**Name:** Basant Raj Kadel   
**ID:** 80117739

**Lab -11: Introduction to Packet Analysis using Wireshark**

**1. Objective**

The primary objective of this lab is to gain hands-on experience with Wireshark, a powerful network protocol analyzer. We will learn to capture, filter, and analyze network traffic to understand fundamental protocols, diagnose performance issues, and identify basic security threats.

**2. Selected Exercises & Procedures**

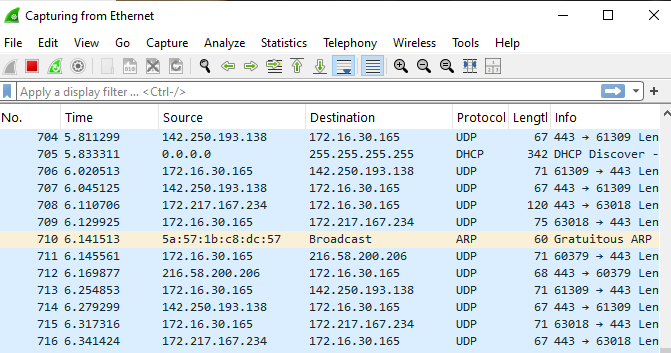
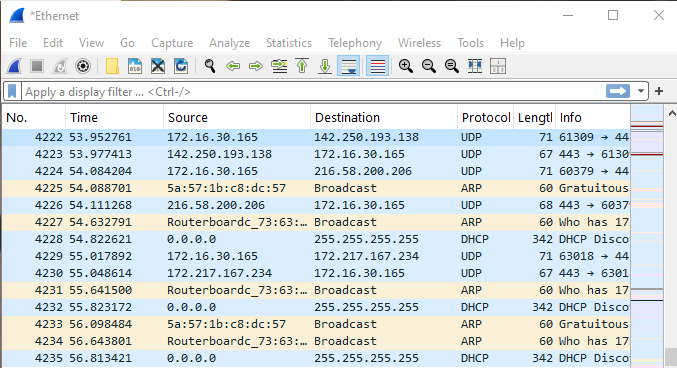
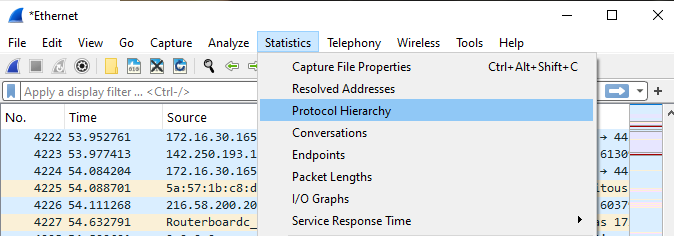
This report documents the following five exercises from the lab manual:

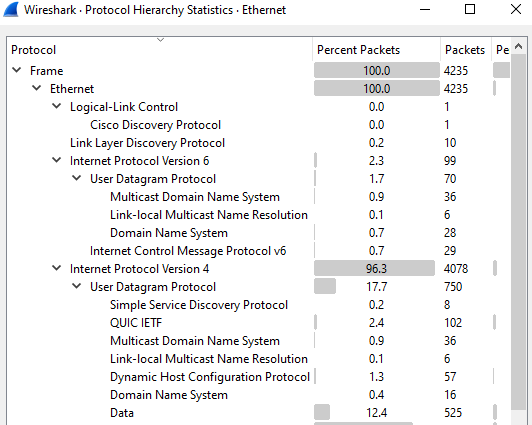
1. **Basic Packet Capture and Analysis**
2. **DNS Query and Response Analysis**
3. **ICMP Ping Request/Reply Analysis**
4. **Identifying Packet Loss (Troubleshooting)**
5. **Analyzing Traceroute and ICMP Traffic**

**3. Step-by-Step Guide & Analysis**

**Procedure 1: Basic Packet Capture and Analysis**

**Objective:** To familiarize with Wireshark's interface and perform a basic capture of local network traffic.

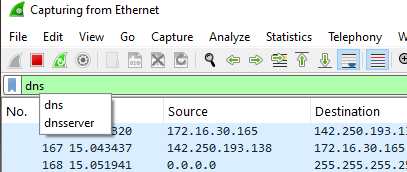
1. **Launch Wireshark** with administrator/root privileges.
2. **Select Interface:** From the main screen, choose your active network interface (e.g., Wi-Fi, Ethernet) from the list.
3. **Start Capture:** Click the blue shark fin (⋮⋮) button to begin capturing all traffic on the selected interface.
4. **Generate Traffic:** While the capture is running, perform normal web browsing (e.g., visit http://example.com).
5. **Stop Capture:** After 15-20 seconds, click the red square (■) button to stop the capture.
6. **Analyze:**
   * Observe the **Packet List** pane (top section) showing all captured frames/packets.  
      
