Sayali Dilip Deshmukh 1001966628

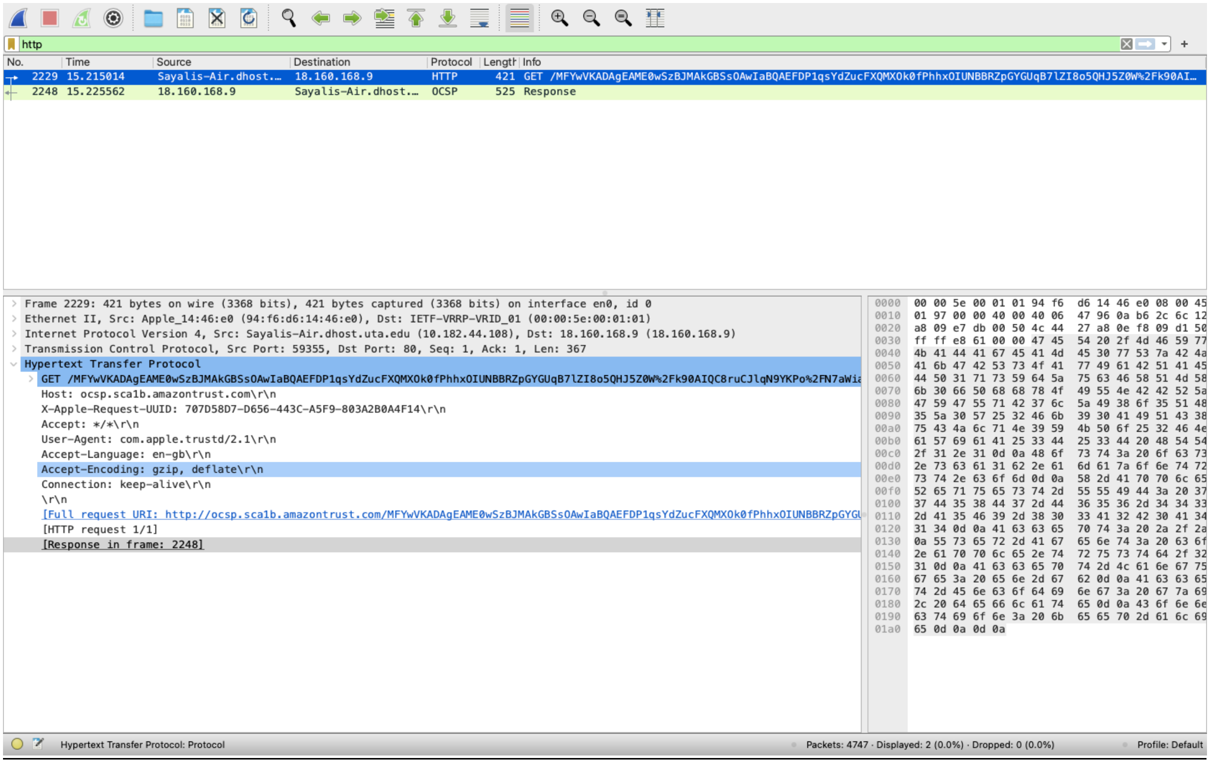
ASSIGNMENT NO:01

Q1)Briefly explain HTTP protocol. Demonstrate a comparison of HTTP request and response messages as learned from the textbook with the Wireshark output.
ANS:
HTTP protocol:
1)Hypertext Transfer Protocol is what it stands for.
2)It is a set of explicit guidelines for communication between a client (the network resource asking for data or services) and a server (the resource that receives and responds to the request).
3) The guidelines for resource requests and answers between web clients and servers are laid out in the HTTP protocol.
4) In the seven-layer OSI networking model, HTTP is an application layer protocol that standardizes communication between computing or telecommunications systems, regardless of underlying internal structure and technology.
5)An multinational group known as the Internet Engineering Task Force is now in charge of the defining and continuous development of this protocol (IETF).
Comparison of HTTP request and response messages:

HTTP request:
1)HTTP requests are messages that the client sends to the server to start an action. Three components make up their starting point:
2)An HTTP method is a verb or noun (such as GET, PUT, or POST) that specifies the action to be taken. For instance, the terms GET and POST denote when a resource should be fetched from and when data should be posted to a server, respectively (creating or modifying a resource, or generating a temporary document to send back).
The request context often identifies the request target, which is either a URL or the absolute path of the protocol, port, and domain. This request target's format differs depending on the HTTP method.
When connected to a proxy, GET typically uses the absolute form of a URL, often known as the entire URL.
The domain name and, optionally, the port make up the authority form, which is a part of a URL. When creating an HTTP tunnel, it is only used in conjunction with CONNECT. JOIN developer.mozilla.org at port 80 HTTP/1.1
A basic asterisk ('*') is used with OPTIONS to represent the server as a whole in the asterisk form.
3)The HTTP version serves as a signal of the anticipated version to use for the response by defining the structure of the remaining message.

HTTP response:
1)The status line, which appears at the beginning of an HTTP response, includes the following data:
2)often HTTP/1.1, the protocol version, a code that represents the request's success or failure. Status codes 200, 404, or 302 are frequently used.
3)an update message. A succinct, primarily informative text explanation of the status code that aids in understanding the HTTP message by humans.

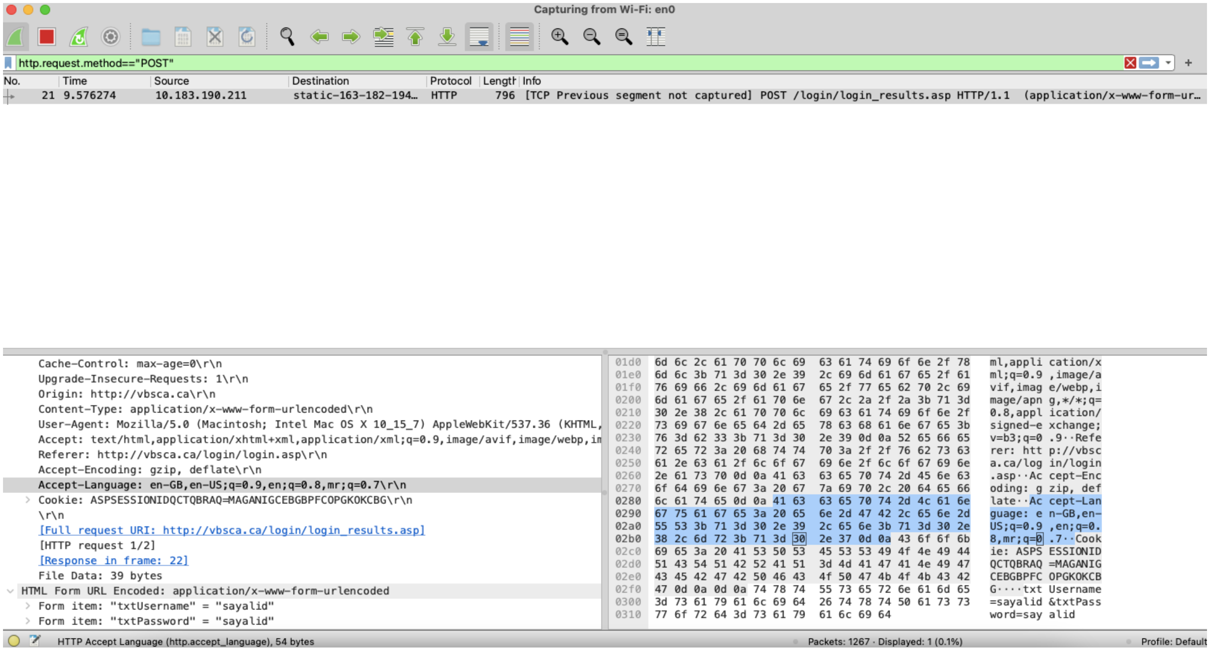
Screenshot from Wireshark:



Q2)Go to any unsecured website (Example: http://vbsca.ca/login/login.asp). You will be prompted to enter login and password. Once you are done with that, use Wireshark to check if the password is encrypted or not. You must be able to find the entered username and password as plain text among the packets exchanged shown on Wireshark. Provide a screenshot of the username and password seen on Wireshark.

