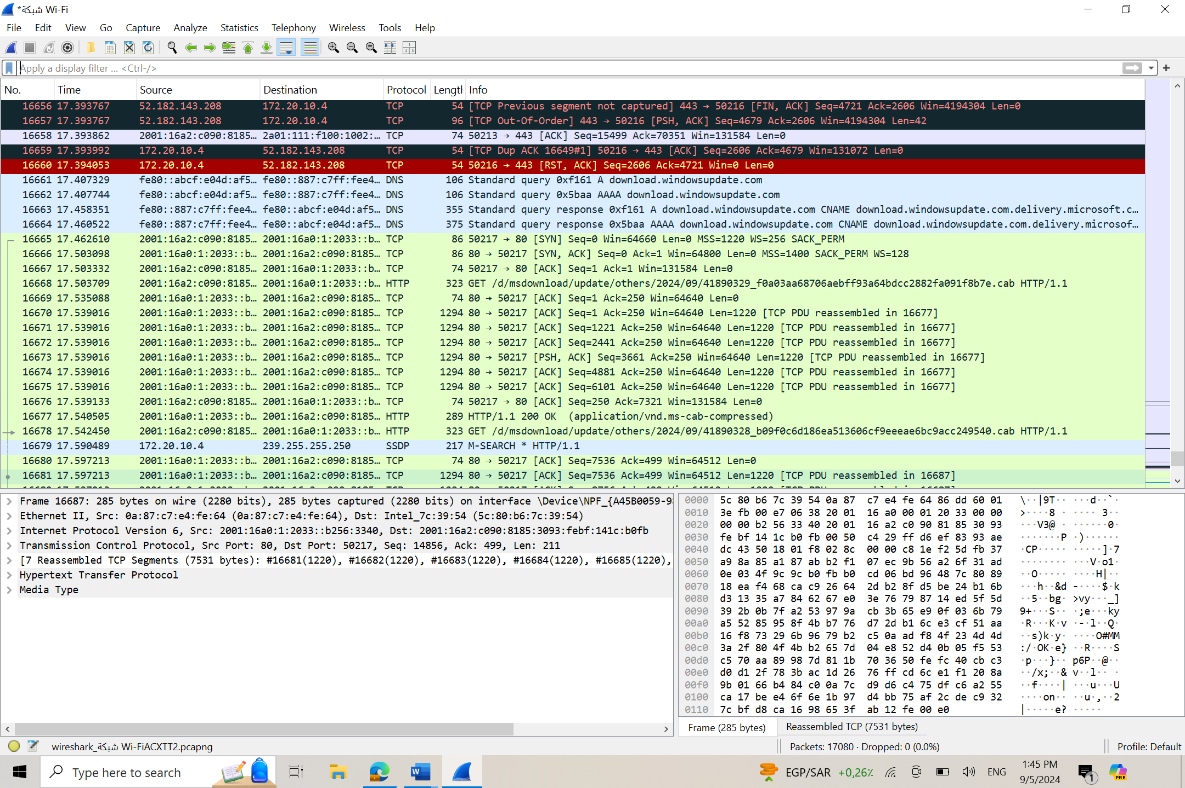
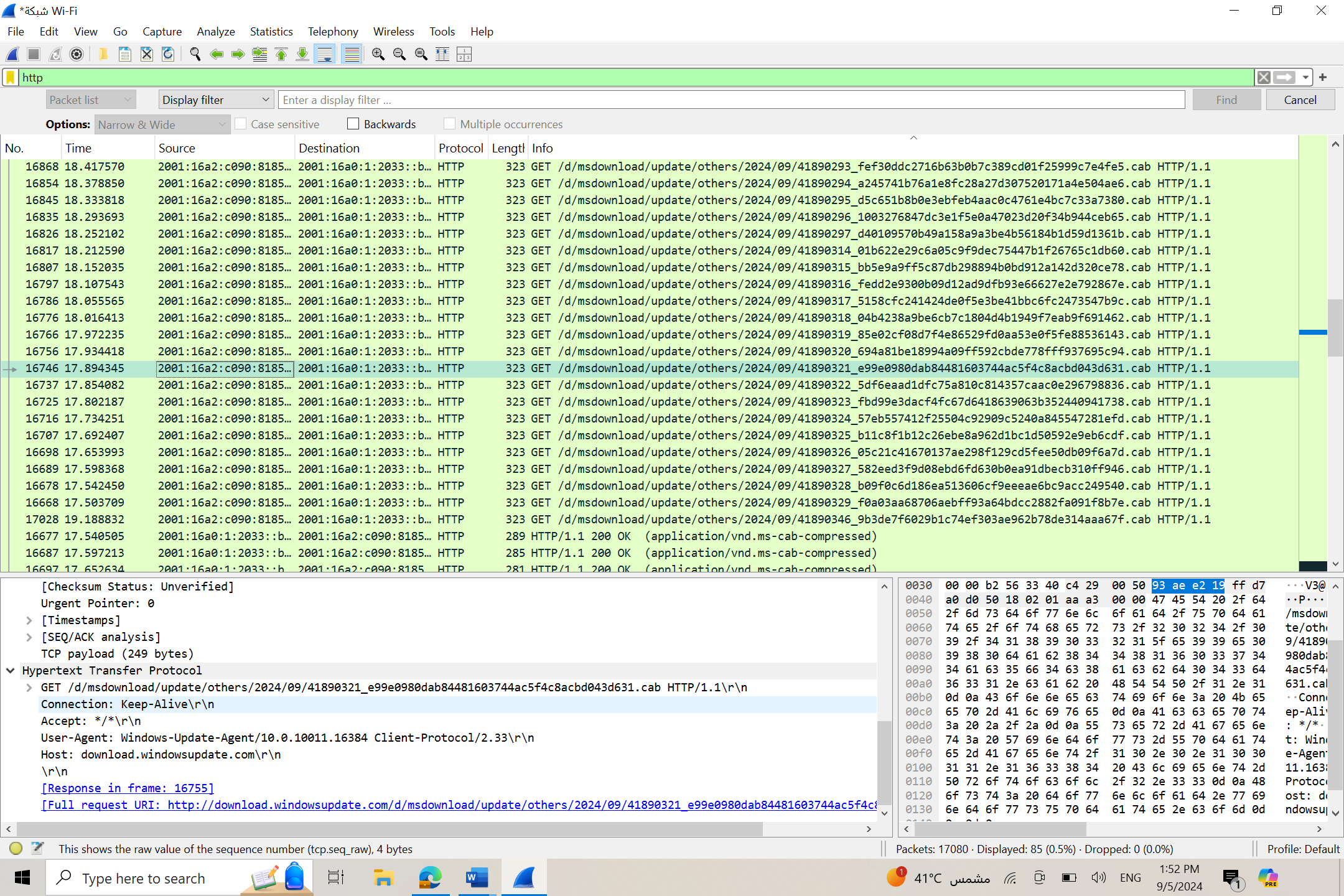
**Lab week2**

