SIT202: Computer Networks and Communication
Leaning Evidence for Active Class Task 5

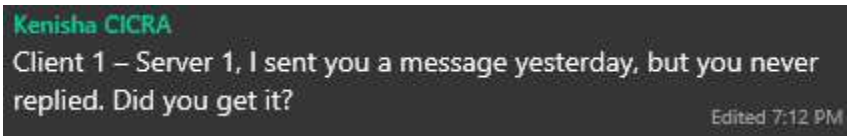
Name: Kenisha
Student ID: C23020001

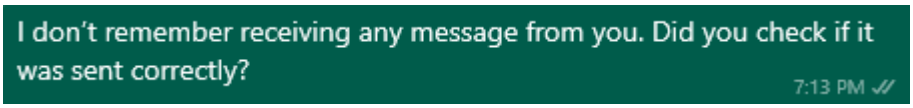
Members in this group activity task:

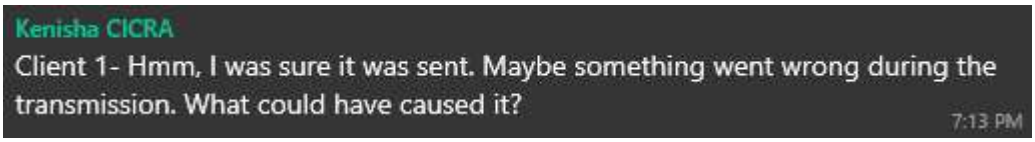
Nishad – C23110001
Kenisha – C23020001
Shekaina – C23110002
Raaid – C23020004
Pavithran – C22060015

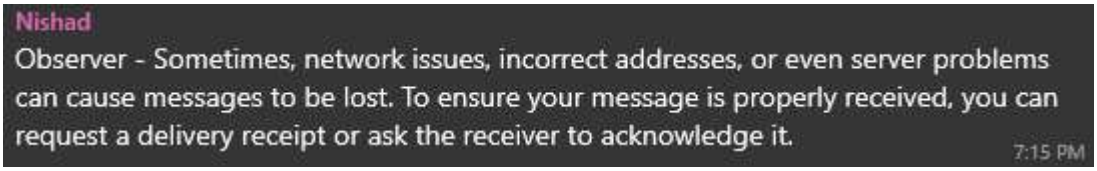
Activity 1

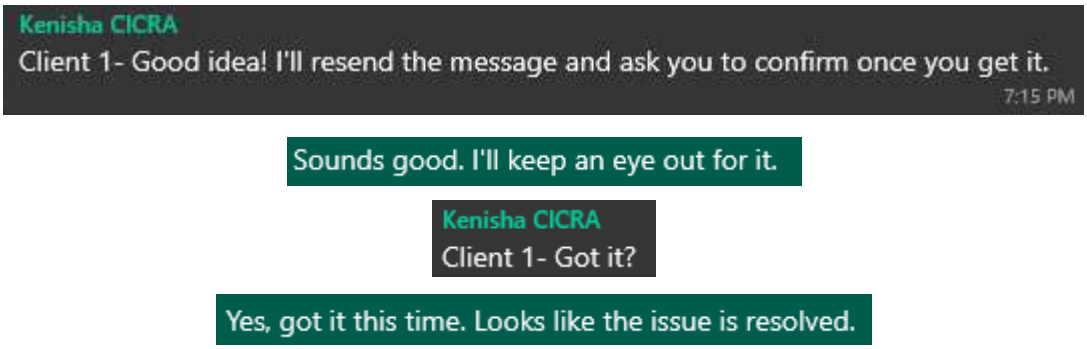
1. Assume one of your group members sent you a message. However, you have never received it. What went wrong? How do you make sure that the message is properly received?



