Projektrapport

MegaLoad

Nätverksprogrammering
EDA095

2012-05-16

Av:
Pétur Hjartarson, ic08phj@student.lth.se
Jelena Mirosavljević, mat08jmi@student.lu.se
Elise Ellerstedt, adi09eel@student.lu.se
Ann-Louise Andersson, adi09aan@student.lu.se

1. Bakgrund

Denna rapport är en dokumentation av projektet inom kursen “Nätverksprogrammering” vid Lunds tekniska högskola. Syftet var att utveckla någon form av nätverkskommunikation. Vårt program möjliggör fildelning för användare som är medlemmar i en specifik multicastgrupp. Medlemmarna uppger vilka av dess filer som är tillgängliga för nedladdning. Vid överföring direktkopplas filägaren till mottagaren med en TCP-anslutning.

2. Kravspecifikation

Programmet ska vara baserat på en multicastgrupp där användare kan ansluta för att bli medlemmar och därmed ange vilka filer de vill dela med sig av. Alla medlemmarna i gruppen kan se vilka filer som finns tillgängliga. Programmet ska klara av att stödja både uploads och downloads. Vid filöverföringen ska mottagaren direkt kopplas till filägaren. Överföringen sker via en TCP-anslutning mellan sändaren och mottagaren.

2.1. Krav

- Vid en ny medlemsanslutning ska multicastgruppen informera vilka filer som finns tillgängliga
- Vid ett nytt filtillägg ska alla medlemmar inom multicastgruppen informeras
- Vid filöverföring ska mottagaren ha en TCP-anslutning till sändaren

3. Modell

Varje applikation fungerar som en server och en klient som skickar iväg sin egen IP-address till Multicast molnet, detta är endast synligt inom dess egna router (alternativt samma subnät). Varje applikation kör sedan två Multicast trådar, en för att ta emot IP-addresser och synliggöra dom för användaren och en för att skicka iväg sin egen. Om en användare inte skickar en Multicast inom loppet av 5 sekunder räknas den användaren som bortkopplad.
När man har IP-addresser synliga under "The Cloud" fliken, kan man välja att synliggöra en användares fillista (vilken är lagrad i en fil som heter shared.txt som skapas i samma mapp som programmet körs). När man sedan tagit emot fillistan kan man välja vilken fil man vill ladda ner. Man skickar då iväg sökvägen för den valda filen och den andra användaren skickar den önskade filen. När filen är överförd, stängs fillistan.
Under fliken "Upload" väljer du dem filer du vill dela. Dessa sökvägar kommer vara synliga för den andra användaren som begär fillistan.

Programmet består av flertal klasser från att representera själva filöverföringen till att skapa ett användarvänligt gränssnitt. Vi har valt att inte förklara GUI-klasserna eftersom de är relativt självklara för programmerare. De är inte heller särskilt relevanta för kursen.
Här kommer en kort beskrivning av de klasserna som vårt program består av:

3.1. MultiCom

Denna klassen kopplar upp användaren mot multicastgruppen genom att skapa en *MulticastSocket* och går med i gruppen. Vår socket är bunden till porten 3333. Klassen innehåller även metoder för att hantera användare så som *sendUserData()*, *setUsers()* och *getName()*. För att spara användare använder vi en *ConcurrentHashMap<K,V>* där *key* består av användarens IP-adress och *value* är tiden i systemet för en användare. Metoden *sendUserData()* hämtar en användares IP-adress och *setUsers()* lägger till användaren i hachmapen. I denna klassen finns också metoden *getFirstNonLoopbackAddress()* som används för att hämta datorns IP-adress.

3.2. MCRecieve

MCRecieve extendar trådklassen *Thread* vars metod *run()* har ersatts till att kontinuerligt anropa metoden, från föregående klass, *setUsers()* på *MultiCom*-objekt.

3.3. MCSend

Även denna klass extendar *Thread*. *Run()* utför metoden *sendUserData()* från *MultiCom*, väntar en viss tid och upprepar sedan metoden. Denna procedur repeteras oändligt.

3.4. Time

Denna klass behandlar tidsformat. Det finns två metoder *getDate()* och *getTime()* som anropas i klassen *MultiCom* för att få fram värdet på *value*.

3.5. DirectCom

DirectCom är klassen som representerar filöverföringen. Metoden *addFile()* lägger till en ny fil i användarens fillista. *SendFileList()* skickar en lista över vilka filer som är tillgängliga hos sändaren till mottagaren. Den valda filen hämtas med *getFile(),* omvandlas med *Converter* och överförs sedan med *sendFile()*.