P1-TASK1: Start Wireshark and capture packets.



P1-TASK2: Filter HTTP packets and analyze them:

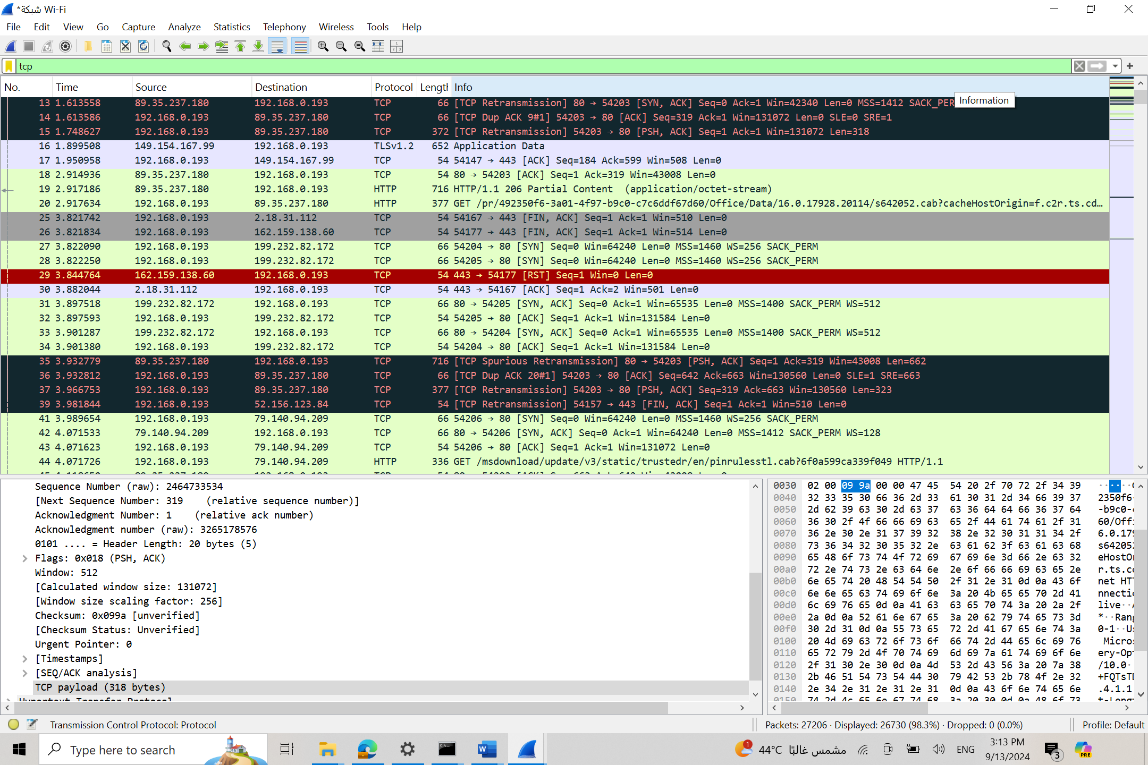


**Observe the HTTP request and response messages. Note the method (GET, POST), URL, response codes (200 OK, 404 Not