ANS:

Screenshot of the username and password seen on Wireshark.



Q3)Using Wireshark demonstrate TCP three-way handshake.

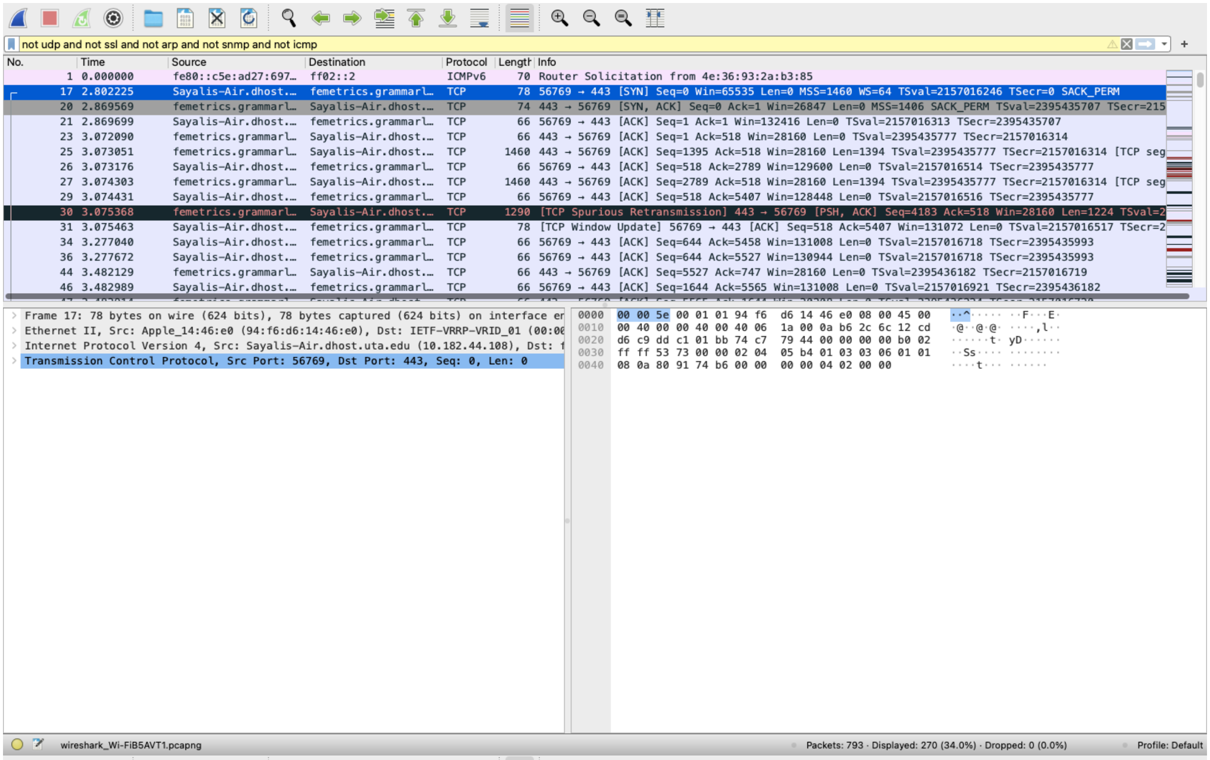
ANS:

