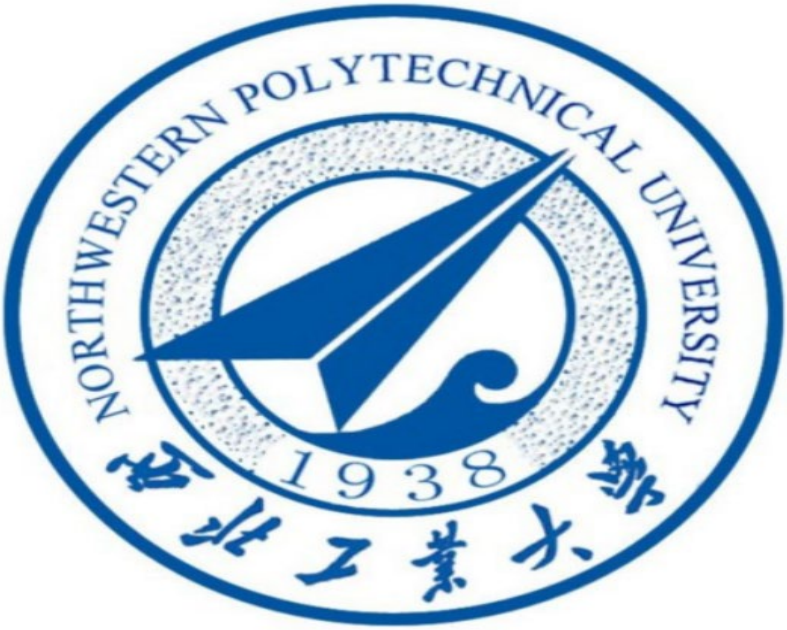


Computer Networks

Project

Student Name : ABID ALI
Student ID : 2019380141
Student Email : abiduu354@gmail.com / **3616990795@qq.com**
Lecture : Professor Bo Yang
Submission : 10/25/22
Submitted Email : liu1218446147@mail.nwpu.edu.cn

Contents

Project Topics

Choose one of them (included but NOT limited!)

- Analyzing HTTP/TCP (handshake) /UDP/... using Wireshark/...
- Realizing the TCP/UDP/... communication example via Matlab/python/C, C++...
- Analyzing MAC/... protocol using Matlab/NS/...
- Other experiment that is related to our course

Chosen Topic

Analyzing HTTP/TCP (handshake) /UDP/... using Wireshark/...

Introduction

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.

You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course).

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

Here are some reasons people use Wireshark:

- Network administrators use it to *troubleshoot network problems*
- Network security engineers use it to *examine security problems*
- QA engineers use it to *verify network applications*
- Developers use it to *debug protocol implementations*
- People use it to *learn network protocol* internals

Here are some things Wireshark does not provide:

- Wireshark isn't an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isn't allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- Wireshark will not manipulate things on the network, it will only "measure" things from it. Wireshark doesn't send packets on the network or do other active things (except domain name resolution, but that can be disabled).

Tools

1) WireShark

Goal

The project's goal is to effectively analyze a network by detecting HTTP, TCP, and UDP packets and collecting the data contained inside them.

My research seeks to analyze networks by looking up various IP addresses and the information they are related with. In order to efficiently search for IP addresses registered on a network and then seek for HTTP, TCP, and UDP data linked with those IPs, a system must be developed. Ip headers can be used to identify an IP packet. Header data for TCP This IP datagram packet contains both TCP and UDP packet data. The information transmitted by the DNS, HTTP, FTP, SMTP, SIP, and other protocols is included in this layer.

We will do those following steps:

1. Sniffing technology for Lan networks that works.
2. Ip packet tracing and continuous monitoring.
3. Monitoring of IP headers, sources, destinations, and other data.
4. Monitoring of TCP header, port, and related data.
5. An efficient GUI for ongoing network parameter monitoring.

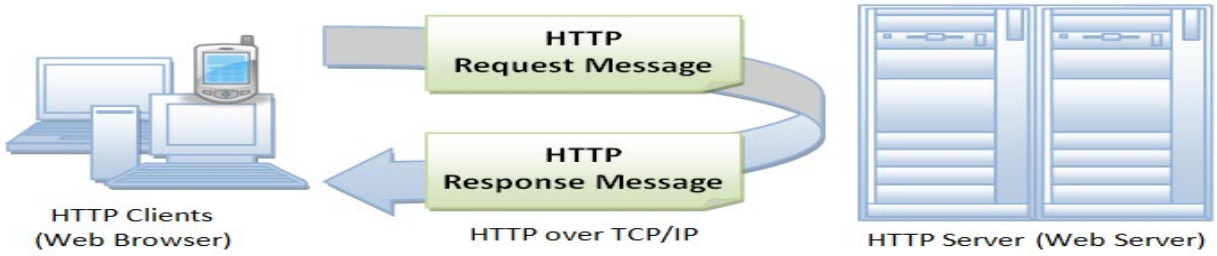
Implementation:

HTTP Analysis using Wireshark

HyperText Transfer Protocol (HTTP)

HTTP (Hypertext Transfer Protocol) is perhaps the most popular application protocol used in the Internet (or The WEB).

- HTTP is an *asymmetric request-response client-server* protocol as illustrated. An HTTP client sends a request message to an HTTP server. The server, in turn, returns a response message. In other words, HTTP is a *pull protocol*, the client *pulls* information from the server (instead of server *pushes* information down to the client).



- HTTP is a stateless protocol. In other words, the current request does not know what has been done in the previous requests.
- HTTP permits negotiating of data type and representation, so as to allow systems to be built independently of the data being transferred.
- Quoting from the RFC2616: "The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers."

