

Dokumentacija

Industrijski komunikacioni protokoli u elektroenergetskim sistemima
2021/22

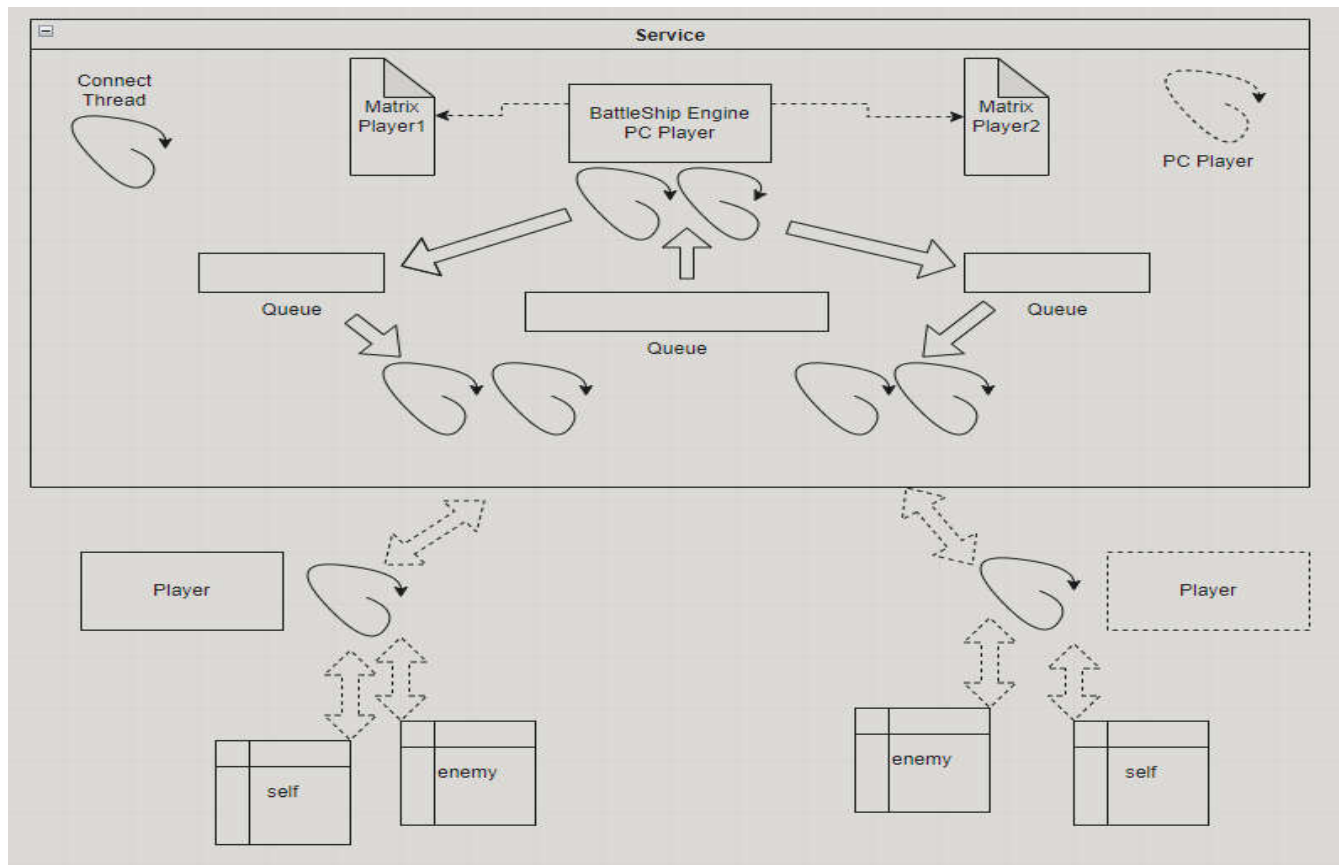
Battleships

Miloš Maksimović PR62/2018
Đorđe Božović PR117/2018

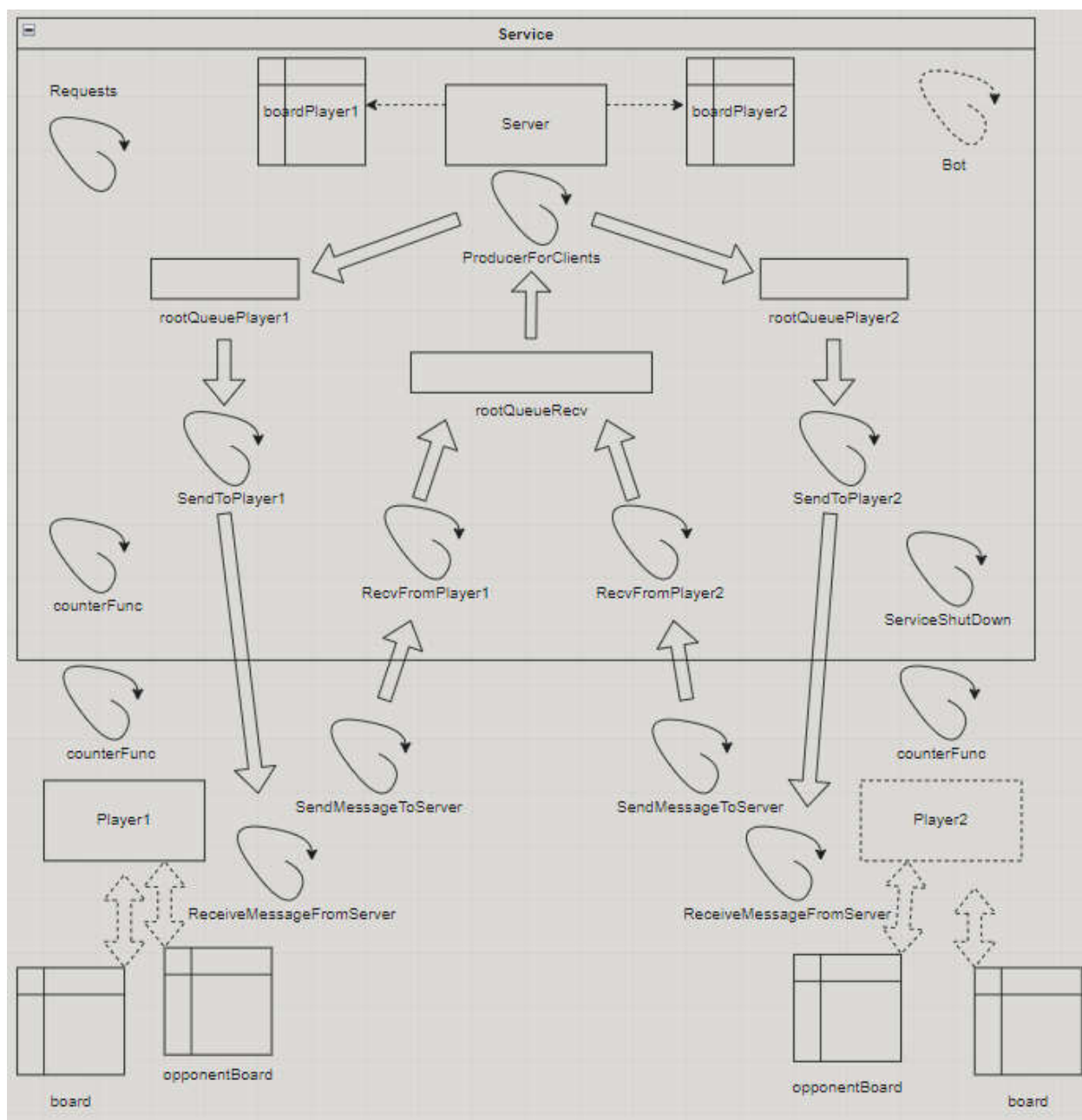
Uvod

Problem koji je bio pred nama je razviti aplikaciju koja predstavlja server i klijente za igranje igrice potapanje brodova (battleships). Program se sastoji od servisa koji je zadužen za sve proračune koji se tiču igre, klijenti su u mogućnosti da igraju jedni protiv drugih ili protiv bota to jest servera. U jednom momentu je moguće da se konektuju samo dva klijenta dok će treći biti odbijen uz obrazložen razlog da igra i dalje traje. Svaki klijent ima mogućnost da postavi brodove ručno ili da se posluži automatskim postavljanjem. Kada igra počne pokreće se i tajmer koji označava koliko vremena klijenti imaju na raspolaganju. Kada klijenti biraju polje koje žele da gađaju unose veliko početno slovo od A do J kao i broj od 1 do 10 (npr: A1). Ako igrač pogodi polje dobija mogućnost da ponovo gađa neko drugo polje. Prilikom promašaja potez se prebacuje na drugog igrača. Kada tajmer istekne potez se prebacuje na narednog igrača, ali ako tajmer istekne dva puta istom igraču onda igrač gubi igru. Kada se igra završi klijenti se zatvaraju i server je spreman da prihvati nove konekcije za narednu igru.

Dizajn



Slika 1: Početni diagram



Slika 2: Trenutni diagram

Dizajn se sastoji od servisa i klijenta koji su dva posebna procesa. Servis se sastoji od više niti koje se pokreću u zavisnosti u kom stanju se servis nalazi. U nastavku će se svaka nit objasniti prvo sa serverske pa onda sa klijentske strane.

