

Assignment 2a (*Networking*)

Oliver Makins MKNOLI003, Masixole Ntshinga NTSMAS016, Daniel Schwartz SCHDAN037

April 7, 2017

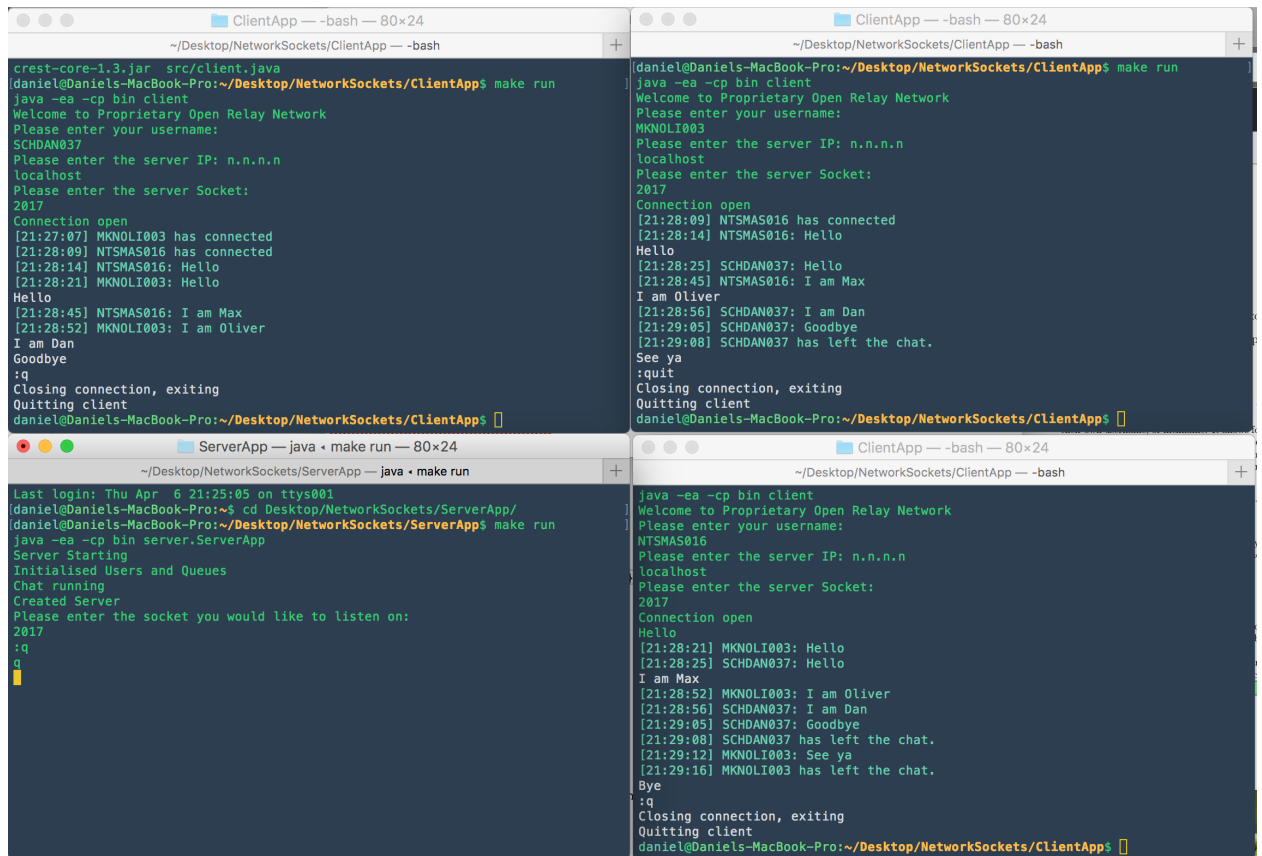
1 Summary

The web chat application we designed for this assignment implements features that most chat applications use today. It is a simple chat room that supports multiple users, which each have their own username, or nickname. It allows for users to transmit messages over the chat application through the chat server. It records the times and who sent each message.

