

EXP:12**Capture, Save, and Analyze Network Traffic using Wireshark****DATE: 13/10/25****Aim:**

To use Wireshark to capture, save, and analyze network traffic over multiple protocols (TCP, UDP, IP, HTTP, ARP, DHCP, ICMP, DNS).

Introduction:

Wireshark is a packet analyzer used for network troubleshooting and protocol analysis.

Algorithm (Procedure):

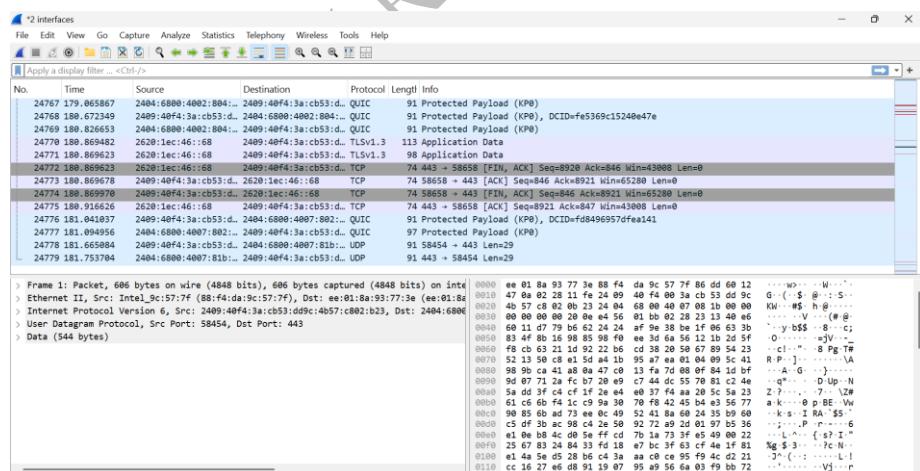
- Launch Wireshark and select the active network interface.
- Start capturing packets.
- Perform network activities (e.g., browsing, ping, DHCP request).
- Stop the capture and save the file.
- Use filters to isolate TCP, UDP, HTTP, ARP, DHCP, ICMP, and DNS traffic.
- Analyze details and packet contents.

Result:

Traffic captured and analyzed; protocol communications are observed for learning or troubleshooting.

Output:**1. Capture or Open a Packet Capture File**

- Start by capturing live traffic via your network interface or open an existing .pcap file.
- Click **File > Open** to load a saved capture file.