1. **Requests** - je funkcija koja obezbeđuje da se prihvati konekcija od strane klijenta, funkcija prihvata dve konekcije dok se treća prihvati kako bi se ugasila jer nije dozvoljeno da budu više od dva klijenta konektovana na servis. Ovde se i šalje inicijalna poruka klijentu koja mu potvrđuje njegovu konekciju sa serverom. Nakon što se utvrdi da se klijent konektovao i da se nije desila neka greška pokreću se dve niti koje će služiti da šalju i primaju poruke ka i od klijenta.

2. **ProducerForClients** - je funkcija koja služi da se sa reda (Queue), ako postoji, očita poruka i prosledi na odgovarajući klijentski red kako bi bile prosleđene klijentima.
3. **SendToPlayer1** - je funkcija koja služi da se poruka, koja se nalazi u redu za prvog klijenta, pošalje preko TCP/IP protokola na klijentsku stranu.
4. **SendToPlayer2** - ova funkcija je ista kao i prethodna samo što se poruka šalje klijentu koji se drugi konektovao.
5. **RecvFromPlayer1** - je funkcija koja je zadužena da primi poruku od klijenta i zatim je postavi u red.
6. **RecvFromPlayer2** - isto kao i prethodna funkcija samo za igrača broj dva.
7. **counterFunc** - je funkcija koja služi za pravilan rad tajmera, za prebacivanje poteza kada istekne tajmer kao i odlučivanje da li je igrač pobedio ili izgubio ako je tajmer istekao više puta.
8. **Bot** - je funkcija koja se pokreće kada igrač želi da igra protiv bota, omogućava da igrač igra protiv servisa.
9. **ServiceShutDown** - je funkcija koja je zadužena da prekine rad servisa ako se unese vrednost 'q' i ako igra nije u toku servis će se ugasi.

Ovo su bile sve funkcije koje se pokreću u nitima na serverskoj strani. **Requests**, **ProducerForClients** i **ServiceShutDown** se pokreću odmah po pokretanju servisa dok se **SendToPlayer** i **RecvFromPlayer** pokreću prilikom prihvatanja konekcije odgovarajućeg klijenta. **counterFunc** se pokreće prvi put prilikom pokretanja igre kako bi se pratilo vreme poteza. **Bot** se pokreće na zahtev klijenta da igra protiv servisa.

