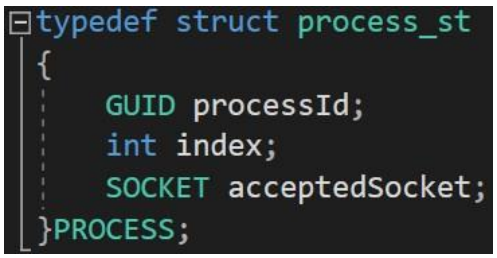
Strukture podataka

Strukture podataka koje smo koristili u projektu i koje su navedene ispod omogućavaju sve funkcionalnosti projekta i pojednostavljuju kompleksnost u cilju efikasnosti. Ne spadaju u grupu ugrađenih struktura podataka(STL), već su samostalno implementirane.

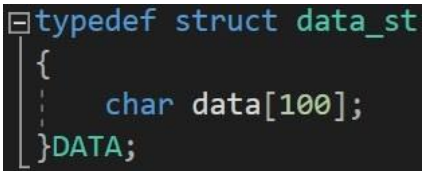
- ***Kružni bafer*** *se koristi za skladištenje podataka primljenih od procesa na replikatorima. Postoje 2 „pokazivača“:* ***Tail*** *i* ***Head****. Head pokazuje na početak bafera,a Tail na kraj. Imamo i polje* ***Buffer*** *tipa Data koje služi za čuvanje podataka i polje* ***Capacity*** *koje definiše njegov kapacitet.*



- ***Process*** *je struktura koja ima zadatak da primi zahtev klijenta preko soketa i sačuva* njegov indeks(ID).

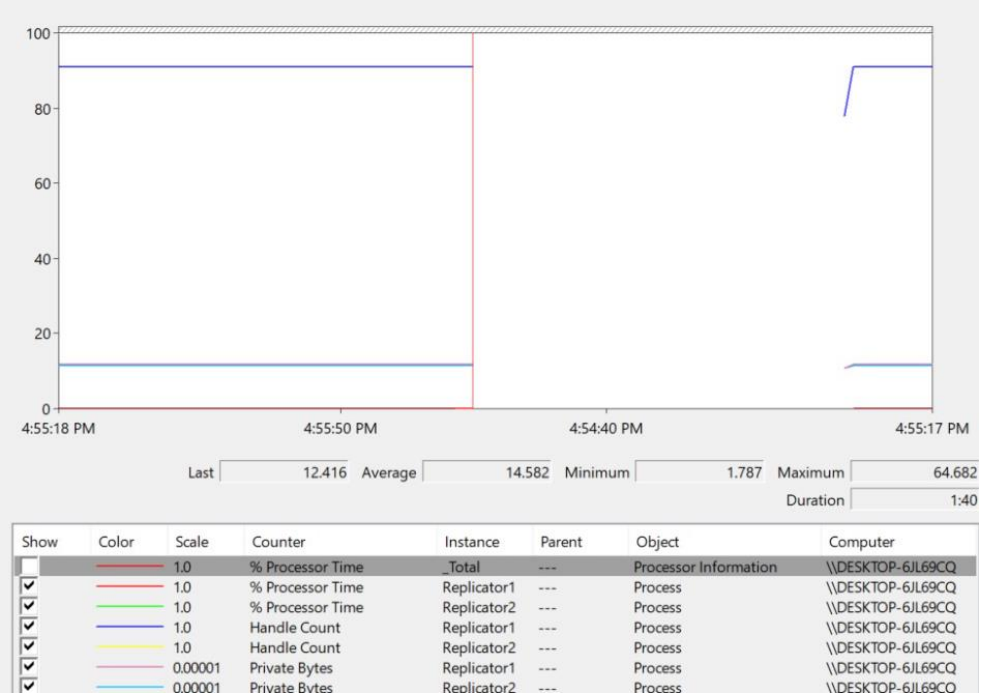


- ***Data*** *predstavlja poruku koju procesi salju servisu. Klijent maksimalno može poslati* 100 bajtova.

