

NPTC

Socket Programming and Concurrency

Assignment 1

261481 (Aaron Mills)
3-31-2022

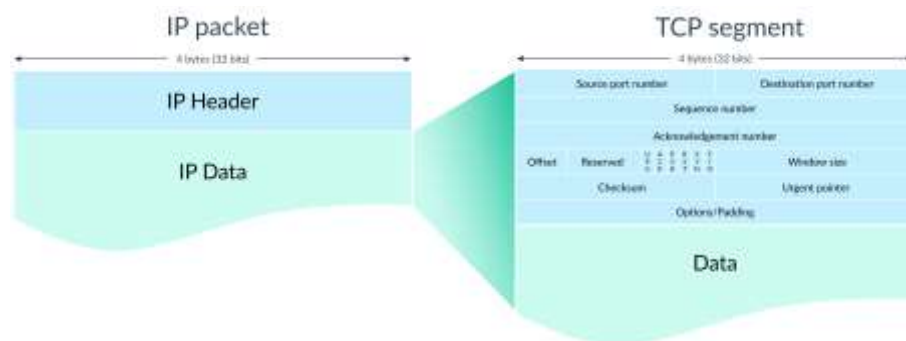
Contents

Transmission Control Protocol.....	2
What is a transmission control protocol.....	2
Importance of Transmission Control Protocol.....	2
Use cases for Transmission Control Protocol	2
Three-way handshake / Two Generals Problem.....	2
Three-Way Handshake.....	2
Two Generals Problem.....	3
Possible Solutions	4
Maximum Transmission Unit	5
Packets and Frames	5
Relevance of PDU.....	6
Synchronicity and Asynchronicity in Network Protocols	7
Examples of Synchronous and Asynchronous	7
Interoperability in Networked Applications	8
Defining Interoperability.....	8
Types of interoperability in interconnected systems	8
Methods of testing.....	8
Realistic scenario of the importance of interoperability	9
Benefits of interoperability	9
Report	10
Ideas and Features Considered.....	10
Implementation	10
Server	11
Client	14
References	17

Transmission Control Protocol

What is a transmission control protocol

Local networks and the world wide web are a vast interconnected mesh of connections which is run on physical and digital technologies one aptly being Transmission Control Protocol (TCP). TCP is an interoperable network protocol that aids in establishing and maintaining connections among heterogeneous networked systems. TCP works within the Network Layer of the Open Systems Interconnection (ISO) model, which is paired with Internet Protocol (IP) which again is a network protocol and is responsible for defining how packeted data is sent to and received by computers. Nodes over a network are assigned an address by the IP protocol to easily identify them for sending and receiving packets. [\[1\]](#)



[\[2\]](#)

Importance of Transmission Control Protocol

The model above displays the structure of a data packet which contains being sent over a network. This protocol is important because it is solely responsible for the communication of the internet. Even though there are checks and systems in place to stop errors occurring just with the structure of the TCP segment. Even if the data packet is correct, problems can occur when sending that packet over a network to another node.

Use cases for Transmission Control Protocol

One alternative to TCP is User Datagram Protocol (UDP) which is used for more high demand data packet connections that require reduced latency such as video streaming or voice over IP (VoIP) connections where consistency of data arriving is important to limit the latency and or delay. [\[3\]](#)

Three-way handshake / Two Generals Problem

Three-Way Handshake

Even though there is secure technology that maintains connections and handles transfers of data packets, this does not exist without problem, one being the "Two Generals' Problem". This problem occurs within three-way handshake. This three-way handshake or also known as SYN-SYN-ACK

(Synchronize, Synchronize, and Acknowledgement) is a method of connection using TCP/IP to connect a client and server. [4] The handshake is a three-step process of firstly maintaining the parameters of a socket before data that contains both Hypertext Transfer Protocol (HTTP) and Secure Shell (SSH).

How SYN-SYN-ACK is relevant is that firstly a connection between the client and server is established through ports that are accepting new connections. The client machine sends the synchronization sequence number, which is a counter that keeps track of outgoing bytes sent by the host machine through data packets. [5] The purpose of the data packet is to ask the receiving server if they are accepting new connections.

After the server receives the SYN data packet, a response is sent back which contains two sequence numbers, one being an ACK number that matches the number received but modified by one e.g. $(X+1)$. The second number being a SYN number which is another random number sent from the server to the client. This confirmation sent back indicates that the server successfully acknowledges the client's request.

The third step contains another acknowledgement where the client's machine needs to send back an ACK which contains the server's sequence number with an added value e.g. $(Y+1)$.

Once the server successfully receives the last sequence number a connection between both nodes is established. These steps are necessary to confirm each sequence number by both parties.

