CS471 – Web Technologies (Laboratory)	Qassim University College of Computer	Lab Week 2
		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 (https://qu.edu.sa) 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.
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	TCP provides reliability through connection establishment (via a three-way handshake) and ensures data is received correctly
Data Integrity and Ordering	TCP	TCP ensures data integrity and correct ordering by using sequence numbers and acknowledgment mechanisms.

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 video or audio streaming - Online gaming - VoIP (Voice over IP)
Performance	Slower compared to UDP due to connection setup, error checking, and retransmissions - Reliable with guaranteed delivery.	Faster due to the lack of connection establishment and error checking mechanisms

The screenshots:

Task2 step2 :

28897	33.612394	2001:16a2:c087:be0b…	2001:16a6:c002:3::2	TCP	86	49959 → 80 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS=256 SACK_PERM
28901	33.685136	2001:16a6:c002:3::2	2001:16a2:c087:be0b…	TCP	86	80 → 49959 [SYN, ACK] Seq=0 Ack=1 Win=64800 Len=0 MSS=1400 SACK_PERM WS=8
28902	33.685341	2001:16a2:c087:be0b…	2001:16a6:c002:3::2	TCP	74	49959 → 80 [ACK] Seq=1 Ack=1 Win=263168 Len=0

Task2 step3 :

244	4.853343	86.60.126.106	192.168.0.130	TLSv1.2	313	Application Data
245	4.853343	86.60.126.106	192.168.0.130	TCP	67	443 → 50388 [ACK] Seq=20665 Ack=6586 Win=10425 Len=13 [TCP segment of a reassembled PDU]
246	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=20678 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
247	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=21958 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
248	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=23238 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
249	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=24518 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
250	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=25798 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
251	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=27078 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
252	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=28358 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
253	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [ACK] Seq=29638 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
254	4.853343	86.60.126.106	192.168.0.130	TCP	1334	443 → 50388 [PSH, ACK] Seq=30918 Ack=6586 Win=10425 Len=1280 [TCP segment of a reassembled PDU]
256	4.853540	192.168.0.130	86.60.126.106	TCP	54	50388 → 443 [ACK] Seq=6586 Ack=32198 Win=64400 Len=0