Rapport för Projekt, Nätverksprogrammering EDA095

Johan Månsson	<ama10jma@student.lu.se>
Emil Pettersson	<atf10epe@student.lu.se)>
Therese Kustvall Larsson	<dic13tla@student.lu.se>

Lunds Tekniska Högskola

2015-05-22

1 Bakgrund

Detta projekt är en del av kursen EDA095 – Nätverksprogrammering. Uppgiften var att tillämpa den kunskap som erhållits under kursens gång, för att implementera en applikation som till största del bygger på nätverkskommunikation. Valet av applikation och dess funktioner fick ske relativt fritt, med begäran att inte vara för omfattande. Utifrån dessa krav utvecklades SimpleChat, en enkel chattapplikation.

2 Kravspecifikation

Följande krav specificerades när arbetet med projektet inleddes. Dessa krav skulle implementeras i första hand.

- I det grafiska klientprogrammet kan användaren välja att starta en chattkonversation med **en** annan ansluten användare. Användaren markerar personen den vill prata med och därefter öppnas ett nytt chattfönster. För att starta en **gruppkonversation** går det att lägga till/ta bort andra anslutna användare i det befintliga chattfönstret.
- I chattfönstret ges möjlighet att skicka filer av olika format till de användare som inkluderats i chattkonversationen. Primärt ska filöverföringsfunktionen fungera i en konversation mellan två användare.

Ytterligare ett krav kunde implementeras i mån av tid.

- Säker anslutning mellan server och klient via SSL där certifikat används för att säkerhetställa identiteten för varje användare.

3 Modell

Funktionaliteten har delats upp i två paket - Server och Client. Båda sidor innehåller trådade avsnitt för att möjliggöra kommunikation på flera håll samtidigt. Nedan följer en kort beskrivning av de viktigaste klasserna i respektive paket.

På serversidan finns klassen **ClientThread**, som med hjälp av trådar låter flera klienter skicka meddelanden och ansluta till samma server. **ChatServer** innehåller mainmetoden som gör det möjligt att starta servern. Ytterligare en trådad funktion på serversidan innehar klassen **UpdateThread**, som uppdaterar klienternas listor över inloggade användare genom att skicka namnen på anslutna klienter via kommandon till samtliga anslutna användare. Två viktiga klasser är också **Mailbox** samt **MailboxReader**, som möjliggör att sända meddelanden till samtliga användare. MailboxReader är trådad, för att kunna sända ut meddelanden från Mailbox till alla klienter samtidigt - där sedan den klient som är avsedd för meddelandet kan visa det på sin skärm.

På klientsidan finns samtliga klasser för det grafiska användargränssnittet. Klasssen **ClientMain** innehåller mainmetoden och initierar det grafiska användargränssnittet tillsammans med ett ClientHandler objekt. Klassen **ClientHandler** tar både hand om klientens anslutning till servern, startar och

sparar aktiva chattfönster i en lista och tar i mot och filtrerar meddelanden skickade från servern. När klientprogrammet startar möts användaren först av ett fönster för anslutning till servern som representeras av klassen **ConnectPane**. Användaren fyller i ip-adress, portnummer och användarnamn och trycker därefter på ”Connect”. Klienten försöker därefter upprätta en anslutning till servern. Då användarnamnet måste vara unikt för att identifiera respektive användare inväntar klienten servern för att jämföra valt användarnamn med redan anslutnas namn.

PeoplePane innehåller en trådad funktion **UpdateList** som uppdaterar listan över inloggade användare i användargränssnittet automatiskt. För att starta en chattkonversation eller skicka en fil markeras personens namn först för att därefter trycka på antingen ”Start Chat” eller ”Send file”.

