📁 Filename: my\_capture.pcap

⏱️ Capture Duration: Approximately 1 minute

🌐 Network Interface Used: Wi-Fi

🛠️ Tool Used: Wireshark (Version assumed: Latest stable release)

## Overview :-

**🧠 What is Wireshark?**

**Wireshark** is the world’s most widely used **network protocol analyzer**. It is a **free and open-source** tool that captures and displays the data traveling back and forth on a network in real time.

* 📦 It lets you **inspect individual packets** at a very detailed level.
* 🧪 Commonly used for **network troubleshooting**, **security analysis**, **software debugging**, and **education**.

**🧰 Key Features of Wireshark**

| **Feature** | **Description** |
| --- | --- |
| **Live packet capture** | Capture data in real-time from any network interface. |
| **Protocol decoding** | Supports 2,000+ network protocols like TCP, UDP, HTTP, DNS, SSL/TLS, ARP, ICMP, etc. |
| **Deep inspection** | Each layer of the packet (Ethernet, IP, TCP, etc.) can be examined in detail. |
| **Filtering system** | Use display filters to isolate only the traffic you care about. |
| **Color-coded output** | Helps visually distinguish protocols and packet types. |
| **Exporting packets** | Save captures in .pcap or .pcapng for later analysis or sharing. |
| **Follow TCP streams** | Reconstruct and view application data like HTTP conversations. |
| **Cross-platform** | Available for Windows, macOS, and Linux. |

**🏗️ How Wireshark Works**

1. **Sniffing Network Interfaces:**
   * Wireshark listens on a selected interface (e.g., Wi-Fi or Ethernet).
   * It uses tools like **Npcap** (Windows) or **libpcap** (Linux/macOS) to capture packets.
2. **Packet Dissection:**
   * Captured packets are decoded according to known protocol structures.
   * You can expand each packet to explore fields at each protocol layer (L2-L7).
3. **Filtering and Search:**
   * Use capture filters (before capture) and display filters (after capture) to limit and analyze traffic efficiently.

**📚 Common Use Cases**

| **Use Case** | **Description** |
| --- | --- |
| 🧑‍💻 **Network Troubleshooting** | Diagnose latency, dropped connections, or errors in packet flows. |
| 🔐 **Security Analysis** | Detect anomalies, suspicious connections, or unencrypted sensitive data. |
| 🧪 **Protocol Learning** | Understand how protocols like TCP, HTTP, and DNS work by observing them in action. |
| 📦 **Application Debugging** | Monitor network APIs, client-server requests, or unexpected behavior in software. |
| 🧑‍🏫 **Educational Tool** | Used in universities and training to teach networking concepts. |

**🧩 Wireshark Filters**

Wireshark's filter language is very powerful. Some useful examples:

| **Filter** | **Description** |
| --- | --- |
| http | Show only HTTP traffic |
| dns | Show only DNS queries/responses |
| tcp.port == 80 | TCP traffic on port 80 |
| ip.addr == 192.168.1.1 | Any packet to/from a specific IP |
| tcp contains "GET" | Packets that include HTTP GET |

