**Task 5: Capture and Analyze Network Traffic Using Wireshark**

**📌 Objective**

The objective of this task was to capture live network packets using Wireshark, identify basic protocols, apply filters, and summarize observations.

**🛠 Tools Used**

* Wireshark (latest version)
* Python (http.server module) for generating traffic
* Windows PowerShell / Linux terminal for ping, nslookup, and curl
* OS: Windows 11 (Host Machine)

**🔍 Step-by-Step Process**

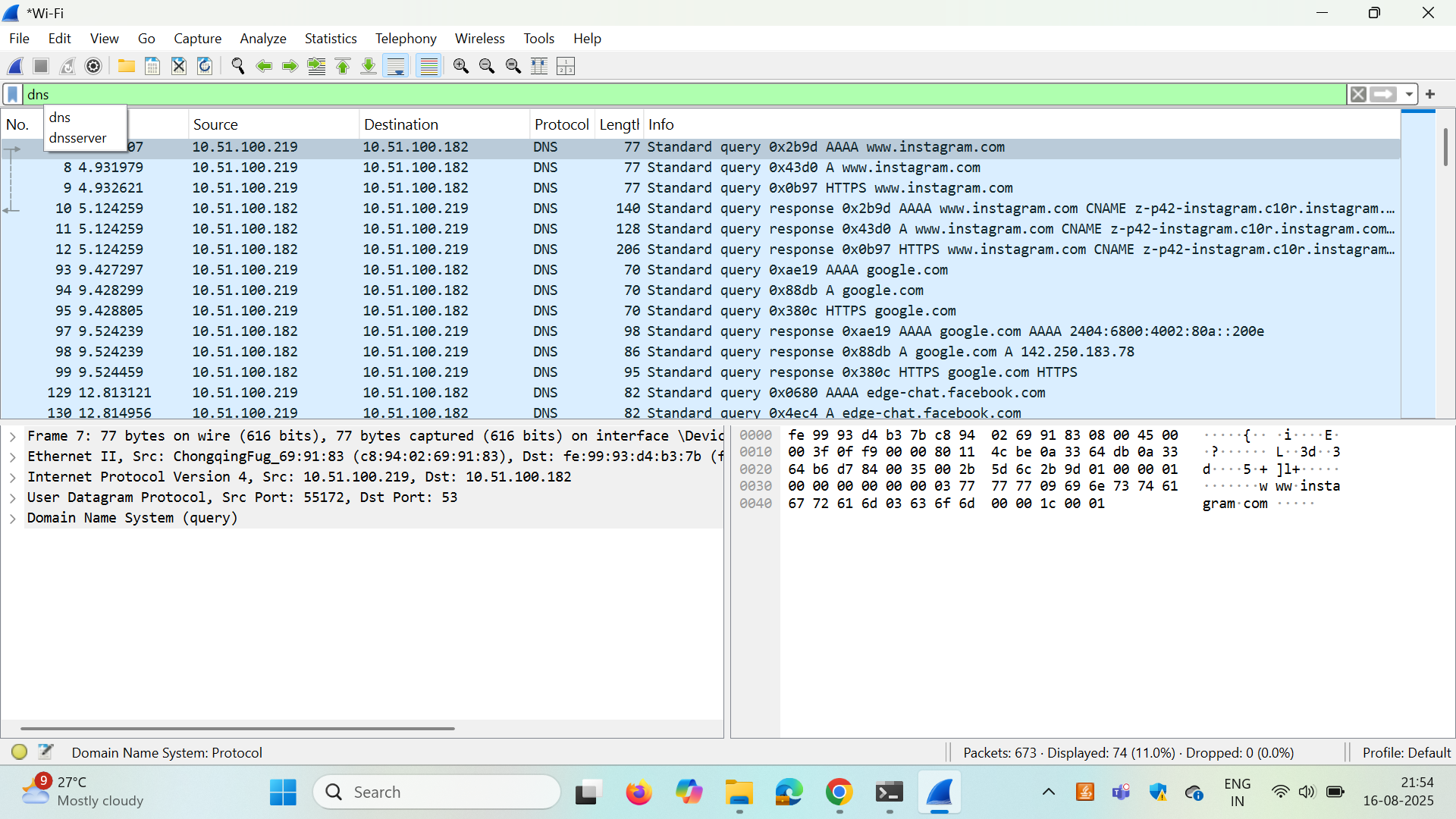
1. Installed Wireshark and Npcap driver (Windows).
2. Opened Wireshark and selected the active interface (Wi-Fi).
3. Started packet capture (blue shark fin button).
4. Generated test traffic while capture was running:
   * DNS Query: nslookup example.com
   * ICMP: ping 8.8.8.8
   * HTTP Request: curl http://neverssl.com/ -I
   * Opened a website in browser to capture HTTPS (TLS).
5. Stopped capture after ~60 seconds (red square button).
6. Applied filters to isolate protocols (dns, icmp, tcp, http, tls).
7. Exported capture as task5\_capture.pcap.

