# Task 3.1p

# Module 2: Evidence of Learning

Summary of Learning:
Throughout this activity, I have deepened my understanding of the application layer protocols, particularly focusing on HTTP (Hypertext Transfer Protocol), and practiced analyzing HTTP interactions using Wireshark. Here's how I've achieved each of the learning objectives:

Explain the role of application layer protocols play in computer networking:
Through discussions and role plays with my peers, I have learned that application layer protocols serve as the interface for communication between software applications over a network. These protocols define the rules and conventions for data exchange, ensuring compatibility and interoperability. We discussed examples such as HTTP, SMTP, FTP, and DNS, understanding their importance in facilitating various internet applications.

Demonstrate the understanding of HTTP:
By participating in role plays and analyzing HTTP interactions using Wireshark, I have gained a deeper understanding of how HTTP operates. I learned that HTTP is a client-server protocol used for fetching resources from web servers. It operates over TCP/IP and follows a request-response model, where clients send requests for resources, and servers respond with the requested content. Analyzing HTTP messages in Wireshark provided insights into the structure of HTTP requests and responses, including details such as HTTP version, status codes, headers, and content.

Evidence of Learning:
I have attached screenshots showcasing our group discussions, role plays, and Wireshark analysis, demonstrating my active participation and engagement in the learning process. These screenshots serve as evidence of my understanding of application layer protocols and HTTP. I have also produced the notes I took for this module.

Overall, this activity has been instrumental in enhancing my knowledge of computer networking concepts, particularly regarding the role of application layer protocols and the operation of HTTP. Through collaborative learning and practical analysis, I feel better equipped to apply these concepts in real-world scenarios.

# Evidence: Module Exercises

Exercise 1:

Which of the following statement is false? 10/10

- ○ DNS is an application layer protocol
- ◉ DNS uses a centralised set of servers
- ○ DNS uses a large number of geographically distributed servers which are organised in a hierarchical manner.
- ○ All Web applications use DNS

Exercise 2:

Which of the following application layer protocol is used by the web? 10/10

- ○ HTML
- ○ TCP
- ○ UDP
- ○ SMTP
- ◉ HTTP

Exercise 3:

Before you started this quiz, you wanted to review Module 2 content available in the unit site. You requested the introduction webpage of Module 2 that consists of some text and two images. To load the page in your browser, your browser sent one request message and received three response messages.
Is the above statement true? 10/10

- ○ True
- ◉ False

Exercise 4:

HTTPS is a completely different application layer protocol compared to HTTP and uses an additional message type called "S". Is the above statement is true/false? 10/10

- ○ True
- ◉ False

Exercise 5:

When you analyse a HTTP GET message in Wireshark you noticed the following line. 10/10

<lf>Connection:keep-alive<cr><lf>

What does the above line mean?

- ○ The Web server requested a non-persistent connection
- ○ The browser requested a non-persistent connection
- ◉ The browser requested a persistent connection
- ○ The Web server requested a persistent connection
- ○ None of the above

Exercise 6:

One HTTP client wants to retrieve a Web document at a given URL. The IP address of the HTTP server is initially unknown. What are the transport and application-layer protocols used in this scenario to retrieve the Web document? 10/10

- ○ HTTP, TCP, SMTP
- ○ HTTP, TCP, DNS
- ○ HTTP, HTML, TCP
- ○ HTTP, TCP
- ◉ HTTP, TCP, DNS, UDP

Exercise 7:

In your active class, one of your group members said " DNS protocol only uses repetitive queries". Is the above statement true/ false? 10/10

- ○ True
- ◉ False

Exercise 8:

When you analyse HTTP protocol in Wireshark, you noticed the following two line in a HTTP response message. 10/10

Accept-Ranges: bytes<cr><lf>

Content-Length: 3874<cr><lf>

Which of the following statement is true?

- ○ The response message is 3874 bits long
- ○ The response message is 3874 Bytes long
- ○ There are 3874 bits in the document being returned.
- ◉ There are 3874 bytes in the document being returned.

Exercise 9:

Which of the following statement is true? 10/10

- ○ With non-persistent connections between your browser and the web server, it is possible for a single TCP segment to carry many HTTP response messages.
- ◉ With non-persistent connections between your browser and the web server, a single TCP segment carry only one HTTP request message.
- ○ With non-persistent connections between your browser and the web server, it is possible for a single TCP segment to carry two HTTP request messages.

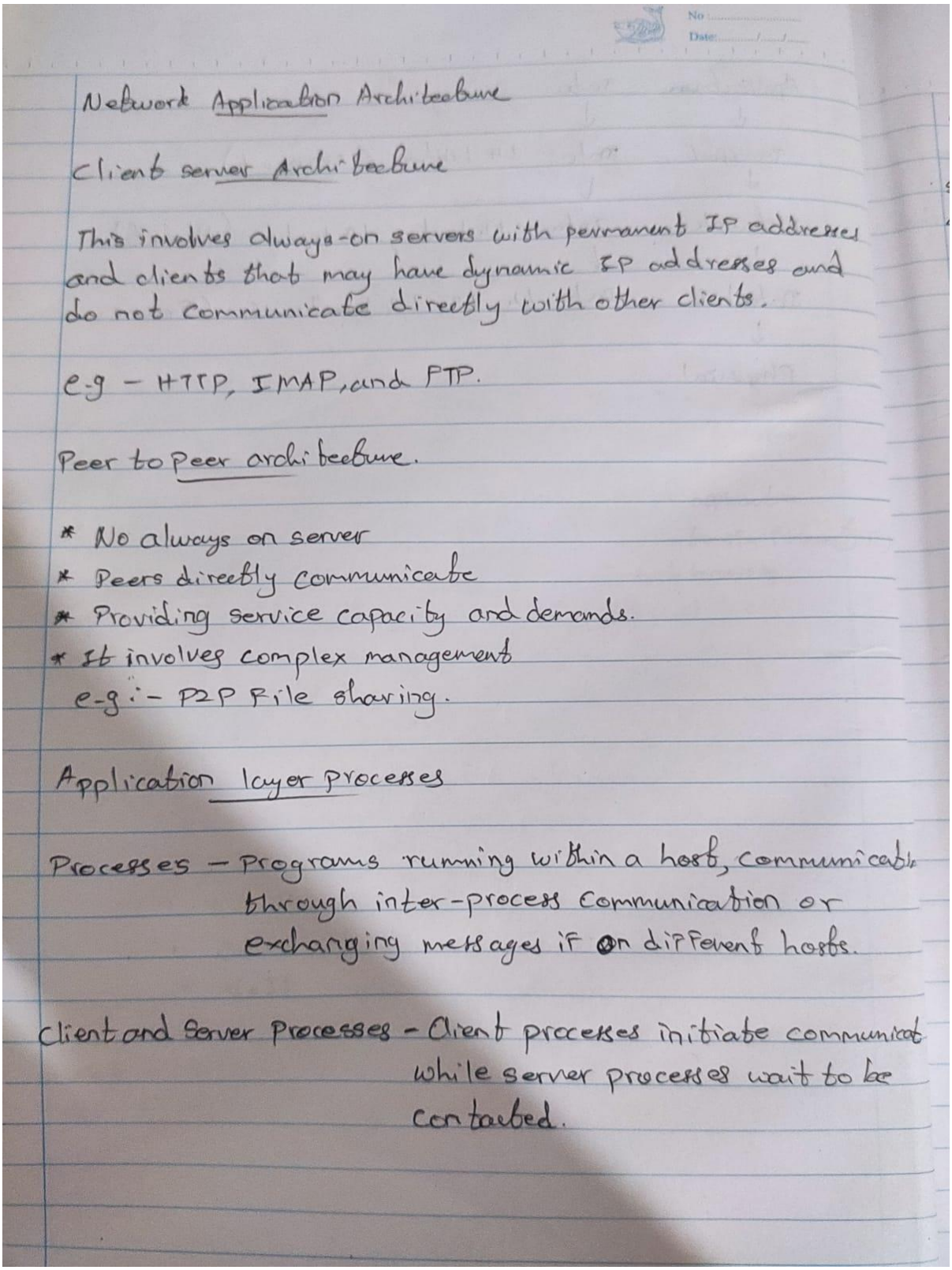
Exercise 10:

What is the encryption service provided by HTTP? 10/10

- ◉ None
- ○ SSL
- ○ TLS
- ○ TCP
- ○ TTL

# Evidence: Active Classes

These are the notes I took for the module



Network Application Architecture

Client server Architecture

This involves always-on servers with permanent IP addresses and clients that may have dynamic IP addresses and do not communicate directly with other clients.

e.g - HTTP, IMAP, and FTP.

