

EXPERIMENT NO. 6

SEMESTER: V

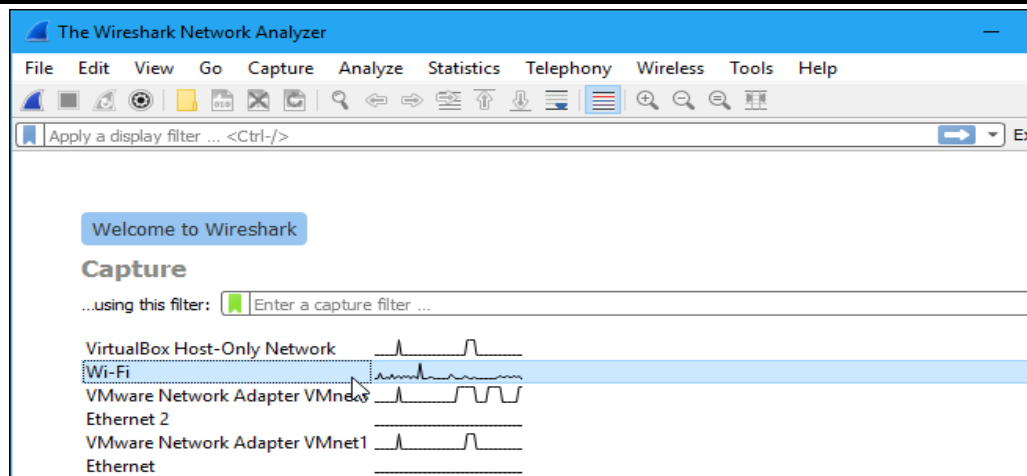
DATE OF PERFORMANCE: 28th August 2024

SUBJECT: CN Lab

DATE OF SUBMISSION: 01st September 2024

NAME OF THE STUDENT: Dwayne George Nixon ROLL NO.: 21

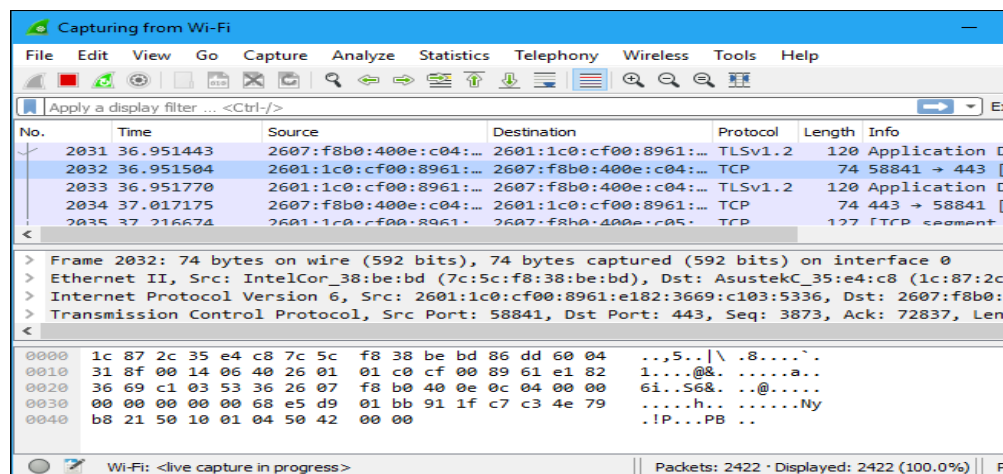
AIM	Use wire shark to understand the TCP/IP operations.Ethernet Layer: Frame header, Frame size etc. <ul style="list-style-type: none">● Data Link Layer: MAC address, ARP (IP and MAC address binding)● Network Layer: IP Packet (header, fragmentation), ICMP (Query andEcho)● Transport Layer: TCP Ports, TCP handshake segments etc.● Application Layer: DHCP, FTP, HTTP header formats
LEARNING OBJECTIVE	Students will be able to sniff and find protocol stack details from packets Students will be able to use Wireshark filters effectively.
LEARNING OUTCOME	The student will illustrate the use of Wireshark tool to understand the operations ifTCP/IP layers.
COURSE OUTCOME	CSL502.5: Review various operations of TCP/IP layers using Wire shark.
PROGRAM OUTCOME	PO1,PO2,PO3,PO4,PO5,PO9,PO10,PSO1,PSO2,PSO3
BLOOM'S TAXONOMY LEVEL	Evaluate
THEORY	<p>Wireshark, a network analysis tool formerly known as Ethereal, captures packetsin real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets.</p> <p>Capturing Packets</p> <p>After downloading and installing Wireshark, you can launch it and double-clickthe name of a network interface under Capture to start capturing packets on thatinterface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features by clicking Capture > Options, but this isn't necessary for now.</p>



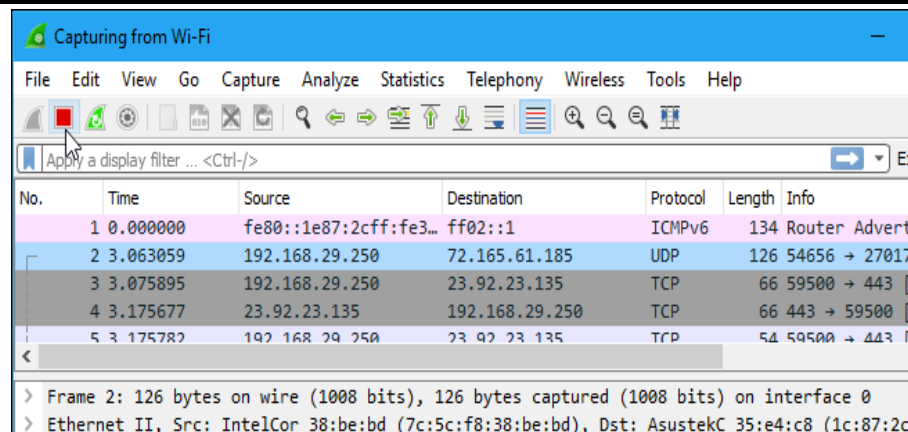
As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture

> Options and verify the "Enable promiscuous mode on all interfaces" checkbox is activated at the bottom of this window.



