

This lab session covers the usage of the Wireshark application to monitor and capture the outgoing and incoming packets from a network connection (WIFI, ethernet, etc.). Specifically, students should be able to analyze HTTP, HTTPS, TCP/IP, and UDP protocols using Wireshark, a network protocol analyzer, and draw conclusions.

Pre-lab Preparation:

1. Review the basics and the structure of HTTP, TCP/IP, and UDP protocols,
2. Install Wireshark and ensure it is running on your computer,
3. Create an online, *publically accessible* Git repository to host and upload your work in the labs. We recommend you use GitHub or GitLab.

Lab Activities:

Part 1: Capturing HTTP Traffic.

Task 1: Start Wireshark and capture packets.

Step 1: Open Wireshark.

Step 2: Select the network interface connected to the internet (e.g., Ethernet or Wi-Fi).

Step 3: Click the "Start Capturing Packets" button (the shark fin icon).

Step 4: Open your favorite web browser and navigate to (<http://neverssl.com/>) website.

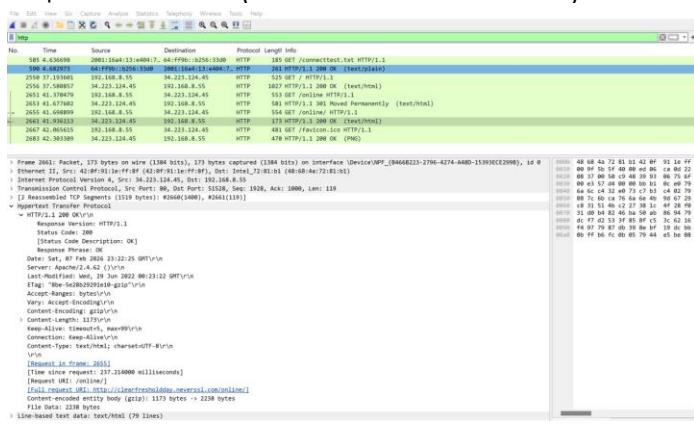
Step 5: After the website has fully loaded, stop capturing packets by clicking the red stop button in Wireshark.

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.

Step 2: Select any HTTP packet to view its details.

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



This HTTP packet shows a response with status code 200 OK, which means the request was successful and the web page loaded correctly.

Part 2: Analyzing TCP/IP Traffic.

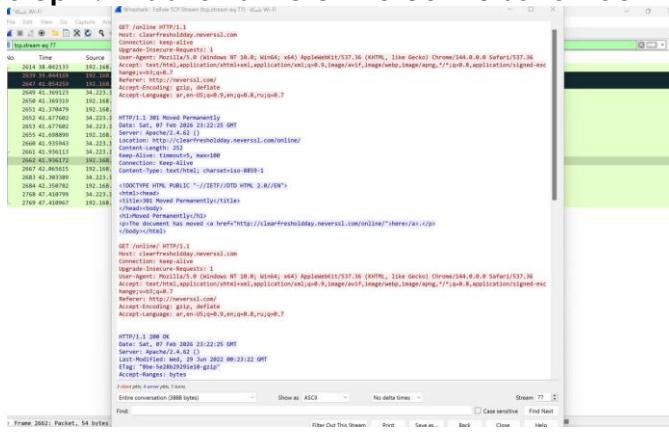
Task 1: Filter TCP packets

Step 1: Clear the previous filter and type TCP to focus on TCP packets.

Step 2: Select a TCP packet related to your HTTP request/response.

Step 3: Right-click on the packet and select "Follow" -> "TCP Stream".

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



This shows the TCP communication between the client and the server, where an HTTP request was sent and the server responded successfully.

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

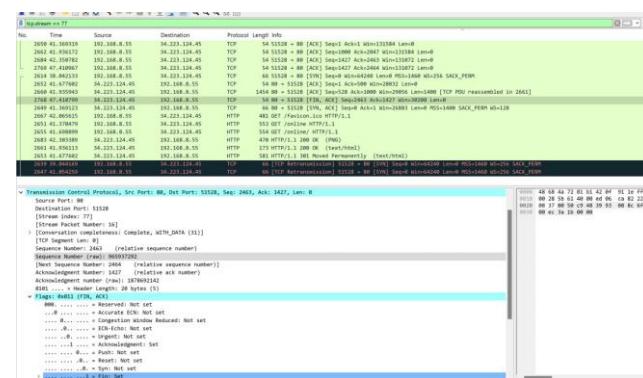
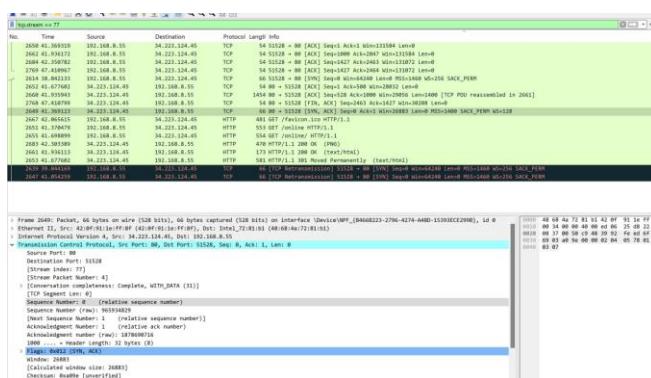
Step 1: Find and select packets related to the TCP three-way handshake:

- 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.

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

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



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).

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.

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

Step 4: Observe the source and destination ports, length, and data.

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

```
Frame 149: Packet, 1292 bytes on wire (10336 bits), 1292 bytes captured (80336 bits) on interface \Device\NPF_{B466B223-2796-4274-A8B0-15393ECE299B}, id 6
Ethernet II, Src: 02:00:00:00:00:00 (02:00:00:00:00:00), Dst: Intel(R) Dual Band Wireless-AC 7218 (08:00:20:72:18:00)
Internet Protocol Version 4, Src: 2001:16a4:13::e084, Dst: 2001:16a4:13::e084
Source Port: 443, Dest Port: 443
[Checksum: 0x8d26] [Unverified]
[Checksum Status: Unverified]
[Stream Index: 7]
[Stream Sequence Number: 24]
[Timestamp]
[Checksum]
UDP payload (1290 bytes)
Data (...): 47ff4098795a4e4eedaa21895587aa86c7fbdcb7e219992a9d98c3c64ffa2d4040ef96c2c9a23c53419a47a8975517aeb387ad8c183ed09cc6216e4ea0af147C
[Length: 1290]
```

UDP headers are simpler than TCP headers because UDP only includes basic fields such as source port destination port length and checksum. It does not use sequence numbers or acknowledgments which makes it faster but less reliable.

TCP headers are more complex because they include sequence numbers acknowledgments and control flags to ensure reliable and ordered 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	Because TCP establishes a connection before sending data and ensures reliable communication between the sender and the receiver.
Data Integrity and Ordering	TCP	Because TCP makes sure the data arrives in the correct order and without loss by using sequence numbers and acknowledgments.

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

	TCP	UDP
Use cases	Used in web browsing, email, and file transfers where reliable data delivery is important.	Used in streaming, online games, and voice/video calls where speed is more important.
Performance	More reliable but slower because it checks delivery and keeps data in order.	Faster with lower delay, but less reliable since it does not guarantee delivery or order.