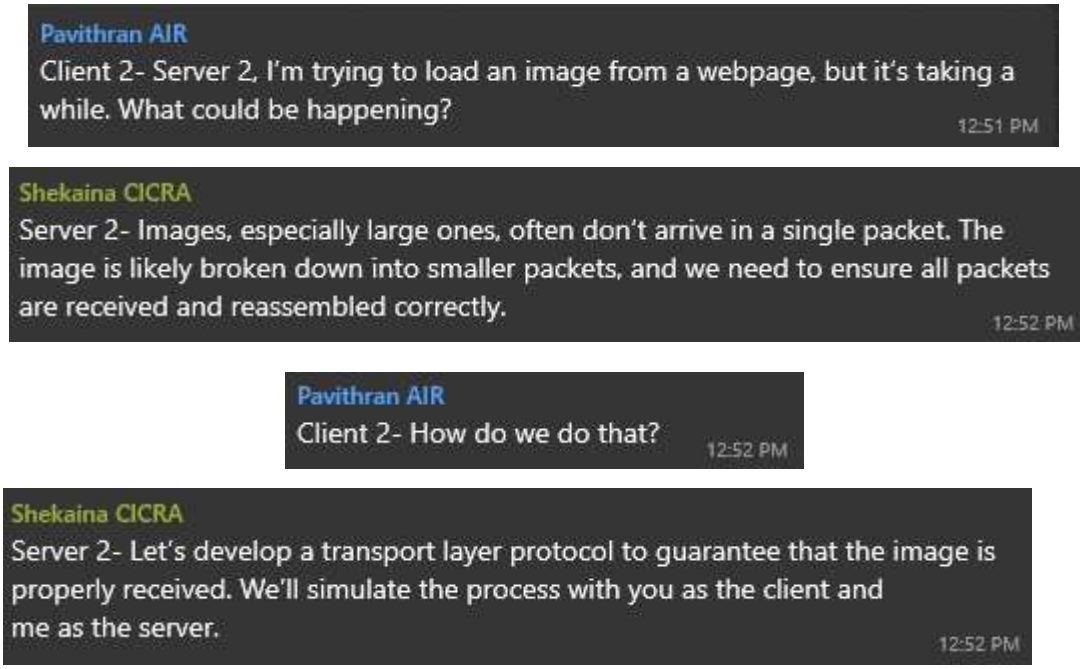




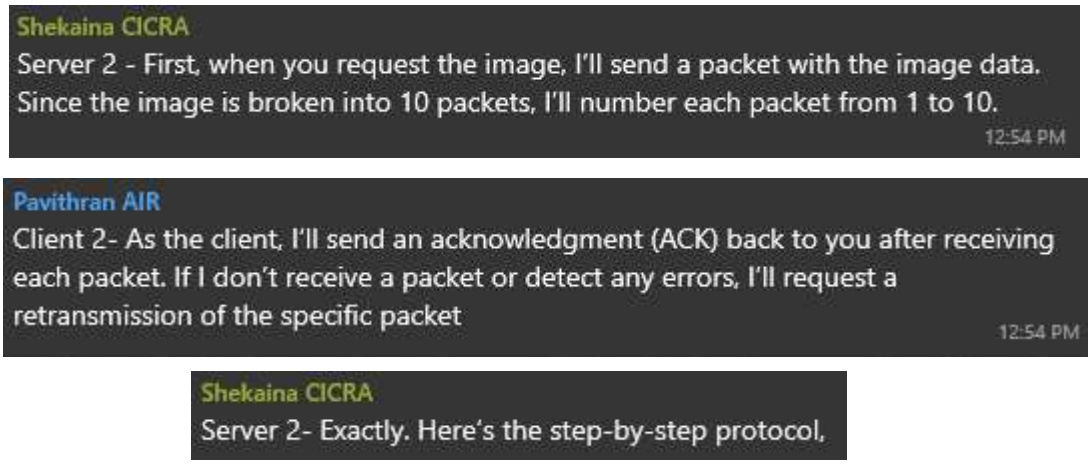




2. Assume you are trying to access a web page containing an image. Now, your job is to make sure that you get the image properly. The image cannot be received in one packet, and it will be broken down to 10 packets.