Step1-Opening the Browser

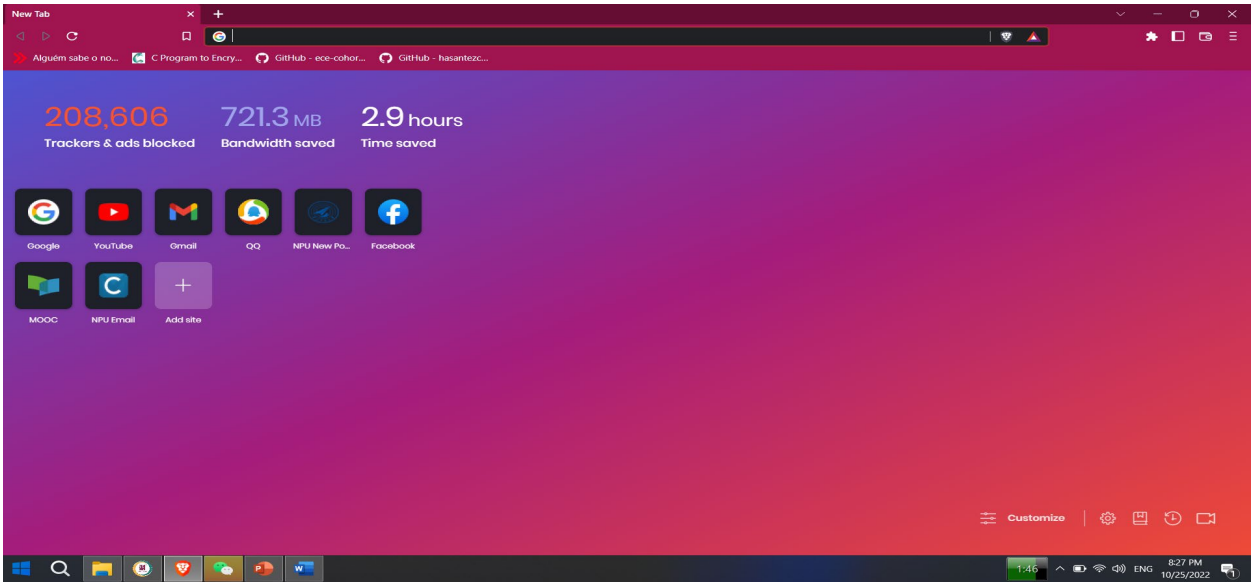


Fig: Opening The Browser

Step2 - Open Wireshark Application

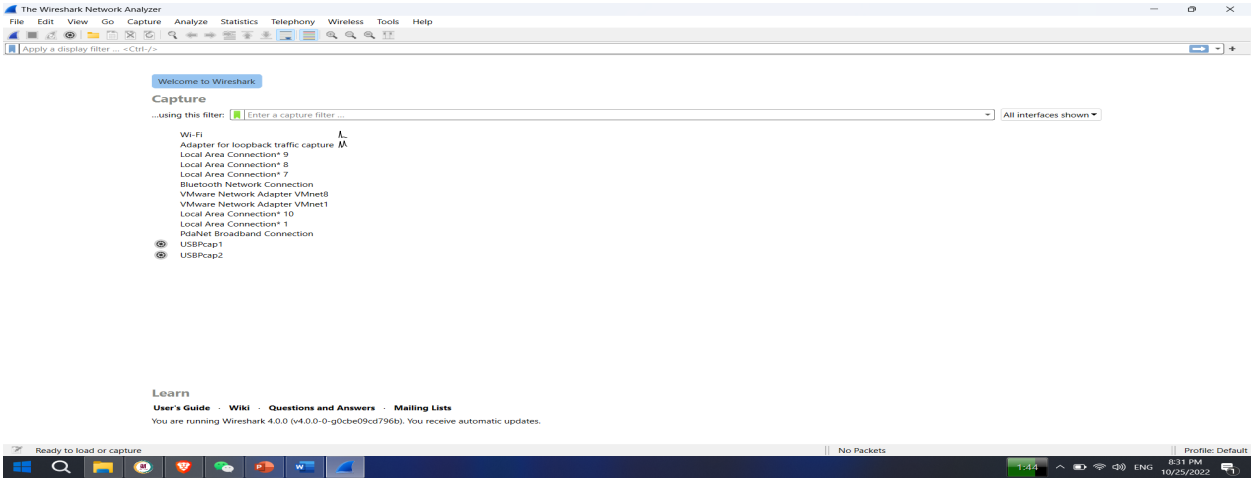


Fig: Open Wireshark Application

Step3 – Starting the WireShark Application & Waiting for few minutes

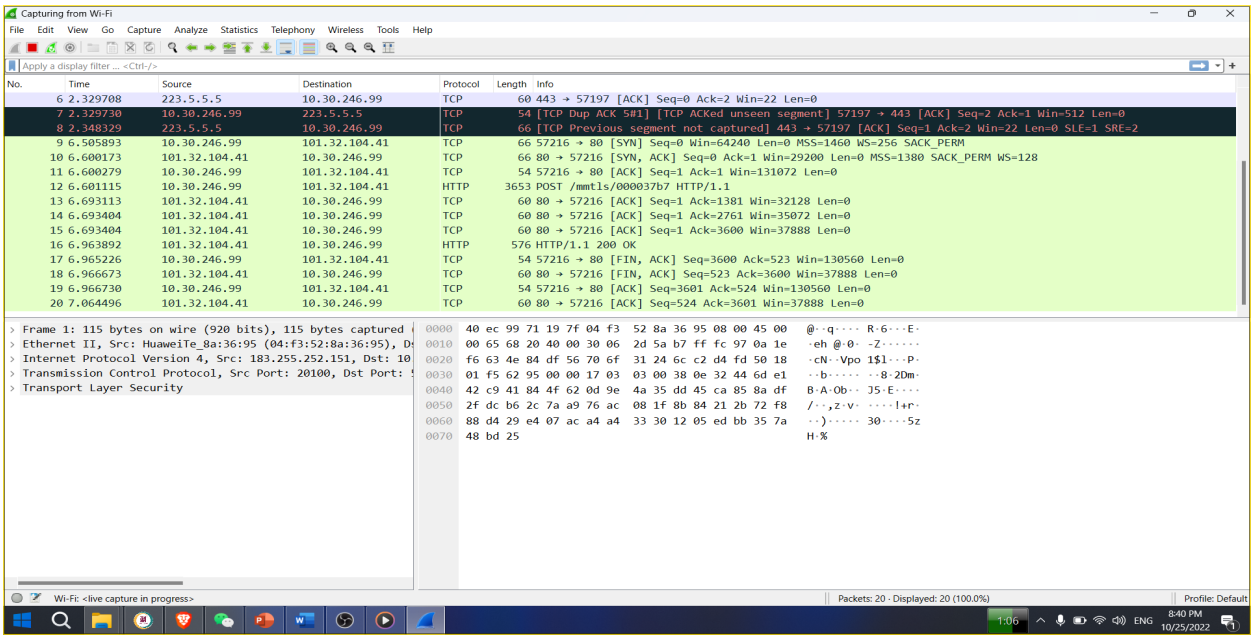


Fig: Started WireShark

Step4 – Typing the link in web-address bar

http://studyat.nwpu.edu.cn/Degree_Program.htm

http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html

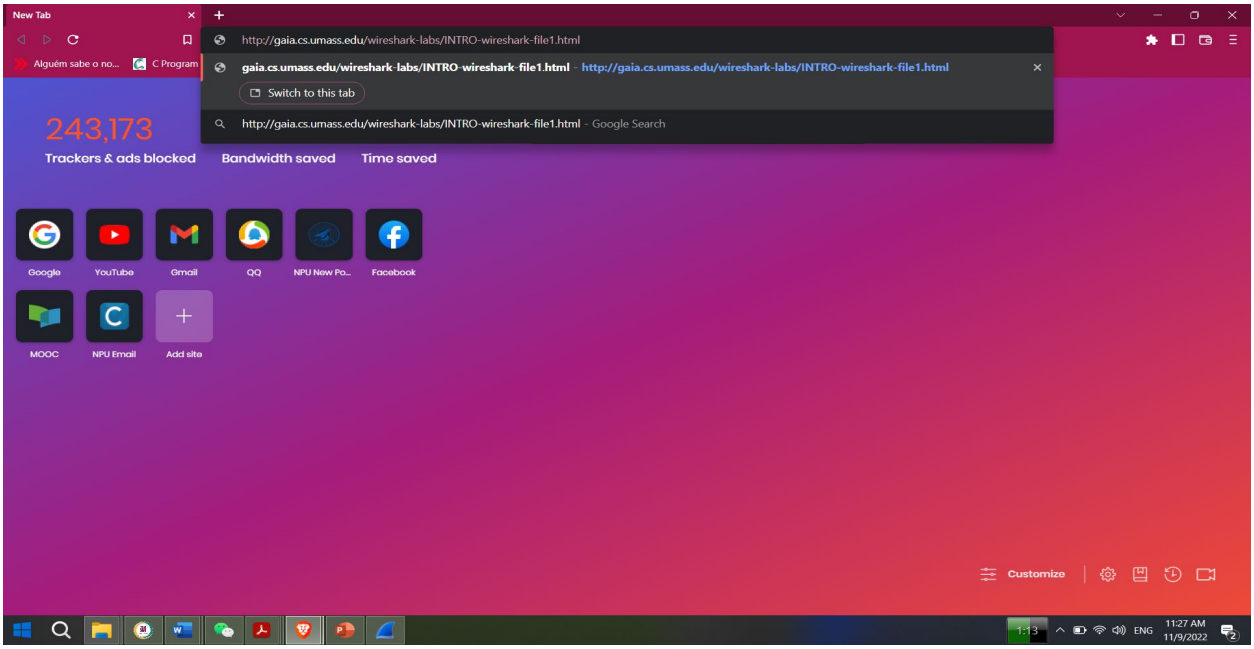


Fig: Pasting the link

Step5 – Observing the difference and Analyzing

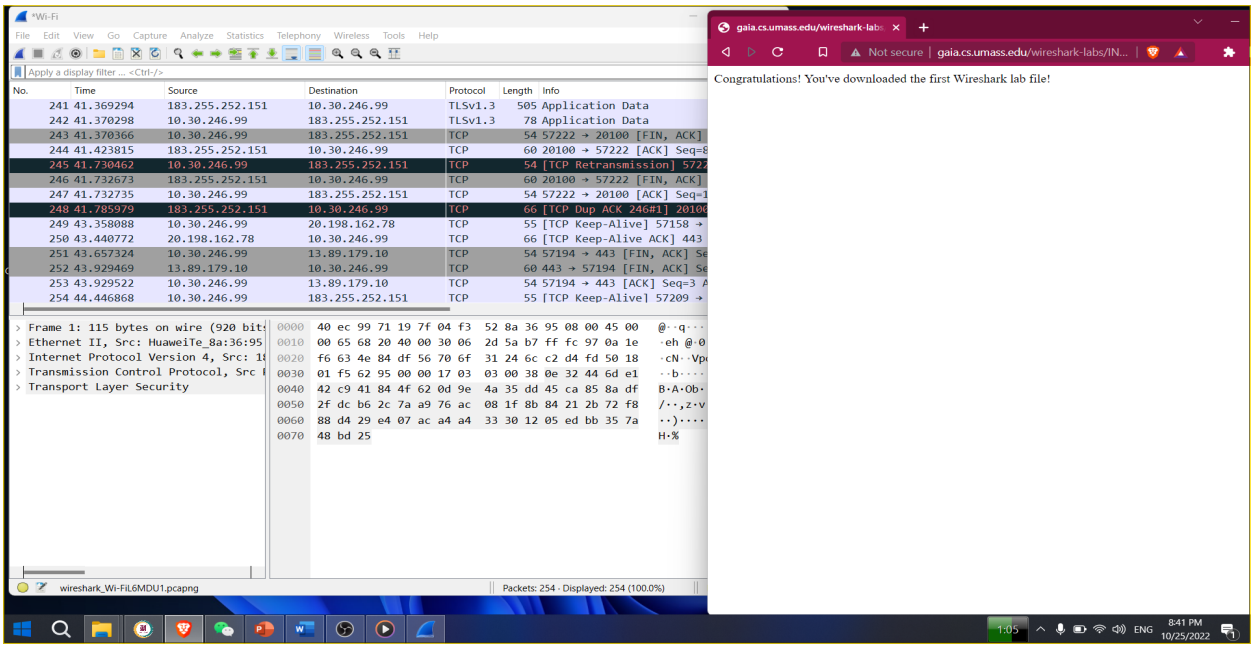


Fig: Congratulation Message Got

Step6 – Http Filtering

Enter and use the "http" filter expression to focus on HTTP traffic. The HTTP requests and answers will be shown by this filter, but not the individual packets that are involved. should see a screen similar to this.

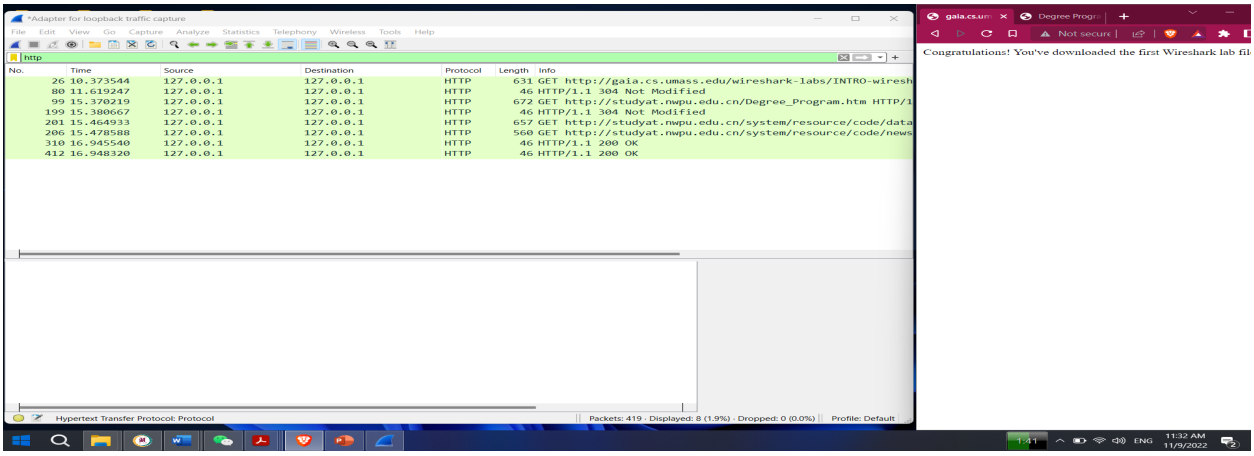
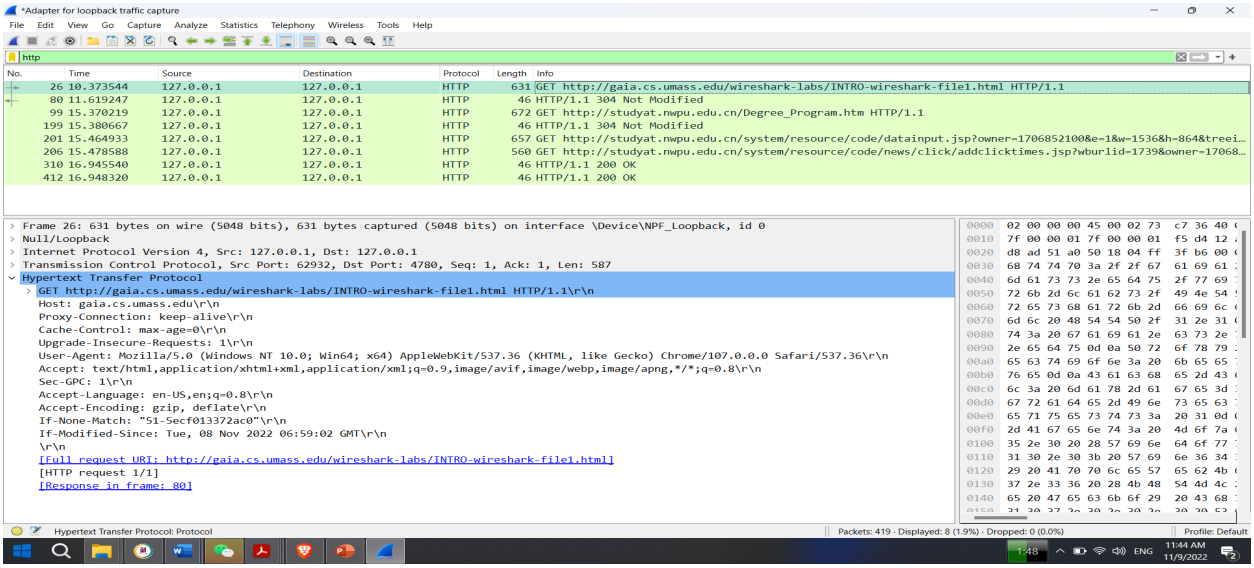


Fig : Got the HTTP/1.1 200 OK

If the transfer is regular and successful, "200 OK" will be shown in the packet's info. As you can see, the response and the request are identical, and the "200 OK" status code is followed by a number of headers. But alternative headers will be employed, and the requested content will come after the headers. Look at the typical headers, like:

- Server. The kind of server and its capabilities.
- Date, Last-Modified. The time of the response and the time the content last changed.
- Cache-Control, Expires, Etag. Information about how the response can be cached.