**📑 Protocols Identified**

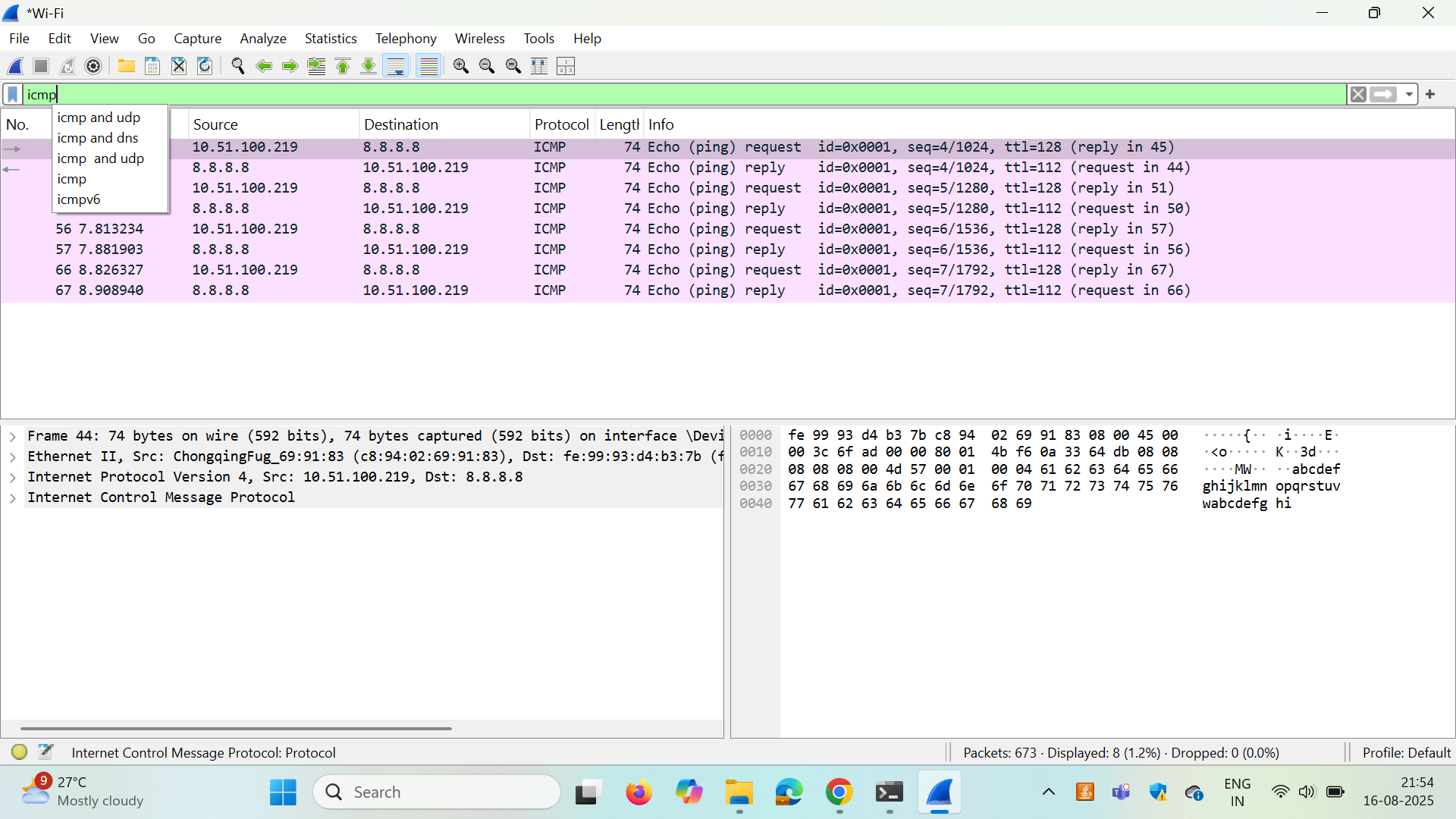
* **DNS**: Queries and responses for domain lookups.
* **ICMP**: Echo requests and replies to/from 8.8.8.8.
* **TCP**: Three-way handshake (SYN, SYN/ACK, ACK).
* **HTTP**: HEAD request and 200 OK response from neverssl.com.
* **TLS**: ClientHello and ServerHello messages observed.