Peer to Peer architecture.

* No always on server
* Peers directly communicate
* Providing service capacity and demands.
* It involves complex management

e.g :- P2P File sharing.

Application layer processes

Processes - Programs running within a host, communicate through inter-process communication or exchanging messages if on different hosts.

Client and Server Processes - Client processes initiate communicat while server processes wait to be contacted.

## Sockets and Addressing

Sockets are interface for sending/receiving messages to/from a process's ~~or~~ socket, analogous to a door.

Addressing - Processes must have identifiers, including IP addresses and port numbers, to receive messages.

## HTTP and Web objects

HTTP protocol defines how web clients request pages from server and how servers transfer pages to clients.

Web objects - webpages consist of objects like HTML files, images, applets, which are addressable by URLs.

## Types of HTTP connections

HTTP connections

Non-persistant connection
↓
Involves opening and closing TCP connections for each objects.

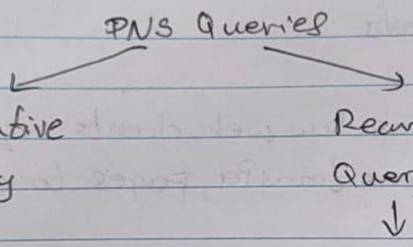
Persistant connection.
↓
Allows multiple objects to be sent over a single TCP connections.

Date: / /

# DNS (Domain Name server)

**DNS services**
* host-to-IP address translation
* host aliasing
* Mail server aliasing
* load distribution

**DNS structure**
* distributed and hierarchial database with root
* TLD
* authoritative servers.




Contacted server replies with the name of the server to contact.

DNS records – stores records (RR) like:
NS, A, CNAME, MX.

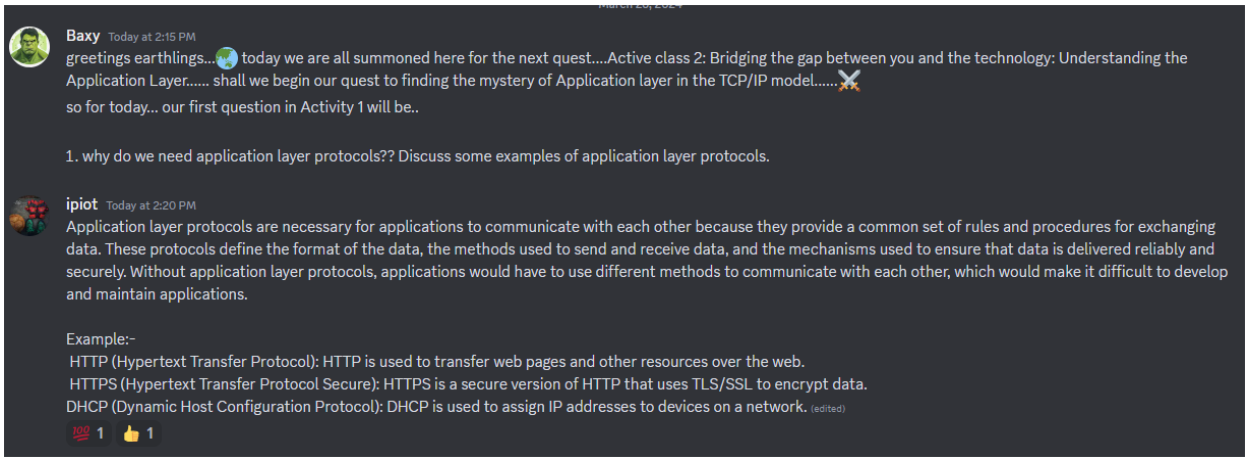
DNS messages – Query and reply messages have the same format, with headers and question answers RR

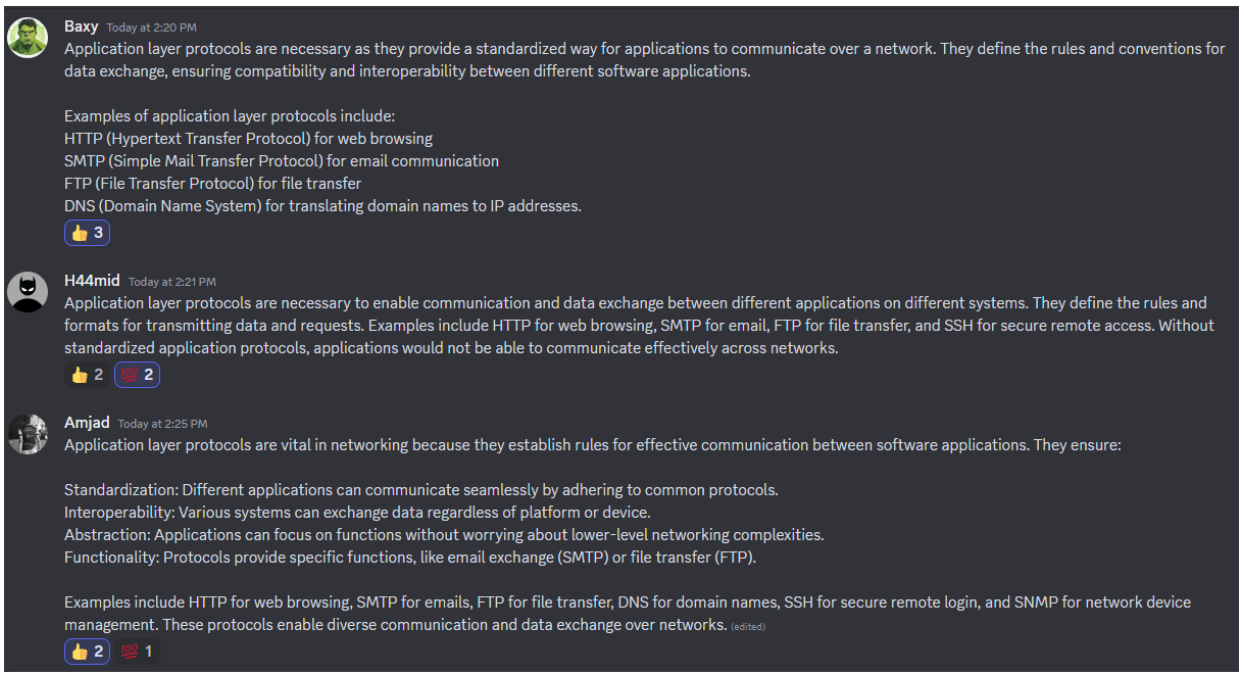
## Active Class 2: Evidence of learning summary

## Activity 1:

## Understanding the Application Layer using HTTP

Discussing  the role of application layer protocols play in computer networking.



**mabrook** Today at 2:33 PM
Application layer protocols are rules governing how software communicates over networks. HTTP fetches web pages, SMTP sends emails, while DNS translates domain names to IP addresses. These protocols ensure data travels securely and accurately between applications, enabling seamless internet functionality.