**Follow TCP streams**

Wireshark · Follow TCP Stream (tcp.stream eq 381) · 2 interfaces

```
g.....6....u.'.x.V...,X...@V .X.`05?M..0.g..V....0$....  
`Na.(....+.0./.$#.('.  
. ....=.<5./...R....  
..  
g.live.com.....+.....  
.....#...  
.....h2.http/1.1.3..... 0.....6.U/l...Z.},.....%zo...A.EX.....E...Q...S.t.86<..N  
.>>.w.(...o...Zg%.B....  
0.....A.E.....9i  
..\\.<[.er...lg0c.t".N&....JL...FD.r...E...l"....8.[.w....r_h.h.nf.ue$.3!..A.}.1.....-....  
...../..w.Y?..d...A.:3t>.....c...X.`05?M..0.g..V....0$....  
`Na....0.+....3.E..A.....!.....V@P;t~...G.}{.ruf...`\\|....xD  
...*....)+1$;.....$.d.....>.z.7....1....\....k.V....&.....{@.7,.r....K.#P.?F$...nM....S....(b  
.).s.-+...P....Cn...I.#....V....E....F....-/..W.O....r.....?....S  
9....<..9s).]2...Ca{[.g....!])...  
r{....4.GF.....j....D....Z.K.c.....~.I....d.E....r.X.R.....?....p.....E>I!LZ.j.....G(I....V...>M]%bj...  
....m.T3....KNG.E....`53xh....8.R.FU.O$....P.....X+....X....I.y....5F....7.N....o.M....$a.a.:3+  
C.J....&....t%....d....~w....H....u%).dz=Fd....k.$....@.T....<....R....x'....  
....f....>G9r.Y....Q.7k....8.f....qX.q....P....6y....g....B....7.A.=....O.;....S.a....j.W.pM....&....#....C9o  
N....t....x....8....s....a....].1.yo...dd....3....a....&Z....d....15....V....D.y....  
....n....8'.GcN{....=The.2...."....W....+....k....L....t....5/....7....].J....u....q.e....).o....b....J%....f....L....@....Q....>N....  
....y....H.h....7.H'....K,...]$....0....)%....v....\....Dt....hc....$....2p....p....$....2^....c....\....V....1....t....  
B.r....Jd....z....ST....Z'....Ex....#....W....B-h|....IM7v.gT....^....0  
!<....LF..../.v....X7....4....V....u....x....6x.D.%....".WE....(#....j....lg....!kW  
....}n.sL....O.z....Pl....D....[....G....!....P....E#....d....q....L....c....u....x....f....  
C....~....'....f....n....=....c....$....L....Q....A....rd....|.Y....\....2....).V....]k9....z}E!  
....Lh1....F....D....u....p....Z....j...._1....%....W....O....  
....o....(....PY....{....[....C....1....?....V....H....!.z...._....;....E...._....z....@....PE....Y....  
....M....q....j....2....1....%....W....O....  
....+....o....&....e{....*....E....d%....D....t....+....Q....!....0n5....c3....R....!5....F....,-....w....dv....~....v....*....?....+....6=X....(^....Ks  
....K....o....#....$.Q;....eW....?....)....d....6....S....x....=....5'....$....9....|.....Q....R....r...._....?....a....).~..../[....  
....g....Sje....L....Q....0....Q....I....\....S0....(....;....N.R....`....a....S....I....iee....t9Z....&....A....+....\....v....sP....<....1`....
```

Protocol Hierarchy:

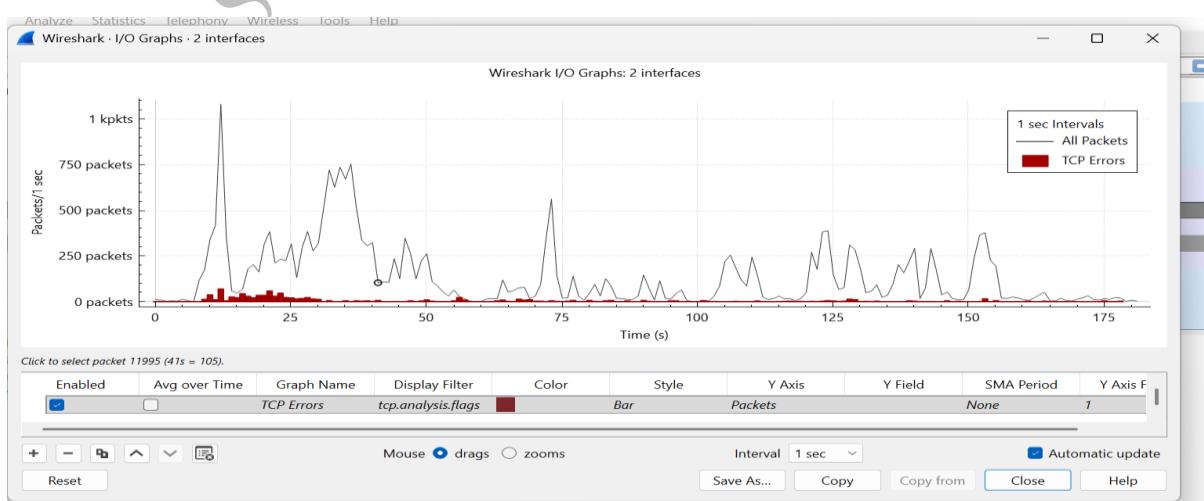
Shows a breakdown of protocols seen in the capture with counts and percentages.