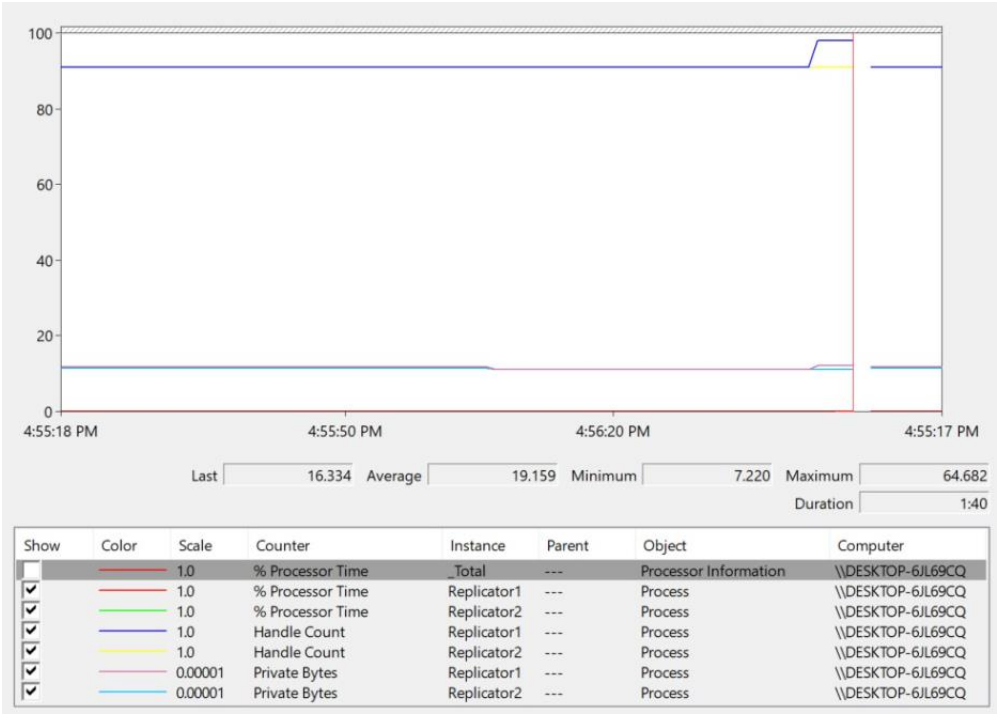


Rezultati testiranja

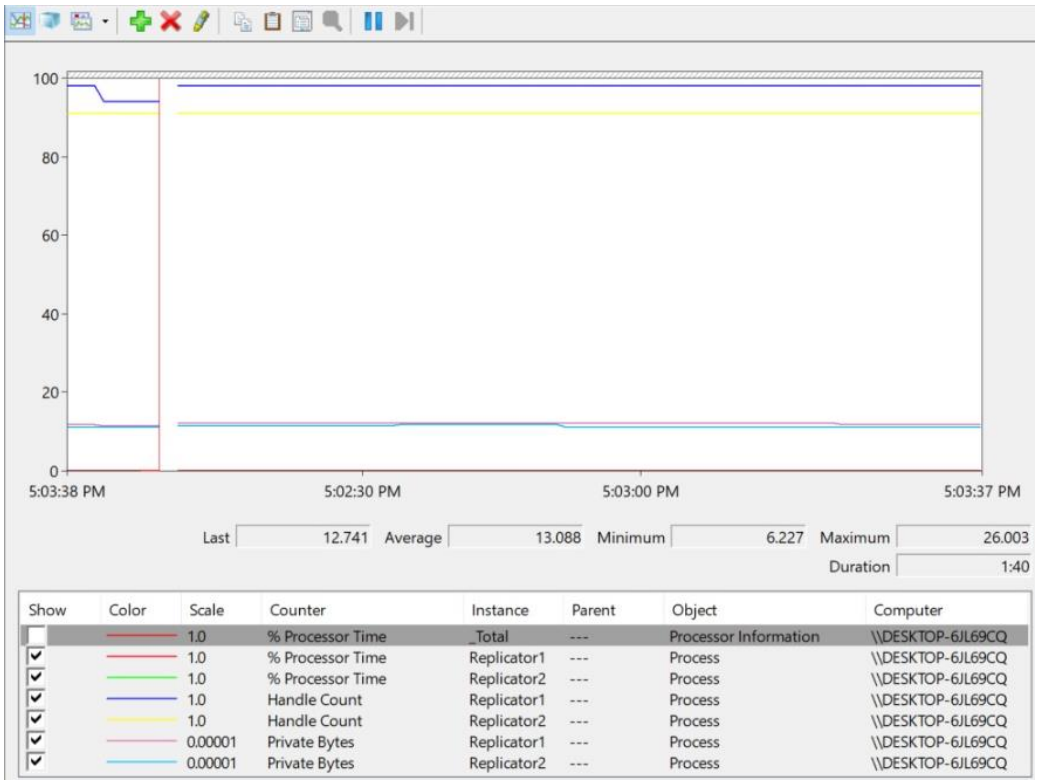
*Testiranje našeg projekta smo vršili pomoću **Performance Monitora** u kojem smo proveravali: broj pokrenutih thredova,koliki