Some examples of application layer protocols are:

1. HTTP (Hypertext Transfer Protocol): Used for fetching web pages and resources from servers.
2. SMTP (Simple Mail Transfer Protocol): Facilitates the sending of emails between servers.
3. FTP (File Transfer Protocol): Enables the transfer of files between client and server.
4. DNS (Domain Name System): Converts domain names to IP addresses for internet communication.
5. DHCP (Dynamic Host Configuration Protocol): Assigns IP addresses to devices on a network automatically.

👍 2 💯 2

**Baxy** Today at 2:46 PM
Let's consider one of the popular application layer protocols, HTTP to learn the principal operation of HTTP. To carry on this discussion, we can do a small role play.

But before that i have question:
a. HTTP is a client-server protocol. Can you identify the key features of HTTP?

👍 3

**ipiot** Today at 2:48 PM
Client-server architecture: HTTP involves a client (e.g., web browser) and a server (e.g., web server) communicating with each other.
Request-response model: The client sends requests to the server, and the server sends responses back to the client.
Stateless: HTTP does not store information about the client's previous requests.
Cacheable: HTTP supports caching, which can improve performance by reducing the number of requests sent to the server.
Extensible: HTTP can be extended with new features and functionality through the use of HTTP extensions.

**Baxy** Today at 2:50 PM
--> HTTP stands for Hypertext Transfer Protocol. It's a protocol used for communication between a client (usually a web browser) and a server (which hosts web resources like web pages, images, etc.). In this communication model, the client initiates requests for resources, and the server responds to those requests.

--> It operates over TCP/IP and typically uses port 80 for communication.

--> HTTP follows a request-response model where clients (web browsers) send requests to servers, and servers respond with the requested resources. (edited)

👍 3

**Amjad** Today at 2:50 PM
HTTP is a client-server protocol with key features including statelessness, request-response communication, text-based messages, support for various media types, connectionless nature, and extensibility. It facilitates communication between web browsers (clients) and web servers, allowing the retrieval and exchange of web resources over the internet.

👍 3 💯 2

**mabrook** Today at 2:54 PM
yah sure, here are some of the key features of http below.

1. Client-server architecture: Here the http operates on a client-server model, where clients send requests to servers for resources,(clients are known as web browsers) and servers respond with the requested data.
2. Request-Response Cycle: Here http follows a request-responce cycle , where clients initiate requests by sending HTTP requests to servers, and servers respond with HTTP responses containing the requested data.
3. Text-Based Protocol: HTTP messages are text-based, consisting of headers and an optional message body. Headers contain metadata about the request or response, while the message body contains the actual data being transferred, such as HTML content, images, or files.
4. Security: HTTPS secures HTTP communication by encrypting data between clients and servers, ensuring confidentiality and integrity.

These features use http protocols for communication between the clients and servers on the www. (edited)

👍 3

**H44mid** Today at 2:58 PM
The Hypertext Transfer Protocol (HTTP) is a client-server protocol that forms the foundation of data communication on the World Wide Web. It follows a client-server architecture, where the client, typically a web browser, initiates requests, and the server responds to those requests. HTTP is a connectionless and stateless protocol, meaning that each request-response transaction is independent, and the server does not maintain information about the client's previous requests.

HTTP defines several request methods, such as GET, POST, PUT, and DELETE, which specify the action to be performed on the server. It also uses numerical status codes to indicate the result of a request, like 200 (OK), 404 (Not Found), or 500 (Internal Server Error). HTTP headers provide additional information about the request or response, such as content type, cache control, and authentication. One of HTTP's key features is its support for caching mechanisms, which improve performance by reusing previously fetched resources. The protocol's simplicity, scalability, and adherence to standards have contributed to its widespread adoption as the backbone of the World Wide Web.

👍 3

**Baxy** Today at 2:59 PM
Now lets jump into role play, first of all we should each decide who is going to take which role, there are 4 roles:
3 clients.
1 server. (edited)

**mabrook** Today at 3:04 PM
Yah lets simulate the communication between the server and clients using the HTTP protocol. Let someone take the role of server and the remaining members take the roles of clients.

**Baxy** Today at 3:24 PM
I'll be the client 1

**mabrook** Today at 3:25 PM
ok, then i will get into the server role 👍 (edited)

**H44mid** Today at 3:25 PM
i'll be the client 2

**Amjad** Today at 3:25 PM
I'll be client 3

**Baxy** Today at 3:27 PM
**client 1:** I'm going to initiate the TCP connection with the server (edited)

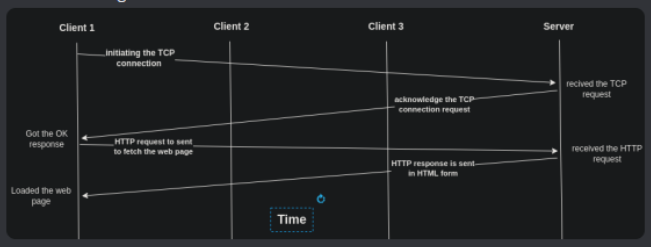
**mabrook** Today at 3:28 PM
**Server**: Ok, since im the **server** i will acknowledge the TCP connection request from you. (edited)

**Baxy** Today at 3:29 PM
**Client 1:** I've sent an HTTP request to the server to fetch a web page (edited)

**mabrook** Today at 4:02 PM
**Server:** Great, i have received the HTTP request client from you (client 1).
I have processed the HTTP request from you (client 1), and now i have sent the requested web page in an HTTP response back to you (client 1).

**Baxy** Today at 4:03 PM
**Client 1:** Thank you, I have received the HTTP response containing the requested web page

**ipiot** Today at 4:03 PM
ill be drawing the time line diagram for each,
here is the diagram for **client 1** and **server**




**H44mid** Today at 4:03 PM
**Client 2:**i'm going to initiate the TCP connection with the server

**mabrook** Today at 4:03 PM
**Server:** Ok i have acknowledged the TCP connection from you (client 2).

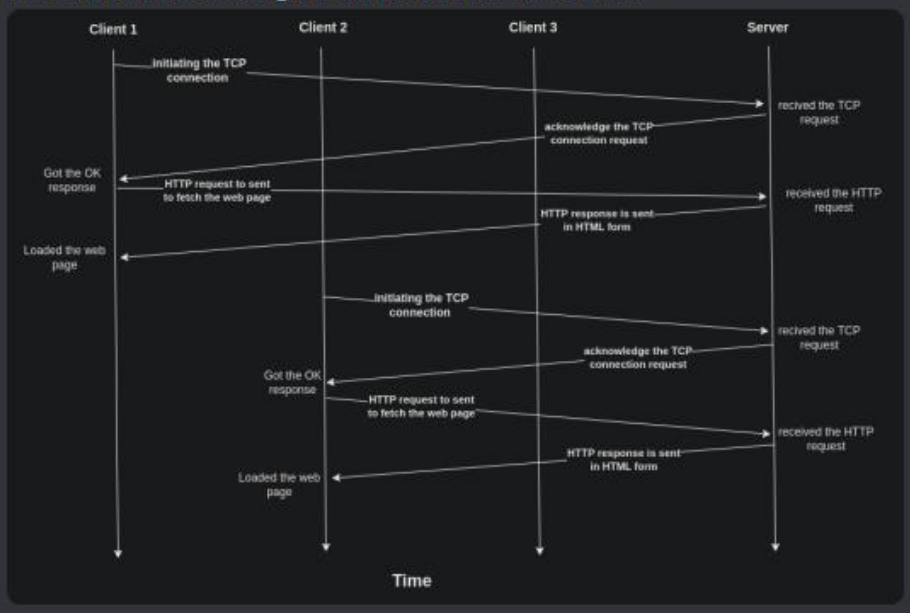
**H44mid** Today at 4:04 PM
**Client 2** : I have sent a HTTP request to the server to fetch a web page

**mabrook** Today at 4:04 PM
**Server:** I have received the HTTP request from you (Client 2).
I have processed the HTTP request from you (client 2), and now i have sent the requested web page in an HTTP response back to you (client 2).