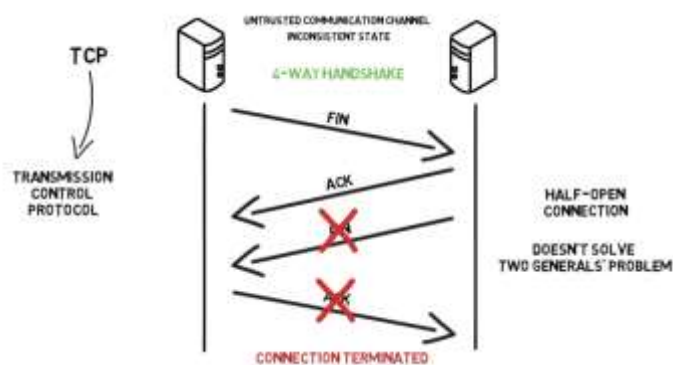
Two Generals Problem

The two general problem is a popular thought experiment. The thought experiment includes the characters of two generals which are representing two computer nodes. [6] The idea is that both generals want to coordinate an attack at the same time so a request and confirmation needs to be sent out from both parties. The first scenario is that General A sends a request to general B on a plan of action. After receiving the plan, General B sends a confirmation back to General A. At all times either party is in a position where they are unknowing whether their confirmation got received by the opposite party. Infinite confirmations cannot be sent and there is a lack of knowledge whether the last confirmation was received.

The second scenario is that General A sends a message out and has not heard any response back, this could be down to two things. The initial message got intercepted and was not received or General B did receive the message, but their reply got intercepted, and from that General B does not know whether General A received any message and so forth. This results in an inconsistent state where one General will be the only one attacking which is not what we want. No matter how many messages are sent, there is no guarantee that a consensus can be reached between both parties. This problem is considered by many to be unsolvable. [6]

When imagining this problem with two computer machines and TCP connections previously discussed. Scenario's such as the four-way handshake to terminate a connection between two machines [7] as a close example should be considered in why the problem is important to discuss.

The four-way handshake is a mechanism used by TCP to terminate connection as seen below. Firstly, a FIN message is sent, an acknowledgment is sent back. A further fin and acknowledgement are sent back which then confirms to both machines to terminate the connection. Applying the two generals' problem here implies that if one machine were to fail at one half of the exchange, one machine may terminate a connection but the other may stay half-open.



[6]

Possible Solutions

Although as mentioned there is no full solution to mitigate the problem of lost messages, there are methods in which more confidence can be gained of the connection between both machines.

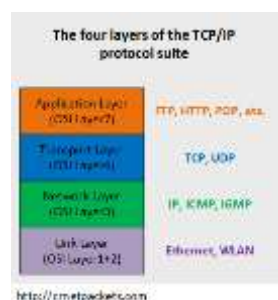
One approach is to send a large volume of messages that are marked with some identification and the receiving party can see how many messages get through on that first contact. By doing this, they are establishing a confidence with how reliable the connection is. Although this could be beneficial in ensuring a great likely hood that the connections will be received, this method could potentially be high in cost. [6]

A second approach is to estimate the time it takes to send a message between both parties. Messages and confirmations will be sent at the end of that estimated time period. After both a request and confirmation has been sent, any deviation in that timescale of messages suggests that the message has been intercepted and both parties can safely assume that no messages will be following and thus agree to the connection.

Maximum Transmission Unit

Packets and Frames

TCP uses limitations within packet sizes that are sent over a network. The packet sizes are determined by Maximum transmission Unit (MTU) which specifies that the largest a packet can be, which is measured in bytes. The largest size of a packet is not only determined by the payload but everything surrounding it which includes the headers. The maximum size of a MTU packet is currently 1500 bytes. [8] This packet is transmitted over the network layer of the OSI model displayed below.



[9]

Other considerations are frames which are units of data that follow the Link layer of the OSI model. [9] Traditionally this is involved in Wi-Fi and Bluetooth. The data link layers header from the OSI model and the payload merge together to form a frame which is different from a packet, think of it as an addition. Apart from having an IP address it seeks to travel to, it instead includes a Medium Access Control (MAC) address. A MAC address ties into a Network Interface Controller (NIC) [10] which is a component that relates to the physical location where a connection is made whether through an ethernet cable or Wi-Fi which was previously touched upon.

