Net2

Setup Instructions

1. Navigate to the textmessenger folder.
2. Enter the following commandline instruction javac *java
3. Enter the following commandline instruction java Main

    a. If you want to specify a username instead type java Main <username>, where <username> is replaced with the desired identifier.

Design

I have completed all the basic requirements as well as extensions A1, A2 and A5.

Discovery

In order to make the program discoverable by other instances of the program on the network I had to follow the protocol outlined in the specification and use a multicast socket. This meant that any other computers on my current network that were subscribed to the same multicast address as my would receive any packets I sent to it.

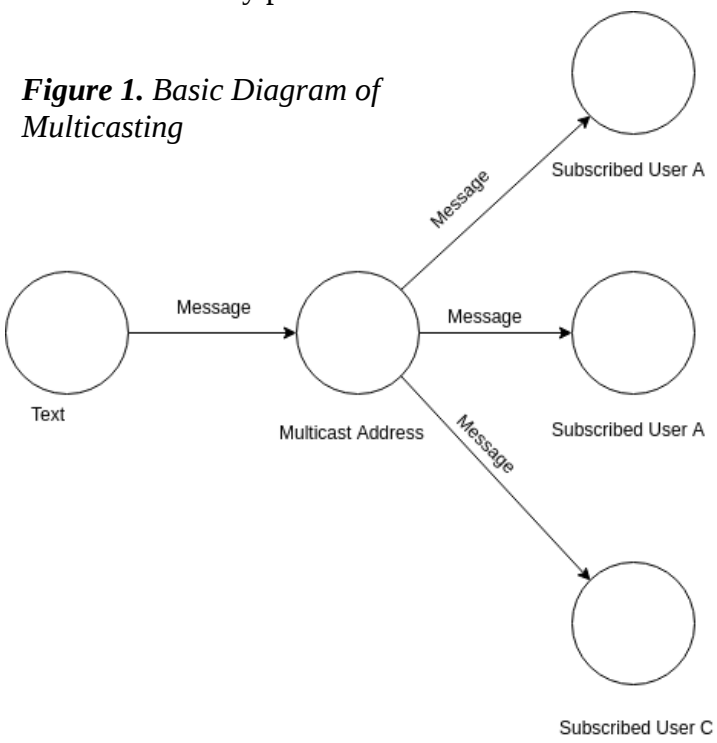


Figure 1. *Basic Diagram of Multicasting*

To discover other users I continuously receive any datagrams on the network for 5 seconds before sending my own beacon packet out. These datagrams are validated to check they match the formattting protocol given in the specification before the user information is stored.

The disadvantage to this method is that if a user follow the message format protocol but sent out a beacon more than once every 5 seconds the GUI will show their name multiple times.

Messaging

When another user is discovered on the multicast address, their information is stored so that the current user can message them.

There is a separate thread continually waiting to receive a message from another user. When it receives a connection, it only waits a short time before timing out as the protocol specifies that only one message needs to be sent before the connection is closed.

Format Validation

To ensure that the messages received are in the proper format I use regex expressions. While this doesn't properly check whether any of the values provided makes sense (e.g. that the timestamp is accurate and not fake) it at least forces a certain amount of ordering on the messages sent, which makes handling the information being sent much easier to manage.

Security

When using a multicast socket there is nothing to stop an individual that knows the address to connect and listen in on any packets sent to the group. In addition as there is no verification of identities, there is no way of no knowing for certain that a username belongs to a specific individual without physically meeting them, as many users can share the same username.

Implementation

Discovery

In the Users class, I use the system time in milliseconds to continually receive any packets sent on the network for 5 seconds, this is so that I receive beacons whenever they are sent.

To store the users information I use a hashtable using the username as the key and a 2-d String array containing the address and portnumber the user provided in their beacon. I chose a hashtable in order to minimize lookup time for the the user details. So that the amount of data stored is minimized, I first validate the beacon received to check it followed the protocol given in the specification before storing it. I also only store users that transmit 'online' and should a user go offline, or their beacon displays them as offline, they are removed from the hashtable.

The getPortNumber and getAddress methods are both synchronized because if a particular user sends another beacon with different information as they are being accessed by a different thread, it could result in abberant behaviour. By making these methods synchronized, threads will have to wait till after these methods are finished before accessing the userDetails hashtable.

Messaging

The Messages class has a separate thread with a ServerSocket continually in the accept state waiting for incoming connections. Upon a successful connection it waits for a message to be sent for only 100 ms. This is to minimize the likelihood of another user trying to connect to the ServerSocket while it is waiting for a message from the user.