**H44mid** Today at 4:04 PM
**Client 2:**I have receive the HTTP response containing the requested web page

**ipiot** Today at 4:04 PM

here is the time line diagram for the **client 2** and **server**




**Amjad** Today at 4:05 PM

**Client 3:** I'm going to initiate the TCP connection with the server

**mabrook** Today at 4:05 PM

**Server:** Ok Thanks, i have acknowledged the TCP connection from you (client 3).

**Amjad** Today at 4:06 PM

**Client 3 :** I have sent a HTTP request to the server to fetch a web page

**Baxy** Today at 4:21 PM
Let's do another role paying to understand the role of proxy server/ web cache. For this scenario, one group member who acted as a client can now act as a proxy server. You can assume that the proxy server has the objects that Client 1 requested but does not have the objects requested by the Client 3.

So for this I'll be acting as the proxy server.

**H44mid** Today at 4:22 PM
ok i'll be client 1 this time

**mabrook** Today at 4:22 PM
Fine, Then as usual i'll be the web server. (edited)

**Amjad** Today at 4:23 PM
I'll be client 2 for the play (edited)

**H44mid** Today at 4:23 PM
**Client 1:** Initiating TCP connection with Proxy Server

**Baxy** Today at 4:23 PM
**Proxy Server:** Ok, I Received connection from Client 1
**Proxy Server:** I'll Check the cache for requested object by client 1
**Proxy Server:** oops! sry mate, Object not found in my cache
**Proxy Server:** So, I'll Initiate a TCP connection with Web Server

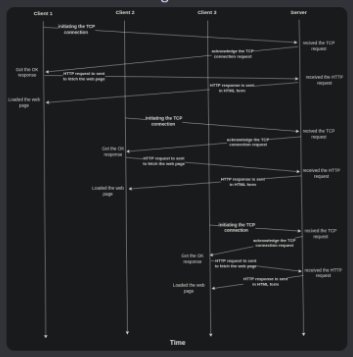
**mabrook** Today at 4:26 PM
**Web Server:** Ok, I have Received the connection from you ( Proxy Server). (edited)

**Baxy** Today at 4:27 PM
**Proxy Server:** Now, I'll Send a GET request to Web Server

**mabrook** Today at 4:06 PM
**Server:** Great, I have received the HTTP request from you (Client 3).
I have processed the HTTP request from you (client 3), and now i have sent the requested web page in an HTTP response back to you (client 3).

**Amjad** Today at 4:06 PM
**Client 3:** I have receive the HTTP response containing the requested web page

**ipiot** Today at 4:07 PM
here is the time line diagram for **client 3** and **server**




**Baxy** Today at 4:21 PM
Let's do another role paying to understand the role of proxy server/ web cache. For this scenario, one group member who acted as a client can now act as a proxy server. You can assume that the proxy server has the objects that Client 1 requested but does not have the objects requested by the Client 3.

**mabrook** Today at 4:27 PM

**Web Server:** I have received the GET request. Thank you

**Web Server:** I have sent the response with the requested object to you (Proxy Server)

**Baxy** Today at 4:28 PM

**Proxy Server:** Thanks mate, I Received the response from you (Web Server)

**Proxy Server:** I'll save the object in to my cache

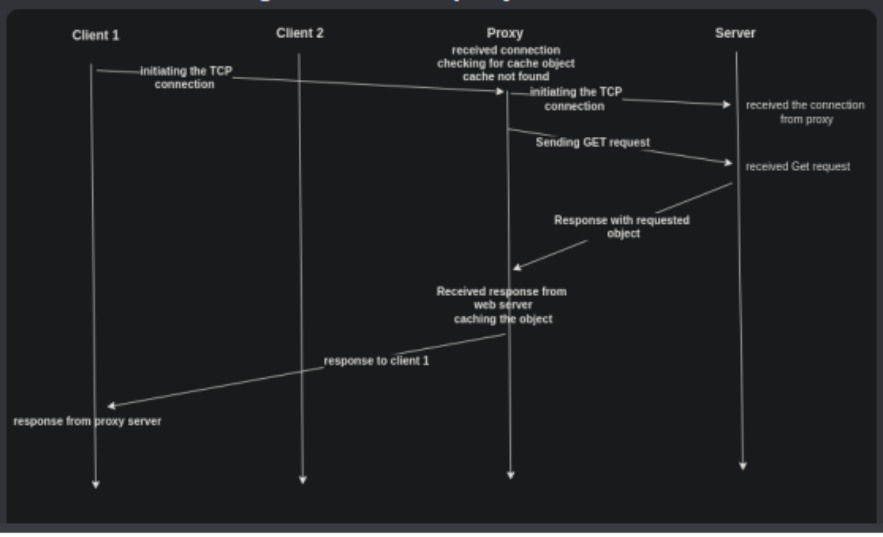
**Proxy Server:** Now I'll Send the response to Client 1