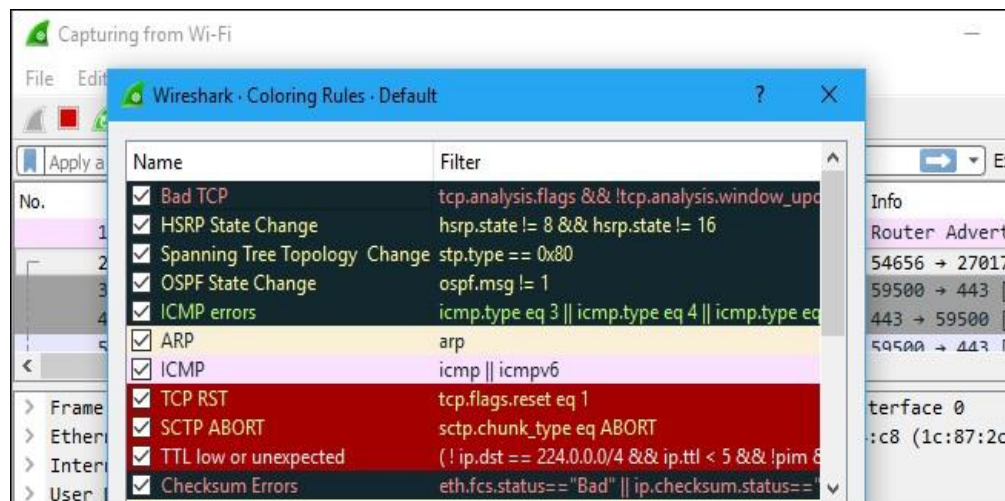
Click the red "Stop" button near the top left corner of the window when you want to stop capturing traffic.



Color Coding

You'll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

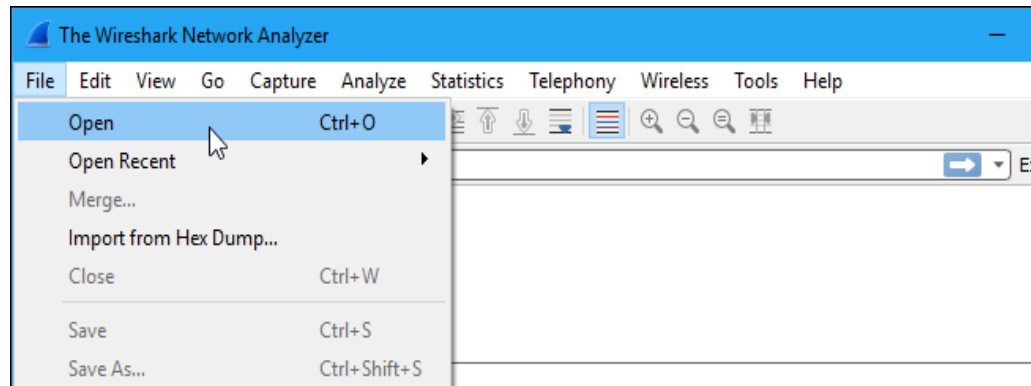
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.



Sample Captures

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a [page of sample capture files](#) that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

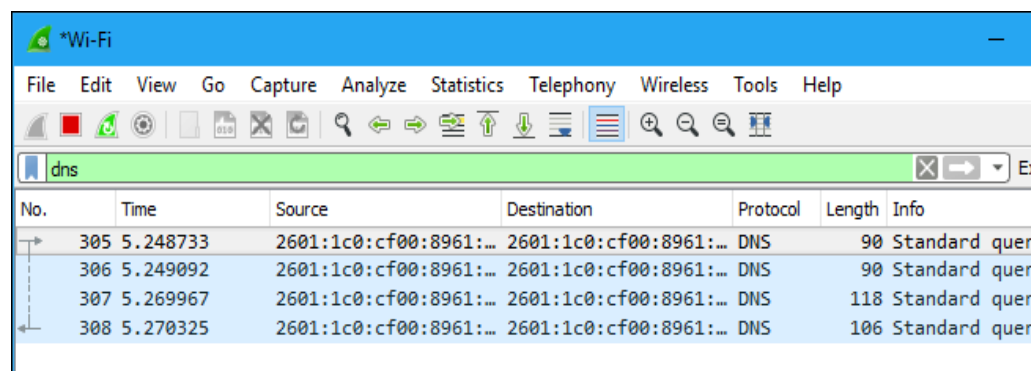
You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.