To check that a message is coming from an online user, the run method checks that the username maps to a non-null value in the userDetails hashtable in the Users class. This should work as only users that are currently online should have values associated with their username stored.

When it attempts to send a message, the send message method attempts to retrieve the values associated with the username from the hashtable of online users. If that value is null, then the user must either be offline or not have given a beacon in the correct format. In this case the socket will throw an error and not send any message.

The Messages class has a static function getOnlineStatus so that it can easily pass the state of the checkbox from the Messages GUI to the Users class.

The port number the ServerSocket is hosted on is stored only in the Users class. When the ServerSocket is being initialized it uses the public method getPortNumber. This is to minimize the duplication of data, so if the port number has to be changed it only needs to be changed in one place, reducing the likelihood of errors occuring.

Testing

Test 1 – Detect Beacons

Test Brief: The program should be able to read packets from a Multicast socket. If those packets contain information in the correct format showing another user is online then it should be displayed on the GUI.

Input: Run program for 10 seconds.

Expected Result: Console shows all packets on network and identifies badly formatted messages. And well formed beacons that show users online should mean that that user is displayed on the GUI.

Result: Successful. Figure 2 and Figure 3.

Figure 2. *Beacons on the Multicast Socket*

```
[20181112-195402.927][js395][online][pc5-027-l.cs.st-andrews.ac.uk][21638]
[20181112-195403.767][edjs][online][138.251.29.192][51234]
[20181112-195404.410][amtht][online][pc2-120-l.cs.st-andrews.ac.uk][51140]
[20181112-195406.875][md225][online][pc3-046-l.cs.st-andrews.ac.uk 138.251.29.208][20605]<-- bad format
[20181112-195407.523][ma247][online][138.251.29.185][55598]
[20181112-195407.929][js395][online][pc5-027-l.cs.st-andrews.ac.uk][21638]
[20181112-195408.768][edjs][online][138.251.29.192][51234]
[20181112-195409.412][amtht][online][pc2-120-l.cs.st-andrews.ac.uk][51140]
[20181112-195411.876][md225][online][pc3-046-l.cs.st-andrews.ac.uk 138.251.29.208][20605]<-- bad format
[20181112-195412.525][ma247][online][138.251.29.185][55598]
[20181112-195412.931][js395][online][pc5-027-l.cs.st-andrews.ac.uk][21638]
[20181112-195413.769][edjs][online][138.251.29.192][51234]
[20181112-195414.415][amtht][online][pc2-120-l.cs.st-andrews.ac.uk][51140]
[20181112-195416.877][md225][online][pc3-046-l.cs.st-andrews.ac.uk 138.251.29.208][20605]<-- bad format
[20181112-195417.527][ma247][online][138.251.29.185][55598]
[20181112-195417.933][js395][online][pc5-027-l.cs.st-andrews.ac.uk][21638]
```

Test 2 – Sending Beacons

Test Brief: Other users should be able to detect any beacons sent out by my program. The beacons sent out should be in the correct format.

Test: Check that beacon is received on another program.

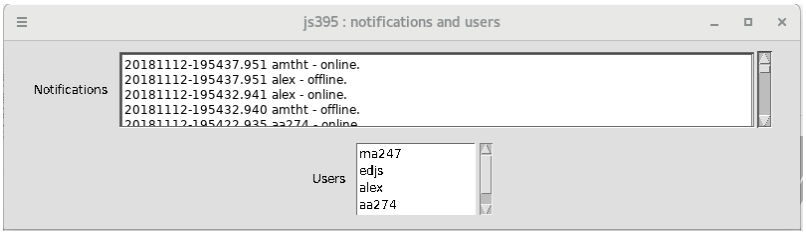


Figure 3. *GUI of online users*

Result: Successful. Figure 4 – js395 is my terminal.

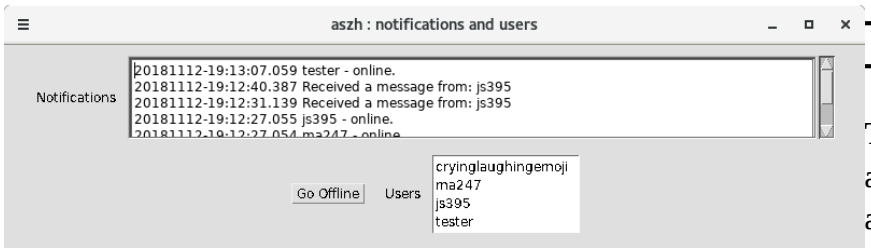


Figure 4. *GUI from another user*

Test 3 – Sending/Receiving Text

Test Brief: The application should be able to send and receive text from another user as long as that user is online.