**📊 Example Outputs**

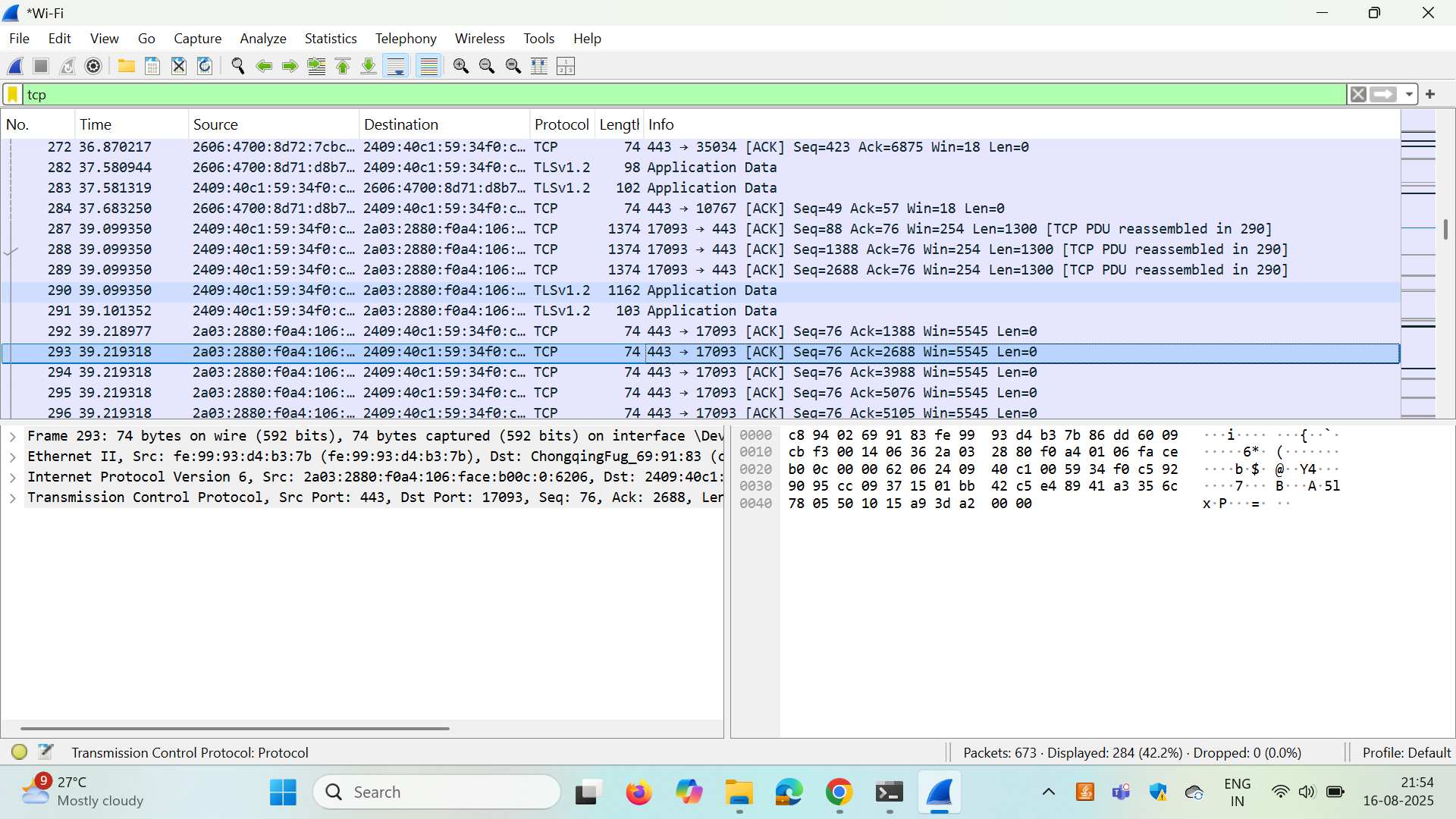
* **DNS Query Packet Example:**  
  Transaction ID, Flags, QNAME=example.com, Answer IP.