Filtering Packets

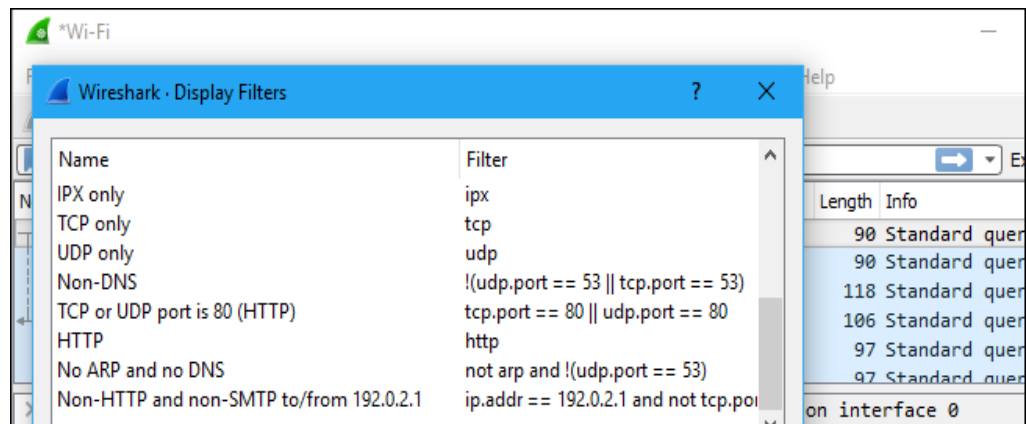
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.



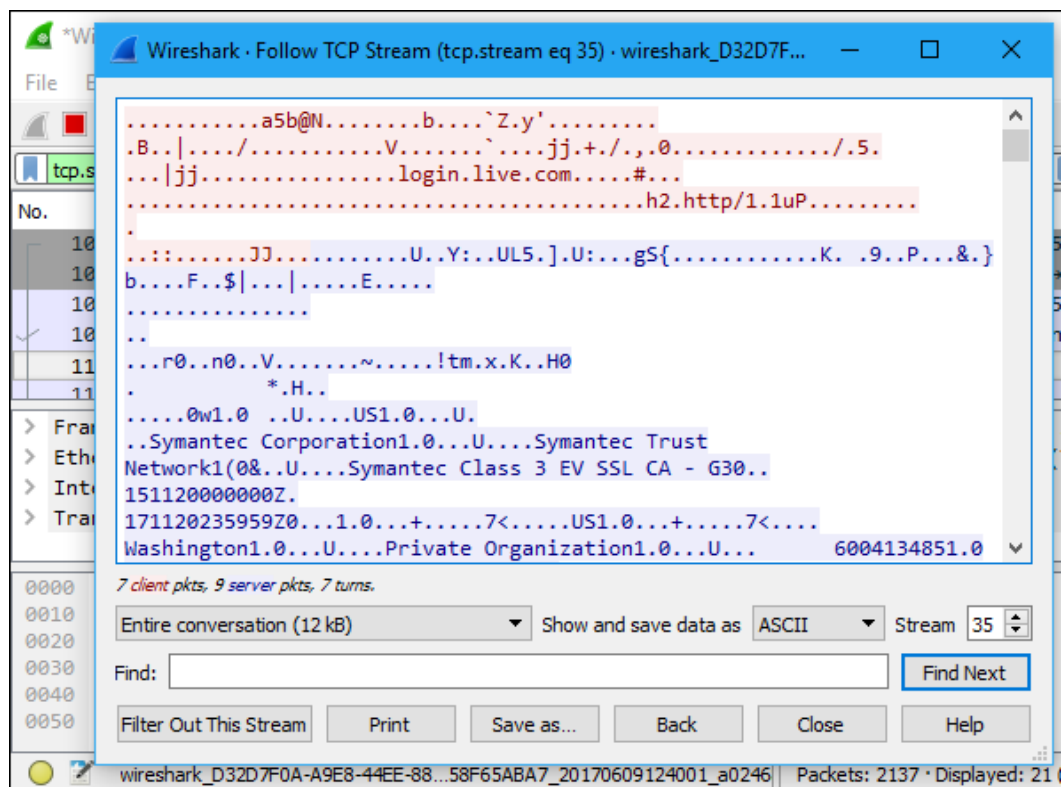
You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

For more information on Wireshark's display filtering language, read the [Building display filter expressions](#) page in the official Wireshark documentation.

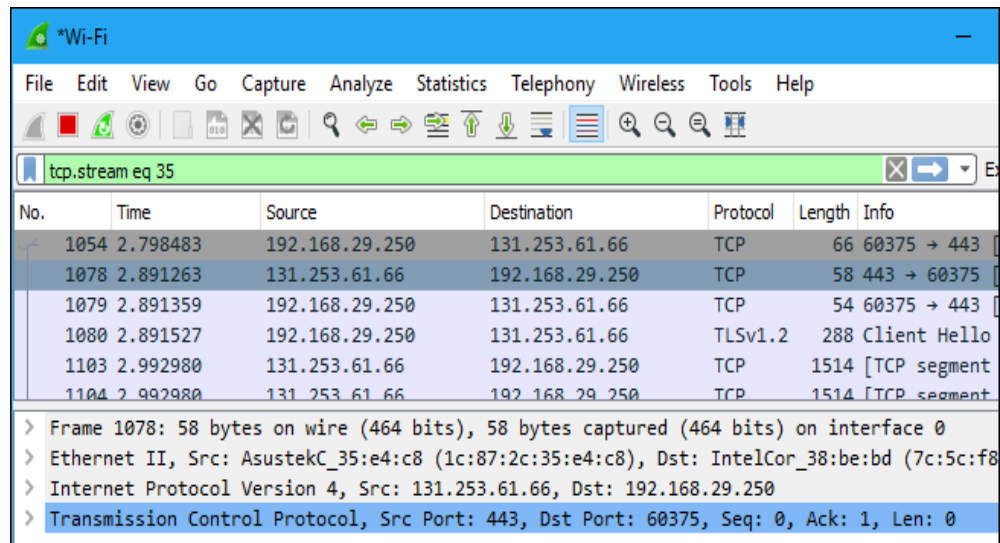


Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.