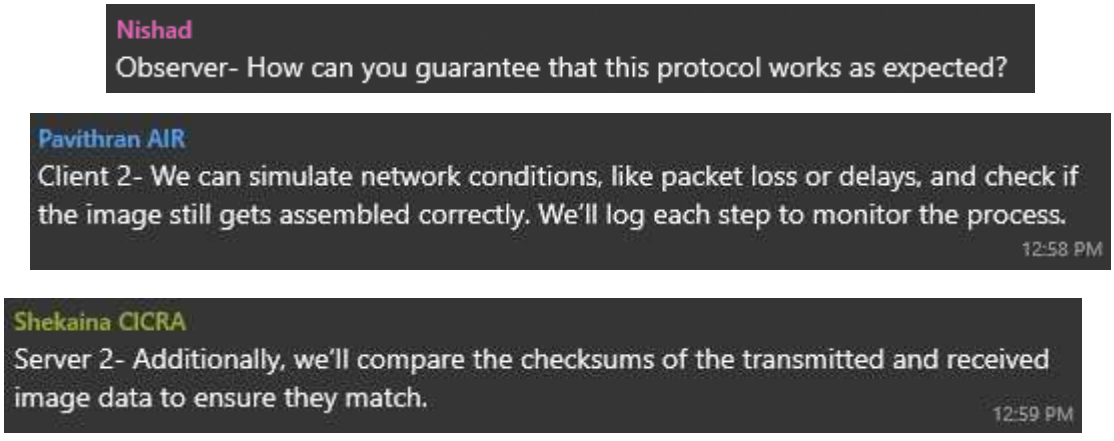
3. Partner up with one of your group members and develop a transport layer protocol to guarantee that the image is properly received by the client. One can act as the server where the webpage/image is stored and the other can act as the client who wants to receive the image (this means your group has 2 pairs of server-client networks).



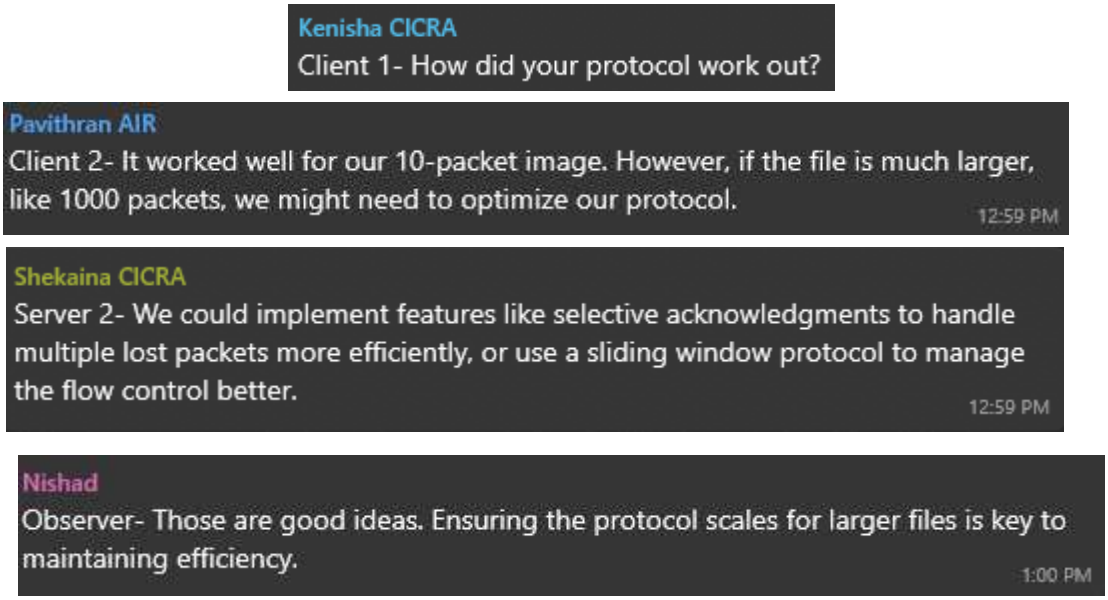
4. Write down protocol that you would use, including the communication between the client and server and all the messages that you are passing.