I klassen **ChatWindow** finns den trådade klassen **SubmitThread** som läser de meddelanden som användare skriver in i chattfönstret. Klassen **NewChatWindow**, som ärver av ChatWindow, skickar sedan detta meddelande till servern via output stream. För att anslutna klienter ska kunna filtrera mottagna meddelanden så taggas samtliga meddelanden som skickas med mottagarens och sändarens användarnamn. I klientens **ClientHandler** skrivs därför bara meddelanden som har taggats med användarens namn ut i respektive chattfönster.

När en användare vill skicka en fil kommer en ActionListener i **PeoplePane** i klienten att skicka ett kommando, innehållande informationen om vilken användare som vill skicka och vilken som ska ta emot, till servern. På servern kommer kommandot att tas emot i tråden **ClientThread** och där genereras en slumptal mellan 30001 och 40000. Ett kommando skickas till mottagarklienten, innehållande slumptalet. I mottagarens **ClientHandler** kommer användaren att tillfrågas huruvida den vill ta emot en fil. Om man väljer Nej"kommer det att skickas ett kommando till servern som i sin tur skickar ett kommando till avsändarens klient som ger användaren ett meddelande om att mottagaren inte vill ha filen. Om man väljer "Ja"kommer det att skapas en tråd **ReceiveFileThread** som väntar på en uppkoppling från avsändaren, användandes slumptalet som portnummer. Efter det skickas ett kommando (innehållande portnumret) till servern om att mottagren är redo. Servern skickar i sin tur kommando (med portnummer, och mottagaraddress) till avsändar-användaren. Där skapas en tråd **SendFileThread**, denna kommer att be användaren att välja en fil med hjälp av ett grafiskt fönster. Sedan ansluter den till mottagaren och skickar först ett meddelande innehållande filnamnet och sedan filen själv. Om avsändaren avbryter det grafiska fönstret utan att välja en fil skickas enbart meddelandet äborttill mottagaren. Mottagaren tar först emot filnamnet (alternativ äbort") och om det inte är äbortså tar den även emot filen och sparar denna i en mapp, efter att ha givit den ett eget namn, i det directory från vilket hen kör sin klient. Om det är den första filen som tas emot genom SimpleChat kommer denna mappen att skapas innan filen läggs dit.

4 Användarhandledning

Logga in: För att använda SimpleChat behöver programmen ChatServer och ClientMain köras. I fönstret som kommer upp ombeds användaren att fylla i IP-adress, portnummer och ett unikt användarnamn. Fyll i samtliga uppgifter och klicka på knappen ”Connect”, vartefter användaren loggas in.

Chatta: För att se en lista över andra inloggade användare, klicka på fliken ”People”. Härifrån är det enkelt att välja en person i listan och klicka på knappen ”Start chat” för att öppna ett nytt fönster och börja konversera. Olika konversationer med olika användare kan vara öppna samtidigt.

Skicka filer: För att sända en fil till någon, markera önskad person i listan och klicka istället på ”Send file”. Mottagaren tillfrågas om denne vill ta emot filen, i så fall kommer då poppa upp ett fönster hos avsändar-användaren, där önskad fil kan väljas. Användaren som tar emot filen kommer få den i en mapp i den directory från vilken hen kör clienten.

Logga ut: Om användaren önskar logga ut från chatten, är det bara att klicka på kryssrutan i övre högra hörnet på startfönstret.

5 Utvärdering

På grund av främst tidsbrist hann vi inte riktigt uppfylla samtliga krav som vi ställt i projektets början. Det går inte att starta en gruppkonversation genom att ansluta användare i det befintliga chattfönstret, som det står i kravspecifikationen. Desigen med taggade meddelande ställde till det här och dessutom var det svårt att komma fram till en snygg lösninng för användargränssnittet. Däremot har vi hunnit implementera de viktigaste funktionerna, såsom filöverföring och chatt med flera personer i parallella chattfönster. Det krav som stod att uppfyllas i mån av tid (SSL-anslutning) är därför inte heller tillgodosett, då detta skulle kräva ytterligare arbetstid.

6 Programlistor

Källkoden finns att ladda ner på projekthemsidan: *http://johanmansson.github.io/simplechat*