procenat procesorskog troška zauzima aplikacija , kao i koliko memorije se zauzima, a i oslobađa.*



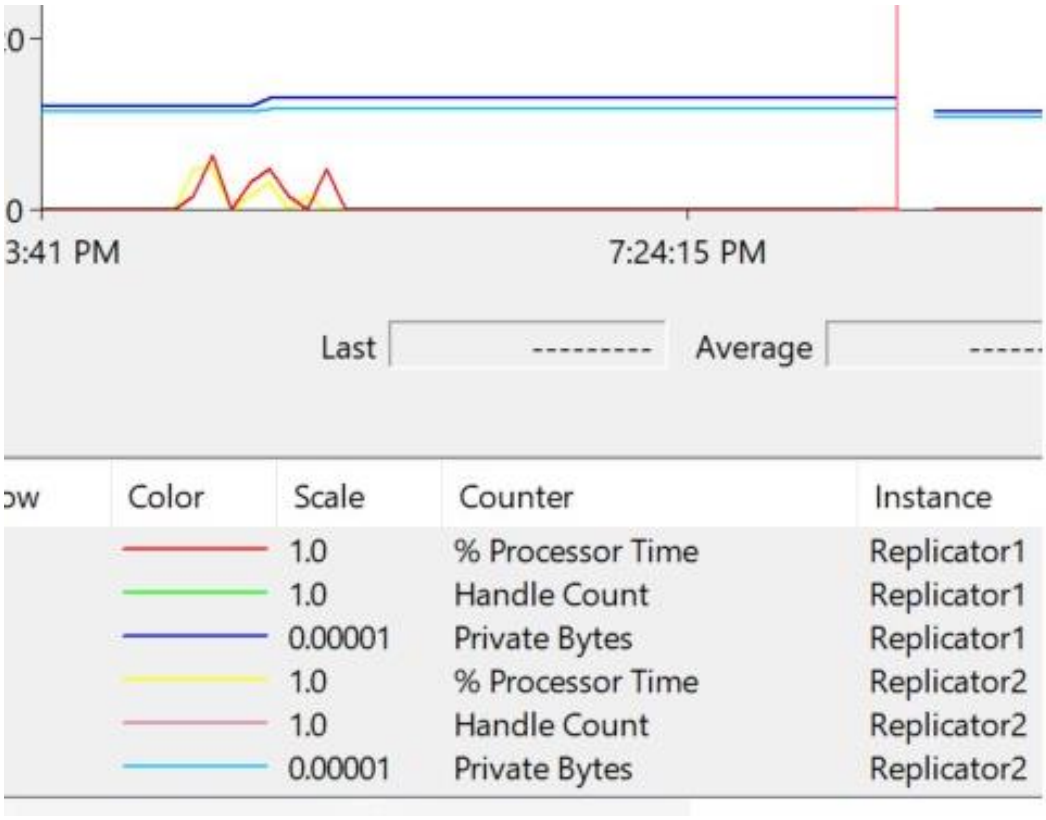
Na prvoj slici vidimo kako izgleda naš Performance Monitor kada se aplikacija pokrene