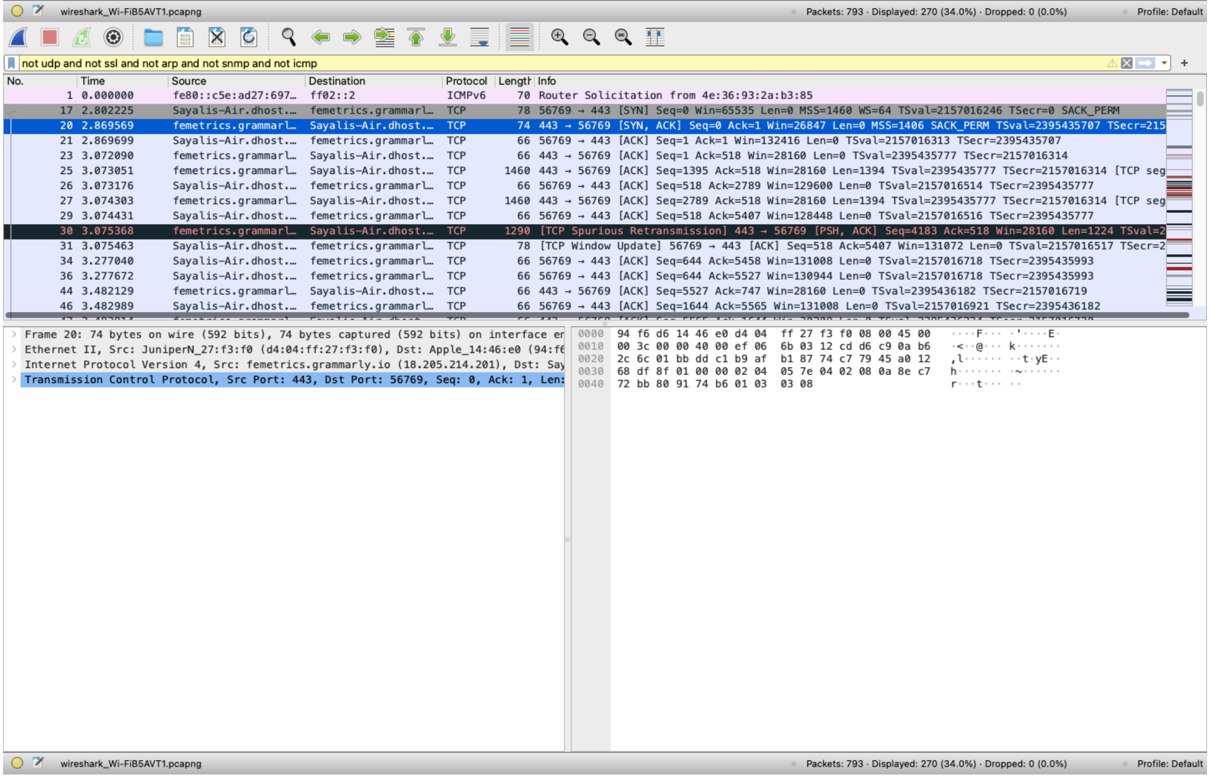
TCP three-way handshake

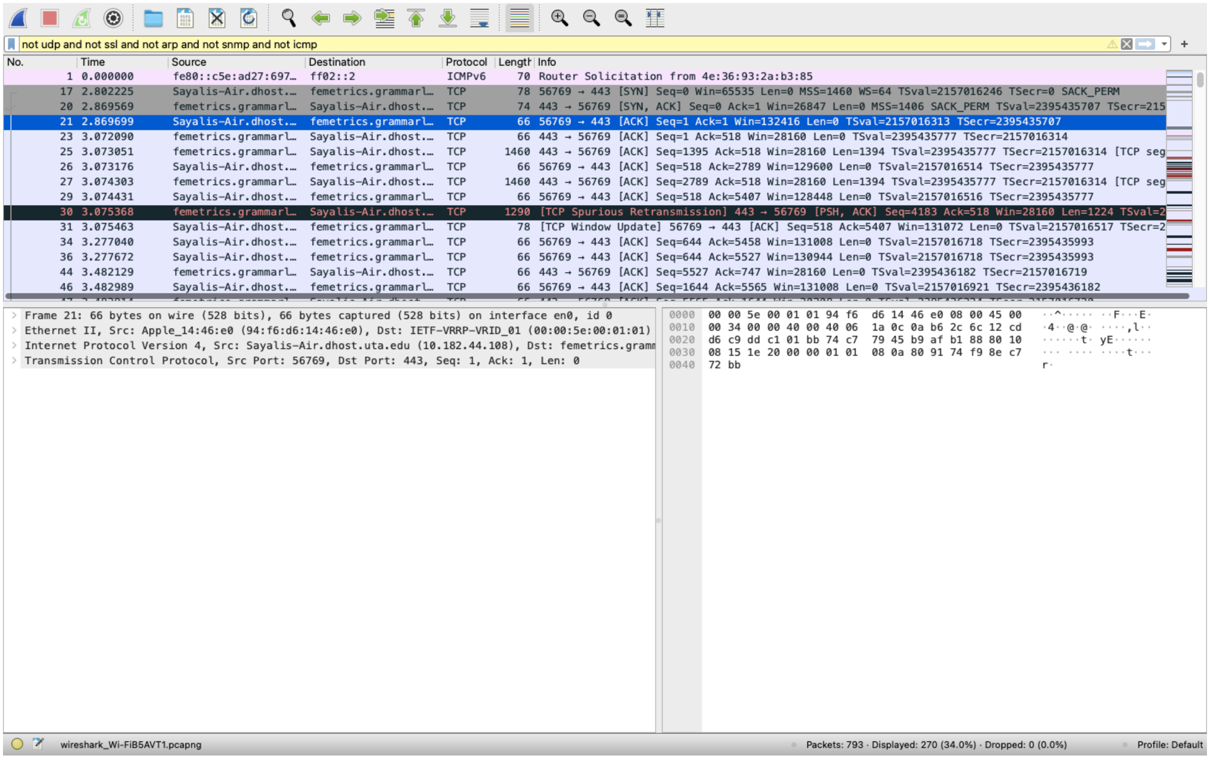
1)A TCP/IP network connection procedure known as the 3-Way handshake links the server and client. Both the client and the server must exchange synchronization and acknowledgment packets before the actual data transmission can start.

2)Prior to data transmission, the 3-way handshake process is intended to allow both communication ends to simultaneously establish and negotiate the network TCP socket connection specifications. It enables the simultaneous transport of a large number of TCP socket connections in both directions.

Screenshots:


not udp and not ssl and not arp and not snmp and not icmp
No. Time Source Destination Protocol Length Info
1 0.000000 fe80::c5e:ad27:697… ff02::2 ICMPv6 70 Router Solicitation from 4e:36:93:2a:b3:85
17 2.802225 Sayalis-Air.dhost.… femetrics.grammarl… TCP 78 56769 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2157016246 TSecr=0 SACK_PERM
20 2.869569 femetrics.grammarl… Sayalis-Air.dhost.… TCP 74 443 → 56769 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1406 SACK_PERM TSval=2395435707 TSecr=215
21 2.869699 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=1 Ack=1 Win=132416 Len=0 TSval=2157016313 TSecr=2395435707
23 3.072090 femetrics.grammarl… Sayalis-Air.dhost.… TCP 66 443 → 56769 [ACK] Seq=1 Ack=518 Win=28160 Len=0 TSval=2395435777 TSecr=2157016314
25 3.073051 femetrics.grammarl… Sayalis-Air.dhost.… TCP 1460 443 → 56769 [ACK] Seq=1395 Ack=518 Win=28160 Len=1394 TSval=2395435777 TSecr=2157016314 [TCP seg
26 3.073176 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=518 Ack=2789 Win=129600 Len=0 TSval=2157016514 TSecr=2395435777
27 3.074303 femetrics.grammarl… Sayalis-Air.dhost.… TCP 1460 443 → 56769 [ACK] Seq=2789 Ack=518 Win=28160 Len=1394 TSval=2395435777 TSecr=2157016314 [TCP seg
29 3.074431 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=518 Ack=5407 Win=128448 Len=0 TSval=2157016516 TSecr=2395435777
30 3.075368 femetrics.grammarl… Sayalis-Air.dhost.… TCP 1290 [TCP Spurious Retransmission] 443 → 56769 [PSH, ACK] Seq=4183 Ack=518 Win=28160 Len=1224 TSval=2
31 3.075463 Sayalis-Air.dhost.… femetrics.grammarl… TCP 78 [TCP Window Update] 56769 → 443 [ACK] Seq=518 Ack=5407 Win=131072 Len=0 TSval=2157016517 TSecr=2
34 3.277040 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=644 Ack=5458 Win=131008 Len=0 TSval=2157016718 TSecr=2395435993
36 3.277672 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=644 Ack=5527 Win=130944 Len=0 TSval=2157016718 TSecr=2395435993
44 3.482129 femetrics.grammarl… Sayalis-Air.dhost.… TCP 66 443 → 56769 [ACK] Seq=5527 Ack=747 Win=28160 Len=0 TSval=2395436182 TSecr=2157016719
46 3.482989 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=1644 Ack=5565 Win=131008 Len=0 TSval=2157016921 TSecr=2395436182
Frame 17: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface en
Ethernet II, Src: Apple_14:46:e0 (94:f6:d6:14:46:e0), Dst: IETF-VRRP-VRID_01 (00:00
Internet Protocol Version 4, Src: Sayalis-Air.dhost.uta.edu (10.182.44.108), Dst: f
Transmission Control Protocol, Src Port: 56769, Dst Port: 443, Seq: 0, Len: 0
0000 00 00 5e 00 01 01 94 f6 d6 14 46 e0 08 00 45 00
0010 00 40 00 00 40 00 40 06 1a 00 0a b6 2c 6c 12 cd
0020 d6 c9 dd c1 01 bb 74 c7 79 44 00 00 00 00 b0 02
0030 ff ff 53 73 00 00 02 04 05 b4 01 03 03 06 01 01
0040 08 0a 80 91 74 b6 00 00 00 00 04 02 00 00
wireshark_Wi-FiB5AVT1.pcapng
Packets: 793 · Displayed: 270 (34.0%) · Dropped: 0 (0.0%)
Profile: Default

