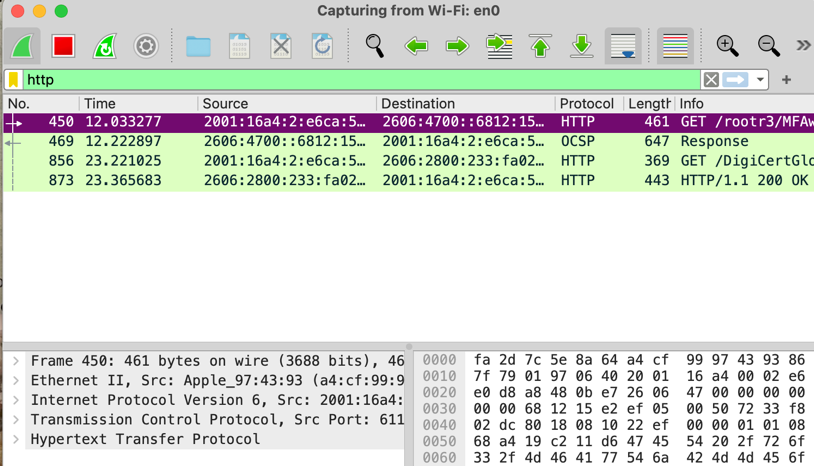
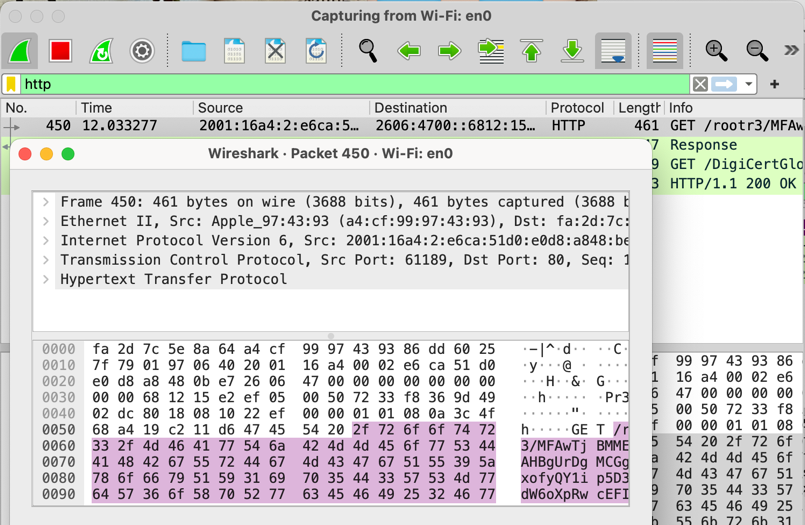
**Lab week 2**

**Task 2: Filter HTTP packets and analyze them.**

**Step 1:** In the filter bar, type http and press Enter. This filters out only the HTTP packets from the capture. (A)



**Step 2:** Select any HTTP packet to view its details. (B)



**Step 3:** Observe the HTTP request and response messages. Note the method (GET, POST), URL, response codes (200 OK, 404 Not Found), etc.

\*Note: From figure (A):

