CSC3002F NETWORKS ASSIGNMENT 2021

Socket Programming Project

Authors
Christopher Tooke (TKXCHR001)
Cameron Misplon (MSPCAM001)
Max Weihe (WHXMAX001)

Introduction

This assignment requires the creation of a chat application coded in java. This application runs off a server locally or globally and receives messages from a client and sends those messages to another client that is also connected to the server. These messages are sent using the UDP protocol. These transactions occur at real-time.

Protocols

What is UDP:

User Datagram Protocol (UDP) is a connectionless protocol. This means that a message can be sent to the server without first establishing a connection or an agreement to send data, as is conducted in a TCP connection. This protocol is therefore faster to send as it does not have to wait for a dedicated connection to start sending data, however it is not as reliable as a TCP connection between server and user. This therefore requires a few methods to ensure that the message reaches its destination without getting lost or altered in the transmission.

What is TCP:

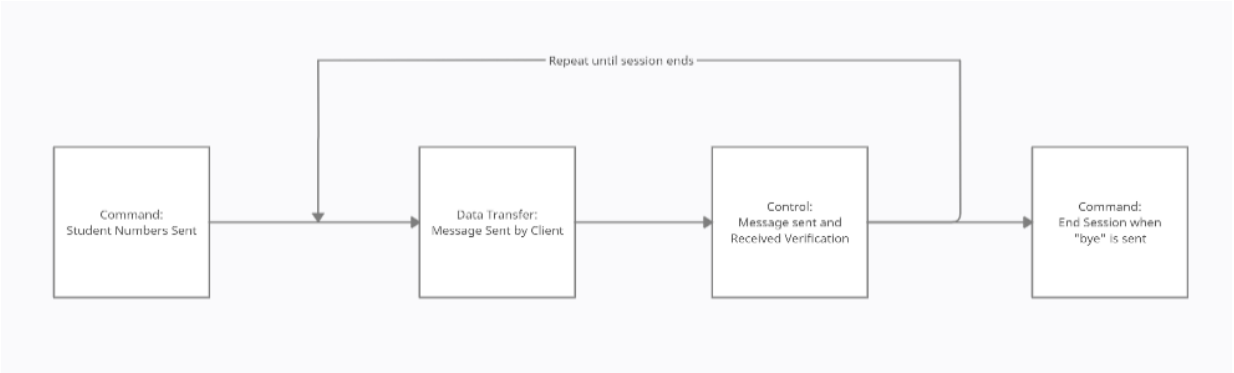
Transmission Control Protocol (TCP) is a connection-based protocol. This means that a connection is established before messages are sent. The connection is maintained while data is being transferred and then dissolved when the connection is no longer required. This protocol is slower to use than a UDP connection but is far more reliable. This means that methods do not need to be devised to ensure messages are sent, just using the built-in system ensures the messages are received.

Message Format and Structure

The messages format is that of text-based messaging (ASCII). This enables easier understanding, monitoring, and testing of the message. It also enables the adding of characters in front of the message easily. This text added serves two purposes, the first is an error check. If the information that is read in front of the message fails to be read properly, it shows that corruption has occurred. Secondly it provides the forwarding location of the message.

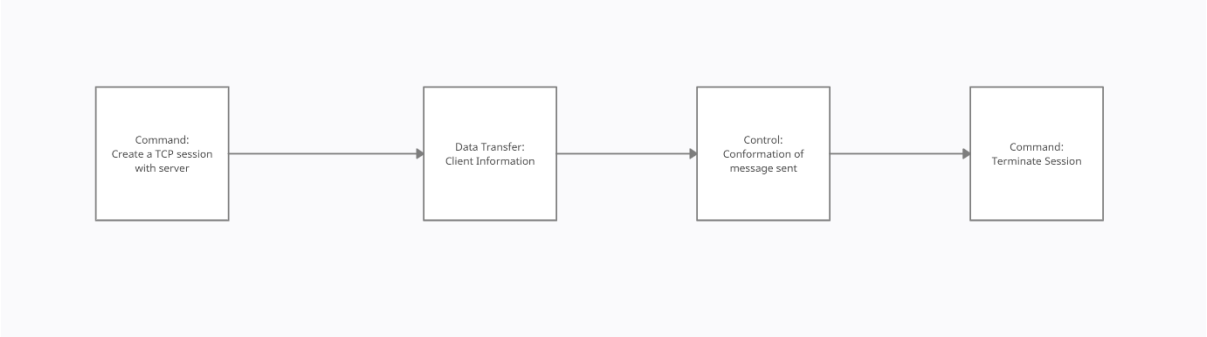
UDP Message types used:

