Concurrent Network Applications – Max Stockton 19017767

# Contents

Contents

[Contents 1](#_Toc70080988)

[Introduction 1](#_Toc70080989)

[Explanation of Strategy 1](#_Toc70080990)

[Critical Reflection 3](#_Toc70080991)

[User Guide 3](#_Toc70080992)

[Index 4](#_Toc70080993)

[Bibliography 4](#_Toc70080994)

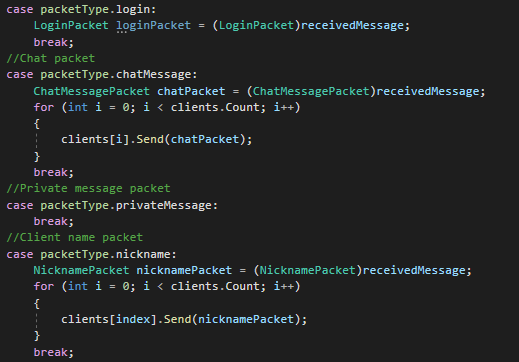
# Introduction

This report will cover the development of the chat system that was developed during this module. It will talk about how the program works on a technical level, as well as my honest feelings about how the module went as a whole as well as where I will expand my knowledge of Network based applications.

# Explanation of Strategy

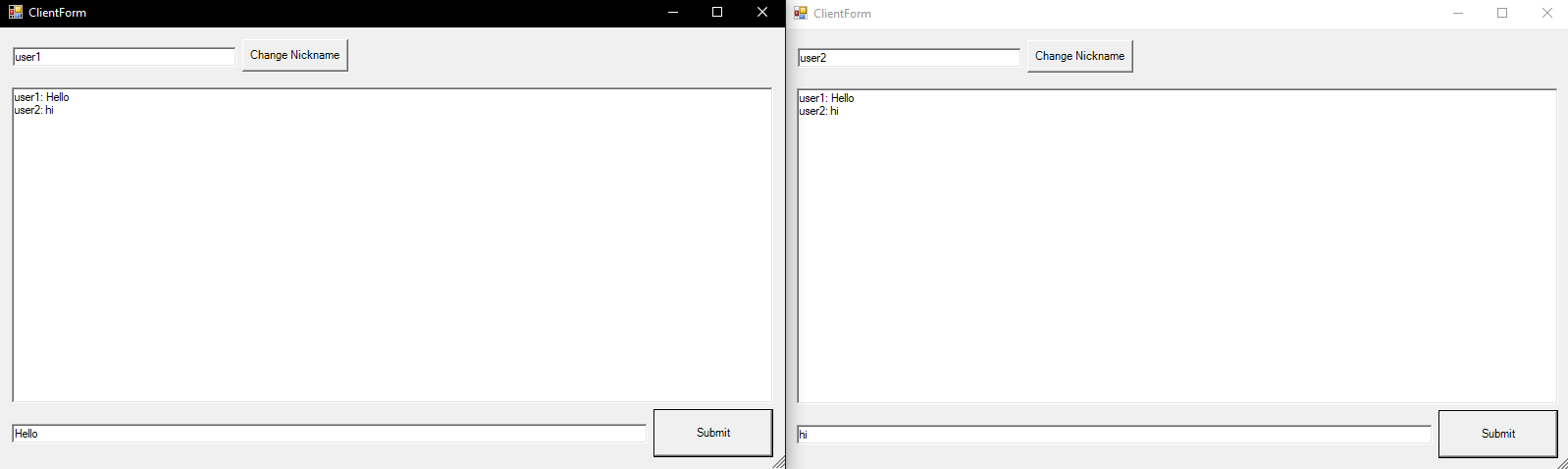
Using multi-threaded code meant that the code had to be kept organised and easy to read. This was done in the form of using relevant names for variables, functions and classes and consistently using industry standard naming conventions. More complicated parts of the code were commented on so that it is easier to read & track of. Transmission Control Protocol (TCP) was used to send off message data as the data from the client was being sent to the server via a serialized packets system. It is also guaranteed that the data will arrive at the destination server. User Datagram Protocol (UDP), however, is normally used for programs that can deal with losing some data like audio & video, which isn’t present in this project, since it sends data across before the server can receive the data. It’s faster but the data isn’t guaranteed to arrive in order or not at all. If we were to use UDP for messaging, the data would come through all jumbled up & not necessarily in the right order.

A thread has been dedicated to each client that is currently connected to the server, so packets sent to/from the server are updated on the client side efficiently. It kept track of how many clients were connected and started a thread every time a new client connected which would run the Client Method function, this function would receive the bytes sent from the client & determine the packet type & then send it back to each client connected. Threads were also used in the client project to process the responses received from the server by using the Process Server Response method. The method also reads the incoming bytes from the server and determines the packet type of the data received & updates it accordingly.