Found(:**

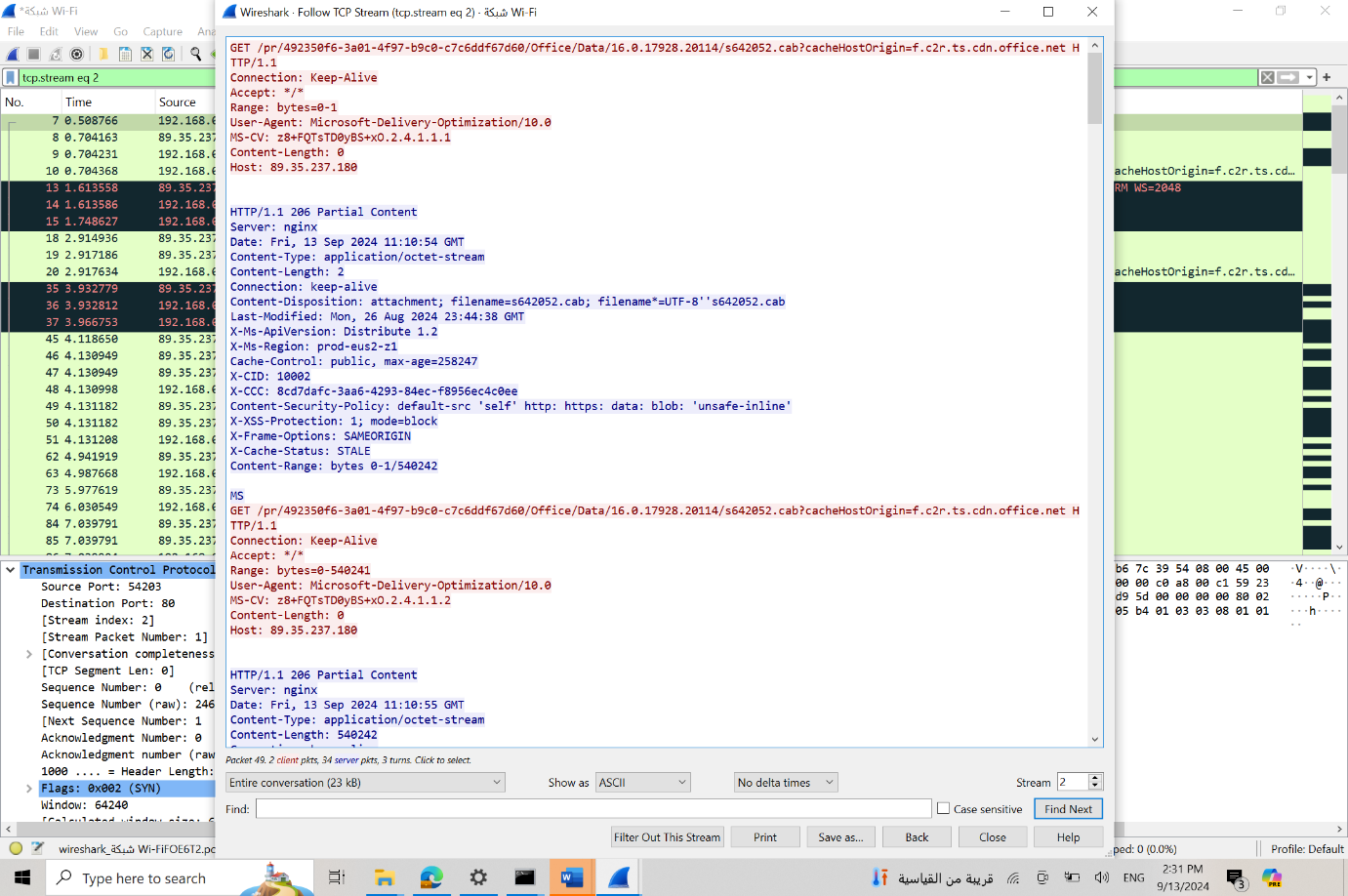
-**Method:** GET  
-**URL:** http://download.windowsupdate.com/msdownload/update/others/2024/09/41890321\_e99e0980dab84481603744ac5f4c8acbd043d631.cab  
-**Response Code:** 404 - File or directory not found.