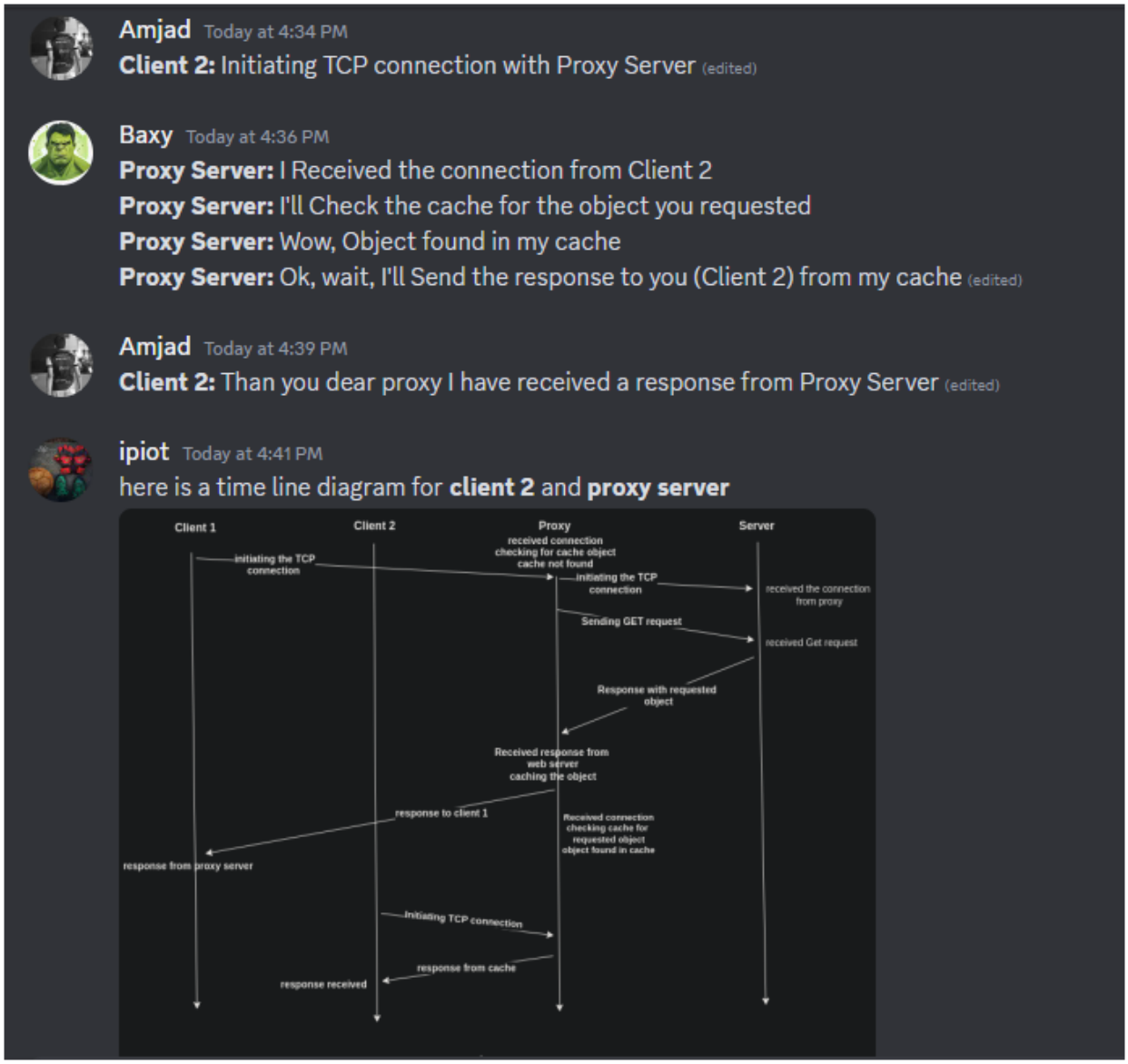
**H44mid** Today at 4:30 PM

**Client 1:** Thank you I Received response from you (Proxy Server) (edited)

**ipiot** Today at 4:32 PM

Here is the time line diagram for **client 1** , **proxy server** and the **web server**






**Activity 2 :**

**Analysing HTTP in Wireshark**

**Baxy** Today at 5:03 PM

Well done Earthlings....🌍 The first quest has been completed successfully. but don't get cheered up so soon, we still have one more realm to conquer⚔️.

Lets begin the **Activity 2**

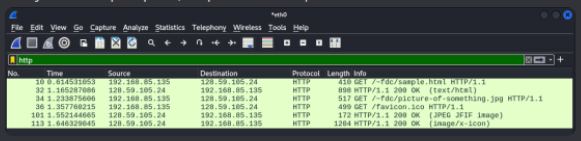
In this we need to analyze the website with the http using Wireshark. Today, we will explore some important aspects of the HTTP protocol including the basic GET/response interaction, and HTTP message formats.

For this we need to analyze the website that was given in the assignment.

👍 1

**ipiot** Today at 5:05 PM

The website which we are going to analyze here for this activity is http://www.columbia.edu/~fdc/sample.html,
Below is the image for the captured packets using wireshark and having http filter to get only the http requests and responses, considering the captured packet lets analyze the http request, response and sequence. (edited)




**Baxy** Today at 5:20 PM

Ok, now there are bunch questions to discuss regarding the packets captured.

So the first question is:

**a. what is the Sequence of HTTP message exchange?**

**Client 1 Sends HTTP GET Request (Frame 10):**
The client initiates the communication by sending an HTTP GET request to the server. The request specifies the desired resource, which in this case is /fdc/sample.html on the server. The request also includes headers that provide additional information, such as the HTTP version (1.1) and the user agent (the type of software making the request).

**Server Sends HTTP Response (Frame 32):**
The server responds to the client's GET request with an HTTP response. The response includes a status code (200 OK), indicating that the request was successful. The response also includes headers that provide information about the response, such as the content type (text/html) and the content length (the size of the data in bytes). The body of the response contains the actual HTML content of the requested webpage (/fdc/sample.html).

**Client 1 Sends HTTP GET Request (Frame 101):**
The client sends another HTTP GET request, this time for a resource named favicon.ico. Favicons are small icons typically displayed in the browser tab or title bar for a website.

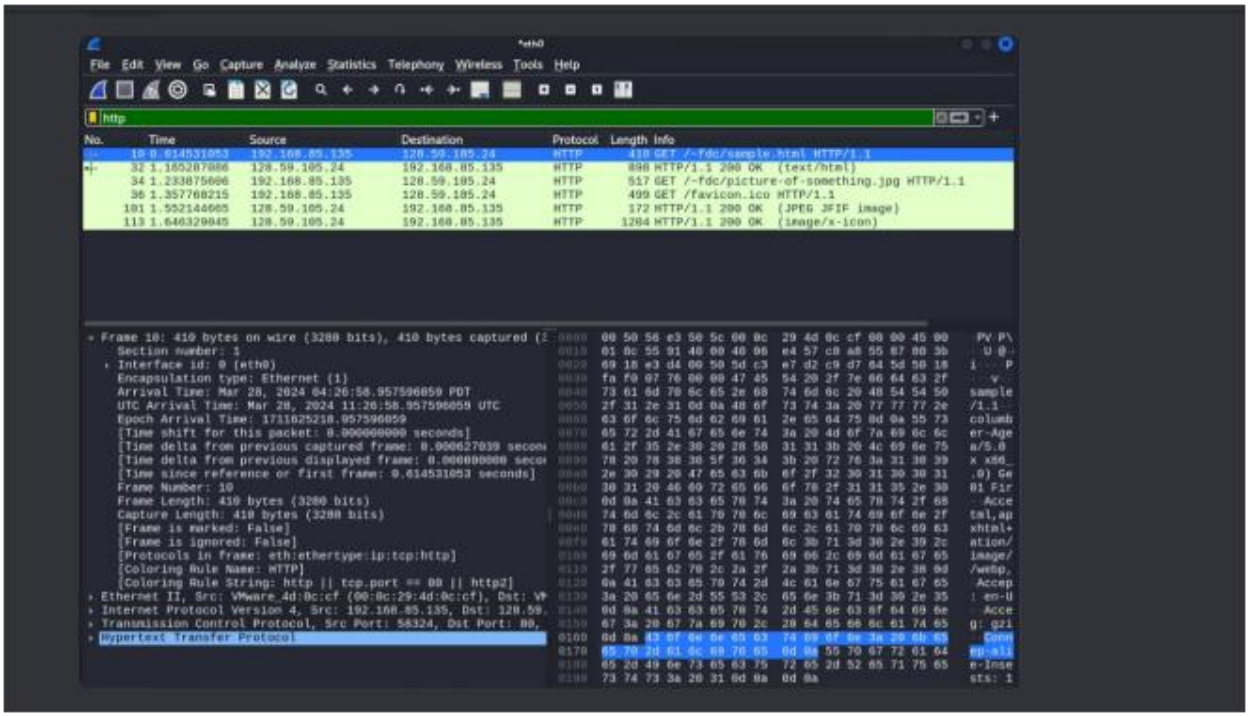
**Server Sends HTTP Response (Frame 113):**
The server responds with a 200 OK status code, indicating success, and sends the requested favicon image data (JPEG format) in the response body.

**Client 1 Sends HTTP GET Request (Frame 34):**
The client sends a third HTTP GET request for a resource named /fdc/picture-of-something.jpg.

**Server Sends HTTP Response (Not shown):**
The capture you provided doesn't show the server's response to this third request, but it would likely follow the same pattern as the previous responses. The server would send a response with a status code (likely 200 OK) and the image data (JPEG format) in the body.

