УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ИСПИТНИ РАД

Кандидат: **Михаило Марковић**
Број индекса: **RA191/2015**

Предмет: **Међурачунарске комуникације и рачунарске мреже 1**
Тема рада: **QT развојно окружење – Интерактивна децентрализована комуникација на LAN мрежи**

Ментор рада: **Проф. Илија Башичевић**

Нови Сад, Децембар 2018.

SADRŽAJ

SPISAK SLIKA

1. Zadatak

Realizacija P2P (peer-to-peer) desktop chat aplikacije korišćenjem QT frameworka, bez postojanja centralne tačke razmene poruka.

Klijent po startovanju mora da otkrije sve ostale klijente u odgovarajućoj podmreži. Otkrivanje aktivnih klijenata se može obaviti slanjem inicijalne poruke svakom računaru u podmreži, na zadatom portu. Zatim, ostvaruje konekciju sa svakim otkrivenim klijentom. Svaki klijent mora da poseduje ime, koje se prosleđuje kao parametar u komandnoj liniji.

Klijent odlazne poruke šalje svakom klijentu sa kojim je povezan. Poruke je potrebno ispisati na ekran, sa podatkom od koga je poruka dobijena. Odgovarajućim ispisom na ekranu potrebno je obavestiti korisnika o događajima kao što su pojava novog klijenta ili isključivanje klijenta.

2. Koncept rešenja

Rešenje je realizovano od nule, korišćenjem dokumentacije QtNetwork biblioteke i referentnog koda dostupnog na sajtu http://doc.qt.io/qt-5/. Prvo su formirani idejni MSC i SDL dijagrami na osnovu kojih je projektovan zadatak. Nakon toga je implementiran kod, koji je povremeno zahtevao izmenu osnovnih MSC i SDL dijagrama zbog specifičnosti QT frameworka.

Frontend je realizovan u QT Designeru i sastoji se iz dve klase, MainWindow i NicknameDialog. MainWindow prozor predstavlja glavni interfejs prema korisniku i predstavlja intuitivno rešenje po ugledu na nekadašnje online „pričaonice“ (npr. Krstarica). NicknameDialog je prozor koji je izveden od glavnog i on predstavlja prompt korisniku da unese ime koje će koristiti prilikom učestvovanja u sesiji.

Backend je realizovan korišćenjem QT Creator razvojnog okruženja. Sastoji se od četiri klase, Client, Server, Transmitter i Connection. Client klasa je „glavna“ backend klasa i sprega sa korisničkim interfejsom. Server klasa služi za otkrivanje i registrovanje dolazećih konekcija. Klasa Transmitter vrši kontrolu nad slanjem i primanjem broadcast datagrama, dok klasa Connection vrši parsiranje, kontrolu i slanje datagrama tokom normalnog toka komunikacije.

3. Opis rešenja

Frontend:

MainWindow:

MainWindow klasa je glavna GUI klasa i u njoj se vrši prikaz poruka, prikaz aktivnih korisnika, kao i polje za unos poruke . U ovoj klasi se instancira klasa NicknameDialog koja omogućava prikaz dodatnog prozora za unos korisničkog imena, kao i Client koja predstavlja spregu između prispelih podataka i GUI-ja. Ona koristi QObject::connect() metodu da poveže različite signale koji dolaze iz klasa NicknameDialog i Client. Ona sadrži i listu trenutno aktivnih korisnika koja se ažurira pomoću signala iz klase Client. Osim različitih getera i setera i pomoćnih metoda, ona sadrži i callback metode (slotove) za elemente svog grafičkog interfejsa kao i za reakcije na prispele signale. Takođe, ima svoj eventFilter koji ima svrhu „hvatanja“ pritiska dugmeta Enter na tastaturi, čime je preklopljena (overrideovana) metoda reakcije na Enter tako da ono sada vrši istu operaciju kao da korisnik klikne dugme Send u grafičkom interfejsu.