1. The methods:

GET

1. URL

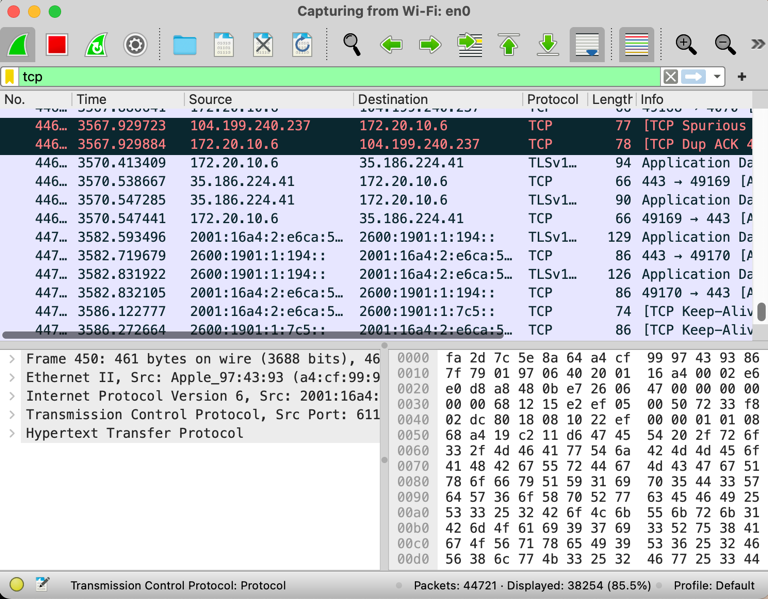
https://myqu.qu.edu.sa

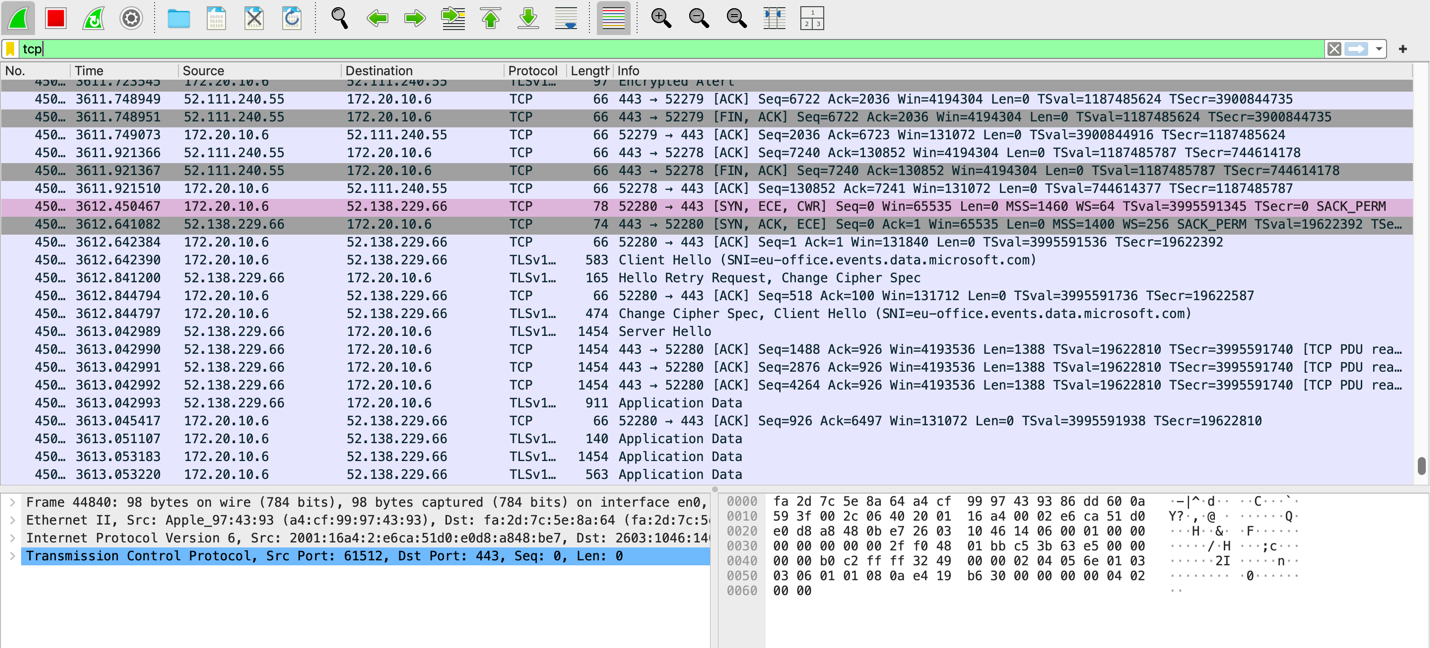
1. Response codes:

