Laboration chattserver

Datakommunikation och internet

DV167

Marc Coquand
id14mcd
mcoquand@gmail.com

Oskar Olausson
id14oon
kullenoskar@gmail.com

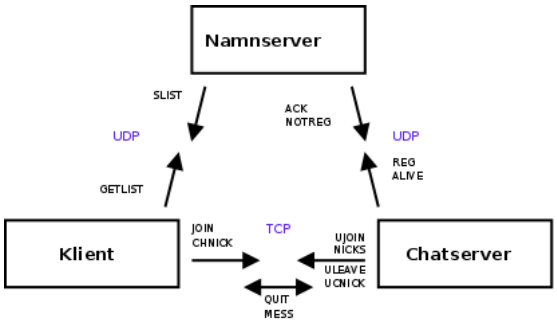
Handledare:
Niklas Fries
Thomas Johansson

9 oktober 2015

Innehåll

1 Systembeskrivning (från originalspecifickationen)

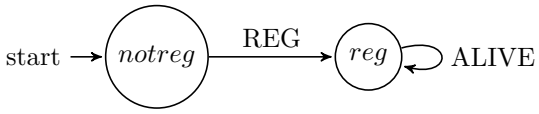
Kommunikationen i systemet består av tre delar: klient-server, server-namnserver och klient-namnserver. En översikt över kommunikationen visas i Figur 1. En kort förklaring för varje PDU finns i Tabell 1.



Figur 1: Översikt över protokollen mellan klient, chattserver och namnserver.

1.1 Server-namnserver

All kommunikation mellan chattserver och namnserver sker via UDP. När en server ska registrera sig hos en namnserver så skickar den först en `REG`-PDU. Som svar får den sedan en `ACK`-PDU med ett unikt ID. Därefter ska servern regelbundet skicka en `ALIVE`-PDU med samma ID. Om namnservern inte får någon `ALIVE`-PDU på 20 sekunder så avregistreras chattservern och ytterligare `ALIVE` kommer besvaras med `NOTREG`. Då ska chattservern registrera om sig med en ny `REG`-PDU. För att undvika att avregistreras ifall ett UPD-datgram försvinner, och då det kan ta lite tid för meddelandena att komma fram så kan ni skicka en `ALIVE`-PDU var 8:e sekund. För ett tillståndsdiagram över chattserver-namnserver-kommunikationen se Figur 2.



Figur 2: Tillståndsdiagram för kommunikationen mellan chattserver och namnserver. Alla giltiga PDU:er och deras respektive övergångar visas. Skickas en *ALIVE*-PDU när anslutningen är i tillståndet *notreg* så svarar namnservern med en *NOTREG*-PDU. Alla andra PDU:er ignoreras. Om ingen *ALIVE*-PDU skickas inom 20 sekunder så försätts kommunikationen i tillståndet *notreg*.

Namn	Syfte
Chattserver-namnserver	
`REG`	Skickas från en chattserver när den vill ansluta till en namnserver.
`ALIVE`	Skickas från chattserver till namnserver för att fortsätta vara registrerad.
`ACK`	Skickas som svar på `REG` eller `ALIVE`.
`NOTREG`	Skickas som svar på `ALIVE` om chattservern inte är registrerad.
Klient-namnserver	
`GETLIST`	Skickas från klient till namnserver för att få en lista med chattservrar.
`SLIST`	Svar på `GETLIST`, innehåller lista med chattservrar.
Klient-chattserver	
`JOIN`	Skickas från klient till chattserver för att ansluta till chatten.
`NICKS`	Skickas som svar på `JOIN`, innehåller lista med anslutna klienter.
`QUIT`	Skickas mellan klient och chattserver för att indikera stängd anslutning.
`MESS`	Chattmeddelande mellan klient och chattserver.
`UJOIN`	Skickas från chattserver till klienter för att indikera att användare anslutit till chatten.
`ULEAVE`	Skickas från chattserver till klienter för att indikera att användare lämnat chatten.
`CHNICK`	Skickas från klient till chattserver för att byta nickname.
`UCNICK`	Skickas från chattserver till klienter för att indikera att en användare bytt nickname.

Tabell 1: En kort förklaring av alla PDU:er och deras syfte.

1.2 Klient-namnserver

För att begära en serverlista så skickar klienter en `GETLIST`-PDU till en namnserver via UDP. Som svar kommer en eller flera `SLIST`-PDU:er, också detta via UDP. Eftersom ett UDP-datagram har en max-storlek på 65507 bytes så kan det skickas flera svar ifall servrarna inte ryms i ett datagram. Därför innehåller varje PDU ett sekvensnummer som börjar om på 0 för varje `GETLIST`-PDU som skickats. I klienten bör detta hanteras så att när en ny serverlista begärs så töms den nuvarande listan, och servrarna i de mottagna PDU:erna läggs till i listan

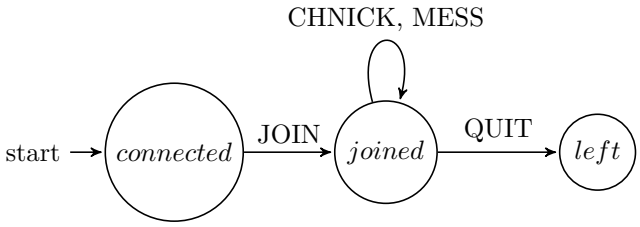
eftersom de dyker upp, så länge ingen PDU med det sekvensnumret tagits emot.