NicknameDialog:

NicknameDialog klasa je pomoćna GUI klasa koja omogućava prikaz dodatnog korisničkog prozora za unos korisničkog imena. Ono što je bitno napomenuti da se u njoj, pored potrebnih getera i setera, instancira i klasa Transmitter. To je bilo neophodno uraditi zbog ideje da se početak emitovanja broadcast signala i slušanja dolazećih signala odloži i da krene tek nakon što korisnik unese svoje korisničko ime. Na ovaj način je osigurano da će se to i desiti.

Backend:

Client:

Client klasa je backend klasa koja predstavlja spregu sa korisničkim interfejsom. Ona instancira klasu Server i povezuje signale newConnection koje očekuje iz klasa Server (dolazne konekcije) i Transmitter (odlazne konekcije). Ona instancira hešmapu *peers* koja kao vrednost drži objekat Connection, a kao ključ IP adresu te konekcije. Poseduje svoje metode newConnection() koja registruje signale konekcije spremne za obradu i prekida konekcije, connectionReady() koja na osnovu signala konekcije spremne za obradu vrši određene provere i ubacuje konekciju u listu peerova, removeConnection(), koja reaguje na signal disconnected(), kao i još neke pomoćne metode.

Server:

Ovo je vrlo jednostavna klasa koja služi da omogući početak osluškivanja ka dolazećim konekcijama. Ta operacija se izvršava u konstruktoru klase. Ova klasa nasleđuje ugrađenu klasu QTcpServer koja pruža metodu incomingConnection(), koja je u našem slučaju preklopljena tako da pravi objekat klase Connection koristeći poseban konstruktor koji služi za inicijalizaciju konekcije koji će biti objašnjen ispod.

Transmitter:

Klasa Transmitter pruža kontrolu nad konekcijama, kao i slanjem i primanjem handshake datagrama. Ona instancira QTimer koji omogućava periodično slanje handshake datagrama, QUdpSocket koji predstavlja socket za slanje i primanje datagrama, kao i liste broadcast (neke su IPv4, neke IPv6) i lokalnih adresa prispelih konekcija za vršenje određenih provera. Metoda sendDatagram() funkcioniše tako što otvara stream writer u koji upisuje niz veličine 2, koji se sastoji od stringa koji predstavlja username korisnika i njegov port na kom sluša. Metoda readDatagram() parsira dobijeni datagram tako što izvlači date informacije i koristi ih za provere tipa da li je datagram stigao od localhosta (samog sebe) ili da li ta konekcija već postoji. Takođe, ova klasa poseduje i metodu getAllAddresses, koja iz ugrađene klase QNetworkInterface poziva metodu allInterfaces(), koja vraća listu struktura svih interfejsa povezanih na server, i pomoću nje puni svoje liste broadcast i lokalnih adresa konekcija.