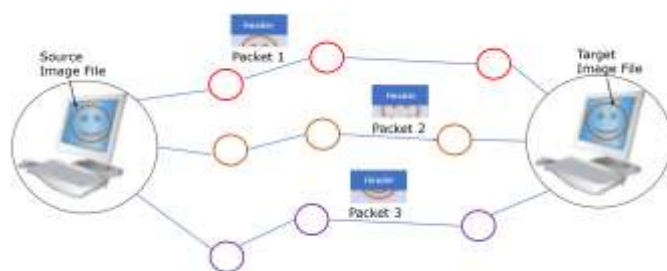
The diagram above depicts the structural differences in packets and frames.



[11]

Since the frame travels from the physical layer to the data link layer of the OSI model, more checks and flags are done to ensure that error correction within the frame occurs. This error correction or

prevention is often referred to as Frame Check Sequence which pinpoints the beginning and end of a frame. [12] Data can come in various sizes and due to the limitations in size of packets or frames, large files being transferred over a network would need to be divided into smaller packets that contain parts of the larger file. Due to what was discussed before with the file size limitations, destination IP's, ports, and MAC addresses. All of this is used to transfer data to a destination and reassemble the file as it was when it was before it was sent. This is briefly shown below in the diagram.



[8]

Frames beyond the size of 9000 bytes are often categorized as Jumbo Frames. [13] Term definitions of frame sizes could change name depending on the vendor according to Cisco. [14] Further expansions of frames are referred to as Baby Giants, which Cisco claims that “The baby giants feature allows a switch to pass or forward packets that are slightly larger than the IEEE Ethernet MTU. Otherwise, the switch declares big frames as oversize and discards them.” [14]

Relevance of PDU

Referring to the OSI model diagram. Each layer of the OSI model consists of different units which are called Protocol Data Units (PDU). [15] Transmission between different layers would be achieved by incorporating the PDU of that layer, whether it is the networking layer or some other layer. According to website Hamy [15], they refer to PDU as packets that travel over a link on the network layer”. The maximum packet size over both IPv4 and IPv6 are 64KB and 4GB respectively. [15] Just because these technologies can transfer packets up to this size, hardware limitations to come into play as a factor as well which can bottle neck transfers. [15] If a packet exceeds traditional sizes, IP fragmentation takes place where the packet is fragmented into smaller ones before being sent and after the MTU path has been determined by the network router. The example below briefly describes the process of fragmenting packets.

- Example 2
1. The application on your hosts tries sending an IPv4 packet bigger than 1500 bytes.
 2. Since its bigger than the default OS MTU, The OS splits it into smaller packets, each with their own special headers so they could be reconstructed later on.
 3. fragmented packets will be sent and assuming no router allowing a defragmented packet with that size, they never get reconstructed along the way.
 4. The other end receives the packet, The OS now defragments them and sends the packet to the receiving application.

[15]

Synchronicity and Asynchronicity in Network Protocols

To understand how these terms apply within the scope of networking protocols it is important to understand the definitions for both terms. Synchronicity protocols as defined by tutorialspoint [\[16\]](#) website is "Synchronous protocols take the whole bitstream and drop it into the character of equal size. It is a data transfer approach in which it is a continual flow of information signals followed by computing signals". Alternatively asynchronous protocols process could be described as transmitting data one byte at a time which operate at irregular time intervals. IBM suggests that "A start bit (one character) precedes the byte (eight characters) of information and a stop bit (one character) follow the byte." [\[17\]](#)

Examples of Synchronous and Asynchronous

Applications that could coincide with synchronous networking protocol is something that runs in real time such as online Voice Over IP (VOIP) or video conferencing chats. [\[18\]](#) These applications require the need for instant input and output (I/O) to be transmitted to and from a network. The instant I/O is detrimental to it being effective. Other examples of applications could be video games which offer a wide variety of real time and less instant feedback reliant systems within the application. Video games that strongly rely on instant synchronous networking, rely on it down to a timeframe of milliseconds. Some real examples are First Person Shoot video games such as Valorant or Counter Strike: Global Offensive where minimal timings between user input, server updates and screen outputs is detrimental to the game. Many modern games like the ones mentioned rely on servers that run at a higher tick rate to benefit the players playing the game by providing updates more frequently. Tick Rate is essentially the amount of time a server updates per second. For example, the video game Valorant uses servers that run at 128-tick [\[19\]](#) while Counter Strike: Global Offensive official matchmaking servers run at 64-tick. [\[19\]](#) These servers will be updates dozens of players positions and actions a second which is enhanced by high rates.

Other than real time communications, asynchronous communication could be categorized as exchanges of data that does not require immediate attention and can be approached at any time frame. Some examples, of asynchronous application could be email, some aspects of social media, and forums. While the information gets posted there, the need for instantaneous and continuous connection and updating is not required. Perhaps one could argue that email would need to be updated at regular intervals to see if any new mail has been received, the need for that update could be determined by the user by simply refreshing the [page](#).