HTTP 443 HTTP/1.1 200 OK

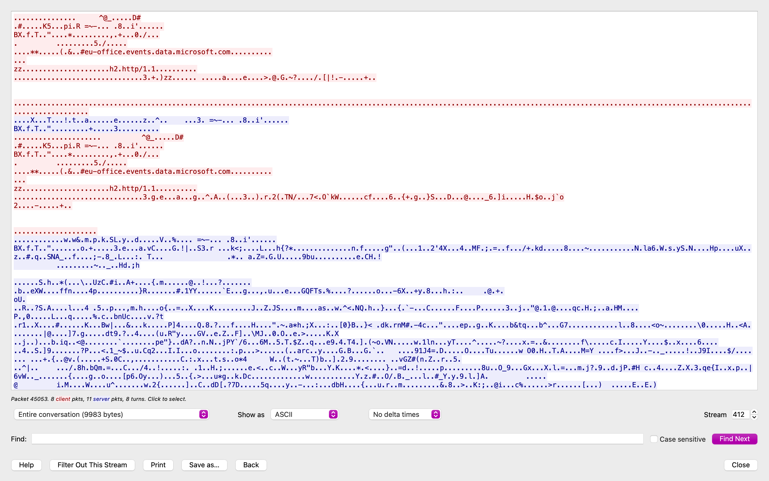
**Part 2: Analyzing TCP/IP Traffic.**

**Task 1: Filter TCP packets**

**Step 1:** Clear the previous filter and type TCP to focus on TCP packets. (C)

**Step 2:** Select a TCP packet related to your HTTP request/response. (D)



**Step 3:** Right-click on the packet and select "Follow" -> "TCP Stream" (E)

**Step 4:** This shows the entire conversation between the client and server.

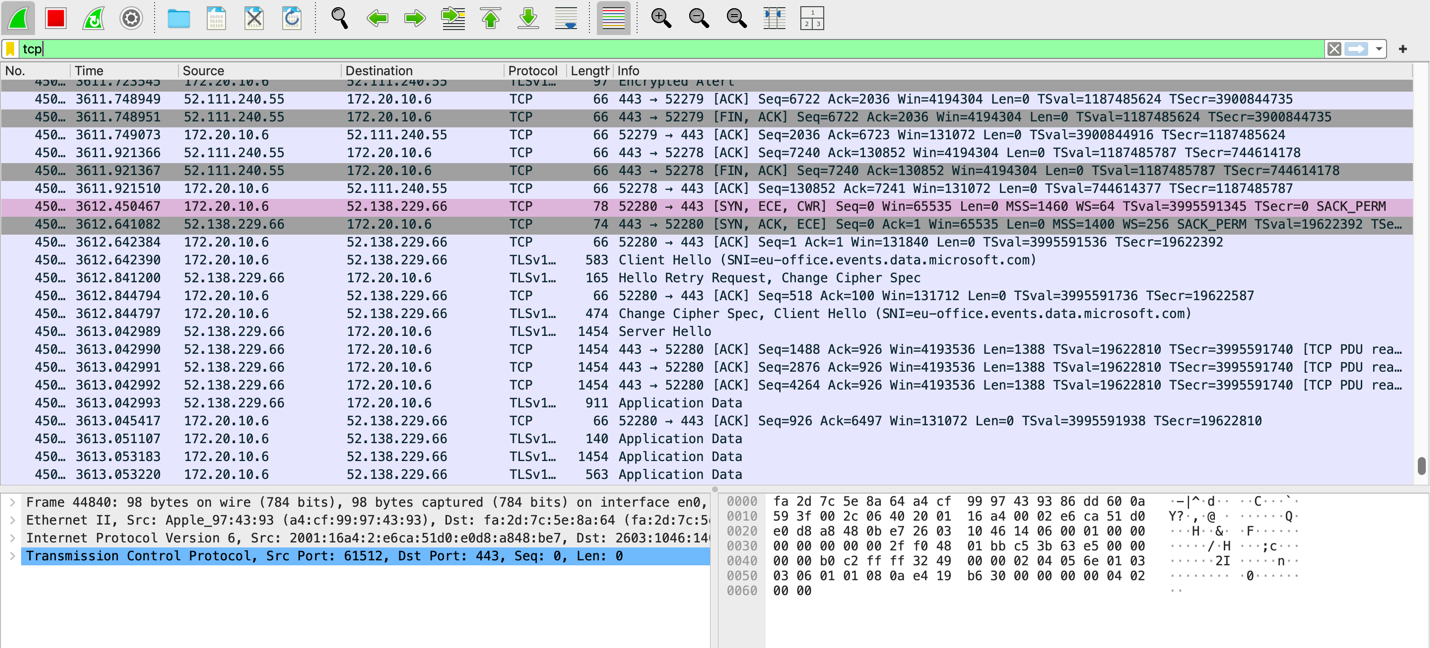
**Task 2: Analyze TCP handshake and investigate Data Transfer and Termination**

**Step 1:** Find and select packets related to the TCP three-way handshake: (D)

o SYN: Initiates a connection.

o SYN-ACK: Acknowledges and responds to the SYN.