   * Click on a packet to view its details in the **Packet Details** pane (middle section). Expand sections like Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol to see addressing and protocol information.  
     
   * Navigate to **Statistics > Protocol Hierarchy**. This shows a breakdown of all captured protocols by percentage. Note the most common protocols (e.g., TCP, UDP, TLS, HTTP).  
     

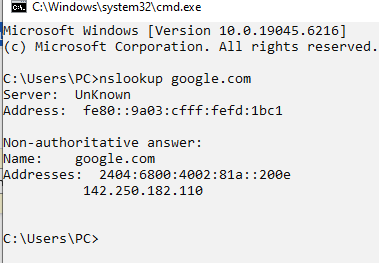


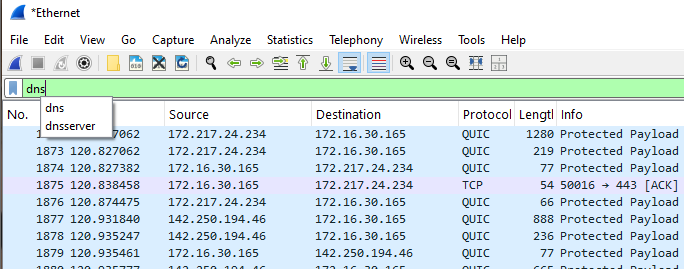
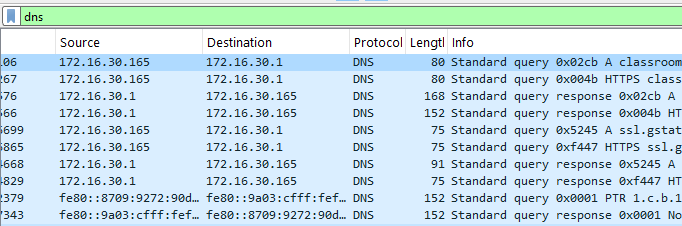
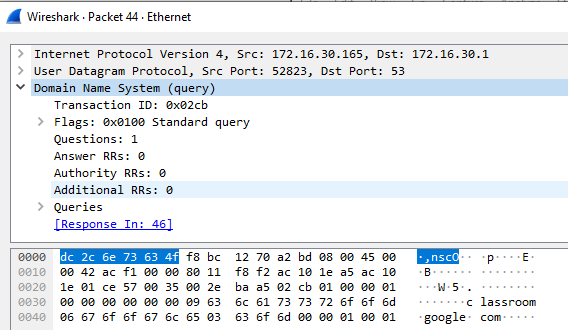
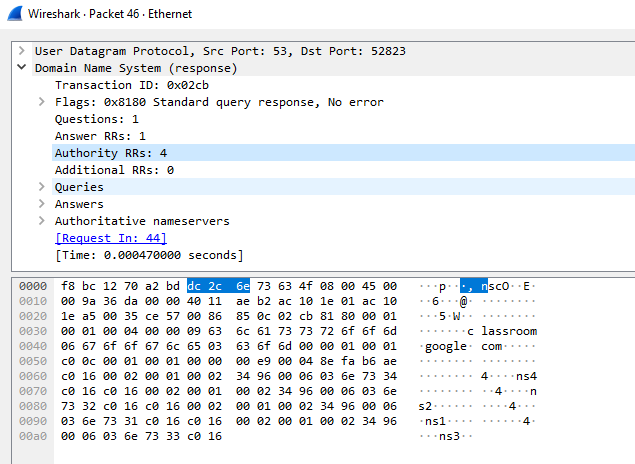
**Conclusion:** This exercise provided a foundational understanding of the packet capture process and Wireshark's core interface components.