Interoperability in Networked Applications

Defining Interoperability

To better understand how interoperability can be applied to different networking concepts, understanding the meaning of the phrase is important. According to the Oxford Learner's Dictionaries [\[20\]](#) interoperability is defined as "the ability of computer systems or programs to exchange information". Outside the scope of definitions, interoperability is not only computer systems that connect and communicate with one other over a network but a multitude of different other devices that communicate through a network. This communication is done in a coordinated way without much or sometimes any input from a user.

Functions relating to interoperability include data access, transmission of data over a network, and depending on the organisation willingness, cross organization collaboration may take place. [\[21\]](#) The website TechTarget claims that "Similar to compatibility, interoperability helps organizations achieve higher efficiency and a more holistic view of information". [\[21\]](#)

Types of interoperability in interconnected systems:

- **Semantic interoperability** – This is where two or more computer systems across a network connect to each other to share all sorts of data. This connection and sharing are done where both systems understand it in a meaningful way. [\[22\]](#)
- **Structural interoperability** – This type handles the data exchange between different systems and defines the formatting standard of packages being transported. The formatting is defined by structural interoperability through the syntax of network exchanges. Healthcare Information and Management Systems Society (HIMSS) notes on the topic of interoperability that [\[23\]](#) "this form of interoperability, which is also known as syntactic or technical interoperability, permits the "uniform movement" of health data from system to system where the "clinical or operational purposes and meaning of the data is preserved and unaltered.""
- **Cross-Organization Interoperability** – This type of interoperability focuses on aspects of organization that fall under non-technical aspects. In addition, it refers to standardizations of systems relating requirements, practices and policies of systems. [\[21\]](#)

Depending on the scale of systems that adhere to interoperability, it can often be important for different systems and technologies to be as compatible as possible to ensure the smooth transition of data over a network. Especially during a time where Internet of Things (IoT) are becoming more common place among many businesses. The syntax used between systems would preferably be as close as possible to minimise friction of compatibility between systems.

Methods of testing

Some methods of improving interoperability are compatibility testing. Testing can occur by setting up a testing environment like the current network or system. Using automated tools to test activities between different systems. Automation tools are used to minimize test cases. Records will be saved where successes and failures occurred, and test plans can be initiated to analyse and fix the root cause of failure. [\[24\]](#) To minimise frictions or failures, as mentioned before using similar technologies and syntax among most systems will hopefully reduce or mitigate any failures. As different syntax's attempting to translate instructions could cause issues as each syntax is different and could be interpreted differently.

One crucial example of systems that must adhere to being interoperable is the health industry where the cross communication between organizations and their systems is often needed. Multiple sources of stored information may need to be accessible by not only one organization but multiple such as clinics, hospitals, pharmacies, and other institutions that require medical documentation. The systems between these organization should be closely linked in similar technologies, software choices and syntax between those choices.

Realistic scenario of the importance of interoperability

The process of a patient booking an appointment with a GP requires the use of a system to receive that call, a method of data entry into a machine which then stores that data somewhere and must relay the message of an appointment to the GP which may require different systems to communicate with one another. After the patient has completed their appointment, they may need prescribed medication where the GP would need to check the patient's information and the information of the medication and then communicate with a pharmacy system to issue that prescription as a certain location and time. These steps involve data exchange and if the systems are not interoperable, this would not be possible electronically and would need to be done manually using paper-based recording.

Benefits of interoperability

Interoperability aims to seek out the most streamlined process which minimises the need for higher cost and more steps in between. Information access and the quality of that information could be in an improved state as it will be adapted to multiple system formats. Lastly is if a system is interoperable, it could be considered highly efficiency compared to systems that do not follow the same design philosophies.

Report

Ideas and Features Considered

The application that was built is a multiuser chat application using the C# programming language which takes advantage of Microsoft Forms App.

Research was conducted as to what could constitute a chat application and the features that come with it. Key points that were derived from research suggested that a chat application can support multiple chat at one time. Easy connection through a button click to chat to a user. Although most refined applications such as Discord offer a system for adding friendly simply by name, for the purpose of this application and to provide a proof of concept in regard to chatting to another user. The IP and Port of the server would need to be entered manually to accept and maintain a connection between two clients. As for multiple chat support, this feature was planned but would be contained within the same textbox, rather than using different windows for different connections. A user would need to click the IP from a list they want to send a message to.