Go to Statistics > Protocol Hierarchy.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	41	100.0	13365	1174	0	0	0	41
Ethernet	100.0	41	4.3	574	50	0	0	0	41
Internet Protocol Version 6	100.0	41	12.3	1640	144	0	0	0	41
Transmission Control Protocol	100.0	41	6.6	880	77	25	560	49	41
Transport Layer Security	39.0	16	81.2	10857	954	16	4052	356	17

I/O GRAPHS;

Use statistics > I/O graphs to visualize traffic rate and identify spikes or drops over time



Endpoints:

Lists all endpoints (IPs, MACs) involved in the capture and summarizes traffic per endpoint.
Go to Statistics > Endpoints.

The screenshot shows the Wireshark interface with the title "Endpoints · 2 interfaces". On the left, there's a panel titled "Endpoint Settings" with checkboxes for "Name resolution", "Display raw data", "Hide aggregated", and "Limit to display filter". Below it are buttons for "Copy" and "Map". A list of protocols is shown, with "Ethernet" selected. At the bottom is a "Filter list for specific type" input field. The main area displays a table of endpoint statistics:

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
01:00:5e:00:00:fb	7	721 bytes	0	0 bytes	7	721 bytes
01:00:5e:7fff:fa	3	501 bytes	0	0 bytes	3	501 bytes
33:33:00:00:00:fb	8	984 bytes	0	0 bytes	8	984 bytes
88:4f:da:9c:57:7f	24,761	18 MB	8,813	3 MB	15,948	15 MB
b8:c7:4a:60:7e:41	18	2 kB	18	2 kB	0	0 bytes
ee:01:8a:93:77:3e	24,761	18 MB	15,948	15 MB	8,813	3 MB

Flow: Depicts directional communication between network hosts, showing data exchanges in sequential context for analysis

The screenshot shows the Wireshark interface with the title "Flow · shark.pcapng". It displays a flow between two hosts: 172.16.53.133 and 34.104.35.123. The flow details show a sequence of HTTP requests and responses. The "Comment" column provides detailed descriptions of each packet. The flow is visualized with green boxes for source and destination hosts, and a pink box for the broadcast channel. The "Comment" column contains the following text:

- HTTP: GET /edged/dfifgen-puffin/hnfpim/mhhg/eaddgfemjhof
- HTTP: HTTP/1.1 416 Requested range not satisfiable
- HTTP: HEAD /edged/dfifgen-puffin/hnfpim/mhhg/eaddgfemjh
- HTTP: HTTP/1.1 200 OK
- HTTP: GET /edged/dfifgen-puffin/hnfpim/mhhg/eaddgfemjhof
- HTTP: HTTP/1.1 416 Requested range not satisfiable
- HTTP: HEAD /edged/dfifgen-puffin/hnfpim/mhhg/eaddgfemjh
- HTTP: HTTP/1.1 200 OK
- ARP: Who has 172.16.53.49? Tell 172.16.53.56
- HTTP: GET /edged/dfifgen-puffin/hnfpim/mhhg/eaddgfemjhof
- HTTP: HTTP/1.1 416 Requested range not satisfiable
- HTTP: HEAD /edged/dfifgen-puffin/hnfpim/mhhg/eaddgfemjh
- HTTP: HTTP/1.1 200 OK
- ARP: Who has 172.16.52.14? Tell 172.16.52.126
- HTTP: GET /edged/dfifgen-puffin/hnfpim/mhhg/eaddgfemjhof
- HTTP: HTTP/1.1 416 Requested range not satisfiable

Display Filters: Enable precise viewing of packets by protocol fields, source/destination, or conditions (e.g., `tcp.port == 80` or `frame.len > 500`).

Wireshark - Display Filters

Filter Name	Filter Expression
Ethernet address 00:00:5e:00:53:00	eth.addr == 00:00:5e:00:53:00
Ethernet type 0x0806 (ARP)	eth.type == 0x0806
Ethernet broadcast	eth.addr == ff:ff:ff:ff:ff:ff
No ARP	not arp
IPv4 only	ip
IPv4 address 192.0.2.1	ip.addr == 192.0.2.1
IPv4 address isn't 192.0.2.1	ip.addr != 192.0.2.1
IPv6 only	ipv6
IPv6 address 2001:db8::1	ipv6.addr == 2001:db8::1
TCP only	tcp
UDP only	udp
Non-DNS port	!(udp.port == 53 tcp.port == 53)
TCP or UDP port is 80 (HTTP)	tcp.port == 80 udp.port == 80
HTTP	http
No ARP and no DNS	not arp and not dns
Non-HTTP and non-SMTP to/from 192.0.2.1	ip.addr == 192.0.2.1 and tcp.port not in {80, 25}

UDP: A connectionless transport protocol providing fast but unreliable message delivery, ideal for real-time or broadcast communications.