Step6 – Expanding HTTP block



We need to remember that because HTTP is an application protocol that uses TCP/IP for transport, it comes after the TCP and IP headers. Select the packet, then look for the HTTP block in the center panel to examine it. a block that is enlarged in Investigate the headers that are included in the request, as shown in the preceding figure. At the beginning of the request, you will first see the GET method along with information like the path. Then, a list of headers in the form of tagged parameters will appear. There might be a lot of headers, and each browser has a different selection of headers and values. Common headers:

- Host. It specifies the name (and port) of the server and is a required header.
- User-Agent. the type of browser used and its features.
- Accept, Accept-Charset, Accept-Language, and Accept-Encoding. a description of the response's acceptable formats, such as text/html, together with information on the encoding (for example, gzip) and language.
- The name and value of cookies that the browser has stored for a particular website.
- Cache-Control. information on how to cache the answer.

Simple text and line-based formatting are used to send the request information. A large portion of the request may be read right from the packet if you look at the bottom panel.

TCP (handshake) Analysis using Wireshark

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP, which is part of the Transport Layer of the TCP/IP suite. SSL/TLS often runs on top of TCP.

TCP is connection-oriented, and a connection between client and server that is established before data can be sent. The server must be listening (passive open) for connection requests from clients before a connection is established. Three-way handshake (active open), retransmission, and error detection adds to reliability but lengthens latency. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP) instead, which provides a connectionless datagram service that prioritizes time over reliability. TCP employs network congestion avoidance. However, there are vulnerabilities in TCP, including denial of service, connection hijacking, TCP veto, and reset attack

TCP 3-way handshake via Wireshark

Three-Way HandShake or a TCP 3-way handshake is a process which is used in a TCP/IP network to make a connection between the server and client. It is a three-step process that requires both the client and server to exchange synchronization and acknowledgment packets before the real data communication process starts.

TCP message types

Message	Description
Syn	Used to initiate and establish a connection. It also helps you to synchronize sequence numbers between devices.
ACK	Helps to confirm to the other side that it has received the SYN.
SYN-ACK	SYN message from local device and ACK of the earlier packet.
FIN	Used to terminate a connection.

TCP Three-Way Handshake Process

TCP traffic begins with a three-way handshake. In this TCP handshake process, a client needs to initiate the conversation by requesting a communication session with the Server:

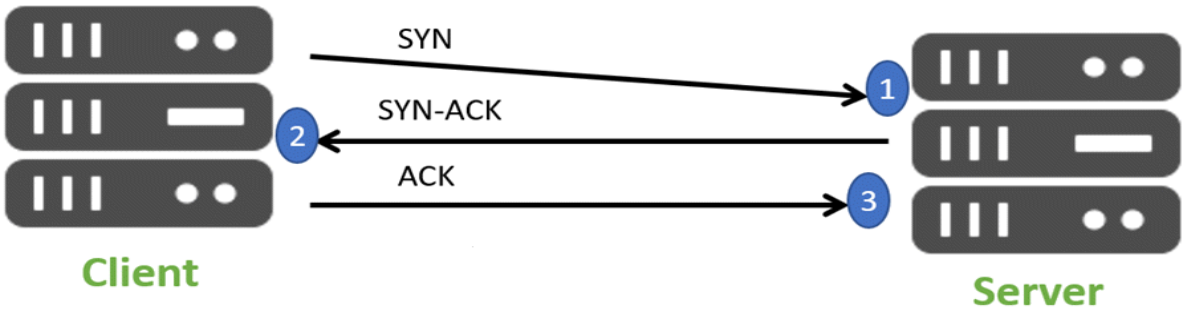


Figure: 3 way Handshake Diagram

- **Step 1:** In the first step, the client establishes a connection with a server. It sends a segment with SYN and informs the server about the client should start communication, and with what should be its sequence number.

- **Step 2:** In this step **s**erver responds to the client request with SYN-ACK signal set. ACK helps you to signify the response of segment that is received and SYN signifies what sequence number it should able to start with the segments.

- **Step 3:** In this final step, the client acknowledges the response of the Server, and they both create a stable connection will begin the actual data transfer process.

Process of capturing

Step 1: Start browser and wait few minutes.

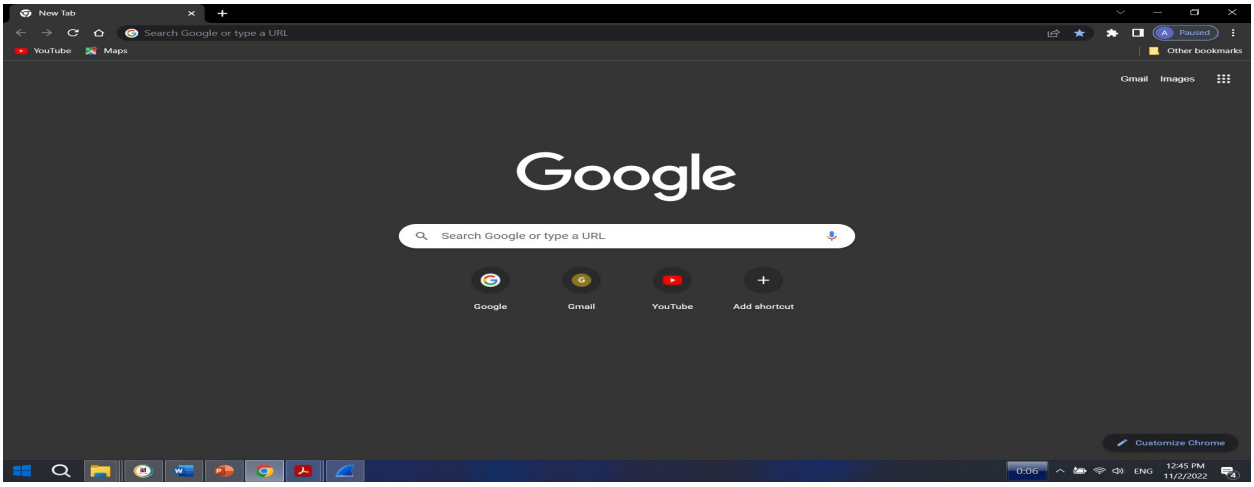


Figure: Opening The Browser

Step 2: Start Wireshark and wait wait few minutes.

Double-click the Wireshark icon 

, which is located on the desktop.

Step 3: Press wifi capturing packets

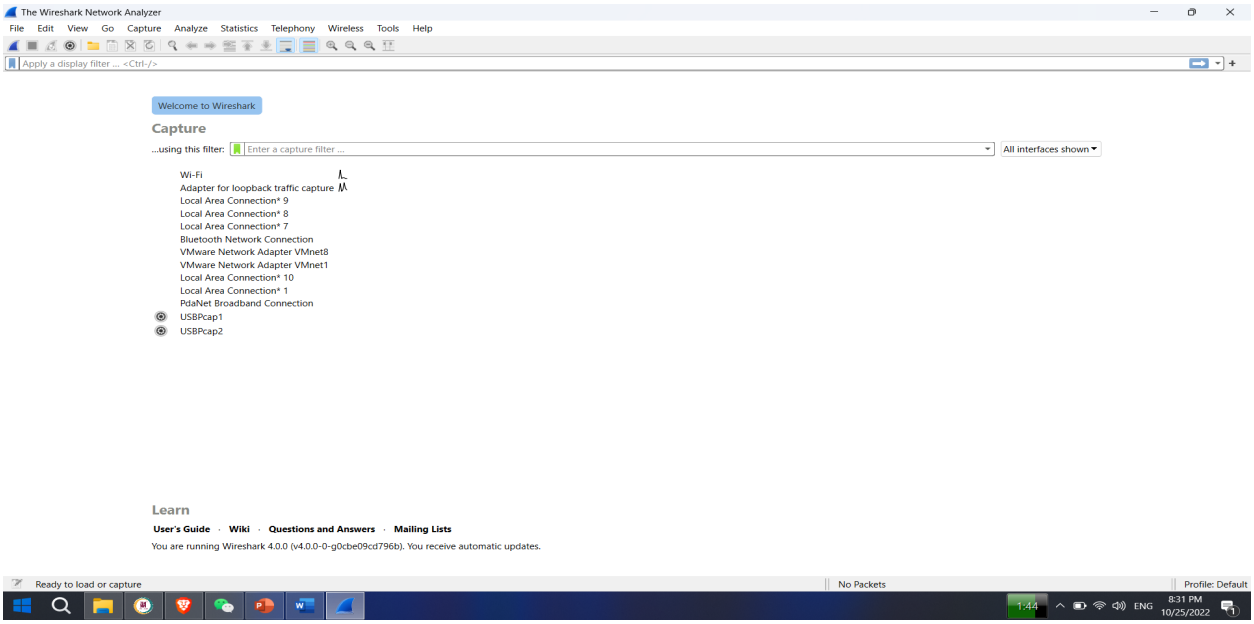


Figure: Opened Wireshark Application and pressing wifi

Step 4:

(1) Press this button  to capture network, pasted the link the browser http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html

(2) When we saw the GET http message along with TCP 3 way handshake ,we press stop the button  .