   - Client Request: Client 2 sends a request for the image.
   - Server Response: Server 2 begins sending packets, starting with packet 1.
   - Client ACK: Client 2 acknowledges the receipt of each packet (ACK 1, ACK 2, etc.).
   - Retransmission Request: If a packet is lost or corrupted, Client 2 requests retransmission of that specific packet.
   - Image Assembly: Once all packets are received, Client 2 assembles the image.

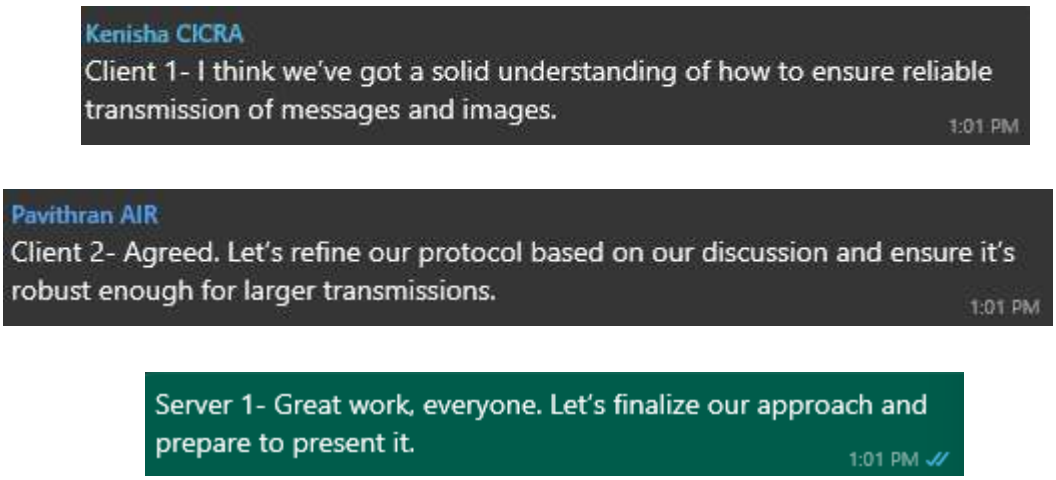
5. How do you guarantee that your protocol does the job as you expected?