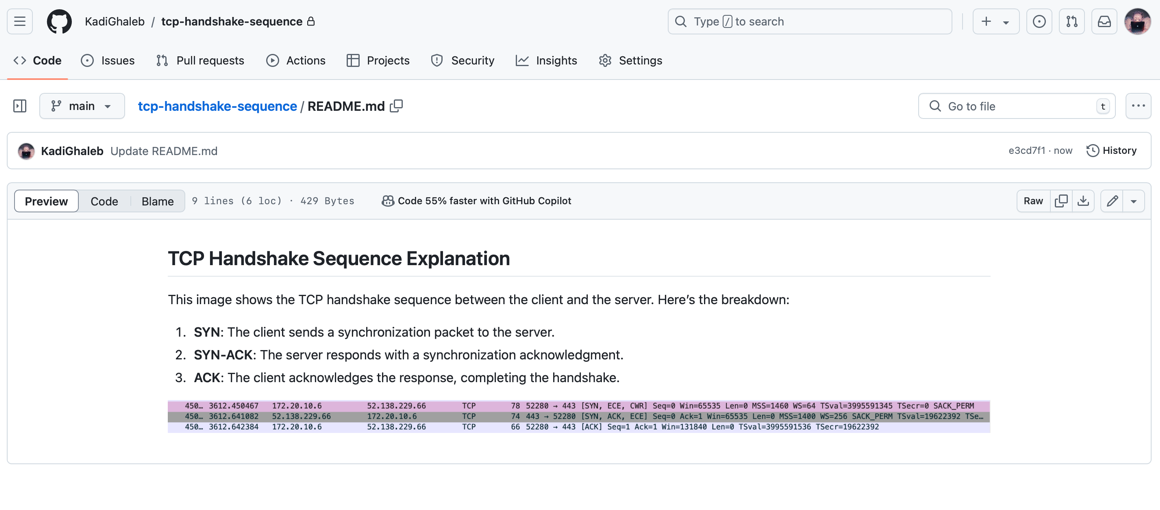
o ACK: Acknowledges the SYN-ACK and establishes the connection.





**Step 2:** Note the sequence and acknowledgment numbers. Screenshot and upload your image to your online git repository.

1. **First Packet (SYN from client)**:
   * Seq=0, Ack=0
   * The client initiates the connection.
2. **Second Packet (SYN-ACK from server)**:
   * Seq=0, Ack=1
   * The server acknowledges the SYN packet from the client.
3. **Third Packet (ACK from client)**:
   * Seq=1, Ack=1
   * The client acknowledges the SYN-ACK packet from the server.

**Step 3:** Observe the data packets exchanged between the client and server. Take a screenshot and upload it to your online git repo. (F)

**Step 4:** Look at the TCP termination process (FIN, ACK packets).

1. **Client FIN Packet:**
   * **Description:** Client initiates termination
   * **Packet:** 52280 -> 443 [FIN, ACK] Seq=1 Ack=1 Win=131840 Len=0 TSval=3995591536 TSecr=19622392
2. **Server ACK Packet:**
   * **Description:** Server acknowledges client’s FIN
   * **Packet:** 443 -> 52280 [ACK] Seq=1 Ack=2 Win=65535 Len=0 TSval=19622393 TSecr=3995591536
3. **Server FIN Packet:**
   * **Description:** Server initiates termination
   * **Packet:** 443 -> 52280 [FIN, ACK] Seq=2 Ack=2 Win=65535 Len=0 TSval=19622394 TSecr=3995591536
4. **Client ACK Packet:**
   * **Description:** Client acknowledges server’s FIN
   * **Packet:** 52280 -> 443 [ACK] Seq=2 Ack=3 Win=131840 Len=0 TSval=3995591540 TSecr=19622394

**Part 3: Capturing and Analyzing UDP Traffic**

**Task 1: Generate UDP traffic and capture packets**

**Step 1:** Open a network application that uses UDP (e.g., streaming video, VoIP software, or custom script). “I have used **YOUTUBE**”