P2-Task1: Filter TCP packets

**Filter TCP packets**



**conversation between the client and server**

****

P2-Task 2: Analyze TCP handshake and investigate Data Transfer and Termination

**Note the sequence and acknowledgment numbers.**

First packet (SYN from client):

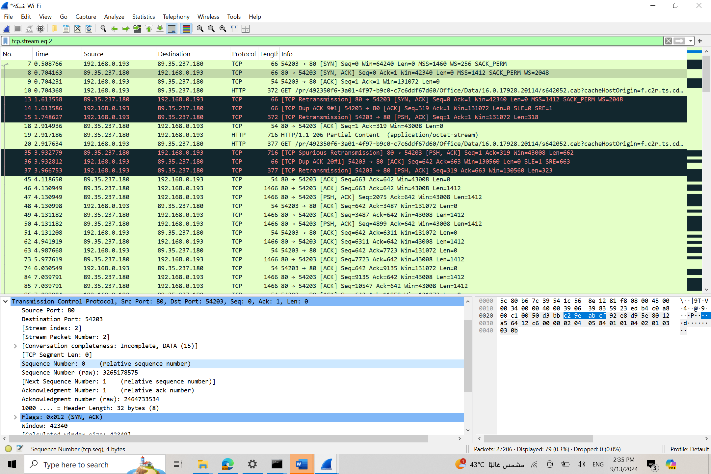
Seq=0, Ack=0

A screenshot of a computer

Description automatically generated

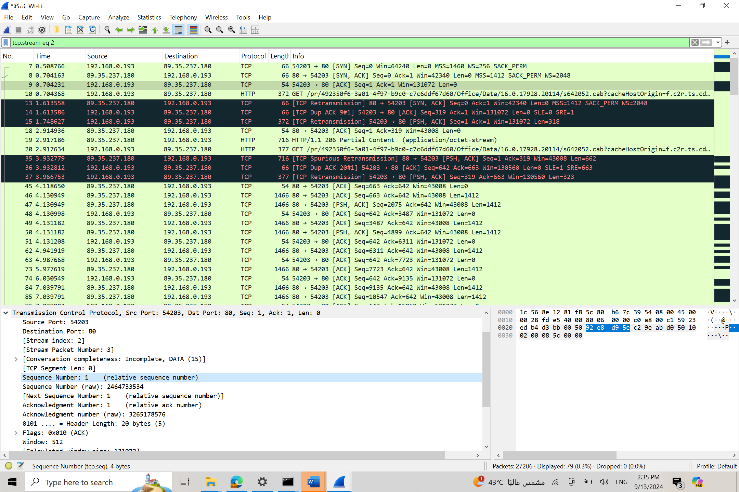
Second packet (SYN-ACK from server):