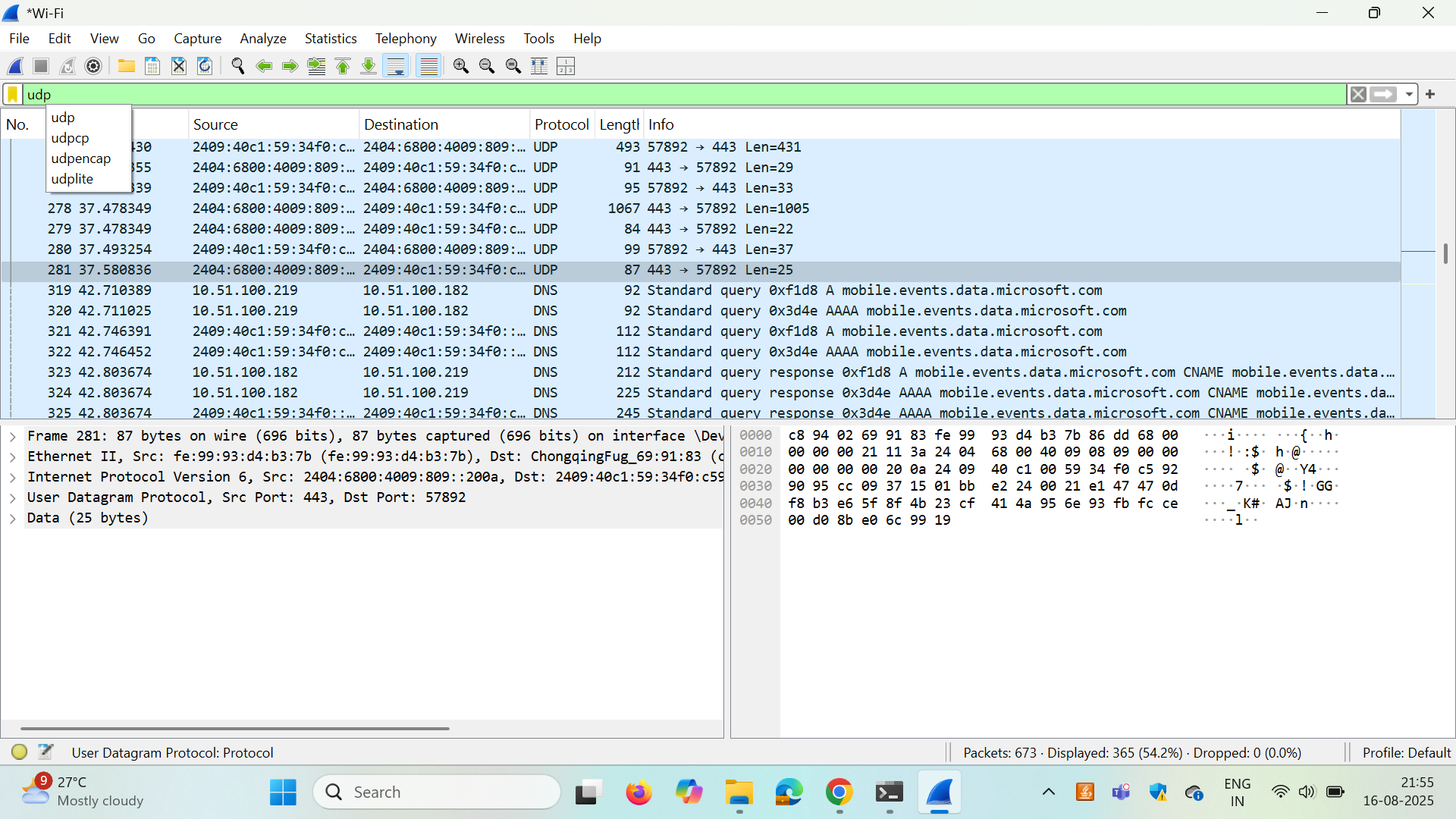
* **ICMP Packet Example:**  
  Type 8 (echo request), Type 0 (echo reply).