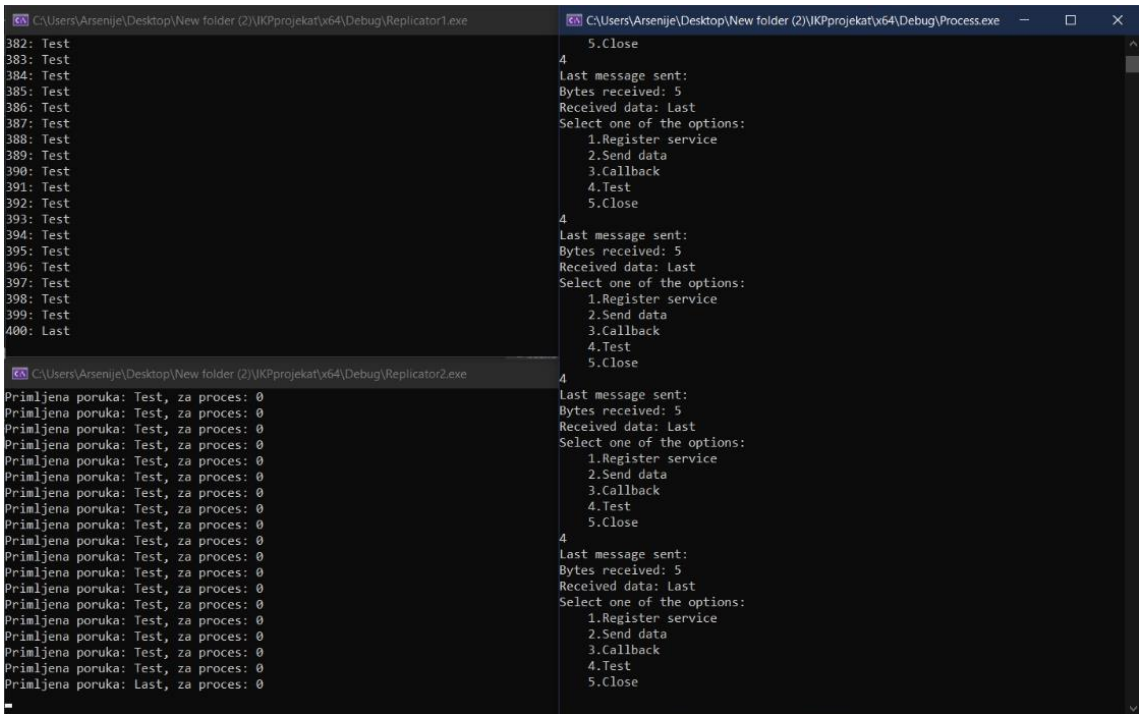
Na drugoj slici vidimo povećavanje broja niti nakon povezivanja korisnika.