**Procedure 2: DNS Query and Response Analysis**

**Objective:** To observe the process of a Domain Name System (DNS) query resolving a domain name to an IP address.

1. **Start a new capture** in Wireshark.
2. **Apply a Display Filter:** Before generating traffic, type dns in the filter bar and press Enter. This will ensure only DNS packets are displayed once captured.  
   
3. **Generate a DNS Query:** Open a command prompt or terminal and run the command: nslookup google.com

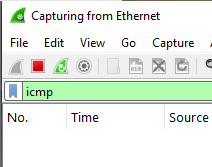
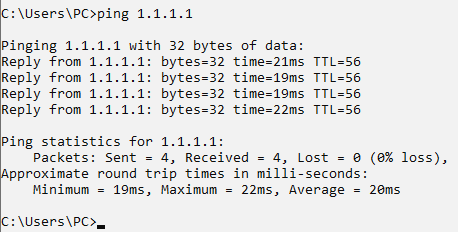
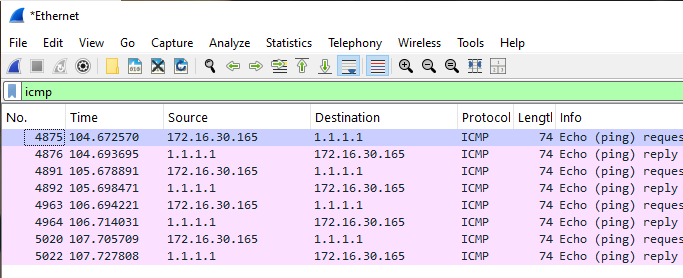
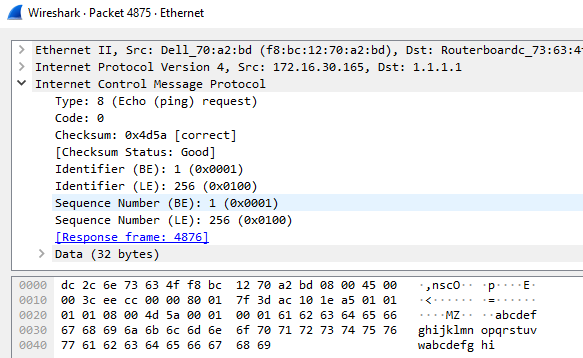
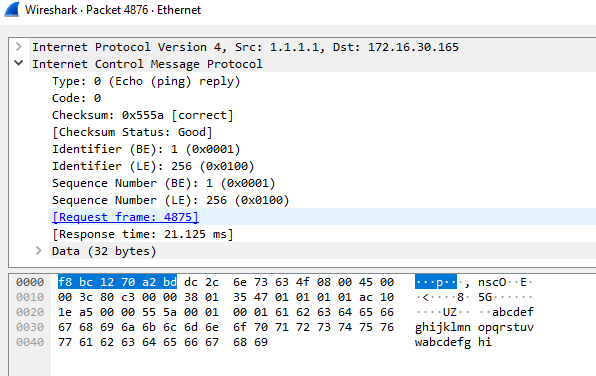


1. **Stop the capture** after the command returns a result.  
   
2. **Analyze:**
   * You should see at least two packets: a **DNS Query** and a **DNS Response**.  
     
   * Select the query packet. In the Packet Details pane, expand the Domain Name System (query) section. Note the Transaction ID and the Queries section showing the name google.com.  
     
   * Select the response packet. Expand the Domain Name System (response) section. Find the Answers section, which contains the A record showing the IP address assigned to example.com. Match the Transaction ID to the original query.  
     

**Conclusion:** This analysis clearly demonstrates how DNS works at the packet level, translating human-readable domain names into machine-routable IP addresses.