The sending and receiving messages through a client and server could be considered the bare minimum for an application to be considered a chat application. Further features added to this project were considered such as quality of life features where it could potentially enhance the user experience. Features that were considered is the ability to change the text size of each window to a larger or smaller font. This was considered to help users see the content of the application more easily, whether they suffer from poor eyesight or would just like larger text. Other features that are prominent on popular applications such as Discord and Twitter is the use of Light and Dark mode. This feature alters the colour of the chat window and content to appear in a darker colour which can aid in visibility of content. Lastly the use of Emoji's is popular among application that contain a chat feature and was planned in iterating a selection of emoji to choose from to incorporate into messages.

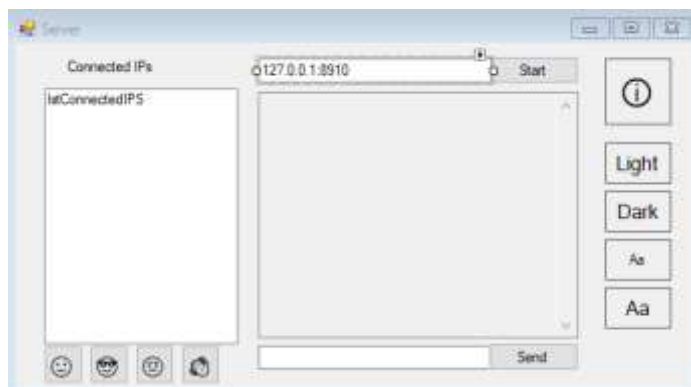
Implementation

To establish back-end connections to and from clients, a TCP wrapper was used. SuperSimpleTcp is a package that allows an application to enable, maintain a connection, send data, receive data and disconnect. This simple wrapper contains Secure Socket Layer (SSL) support. This package will server to be the logic behind the chat application. Firstly, two C# forms applications with Dot Net Framework 4.8 support. These two forms will act as a server and client.

Each form needs the wrapper previously mentioned which supports the application in opening, maintaining and closing connections. The SuperSimpleTcp package is applied through the NuGet Package Manager. Once rebuilding each form, development can begin to link the client and server.

Server

The server form was designed below to include previously mentioned features.



A textbox to input an IP and PORT is included to grab the address and host the server, this hosting starts by pressing the start button. Before the server can be started, declarations need to be made. Within the function of the button the server starts.

On form load, the content of the IP textbox is held and once the server start button is pressed it calls the function to start the server.

```
1 reference
private void Form1_Load(object sender, EventArgs e)
{
    //restricting access on form load
    btnMessageSend.Enabled = false;
    // Obtaining the IP and PORT from the textbox for the server
    server = new SimpleTcpServer(txtIPAddress.Text);
    // declaring events for connection, received and disconnecting which will be used with the client also
    server.Events.ClientConnected += Events_ClientConnected;
    server.Events.ClientDisconnected += Events_ClientDisconnected;
    server.Events.DataReceived += Events_DataReceived;
}
```

Events are declared which are supported by the Tcp package which handle, connection, data and disconnection which the client uses also.

```
// declaring the server which is support by the imported package
SimpleTcpServer server;

1 reference
private void btnStart_Click(object sender, EventArgs e)
{
    //starting the server which is support by the imported package
    server.Start();
    //keeping the connection open so that further connections can be made
    server.KeepAlive.EnableTcpKeepAlives = false;
    // dialog messages to notify the user of the server status
    txtDialog.Text += $"Starting the server...{Environment.NewLine}";
    txtDialog.Text += $"Server has started!{Environment.NewLine}";
    //disabling the start button to stop any breaking of the server
    btnStart.Enabled = false;
    //allowing the user access to the send button
    btnMessageSend.Enabled = true;
}
```

The invoke method is used to update the GUI after a change has been made. Below is the update of the dialog box to contain the server IP.

```
private void Events_DataReceived(object sender, DataReceivedEventArgs e)
{
    //Invoke used to update the application so it doesnt break
    //obtains the IP and includes it into the Dialog box
    this.Invoke((MethodInvoker)delegate
    {
        txtDialog.Text += $"{e.IPAddress}: {Encoding.UTF8.GetString(e.Data)}{Environment.NewLine}";
    });
}
```

This produces the disconnection dialog which includes the IP address. After disconnection the list box is updated to remove the recently disconnected IP address.

```
private void Events_ClientDisconnected(object sender, ConnectionEventArgs e)
{
    //Invoke used to update the application so it doesnt break
    this.Invoke((MethodInvoker)delegate
    {
        // displays which IP has disconnected from the chat
        txtDialog.Text += $"{e.IPAddress} disconnected.{Environment.NewLine}";
        // removes the IP from the list who has disconnected
        lstConnectedIPS.Items.Remove(e.IPAddress);
    });
}
```