The chat application runs off of a server which each user client connects to. As a user connects it notifies the other users that that person has connected. And the same for disconnection of users.

2 Features

2.1 Multi-user messaging



The figure displays four terminal windows arranged in a 2x2 grid, illustrating the operation of a chat application. The top-left window shows the ServerApp running, with output indicating it is listening on port 2017 and receiving connections from MKNOLI003, NTSMAS016, and SCHDAN037. The top-right window shows a ClientApp instance where a user enters 'localhost' as the server IP and '2017' as the server socket, successfully connecting and sending a 'Hello' message. The bottom-left window shows another ClientApp instance where a user enters 'localhost' as the server IP and '2017' as the server socket, successfully connecting and sending a 'Hello' message. The bottom-right window shows a third ClientApp instance where a user enters 'localhost' as the server IP and '2017' as the server socket, successfully connecting and sending a 'Hello' message. The ServerApp window also shows the messages received from the clients, including 'Hello' and 'I am Max'.

Figure 1: 4 terminal windows showing the server and 3 clients working

As one can see in Figure 1, the server can connect multiple users at the same time and one can see when, and who is sending each message as they come through. To connect: first the ServerApp must be initialized –

```
make run
```

– and a port chosen to open up to listen for clients to connect on. As the ClientApps are initialized – again using the makefile – the user chooses a username, the server IP (n.n.n.n format or localhost) and the port number that the server is listening on. Then as seen in Figure 1, The users can message each other and have all the messages sent to all clients. To quit a user needs only type either

```
:q  
or  
:quit
```

The server and all clients will be notified that the user is disconnecting and the client will close those ports and data streams and quit the process.