**Procedure 3: ICMP Ping Request/Reply Analysis**

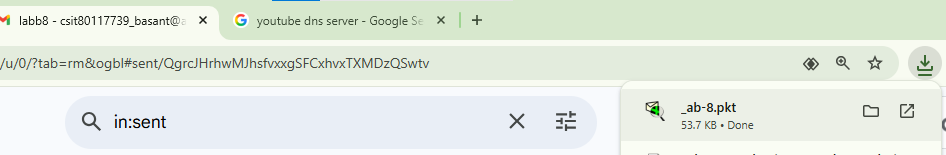
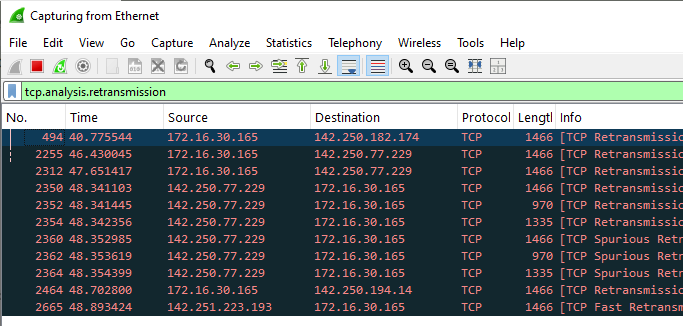
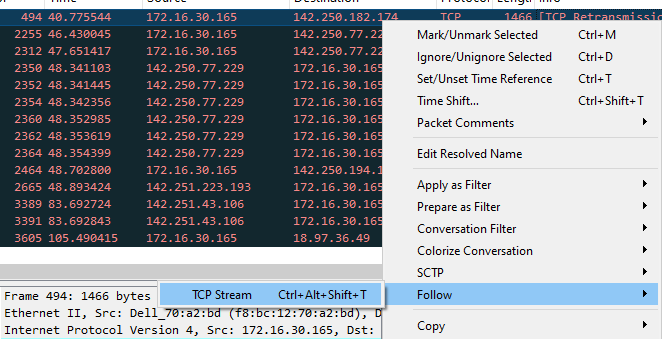
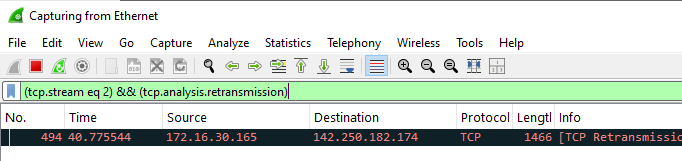
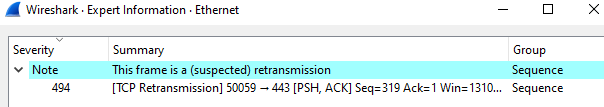
**Objective:** To analyze the structure of Internet Control Message Protocol (ICMP) packets used by the ping utility.

1. **Start a new capture.**
2. **Apply a Display Filter:** Type icmp in the filter bar and press Enter.  
   
3. **Generate ICMP Traffic:** In your command prompt or terminal, run: ping 1.1.1.1 (Cloudflare's DNS server). Let it run for 3-4 pings.  
   
4. **Stop the capture.**
5. **Analyze:**
   * You will see a series of **Echo (ping) request** and **Echo (ping) reply** packets.  
     
   * Select an Echo (ping) request packet. In the Packet Details pane, expand the Internet Control Message Protocol section. Note the Type: 8 (Echo (ping) request) and the Identifier and Sequence number.  
     
   * Find the corresponding Echo (ping) reply packet (Type: 0). Observe that the Identifier and Sequence number match the request, confirming the pairing.  
     

**Conclusion:** This procedure illustrated the simple request-reply mechanism of ICMP Echo messages, which is fundamental for network connectivity testing.

**Procedure 4: Identifying Packet Loss (Troubleshooting)**

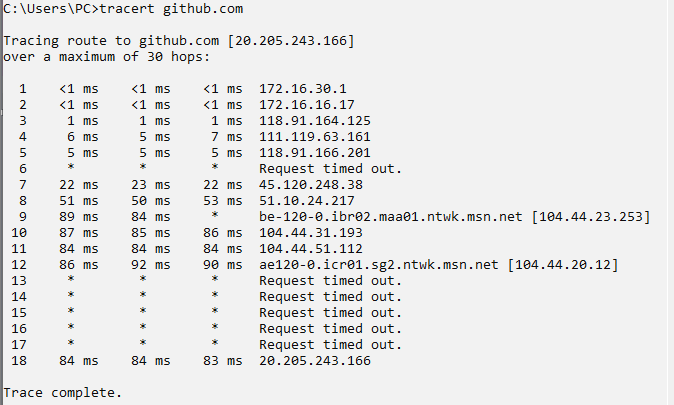
**Objective:** To use Wireshark's analysis features to identify symptoms of network packet loss.