6. Once you have finalized your protocol, check how other group members developed their protocols.

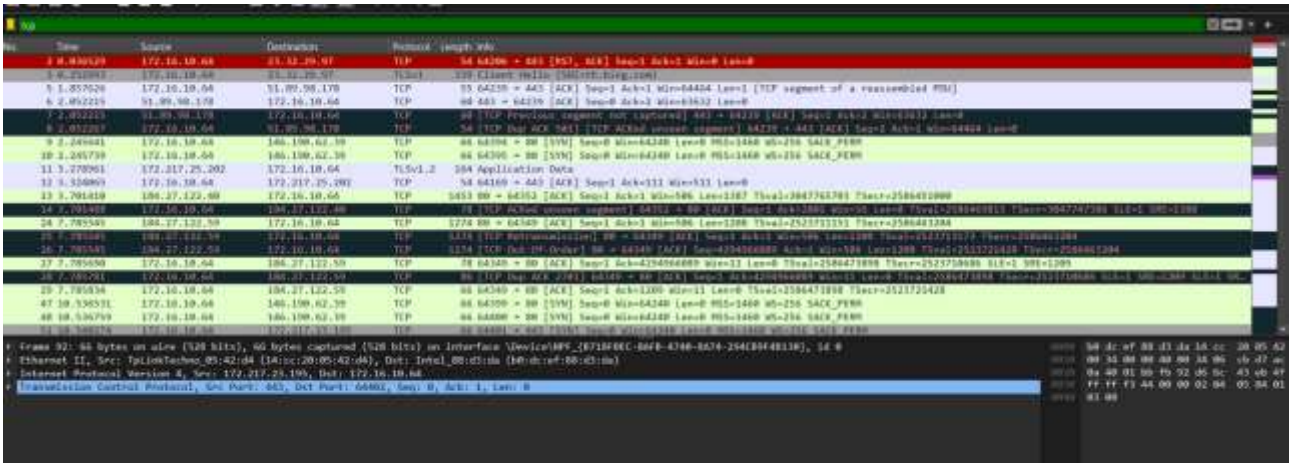


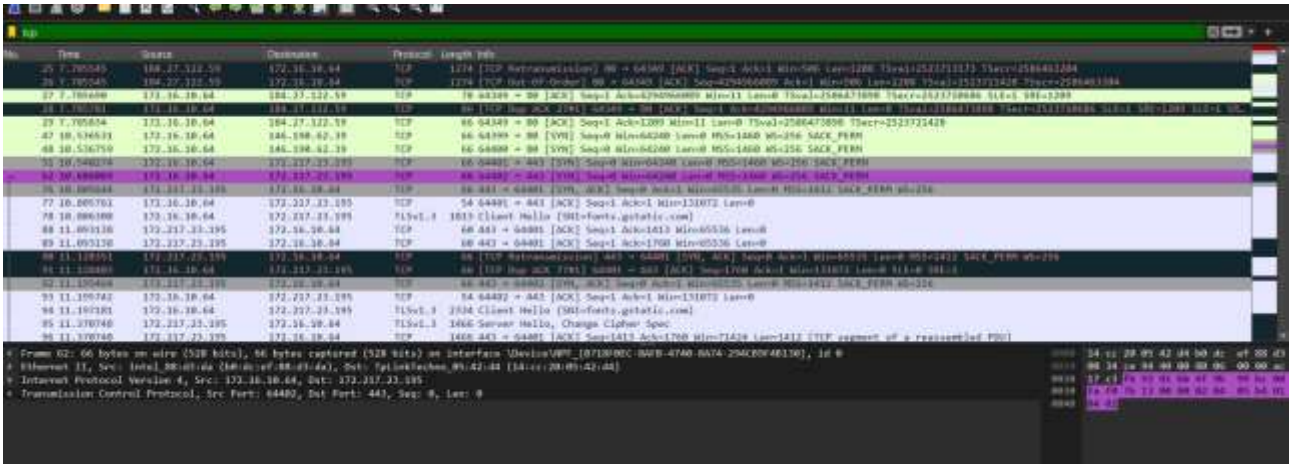
7. With your group, discuss whether your protocol valid when you need to access a large file that might be broken down into 1000 packets. If not, what are the changes you would like to make in your protocol?



Activity 2

1. You can analyze the TCP three-way handshake in Wireshark packet capture. Analyze the order of segments sent/received by two end systems to establish a TCP connection.
2. Analyze the segment headers used, their purpose and sizes.
3. Make sure you take the screenshots of TCP segments that will be used for your task submissions.