**Step 2:** Start the application to generate UDP traffic.

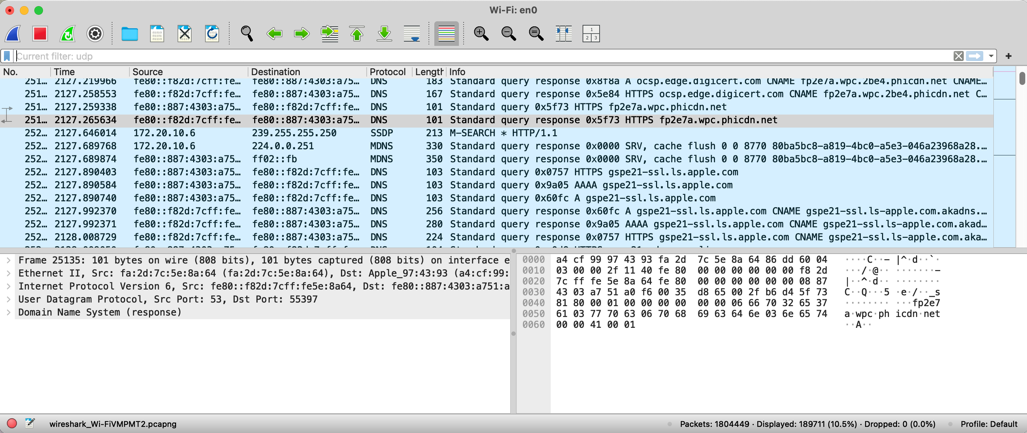
**Step 3:** Start capturing packets in Wireshark while the UDP application is running.

**Step 4:** After sufficient traffic is generated, stop capturing packets.

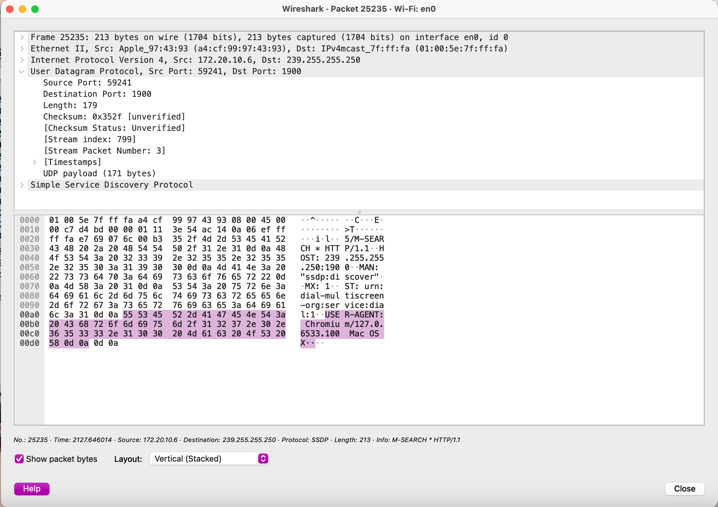
**Task 2: Filter and analysis UDP Packets**

**Step 1:** In the filter bar, type UDP and press Enter.

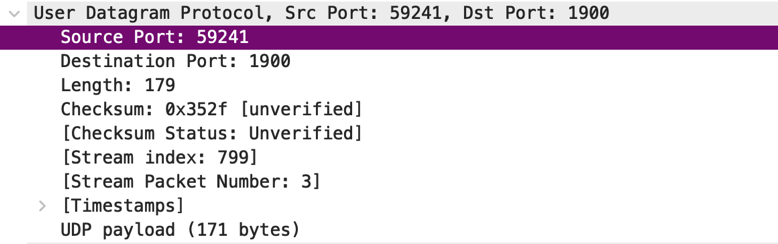
**Step 2:** This filters out only the UDP packets from the capture. (G)



**Step 3:** Select any UDP packet to view its details.



**Step 4:** Observe the source and destination ports, length, and data.(H)

****

**Step 5:** Compare the simplicity of UDP headers with TCP headers.

UDP headers are simpler, with only 8 bytes: Source Port, Destination Port, Length, and Checksum. TCP headers are more complex, starting at 20 bytes and include additional fields like Sequence Number, Acknowledgment Number, and Flags for managing connections and ensuring reliable data delivery.

**Part 4: Comparing TCP and UDP by filling in the following tables. Save your work (e.g., in an MS Word document), and upload it to your online git repo.**