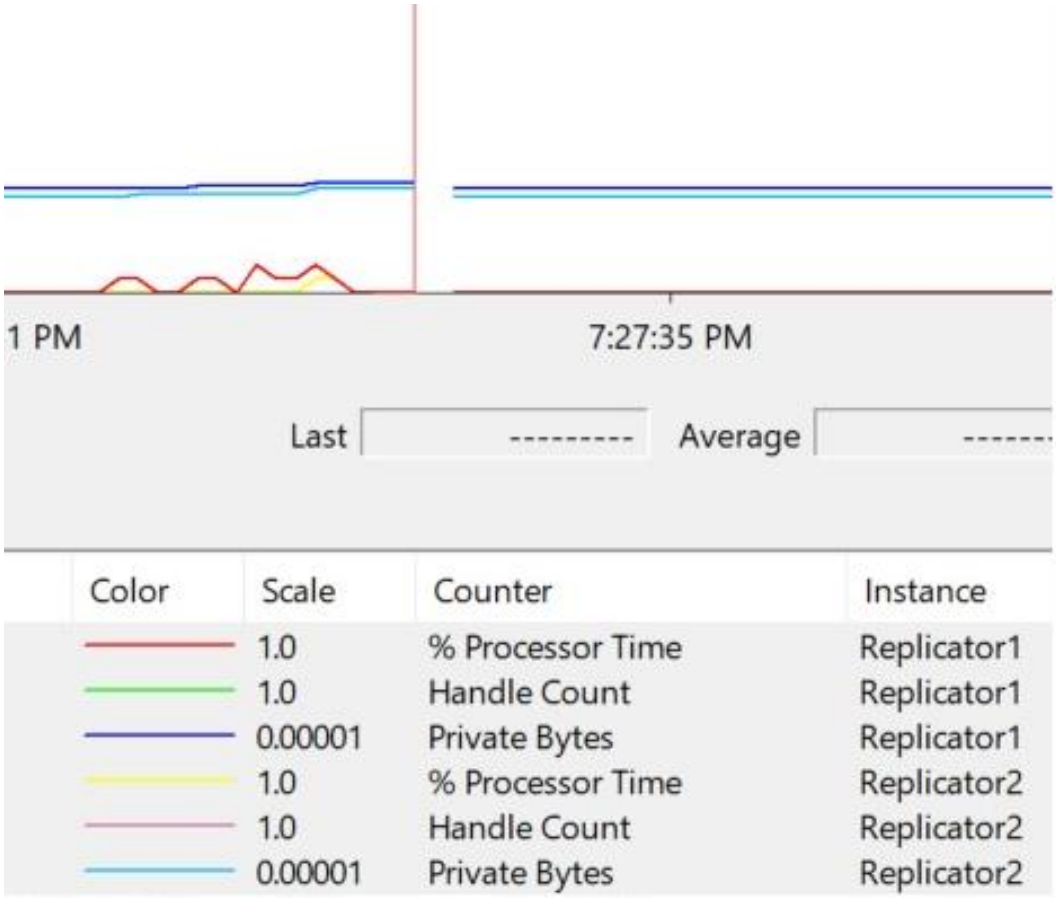
Dok na trećoj slici vidimo smanjenje broja niti nakon prekidanja konekcije sa korisnikom.