Funkcije koje se nalaze na klijentskoj strani i pokreću se u nitima su:

1. **SendMessageToServer** - je funkcija koja se koristi da bi se serveru poslala poruka.
2. **ReceiveMessageFromServer** - je funkcija koja se koristi da se primi i obradi poruka koja je pristigla od servera.
3. **counterFunc** - je funkcija koja služi da se vodi evidencija o preostalom vremenu tokom igranja partije i da se izmeni prikaz na UI.

Ove funkcije se pokreću kao niti na klijentskoj strani nakon što partija počne.

Strukture podataka

Pri realizaciji projekta smo koristili složenu strukturu red tj. Queue. Queue je odabran iz razloga što je potrebno da se poruke razmenjuju između niti. Ako se pogleda slika 2 možemo videti da postoje tri implementacije reda. Red koji je zadužen da prihvata poruke od klijenta je red pod nazivom **rootQueueRecv**, niti **RecvFromPlayer** prihvataju poruku preko TCP kanala i postavljaju je na red odakle je nit **ProducerForClients** preuzima i vrši proračune u zavisnosti od vrste poruke. **ProducerForClients** takođe generiše novu poruku kao odgovor klijentima na akciju koja je prethodila. Zatim se ta novogenerisana poruka postavlja u klijentske redove **rootQueuePlayer1** i **rootQueuePlayer2** odakle će niti **SendToPlayer** preuzeti poruku i putem TCP/IP protokola poslati na klijentsku stranu.

Poruka koja se spominje u tekstu iznad je struktura pod nazivom **MESSAGE_TCP**. Ova struktura se sastoji iz **MessageType** polja koje sadrži vrstu poruke koja se predstavlja kao enumeracija, zatim imamo **ActionPlayer** što nam označava koji igrač je izvršio akciju ili kojem igraču se prosleđuje poruka. Sadrži i polje koje se zove **argument** koje nam je predstavljeno kao niz char-ova dužine 4. I kao poslednje polje u strukturi imamo **matrixArgument** što je dvodimenzionalni niz char-ova koji predstavlja matricu prilikom igranja igre.

Način rada

U tekstu koji sledi će se objasniti kako pravilno pokrenuti i koristiti aplikaciju.

Igra protiv igrača:

1. Pokrenuti server.exe.
2. Pokrenuti dva klijenta (client.exe).
3. Na klijentu izabrati da se želi igrati protiv drugog igrača.
4. Postaviti brodove po formatu veliko slovo od A - J zajedno sa brojem od 1 - 10, npr: A10, G4, E1. Traži se da se unese početna i krajnja koordinata koja će biti dužine broda, u suprotnom neće se postaviti brod. Unosom 0 se brodovi postave automatski.
5. Prilikom postavljanja brodova nije dozvoljeno da se brodovi dodiruju, neophodno je postaviti brodove tako da između svakog broda barem jedno mesto bude prazno.
6. Kada oba klijenta postave svoje brodove igra počinje. Prvi potez ima klijent koji se prvi konektovao.
7. Kada jedan igrač gađa neko mesto na tabli potrebno je uneti komandu u formatu: veliko slovo od A - J zajedno sa brojem od 1 - 10, npr: A10, G4, E1.
8. Svaki igrač ima 30 sekundi da odigra svoj potez, ako vreme istekne potez se prebacuje na sledećeg igrača, ako igrač dva puta ne odigra potez igra se završava i korisnik koji nije odigrao svoje poteze gubi igru a drugi igrač pobeđuje.
9. Nakon završetka igre klijenti mogu da se ugase, a servis je spreman da prihvati nove konekcije za novu igru.

Igra protiv bota:

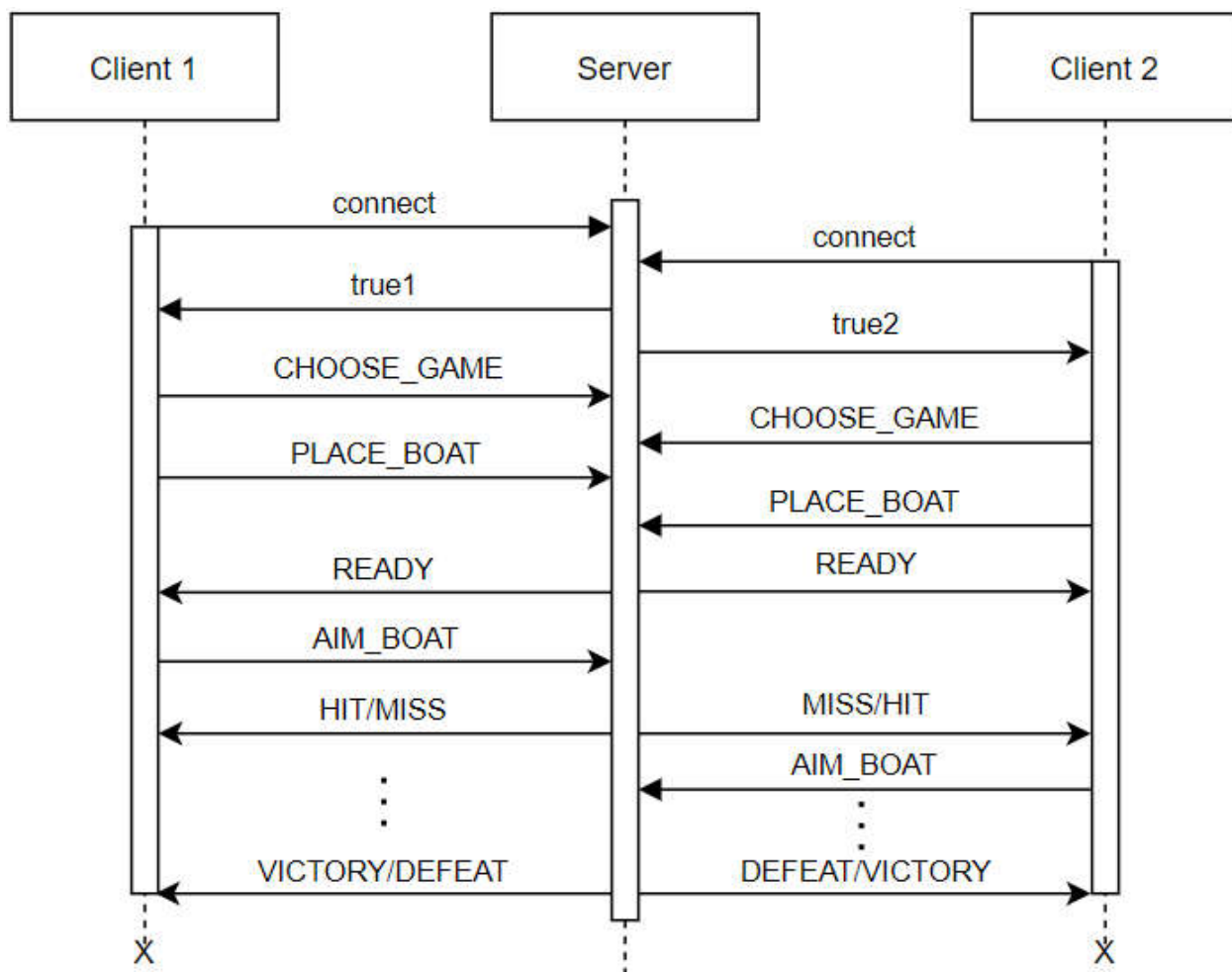
1. Pokrenuti server.exe.
2. Pokrenuti jednog klijenta (client.exe).
3. Na klijentu izabrati da se želi igrati protiv bota.
4. Postaviti brodove po formatu veliko slovo od A - J zajedno sa brojem od 1 - 10, npr: A10, G4, E1. Traži se da se unese početna i krajnja koordinata koja će biti dužine broda, u suprotnom neće se postaviti brod. Unosom 0 se brodovi postave automatski.
5. Prilikom postavljanja brodova nije dozvoljeno da se brodovi dodiruju, neophodno je postaviti brodove tako da između svakog broda barem jedno mesto bude prazno.
6. Kada klijent postavi svoje brodove igra počinje. Prvi potez ima klijent.
7. Kada igrač gađa neko mesto na tabli potrebno je uneti komandu u formatu: veliko slovo od A - J zajedno sa brojem od 1 - 10, npr: A10, G4, E1.
8. Igrač ima 30 sekundi da odigra svoj potez, ako vreme istekne potez se prebacuje na bota, ako igrač dva puta ne odigra potez igra se završava i korisnik gubi igru a bot pobeđuje.
9. Nakon završetka igre klijent može da se ugasi, a servis je spreman da prihvati nove konekcije za novu igru.

Napomena prilikom igranja:

1. Očekuje se da će klijenti prilikom inicijalizacije (postavljanja brodova) biti konektovani.
2. Očekivano je da će server moći da se ugasi samo kada nema igre u toku.
3. Mogu se klijenti diskonektovati tokom igre, svaki put kada se konektuju dobiće svoju tabelu sa postavljenim brodovima kao i tabelu pokušaja koju su gađali.