1.3 Klient-chattserver

Kommunikation mellan klient och chattserver sker via en TCP-anslutning. När anslutningen öppnats så skickar klienten en `JOIN`-PDU och får som svar en `NICKS`-PDU. Därefter kan klienten skicka `MESS` för att skicka meddelanden och `CHNICK` för att byta nickname. Dessutom kan anslutningen avslutas genom att skicka en `QUIT`-PDU. Alla andra PDU:er är ogiltiga och ska besvaras med en `QUIT`-PDU följt av att chattservern stänger anslutningen. För ett tillståndsdiagram över anslutningen mellan klient och chattserver ur serverns perspektiv, se Figur 3.

När en klient har anslutit till chatten så kan servern skicka `MESS` om någon annan klient skickat ett meddelande, `UJOIN` eller `ULEAVE` om en annan klient anslutit till eller lämnat chatten, `UCNICK` om en klient ändrat nickname samt `QUIT` om servern avslutas.

När en klient skickar ett meddelande till en chattserver så ska meddelandet inte ha någon tidsstämpel eller någon avsändare. Detta läggs till i chattservern så att klienter inte ska kunna ange ett annat namn eller en annan tid än den korrekta. När servern tar emot meddelanden så är det viktigt att de skickas till alla anslutna klienter i samma ordning.



Figur 3: Tillståndsdiagram för kommunikationen mellan klient och chattserver. Alla giltiga PDU:er och deras respektive övergångar visas. Alla ogiltiga PDU:er resulterar i att anslutningen övergår i tillståndet $left$ och anslutningen stängs.

2 Trådar

De flesta enklare program är ofta enkeltrådiga och körs sekventiellt på en tråd. Det vill säga att alla instruktioner som körs i programmet läses av en processor och att inga instruktioner körs samtidigt som någon annan. Detta är oftast lättare att implementera men har vissa begränsningar om man till exempel vill läsa in saker från användaren och kunna ta emot meddelanden samtidigt. Därför kan program vara flertrådiga så att en tråd ”tar hand” om inläsning samtidigt som en annan tar emot meddelanden. Implementation: I programmet som talas om i rapporten så används en del trådar.

I chatservern så används en tråd per ansluten klient, en för utskick till klienterna och en för utskick till namnservern. En tråd används också för att ta

emot nya klienter. I chatklienten så har vi en tråd för att skicka till chatservern och en för att ta emot meddelanden från chatservern. En extra tråd för att byta smeknamn (”nickname”) och en för att avsluta . Kopplingen till namnservern sker också på en separat tråd.

3 Användarhandledning

3.1 Server

För att köra igång servern krävs att man kompilerar Main-klassen. Som inparametrar tar den address och port.

- **Adress** Standard DNS-adress.
- **Port** Porten som servern skall lyssna på. Normalt sätt fyra siffror.

3.2 Klient

För att starta klienten så kompilera och kör mainmetoden i klassen ChatClientController. I klienten kan man skriva in address och portnamn till en namnserver, som innehåller alla tillgängliga chattservrar eller rum". Därefter för att ansluta till en server klickar man på den och trycker Join chat. För att ansluta krävs att man har skrivit in ett användarnamn.

4 Begränsningar

Servern klarar inte av att hantera inmatning av extremt stora meddelanden. Om en användare ändrar sitt namn till en tom sträng så kommer meddelandena han skickar skrivas ut som servermeddelanden.

5 Testprotokoll

5.1 Test 1

Syfte: syftet är att testa att användaren kan göra alla förväntade möjliga handlingar.

1. Starta namnserver som lyssnar på port 8082 och 8083
2. Starta server på “localhost” port: 8082
3. Lägger till tre automatiska klienter med namn Bot1 Bot2 och Bot3
4. Starta GUI:t och försök joina med namn Bot3, detta ska inte fungera.
5. Försök igen med namn Oskar, detta ska gå

6. Det namnet plus de tre Botarnas namn ska synas uppe till höger
7. Lägg till en användare med namn Marc, och byt tillbaka till Oskar fliken Marc ska då vara tillagd.
8. Skicka ett meddelande "Hej". Meddelandet ska dyka upp på de båda anslutna klienterna
9. Stäng ner Oskaranvändaren och den ska försvinna från listan på anslutna klienter som Marc har.
10. Byt namn på Marc till Oskar och det ska fungera, byt namn till Marc igen
11. Byt namn till Bot1 detta ska inte fungera

6 Diskussion

Samarbetet och arbetsfördelningen har varit bra. Samt har planeringen också gått bra. För oss var den givna koden ganska självklar men det är konstigt att det används lambda-funktioner när vi inte fått lära oss det sen tidigare.

De givna övningsuppgifterna var väldigt bra för att lära sig sockets och trådar. Problemet var att det var en total bortkoppling mellan det som gicks igenom på föreläsningarna och själva laborationen.

Inget har varit så svårt, däremot lade vi mycket tid åt debugging. Det som var oklart var hur man skulle använda UTF_8 för att formatera strängarna. Det var också oklart hur man skulle konvertera datumen så att millisekunderna togs bort. Vi hade också problem när vi skulle göra övningsuppgifterna att det försvann tecken när vi skickade via netcat.