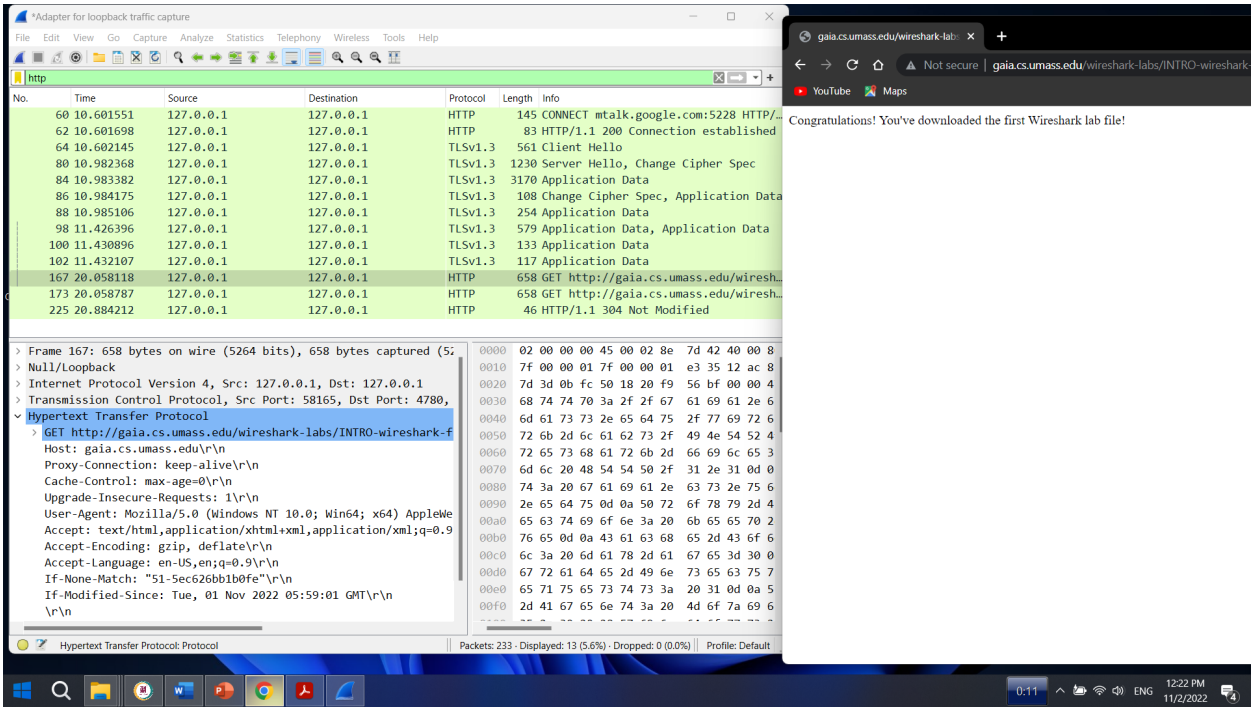


Figure 1: Http Get message

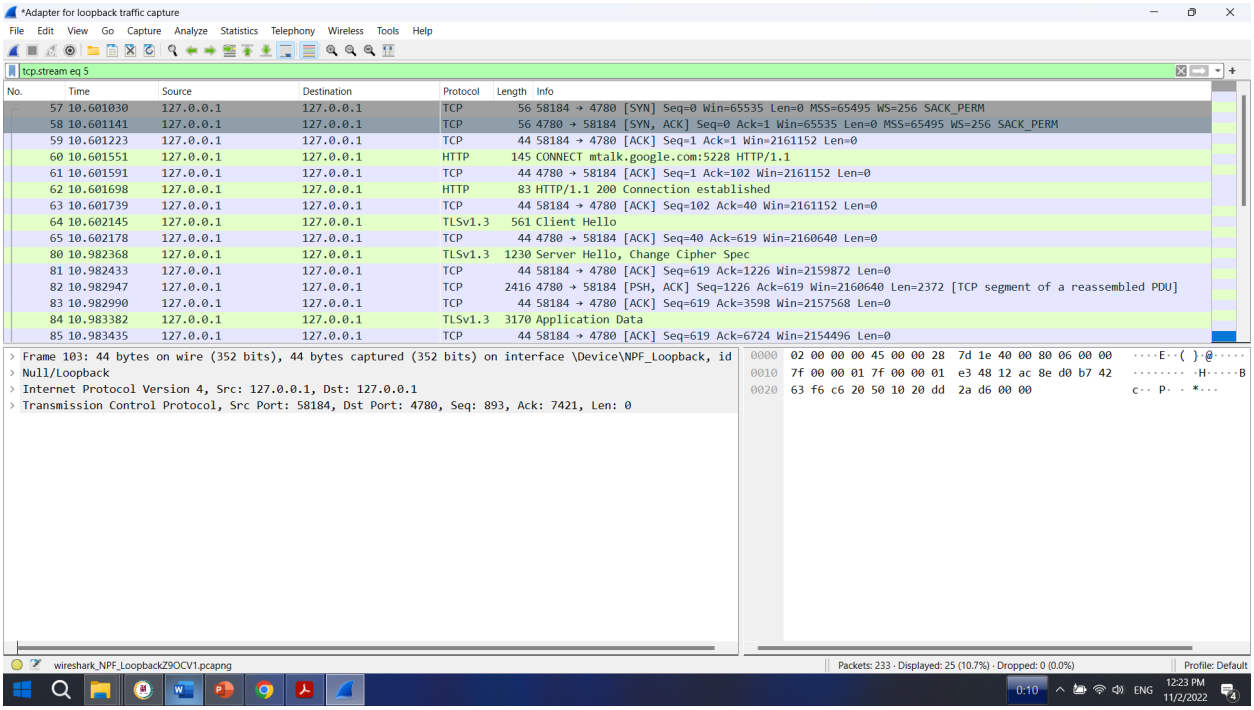


Figure 2: 3 way handshake

Step 5:

We are analyzing the packets

(1) (client) SYN

(2) (server) SYN ACK

(3) (client) ACK

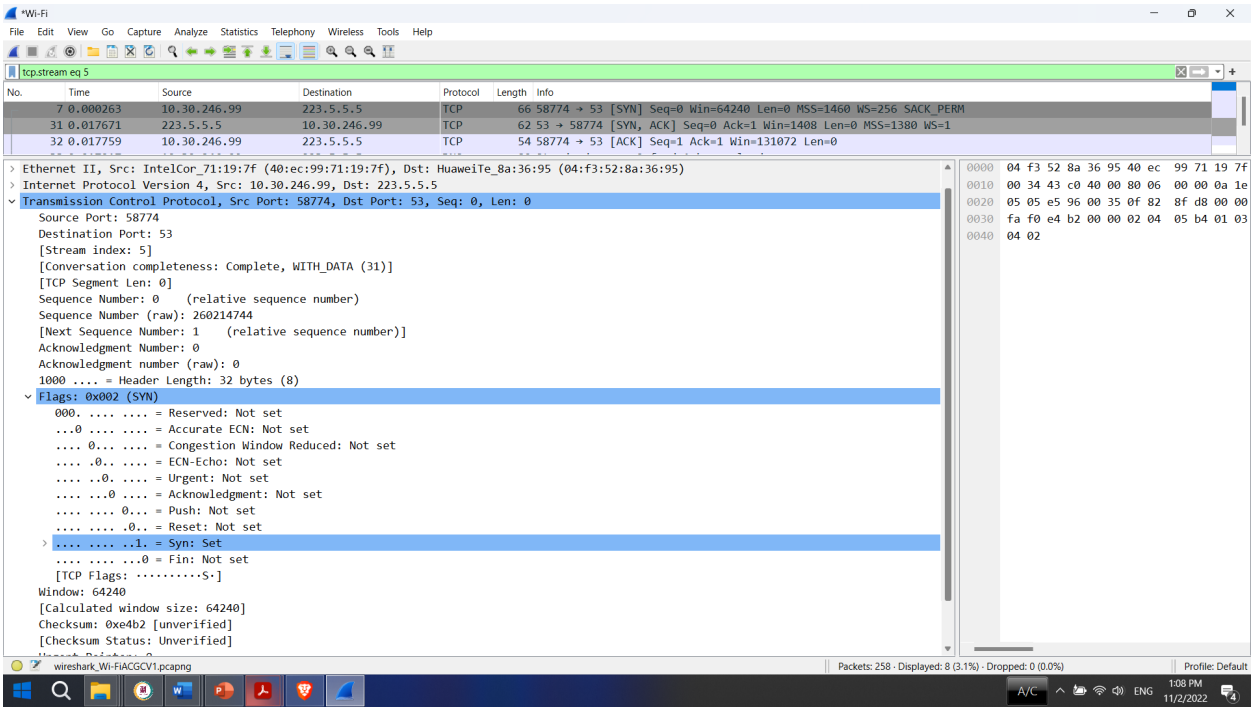


Figure 3: (client) SYN

- (client) SYN:
  - Flag = SYN
  - Seq. No = 0;

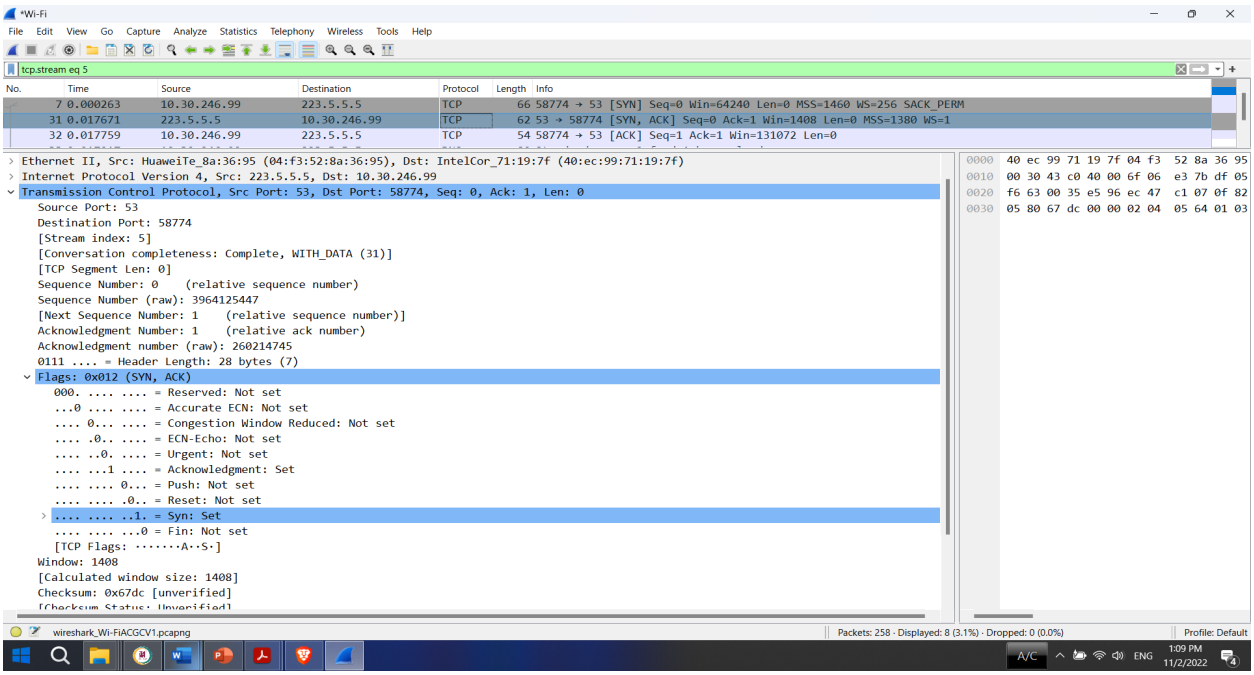


Figure 4: (server) SYN ACK

- (server) SYN ACK:
  - Flag = SYN ACK
  - Seq. No = 0;
  - ACK No = 1;

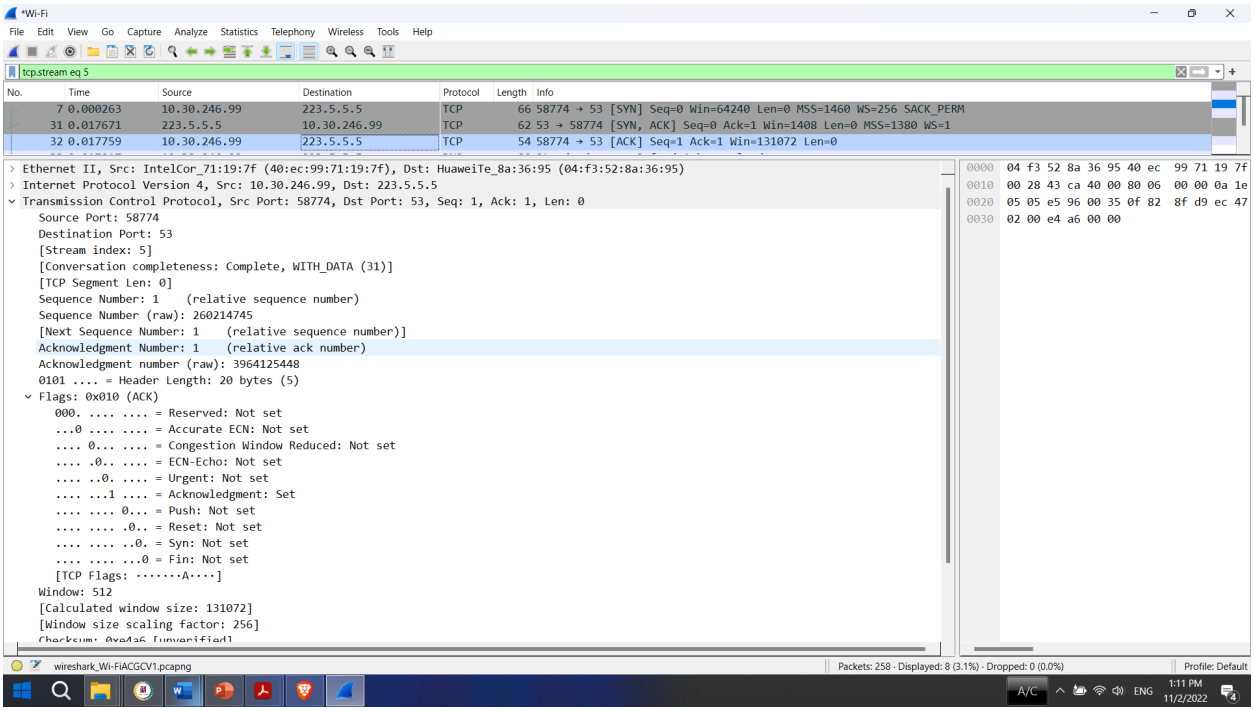


Figure 6: (client) ACK

- (client) ACK:
  - Flag = ACK;
  - Seq. No = 1;
  - ACK No = 1;

Main source 10.30.246.99(Client) and destination 223.5.5.5(Server)

We can see from the experiment the 3 way communication was done successfully.