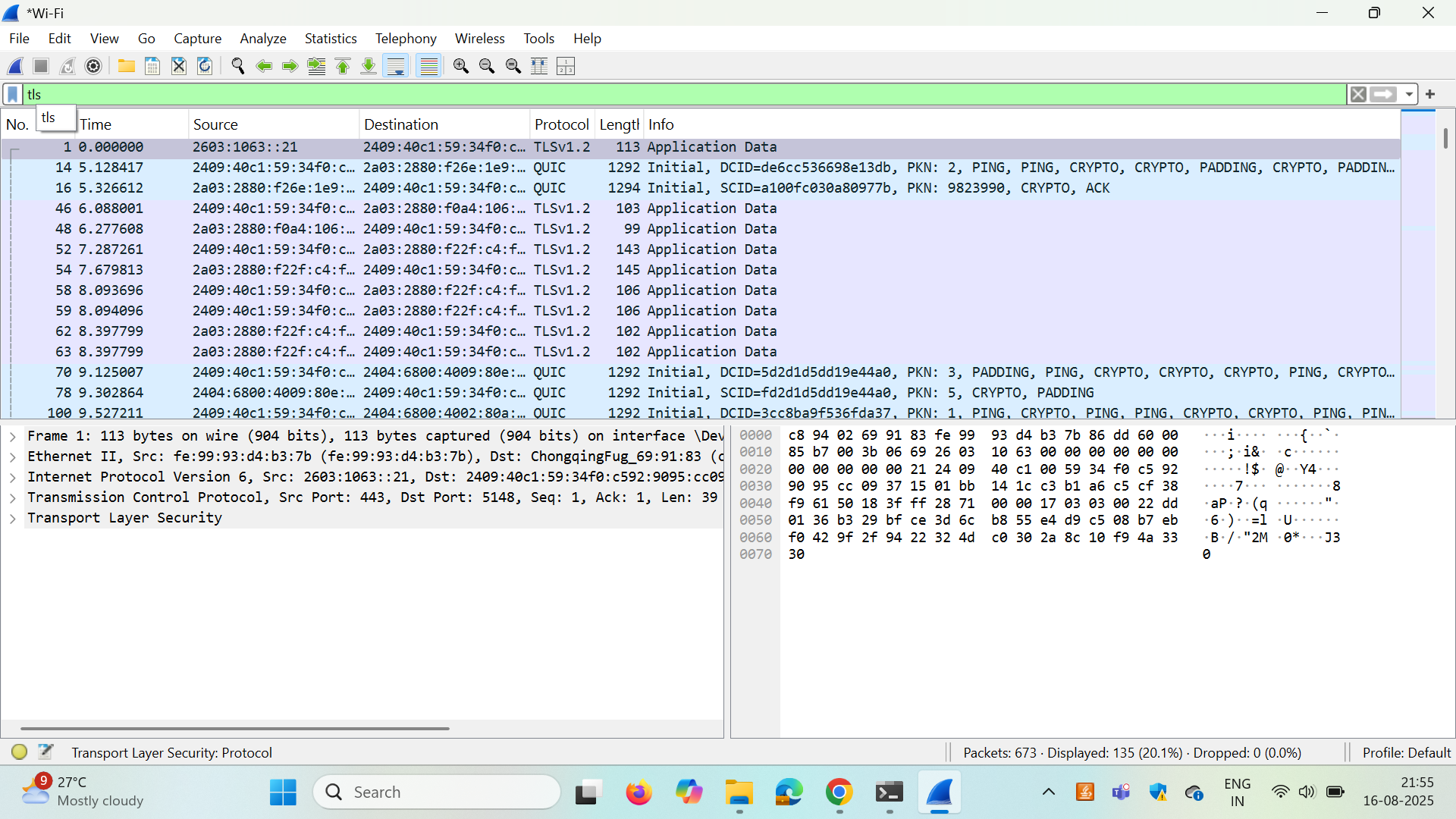
* **TCP Example:**  
  SYN → SYN/ACK → ACK sequence establishing connection.



