UDP Chat

Alexander Milton – b13alemi – 20 mars 2015

# Uppgiftsbeskrivning

Uppgiften tillhanda var att skapa en chattapplikation i Java som nyttjar UDP-protokollet (*User Datagram Protocol*) för att kommunicera mellan klienter med hjälp av en server.

# Implementation

I det skapade programmet finns funktionalitet som tillåter för en användare att skicka ett meddelande eller kommando genom ett grafiskt gränssnitt och få gensvar genom en server. Meddelanden distribueras till andra klienter som är uppkopplade till samma server och mottas i samma grafiska gränssnitt. Vissa kommandon hanteras och besvaras helt och hållet utav servern.

Programmet är i sin struktur baserad på TCP-programmet som skapades i Uppgift 2, trots att uppgiftsbeskrivningen specificerar att programmeraren bör utgå från sin UDP- programmet. Eftersom att TCP-programmet färdigställdes före detta program blev utgångspunkten den motsatta.

Klientapplikationen tar emot input från användaren och kompilerar ett meddelande med information genom så kallad *marshalling*. En sträng av information, internt avgränsad med specifika symboler, översätts till ett bytefält av data i form av ett datagrampaket. Detta paket innehåller information om mottagarens adress och port som tar emot inkommande meddelanden och skickas genom en socket, bunden till en port hos avsändaren.

På serversidan arbetar en tråd konstant med att ta emot och vidarebefordra meddelanden för behandling och distribution, en tråd kontrollerar att de inloggade klienterna fortfarande är uppkopplade och en tredje tråd itererar över listan av klienter för att plocka bort de som förlorat sin uppkoppling till servern.

För att kontrollera om en klient fortfarande är uppkopplad används en teknik som kallas *heartbeat* där ett meddelande skickas ut till samtliga meddelanden. Om meddelandet inte besvaras antas det inte ha kommit fram. För att en klient ska ses som frånkopplad måste totalt tio ”hjärtslag” ha missats av klienten. Då markeras klienten som frånkopplad och raderas ur systemet. Samma klient, eller en ny klient med samma namn, kan sedan kopplas upp mot servern på nytt utan problem.

## Sekretess

Ett skickat meddelande är endast en sträng med bytes, och således oläsligt för människor. Skulle dock meddelandet tas över av en extern part skulle en enkel konvertering från bytefält till sträng ge parten omedelbar tillgång till meddelandets information. Ingen kryptering sker, vilket gör att chattapplikationen har väldigt dålig sekretess och är opålitlig som kommunikationsformat för meddelanden med känslig information.

## Integritet

Om en extern part hade tillgång till den enhet eller det nätverk från vilket en klient eller server exekveras finns det en risk att denne skulle kunna kontrollera, stjäla, manipulera eller vidarebefordra meddelanden. Sådan enkel information som användarens adress och chattens aktiva port skulle möjliggöra för falska meddelanden att skickas till denna användare, förutsatt att meddelandet är korrekt formaterat. Ett felformaterat meddelande kan inte tas emot av varken en server eller kliet. Ett befintligt men svagt skydd mot angrepp mot programmets integritet.

## Felmodell

UDP är ett protokoll som inte kan garantera korrekt pakettransport. Försummelsefel (*omission failures*) innebär att paket kan råka dupliceras, komma fram i fel ordning eller förloras på vägen utan mekanik som skyddar mot detta, till skillnad från protokoll som TCP. För att accentuera utmaningen med försummelsefel finns en artificiell risk om 30% för automatiskt förlorade paket. I snitt var tredje meddelande kommer inte skickas korrekt från en klient eller server, och programmeraren måste motverka detta.

För att se till att ett meddelande kommer fram skickas meddelandet upprepade gånger fram tills ett bekräftelsemeddelande från servern mottagits. Om ingen bekräftelse kommit fram före det tionde försöket slutar klienten skicka meddelandet. Programmets felmodell förutsätter att ett meddelande behöver högst tio försök för att komma fram och stödjer inte en högre frekvens av överföringsfel.

Meddelandet är tidsstämplat. Tidsstämpeln lagras i klientkopplingen hos servern för att avgöra när det senaste meddelandet skickades från klienten. Nya meddelanden jämförs mot den lagrade tidsstämpeln för att se om meddelandet är gammalt eller upprepat. Om en bekräftelse om lyckad överföring inte når fram till klienten och samma meddelande upprepas kommer det inte att distribueras igen (likt *at-least-once*-semantik) då det skulle bryta mot strävan att motverka försummelsefel. Istället skickas en ny bekräftelse till klienten och meddelandet ignoreras, i enlighet med *at-most-once*-semantik.

# Reflektion