Primer toka igre sa dva igrača

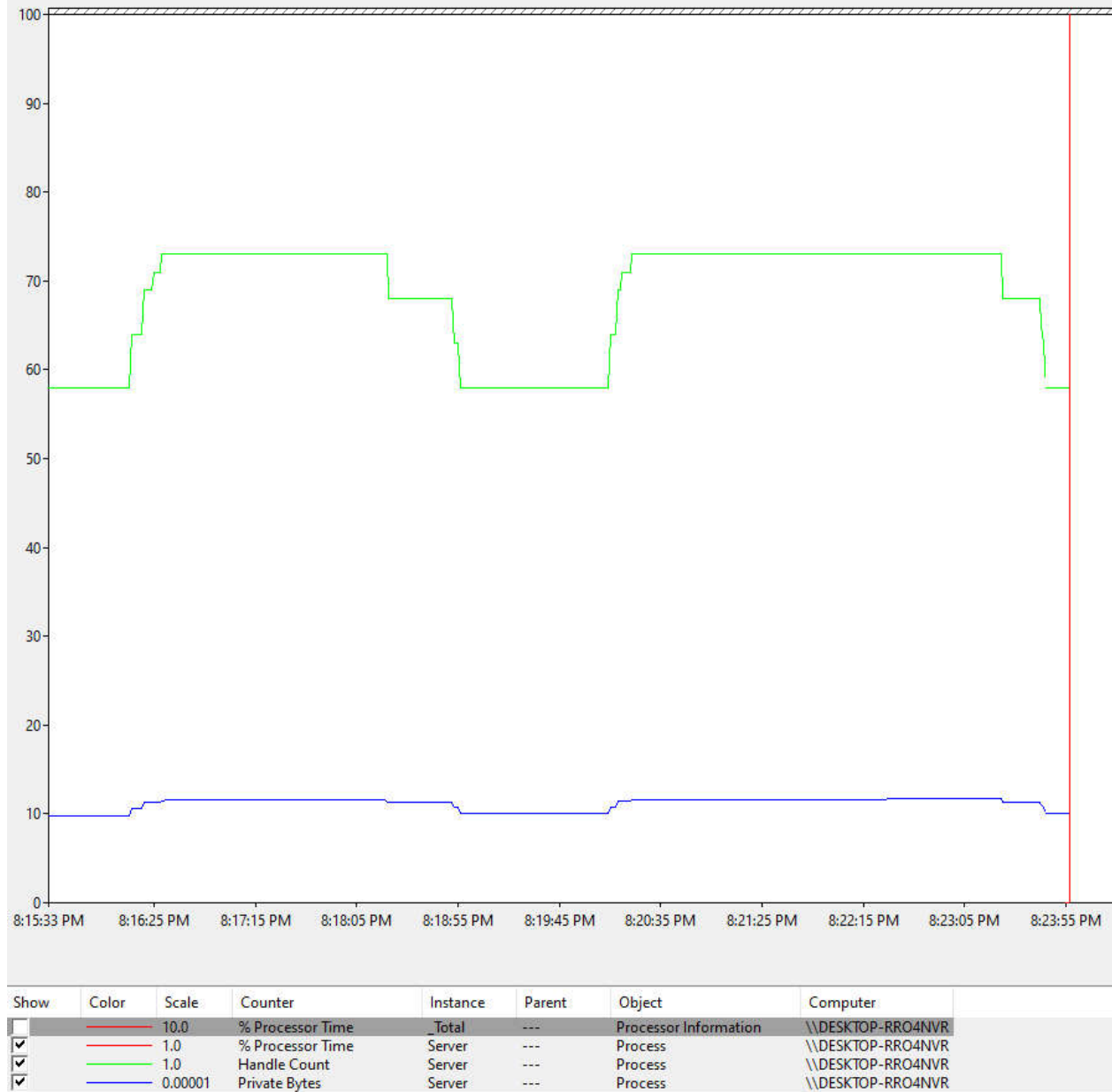
Kako bi tok igre bio jasniji, objasnićemo ga uz pomoć slike 3. Kao što se može videti na slici, prvi korak je konekcija klijenta, nakon toga server šalje potvrdu o uspostavi veze, poruka je reč true zajedno sa rednim brojem konekcije. Nakon što klijent primi potvrdu od servera, uz pomoć **MESSAGE_TCP** strukture šalje pozicije svojih brodova. Kada oba klijenta pošalju svoje brodove, igra počinje tako što server odgovara klijentima sa porukom READY. Klijenti naizmenično gađaju protivničke table, prvi igrač koji pogodi sve protivničke brodove pobeđuje.



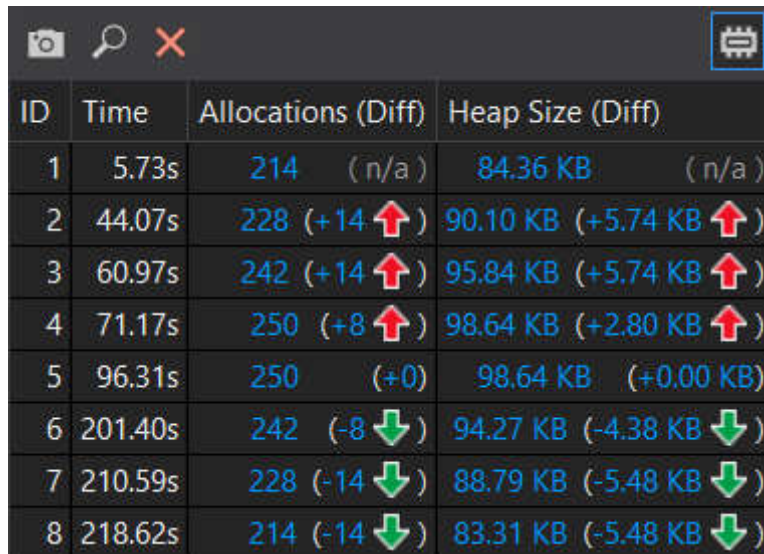
Slika 3: tok igre dva igrača

Rezultati testiranja

Prilikom testiranja servera pokrenuta su dva klijenta koja su igrala igru. Na slici 4. se može videti da se broj handle-ova povećava prilikom pokretanja klijenta. Nakon gašenja klijenta handle-ovi se oslobađaju i vraćaju u početno stanje, što predstavlja optimalno ponašanje servera. Memorija je predstavljena plavom linijom i vidi se da se prilikom pokretanja povećava koliko je potrebno da bi klijenti funkcionisali. Memorija se pravilno oslobađa i po završetku rada klijenta. Procesorsko vreme je izuzetno malo pa se može reći da je minimalna potrošnja.



Slika 4. Testiranje sa dva klijenta



ID	Time	Allocations (Diff)	Heap Size (Diff)
1	5.73s	214 (n/a)	84.36 KB (n/a)
2	44.07s	228 (+14 ↑)	90.10 KB (+5.74 KB ↑)
3	60.97s	242 (+14 ↑)	95.84 KB (+5.74 KB ↑)
4	71.17s	250 (+8 ↑)	98.64 KB (+2.80 KB ↑)
5	96.31s	250 (+0)	98.64 KB (+0.00 KB)
6	201.40s	242 (-8 ↓)	94.27 KB (-4.38 KB ↓)
7	210.59s	228 (-14 ↓)	88.79 KB (-5.48 KB ↓)
8	218.62s	214 (-14 ↓)	83.31 KB (-5.48 KB ↓)

Slika 5. Testiranje sa dva klijenta

Slika 5 predstavlja testiranje memorije prilikom igranja igre dva klijenta. Ako se pogleda slika može se uvideti da postoji 8 faza za koje su urađeni snapshotovi.