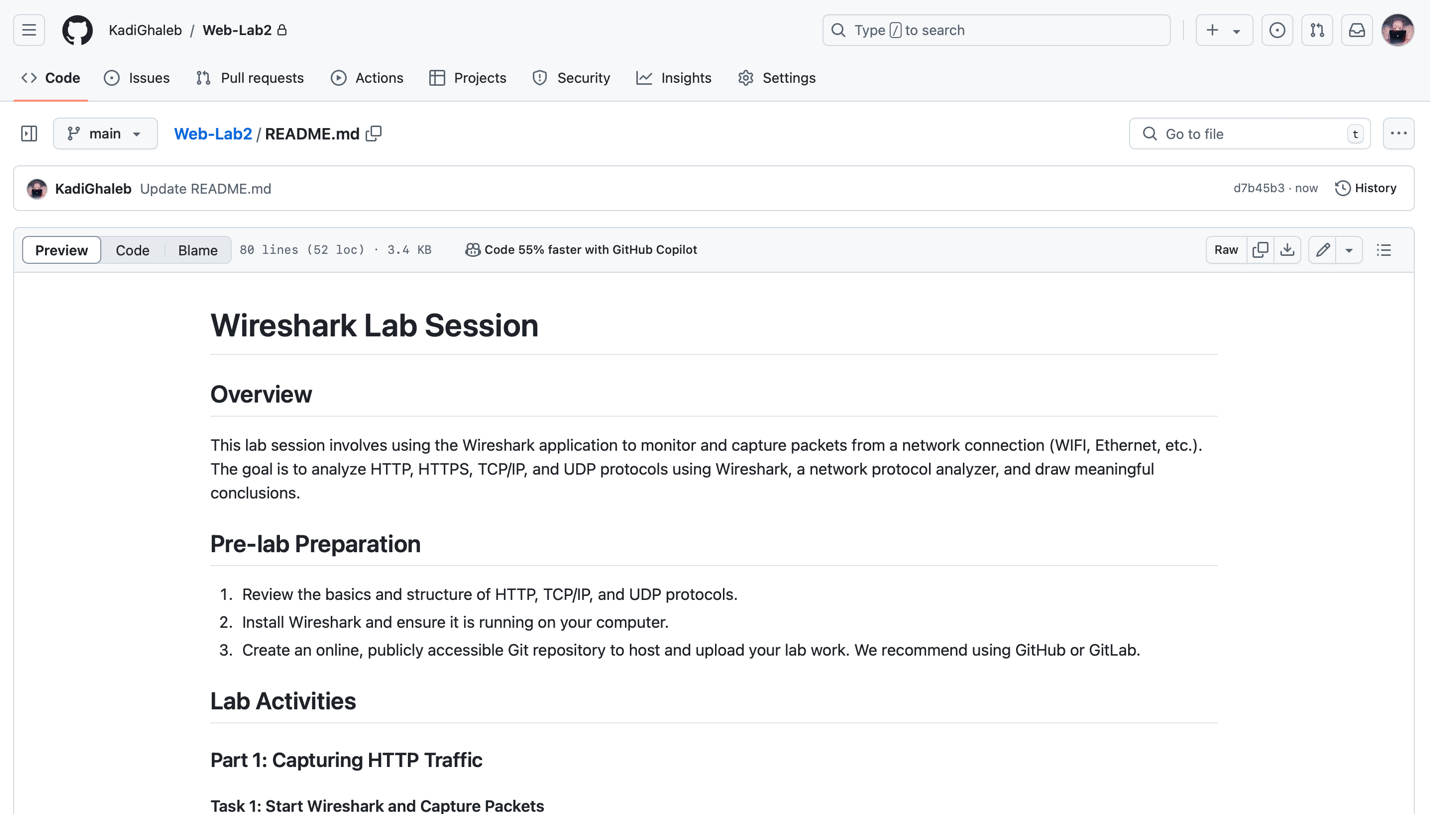
**Task 1: Fill in the following table and provide reasons.**

|  |  |  |
| --- | --- | --- |
|  | TCP or UDP | Reasons |
| Reliability and Connection  Establishment | **TCP** | Establishes a connection with a handshake (SYN, SYN-ACK, ACK). Ensures reliable data transfer through acknowledgments and retransmissions. |
| Data Integrity and Ordering | **TCP** | Guarantees data is received in the correct order and uses sequencing and checksums to ensure data integrity. |

**Task 2: Identify the use Cases and Performance of TCP and UDP.**

|  |  |  |
| --- | --- | --- |
|  | TCP | UDP |
| Use Cases | Suitable for applications requiring reliable communication, such as web browsing (HTTP), email (SMTP, IMAP), and file transfer (FTP). | Ideal for applications where speed is crucial and occasional data loss is acceptable, such as live streaming, online gaming, and VoIP. |
| Performance | Provides reliable data transfer with guaranteed delivery and order, but may introduce higher latency due to connection establishment and error correction. | Offers lower latency and faster data transfer since there is no connection establishment or error correction, but can result in packet loss or out-of-order delivery. |

**The entire file was uploaded to my GitHub accoount: (I)**

****

**Student: Kadi Ghaleb Alharbi**

**ID: 412205583**