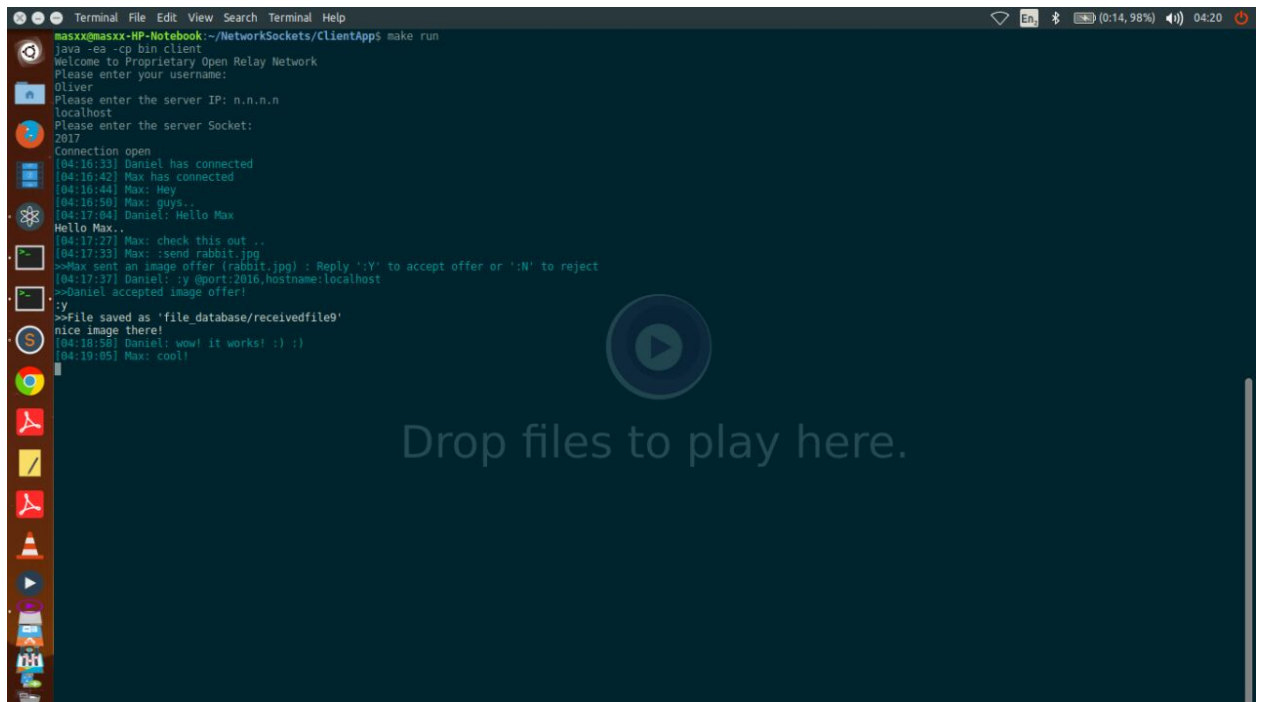


Figure 2: Terminal showing client user named Oliver

```

Terminal File Edit View Search Terminal Help
maxxx@maxxx-HP-Notebook:~/NetworkSockets/ClientApp$ make run
java -ea -cp bin client
Welcome to Proprietary Open Relay Network
Please enter your username:
Daniel
Please enter the server IP: n.n.n.n
localhost
Please enter the server Socket:
2017
Connection open
[04:16:42] Max has connected
[04:16:44] Max: Hey
[04:16:50] Max: guys..
Hello Max
[04:17:20] Oliver: Hello Max..
[04:17:27] Max: check this out ..
[04:17:33] Max: :send rabbit.jpg
>>Max sent an image offer (rabbit.jpg) : Reply 'Y' to accept offer or 'N' to reject
:Y
>>File saved as 'file.database/receivedfile0'
[04:17:44] Oliver: :y @port:2016,hostname:localhost
>>Oliver accepted image offer!
[04:18:31] Oliver: nice image there!
wow! it works! :) :)
[04:19:05] Max: cool!

```

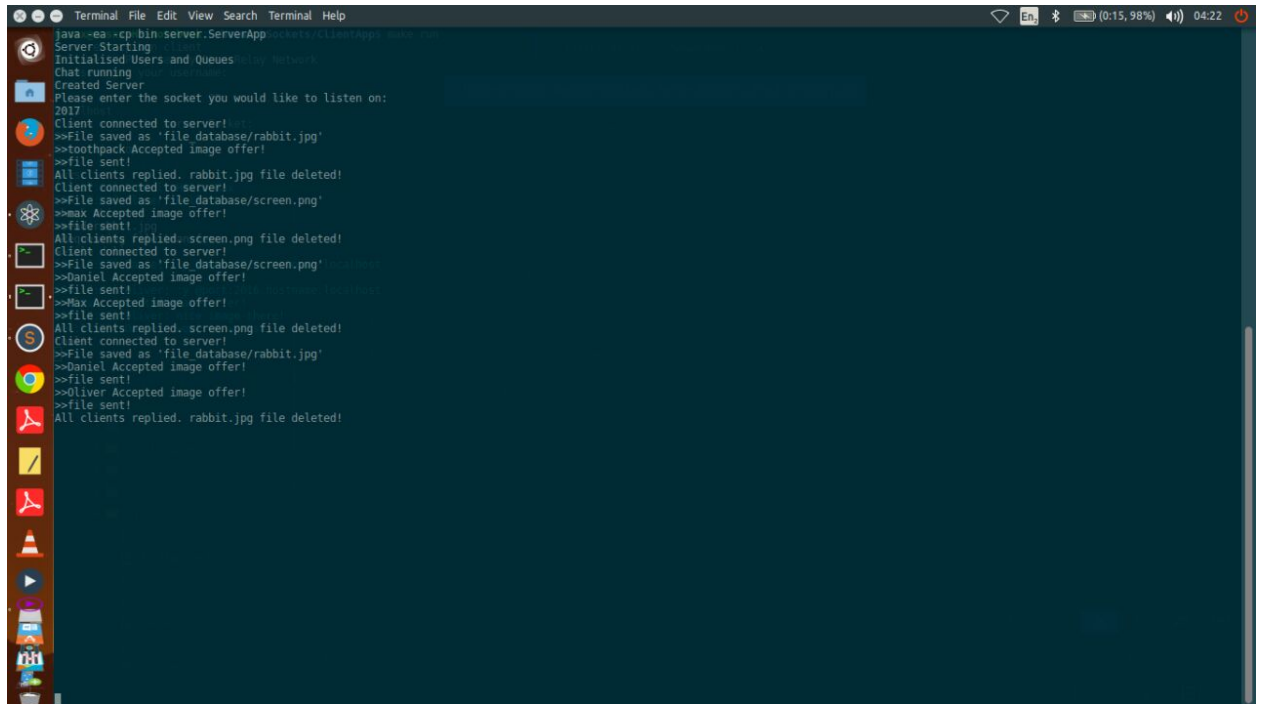
Figure 3: Terminal showing client user named Daniel

```

Terminal File Edit View Search Terminal Help
maxxx@maxxx-HP-Notebook:~/NetworkSockets/ClientApp$ make run
java -ea -cp bin client
Welcome to Proprietary Open Relay Network
Please enter your username:
Max
Please enter the server IP: n.n.n.n
localhost
Please enter the server Socket: screen.png
2017
Connection open
Hey
[04:17:04] Daniel: Hello Max
[04:17:20] Oliver: Hello Max..
check this out ..
:send rabbit.jpg
>>Requesting file transfer ...
>>file delivered to server!
[04:17:37] Daniel: :y @port:2016,hostname:localhost
>>Daniel accepted image offer!
[04:17:44] Oliver: :y @port:2016,hostname:localhost
>>Oliver accepted image offer!
[04:18:31] Oliver: nice image there!
[04:18:58] Daniel: wow! it works! :) :)
cool!