1. Stanje inicijalnog pokretanja servera, dok još klijenti nisu pokrenuti.
2. Stanje kada se jedan klijent pokrene.
3. Stanje kada se drugi klijent pokrene.
4. Ovako izgleda zauzeće memorije kada igra počne.
5. Prilikom igranja igre nema curenja memorije, to se i vidi na slici.
6. Stanje kada se igra završi i proglašeni su pobednik i gubitnik.
7. Klijent jedan se diskonektuje i napušta igru.
8. Klijent dva se diskonektuje i napušta igru.

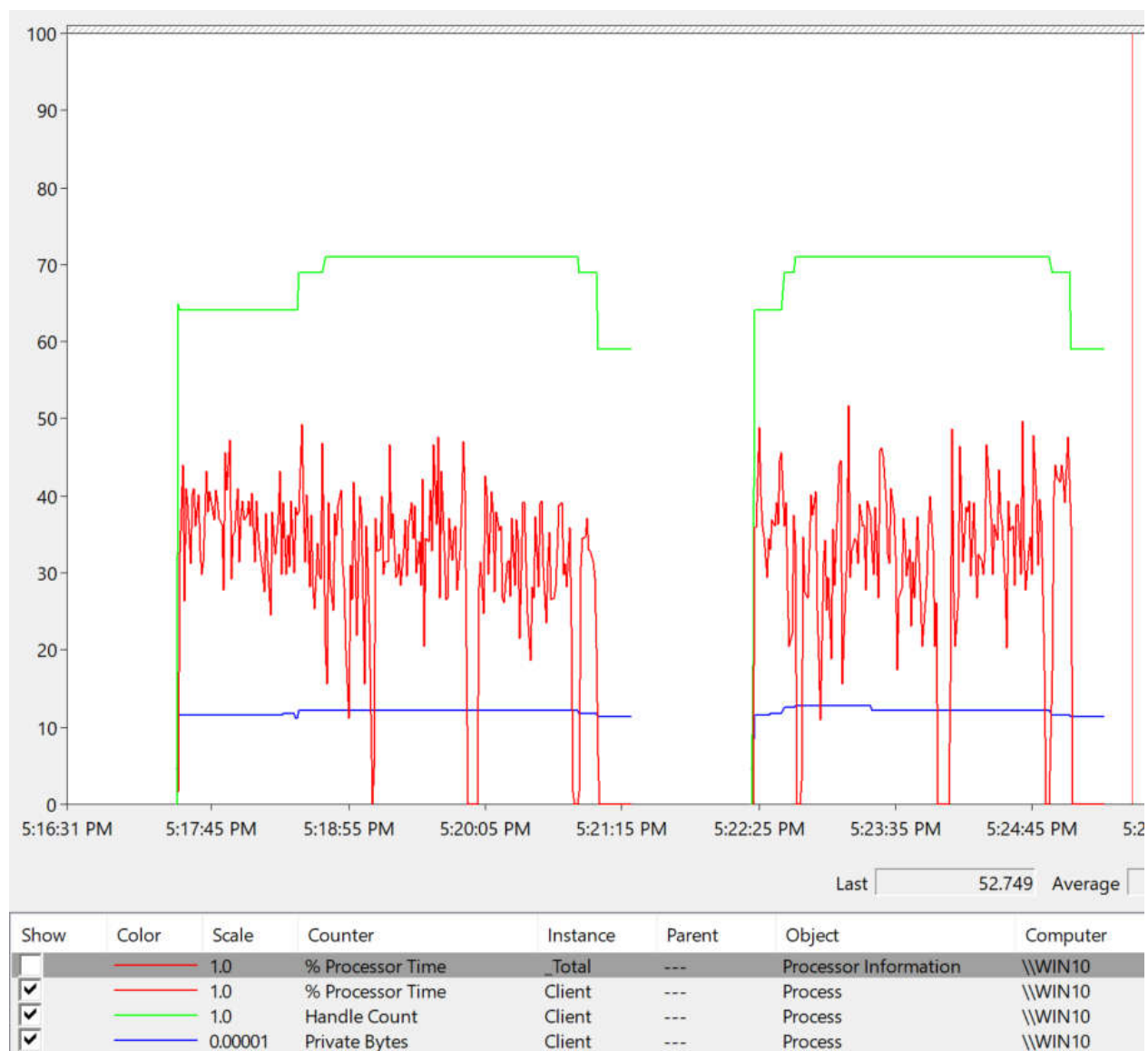
Pored ovih snapshotova napravljen je i test korišćenjem UMDH alata za praćenje memorijskog zauzeća. Rezultati se mogu videti u sklopu foldera Test1, gde postoje izveštaji o zauzeću na početku i na kraju rada servera zajedno sa izveštajem o njihovoj razlici.

Za drugi test je uz server pokrenut samo jedan klijent koji je pokrenuo igru sa botom. Na slikama 6 i 7 se može videti korišćenje memorije, handle-ova i procesorskog vremena na serveru (slika 6) i klijentu (slika 7). U svrhu boljeg testiranja i da bi rezultati bili jasniji odigrane su dve partije zaredom bez gašenja servera. Zbog gašenja klijenta nije bilo moguće videti pad broja handle-ova na istom jer je bio trenutan pa je za potrebe testiranja klijent stopiran blokirajućom funkcijom pre potpunog zatvaranja.



Slika 6. Testiranje servera sa jednim klijentom

Broj handle-ova raste prvo nakon što se klijent priključi, zatim nakon što odabere igru i u toku je postavljanja brodova i nakon što igra počne. Tokom igre broj handle-ova je konstantan. Prvi pad handle-ova se dešava nakon što se igra završi, a nakon što se klijent diskonektuje vraća se na početno stanje. Ove promene ispraćene su zelenom linijom. Plavom linijom je ispraćeno korišćenje memorije dok crvena prikazuje zauzeće procesorskog vremena.