Expected Result: Messages should be sent/received and notifications should be displayed in notification bar.

Result: Successful. Figure 7a and b. Figure 8a and b. Figure 8b, shows notifications being correctly displayed for the messages sent.

Test 4 – Receiving Messages from offline users

Test Brief: If a user is not currently online, then any messages from them should not be displayed.

Input: Receive a message from a user not currently online (buddy).

Expected Result: The message is not displayed on the GUI, the console shows that messages have been sent but not displayed.

Successful: Buddy is offline and messages they sent were not displayed. Figure 5.

```
[20181112-200334.963][buddy][offline][pc3-059-l.cs.st-andrews.ac.uk][21166]

[20181112-200335.547][alex][online][pc3-066-l.cs.st-andrews.ac.uk][20959]
[20181112-200336.217][aa274][online][pc3-062-l.cs.st-andrews.ac.uk][21037]
[20181112-200336.280][ma247][online][138.251.29.185][55598]
[20181112-200337.092][amtht][online][pc2-120-l.cs.st-andrews.ac.uk][51140]
User that is not online has sent you a message, message was discarded.
[20181112-200339.007][js395][online][pc5-027-l.cs.st-andrews.ac.uk][21638]
User that is not online has sent you a message, message was discarded.
[20181112-200339.963][buddy][offline][pc3-059-l.cs.st-andrews.ac.uk][21166]
```

Figure 5. Console snapshot.

Test 5 – Extension: Display as offline

Test Brief: The user should be able to display themselves as offline while still seeing which users are online.

Input: Set the ‘Online’ checkbox to off.

Expected output: The Console should display the beacon as showing ‘offline’.

Result: Successful. Figure 6. The beacon being sent displays my applications as online (line 2)

```
[20181112-201940.030][alex][offline][pc3-066-l.cs.st-andrews.ac.uk][20959]
[20181112-201940.061][js395][offline][pc5-027-l.cs.st-andrews.ac.uk][21638]
[20181112-201941.543][amtht][online][pc2-120-l.cs.st-andrews.ac.uk][51140]
[20181112-201943.046][aa274][online][pc3-062-l.cs.st-andrews.ac.uk][21037]
```

