Almost a conference call

EDA095 - Nätverksprogrammering

Viktor Selleby	<tna12vse@student.lu.se>
Johan Nilsson	<dat12jn1@student.lu.se>
Christine Boghammar	<dat12cbo@student.lu.se>
Sava Pokrajac	<dic12spo@student.lu.se>

20 maj 2016

1 Bakgrund

1.1 Inledning

Kommunikation mellan datorer är idag en central del av samhället. Det känns för många som en självklarhet att spela spel, interagera via sociala medier, skicka meddelanden och bilder till andra människor över internet. Betydligt färre är däremot medvetna om hur den faktiska kommunikationen över nätverket fungerar. Den här rapporten beskriver ett program för telefonsamtal över nätverket där två eller flera personer kan kommunicera med varandra.

1.2 Syfte

Syftet med rapporten är att fördjupa förståelsen inom nätverksprogrammering samt att träna på problemlösning.

1.3 Teori

För att åstadkomma kommunikation mellan två enheter krävs en uppkoppling mellan dem. I projektet har detta lösts genom att ha ett serverprogram och ett klientprogram separat. Klienter kopplar sedan upp sig mot servern via en TCP anslutning. TCP är ett protokoll som används för att skapa och upprätthålla nätverksanslutningar över internet. Det används i samband med IP protokollet då det är IP som bestämmer hur paket skickas mellan datorer.TCP kännetecknas av det faktum att den kräver en anslutning för att upprätthålla kommunikation. Anslutningen skapas med hjälpa av en så kallad three way handshakeöch upprätthålls så länge någon av parterna har något kvar att skicka. Protokollet ansvarar även för att ta emot paket från applikations nivån, bryta ned dem till mindre delar som nätverksnivån kan hantera och för att acceptera paket från nätverkslagret. TCP är ett tillförlitligt protokoll på så sätt att alla paket kommer fram till destinationsapplikationen i rätt ordning. Det innebär att den ansvarar för omsändningar av tappade paket och strukturerar om paketen vid mottagarsidan ifall de hamnar i fel ordning. Eftersom TCP har hand om felhantering så tillämpar den även congestion control för att slippa massa omsändningar vid överbelastning. Fördelarna med TCP är att man kan vara säker på att innehållet kommer komma fram och att det kommer komma fram i rätt ordning. Nackdelen med TCP är att header-filen är större än hos UDP-pakten, samt att congestion control kan medföra en viss fördröjning i kommunikationen.

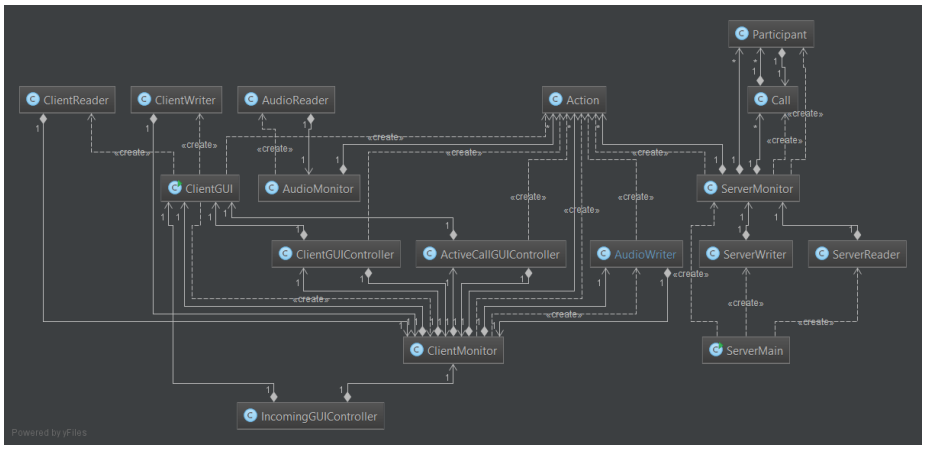
2 Kravspecifikation

De krav som sattes på programmet var i första hand att skapa och upprätthålla ett telefonsamtal över ett nätverk. Vidare ska programmet även kunna hantera gruppsamtal samt möjlighet att lämna ett meddelande om en uppringd person inte svarar. För göra funktionaliteten mer tillgänglig för användare sattes även ett krav på ett fungerande GUI.

Om programmet blev färdigt tidigare än utsatt deadline bestämdes även potentiella extrafunktioner som videosamtal och chatt.

3 Modell

Programmet innehåller fyra stycken paket; client, server, protocol och gui. Vilken funktionalitet som återfinns i respektive paket förefaller ganska tydligt men värt att nämna är att alla klasser i client, server och protocol är skrivna i Java kod medan gui är skrivet i JavaFX. Figur 1 nedan visar ett klassdiagram för programmet.