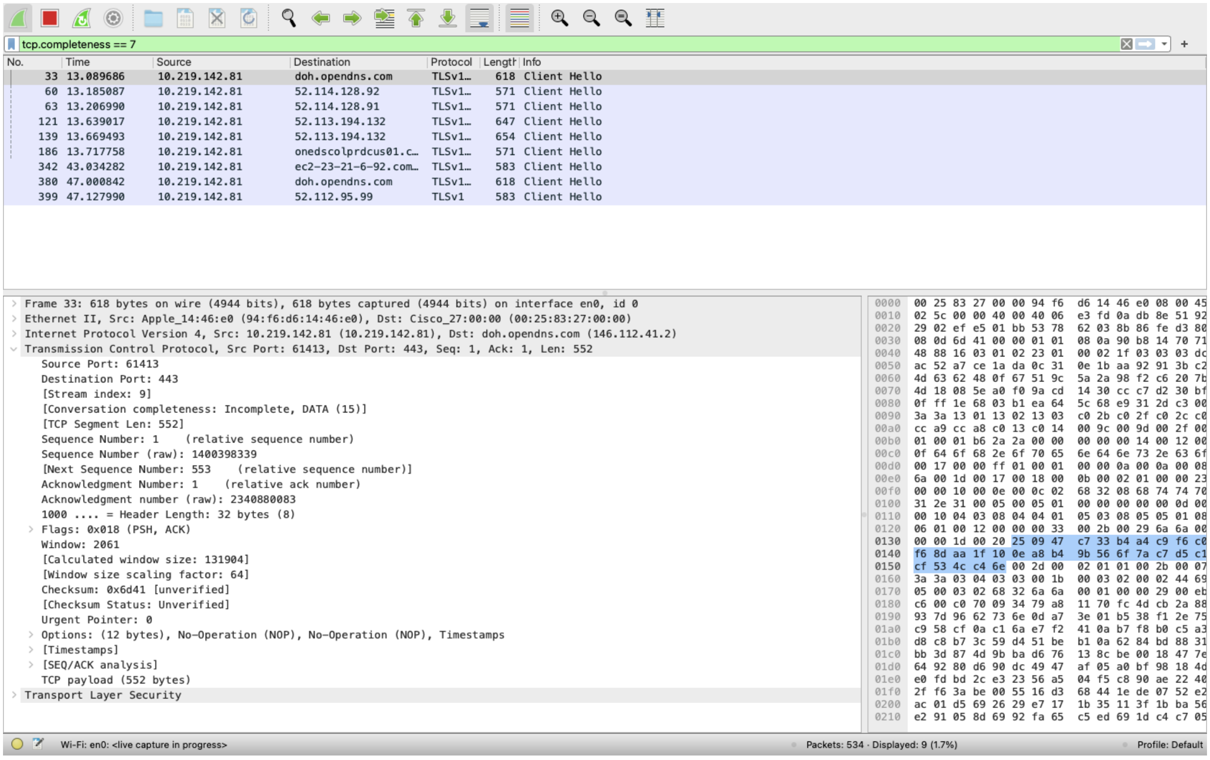

not udp and not ssl and not arp and not snmp and not icmp
No. Time Source Destination Protocol Length Info
1 0.000000 fe80::c5e:ad27:697… ff02::2 ICMPv6 70 Router Solicitation from 4e:36:93:2a:b3:85
17 2.802225 Sayalis-Air.dhost.… femetrics.grammarl… TCP 78 56769 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2157016246 TSecr=0 SACK_PERM
20 2.869569 femetrics.grammarl… Sayalis-Air.dhost.… TCP 74 443 → 56769 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1406 SACK_PERM TSval=2395435707 TSecr=215
21 2.869699 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=1 Ack=1 Win=132416 Len=0 TSval=2157016313 TSecr=2395435707
23 3.072090 femetrics.grammarl… Sayalis-Air.dhost.… TCP 66 443 → 56769 [ACK] Seq=1 Ack=518 Win=28160 Len=0 TSval=2395435777 TSecr=2157016314
25 3.073051 femetrics.grammarl… Sayalis-Air.dhost.… TCP 1460 443 → 56769 [ACK] Seq=1395 Ack=518 Win=28160 Len=1394 TSval=2395435777 TSecr=2157016314 [TCP seg
26 3.073176 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=518 Ack=2789 Win=129600 Len=0 TSval=2157016514 TSecr=2395435777
27 3.074303 femetrics.grammarl… Sayalis-Air.dhost.… TCP 1460 443 → 56769 [ACK] Seq=2789 Ack=518 Win=28160 Len=1394 TSval=2395435777 TSecr=2157016314 [TCP seg
29 3.074431 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=518 Ack=5407 Win=128448 Len=0 TSval=2157016516 TSecr=2395435777
30 3.075368 femetrics.grammarl… Sayalis-Air.dhost.… TCP 1290 [TCP Spurious Retransmission] 443 → 56769 [PSH, ACK] Seq=4183 Ack=518 Win=28160 Len=1224 TSval=2
31 3.075463 Sayalis-Air.dhost.… femetrics.grammarl… TCP 78 [TCP Window Update] 56769 → 443 [ACK] Seq=518 Ack=5407 Win=131072 Len=0 TSval=2157016517 TSecr=2
34 3.277040 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=644 Ack=5458 Win=131008 Len=0 TSval=2157016718 TSecr=2395435993
36 3.277672 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=644 Ack=5527 Win=130944 Len=0 TSval=2157016718 TSecr=2395435993
44 3.482129 femetrics.grammarl… Sayalis-Air.dhost.… TCP 66 443 → 56769 [ACK] Seq=5527 Ack=747 Win=28160 Len=0 TSval=2395436182 TSecr=2157016719
46 3.482989 Sayalis-Air.dhost.… femetrics.grammarl… TCP 66 56769 → 443 [ACK] Seq=1644 Ack=5565 Win=131008 Len=0 TSval=2157016921 TSecr=2395436182
Frame 20: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en
Ethernet II, Src: JuniperN_27:f3:f0 (d4:04:ff:27:f3:f0), Dst: Apple_14:46:e0 (94:f6
Internet Protocol Version 4, Src: femetrics.grammarly.io (18.205.214.201), Dst: Say
Transmission Control Protocol, Src Port: 443, Dst Port: 56769, Seq: 0, Ack: 1, Len:
0000 94 f6 d6 14 46 e0 d4 04 ff 27 f3 f0 08 00 45 00
0010 00 3c 00 00 40 00 ef 06 6b 03 12 cd d6 c9 0a b6
0020 2c 6c 01 bb dd c1 b9 af b1 87 74 c7 79 45 a0 12
0030 68 df 8f 01 00 00 02 04 05 7e 04 02 08 0a 8e c7
0040 72 bb 80 91 74 b6 01 03 03 08
wireshark_Wi-FiB5AVT1.pcapng
Packets: 793 · Displayed: 270 (34.0%) · Dropped: 0 (0.0%)
Profile: Default



Q4)Explain the Wireshark output of following filters
ANS:
TCP interactions are considered complete when they have both the opening and closing handshakes, regardless of any data transfer.
For instance, the filter "tcp.completeness==7" will detect a conversation that just involves a three-way handshake
The lengthier filter will identify a conversation that also involves data transfer
FIN or RST packets, or even both, can be used to signify the closing of a connection 'tcp.completeness==31.