TCP 4-way handshake via Wireshark

Step 1 (FIN From Client) – Suppose that the client application decides it wants to close the connection. (Note that the server could also choose to close the connection). This causes the client to send a TCP segment with the FIN bit set to 1 to the server and to enter the FIN_WAIT_1 state. While in the FIN_WAIT_1 state, the client waits for a TCP segment from the server with an acknowledgment (ACK).

Step 2 (ACK From Server) – When the Server received the FIN bit segment from Sender (Client), Server Immediately sends acknowledgement (ACK) segment to the Sender (Client).

Step 3 (Client waiting) – While in the FIN_WAIT_1 state, the client waits for a TCP segment from the server with an acknowledgment. When it receives this segment, the client enters the FIN_WAIT_2 state. While in the FIN_WAIT_2 state, the client waits for another segment from the server with the FIN bit set to 1.

Step 4 (FIN from Server) – The server sends the FIN bit segment to the Sender(Client) after some time when the Server sends the ACK segment (because of some closing process in the Server).

Step 5 (ACK from Client) – When the Client receives the FIN bit segment from the Server, the client acknowledges the server's segment and enters the TIME_WAIT state. The TIME_WAIT state lets the client resend the final acknowledgment in case the ACK is lost. The time spent by clients in the TIME_WAIT state depends on their implementation, but their typical values are 30 seconds, 1 minute, and 2 minutes. After the wait, the connection formally closes and all resources on the client-side (including port numbers and buffer data) are released.

Process of capturing

Step 1: Start browser and wait few minutes.

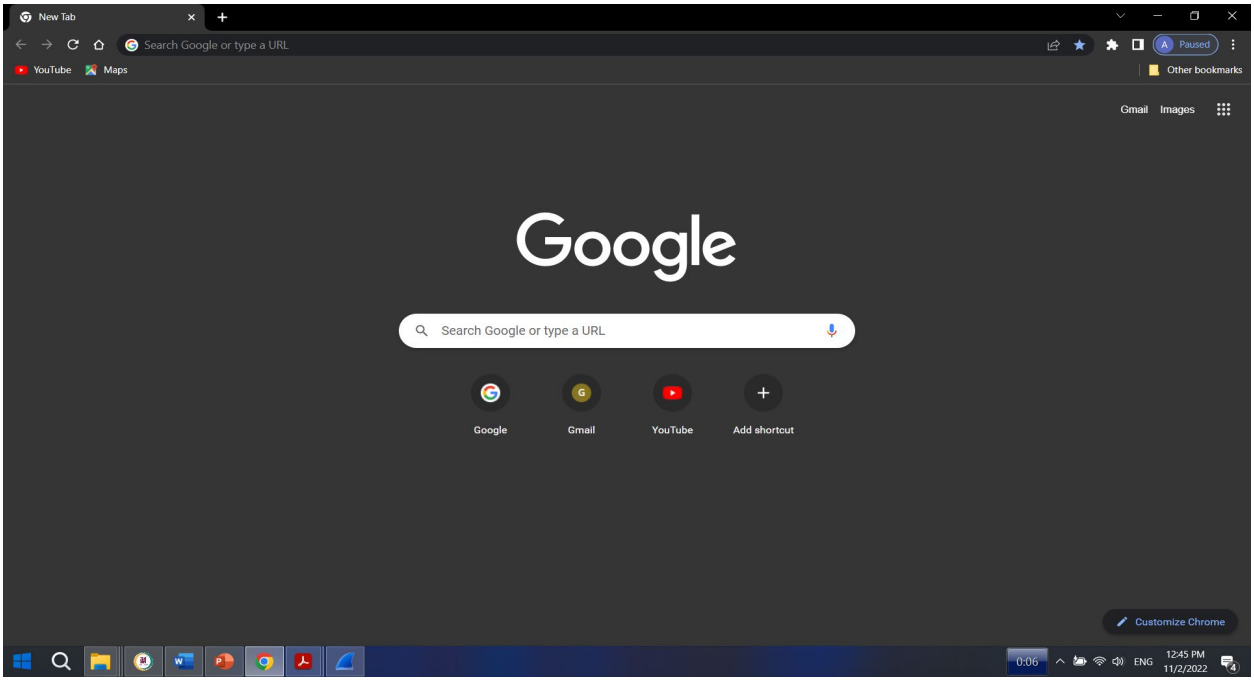


Figure: Opening The Browser

Step 2: Start Wireshark and wait wait few minutes .

Double-click the Wireshark icon 

, which is located on the desktop.

Step 3: Press wifi capturing packets

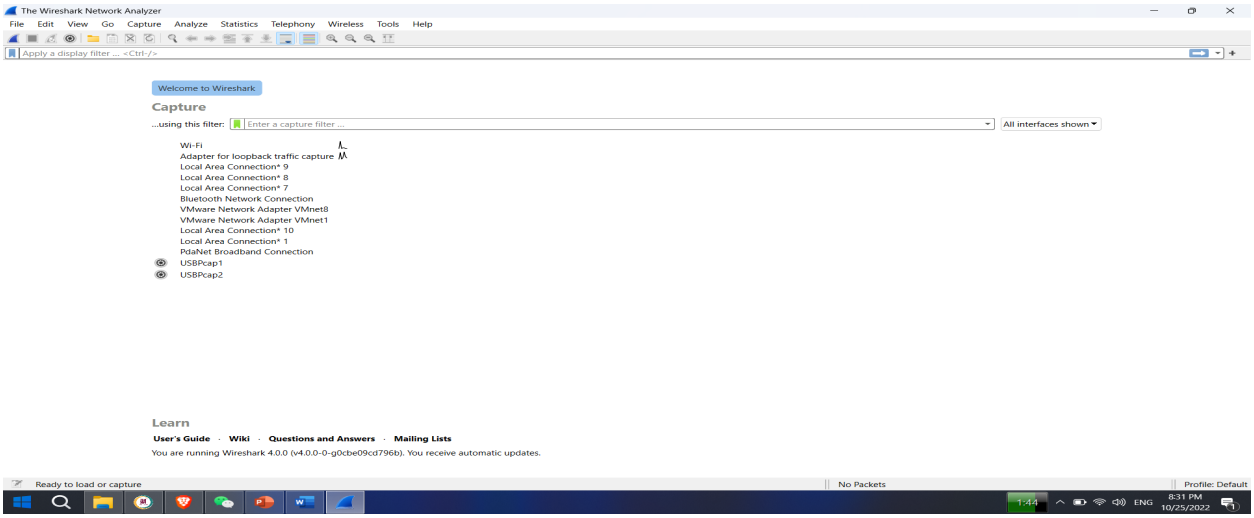


Figure: Opened Wireshark Application and pressing wifi

Step 4:

(1) Press this button  to capture network, pasted the link the browser http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html

(2) When we saw the GET http message along with TCP 4 way handshake ,we press stop the button  .

Step 5:

4-way handshake analysis via wireshark

- **FIN, ACK - ACK - FIN, ACK - ACK**

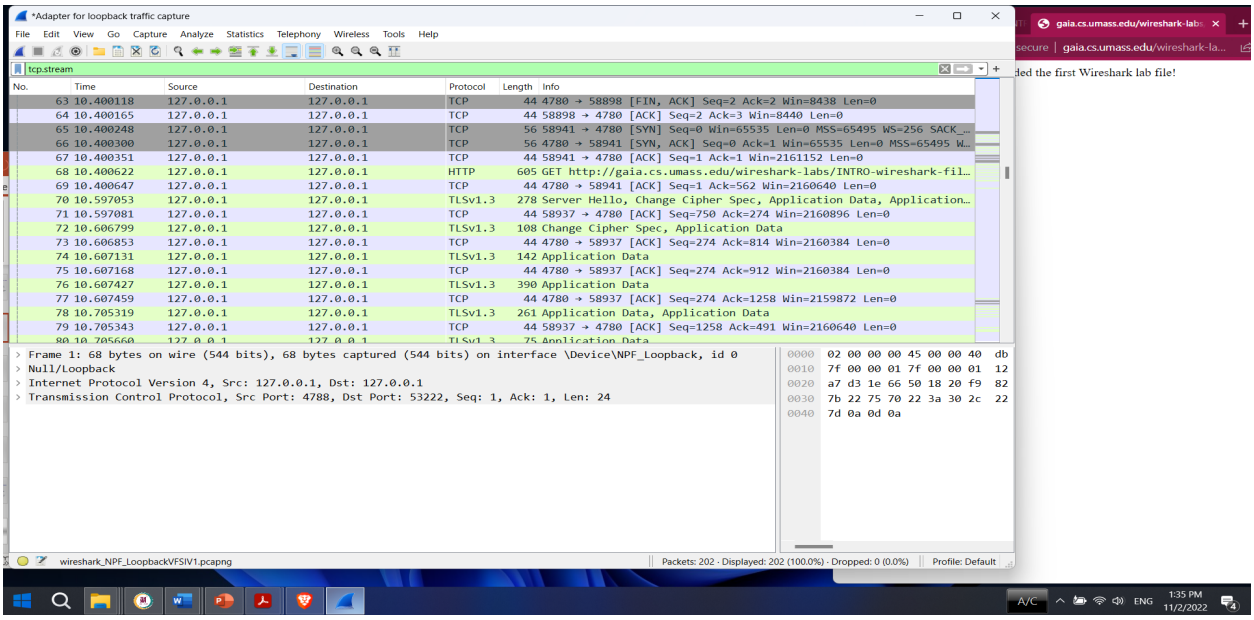


Figure 1: Http Get message

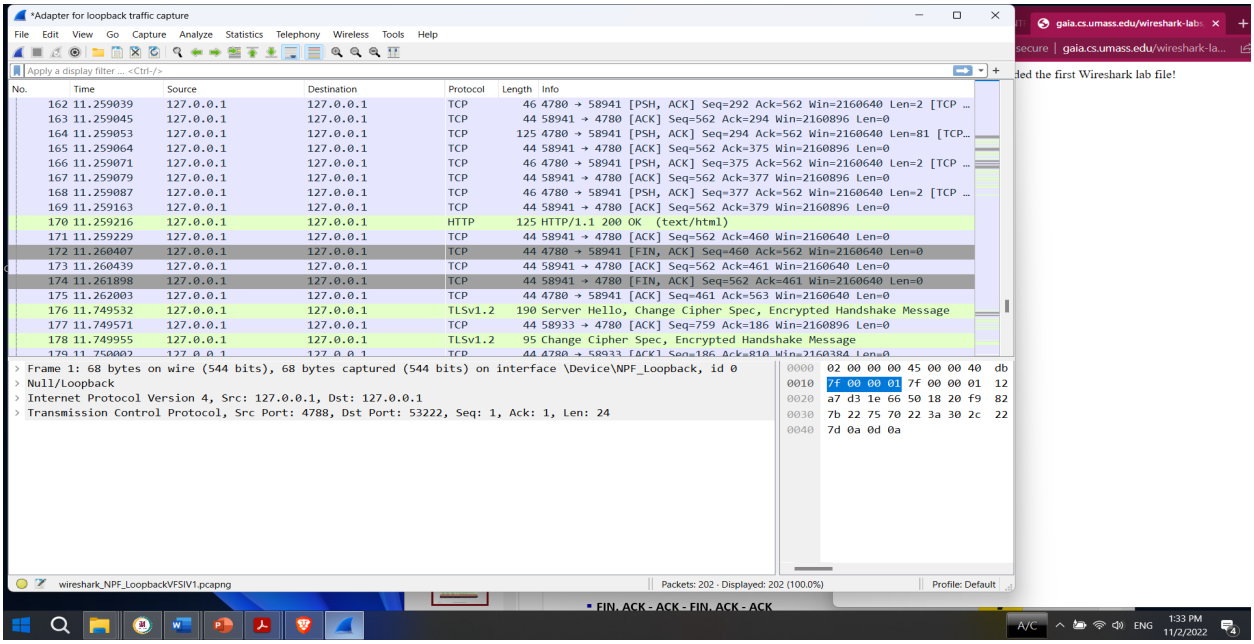


Figure : Http Get message (1)