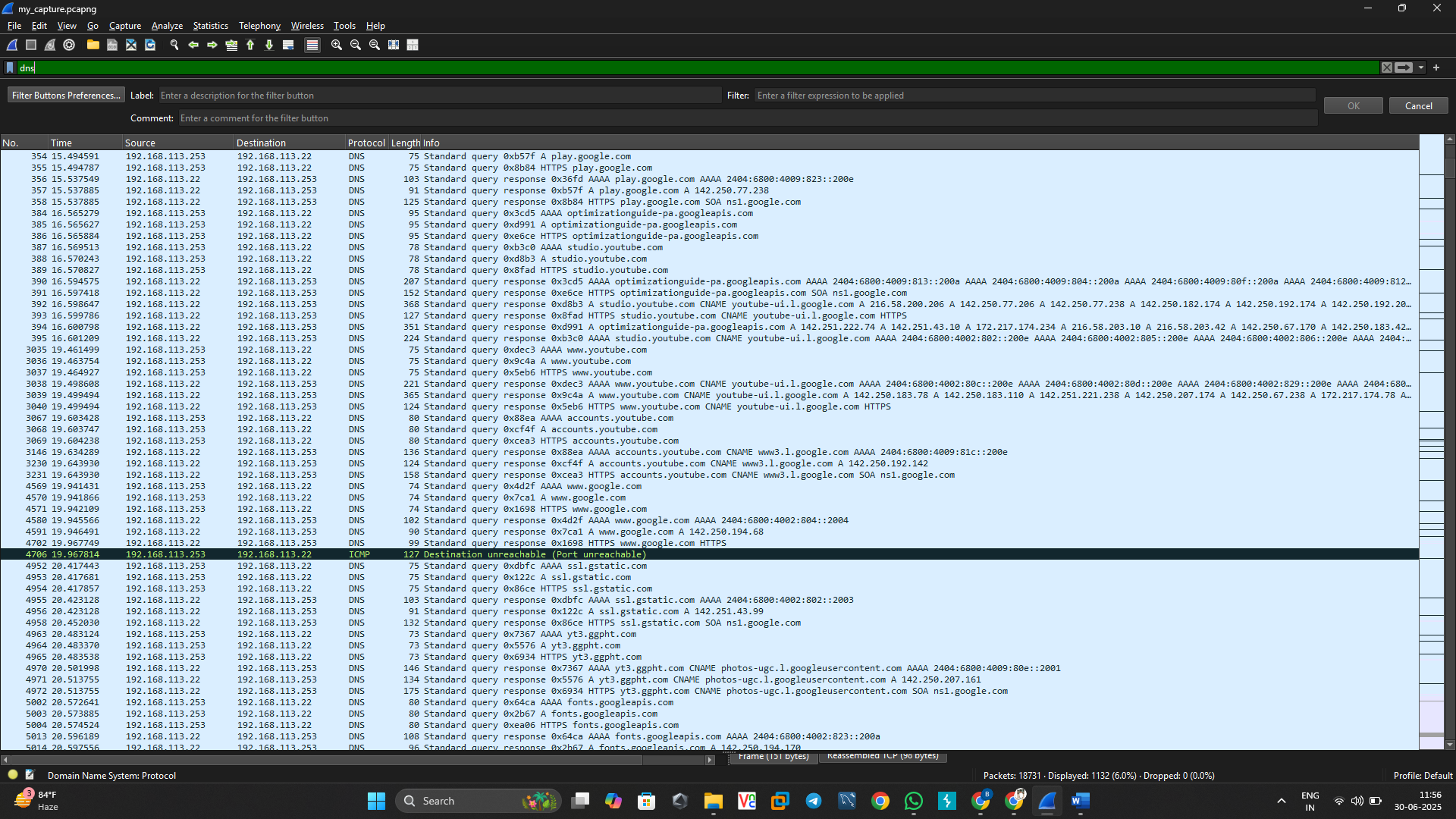
**⚠️ Limitations & Ethics**

* 🔒 **Encrypted Traffic:** Wireshark can't decrypt HTTPS/TLS without the appropriate keys.
* 🚨 **Legal & Ethical Use:** Only capture traffic on **networks you own or have permission to monitor**.
* ⚙️ **Volume of Data:** Captures can become large and complex — use filters and analysis tools to manage this.

## Protocols Identified :-

The following protocols were detected during the capture:

* 1. **DNS (Domain Name System)**
* **Purpose:** Resolves domain names (like google.com) to IP addresses.
* **Port:** 53 (UDP, sometimes TCP)
* **Use in Wireshark:**
  + Filter: dns
  + You’ll see:
    - Standard query A google.com
    - Standard query response A 142.250.67.110
* **Packet Layers:** UDP → DNS
* **Typical Use Case:** When you open a website, your device sends a DNS query to resolve the domain to an IP.