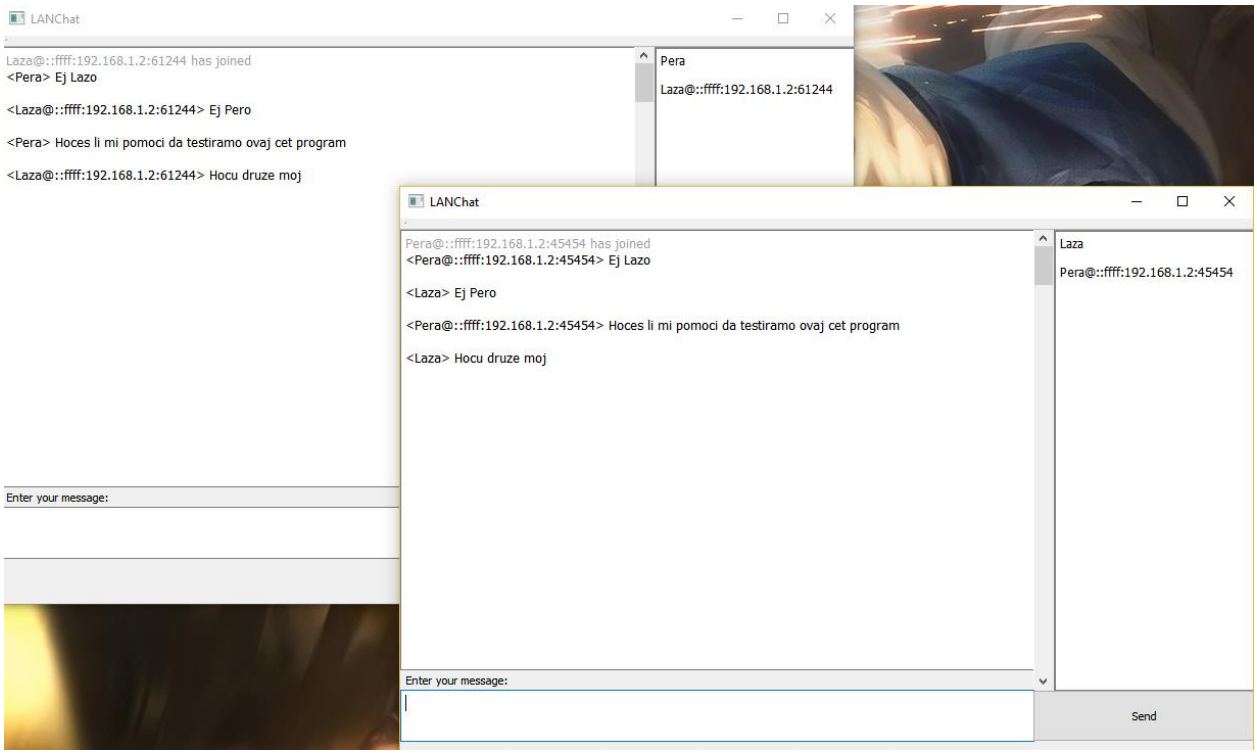
Connection:

Ova klasa je glavna funkcionalna klasa aplikacije. Ona definiše protokol komunikacije. Protokol ima tri stanja konekcije: *WaitingGreeting, ParsingGreeting,* i *ConnectionReady*. Njegovi datagrami mogu biti tipa *Message, Greeting* i *Undefined*. Protokol je osmišljen tako da, nakon što se uspostavi konekcija, šalje *Greeting* datagram ka drugoj strani koji nosi greeting poruku koja sadrži podatke o korisniku koji šalje datagram. Nakon što obe strane uspešno razmene Greeting datagram, stanje protokola se menja i postaje *connectionReady*, što označava da se nadalje očekuju standardni datagrami tipa *Message*. Najveći deo ove klase predstavlja metoda processReadyRead(), koja predstavlja parser sadržaja datagrama. Ona na osnovu trenutnog stanja konekcije i sadržaja datagrama, kao i dužinom sadržaja (sadržaj inicijalizacije konekcije u klasi Transmitter je niz dužine 2, dok se u klasi Connection podaci šalju kao mapa dužine 1, gde je ključ tip poruke *Greeting/Message*, a vrednost sadržaj) upravlja stanjima konekcije i parsira njen sadržaj u odgovarajući bafer. Nakon toga, na osnovu utvrđenog stanja konekcije, poziva metode processGreeting() odnosno processData().