Seq=0, Ack=1

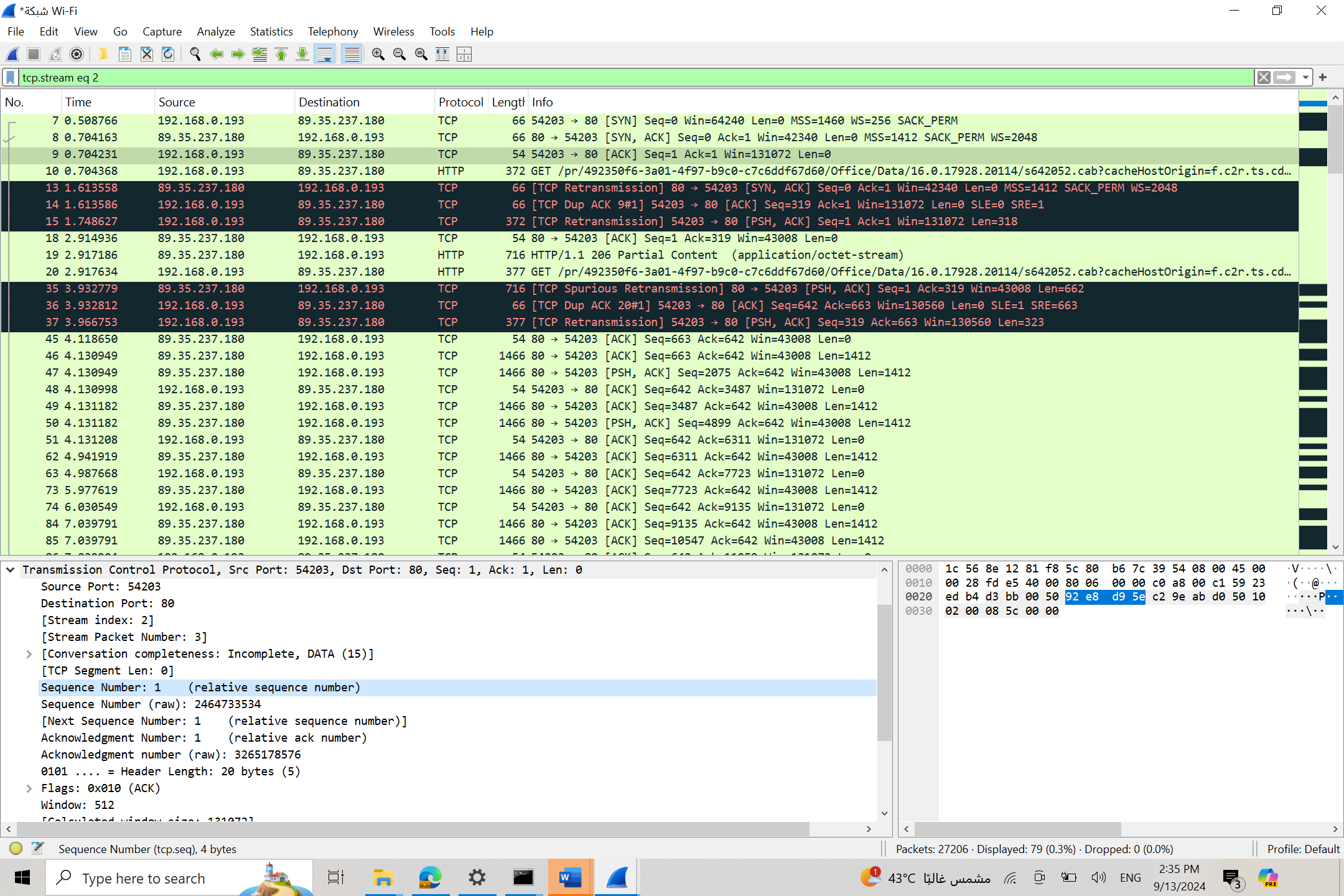


Third packet

Seq=1, Ack=1

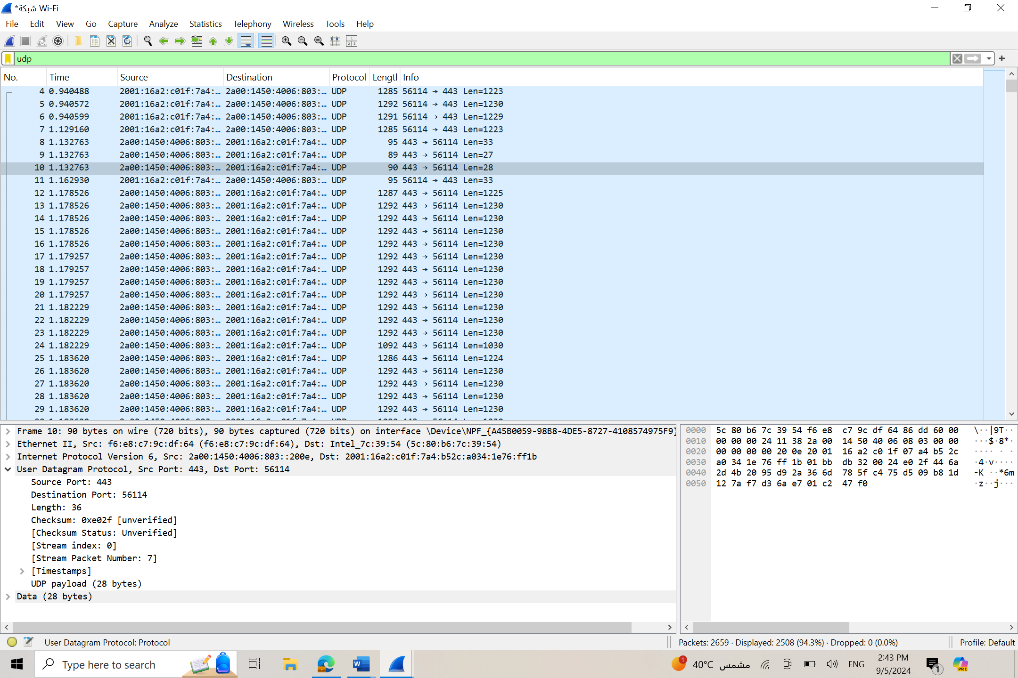


**Observe the data packets exchanged between the client and server**



* **SYN (Synchronization)**:
  + The client sends a SYN packet to the server to initiate a connection. This packet is essentially a request to start communication and synchronize the sequence numbers.
* **SYN-ACK (Synchronization Acknowledgment)**:
  + The server responds to the SYN packet with a SYN-ACK packet. This packet serves two purposes: it acknowledges the receipt of the SYN packet from the client and sends its own SYN request back to the client.
* **ACK (Acknowledgment)**:
  + The client sends an ACK packet back to the server, acknowledging the receipt of the server's SYN-ACK packet. This completes the three-way handshake, establishing a TCP connection for further data exchange.

P3: Capturing and Analysing UDP Traffic



**Observe the source and destination ports, length, and data.**

**Observation:**

Source port:443

Destination port: 56114

Length: 36

UDP payload :28 bytes

**Compare the simplicity of UDP headers with TCP headers.**

* **UDP Header:**
  + Only **8 bytes** long with **4 fields**: Source Port, Destination Port, Length, and Checksum.
  + Simple, minimal overhead, no connection management, making it fast but less reliable.
* **TCP Header:**
  + **20 to 60 bytes** long with **12 fields**, including Sequence Number, Acknowledgment Number, Flags, Window Size, and more.