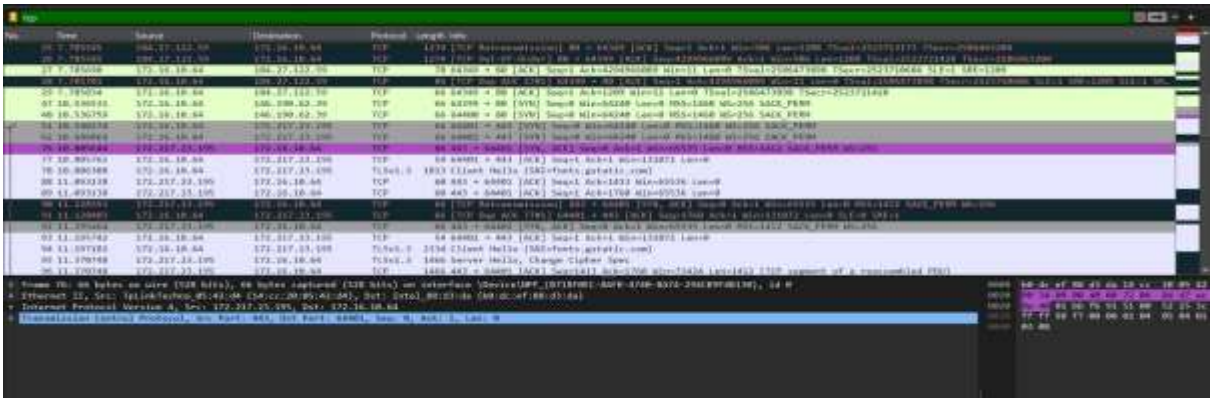




SYN Packet Analysis:

- **Source Port (Src Port)**: 64402
  - The client selected this port number in order to establish the connection. The port number is usually high and transient, used by the client.
- **Destination Port (Dst Port)**: 443
  - This shows that the server's port 443, which is frequently used for HTTPS traffic, is the destination of the SYN packet.
- **Sequence Number (Seq)**: 0
  - In this case, the client selected sequence number 0 as the starting point. The bytes of data sent from the client in this connection will be numbered starting at this point.

- **Length (Len)**: 0
    - Since this SYN packet is just used for the handshake and contains no data, the payload length is 0.



SYN-ACK Packet Analysis:

- **Source Port (Src Port)**: 443
    - This shows that the server is sending the SYN-ACK packet via port 443, which is frequently used for HTTPS.
- **Destination Port (Dst Port)**: 64401
    - The client-side port that started the connection is this one.
- **Sequence Number (Seq)**: 0
    - The server selected this as the first number in the sequence. This connection's sequence number begins at 0, which is typical.
- **Acknowledgment Number (Ack)**: 1
    - This is the number that verifies that the client sent the SYN packet and that it was received. The server sends the next expected sequence number, which is 0 + 1 = 1, in response to the client's likely initial sequence number of 0.
- **Length (Len)**: 0
    - Since the SYN-ACK packet is only being sent as part of the connection setup procedure and no actual data is being delivered yet, the payload length in this TCP segment is 0.

ACK Packet Analysis:

- **Source Port (Src Port)**: 64401
    - This is the client-side port number that was utilized to create the connection. It corresponds to the SYN-ACK packet's source port.

- **Destination Port (Dst Port)**: 443
    - This is the server's port number, which is 443 in this case, used for HTTPS.
- **Sequence Number (Seq)**: 1
    - Since 1 is the sequence number, it may be assumed that data transmission will use this as the subsequent sequence number. That being said, the sequence number in this instance hasn't increased over what was set in the SYN packet because the handshake is still in progress.
- **Acknowledgment Number (Ack)**: 1
    - The sequence number of the server from the SYN-ACK packet plus one represents the acknowledgment number that the client sent. A 1 for the acknowledgment number in the SYN-ACK packet signifies that the client is prepared to receive the following byte, which begins at sequence number 1, and validates receipt of the SYN-ACK packet, since the server's sequence number occurred to be 0.

.

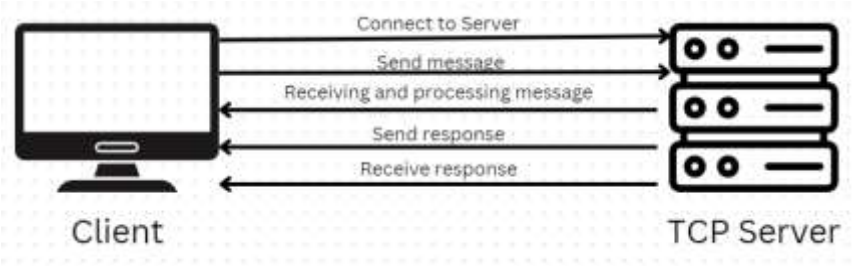
- **Length (Len)**: 0
    - The length of the payload in this packet is 0, consistent with the fact that this ACK packet is part of the handshake process and does not contain any data.