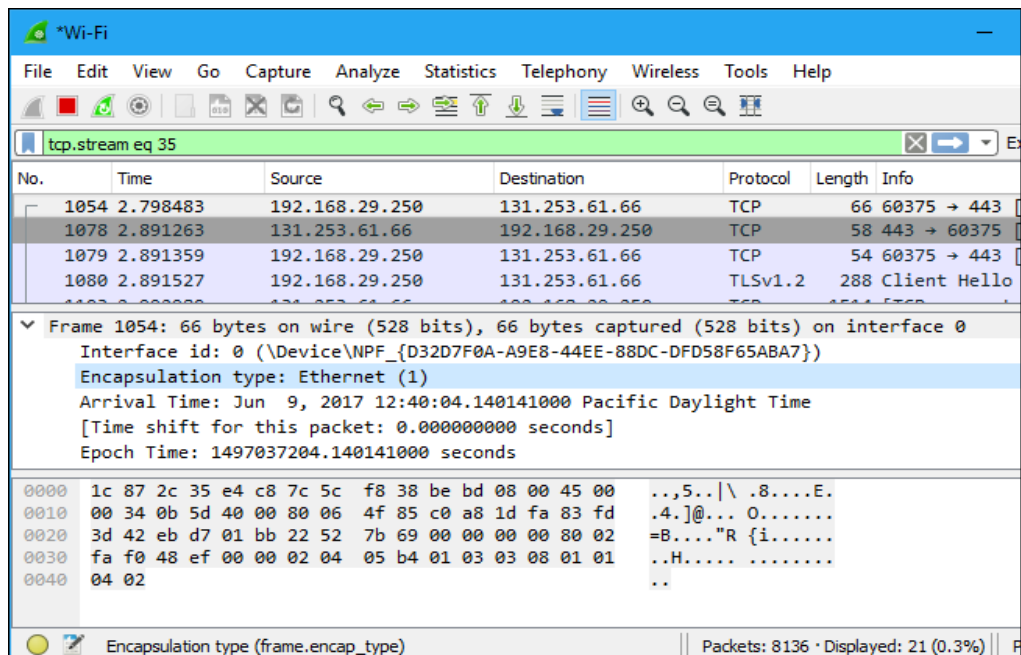


Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

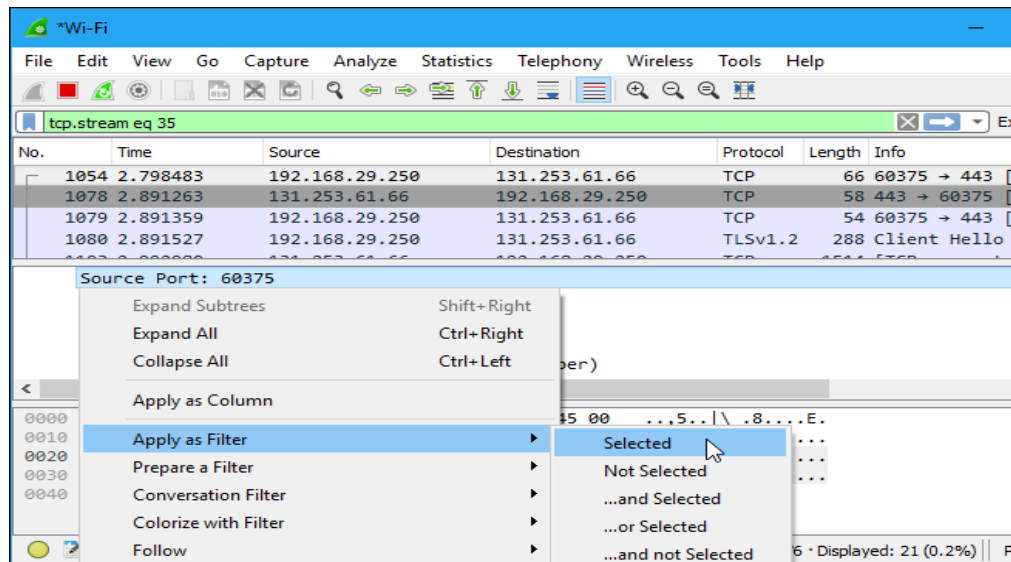


Inspecting Packets

Click a packet to select it and you can dig down to view its details.



You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.



Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

LAB EXERCISE

- Trace one connection using Wireshark and answer the following question with screenshots

Is the frame an outgoing or an incoming frame?

Source IP address of the network-layer header in the frame:

Destination IP address of the network-layer header in the frame:

Total number of bytes in the whole frame:

Number of bytes in the Ethernet (data-link layer) header:

Number of bytes in the IP header:

Number of bytes in the TCP header:

Total bytes in the message (at the application layer):

Also try:

- Capture packets if you were using youtube (TCP contains youtube or ipaddress of youtube)
- Login through some unsecure website and capture the password

LAB EXERCISE OUTCOMES:

- Trace one connection using Wireshark and answer the following question with screenshots

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The packet list pane on the left shows a series of captured packets. The selected packet (No. 7410) is highlighted in blue. The packet details pane on the right shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, and User Datagram Protocol (UDP). The packet bytes pane at the bottom shows the raw data of the selected packet.