Slika 7. Testiranje klijenta u igri sa botom

Na klijentu, slično kao i na serveru, broj handlova se menja u trenucima kada se odabere igra i nakon što se završi postavljanje brodova i igra počne. Tokom igre ovaj broj se ne menja, ali pada prvo nakon što se igra završi a zatim i kada se pritisne taster za zatvaranje klijenta. Ceo proces je grafički ispraćen zelenom linijom. Plava linija pokazuje zauzeće memorije koje raste nakon početka partije i opada na početnu vrednost nakon završetka. Zauzeće procesorskog vremena ispraćeno je crvenom linijom i veće je nego na serveru. Razlog tome su funkcije koje osvežavaju UI i čekaju da klijent unese parametre.

Take Snapshot

View Heap

Delete

ID	Time	Allocations (Diff)	Heap Size (Diff)
1	16.86s	214 (n/a)	84.36 KB (n/a)
2	56.62s	228 (+14 ↑)	90.10 KB (+5.74 KB ↑)
3	74.25s	230 (+2 ↑)	90.34 KB (+0.23 KB ↑)
4	85.55s	238 (+8 ↑)	94.99 KB (+4.66 KB ↑)
5	102.58s	238 (+0)	94.99 KB (+0.00 KB)
6	161.22s	234 (-4 ↓)	92.95 KB (-2.05 KB ↓)
7	210.50s	226 (-8 ↓)	88.29 KB (-4.66 KB ↓)
8	219.74s	212 (-14 ↓)	82.81 KB (-5.48 KB ↓)

Slika 8. Testiranje sa jednim klijentom

Na slici 8 se mogu videti snapshoti koji su napravljeni prilikom testiranja igre jednog klijenta protiv bota. Osam različitih trenutaka u kojima su napravljeni snapshotovi predstavljaju:

1. Stanje inicijalnog pokretanja servera, pre nego što se pokrene klijent.
2. Stanje kada se klijent pokrene.
3. Stanje tokom kog klijent raspoređuje brodove.
4. Stanje nakon što je igra počela.
5. Stanje u toku igre.
6. Stanje nakon poslednjeg poteza u toku igre.
7. Stanje nakon završetka igre.
8. Stanje nakon što se klijent diskonektovao i napustio igru.

Pored ovih snapshotova napravljen je i test korišćenjem UMDH alata za praćenje memorijskog zauzeća. Rezultati se mogu videti u sklopu foldera Test2, gde postoje izveštaji o zauzeću na početku i na kraju rada servera zajedno sa izveštajem o njihovoj razlici.

Name ?	Exc % ?	Exc ?	Inc % ?	Inc ?	Fold ?	When ?	First ?	Last ?
OTHER <<kernelbase!SleepEx>>	59.9	206	59.9	206.0	0	1,621.822	220,453.274
OTHER <<ucrtbased!>>	14.2	49	14.2	49.0	0	1,166.507	216,795.690
OTHER <<ws2_32!>>	7.6	26	7.6	26.0	0	8,426.601	194,829.388
OTHER <<ntoskrnl!>>	6.1	21	6.1	21.0	0	25,956.217	220,092.115
OTHER <<kernelbase!Sleep>>	2.9	10	2.9	10.0	0	19,479.689	125,659.365
server!ProducerForClients	2.0	7	31.1	107.0	2	458.412	220,453.274
server!_CheckForDebuggerJustMyCode	2.0	7	2.0	7.0	0	458.412	188,556.967
server!SendToPlayer1	1.7	6	11.9	41.0	0	20,435.345	207,261.738
server!QueueCount	1.2	4	5.5	19.0	0	458.412	219,508.652
server!counterFunc	1.2	4	25.9	89.0	1	27,036.692	201,038.259
Process64 Server (6228)	0.3	1	100.0	344.0	1	458.412	220,453.274
server!Requests	0.3	1	2.6	9.0	1	5,390.321	188,540.971
server!SendToPlayer2	0.3	1	9.0	31.0	0	10,918.087	205,192.516
server!ServiceShutDown	0.3	1	14.0	48.0	0	1,166.507	216,795.690
Thread (7848) CPU=89ms	0.0	0	25.9	89.0	0	27,036.692	201,038.259
Thread (14728) CPU=108ms	0.0	0	31.4	108.0	0	458.412	220,453.274
ROOT	0.0	0	100.0	344.0	0	458.412	220,453.274
Thread (12396) CPU=49ms	0.0	0	14.2	49.0	0	1,166.507	220,092.115
server!_vfprintf_l	0.0	0	1.7	6.0	0	73,101.161	170,273.705
server!printf	0.0	0	1.7	6.0	0	73,101.161	170,273.705
Thread (4204) CPU=41ms	0.0	0	11.9	41.0	0	20,435.345	207,261.738
BROKEN	0.0	0	1.7	6.0	0	67,442.839	220,092.115
Thread (10680) CPU=8ms	0.0	0	2.3	8.0	0	11,311.097	155,702.190
server!RecvFromPlayer1	0.0	0	1.5	5.0	0	22,596.180	185,902.143
Thread (7540) CPU=34ms	0.0	0	9.9	34.0	0	10,918.087	205,192.516
server!RecvFromPlayer2	0.0	0	2.0	7.0	0	11,311.097	133,234.952
OTHER <<ntdll!RtlUserThreadStart>>	0.0	0	98.0	337.0	0	458.412	220,453.274
Thread (21092) CPU=9ms	0.0	0	2.6	9.0	0	5,390.321	188,540.971
Thread (21432) CPU=5ms	0.0	0	1.5	5.0	0	22,596.180	185,902.143