- Command
    - When the student Numbers are sent it creates a new session
    - When “bye” is sent, it terminates the session.
- Control
    - The set confirmation messages used for sent verification.
- Data Transfer
    - Messages sent by the client to the other client.



TCP Message types used:

- Command
  - Creates a TCP session with the server.
  - Terminates the session.
- Control
  - Allocate port for client to communicate on.
- Data Transfer
  - Client Information



Object Orientated Design

Client_GUI:

This class creates a client interface that provides easier reading and writing for the users. This enables a far nicer experience for the user when sending and receiving messages from each other.

Client:

A client decides if they are connecting to a local or global server, in which case they connect to the server using TCP. This then sends their connection information to the server and terminates the TCP connection. Their student number is then required and the student they wish to connect to. This is then sent to the server using UDP. When a connection is identified as available, the client_GUI is called, and a frame is created for use. The messages by the user are typed and sent. Messages are also received using client Receiver Thread. When “bye” is sent by either user, the chat session terminates.

ClientInformation:

This contains the information about clients that connect to the server. This includes their student number, internet address and port number.

ClientReceiverThread:

This class receives a message from the server and then unpacks it, it can either be a terminating message, a confirmation message, a message indicating the server is waiting for the other individual to join or it could be a message from another user, which results in a confirmation message being sent. The message received is then displayed to the user.

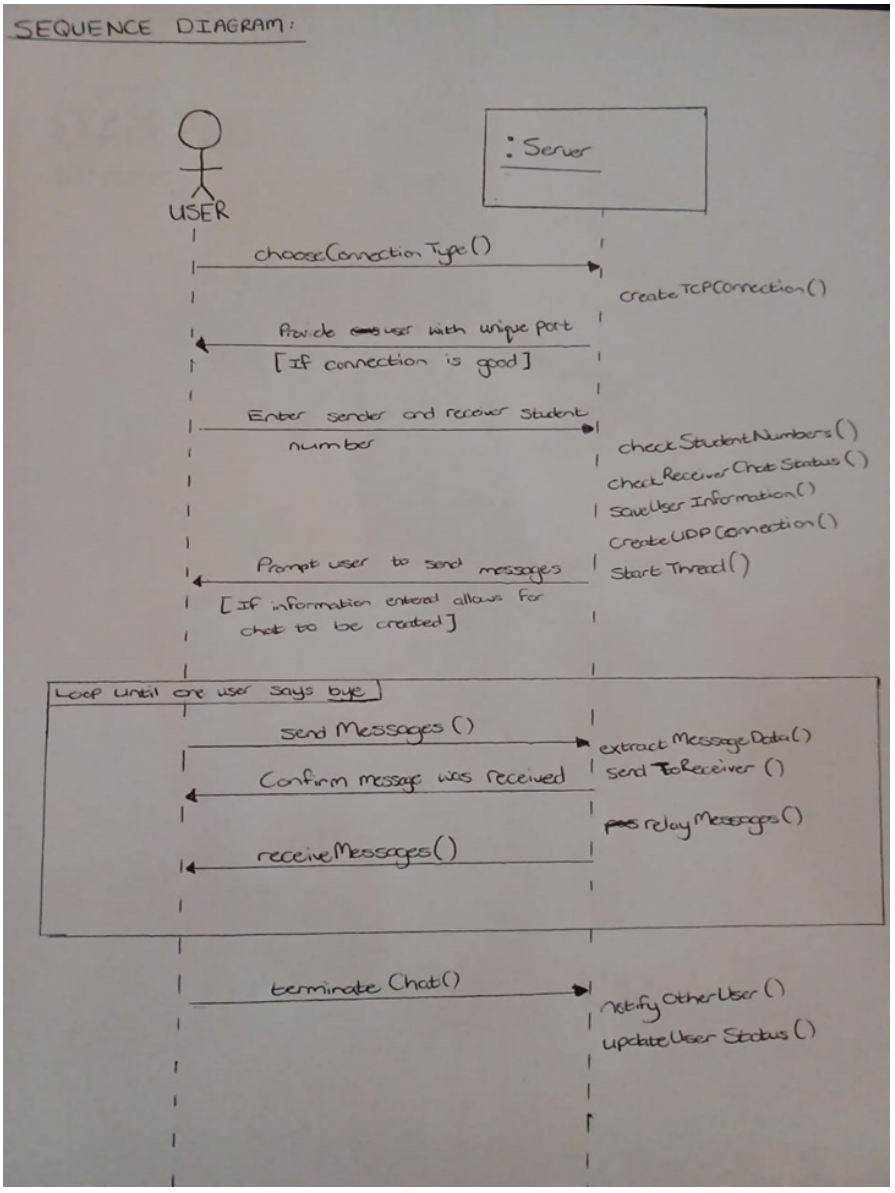
Server:

The server runs permanently and receives messages from clients. It stores the information of the clients in an array and calls the ServerThread whenever the server receives a message.

ServerThread:

This class receives a message and then forwards the message on to the correct client. It also checks that the specific clients are correct and that a chatroom can be created between two clients.

Sequence Diagram



Explained:

The User chooses whether they are connected via LAN to the server or by WAN to the dedicated server. The user connects and sends student number and student number to connect to. Server stores this information and terminates TCP connection and moves to UDP and tries to connect you to the other client. Once connected the user is prompted to send messages. Messages are sent from client to server to another client and back. When chat is completed, the chat is terminated and the server registers this and notifies the other client.

Features

Loss Prevention:

TCP connection is used to create the initial link between server and client to ensure that the correct information is transferred to the server to identify a specific client. This is essential as if this data does not reach the server, a client is not recorded and cannot be contacted, therefore cannot use the chat platform at all.

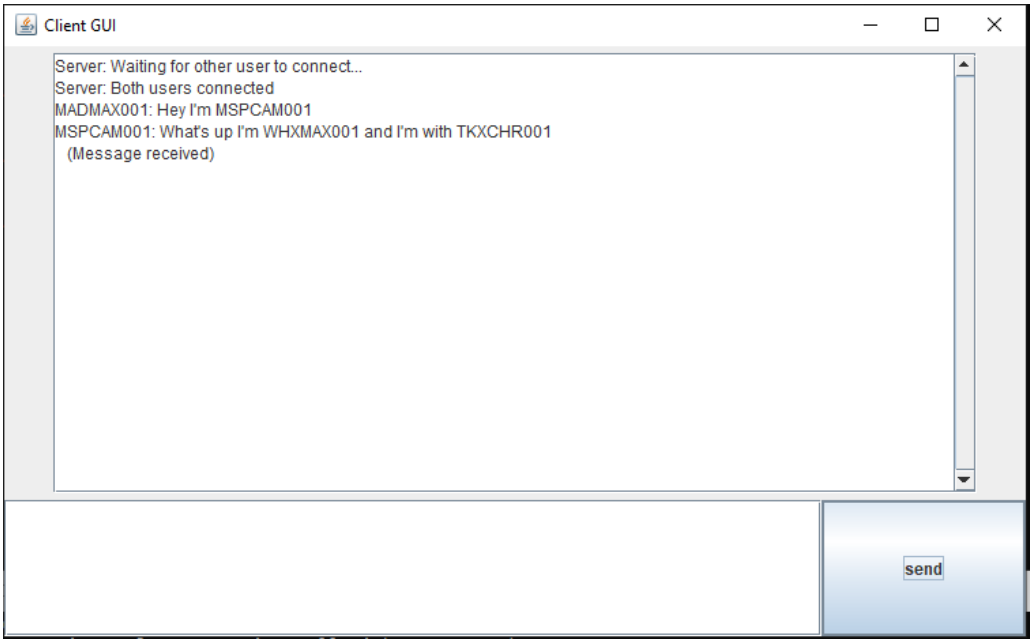
For the UDP connection there is a packet loss test that is applied by the clients. It sends a set message to indicate receipt of the message from that client, that is used as a conformation message that the message has been received by the client. This is all forwarded through the server to where the main message originated. It shows this with a message saying (message received). Also, if a message is corrupted, the message will not be confirmed as sent.