Figur 1: Java klassdiagram

3.1 Viktiga klasser

3.1.1 Action

Action är ett protokoll som styr all kommunikation mellan klient och server. En action innehåller bland annat ett command som beskriver vilket typ av data det är som skickas och hur server respektive klient ska tolka den data som skickas. Exempel på möjliga commands är: 0 (connect to server), 2 (initiate call), 7 (deny call) o.s.v.

3.1.2 ServerMain

Startar en server genom att skapa en ServerSocket för en given IP-adress och port. Skapar även ett ServerReader objekt och ServerWriter objekt för varje klient som ansluter. ServerReader lyssnar på data från klienten medan ServerWriter kallar på olika metoder i ServerMonitor beroende på vilket Action som mottas från klienten. ServerMain skapar även ett ServerMonitor objekt.

3.1.3 ServerMonitor

I ServerMonitor ligger majoriteten av all logik för kommunikation med de anslutna klienterna. Beroende på vilket command som Action innehåller kallar ServerReader och ServerWriter på olika metoder i ServerMonitor.

3.1.4 ClientGUI

ClientGUI innehåller main-metoden för klient delen av programmet. I ClientGUI skapas en anslutning till servern. Skapar även en ClientMonitor och GUI:t samt en ClientReader respektive ClientWriter tråd.

3.1.5 ClientMonitor

Precis som ServerMonitor står för stor del av logiken på server-sidan står ClientMonitor för stor del av logiken på klient-sidan. ClientReader lyssnar på data från servern vilken den sedan lägger på ClientMonitor. Beroende på vilket action som den mottagna datan innehåller kallar sedan ClientWriter på en specifik metod i ClientMonitor.

3.1.6 AudioMonitor

För att separera ljuddata skapades klassen AudioMonitor, tillsammans med AudioWriter och AudioReader. AudioMonitor spelar upp det ljudpaket som kommer från servern i högtalarna på datorn. För att åstadkomma detta används ”SourceDataLine” i Java Sound.

3.1.7 AudioWriter

AudioWriter tar indata från användarens mikrofon och skickar det till servern som en Action.

4 Användarhandledning

För att använda programmet krävs minst två personer som sitter vid varsin dator. Vidare krävs att antingen någon av de som ingår i samtalet eller en tredje part har server-delen av programmet igång med en känd IP-adress och port. För att ansluta till server startas ClientGUI med argument för serverns IP-adress respektive port.

Vid anslutning uppger klienten ett användarnamn för sessionen och ett GUI startas. I GUI:t visas sedan uppkopplade användare i en lista samt eget användarnamn. För att ringa ett samtal till en användare klickar man på användarnamnet och sedan på knappen call contact(s). För att istället ringa ett gruppsamtal markeras istället de användare som ska ringas genom att hålla in ctrl samtidigt som man klickar på de användare som ska ringas och sedan klickas återigen på knappen call contact(s).

En användare som blir uppringd får upp en pop-up ruta som visar vem som ringer. För att acceptera samtalet klickas knappen accept, och för att avvisa samtalet klickas knappen reject. Om samtalet accepteras visas en ny vy som innehåller en lista på de användare som ingår i det aktuella samtalet samt en disconnect knapp som används för att avsluta samtalet. I samtalsvyn finns även möjlighet att skriva ett chattmeddelande till de användare som ingår i samtalet.

Om en användare inte skulle vara vid uppringning finns det även möjlighet att skicka ett röstmeddelande till användaren. Detta görs genom att klicka på knappen record message i startvyn vilket öppnar en pop-up meny där ett röstmeddelande kan spelas in genom att klicka på knappen start.

5 Utvärdering

Den huvudsakliga funktionaliteten i programmet, d.v.s. möjlighet att upprätta ett samtal mellan två enheter fungerar enligt kravspecifikationen. Gruppen stötte däremot på problem när det kom till implementationen av gruppsamtal. Även om själva anslutningen fungerar gör det API som används för att spela upp ljud (Java Sound) att fördröjningen snabbt växer och efter bara några sekunder blir samtalet ogenomförbart.

Efter lång analysering av problemet valde gruppen tillslut att istället fokusera på att implementera röstmeddelande samt chattfunktionalitet i programmet istället. Eftersom både chattfunktion och röstmeddelande ingår i ”scopet” för uppgiften känner sig gruppen trots motgångarna med gruppsamtal nöjda. En sak som hade kunnat göras för att få gruppsamtalen att fungera är att byta ut Java Sound mot ett annat API för ljuduppspelningen.

Implementeringen av programmet har bidragit till en djupare förståelse inom nätverksprogrammering för samtliga gruppmedlemmar vilket gör att syftet med projektet är uppfyllt.

6 Programlistor

Programlistor återfinns på följande adress: `https://github.com/ChristineBoghammar/NetworkProject`