3.6. DCServer

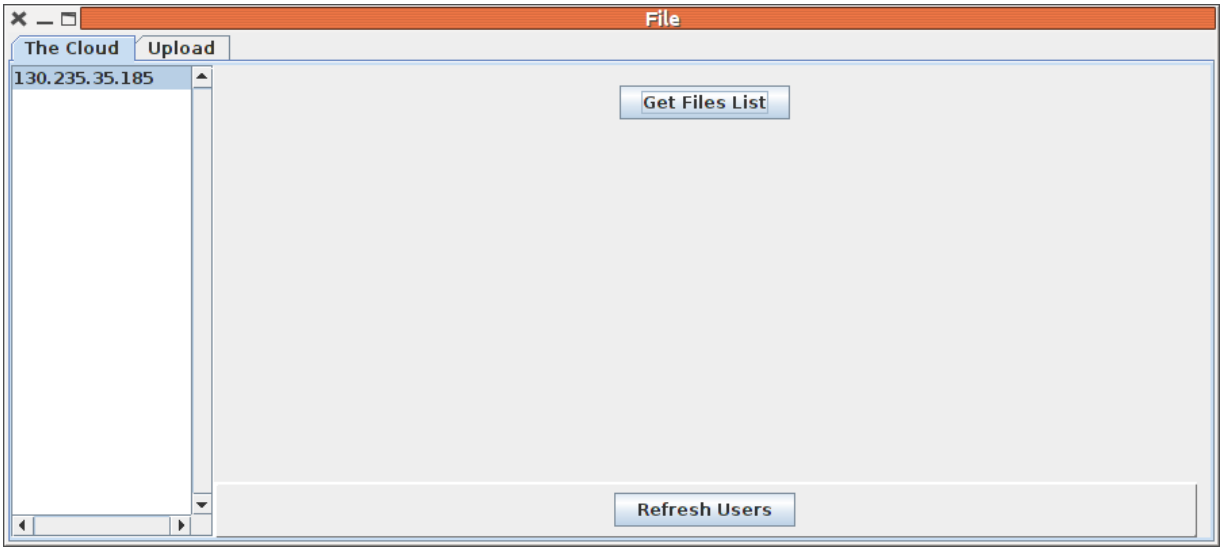
DCServer använder trådar. I *run()* körs de metoderna som implementerats i *DirectCom*. Varje program består också av en server. Då den mottager ett meddelande som börjar på “f” ser han att det är en fil som begärs, han väntar på att filnamnet ska skickas från klienten och kan sedan skickar över denna. Om meddelandet däremot börjar med ett “l” så är det fillistan mottagaren efterfrågar.

3.7. Converter

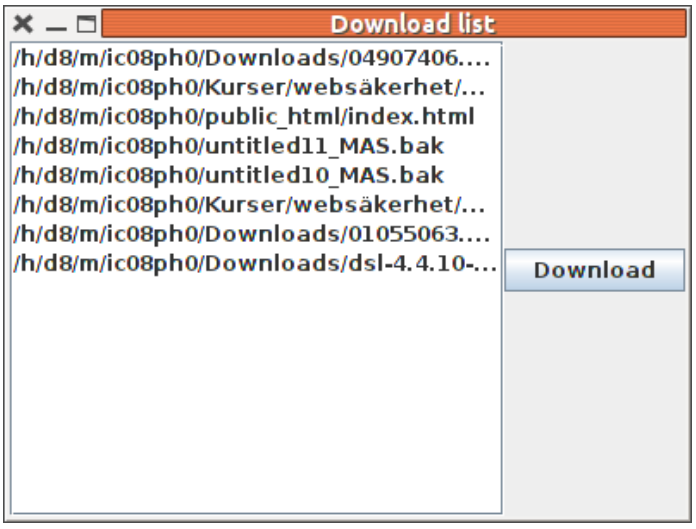
Denna klassen används för att omvandla filnamn av datatypen *String* till *byte* och tvärtom. Detta för att man ska kunna skicka filerna över kommunikationskanalen. Klassen innehåller metoderna *FiletoByte()* och *BytetoFile()*. Innan en fil skickas anropas metoden *FiletoByte()* och när den mottagits omvandlas den tillbaka med metoden *BytetoFile()*.