2. **Segment headers overview**

- **Source/Destination Port**: These numbers tell which programs or services are talking to each other.
- **Sequence Number**: This keeps track of where each piece of data is in the conversation, making sure everything arrives in the right order.
- **Acknowledgment Number**: This number confirms that data was received successfully.
- **Flags**: These tiny signals manage the connection, like starting it (SYN), confirming receipt of data (ACK), or ending it (FIN).
- **Window Size**: This shows how much data the receiver can handle at one time, helping control the flow of information.
- **Checksum**: This is like a safety check to make sure the data isn't messed up.
- **Options**: These are extra features that can improve how TCP works, like setting the biggest piece of data that can be sent at once (MSS).
- **Header Sizes**: The basic size is 20 bytes, but it can be 32 bytes or more if extra features are added.

Activity 3

1. First you need to draw a diagram to explain the operation and communication of the client-server program that uses TCP.



2. Then one of you can develop the client side and the other can develop the server side of the program (you can also develop both and demonstrate the outcome if you chose to do so)

```python
import socket

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server_socket.bind(('localhost', 65534))

    server_socket.listen(5)
    print("Server is listening on port 65534...")

    client_socket, client_address = server_socket.accept()
    print(f"Connection from {client_address} has been established.")

    message = client_socket.recv(1024).decode()
    print(f"Received message: {message}")

    num_chars = len(message)
    print(f"Number of characters received: {num_chars}")

    uppercase_message = message.upper()

    response = f"{num_chars} {uppercase_message}"

    client_socket.send(response.encode())

    client_socket.close()
    print("Connection closed.")

if __name__ == "__main__":
    start_server()
```

server.py code

```python
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    client_socket.connect(('localhost', 65534))

    message = "Hello SIT202"
    client_socket.send(message.encode())
    print(f"Sent message: {message}")

    response = client_socket.recv(1024).decode()

    print(f"Received response: {response}")

    client_socket.close()

if __name__ == "__main__":
    start_client()
```

client.py code

3. You need to show the output of the client- server program to demonstrate that your program works as expected. Please make sure to include the server and client codes and output (screenshots) in your lesson review.

```
Server is listening on port 65534...
```

TCP server listening for any messages

```
Sent message: Hello SIT202
Received response: 12 HELLO SIT202
```

Message sent to server and received back from server

```
Server is listening on port 65534...
Connection from ('127.0.0.1', 55441) has been established.
Received message: Hello SIT202
Number of characters received: 12
Connection closed.
```

Complete output from TCP server

Above and Beyond Tasks

- Discuss the differences between the following three protocols and the reasons for using these protocols.
    - Stop-and-wait
    - The Stop-and-wait protocol is a simple Automatic method where the sender sends one packet and waits for acknowledgments from the receiver before sending the next frame.
    - Go-back-N
    - In this protocol, the sender can send multiple frames before needing an acknowledgment. Howver if there an error detected in the any of the frames in the window all frames in that window must be retransmitted
    - Selective repeat
    - Selective repeat can also send multiple frames before receiving an acknowledgment and if there's an error in a frame, only thst frame will be retransmitted, not the entire window. Selective repeat protocol is therefore considered an improvement from Go-Back-N protocol.

- Why do we need congestion control in TCP? Explain your answer.
    - Congestion control is needed in TCP for maintaining network stability and ensuring efficient us of available bandwifth. Withput congestion control, network instabilities can occur due to uncontrolled data flow where congestion collapses occur.