116

	Source	Destination	Protocol	Len	Info
1	udp				
2	udp[0] 0708 2404:6800:4002:828... 2409:40f4:3a:cb53:d.. QUIC			96	Protected Payload (K0)
2	udp[0] 9897 2404:6800:4002:828... 2409:40f4:3a:cb53:d.. QUIC			314	Protected Payload (K0)
2	udp[0] 3668 2404:6800:4002:828... 2409:40f4:3a:cb53:d.. QUIC			89	Protected Payload (K0)
24764 178.954130 2409:40f4:3a:cb53:d.. 2404:6800:4002:828.. QUIC				102	Protected Payload (K0), DCID=e0c6443b04d41934
24765 178.969965 2409:40f4:3a:cb53:d.. 2404:6800:4002:804.. QUIC				91	Protected Payload (K0), DCID=fe5369c15240e47e
24766 179.050618 2404:6800:4002:828... 2409:40f4:3a:cb53:d.. QUIC				96	Protected Payload (K0)
24767 179.065867 2404:6800:4002:804... 2409:40f4:3a:cb53:d.. QUIC				91	Protected Payload (K0)
24768 188.672349 2409:40f4:3a:cb53:d.. 2404:6800:4002:804.. QUIC				91	Protected Payload (K0), DCID=fe5369c15240e47e
24769 188.826653 2404:6800:4002:804... 2409:40f4:3a:cb53:d.. QUIC				91	Protected Payload (K0)
24776 181.041037 2409:40f4:3a:cb53:d.. 2404:6800:4007:802.. QUIC				91	Protected Payload (K0), DCID=fd8496957dfea141
24777 181.094956 2404:6800:4007:802... 2409:40f4:3a:cb53:d.. QUIC				97	Protected Payload (K0)
24778 181.665084 2409:40f4:3a:cb53:d.. 2404:6800:4007:81b.. UDP				91	58454 + 443 Len=29
24779 181.753704 2404:6800:4007:81b... 2409:40f4:3a:cb53:d.. UDP				91	443 + 58454 Len=29

TCP: A connection-oriented transport protocol that ensures reliable data delivery through mechanisms like acknowledgments and retransmissions.

116

*2 interfaces					
No.	Time	Source	Destination	Protocol	Length Info
24767 179.065867 2404:6800:4002:804... 2409:40f4:3a:cb53:d.. QUIC				91	Protected Payload (K0)
24768 188.672349 2409:40f4:3a:cb53:d.. 2404:6800:4002:804.. QUIC				91	Protected Payload (K0), DCID=fe5369c15240e47e
24769 188.826653 2404:6800:4002:804... 2409:40f4:3a:cb53:d.. QUIC				91	Protected Payload (K0)
24770 188.869482 2620:1ec:46:68 2409:40f4:3a:cb53:d.. TLSv1.3				113	Application Data
24771 180.869623 2620:1ec:46:68 2409:40f4:3a:cb53:d.. TLSv1.3				98	Application Data
24772 180.869623 2620:1ec:46:68 2409:40f4:3a:cb53:d.. TCP				74	443 + 58658 [FIN, ACK] Seq=8920 Ack=846 Win=43008 Len=0
24773 180.869678 2409:40f4:3a:cb53:d.. 2620:1ec:46:68 TCP				74	58658 + 443 [ACK] Seq=846 Ack=8921 Win=65280 Len=0
24774 180.869970 2409:40f4:3a:cb53:d.. 2620:1ec:46:68 TCP				74	58658 + 443 [FIN, ACK] Seq=846 Ack=8921 Win=65280 Len=0
24775 180.916626 2620:1ec:46:68 2409:40f4:3a:cb53:d.. TCP				74	443 + 58658 [ACK] Seq=8921 Ack=847 Win=43008 Len=0
24776 181.041037 2409:40f4:3a:cb53:d.. 2404:6800:4007:802.. QUIC				91	Protected Payload (K0), DCID=fd8496957dfea141
24777 181.094956 2404:6800:4007:802... 2409:40f4:3a:cb53:d.. QUIC				97	Protected Payload (K0)
24778 181.665084 2409:40f4:3a:cb53:d.. 2404:6800:4007:81b.. UDP				91	58454 + 443 Len=29
24779 181.753704 2404:6800:4007:81b... 2409:40f4:3a:cb53:d.. UDP				91	443 + 58454 Len=29