Connection to the Internet via Ethernet

Don Bosco Institute of Technology, Kurla

Academic Year 2023-24

No.	Time	Source	Destination	Protocol	Length	Info
7380	53.483932661	142.250.183.142	192.168.0.177	QUIC	76	Protected Payload (KPO), DCID=a2a0ad
7387	53.484182417	192.168.0.177	142.250.183.142	QUIC	594	Protected Payload (KPO), DCID=f507527cee2c55fb
7388	53.485350999	142.250.183.142	192.168.0.177	QUIC	71	Protected Payload (KPO), DCID=a2a0ad
7389	53.488035760	142.250.183.142	192.168.0.177	QUIC	71	Protected Payload (KPO), DCID=a2a0ad
7390	53.571487104	142.250.183.142	192.168.0.177	QUIC	1394	Protected Payload (KPO), DCID=a2a0ad
7391	53.571488990	142.250.183.142	192.168.0.177	QUIC	1399	Protected Payload (KPO), DCID=a2a0ad
7392	53.571490387	142.250.183.142	192.168.0.177	QUIC	1399	Protected Payload (KPO), DCID=a2a0ad
7393	53.571491305	142.250.183.142	192.168.0.177	QUIC	254	Protected Payload (KPO), DCID=a2a0ad
7394	53.571492761	142.250.183.142	192.168.0.177	QUIC	1399	Protected Payload (KPO), DCID=a2a0ad
7395	53.571494228	142.250.183.142	192.168.0.177	QUIC	116	Protected Payload (KPO), DCID=a2a0ad
7396	53.572346584	192.168.0.177	142.250.183.142	QUIC	81	Protected Payload (KPO), DCID=f507527cee2c55fb
7397	53.576765789	142.250.183.142	192.168.0.177	QUIC	71	Protected Payload (KPO), DCID=a2a0ad
7398	53.616418226	192.168.0.177	142.250.70.99	TCP	66	[TCP Keep-Alive] 54322 - 80 [ACK] Seq=2615 Ack=4212 Win=64128 Len=0 TSval=183575560 TSecr=1860470716
7399	53.619820946	142.250.70.99	192.168.0.177	TCP	66	[TCP Keep-Alive ACK] 80 - 54322 [ACK] Seq=4212 Ack=2616 Win=82432 Len=0 TSval=1860480972 TSecr=1835653606
7400	54.042611977	192.168.0.177	192.168.0.1	DNS	89	Standard query 0x12ff HTTPS bbc.map.fastly.net OPT
7401	54.043249428	192.168.0.177	192.168.0.1	DNS	89	Standard query 0x5b1d A bbc.map.fastly.net OPT
7402	54.043840790	192.168.0.177	192.168.0.1	DNS	89	Standard query 0xf11c AAAA bbc.map.fastly.net OPT
7403	54.045678804	192.168.0.177	192.168.0.1	DNS	100	Standard query response 0xf11c AAAA bbc.map.fastly.net SOA ns1.fastly.net OPT
7404	54.046851757	192.168.0.177	192.168.0.1	DNS	100	Standard query response 0x12ff HTTPS bbc.map.fastly.net SOA ns1.fastly.net OPT
7405	54.047809645	192.168.0.177	192.168.0.1	DNS	105	Standard query response 0x5b1d A bbc.map.fastly.net A 151.101.152.01 OPT
7406	55.152428035	192.168.0.177	142.250.70.99	TCP	66	[TCP Keep-Alive] 54314 - 80 [ACK] Seq=3919 Ack=6314 Win=64128 Len=0 TSval=183577096 TSecr=3541187898
7407	55.152458068	192.168.0.177	142.250.70.99	TCP	66	[TCP Keep-Alive] 54326 - 80 [ACK] Seq=3952 Ack=4915 Win=64128 Len=0 TSval=183577096 TSecr=333985523
7408	55.155366583	142.250.70.99	192.168.0.177	TCP	66	[TCP Keep-Alive ACK] 80 - 54326 [ACK] Seq=4915 Ack=3953 Win=73216 Len=0 TSval=3339995578 TSecr=183567044
7409	55.155433422	142.250.70.99	192.168.0.177	TCP	66	[TCP Keep-Alive ACK] 80 - 54314 [ACK] Seq=6314 Ack=3920 Win=75264 Len=0 TSval=3541189893 TSecr=183566904

Frame 11: 387 bytes on wire (2456 bits), 387 bytes captured (2456 bits) on interface enp4s0, id 0

Section number: 1

Interface id: 0 (enp4s0)

Encapsulation type: Ethernet (1)

Arrival Time: Sep 1, 2024 14:26:18.754083320 IST

UTC Arrival Time: Sep 1, 2024 08:56:18.754083320 UTC

Epoch Arrival Time: 1725180978.754083320

[Time shift for this packet: 0.000000000 seconds]

[Time delta from previous captured frame: 0.000000000 seconds]

[Time delta from previous displayed frame: 0.000000000 seconds]

[Time since reference or first frame: 0.000000000 seconds]

Frame Number: 1

Frame Length: 387 bytes (2456 bits)

Capture Length: 387 bytes (2456 bits)

[Frame is marked: False]

[Frame is ignored: False]

[Protocols in frame: eth:ethertype:ip:udp:ssdp]

[Coloring Rule Names: UDP]

[Coloring Rule String: udp]

Ethernet II, Src: TP-Link Techno_26:71:37 (a4:2b:b0:26:71:37), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)