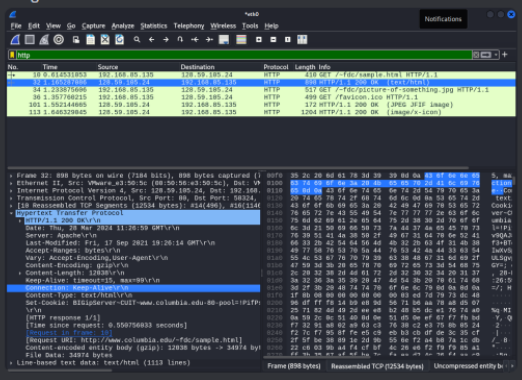
The captured sequence shows three HTTP request-response exchanges between a client and a server. The client retrieves three resources from the server: an HTML webpage, a favicon icon, and an image file. (edited)

🔥 1




**ipiot** Today at 5:23 PM

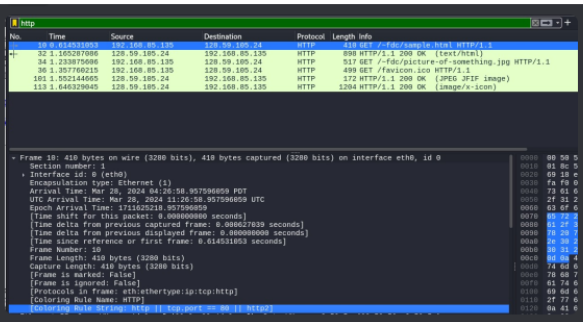
b. Are we using persistence/ non-persistence connection?
when analyzing the connection we also could see that it is a persistence connection because we could see that, in the HTTP protocol section under the connection status it is mentioned as "Keep-Alive". when it is mentioned as Keep-Alive it is a persistence connection or else if it is mentioned as "close" it is non-persistence connection.
Image



**Amjad** Today at 5:35 PM

c. What are details you can find in HTTP GET message?
In an HTTP GET message, you'll find:
Request Line: specifies the action (GET), requested resource, and HTTP version.
Headers (optional): provide additional info like the client software and the website visited before (Referer).

👍 1




d. Check the packet details in the middle Wireshark packet details pane. Can you identify the details in Ethernet II / Internet Protocol Version 4 / Transmission Control Protocol / Hypertext Transfer Protocol frames?

Wireshark lets you drill down into packet details. Here, you can see information for each layer:

Ethernet II: This shows the source and destination MAC addresses, identifying the specific devices involved.

IPv4: This layer reveals the IP addresses of the sender and receiver.

TCP: Here, you'll find the port numbers, with port 80 indicating standard HTTP traffic.

HTTP: While not directly visible, the port number suggests this layer carries HTTP data, likely an HTTP request or response.

💯 1 👍 1

HTTP: While not directly visible, the port number suggests this layer carries HTTP data, likely an HTTP request or response.

💯 1 👍 1

```
▾ Ethernet II, Src: VMware_4d:0c:cf (00:0c:29:4d:0c:cf), Dst: VMware_e3:50:5c (00:50:56:e3:50:5c)
  ▾ Destination: VMware_e3:50:5c (00:50:56:e3:50:5c)
      Address: VMware_e3:50:5c (00:50:56:e3:50:5c)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: VMware_4d:0c:cf (00:0c:29:4d:0c:cf)
      Address: VMware_4d:0c:cf (00:0c:29:4d:0c:cf)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.85.135, Dst: 128.59.105.24
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 396
    Identification: 0x5591 (21905)
  ▸ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0xe457 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.85.135
    Destination Address: 128.59.105.24
```

**mabrook** Today at 6:00 PM

k. When was the HTML file that you are accessing last modified at the server?

Ans : According to the response headers shown in the image, the Last-Modified date for the file being accessed is Fri, 17 Sep 2021 19:26:14 GMT.

```
▾ HTTP/1.1 200 OK\r\n
  ▸ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Date: Thu, 28 Mar 2024 11:26:59 GMT\r\n
  Server: Apache\r\n
  Last-Modified: Fri, 17 Sep 2021 19:26:14 GMT\r\n
  Accept-Ranges: bytes\r\n
  Vary: Accept-Encoding,User-Agent\r\n
```

💯 2

**H44mid** Today at 5:58 PM

j. What is the status code returned from the server to your browser?
The status code returned from the server to the browser is 200, according to the information shown in the image.

The image appears to be a captured HTTP response from a web server, likely obtained through a network traffic analysis tool like Wireshark or a web debugging proxy.

The key details shown are:

"HTTP/1.1 200 OK\r\n" This line indicates the HTTP protocol version (1.1) and the status code (200) with the description "OK", which means the request was successful.
"Status Code: 200" This explicitly states the numeric status code is 200.
"[Status Code Description: OK]" This provides the textual description for status code 200, which is "OK".
A status code of 200 is the standard response for successful HTTP requests, indicating that the requested resource (e.g., a web page or file) was retrieved and transmitted successfully from the server to the client (browser).

The image also shows some additional response headers like "Server", "Last-Modified", "Accept-Ranges", and "Vary", providing further details about the server software, caching directives, and supported encodings.

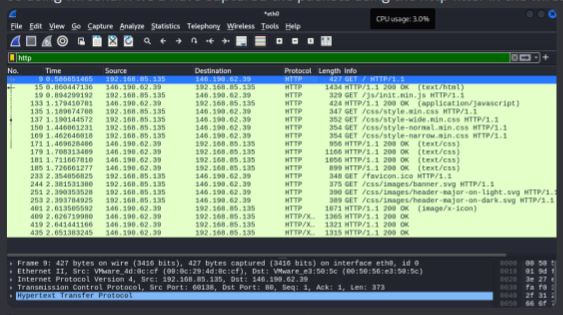
```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Date: Thu, 28 Mar 2024 11:26:59 GMT\r\n
  Server: Apache\r\n
  Last-Modified: Fri, 17 Sep 2021 19:26:14 GMT\r\n
  Accept-Ranges: bytes\r\n
  Vary: Accept-Encoding,User-Agent\r\n
  Content-Encoding: gzip\r\n
```

💯 2

**ipiot** Today at 7:15 PM

l. Repeat the analysis by accessing a different web page of your choice.

for this we are going to analyze the website http://httpforever.com/
so using wireshark we a have captured the packets using the http filter in the wireshark filter column




**HTTP Forever**

A site that will always be available over HTTP!

💯 2

**mabrook** Today at 7:24 PM

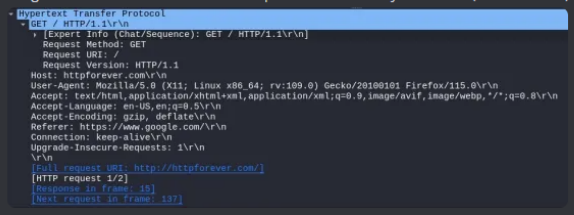
The HTTP message exchange sequence shown in the image is a GET request. The "Request Method: GET" line clearly indicates that this is a GET request being sent over HTTP/1.1 protocol.

**H44mid** Today at 7:24 PM

a. What is the sequence of HTTP message exchange?

This image displays information about an HTTP GET request made by a web browser. It shows the details of the request, including the request method (GET), URI (/), version (HTTP/1.1), host (httpforever.com), user-agent (Mozilla Firefox on Linux), accepted content types, languages, encodings, referrer, and connection settings.

The request is part of the Hypertext Transfer Protocol (HTTP), which is the underlying protocol used for data communication on the World Wide Web. The image provides insights into the various headers and parameters sent by the client (web browser) when making a GET request to retrieve a resource from a web server. (edited)



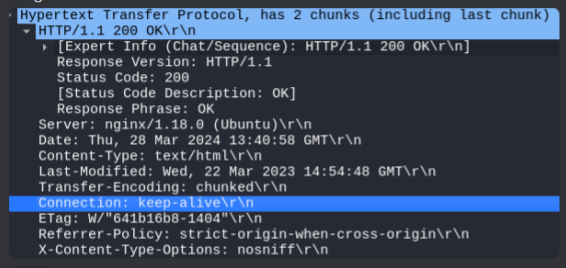

💯 2

**ipiot** Today at 7:25 PM

b. Are we using persistence/ non-persistence connection?

when analyzing the connection we also could see that it is a persistence connection because we could see that, in the HTTP protocol section under the connection status it is mentioned as "Keep-Alive". when it is mentioned as Keep-Alive it is a persistence connection or else if it is mentioned as "close" it is non-persistence connection.