> Frame 1: Packet, 606 bytes on wire (4848 bits), 606 bytes captured (4848 bits) on interface
> Ethernet II, Src: Intel_9c:57:7f (88:f4:da:9c:57:7f), Dst: ee:01:8a:93:77:3e (ee:01:8a:93:77:3e)
> Internet Protocol Version 6, Src: 2409:40f4:3a:cb53:dd9c:4b57:c082:b23, Dst: 2404:6800:
> User Datagram Protocol, Src Port: 58454, Dst Port: 443
> Data (544 bytes)

```

0000  ee 01 8a 93 77 3e 88 f4 da 9c 57 f7 86 dd 60 12 ...w>... W...
0001  47 0a 02 28 11 fe 24 09 40 f4 00 3a cb 53 dd 9c G-(-$ @:-S-
0002  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 KW-#$ h@#-
0003  00 00 00 00 20 0e e4 56 01 bb 02 28 23 13 40 e6 .....V-#(@-
0004  4b 57 c8 02 00 23 24 04 68 00 40 07 08 1b 00 00 00 60 11 79 b0 62 24 24 af 9e 38 be 1f 06 63 3b `y b$< 8-c;
0005  83 4f 8b 16 98 85 98 f0 e3 6d 56 12 1b 2d 5f O-----J\`V-`-
0006  f8 cb 63 21 1d 92 22 b6 cd 38 28 58 67 89 54 23 c1-`-8 Pg T#
0007  52 13 50 c8 e1 5d a4 1b 95 a7 ea 01 04 09 5c 41 R P-]-----\A
0008  98 9b ca 41 a8 04 47 c8 13 fa 7d 08 0f 84 1d bf A-G-}-----
0009  0d 07 71 2a fc b7 20 e9 c7 44 55 70 81 c2 4e q-`D Up-N
0010  5a dd 3f c4 cf 1f 2e e4 a0 37 f4 aa 20 5c 5a 23 Z?...-7-Z#
0011  61 c6 6b f4 1c c9 9a 30 70 f8 42 45 b4 e3 56 77 a-k-`0 p BE-Vw
0012  98 85 6b ad 73 ee 0c 49 52 41 8a 60 24 35 b9 60 `k-s` I RA-$5`-
0013  c5 df 3b ac 98 c4 2e 50 92 72 a9 2d 01 97 b3 36 ;`P-r-`-6
0014  e1 0e b8 4c d0 5e ff cd 7b 1a 73 3f e5 49 00 22 L-`-{ s? I-
0015  25 67 83 24 84 33 fd 18 e7 bc 3f 63 cf 4e 1f 81 %g $ 3-`?c N-
0100  e1 4a 5e d5 28 b6 c4 3a aa c8 c9 95 f9 4c d2 21 J^(-:-L-!
0110  c1 16 27 e6 dd 91 07 e9 a5 6a 03 f9 b2 72 ..`Vj-r

```

Packets: 24779 - Dropped: 0 (0.0%) Profile: Default

Conversations:

Displays all conversations between IP addresses or MAC addresses. Useful for identifying

communication pairs.

Go to Statistics > Conversations.