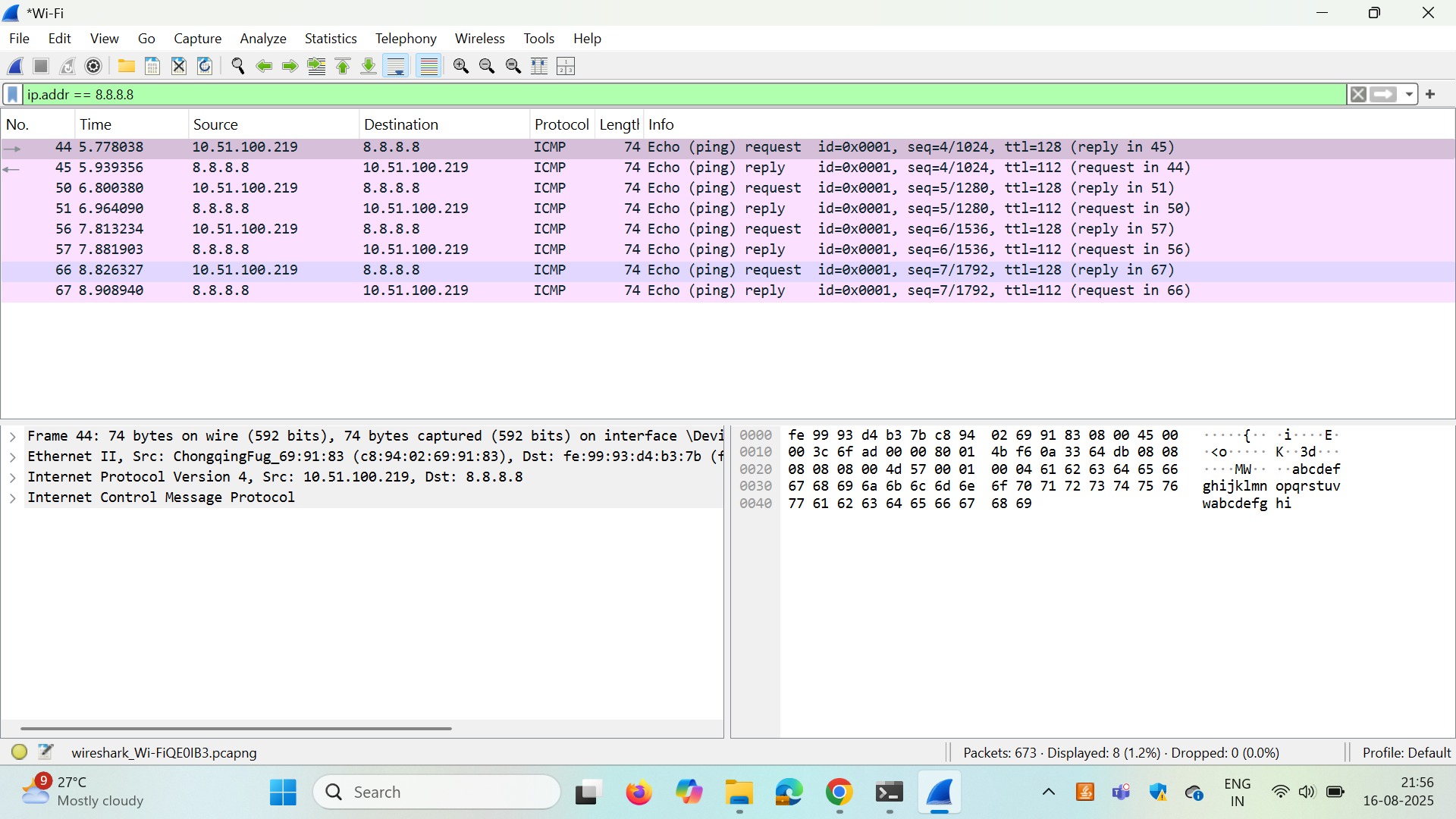
UDP :