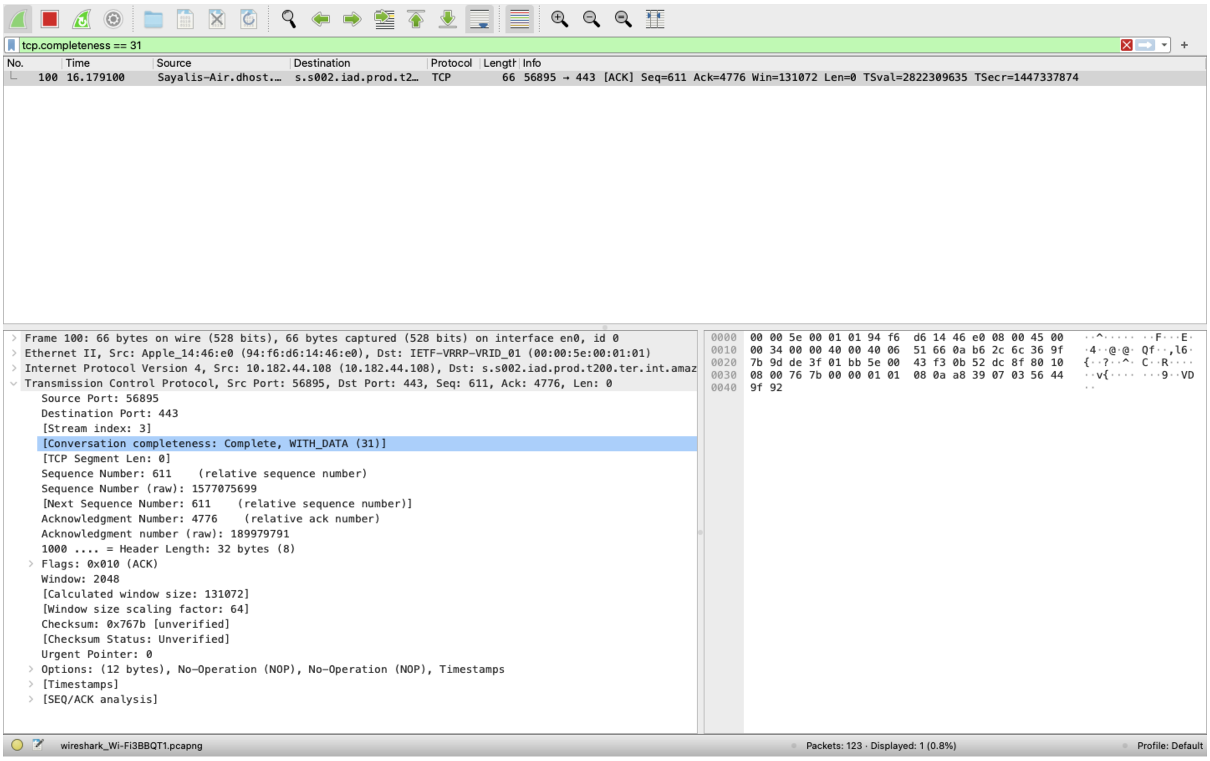
<u>a)tcp.completeness == 7</u>

1)Here in Transmission Control Protocol, It uses the source port = 61413 and destination port 443 and regarding conversion completeness, it shows incomplete on data(15).



b)tcp.completeness == 31

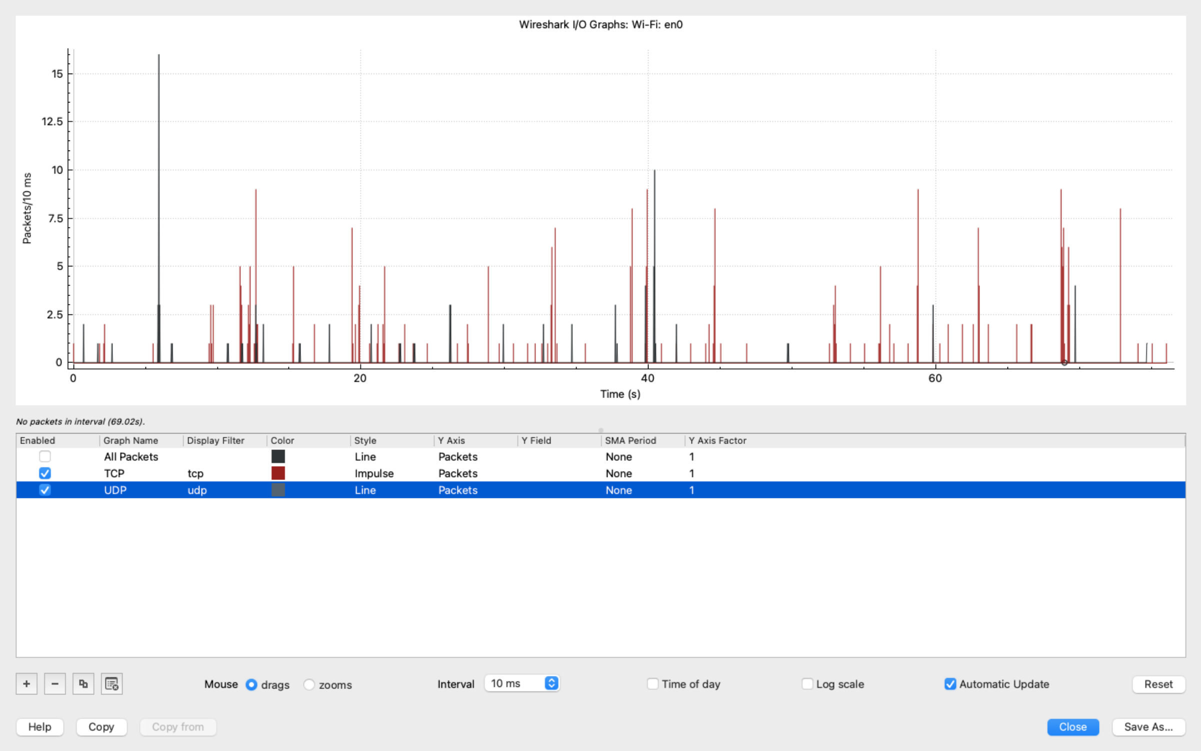
1)Here in Transmission Control Protocol, It uses the source port = 56895 and destination port 443 and regarding conversion completeness, it shows complete with data(31).



Q5)Using Wireshark observe packets that follows DNS and TCP. Plot a graph for the same (in Wireshark) and provide screenshot for an interval of per 10ms.

By observing the packet that follows DNS and TCP here we are plotting a graph in Wireshark for an interval of per 10ms.