Internet Protocol Version 4, Src: 192.168.1.1, Dst: 239.255.255.250

User Datagram Protocol, Src Port: 1990, Dst Port: 1990

Simple Service Discovery Protocol

Frame Details

No.	Time	Source	Destination	Protocol	Length	Info
7380	53.483932661	142.250.183.142	192.168.0.177	QUIC	76	Protected Payload (KPO), DCID=a2a0ad
7387	53.484182417	192.168.0.177	142.250.183.142	QUIC	594	Protected Payload (KPO), DCID=f507527cee2c55fb
7388	53.485350999	142.250.183.142	192.168.0.177	QUIC	71	Protected Payload (KPO), DCID=a2a0ad
7389	53.488035760	142.250.183.142	192.168.0.177	QUIC	71	Protected Payload (KPO), DCID=a2a0ad
7390	53.571487104	142.250.183.142	192.168.0.177	QUIC	1394	Protected Payload (KPO), DCID=a2a0ad
7391	53.571488990	142.250.183.142	192.168.0.177	QUIC	1399	Protected Payload (KPO), DCID=a2a0ad
7392	53.571490387	142.250.183.142	192.168.0.177	QUIC	1399	Protected Payload (KPO), DCID=a2a0ad
7393	53.571491305	142.250.183.142	192.168.0.177	QUIC	254	Protected Payload (KPO), DCID=a2a0ad
7394	53.571492761	142.250.183.142	192.168.0.177	QUIC	1399	Protected Payload (KPO), DCID=a2a0ad
7395	53.571494228	142.250.183.142	192.168.0.177	QUIC	116	Protected Payload (KPO), DCID=a2a0ad
7396	53.572346584	192.168.0.177	142.250.183.142	QUIC	81	Protected Payload (KPO), DCID=f507527cee2c55fb
7397	53.576765789	142.250.183.142	192.168.0.177	QUIC	71	Protected Payload (KPO), DCID=a2a0ad
7398	53.616418226	192.168.0.177	142.250.70.99	TCP	66	[TCP Keep-Alive] 54322 - 80 [ACK] Seq=2615 Ack=4212 Win=64128 Len=0 TSval=183575560 TSecr=1860470716
7399	53.619820946	142.250.70.99	192.168.0.177	TCP	66	[TCP Keep-Alive ACK] 80 - 54322 [ACK] Seq=4212 Ack=2616 Win=82432 Len=0 TSval=1860480972 TSecr=1835653606
7400	54.042611977	192.168.0.177	192.168.0.1	DNS	89	Standard query 0x12ff HTTPS bbc.map.fastly.net OPT
7401	54.043249428	192.168.0.177	192.168.0.1	DNS	89	Standard query 0x5b1d A bbc.map.fastly.net OPT
7402	54.043840790	192.168.0.177	192.168.0.1	DNS	89	Standard query 0xf11c AAAA bbc.map.fastly.net OPT
7403	54.045678804	192.168.0.177	192.168.0.1	DNS	100	Standard query response 0xf11c AAAA bbc.map.fastly.net SOA ns1.fastly.net OPT
7404	54.046851757	192.168.0.177	192.168.0.1	DNS	100	Standard query response 0x12ff HTTPS bbc.map.fastly.net SOA ns1.fastly.net OPT
7405	54.047809645	192.168.0.177	192.168.0.1	DNS	105	Standard query response 0x5b1d A bbc.map.fastly.net A 151.101.152.01 OPT
7406	55.152428035	192.168.0.177	142.250.70.99	TCP	66	[TCP Keep-Alive] 54314 - 80 [ACK] Seq=3919 Ack=6314 Win=64128 Len=0 TSval=183577096 TSecr=3541187898
7407	55.152458068	192.168.0.177	142.250.70.99	TCP	66	[TCP Keep-Alive] 54326 - 80 [ACK] Seq=3952 Ack=4915 Win=64128 Len=0 TSval=183577096 TSecr=333985523
7408	55.155366583	142.250.70.99	192.168.0.177	TCP	66	[TCP Keep-Alive ACK] 80 - 54326 [ACK] Seq=4915 Ack=3953 Win=73216 Len=0 TSval=3339995578 TSecr=183567044
7409	55.155433422	142.250.70.99	192.168.0.177	TCP	66	[TCP Keep-Alive ACK] 80 - 54314 [ACK] Seq=6314 Ack=3920 Win=75264 Len=0 TSval=3541189893 TSecr=183566904

Frame 11: 387 bytes on wire (2456 bits), 387 bytes captured (2456 bits) on interface enp4s0, id 0

Ethernet II, Src: TP-Link Techno_26:71:37 (a4:2b:b0:26:71:37), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)

Internet Protocol Version 4, Src: 192.168.1.1, Dst: 239.255.255.250

6100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

- Differentiated Services Fields: 0x00 (DSCP: CS0, ECN: Not-ECT)

0000 00.. = Differentiated Services Codepoint: Default (0)

.... 00.. = Explicit Congestion Notification: Not ECN-Capable Transport (0)

Total Length: 293

Identification: 0x0000 (59502)

0000 = Flags: 0x0

0..... = Reserved bit: Not set

0..... = Don't fragment: Not set

0..... = More fragments: Not set

...0 0000 0000 0000 = Fragment Offset: 0

Time to Live: 3

Protocol: UDP (17)

Header Checksum: 0xc17a [Validation disabled]

[Header checksum status: Unverified]

Source Address: 192.168.1.1

Destination Address: 239.255.255.250

User Datagram Protocol, Src Port: 1990, Dst Port: 1990

Simple Service Discovery Protocol

Internet Protocol Details

Don Bosco Institute of Technology, Kurla

Academic Year 2023-24

Screenshot of Wireshark interface showing network traffic analysis. The packet list pane displays various protocols including QUIC, TCP, and DNS. The packet details pane shows the structure of a packet, including Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Simple Service Discovery Protocol (SSDP). The packet bytes pane shows the raw data of the packet.