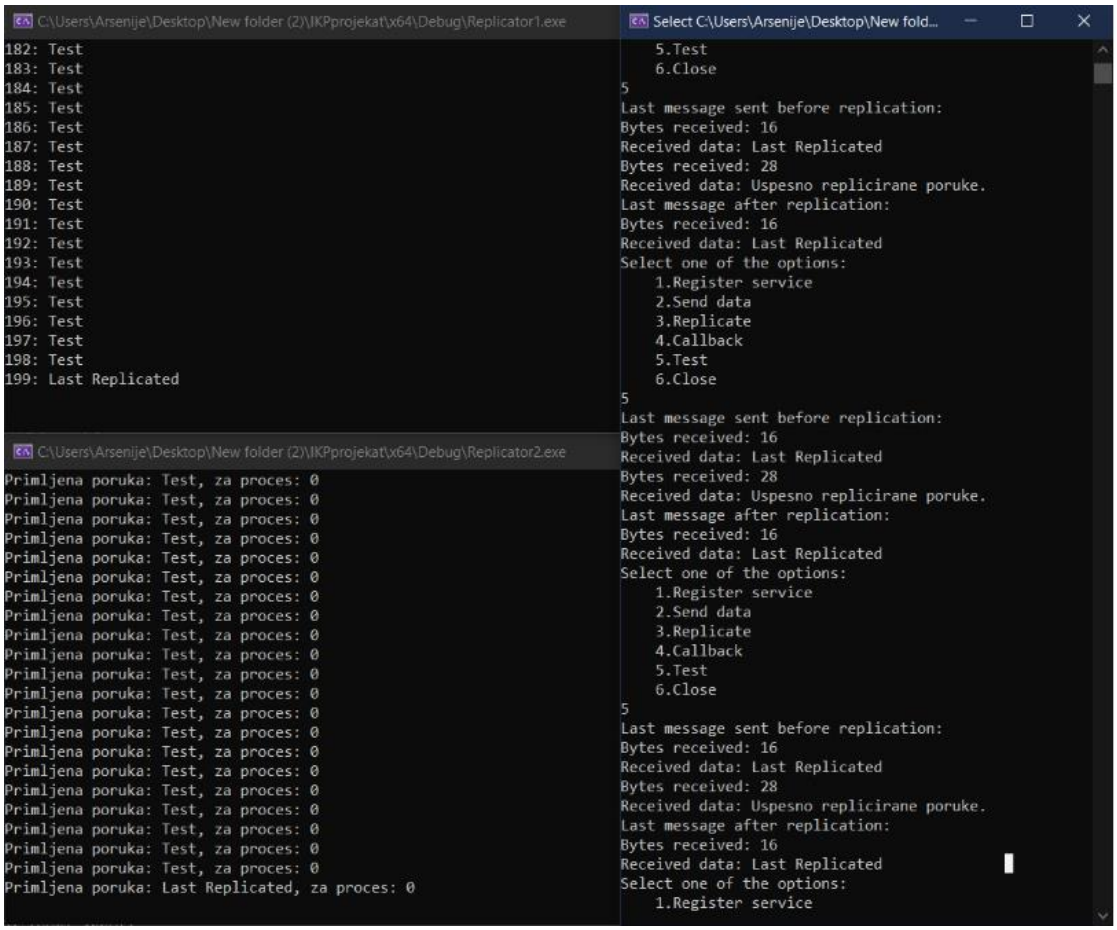
Na četvrtoj slici vidimo povećanje %CPU kao i memorije kada vršimo sinhrono testiranje.



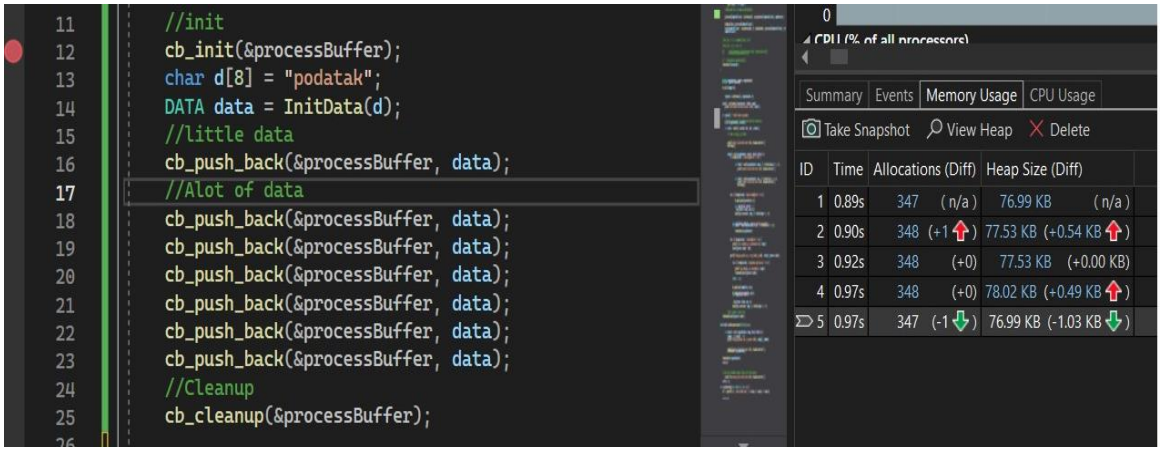
Na petoj slici su prikazane konzole tokom sinhronog testiranja pozivom metode „Test“