4. Testiranje

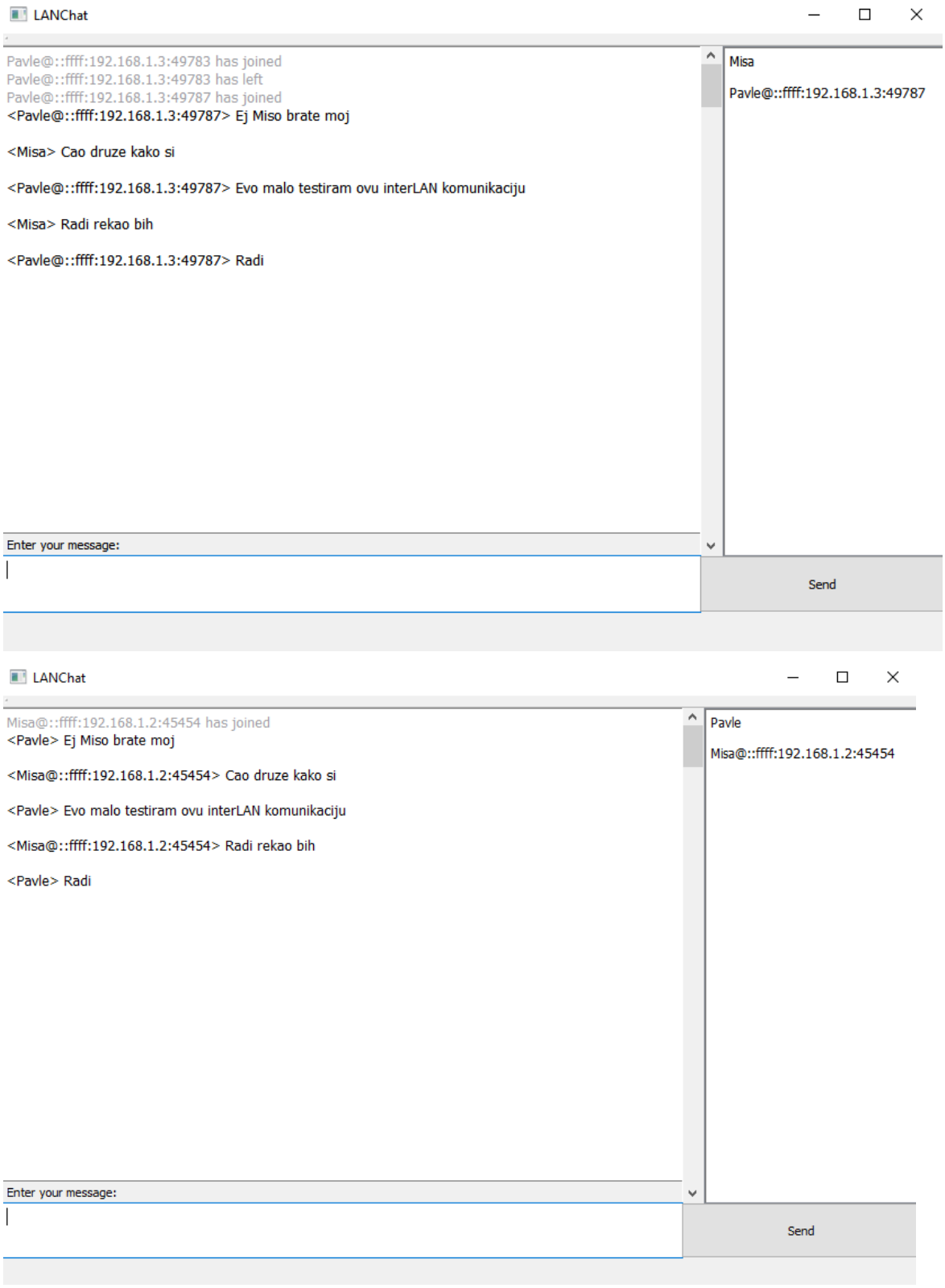
Program je istestiran simuliranjem različitih scenarija korišćenja aplikacije. Poenta aplikacije (razgovor sa drugim korisnicima na mreži), je dovoljno jednostavna da se samim korišćenjem aplikacije mogu uočiti potencijalni problemi.

U test situaciji u kojoj je autor programa najviše imao prilike da radi tokom faze razvoja istog, tj. u situaciji pokretanja više instanci programa na istom računaru, program radi kako je i namenjeno, bez uočenih bagova.