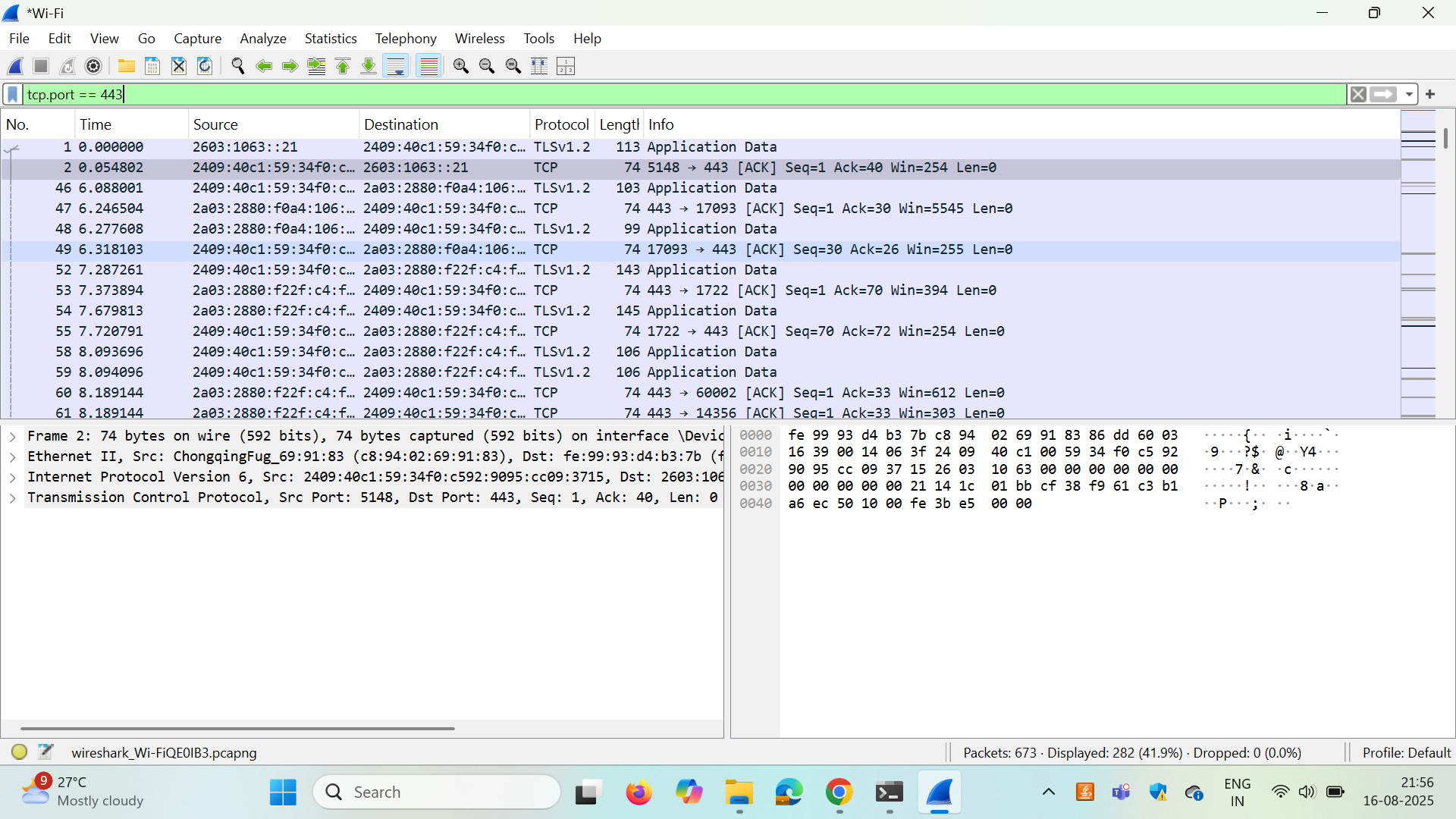
* **HTTP Example:**  
  HEAD request headers, Server response: HTTP/1.1 200 OK.
* **TLS Example:**  
  ClientHello with supported cipher suites and SNI field.



* SPECIFIC IP :



* Specific port:



**🔍 Observations**

* Most traffic on modern websites is encrypted (TLS).
* DNS traffic shows which domains are queried before connection.
* ICMP verified reachability to public IP 8.8.8.8.
* Filters like dns, icmp, tcp.port==443 make analysis simple.

**🧠 Key Learnings**

* Wireshark helps visualize and analyze traffic at packet-level.
* Display filters are powerful for focusing on specific protocols.
* Packets consist of headers and payloads defined by protocols.
* Cleartext protocols (HTTP, DNS) reveal information; TLS hides payloads but still shows metadata.

**✅ Conclusion**

This task demonstrated hands-on use of Wireshark to capture, filter, and analyze network traffic. I identified multiple protocols, observed packet details, and learned how Wireshark assists in both troubleshooting and understanding network communications. The .pcap file serves as proof of capture and can be revisited for deeper analysis.