  + Complex, with fields for reliable communication, ordering, and error-checking, adding overhead and reducing speed.

P4: comparing TCP and UDP by filling in the following tables

**Comparing TCP and UDP by filling in the following tables.**

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

|  |  |  |
| --- | --- | --- |
|  | TCP or UDP | Reasons |
| **Reliability and Connection Establishment** | TCP | TCP establishes a connection before data transmission (3-way handshake), ensuring reliability. |
| **Data Integrity and Ordering** | TCP | |  | | --- | |  |  |  | | --- | | TCP ensures data packets are delivered in order and checks for errors, guaranteeing data integrity. | |

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

|  |  |  |
| --- | --- | --- |
|  | TCP | UDP |
| **Use cases** | Web browsing, email, file transfer, remote access (e.g., SSH), applications requiring reliability and data integrity. | Streaming media, online gaming, Voice over IP (VoIP), applications where speed is prioritized over reliability. |
| **Performance** | TCP has higher overhead due to connection establishment, acknowledgment, and retransmission processes. It is slower but more reliable. | |  | | --- | |  |  |  | | --- | | UDP has lower overhead and is faster because it doesn't establish connections or guarantee packet delivery. It is suitable for real-time applications. | |

**Student Name:** Emtenan Alfozan

**Student Number:**421215099