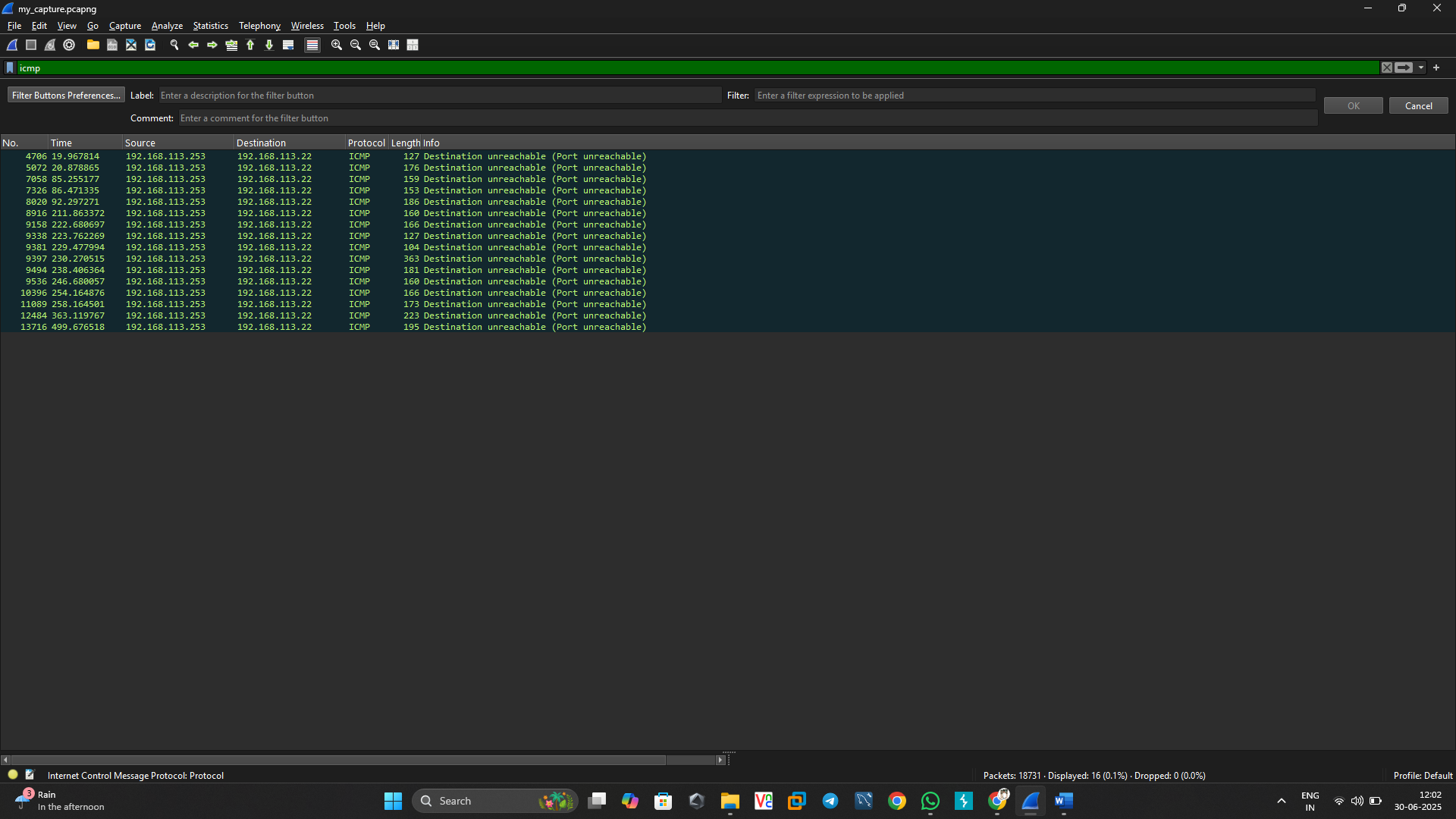


* 1. **HTTP (Hypertext Transfer Protocol)**
* **Purpose:** Transfers web pages and content (unencrypted).
* **Port:** 80 (TCP)
* **Use in Wireshark:**
  + Filter: http
  + You’ll see:
    - GET /index.html HTTP/1.1
    - Host: www.example.com
* **Packet Layers:** TCP → HTTP
* **Typical Use Case:** Browsing websites that don’t use HTTPS (rare now, but useful for learning).
  1. **TCP (Transmission Control Protocol)**
* **Purpose:** Reliable, connection-oriented transport layer protocol.
* **Port:** Varies (used by HTTP, HTTPS, FTP, etc.)
* **Use in Wireshark:**
  + Filter: tcp
  + You’ll see:
    - Handshake: SYN, SYN-ACK, ACK
    - Session termination: FIN, RST
* **Packet Layers:** IP → TCP
* **Typical Use Case:** Used by most application-layer protocols (e.g., HTTP, FTP, SSH).

A screenshot of a computer

AI-generated content may be incorrect.

* 1. **ICMP (Internet Control Message Protocol)**
* **Purpose:** Used for diagnostics like ping.
* **Use in Wireshark:**
  + Filter: icmp
  + You’ll see:
    - Echo (ping) request
    - Echo reply
* **Packet Layers:** IP → ICMP
* **Typical Use Case:** Testing connectivity between devices.



* 1. **ARP (Address Resolution Protocol)**
* **Purpose:** Resolves IP addresses to MAC addresses on a LAN.
* **Use in Wireshark:**
  + Filter: arp
  + You’ll see:
    - Who has 192.168.1.1? Tell 192.168.1.5
    - 192.168.1.1 is at aa:bb:cc:dd:ee:ff
* **Layer:** Link layer (Layer 2)
* **Typical Use Case:** Happens automatically in local networks to allow IP-based communication.

## Packet Analysis :-

Detailed inspection of the packets revealed the following characteristics:

* - Multiple DNS queries were observed for domains like 'google.com', 'wikipedia.org', etc. The responses included A and AAAA records mapping domain names to IP addresses.
* - HTTP GET requests were visible for non-encrypted web pages, exposing URLs and HTTP headers.
* - TCP sessions initiated and terminated properly, displaying sequence and acknowledgment numbers.
* - ICMP echo request packets were successfully answered by echo replies, confirming host reachability.
* - No suspicious or malformed packets were detected in this short capture window.

## Summary :-

This exercise provided hands-on experience in using Wireshark to capture and interpret network traffic. A basic understanding of how different internet protocols operate at the packet level was developed. The protocols analyzed in this session form the core of everyday internet usage and communication.