This produces the connection dialog which includes the IP address. After successful connection, the message is displayed, and the list box content is updated to include the IP address.

```
private void Events_ClientConnected(object sender, ConnectionEventArgs e)
{
    //Invoke used to update the application so it doesnt break
    this.Invoke((MethodInvoker)delegate
    {
        // displays which IP has connected
        txtDialog.Text += $"{e.IPAddress} connected.{Environment.NewLine}";
        // adds the IP to the list
        lstConnectedIPS.Items.Add(e.IPAddress);
    });
}
```

The quality-of-life features previously mentioned were the inclusion of giving the user the ability to change the text size of the application content, whether that is making it larger or smaller.

```
private void btnBig_Click(object sender, EventArgs e)
{
    //declaring variables for specific font sizes for the large button
    int fontBig = 18;
    int fontMainB = 17;
    // changing the content to match the font size

    label3.Font = new Font(label3.Font.Name, fontBig, label3.Font.Style);
    txtDialog.Font = new Font(txtDialog.Font.Name, fontMainB, txtDialog.Font.Style);
    txtIPAddress.Font = new Font(txtIPAddress.Font.Name, fontBig, txtIPAddress.Font.Style);
    txtMessage.Font = new Font(txtMessage.Font.Name, fontBig, txtMessage.Font.Style);
    lstConnectedIPS.Font = new Font(lstConnectedIPS.Font.Name, fontBig, lstConnectedIPS.Font.Style);
}
```

Unique values for each instance of the change could have been used but for the sake of easy changes later on, declarations were used to easy change the content all at once and with one value.

The occurs within the smaller text change but changes the text to its original size, rather than smaller than default.

```
private void btnSmall_Click(object sender, EventArgs e)
{
    //declaring variables for specific font sizes for the large button
    int fontSmall = 8;
    int fontMainS = 15;

    // changing the content to match the font size

    label3.Font = new Font(label3.Font.Name, fontSmall, label3.Font.Style);
    txtDialog.Font = new Font(txtDialog.Font.Name, fontMainS, txtDialog.Font.Style);
    txtIPAddress.Font = new Font(txtIPAddress.Font.Name, fontSmall, txtIPAddress.Font.Style);
    txtMessage.Font = new Font(txtMessage.Font.Name, fontSmall, txtMessage.Font.Style);
    lstConnectedIPS.Font = new Font(lstConnectedIPS.Font.Name, fontSmall, lstConnectedIPS.Font.Style);
}
```

A dialog window was added to provide the user with information on how to use the application. This simply outputs a message box with general information.

```
private void btnInfoDialog_Click_1(object sender, EventArgs e)
{
    this.btnInfoDialog.FlatAppearance.BorderSize = 0;
    //displays instructions to the user in a message box upon button click
    MessageBox.Show("1. Start the server.\n" + "2. Open a client and connect \n" + "3. Select the IP from ");
}
```

To change the colour of the application to either light or dark mode, declarations were made with a value which was used to match an RGB value. This declaration was done to make development easier and maintain consistent colours throughout.

```
//declaring values for different RGB colour values
private int c = 32;
private int cc = 43;
private int w = 255;

private int LBack = 240;
private int LBox = 255;
private int LText = 0;
```

Here is an example of a portion of the changes being made.

```
//changing the colours of the application to match a dark mode theme
this.BackColor = Color.FromArgb(c, c, c);
txtDialog.BackColor = Color.FromArgb(cc, cc, cc);
txtDialog.ForeColor = Color.FromArgb(w, w, w, w);

txtMessage.BackColor = Color.FromArgb(cc, cc, cc);
txtMessage.ForeColor = Color.FromArgb(w, w, w, w);
```

The nested IF statement below resides in the send message button which is linked to the chat box. This uses the listening function of the Tcp package and if a string is contained within the textbox and selecting an Ip from the list, a message will be sent to the destination IP address, which in this case is the connected client.

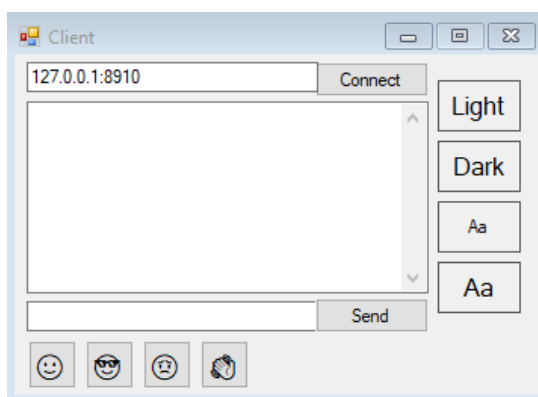
```
private void btnMessageSend_Click(object sender, EventArgs e)
{
    //uses the public boolean, using an if statement to listen after message button click
    if (server.IsListening)
    {
        //using a check to see if the message box is empty and send a message to the selected ip from the list
        if (!string.IsNullOrEmpty(txtMessage.Text) && lstConnectedIPS.SelectedItem != null)
        {
            server.Send(lstConnectedIPS.SelectedItem.ToString(), txtMessage.Text);
            txtDialog.Text += $"Server: {txtMessage.Text}{Environment.NewLine}";
            txtMessage.Text = string.Empty;
        }
    }
}
```