4. Användarhandledning

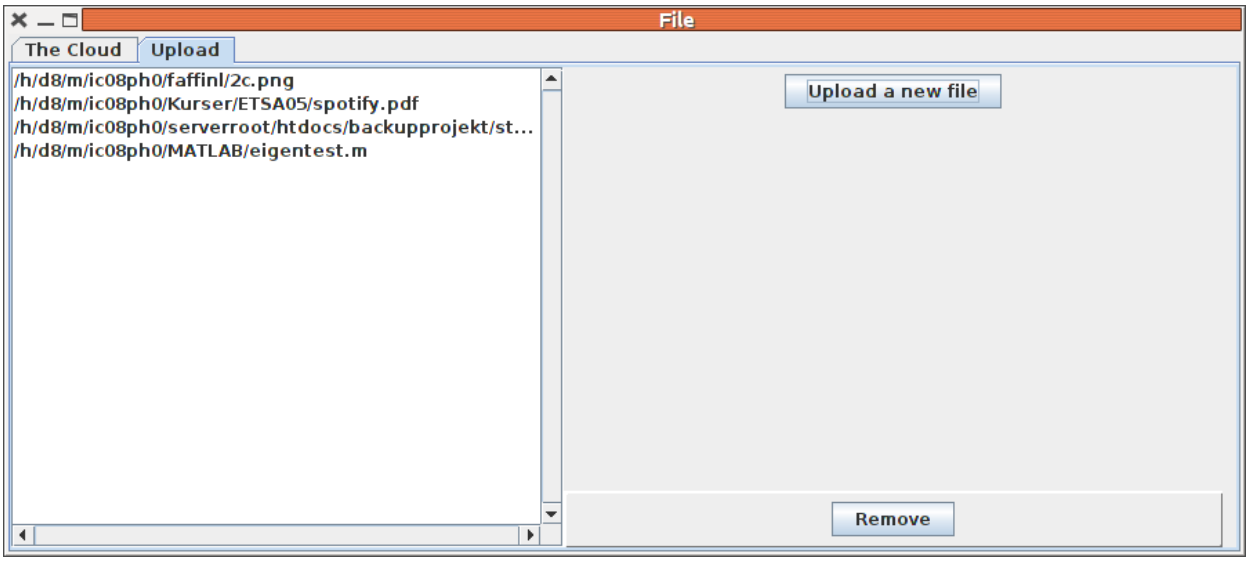
När man kör programmet kommer man till GUI:t vilket ser ut som nedan.



Från början visas inga användare under “The Cloud” fliken (de användare som är anslutna till gruppen). För att uppdatera listan och få fram de som är anslutna måste man trycka på knappen “Refresh Users”. Varje användare visas i listan med sin IP-adress. Genom att välja en användare i listan och sedan trycka på knappen “Get Files List” får man upp ett nytt fönster med alla de filer den användaren har laddat upp. Du väljer den filen du vill ladda ner och trycker på “Download” för att ladda ner filen till din dator.



För att ladda upp dina egna filer väljer du fliken “Upload”. Genom att trycka på knappen “Upload a new file” kan du välja vilken fil du vill ladda upp. Vill du ta bort en fil du inte längre vill ladda upp, markerar du filen och trycker på knappen “Remove”.



5. Utvärdering

Eftersom vi satte upp krav som var relativt grundläggande för att programmet skulle fungera så lyckades vi uppfylla alla. Dock har vi omformulerat kravet att medlemmarna ska meddelas då en ny fil läggs till. Istället så måste användaren trycka på "Get Files List" i GUI:t för att få reda på vilka filer som är tillgängliga just då. Denna procedur bör upprepas varje gång användaren vill ladda ner för att få uppdaterad information om vilka filer man kan komma åt.

En svårighet vi stött på under projektets gång var att ta reda på hur stor bufferstorleken ska vara för inkommande filer. Först använde vi färdigskapade *byte* vektorer med en bestämd storlek, vilket inte fungerade som vi ville. Tillslut hittade vi en algoritm som skapar en *byte* vektor som kan byggas på. När man har läst in hela filen med *InputStreamreader()* vet vi att *byte* vektorn är färdig och vi kan konvertera om den till en fil med *Converter* klassen.

Projektet var väldigt givande och relevant för kursen. Vi utvecklade nya kunskaper inom projektutveckling, grupparbete samt programmering. Vi upplevde att strukturen och omfattningen av uppgiften var väl vald.

6. Programlistor

Källkoden finns att hitta på hemsidan: http://users.student.lth.se/ic09aa3/EDA095/