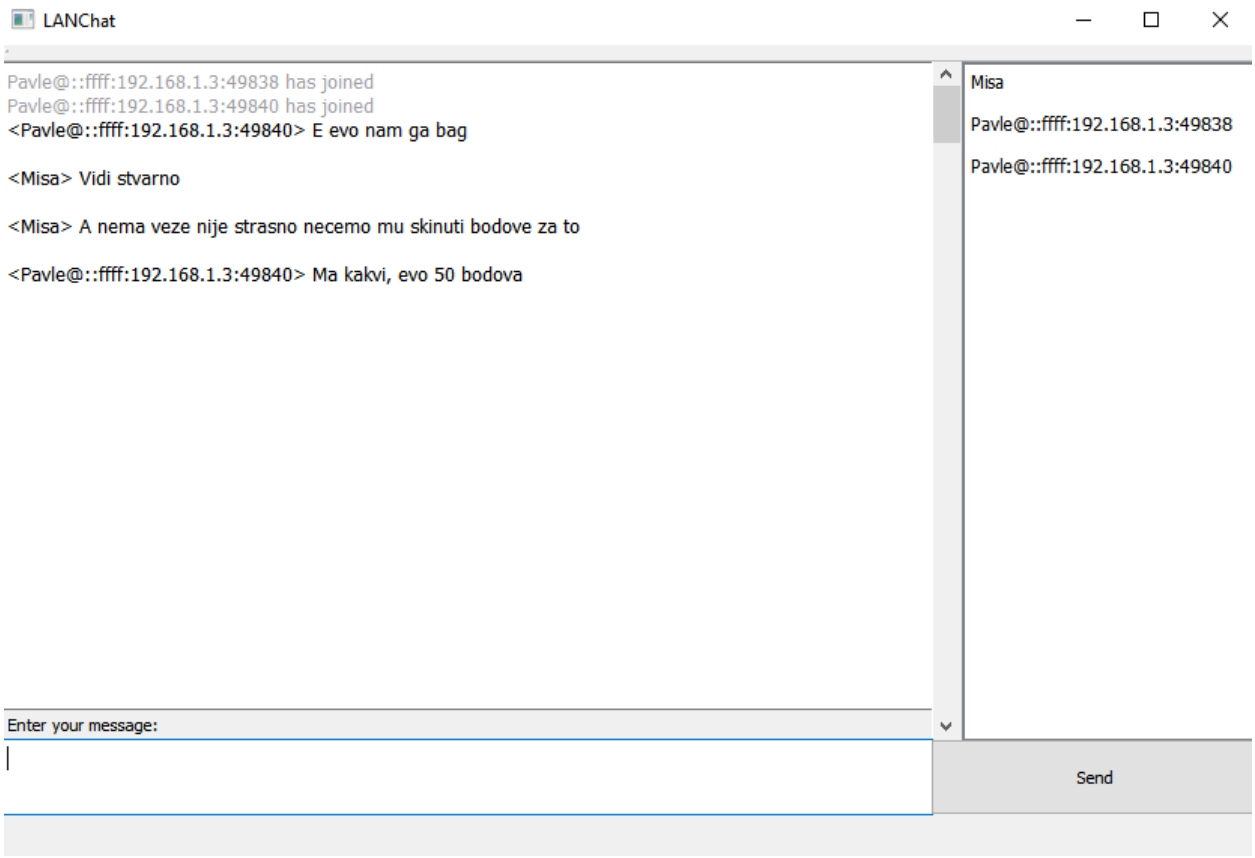
Slika 1. Dve instance aplikacije na istom računaru

Daljim testiranjem uočeno je da i u realnom use case-u, tj situaciji međuračunarske komunikacije na istoj mreži, program ponaša kako je zamišljeno bez ikakvih problema.



Slike 2 i 3. Realna situacija iz perspektive oba korisnika na različitim IP adresama

Autor je uočio, da se u određenim broju slučajeva u međuračunarskoj komunikaciji (ne pojavljuje se uvek) može pojaviti bag, koji se manifestuje tako što se jedan korisnik pojavi na dva porta što se i pokazuje u listi aktivnih korisnika sa desne strane, ali to ne utiče na realnu komunikaciju između klijenata. Autor pretpostavlja da je razlog tome to što je pretpostavljeni port koji je autor definisao za korišćenje programa u tom trenutku zauzet, pa sistem automatski dodeli novi port. Autor nije imao vremena da reši taj bag, tako da će to ostaviti narednim generacijama.



Slika 4. Bag jednog korisnika na više portova

5. Zaključak

Autor je izveo zaključak da je u najvećoj meri uspeo da ostvari zadatak koji je bio pred njim. Najveći izazov predstavljalo je učenje korišćenja QT frameworka i njegove QtNetwork biblioteke, ali je uspešno savladan. Izuzimajući retke bagove, program je funkcionalan i ispunjava svoju svrhu.

6. Literatura

[1] *Priručnik radnog okruženja za pisanje protokola, Verzija 0.2*, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2007