Image




💯 1

**Amjad** Today at 7:27 PM

c. What are details you can find in HTTP GET message?

Based on the capture, here are some details you can find in an HTTP GET message:

Request Line: This line specifies the action (GET), requested resource, and HTTP version. In the capture, the first line for each request is the request line. For instance, the first request line is: GET / -fdc/sample.html HTTP/1.1

Headers (optional): These provide additional info like the client software (User-Agent) and the website visited before (Referer). The lines following the request line and before the blank line are the headers. For example, the headers for the first request include Host: 128.59.105.24 (edited)

💯 2

**H44mid** Today at 5:41 PM

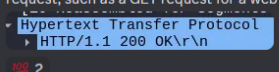
e. What is the HTTP version your browser used?

"HTTP/1.1" refers to the version of the HTTP protocol being used, which is version 1.1.

"200 OK" is an HTTP status code that indicates the request was successful. The code 200 means "OK" and is the standard response for successful HTTP requests.

"\r\n" represents the carriage return and line feed characters, which are used to denote the end of the status line in HTTP.

So, when combined, "HTTP/1.1 200 OK\r\n" is the standard success response line that a web server sends back to the client after receiving and processing a valid HTTP 1.1 request, such as a GET request for a web page. This indicates that the requested resource can be transferred successfully over the HTTP protocol.




💯 2

**Baxy** Today at 5:54 PM

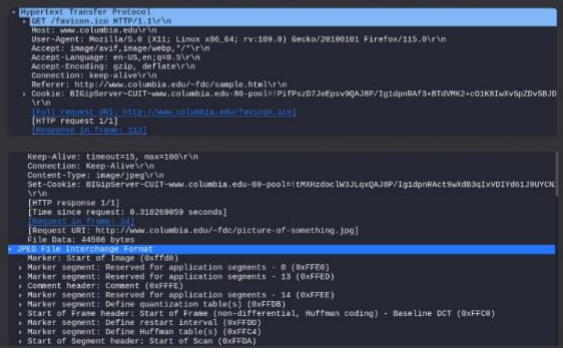
**f. Identify the response message received from the server?**

The client sent an HTTP GET request for a resource named favicon.ico. This suggests the client is requesting the favicon (small icon) for a website.

HTTP communication follows a request-response pattern. A client sends a request, and the server responds with a message containing the requested data or an error message.

The server most likely responded with a status code of 200 OK, indicating the request was successful. The response likely included a header field called Content-Type, which would be set to image/jpeg (or potentially image/x-icon) because the client requested a favicon, which is typically an image format. The body of the response would contain the actual image data representing the favicon for the website.

The information available suggests the server successfully responded to the client's request for a favicon image. The response likely included a 200 OK status code, an image content type header, and the actual image data for the favicon.



**Amjad** Today at 5:55 PM

g. What is version of HTTP that the server is running?

Server is using the same version as the HTTP 1.1 version

```
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
```

💯 2

**ipiot** Today at 5:56 PM

h. Can you identify the IP address of your computer?
i. Can you identify the IP address of http://www.columbia.edu/ server?

the ip address of my pc and the servers ip address is been shown in the below image as the "src" for the source ip address which is my pc ip address and the "dst" as the destination which is the servers ip address

```
Internet Protocol Version 4, Src: 192.168.85.135, Dst: 128.59.105.24
```

💯 2

**mabrook** Today at 5:40 PM

This image shows a network packet capture analysis output for an IPv4 TCP packet. It reveals details like the source (192.168.85.135) and destination (128.59.105.24) IP addresses, source (58324) and destination (80) ports, sequence and acknowledgment numbers, TCP flags (PSH, ACK), window size (64240 bytes), and conversation completeness status (Incomplete, DATA). Such outputs are used for network troubleshooting, traffic analysis, security monitoring, and performance optimization. (edited)

```
▸ Internet Protocol Version 4, Src: 192.168.85.135, Dst: 128.59.105.24
▾ Transmission Control Protocol, Src Port: 58324, Dst Port: 80, Seq: 1, Ack: 1, Len: 356
    Source Port: 58324
    Destination Port: 80
    [Stream index: 0]
  ▸ [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 356]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 157311954
    [Next Sequence Number: 357    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3386336349
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0x0776 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]
    TCP payload (356 bytes)
```

💯 2 👍 1

The below image showcases an HTTP request sent using the Hypertext Transfer Protocol (HTTP). It contains details such as the request method (GET), the requested resource URI (/~fdc/sample.html HTTP/1.1), the host (www.columbia.edu), the user-agent (Mozilla/5.0 on Linux), and various headers like Accept, Accept-Language, Accept-Encoding, and Connection. The request is marked as [HTTP request 1/1], indicating it is the first and only HTTP request in this exchange. This type of output is typically seen when inspecting network traffic or debugging web applications.

```
▾ Hypertext Transfer Protocol
  ▸ GET /~fdc/sample.html HTTP/1.1\r\n
    Host: www.columbia.edu\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://www.columbia.edu/~fdc/sample.html]
    [HTTP request 1/1]
    [Response in frame: 32]
```

💯 2

**Amjad** Today at 7:27 PM

c. What are details you can find in HTTP GET message?
Based on the capture, here are some details you can find in an HTTP GET message:

Request Line: This line specifies the action (GET), requested resource, and HTTP version. In the capture, the first line for each request is the request line. For instance, the first request line is: GET / -fdc/sample.html HTTP/1.1
Headers (optional): These provide additional info like the client software (User-Agent) and the website visited before (Referer). The lines following the request line and before the blank line are the headers. For example, the headers for the first request include Host: 128.59.105.24 (edited)

💯 2


**Baxy** Today at 7:37 PM

d. Check the packet details in the middle Wireshark packet details pane. Can you identify the details in Ethernet II / Internet Protocol Version 4 / Transmission Control Protocol / Hypertext Transfer Protocol frames?

👍 1

**Ethernet II Frame:**

Destination MAC Address: This is the physical address of the network interface card (NIC) intended to receive the packet.
Source MAC Address: This is the physical address of the NIC that sent the packet.
EtherType: This field indicates the higher-level protocol encapsulated within the Ethernet frame (in this case, it should be 0x0800 for IPv4).

**Internet Protocol Version 4 (IPv4):**

Version: This field specifies the IP version (should be 4 for IPv4).
Header Length: This indicates the length of the IP header in 32-bit words (typically 5 for a standard header).
Type of Service (TOS): This field defines how the packet should be treated by the network (e.g., priority, low delay).
Identification: This is a unique identifier for the datagram, used for fragmentation and reassembly.
Protocol: This field specifies the transport layer protocol used within the datagram (e.g., 6 for TCP).
Source IP Address: This is the logical address of the device that sent the datagram.
Destination IP Address: This is the logical address of the device intended to receive the datagram.