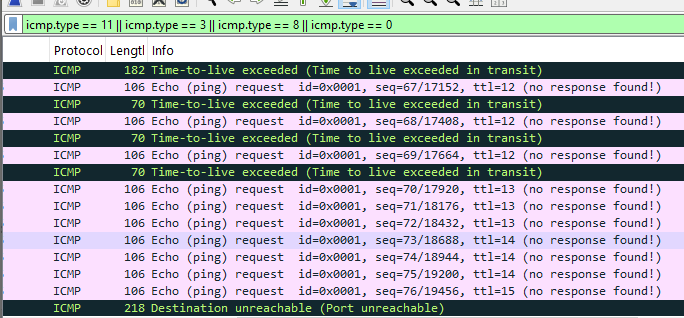
1. **Start a new capture.**
2. **Generate Traffic with Potential Loss:** While capturing, perform an action that might be unreliable (e.g., a large file download over a weak Wi-Fi connection).  
   
3. **Stop the capture** after a minute.
4. **Analyze for Retransmissions:**
   * In the filter bar, apply the filter: tcp.analysis.retransmission.  
     
   * If any packets appear, they are TCP segments that Wireshark has identified as being retransmitted because the sender did not receive an acknowledgment (ACK).
   * **Right-click** on a retransmitted packet and select **Follow > TCP Stream**. This isolates the conversation. Close the stream window.  
     
   * Notice the filter has changed to something like tcp.stream eq 5. Modify this filter to also show retransmissions for this stream: (tcp.stream eq 5) && (tcp.analysis.retransmission).  
     
5. **View Expert Information:** Navigate to **Analyze > Expert Information**. Note any warnings about retransmissions, which are indicators of packet loss.  
   

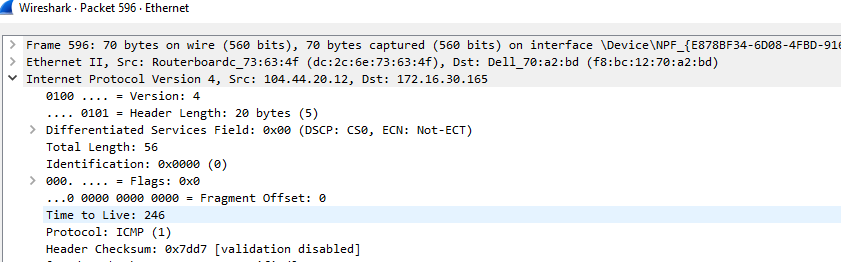
**Conclusion:** A high number of TCP retransmissions is a strong indicator of packet loss, which can be caused by congestion, faulty hardware, or poor wireless signals.

**Procedure 5: Analyzing Traceroute and ICMP Traffic**

**Objective:** To capture and analyze the ICMP packets generated by the tracert/traceroute command.

1. **Start a new capture.**
2. **Apply a Filter:** Use a comprehensive ICMP filter: icmp.type == 11 || icmp.type == 3 || icmp.type == 8 || icmp.type == 0. This filters for Time Exceeded, Destination Unreachable, and Echo Request/Reply messages.
3. **Run Traceroute:** In your command prompt or terminal, run:
   * **Windows:** tracert github.com  
     
   * **Linux/macOS:** traceroute -I example.com (The -I flag uses ICMP Echo requests like Windows).
4. **Stop the capture** once the command completes.
5. **Analyze:**
   * Observe the sequence of packets. For each hop (router) in the path to the destination, we should see:
     1. An **Echo Request** (Type 8) sent from your machine with an increasing TTL.
     2. A **Time Exceeded** (Type 11) response from a router when the TTL expires.



* + The final packet should be an **Echo Reply** (Type 0) from the destination host or a **Destination Unreachable** (Type 3) message.
  + In the Packet Details pane, expand the IP header for a Time Exceeded packet to see the Time to live: 1 field, which caused the router to generate the response.  
    

**Conclusion:** This analysis deconstructs how traceroute maps the network path by manipulating the TTL field in IP headers and interpreting the ICMP error messages returned by routers.

**4. Overall Conclusion**

This lab successfully provided a practical introduction to network packet analysis using Wireshark. Through these five procedures, we learned to capture live traffic, apply precise filters to isolate specific protocols, and analyze the contents of packets for common protocols like DNS, ICMP, and TCP. Furthermore, we applied this knowledge to basic network troubleshooting by identifying packet loss and understanding the mechanics of path discovery with traceroute. These skills form a critical foundation for advanced network security, performance monitoring, and diagnostic tasks.