Design Document

Network Technologies – Final Assignment

### Luc McAuley – G003883057

# High level overview of submission.

At its most basic form, this application is a UDP based chat application. Using datagram packets to send the chats across to the server and clients, it is easier to implement, offers faster delivery and is suitable for small data packets such as chat messages.

To offset a downfall of UDP packets, namely the fact that packets are not guaranteed to be delivered, a check is performed on the client side. Once a server receives a message, it is sent back to the sending client and verified with what was sent.

Through the use of arrays, the server keeps track of users, identified through their port numbers. A TCP socket is created in order to block that socket from other users and applications but no packets are transmitted through TCP.

# How to use the application.

## Server application.

Once the application is started, the server application scans all of the ports from 1 to 65535 with used ports displayed to the user for their information. This section of the application is adapted from the LowPortScanner.java code provided in week 9 of the lectures.

The user is then asked to select the port they would like to use. If valid, the server remains active and is listening on the given port. Any received message is displayed on the console. If a user leaves or their connection drops, a list of active chat users, identified by their port number is displayed on the console. This list is maintained by an integer array, stored in the PortManagement class.

## Client application.

Once the application is started, all of the ports from 1 to 65535 are scanned, with used ports displayed to the user for their information. This section of the application is adapted from the LowPortScanner.java code provided in week 9 of the lectures.

The user is asked to enter the server port they would like to connect to as well as the port they would like to use. If the selections are valid and the client connects to the server, a message is displayed to the user to let them know they have connected.

The messages send from the client are sent to the server which then sends it to all other active users. Each user is identified by their port number, prefixed to each message displayed on the console.

When a user wishes to leave the chat app, they can send “/q” to the server. The user is then asked if they would like to restart the application. Other users receive a message indicating the user, identified by their port number, has left the chat.

# Checks performed

## Common to both applications.

When selecting port numbers, the user’s input is checked to be at least 1 and less than 65535, the range of available ports. A TCP connection is then created at this port in order to block it. If an error occurs, such as a server is already present at that port, the user is unable to use this selected port and must enter a new one.

## Server application.

When a message is received by the server each saved port is checked to see if there is still an active TCP connection at that port. If the connection is not detected, the port is removed, the other users are informed and the client’s application is closed if it is still receiving UDP packets.

## Chat application.

When a message is sent to the server, it is expected to be sent back to the client. If it isn’t received back, 100 total attempts to send the datagram packet are made. If the expected message is not received, the user is informed that the message failed to send and to try again. Additionally, the user is informed if there is an active server on that port to let them know if the connection dropped.

If the initial message is not received back, but there is an active port, the user is informed that although there is a server at their selected port, it may not be the chat server application and they are prompted to restart the application.

# Known issues.

Unfortunately, due to time constraints, there are some known issues that have not been addressed.

* When a client application sends “/q” and restarts, the client port they initially selected is not closed until they select a new client port. Although the ServerSocket is closed when exiting the application, the TCP connection appears to still be alive.
* When a client server has connected with a server which subsequently disconnects, the user cannot quit the application, even if typing “/q”. The only way to exit the chat application is to use Ctrl+C.
* A deprecated method is used to close threads in the client application.
* There is no way of shutting down the server gracefully. The only option is Ctrl+C which kills the application.

# Sources.

The Port Scanner used in both applications was taken from the Week 9 lecture application LowPortScanner.java.

Documentation and examples of sending and receiving datagram packets:

https://www.geeksforgeeks.org/working-udp-datagramsockets-java/