*Server determining packet type of received data & sending it back to clients*

The nickname system implemented into the project is handled on the client. The nickname is set by the user from the client interface & then put as a prefix to each message sent from that client & sent to the server as part of the message chat packet. This can be updated quickly by changing the name in the nickname box & using “change nickname”.



*Nicknames*

Any packets received by the server from a client are read & then sent out back to the original client & any others connected to the server. This is done by using a concurrent dictionary of clients which is stored on the server, it’s updated whenever a new client connects to the server. When the server sends a packet out to the clients, a for loop is used to send the packet back to each individual client.



*Defining the Concurrent Dictionary*

# Critical Reflection

This assignment has allowed me to learn the basics of C# which will be useful for working on a variety of projects whether it’s in the industry itself or my own personal projects in Unity or other network-based projects, as well as the fundamentals and basic principles of network-based application development. I will also be able to apply these principles & fundamentals to future projects in the industry or my own personal projects. I also learned the basics of using Windows Forms which has allowed me to create simple interactive GUIs for my programs.

I was able to set up a packet-based chat program that connected to a server project. The client gave the user the ability to send and receive messages from the server as well as send & receive data from other clients. I was able to send packets across and have the client & server recognise what kind of data was being transmitted & act differently accordingly. I was also able to use Windows Forms to create a simple interactive GUI that the user could use to send data across to the server.

However, I struggled with a lot of issues with the program. Due to working from home I was afraid to ask for help from tutors for a good portion of the semester & I only asked for help closer to the assignment deadline. This halted my progress by a good amount which meant I fell behind with my project quickly. I’ve come across many problems in my project including messages not being sent to the server and having issues with putting code in the right places. Though due to re-sits I’ve been able to add more onto the project, however this was harder due to focusing on other assignments & an overall negative mental state.

If I were to attempt this project again, I would have done more reading into C# in my own time, so I wasn’t struggling as much with the syntax. I also would have asked for help earlier when I encountered a problem but also taken more of an initiative to investigate problems myself properly before asking for help with my work. I would have also liked to have moved onto the security & encryption section of the project as this is a fundamental part of any server-based application & I would like to learn how to do it. I will practise doing this in my free time, so I don’t fall behind. I would also be more organised with my code so I can find certain sections of my code easier & apply the same systems to other parts of my project if it’s necessary. I also would’ve liked to take the time to create a graphics-based program that sent data to the server via UDP, so I have experience in developing graphical network-based projects & using different methods of sending data to and from the server. Whilst I used Windows Forms to create my GUI, I would also like to learn how to use WPF to create programs too, so I have experience in using different GUI libraries.

# User Guide

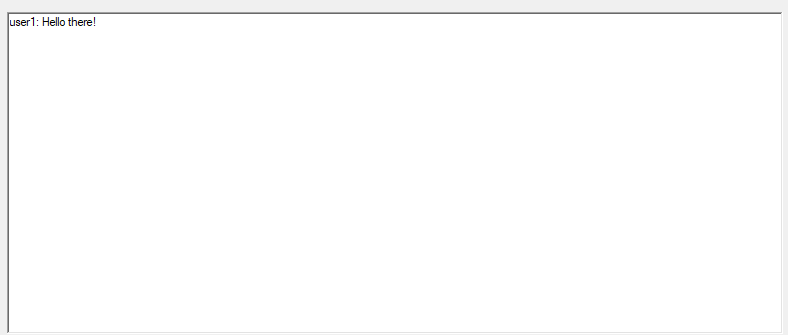
The user will be able to set themselves a nickname before they will be able to send messages to the server. This is done by typing a nickname in the box at the top & hitting “change nickname”.



The user will be able to then type messages into the box at the bottom & hit “submit”.



The message will then be sent to the server as a chat packet, with the username before it. When the data is received back from the server, if the data is a chat packet it will be displayed in the bigger text box. This is updated across any clients.



However if the user tries to send a message without setting a nickname, a message will be sent out.



# Index

Transmission Control Protocol (TCP) – Method of sending data that is normally used in messaging programs that ensures all data reaches server - Explanation of Strategy.

User Datagram Protocol (UDP) – Method of sending data that is normally used in games, audio & video streaming, not all data necessarily reaches server though faster than TCP - Explanation of Strategy.

Packet – How data is sent to the server & vice versa - Explanation of Strategy.

# Bibliography

Sciencedirect.com. 2021. *User Datagram Protocol - an overview | ScienceDirect Topics*. [online] Available at: <https://www.sciencedirect.com/topics/computer-science/user-datagram-protocol#:~:text=UDP%20is%20commonly%20used%20for,applications%2C%20such%20as%20DNS%20queries.> [Accessed 23 April 2021].