To include Emoji in the application, because this is not a rich textbox, ascii characters were used to implement the feature of sending emoji's which simply inputs the character as an addition to what is currently in the textbox.

```
1 reference
private void btnSmile_Click(object sender, EventArgs e)
{
    //inputs the emoji into the textbox string message
    this.txtMessage.Text += "😄";
}
```

Client

The design of the client form.



The client is almost identical to the server code with some minor changes. Shown below is declaring the client and a try attempt to connect using a button click. If the connection fails a catch exception is made to give error feedback.

```
SimpleTcpClient client;

1 reference
private void btnConnect_Click(object sender, EventArgs e)
{
    try
    {
        //connecting the client which is support by the imported package
        client.Connect();
        //disabling and enabling buttons
        btnMessageSend.Enabled = true;
        btnConnect.Enabled = false;
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Form loading aspects and events like the server.

```
private void Form1_Load(object sender, EventArgs e)
{
    // Obtaining the IP and PORT from the textbox for the client
    client = new SimpleTcpClient(txtIPAddress.Text);
    // declaring events for connection, received and disconnecting which will be used with the client also
    client.Events.Connected += Events_Connected;
    client.Events.DataReceived += Events_DataReceived;
    client.Events.Disconnected += Events_Disconnected;
    btnMessageSend.Enabled = false;
}
```

Disconnection message which references the IP address.

```
private void Events_Disconnected(object sender, ConnectionEventArgs e)
{
    //Invoke used to update the application so it doesn't break
    this.Invoke((MethodInvoker)delegate {
        //dialog for disconnected
        txtDialog.Text += $"Server disconnected.{Environment.NewLine}";
    });
}
```

The server message template which shows before message content.

```
private void Events_DataReceived(object sender, DataReceivedEventArgs e)
{
    //Invoke used to update the application so it doesn't break
    this.Invoke((MethodInvoker)delegate {
        // dialog
        txtDialog.Text += $"Server: {Encoding.UTF8.GetString(e.Data)}{Environment.NewLine}";
    });
}
```

Server connected dialog.

```
1 reference
private void Events_Connected(object sender, ConnectionEventArgs e)
{
    //Invoke used to update the application so it doesnt break
    this.Invoke((MethodInvoker)delegate
    {
        //connected dialog
        txtDialog.Text += $"Server connected.{Environment.NewLine}";
    });
}
```

Message sending function like the server.