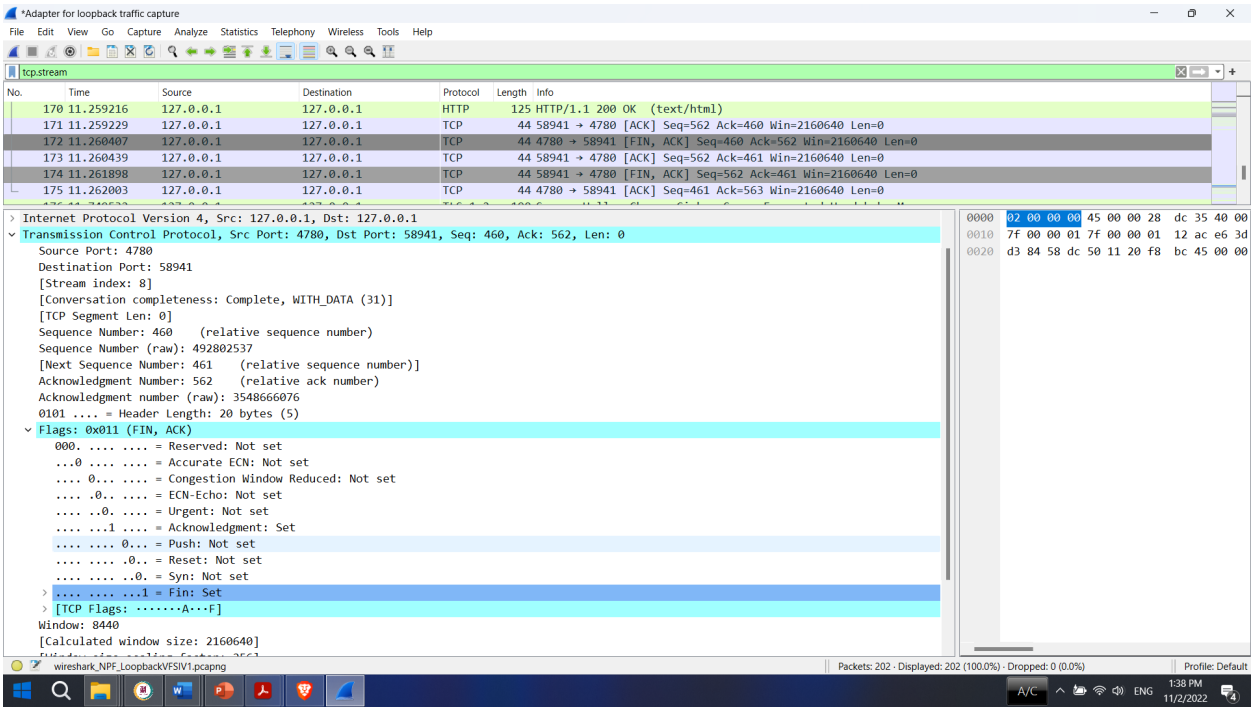


Figure : FIN,ACK (Server)

- FIN,ACK (Server)
  - Seq. No = 460;
  - ACK No = 562

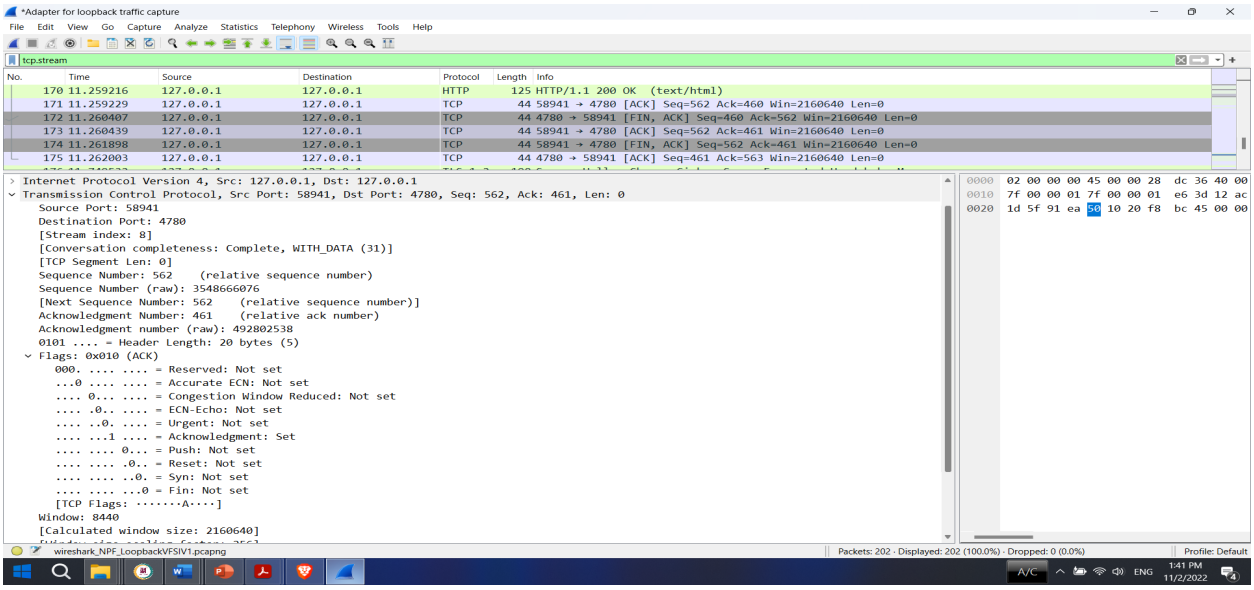


Figure : ACK (Client)

- ACK (Client)

- Seq. No = 562;
- ACK No = 461

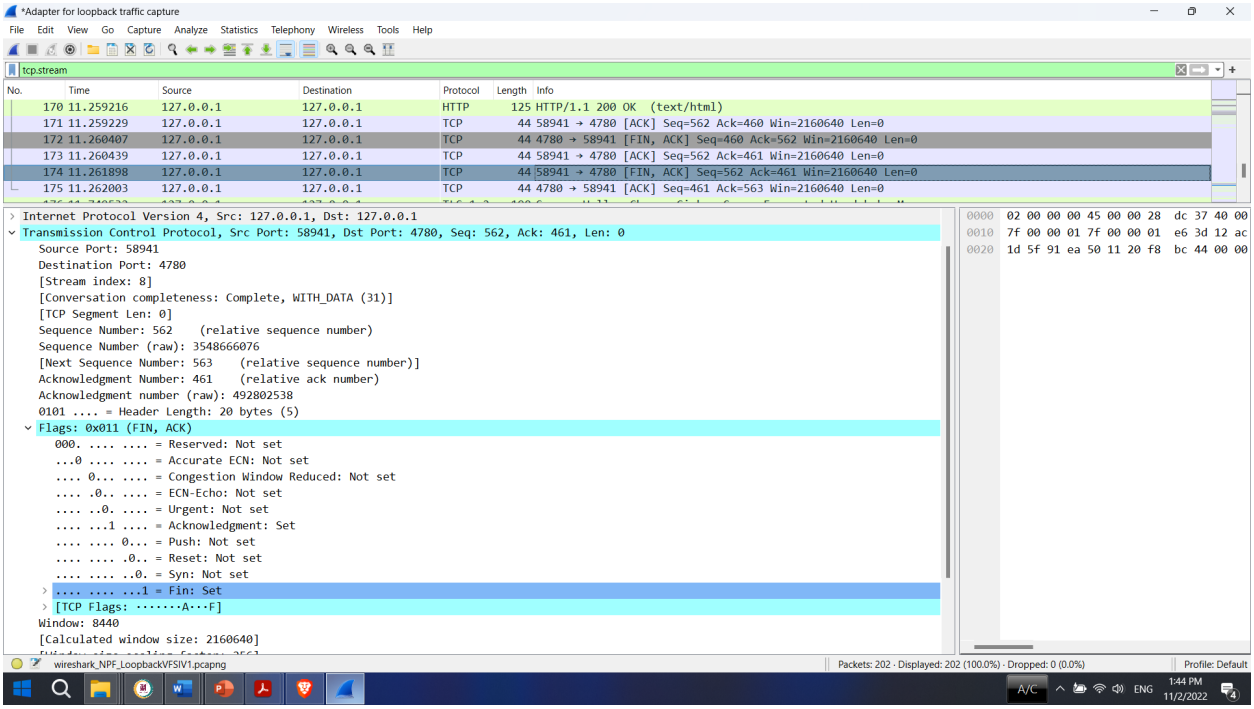


Figure : FIN, ACK (Client)

- FIN, ACK (Client)
  - Seq. No = 562;
  - ACK No = 461

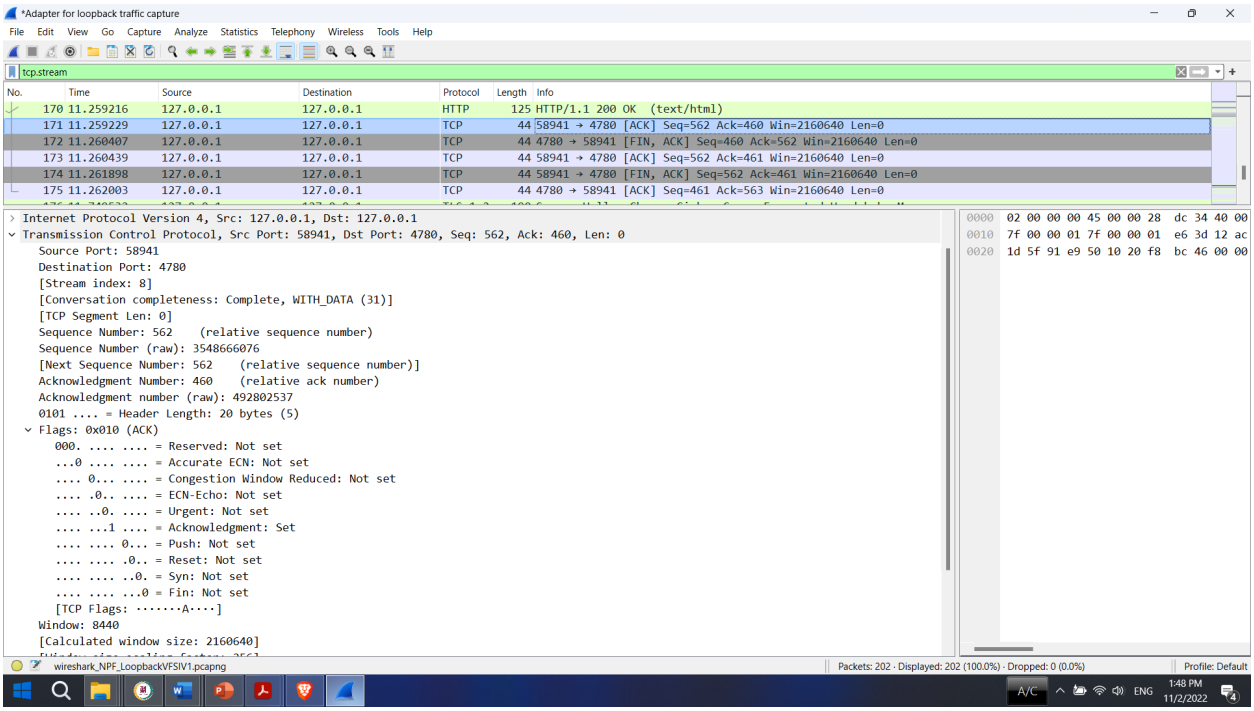


Figure : ACK (Server)

- ACK (Server)
    - ACK No = 460

Let's examine each of them and their relevance individually:

- **Source port**: This is the port of your host network used for communication.
- **Destination port**: This is the port of the destination server.
- **TCP segment length**: It represents the data length in the selected packet.
- **Sequence number**: It is a method used by Wireshark to give particular indexing to each packet for tracking packets with ease. This indexing starts from 0.
- **Next sequence number**: It is the sum of the sequence number and the segment length of the current packet.
- **Acknowledgment number**: It contains the byte length of data received.
- **Header length**: It is the length of the TCP header and can vary from 20 to 60.