The screenshot shows the 'Conversations' window in Wireshark. The table displays communication between two hosts (Address A and Address B) over four interfaces (Ethernet, IPv4, IPv6, TCP, UDP). The columns include: Conversation Settings, Address A, Address B, Packets, Bytes, Stream ID, Packets A → B, Bytes A → B, Packets B → A, Bytes B → A, Rel Start, Duration, Bits/s A → B, and Bits/s B → A. The table shows several entries, with the first entry being 88:ff:da:9c:57:7f ee:01:8a:93:77:3e.

Conversation Settings	Ethernet	IPv4	IPv6	TCP	UDP	Rel Start	Duration	Bits/s A → B	Bits/s B → A
<input type="checkbox"/> Name resolution	88:ff:da:9c:57:7f	ee:01:8a:93:77:3e	24,761	18 MB	0	8,813	3 MB	15,948	15 MB
<input type="checkbox"/> Absolute start time	b8:c7:4a:60:7e:41	01:00:5e:00:00:fb	7	721 bytes	1	7	721 bytes	0	0 bytes
<input type="checkbox"/> Display raw data	b8:c7:4a:60:7e:41	01:00:5e:7ffffa	3	501 bytes	3	3	501 bytes	0	0 bytes
<input type="checkbox"/> Limit to display filter	b8:c7:4a:60:7e:41	33:33:00:00:00:fb	8	984 bytes	2	8	984 bytes	0	0 bytes

Expert Information: Displays protocol anomalies, warnings, and potential issues detected during packet capture for network troubleshooting.

The screenshot shows the 'Expert Information' window in Wireshark. The table lists various protocol anomalies and warnings, grouped by severity (Warning, Note, Chat, Deprecated). The columns include: Severity, Summary, Group, Protocol, and Count. The table contains numerous entries, such as 'Warning: TCP window specified by the receiver is now completely full' and 'Note: ACK to a TCP keep-alive segment'.

Severity	Summary	Group	Protocol	Count
> Warning	TCP window specified by the receiver is now completely full	Sequence	TCP	1
> Warning	Ignored Unknown Record	Protocol	TLS	1
> Warning	DNS query retransmission	Protocol	DNS	1
> Warning	DNS response missing	Protocol	mDNS	1
> Warning	DNS response missing	Protocol	DNS	1
> Warning	Connection reset (RST)	Sequence	TCP	1
> Warning	This frame is a (suspected) out-of-order segment	Sequence	TCP	1
> Warning	Previous segment(s) not captured (common at capture start)	Sequence	TCP	1
> Warning	Failed to decrypt handshake	Decryption	QUIC	1
> Warning	D-SACK Sequence	Sequence	TCP	1
> Note	ACK to a TCP keep-alive segment	Sequence	TCP	1
> Note	TCP keep-alive segment	Sequence	TCP	1
> Note	A new tcp session is started with the same ports as an earlier session in t...	Sequence	TCP	1
> Note	The acknowledgment number field is nonzero while the ACK flag is not set	Protocol	TCP	1
> Note	This frame is a (suspected) fast retransmission	Sequence	TCP	1
> Note	This frame undergoes the connection closing	Sequence	TCP	1
> Note	This session reuses previously negotiated keys (Session resumption)	Sequence	TLS	1
> Note	Duplicate ACK	Sequence	TCP	1
> Note	This frame is a (suspected) spurious retransmission	Sequence	TCP	1
> Note	Partial Acknowledgement of a segment	Sequence	TCP	1
> Note	Ambiguous ACK following Karn's definition	Sequence	TCP	1
> Note	This frame is a (suspected) retransmission	Sequence	TCP	1
> Note	This QUIC frame has a reused stream offset (retransmission?)	Sequence	QUIC	1
> Note	This packet's length exceeds MSS (common with TSO or incomplete con...	Protocol	TCP	1
> Note	This frame initiates the connection closing	Sequence	TCP	1
> Chat	TCP window update	Sequence	TCP	1
> Chat	Connection establish acknowledge (SYN+ACK)	Sequence	TCP	1
> Chat	This legacy_version field MUST be ignored. The supported_versions exte...	Deprecated	TLS	1

No display filter set.

Limit to Display Filter Group by summary Search: Show...

Result

Hence using windows Wireshark, we captured the packets and executed the functions.