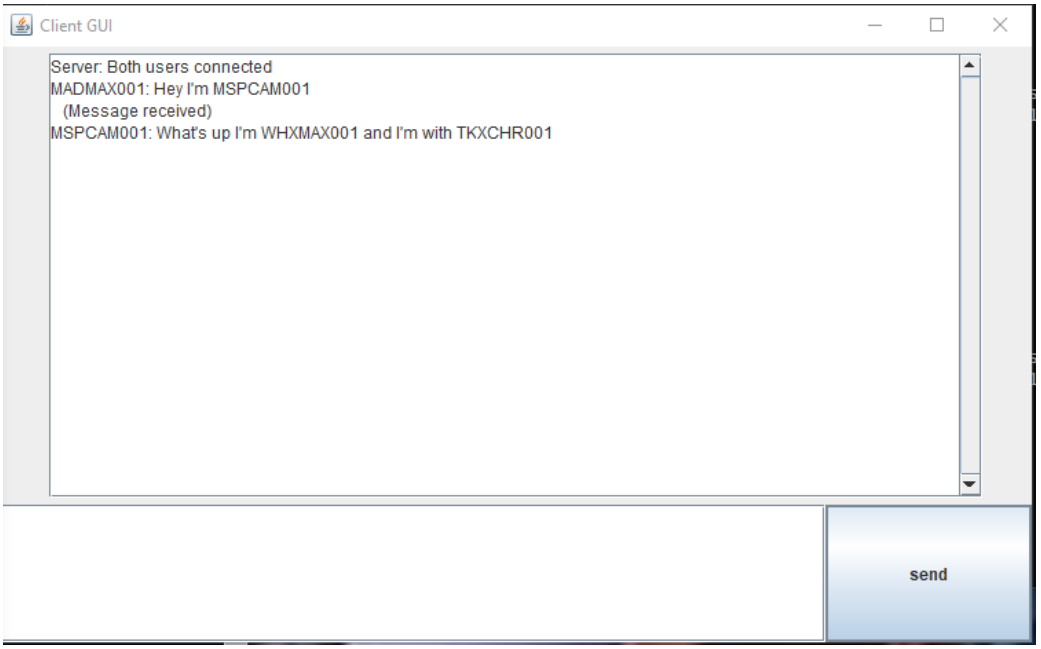
Threads:

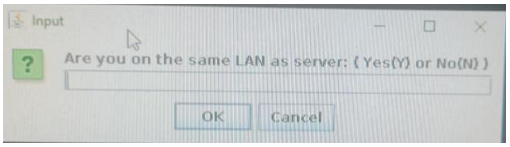
This allows the server to handle multiple chats at once instead of having to deal with messages one at a time. Threads also allow the client to receive and send messages at the same time.

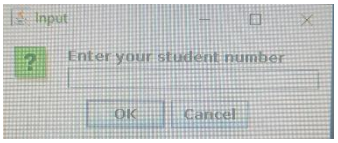
GUI:

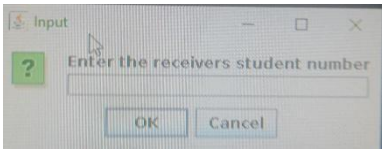
The GUI provides a method for the user to both receive and send messages at the same time. It also creates a better display which entices people to use our platform over a completely terminal based chat application, as a terminal is not required once the application is up and running.











Unable to send a message to the wrong individual:

Another user cannot connect to the same client while the client is messaging someone, this prevents sending a message to the wrong individual.

User Identification:

Users must provide student numbers upon entry which means you know who you are talking to within the application.