Figure 6. *Console snapshot of Beacon*

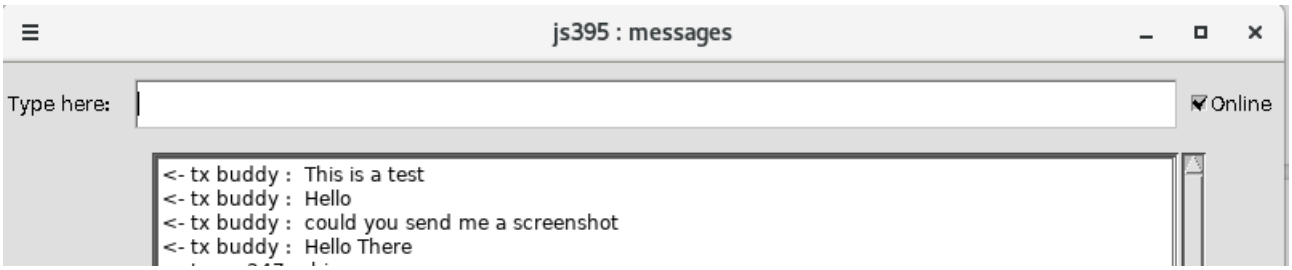


Figure 7a. *My GUI while chatting with user ‘buddy’*

Figure 7b. *GUI of user 'buddy'*

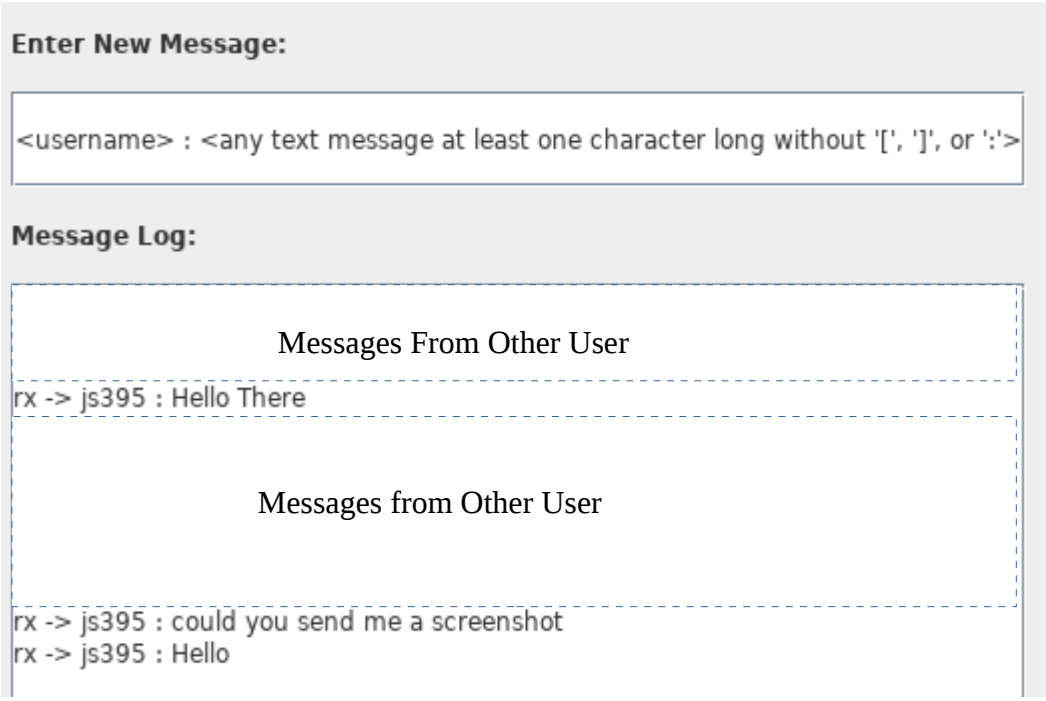


Figure 8a. *GUI of user 'alex'*

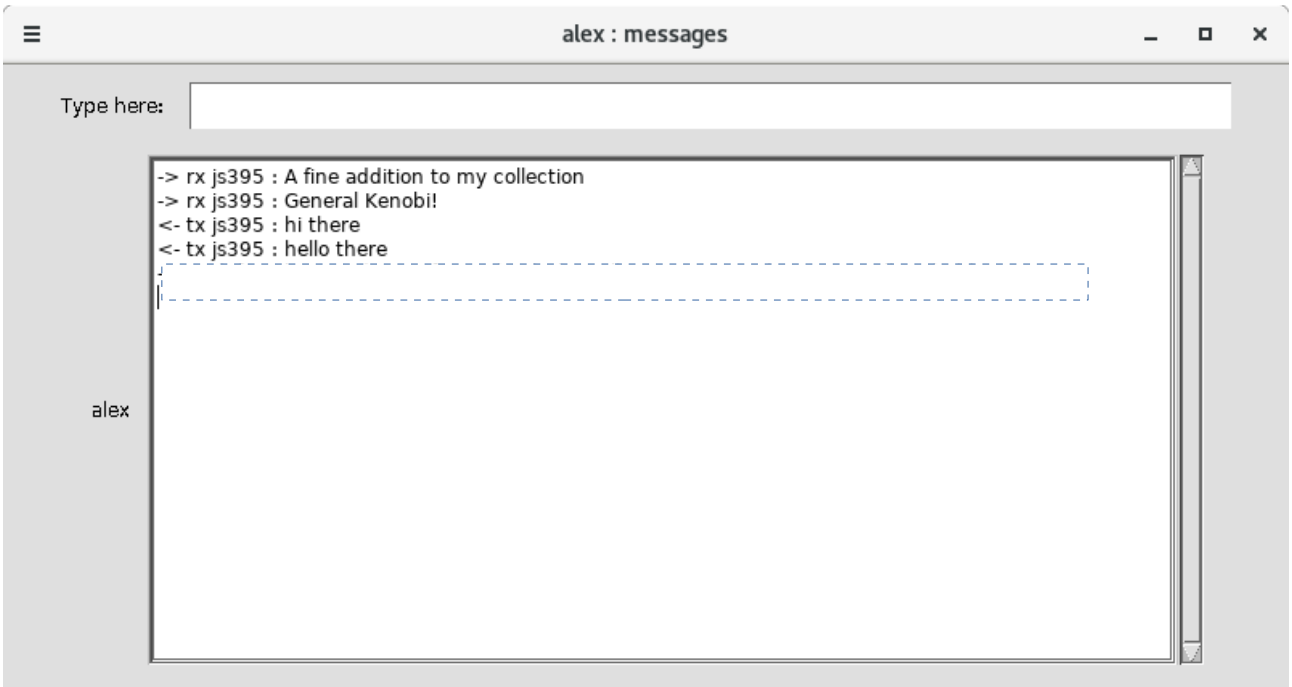
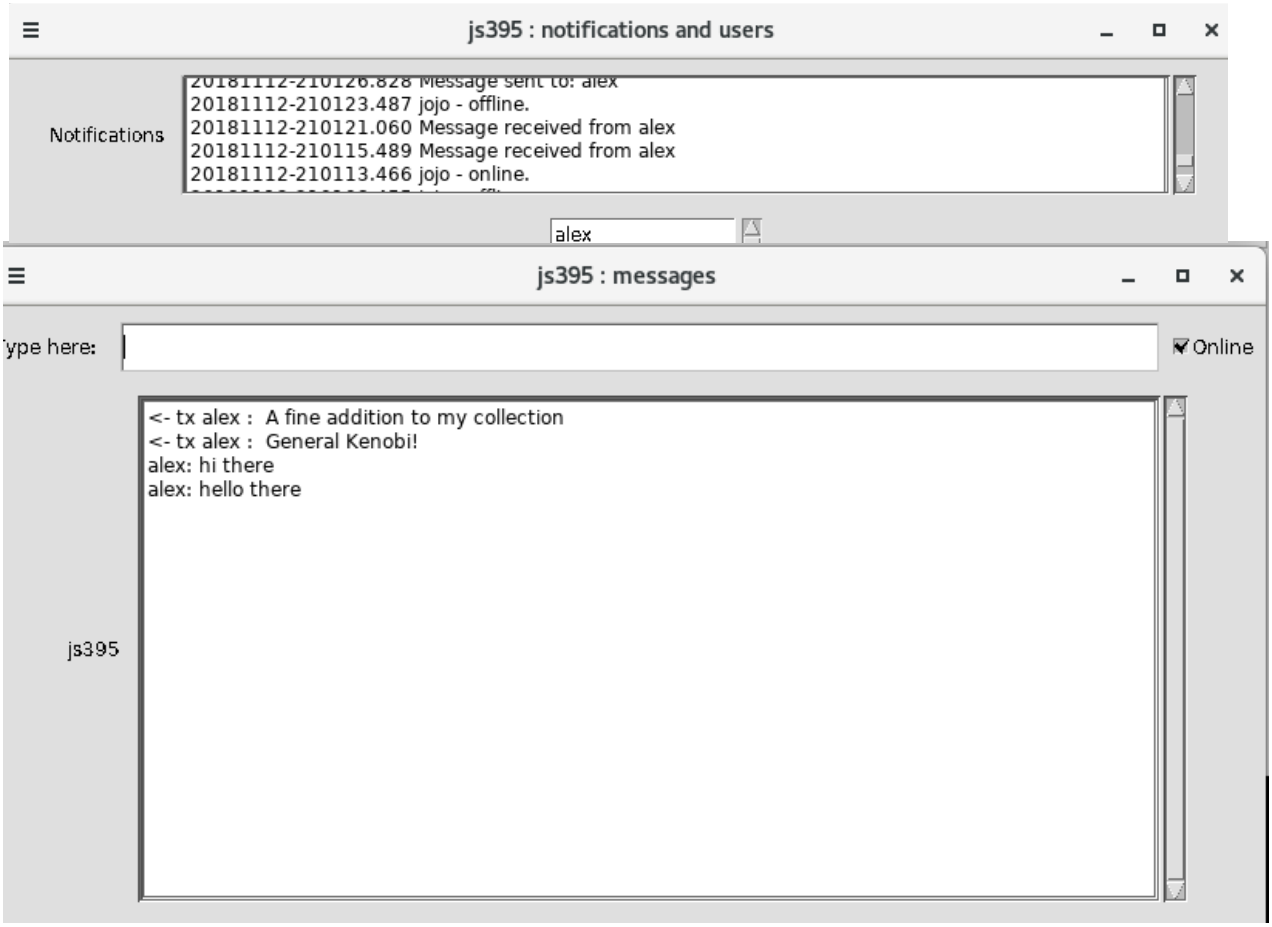


Figure 8b. *My GUI while chatting with user ‘alex’.*



Acknowledgements

Matriculation No. ‘buddy’ - 160027001

Matriculation No. ‘alex’ - 170006583

Base Code Provided by: https://studres.cs.st-andrews.ac.uk/CS2003/Practicals/Net2/

When designing and implementing the discovery functionality(the Users. class), I used the following code and modified it for my own implentation: https://studres.cs.st-andrews.ac.uk/CS2003/Examples/wk07/heartbeat/HeartBeat.java