Source Address (ip.src), 4 bytes

Packets: 7410 - Displayed: 7410 (100.0%) - Dropped: 0 (0.0%)

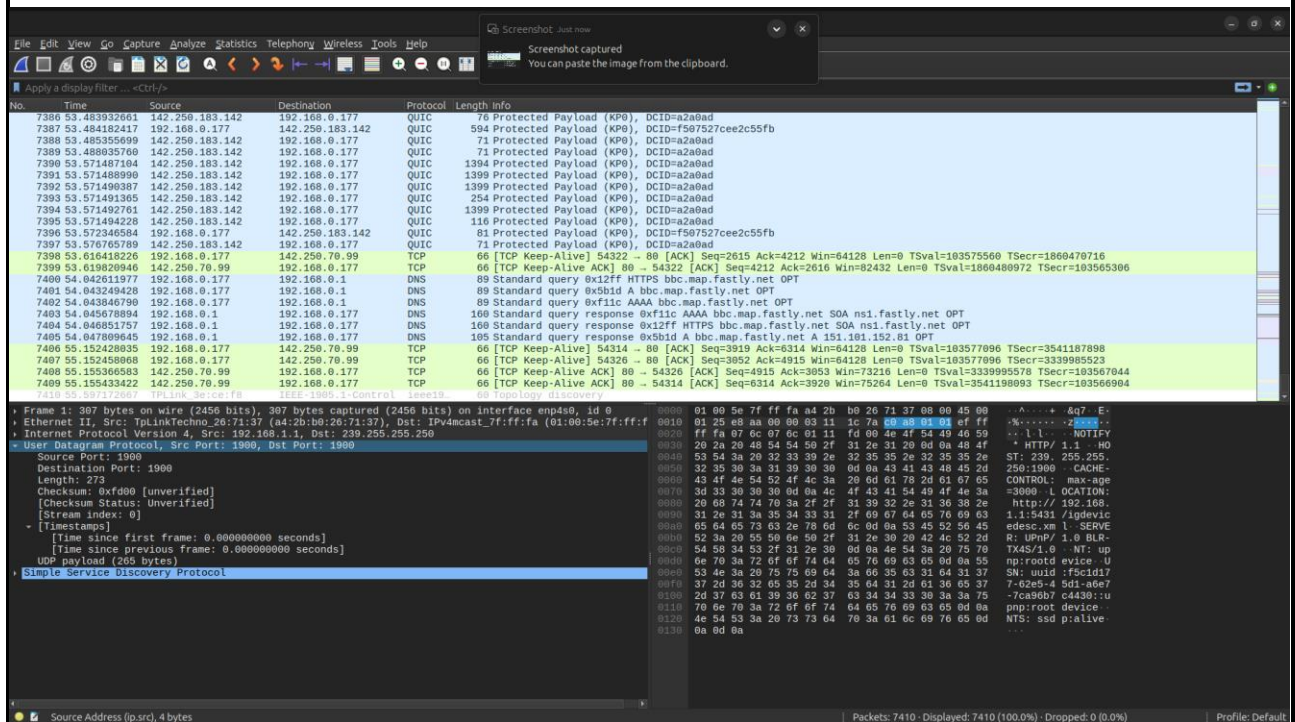
Ethernet Details

Screenshot of Wireshark interface showing network traffic analysis. The packet list pane displays various protocols including QUIC, TCP, and DNS. The packet details pane shows the structure of a packet, including Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Simple Service Discovery Protocol (SSDP). The packet bytes pane shows the raw data of the packet.

Source Address (ip.src), 4 bytes

Packets: 7410 - Displayed: 7410 (100.0%) - Dropped: 0 (0.0%)

