

CS471 – Web Technologies (Laboratory)		Lab 1
		The Internet Protocols

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.

#### 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.

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

- Step 1:** Find and select packets related to the TCP three-way handshake:
- SYN: Initiates a connection.
  - SYN-ACK: Acknowledges and responds to the SYN.
  - 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.

CS471 – Web Technologies (Laboratory)		Lab 1
		The Internet Protocols

**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.

**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	Uses a 3-way handshake to establish a connection before sending data guarantees delivery with acknowledgments (ACKs).
Data Integrity and Ordering	TCP	Uses sequence numbers to reorder packets at the destination and checksums to ensure data arrives without errors.

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

	TCP	UDP
Use cases	Web browsing (HTTP/HTTPS), Email (SMTP/IMAP), File Transfer (FTP).	Live streaming (VoIP, Video), Online Gaming, DNS queries
Performance	Slower due to overhead from handshakes, error checking, and retransmission mechanisms.	Faster and more efficient (low latency) because it sends packets without checking if they arrived