Name ?	Exc % ?	Exc ?	Inc % ?	Inc ?	Fold ?	When ?	First ?	Last ?
OTHER <<ucrtbased!>>	97.6	48,620	97.6	48,620.0	0	242113230341.0432.1434443	9,283.321	212,981.592
OTHER <<ntoskrnl!>>	1.2	583	1.2	583.0	0	00000000.0000.0000.00000000	9,371.024	212,985.247
client!main	0.7	356	15.0	7,449.0	356	243	9,260.336	25,961.387
client!userInputFunction	0.3	126	78.5	39,111.0	4	113230341.0432.1434443	31,296.290	206,699.610
Process64 Client (6916)	0.1	46	100.0	49,814.0	46	243113230341.0432.0.1434443	9,070.846	212,985.247
client!SendMessageToServer	0.1	41	78.6	39,152.0	38	113230341.0432.1434443	27,401.709	206,699.610
client!chooseGameType	0.0	16	14.2	7,093.0	0	243	9,283.321	24,982.299
Thread (7304) CPU=7531ms (Startup Threa	0.0	10	15.1	7,531.0	10	243	9,070.846	212,985.247
client!pressEnterToContinue	0.0	8	5.2	2,568.0	0	03	206,836.877	212,967.471
client!ReceiveMessageFromServer	0.0	6	5.2	2,574.0	6	03	45,443.759	212,967.471
OTHER <<ntdll!RtlUserThreadStart>>	0.0	1	98.7	49,178.0	0	24.113230341.0432.1434443	9,260.336	212,981.592
BROKEN	0.0	1	1.2	580.0	1	000000000000.0000.00000000	9,371.024	212,985.247
ROOT	0.0	0	100.0	49,814.0	0	243113230341.0432.0.1434443	9,070.846	212,985.247
Thread (7616) CPU=39637ms	0.0	0	79.6	39,637.0	0	113230341.0432.0.1434443	27,401.709	206,822.663
Thread (16852) CPU=2600ms	0.0	0	5.2	2,600.0	0	03	25,961.865	212,967.471
client!_sclr_common_main	0.0	0	15.0	7,451.0	0	243	9,260.336	212,981.592
client!mainCRTStartup	0.0	0	15.0	7,451.0	0	243	9,260.336	212,981.592
client!invoke_main	0.0	0	15.0	7,449.0	0	243	9,260.336	25,961.387
client!_sclr_common_main_seh	0.0	0	15.0	7,451.0	0	243	9,260.336	212,981.592

Za kraj je urađena i PerfView analiza, to jest prikupljanje podataka o radu procesa Servis.exe kao i Client.exe, pri igranju igre između dva igrača. Rezultati analize mogu se videti na slikama 9 i 10. Na njima se može videti da je potrošnja procesorskog vremena optimalna i očekivana. Podaci nastali analizom su veci od 70MB te zbog toga neće biti poslati zajedno sa projektom. Moguće ih je dostaviti na USB drive-u zajedno sa PerfView.exe aplikacijom radi detaljnije analize procesa tokom odbrane projekta.

Zaključak

Iz priloženih testova se može zaključiti da se memorijom rukuje na ispravan način i da nema curenja. Memorija se u programu zauzima dinamički u slučaju rukovanja queue-om. U slučaju da se memorija koja je dinamički alocirana na heap-u ne oslobodi došlo bi do curenja memorije, odnosno memorijske lokacije bi ostale zauzete konstantno.

Razlika izveštaja koja je dobijena testiranjem UMDH alatom i PerfView-om potvrđuje ranije tvrdnje.

Broj handle-ova u programu se menja onako kako je očekivano. Handle-ovi se kreiraju u slučaju pokretanja niti i nakon što se niti zaustave isti se brišu. To garantuje da u toku rada programa neće doći do nagomilavanja handle-ova, tj. da se iz partije u partiju broj handle-ova ne povećava već ostaje konstantan.

Korišćenje procesorskog vremena na serveru se razlikuje za slučajeve partije igrača i partije sa botom. U drugom slučaju je korišćenje veće. Ovakvo ponašanje je očekivano zato što se u slučaju partije jednog igrača sa botom server preuzima ulogu drugog igrača tj. bota. Korišćenje procesorskog vremena na klijentu je znatno veće nego na serveru što je i očekivano zbog kompleksnosti UI i potrebe za interakcijom sa korisnikom.

Rezultati dobijeni nakon testova su u granici očekivanih. Rezultat koji se može izdvojiti kao bolji od očekivanih je rezultat testa korišćenja procesorskog vremena na serveru, jer je vrednost iznenađujuće mala. Kao razlog za ovako dobar rezultat može se navesti dobra raspodela poslova između serverskih niti i korišćenje povoljne strukture za buffer-ovanje poruka.

Potencijalna unapređenja

Analizirajući kod došli smo do nekoliko zaključaka o budućim unapređenjima.

- Jedno potencijalno unapređenje može biti implementacija grafičkog korisničkog interfejsa. Igračima bi bilo omogućeno da sa drag&drop metodom postavljaju brodove kao i da sa klikom miša na protivničku tabelu gađaju brodove.
- Drugo potencijalno unapređenje može da bude da server omogućava više igra u isto vreme, to jest da je moguće da se više klijenata konektuje i igra u isto vreme.
- Kada pričamo o botu, tj. o igri protiv računara, kao unapređenje može se kreirati bot koji kada pogodi brod gađa mesta u okolini pogotka, kao i da kada pogodi brod preskoči kordinate oko broda jer se po pravilu tamo neće ništa nalaziti. Idealno unapređenje bota bi bilo da se koristi neki algoritam koji će mašinski učiti i vremenom postajati sve bolji i bolji. Nakon implementacije ovih unapređenja moguće je dodati nivoe težine igre protiv bota npr: lako, srednje i teško.