Portion of this TCP packet analysis. The factors given below, together with their relative significances, make up the flag section.

• **Congestion window reduced (CWR)**: It signals a decrease in transmission rate.

• **ECN-Echo**: It is set on receiving earlier congestion notifications.

• **Urgent**: It is set when the packet is to be considered a priority.

• **Acknowledgment**: It indicates whether the current packet contains an acknowledgment packet or not.

• **Push**: The data should be saved and removed from the communication channel.

• **Reset**: It indicates an error in the communication.

• **Syn**: It denotes whether the packet is synchronization or SYN packet or not.

• **Fin**: It indicates finalization i.e., end of the communication

The experiment is concluded and now it's time to understand how the TCP connection is closed. Generally, it is known to us a TCP termination handshake. The subsequent phases are when it continues to occur: The FIN or finalization packet is sent by the closing side or local host. The server transmits an ACK indicating that it has received the FIN packet and a FIN packet for closing-side confirmation.

Finally, the closing side acknowledges the connection termination by receiving the FIN packet and responding by sending the ACK message. You can glance at the diagram below for a better understanding.

UDP Analysis using Wireshark

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network.The UDP enables process to process communication.

Though Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services; provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency. Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth.

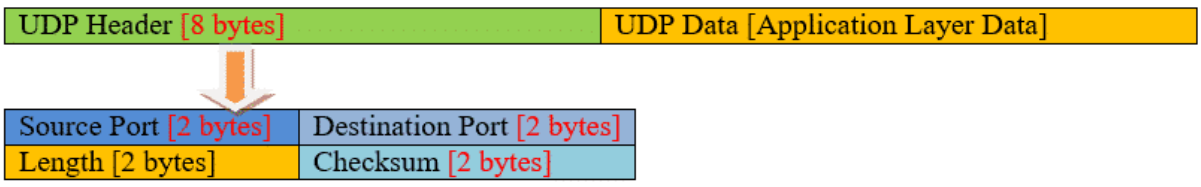
User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

Another similar protocol is Transport Control Protocol (TCP), which aims to offer dependable connections with built-in error management. It is intended for applications that do not emphasize traffic latency but need to ensure that data reaches its destination unaltered.
UDP, on the other hand, is intended for usage in situations where reliability takes precedence above delay. This protocol is "fire and forget," not keeping track of state. The destination does not acknowledge receipt of a sent UDP packet, and any missing packets are not forwarded.
UDP is therefore frequently utilized in applications where all of the data can fit into a single packet, packet loss is not a big issue (as in DNS), or high-speed transfers are required (like gaming).

Why UDP when we have TCP: The main reason is because UDP is a connectionless protocol, in contrast to TCP. This breakthrough has made UDP faster than TCP. UDP does not, however, have TCP's strong reliability. In conclusion, UDP is the transport layer protocol to employ if you don't mind giving up some reliability for increased speed.

UDP header:

UDP header is very simple and only 8 bytes.



Originating port: The packet's source port number. Instance: 4444. Port of destination: The packet's designated port. Instance: 51164. Length: The sum of the UDP data and UDP header length. Checksum: Checksum is there to find mistakes. Checksum computation is not required in UDP, in contrast to TCP. UDP doesn't offer any flow control or error control. Because of this, UDP relies on IP and ICMP for error reporting.

Process of capturing

Step 1: Start browser and wait few minutes.

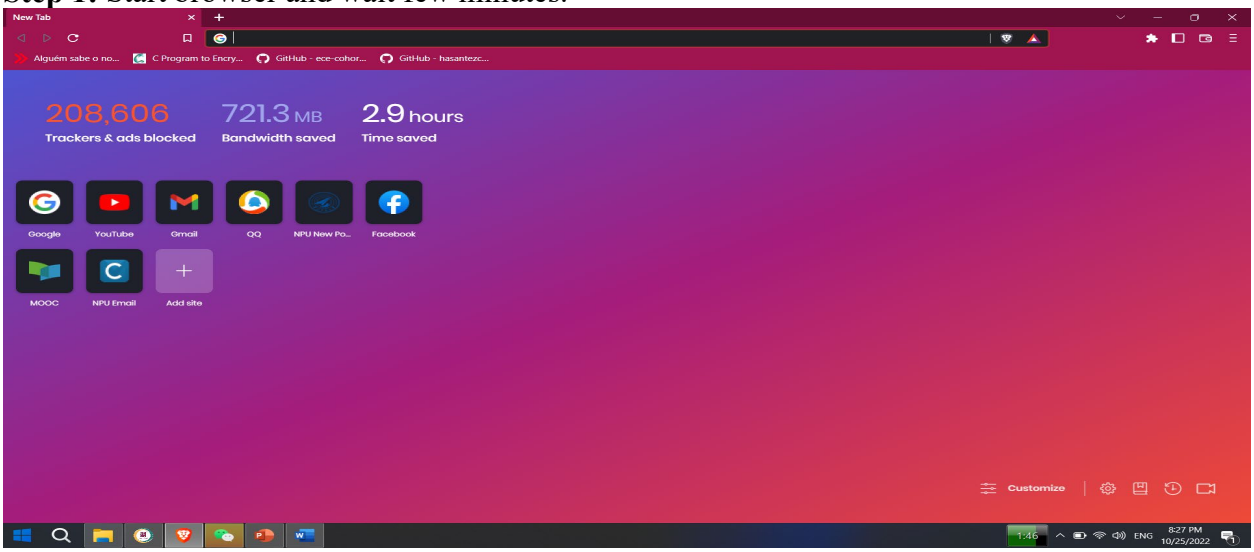


Fig: Opening The Browser

Step 2: Start Wireshark and wait wait few minutes .

Double-click the Wireshark icon 

, which is located on the desktop.

Step 3: Press wifi capturing packets

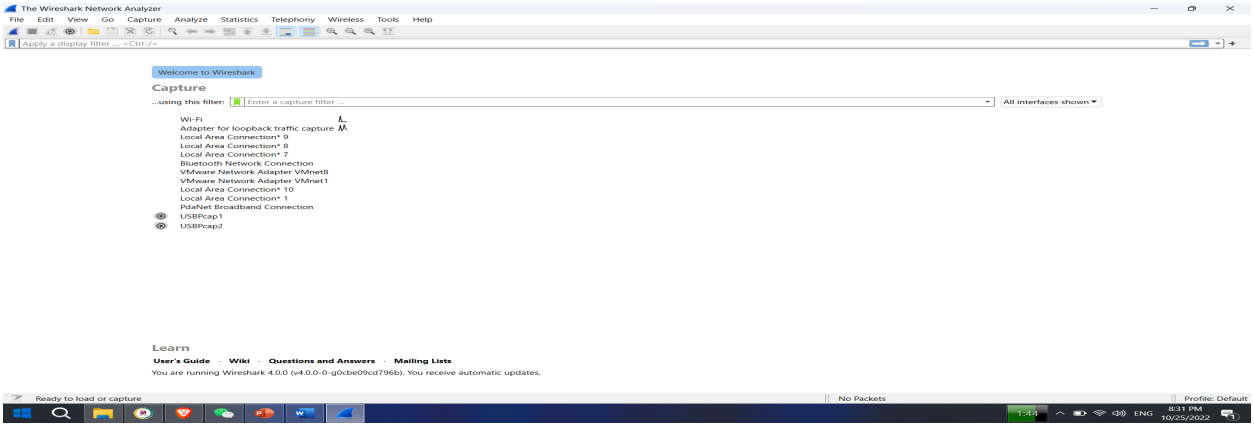


Figure: Opened Wireshark Application and pressing wifi

Step 4:

(1) Press this button  to capture UDP packets

(2) When we saw the UDP packet, we press stop the button .

Step 5:

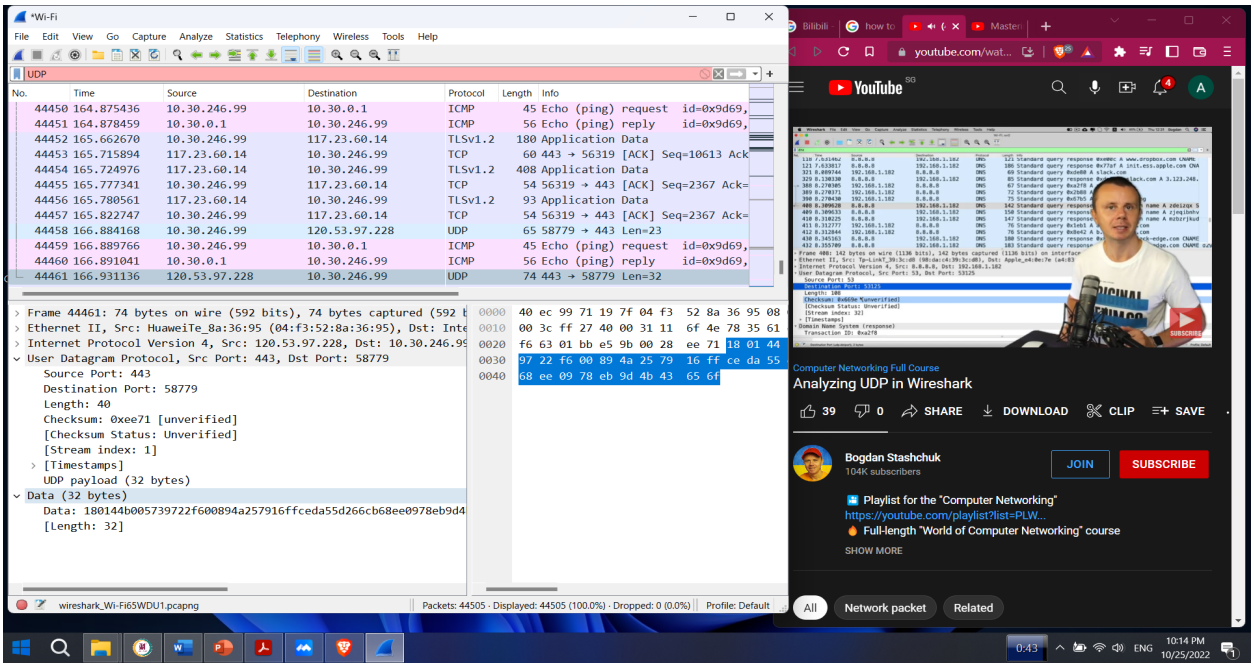


Fig: Wireshark Started and Streaming the Video

Step 6:

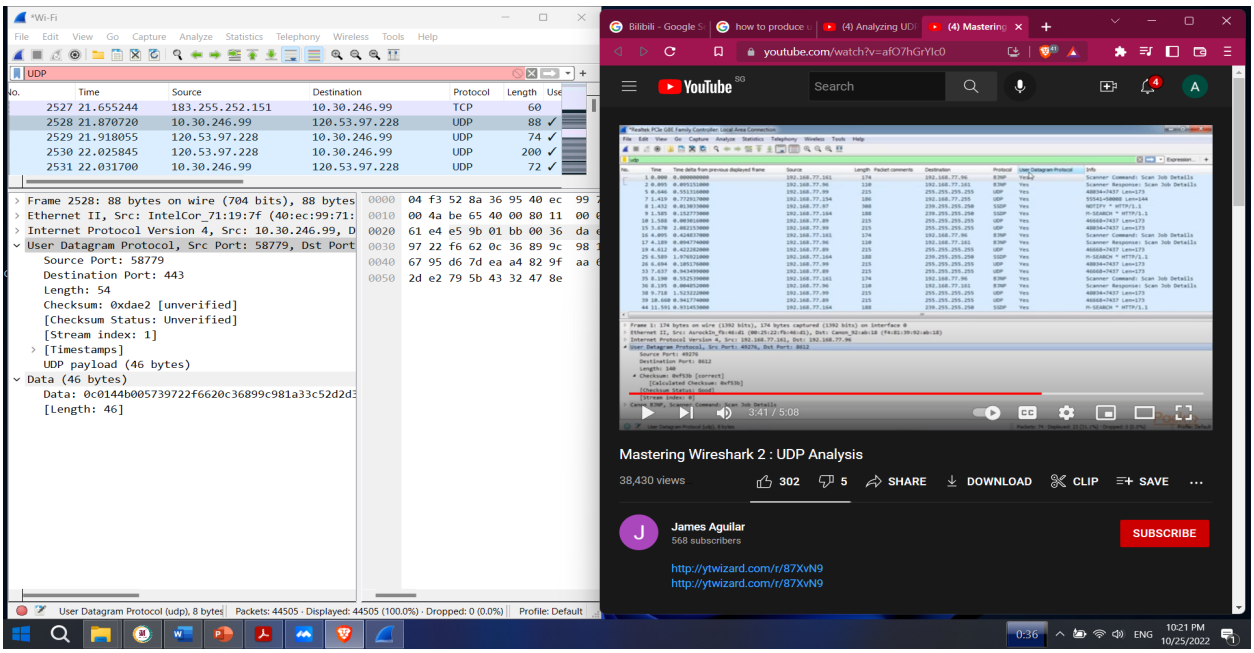


Fig: Wireshark Stopped and filtering UDP

Alternate Process:

Start tencent meeting application and wait few minutes.

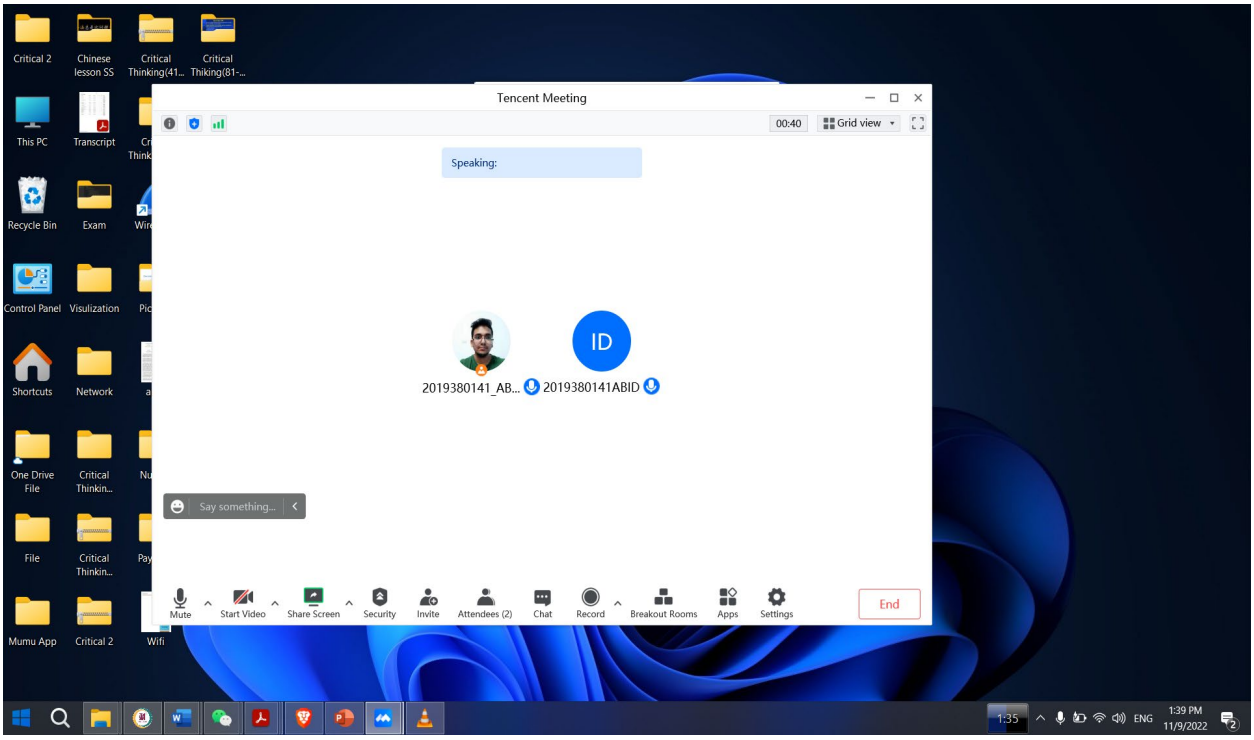


Figure: Tencent Meeting Application

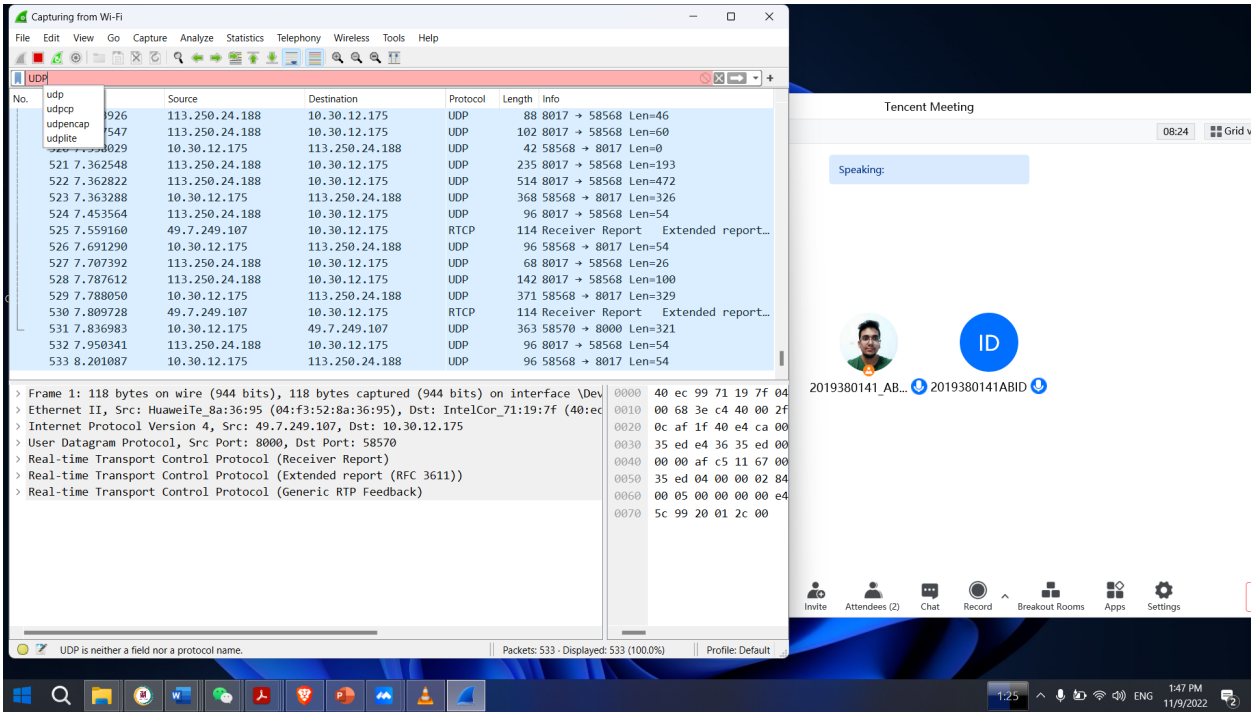


Figure: UDP Filtering

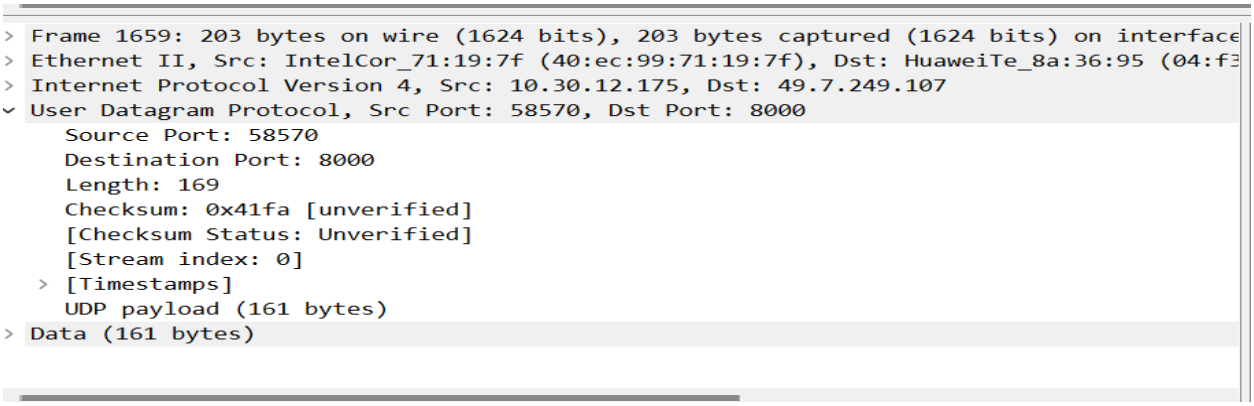


Figure: User Data Protocol Expanded

UDP's focal points are:

1. There is no frame exchange for UDP connections.
2. For UDP packets, there is no UDP transport layer ACK.
3. You can employ the UDP protocol depending on your application's needs.

Conclusion

Issues faced during Experiment

1. WireShark app was new to me so I faced at the beginning to understand.

2. Sometimes due to different reason, I didn't get the desire result. I had to restart the internet or restart the Wireshark then we got our desired result.

3. When I didn't follow the exact sequence then I didn't get the desired result. Then I refer to teacher lecture or online to solve the problem.

The gains from this Course/Experiment

This course introduces to me a new world of Internet which I was not familiar with those concepts. I learned about HTTP, UDP and TTP. Now I learned about WireShark Interface how to capture packets and analyzing those packets.

I really enjoyed learning the course. We learned those knowledge steps:

1. Sniffing technology for Lan networks that works.
2. Ip packet tracing and continuous monitoring.
3. Monitoring of IP headers, sources, destinations, and other data.
4. Monitoring of TCP header, port, and related data.
5. An efficient GUI for ongoing network parameter monitoring.