```

Figure 4: Terminal showing client user named Max

A screenshot of a Linux terminal window with a dark blue background. The window title is 'Terminal' and it has standard menu options (File, Edit, View, Search, Terminal, Help). The terminal displays the output of a Java application. It starts with 'Server Starting', followed by 'Initialised Users and Queues', 'Chat running', and 'Created Server'. It then prompts for a socket to listen on, and shows a series of log messages: 'Client connected to server!', 'File saved as file database/rabbit.jpg', 'toothpeck Accepted image offer!', 'file sent!', 'All clients replied. rabbit.jpg file deleted!', and so on, repeating for 'screen.png' and 'rabbit.jpg' with different client names like 'Max', 'Daniel', and 'Oliver'. The system tray at the top right shows network, battery (0:15, 98%), and time (04:22). The left sidebar shows various application icons.

```
java -ea -cp bin server.ServerApp:server.ClientApp:server...
Server Starting
Initialised Users and Queues
Chat running
Created Server
Please enter the socket you would like to listen on:
2017
Client connected to server!
>>File saved as 'file database/rabbit.jpg'
>>toothpeck Accepted image offer!
>>file sent!
All clients replied. rabbit.jpg file deleted!
Client connected to server!
>>File saved as 'file database/screen.png'
>>Max Accepted image offer!
>>file sent!
All clients replied. screen.png file deleted!
Client connected to server!
>>File saved as 'file database/screen.png'
>>Daniel Accepted image offer!
>>file sent!
>>Max Accepted image offer!
>>file sent!
All clients replied. screen.png file deleted!
Client connected to server!
>>File saved as 'file database/rabbit.jpg'
>>Daniel Accepted image offer!
>>file sent!
>>Oliver Accepted image offer!
>>file sent!
All clients replied. rabbit.jpg file deleted!
```

Figure 5: Terminal showing the Server running

2.2 Image transferring

The applications also accommodated for image transferring between users. The user could write a command,

```
:send <filename>
```

to send a file to the server. That image file will then be sent to the server over the network completely. Then the server will notify all the users that there is an image available to be downloaded. The users may accept or decline the image using the command ':Y' or ':N', respectively.

Once all the users either accept or decline the image, except the client that sent the image, the image is deleted off the server. In other words, for N clients, once the number of answered clients is N-1 it deletes the image off the server. The Server keeps a log of the image transfer stages. This approach was chosen to keep the server lightweight and maximise user privacy by mitigating the risks associated with retaining data.