```
private void btnMessageSend_Click(object sender, EventArgs e)
{
    if (client.IsConnected)
    {
        //using a check to see if the message box is empty and send a message
        if (!string.IsNullOrEmpty(txtMessage.Text))
        {
            client.Keepalive.EnableTcpKeepAlives = false;
            client.Send(txtMessage.Text);
            txtDialog.Text += $"You: {txtMessage.Text}{Environment.NewLine}";
            txtMessage.Text = string.Empty;
        }
    }
}
```

The remaining code is near identical which includes the quality-of-life features but adapted for a different form.

References

1. RFC 791 - Internet Protocol [Internet]. Datatracker.ietf.org. 2022 [cited 20 May 2022]. Available from: <https://datatracker.ietf.org/doc/html/rfc791>
2. Transmission Control Protocol (TCP) (article) | Khan Academy [Internet]. Khan Academy. 2022 [cited 20 May 2022]. Available from: <https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp>
3. What is Transmission Control Protocol (TCP)? Definition from SearchNetworking [Internet]. SearchNetworking. 2022 [cited 20 May 2022]. Available from: [https://www.techtarget.com/searchnetworking/definition/TCP#:~:text=Transmission%20Control%20Protocol%20\(TCP\)%20is,of%20data%20to%20each%20other.](https://www.techtarget.com/searchnetworking/definition/TCP#:~:text=Transmission%20Control%20Protocol%20(TCP)%20is,of%20data%20to%20each%20other.)
4. What is a Three-Way Handshake? - Definition from Techopedia [Internet]. Techopedia.com. 2022 [cited 20 May 2022]. Available from: <https://www.techopedia.com/definition/10339/three-way-handshake>
5. IBM Docs [Internet]. Ibm.com. 2022 [cited 20 May 2022]. Available from: <https://www.ibm.com/docs/en/zos-basic-skills?topic=layer-transmission-control-protocol-tcp>
6. Two Generals' Problem – Finematics [Internet]. Finematics.com. 2022 [cited 20 May 2022]. Available from: <https://finematics.com/two-generals-problem/>
7. Taking advantage of the 4-way handshake – UHWO Cyber Security [Internet]. Westoahu.hawaii.edu. 2022 [cited 20 May 2022]. Available from: <https://westoahu.hawaii.edu/cyber/forensics-weekly-executive-summaries/taking-advantage-of-the-4-way-handshake/>
8. [Internet]. 2022 [cited 20 May 2022]. Available from: <https://www.baeldung.com/cs/tcp-max-packet-size>
9. Reusch C. The relationship between the Maximum Transmission Unit (MTU) and the Maximum Segment Size (MSS) [Internet]. CRnetPACKETS. 2022 [cited 20 May 2022]. Available from: <https://crnetpackets.com/2016/01/27/the-relation-between-maximum-transmission-unit-mtu-and-the-maximum-segment-size-mss/>
10. What's a MAC Address and how do I find it? : Technology Services [Internet]. Slts.osu.edu. 2022 [cited 20 May 2022]. Available from: <https://slts.osu.edu/articles/whats-a-mac-address-and-how-do-i-find-it/>
11. [Internet]. 2022 [cited 20 May 2022]. Available from: <https://www.baeldung.com/cs/networking-packet-fragment-frame-datagram-segment>
12. [Internet]. 2022 [cited 20 May 2022]. Available from: <https://www.baeldung.com/cs/osi-packets-vs-frames>
13. [Internet]. Ce.sc.edu. 2022 [cited 20 May 2022]. Available from: [http://ce.sc.edu/cyberinfra/workshops/Material/NTP/Lab 13.pdf](http://ce.sc.edu/cyberinfra/workshops/Material/NTP/Lab%2013.pdf)

14. Support P, Switches C, TechNotes C. Configure Jumbo/Giant Frame Support on Catalyst Switches [Internet]. Cisco. 2022 [cited 20 May 2022]. Available from: <https://www.cisco.com/c/en/us/support/docs/switches/catalyst-6000-series-switches/24048-148.html>

15. Understanding Network IP Fragmentation | Hamy - The IT Guy [Internet]. Hamy - The IT Guy. 2022 [cited 20 May 2022]. Available from: <https://hamy.io/post/000b/understanding-network-ip-fragmentation/>

16. What are Synchronous Protocols? [Internet]. Tutorialspoint.com. 2022 [cited 20 May 2022]. Available from: <https://www.tutorialspoint.com/what-are-synchronous-protocols>

17. Asynchronous Protocol [Internet]. Ibm.com. 2022 [cited 20 May 2022]. Available from: <https://www.ibm.com/support/pages/asynchronous-protocol>

18. Synchronous vs Asynchronous eLearning Tools and Technologies [Internet]. TalentLMS. 2022 [cited 20 May 2022]. Available from: <https://www.talentlms.com/elearning/synchronous-vs-asynchronous-elearning>

19. Explaining tick rates in FPS games: Difference between 64 and 128 tick [Internet]. WIN.gg. 2022 [cited 20 May 2022]. Available from: <https://win.gg/news/explaining-tick-rates-in-fps-games-difference-between-64-and-128-tick/>

20. interoperability noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com [Internet]. Oxfordlearnersdictionaries.com. 2022 [cited 20 May 2022]. Available from: <https://www.oxfordlearnersdictionaries.com/definition/english/interoperability>

21. What is interoperability? - Definition from WhatIs.com [Internet]. SearchAppArchitecture. 2022 [cited 20 May 2022]. Available from: <https://www.techtarget.com/searchapparchitecture/definition/interoperability>

22. Hare V. What Is Interoperability and Why Is It Important? [Internet]. Tokenex.com. 2022 [cited 20 May 2022]. Available from: <https://www.tokenex.com/blog/what-is-interoperability-and-why-is-it-important>

23. [Internet]. 2022 [cited 20 May 2022]. Available from: <https://www.wolterskluwer.com/en/expert-insights/understand-the-three-levels-of-interoperability>

24. What is Interoperability Testing in Software Testing? (with Examples) [Internet]. Guru99. 2022 [cited 20 May 2022]. Available from: <https://www.guru99.com/interoperability-testing.html>