GRAPH SCREENSHOT:

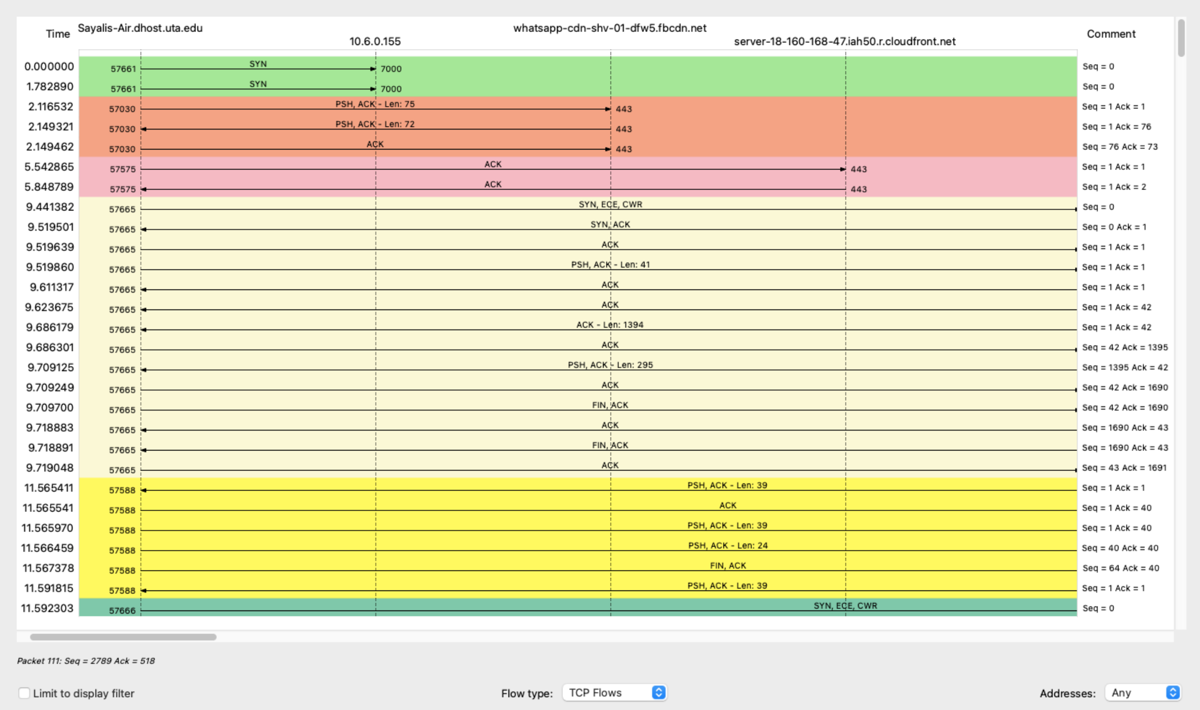


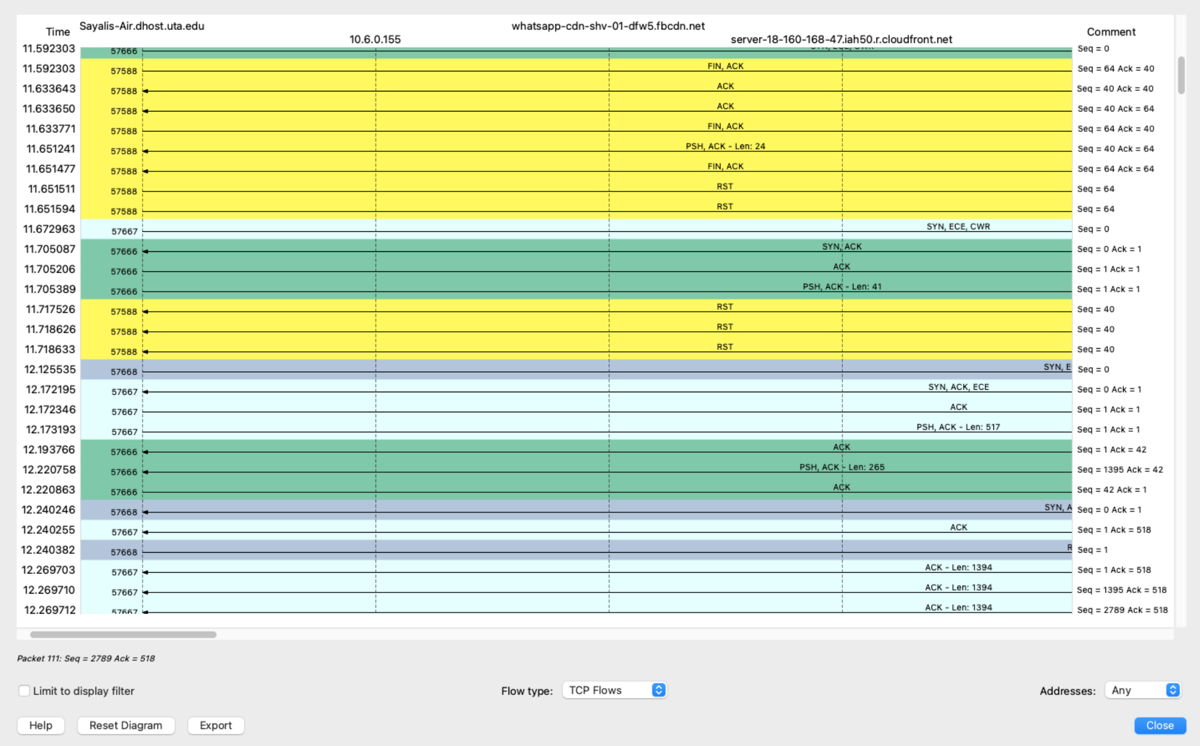
Q6)For a request, capture the TCP flow using the Wireshark Flow chart that highlights the [SYN], [SYN,ACK], [ACK] and [FIN,ACK]. Explain what it does.

TCP Flows:

1) When a client tries to establish a TCP connection with a server, SYN packets are typically created, and the client and server exchange a series of messages that typically go like this
2)By communicating with the server via a SYN (synchronize) message, the client seeks a connection.
3)This request is acknowledged by the server by returning SYN-ACK to the client.
After receiving an ACK from the client, the connection is established.
4) Acknowledgment Code, or ACK, refers to a service that is offered by mail companies to inform the sender of a letter that the recipient has received the delivery. It is usually a form signed by the receiver and then delivered to the sender. This gives proof to the sender that the letter has been received.
5) The sender sends TCP FIN to the receiver for an outgoing stream. The packet has a FIN flag set as another type of TCP message. The packet has a sequence number, the receiver sends the FIN Ack with one more sequence number received in the FIN. Now the connection is closed in one direction.

Screenshots:





Q7) Briefly explain the function of DNS? Provide a screenshot of Wireshark that includes the Source Port and Destination port for the DNS queried message

ANS:

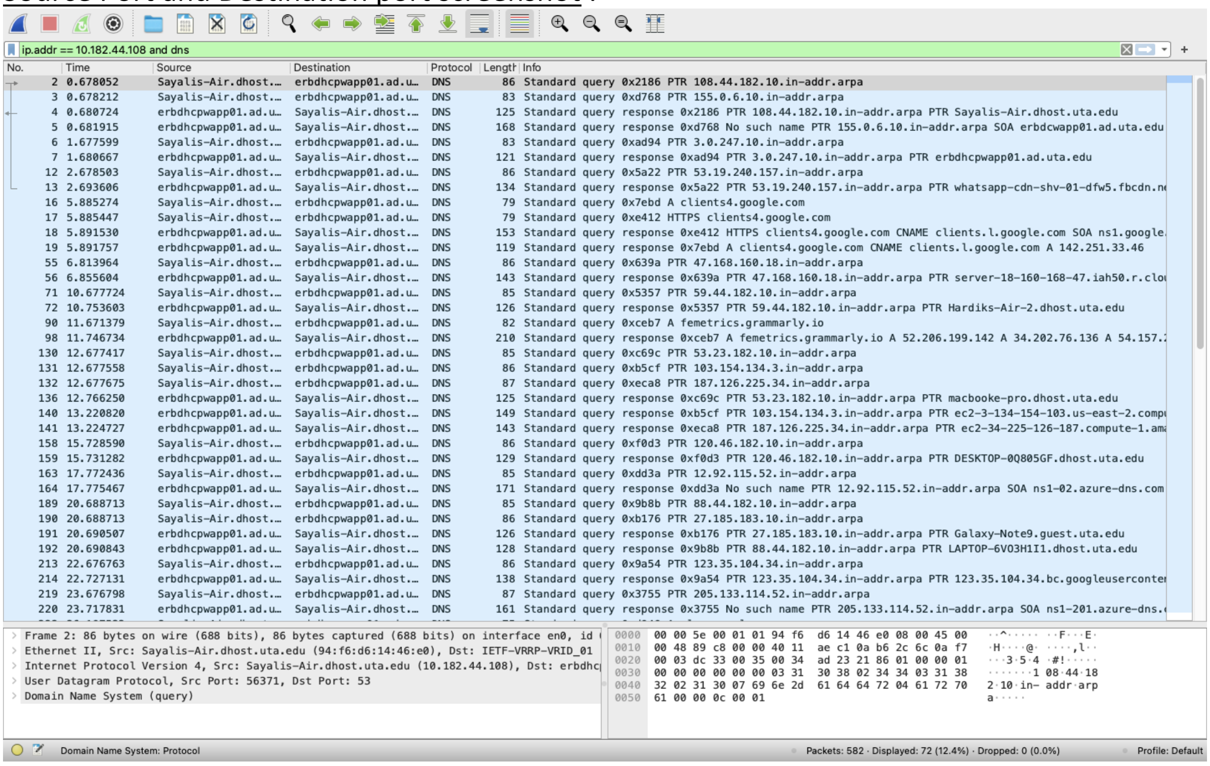
Function of DNS:

1)A hostname is transformed into an IP address that computers can understand as part of the DNS resolution process.

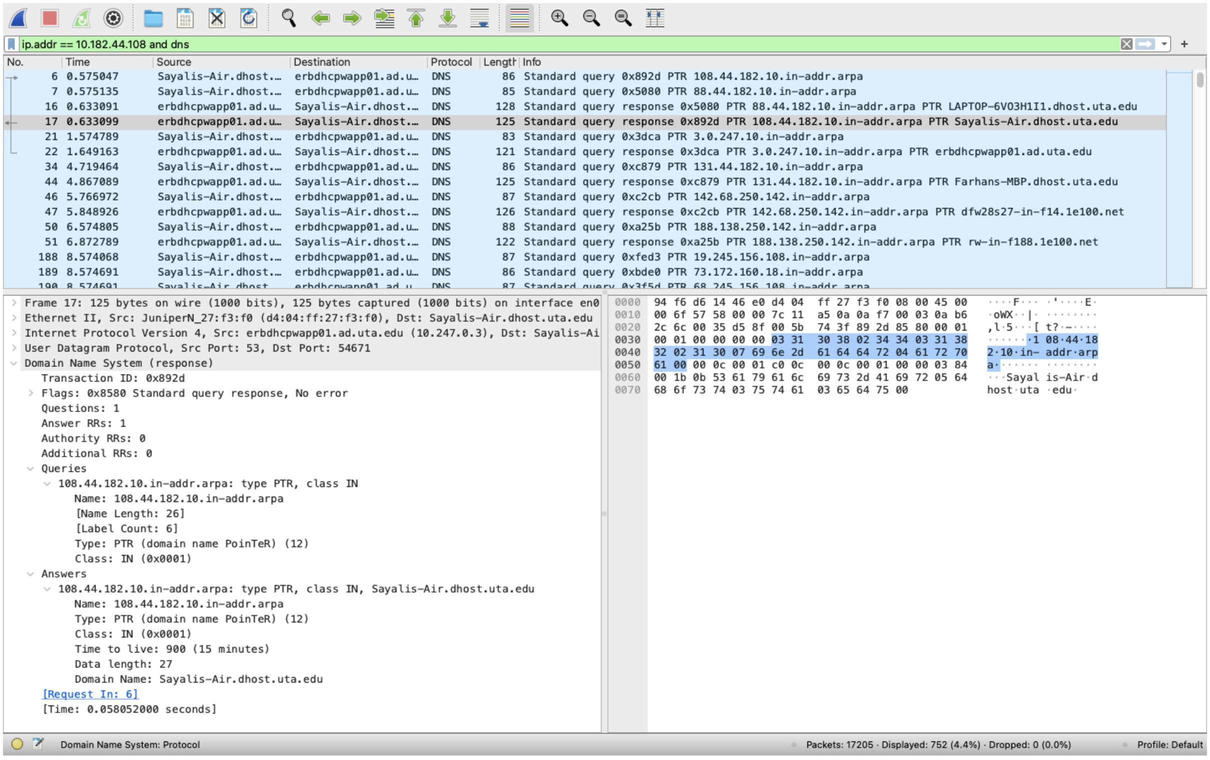
2)Each Internet-connected device has a unique IP address, which is required to identify the right item, just as a street address is required to identify a certain residence.

3)When a user requests a webpage to load, a translation between what they type into their web browser and the machine-friendly address required to find the webpage must take place.
4)DNS Server has 4 name server and they plays a role:
1. DNS recursor - The recursor is comparable to a librarian who is asked to look for a specific book in a library. The DNS recursor is a server made to take requests from client machines using programs like web browsers. The recursor is typically thereafter in charge of submitting further queries to respond to the client's DNS query.
2. The root nameserver: IT is the first stage in converting human readable host names into IP addresses (resolving). It can be compared to an index that directs readers to certain book racks in a library; often, it acts as a guide to other, more precise locations.
3. TLD nameserver - A top-level domain (TLD) server might be compared to a particular shelf of books in a library. This nameserver, which hosts the final part of a hostname (in the case of example.com, the TLD server is "com"), is the next stage in the process of locating a specific IP address.
4. Authoritative nameserver - This last nameserver can be compared to a dictionary on a shelf of books, allowing one to look up a specific name and get its definition. In the nameserver inquiry, the authoritative nameserver is the last stop. If the authoritative name server has access to the requested record, it will provide the DNS Recursor (the librarian) with the IP address for the requested hostname.

<u>Source Port and Destination port screenshot</u> :



Q8) Locate the DNS query and response messages. Are they sent over UDP or TCP?
ANS:
Packets are sent over UDP.
<u>Screenshot of Query and response message:</u>



Q9) What are the different types of DNS records? Using Wireshark examine the DNS query message and write the "TYPE" of the DNS record.
ANS:

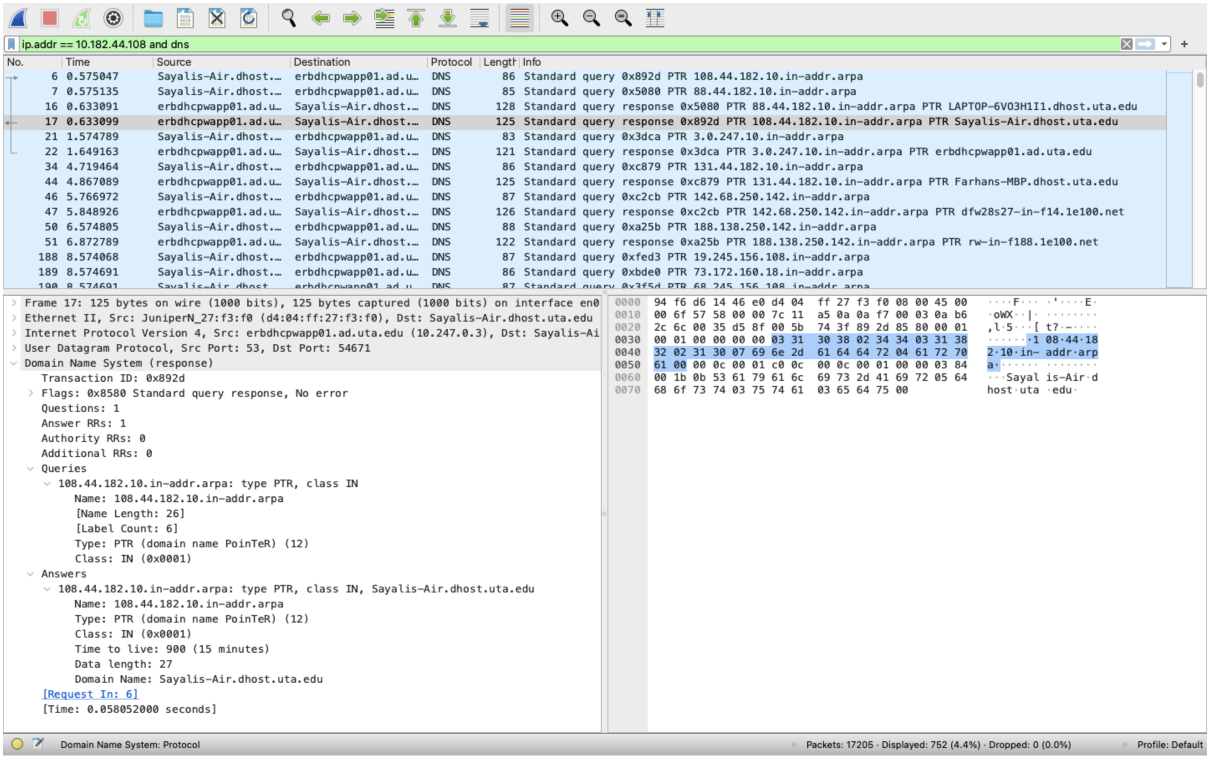
types of DNS records:

- A record
- AAAA record
- CNAME record
- Nameserver (NS) record
- Mail exchange (MX) record
- SOA record
- TXT record
- PTR record
- SRV record
- CERT record
- DCHID
- DNAME

Screenshot of the type and query message:

Type in the snapshot is PTR:

A pointer (PTR) record provides a domain name for reverse lookup. It's the opposite of an A record as it provides the domain name linked to an IP address instead of the IP address for a domain.



Q10) What are Authoritative and Recursive nameservers? Demonstrate either of them using Wireshark.

ANS:

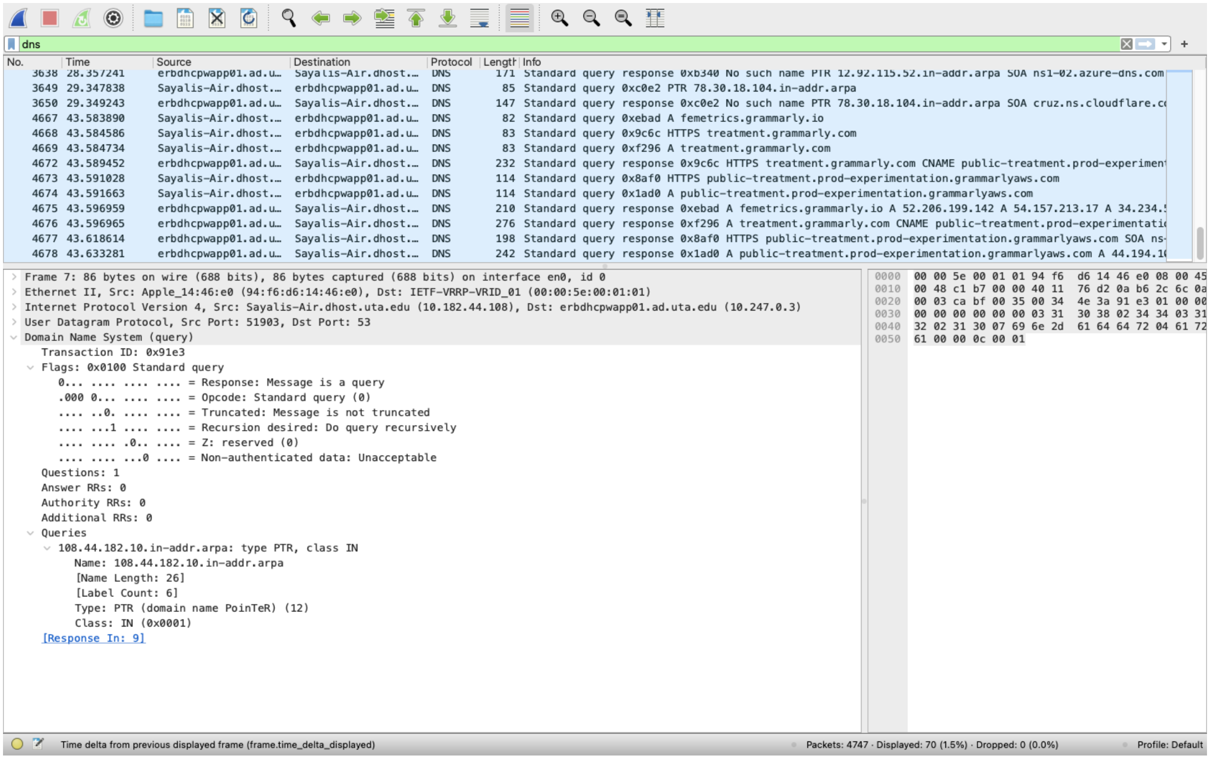
1)When attempting to connect to a website, your queries will be processed by one of two different types of servers.

2)Authoritative and Recursive DNS servers are the ones that reply to your requests and maintain the canonical data that specifies which IP address corresponds to which domain.

3)The "mappings" of your domain names to IP addresses are kept on Authoritative DNS servers, to put it briefly. System administrators typically configure this domain name to IP mapping. When someone visits a website, Recursive DNS servers are contacted for lookups. Recursive DNS servers then inquire about the solution from the required Authoritative Name Server. The individual who requests the information will then receive this response from the Recursive name server.

4) The mainstays of the DNS lookup process are recursive servers. In order to provide the correct IP for the inquiring client, they frequently need to perform several DNS lookups. These servers are often run by an ISP (Internet Service Provider) or specialized DNS providers. For instance, Google manages its own public recursive DNS servers.

<u>Screenshot showing Recursive nameservers:</u>



References:

https://youtu.be/bEXEEfbNADs
https://www.youtube.com/watch?v=C0cX_AqMul8
https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTCPAnalysis.html
https://www.youtube.com/watch?v=qYh6k-S5xC4
https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages#:~:text=HTTP%20messages%20are%20how%20data,the%20answer%20from%20the%20server
https://www.site24x7.com/learn/dns-record-types.html
https://developer.mozilla.org/en-US/docs/Glossary/TCP_handshake
https://www.cspsprotocol.com/tcp-connection-termination/