Simple Service Discovery Protocol

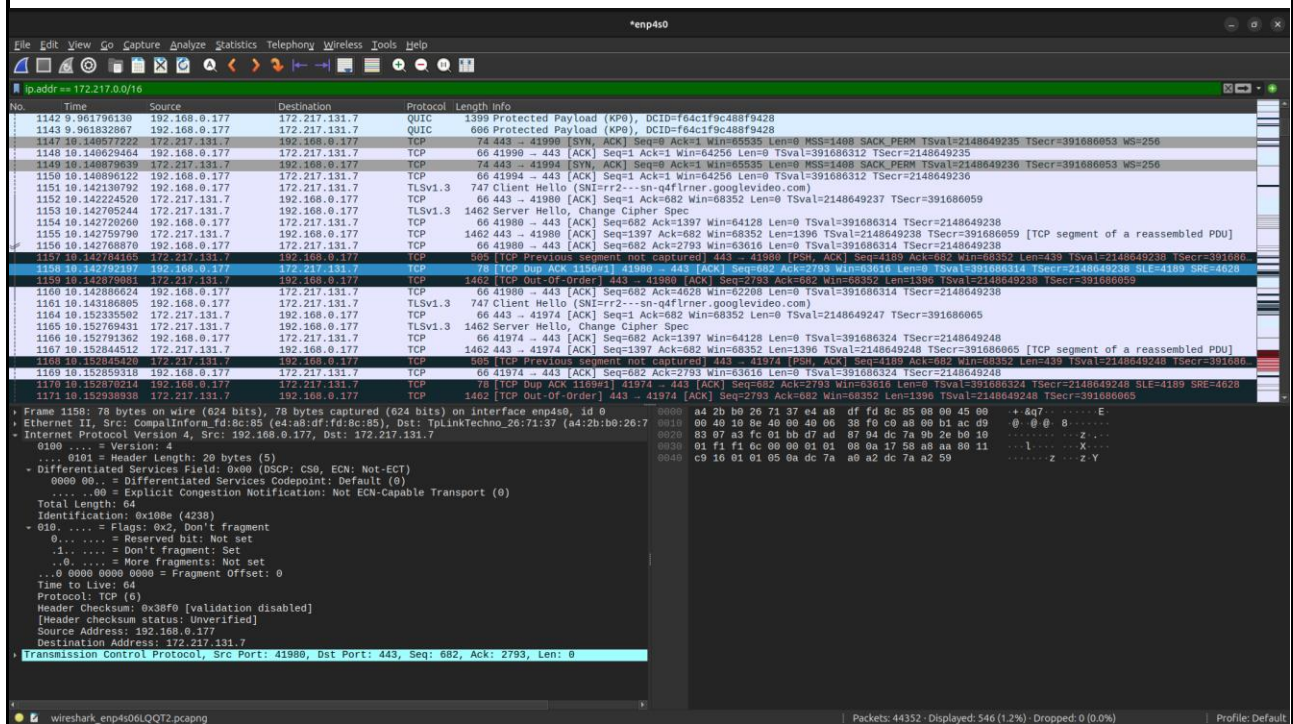


User Datagram Protocol

Questions on the Given Images:

- **Is the frame an outgoing or an incoming frame?**
The given frame is an incoming frame as the Destination MAC Address matches the computers MAC Address.
- **Source IP address of the network-layer header in the frame:**
The Source Address is 192.168.1.1
- **Destination IP address of the network-layer header in the frame:**
The Destination Address is 239.255.255.250
- **Total number of bytes in the whole frame:**
The number of bytes in the whole frame is 307 bytes.
- **Number of bytes in the Ethernet (data-link layer) header:**
The number of bytes in the Ethernet header is 14 bytes.
- **Number of bytes in the IP header:**
The number of bytes in the IP header is 20 bytes.
- **Number of bytes in the TCP header:**
The number of bytes in the TCP header is 20 bytes.
- **Total bytes in the message (at the application layer):**
The total bytes in the message is 273 bytes.

- Capture packets if you were using youtube (TCP contains youtube or ipaddress of youtube)



Watching a Youtube Video while having Wireshark enabled.

When capturing and analyzing YouTube traffic using Wireshark, I observed the following:

- 1. DNS Resolution:** My computer requested the IP address for `youtube.com`, and the DNS server responded with YouTube's IP addresses, allowing the connection to initiate.
- 2. TCP Handshake:** A three-way TCP handshake was established with YouTube's server, confirming a reliable connection for data transfer.
- 3. TLS Handshake:** The TLS handshake followed, securing the connection through encryption, ensuring that the video data was transmitted securely.
- 4. Video Data Transfer:** I observed large, continuous TCP packets representing the streaming video data. Although encrypted, the size and frequency of the packets indicated active video streaming.
- 5. Session Characteristics:** The streaming session was marked by consistent data flow and occasional retransmissions, reflecting the nature of video streaming traffic.

These results provide a clear view of how YouTube streams content, from DNS resolution to encrypted data transfer, ensuring secure and reliable video delivery.

- Login through some unsecure website and capture the password

The image displays a Wireshark packet capture of an HTTP session. The packet list on the left shows a GET request (No. 1049) to /SignIn.aspx?DXCache=2ffdcac7-e1dc-48f8-9b22-312fd1b7c104 HTTP/1.1. The packet details pane shows the request headers, including User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0 and Accept-Language: en-US,en;q=0.5. The packet bytes pane shows the raw data of the request, including the POST body: { "Username": "Dwayne", "Password": "1234567890" }. The packet list also shows a 302 Found response (No. 1050) to /CommonTool/Images/Fb_150.png HTTP/1.1.

Creating an Account in an Unsafe Website.

The image displays a Wireshark packet capture of an HTTP session. The packet list on the left shows a POST request (No. 1049) to /Login.aspx?DXCache=2ffdcac7-e1dc-48f8-9b22-312fd1b7c104 HTTP/1.1. The packet details pane shows the request headers, including User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0 and Accept-Language: en-US,en;q=0.5. The packet bytes pane shows the raw data of the request, including the POST body: { "Username": "Dwayne", "Password": "1234567890" }. The packet list also shows a 302 Found response (No. 1050) to /CommonTool/Images/Fb_150.png HTTP/1.1.

Logging into that Account on the Same Website.

	<p>When logging into an unsecure website and capturing the traffic with Wireshark, here's what I observed:</p> <p>1. Unencrypted Login Data: Upon logging into the unsecure website (using HTTP), I captured the traffic and found that the login credentials, including the username and password, were transmitted in plain text.</p> <p>2. HTTP POST Request: The credentials were included in an HTTP POST request, which was sent from my computer to the server. Since the connection was not encrypted, the data was easily visible in the packet capture.</p> <p>3. Captured Password: By inspecting the HTTP packet, I was able to see the password and other form data directly in the packet details under the `Form Data` or `Parameters` section.</p> <p>This result highlights the risks of using unsecure websites, as sensitive information like passwords can be intercepted easily by anyone capturing the network traffic.</p>
REFERENCES	<ul style="list-style-type: none">• A complete guide to Wireshark- https://www.javatpoint.com/wireshark• https://www.youtube.com/watch?v=TkCSr30UojM