Na šestoj slici vidimo povećanje %CPU kao i memorije kada vršimo asinhrono testiranje.



Na sedmoj slici su prikazane konzole tokom asinhronog testiranja pozivom metode „Test“.



Na osmoj slici su sa lijeve strana prikazane metode za rad sa memorijom, a sa desne strane ***<u>Heap Snapshot</u>*** *naše memorije kao i njeno zauzimanje i oslobađanje. Prvo je prikazana memorija pre inicijalizacije bafera(1),zatim posle inicijalizacije(2) Zatim posle prve poruke(3)(nije se jos povecao bafer), pa nakon više poruka(4)(dovoljno da se poveća) i na kraju CleanUp(5) tj. Oslobađanje memorije.*

Zaključak

Na osnovu rezultata testiranja korišćenjem Performance Monitora, možemo zaključiti da naša aplikacija pokazuje zadovoljavajuće performanse i stabilnost tokom različitih scenarija upotrebe. Ključni parametri, poput broja pokrenutih niti, procenta procesorskog troška i upotrebe memorije, praćeni su tokom različitih aktivnosti.

Prilikom pokretanja aplikacije, zabeležili smo stabilno ponašanje i optimalnu upotrebu resursa. Povećanje broja niti prilikom povezivanja korisnika, kao i smanjenje nakon prekida konekcije, ukazuje na efikasno upravljanje resursima. Sinhrono testiranje je rezultiralo očekivanim povećanjem procesorskog opterećenja i memorije, dok je asinhrono testiranje pokazalo slične karakteristike.

Pratili smo konzole tokom testiranja metode "Test" i utvrdili da nema značajnih grešaka ili nepravilnosti. Takođe, analizirali smo rad sa memorijom i utvrdili da se resursi pravilno alociraju i oslobađaju, što je posebno vidljivo kroz korake inicijalizacije bafera, primene poruka i finalnog čišćenja.

Heap Snapshot prikazuje kontrolisano zauzimanje i oslobađanje memorije tokom različitih faza izvršavanja aplikacije, a čišćenje memorije nakon završetka procesa dodatno potvrđuje dobar dizajn i implementaciju.

Sveukupno, naš zaključak je da su rezultati testiranja pozitivni, te da naša aplikacija zadovoljava postavljene zahteve u pogledu performansi, stabilnosti i efikasnog upravljanja resursima.

Potencijalna unapredjenja

Potencijalna poboljšanja za našu aplikaciju mogu uključivati implementaciju thread pool-a radi efikasnijeg upravljanja niti prilikom kreiranja novih procesa. Ovo može dovesti do smanjenja opterećenja sistema, bržeg odziva i bolje iskorišćenosti resursa.

Još jedna mogućnost unapređenja je optimizacija upravljanja porukama u kružnom baferu. Nakon što se poruke ispišu, trenutno ostaju u baferu. Poboljšanje bi moglo uključivati mogućnost upotrebe mehanizma za brisanje poruka iz bafera nakon što su uspješno replicirane. Ovo može doprinjeti efikasnijem upravljanju memorijom i smanjenju nepotrebnog zauzeća resursa.

Ove promjene bi mogle donjeti dodatnu efikasnost, bolje performanse i optimizaciju resursa za našu aplikaciju, čime bi se osiguralo dugoročno održavanje i poboljšano korisničko iskustvo.