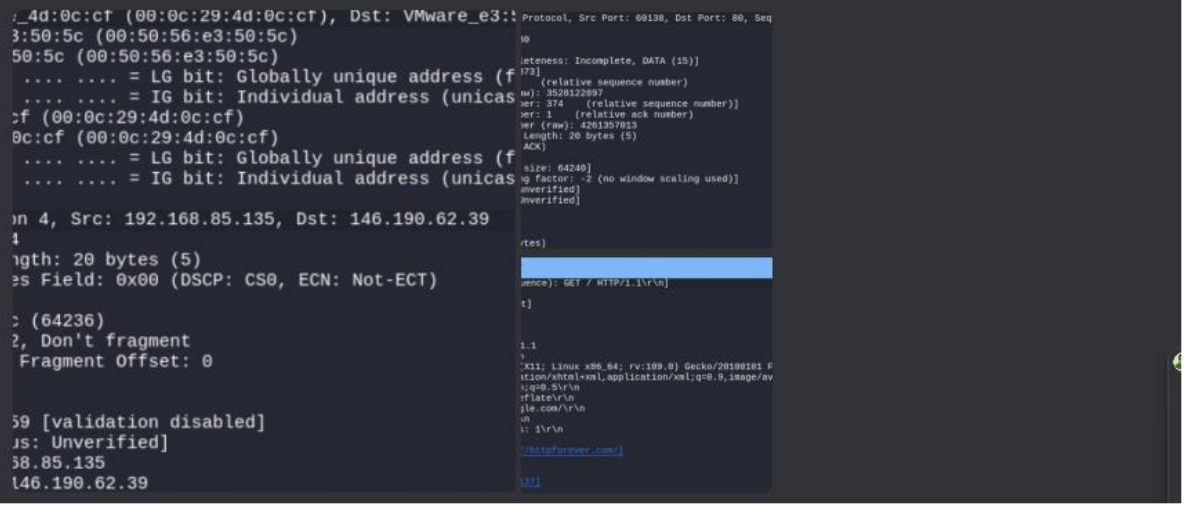
**Transmission Control Protocol (TCP):**

Source Port: This identifies the application on the sending device that originated the communication.
Destination Port: This identifies the application on the receiving device that should receive the data.
Window: This field specifies the amount of data the sender is willing to receive without an acknowledgment.

**Request Line (if request):**

This line specifies the HTTP method (e.g., GET, POST), requested resource (URL path), and HTTP version.
Status Line (if response): This line includes the HTTP version, status code (e.g., 200 OK, 404 Not Found), and a reason phrase
Headers: These fields provide additional information about the request or response (e.g., Host, Content-Type, User-Agent).
Message Body: This section contains the actual data being transferred (e.g., webpage content, image data). (edited)

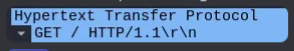
💯 1




**mabrook** Today at 7:41 PM

e. What is the HTTP version your browser used?

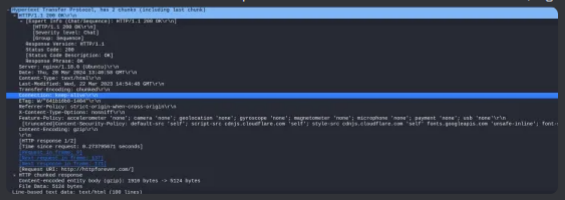
Ans : In the above image, the HTTP version used by the browser is HTTP/1.1. This can be seen from the line "GET / HTTP/1.1" shown in the "Hypertext Transfer Protocol" section at the top of the image.




💯 3

f. Identify the response message received from the server?

Ans : The response message received from the server is HTTP/1.1 200 OK. This can be seen from the line "Expert Info (Chat/Sequence): HTTP/1.1 200 OK" in the image, which indicates that the server responded with an HTTP status code 200 OK, signifying a successful response.



💯 3

**ipiot** Today at 7:42 PM

g. What is version of HTTP that the server is running?
h. Can you identify the IP address of your computer?
i. Can you identify the IP address of http://httpforever.com/ server? (edited)

```
TTP/1.1 200 OK\r\n
urce Address: 192.168.85.135
stination Address: 146.190.62.3
```

> **HTTP Forever**
> A site that will always be available over HTTP!

💯 2 🔥 1

**Baxy** Today at 7:46 PM

j. What is the status code returned from the server to your browser?

**Answers:**

The image showed a response packet with a status code of 200 OK. This indicates that the server successfully fulfilled the client's request. (edited)

```
Hypertext Transfer Protocol, has 2 chunks (including last chunk)
  HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      [HTTP/1.1 200 OK\r\n]
      [Severity level: Chat]
      [Group: Sequence]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
```

🔥 2

**Amjad** Today at 7:46 PM

k. When was the HTML file that you are accessing last modified at the server?

```
Last-Modified: Wed, 22 Mar 2023 14:54:48 GMT\r\n
```

💯 3

**mabrook** Today at 7:51 PM

Guys What do u think happens when you use "https"? Do u think you can analyze the responses?

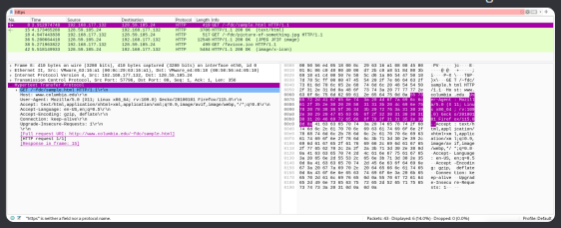
🔥 2

**Baxy** Today at 7:56 PM

I think it is quite impossible to analyze the "https" because of encryption:

When you use https, the communication between your browser and the website is encrypted using Secure Sockets Layer (SSL) or its successor, Transport Layer Security (TLS). This encryption scrambles the data being sent, making it unreadable to anyone intercepting it. This protects sensitive information like passwords, credit card details, and login credentials from eavesdroppers.

In the screenshot i included shows that we can't filter those traffic using Wireshark. there is a red line showing an error



💯 1

**H44mid** Today at 7:57 PM

Obtaining a server's private key without proper authorization raises significant ethical and legal concerns. It violates principles of confidentiality, data privacy, and undermines the trust and integrity of secure communication systems. While tools like Wireshark are valuable for network analysis, attempting unauthorized access to private keys may constitute illegal activities such as computer crime or data breach. Prioritizing ethical conduct, complying with relevant laws, and exploring authorized or simulated environments for educational purposes is crucial. In professional settings, strict protocols and oversight should govern any authorized access to sensitive information. Ultimately, advancing cybersecurity expertise must be balanced with a firm commitment to ethics, respect for the law, and responsible handling of sensitive data.

💯 2 🔥 2

**mabrook** Today at 8:00 PM

Using wireshark i guess its difficuilt in an ethical manner, it may be possible with using some specialized tools and techniques, but it is generally considered as unethical and illegal to do without the proper consent or authority. Since when a client makes an HTTPS request, the server responds with its SSL/TLS certificate, which contains the server's public key. The client verifies the certificate's validity and, if valid, establishes a secure connection using the server's public key for encrypting the subsequent communication.

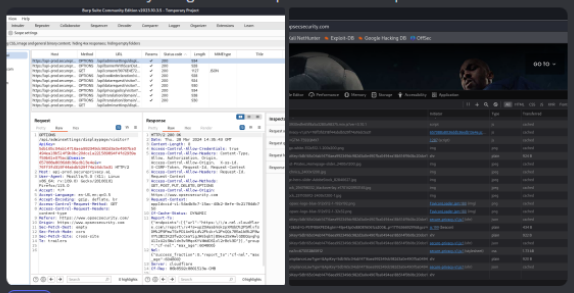
Another possible way to analyse it to consult with cybersecurity professionals or legal experts to ensure compliance with relevant laws and ethical standards.

As a legal matter it is not possible to analyse because, of some  benifits such as  confidentiality, integrity and authentication etc.

💯 3 🔥 1

**ipiot** Today at 8:08 PM

There is one more option for us as that we could use alternative tools or methods. we can go with similar tools such as, burp suit, browser developer tools and more. These are similar way using web developer tools and burp suit.



💯 1

**Amjad** Today at 8:09 PM

I also think that we cannot capture because the site is encrypted

Privacy and Security Implications: Discuss the privacy and security implications of analyzing HTTPS traffic, even if you have the server's private key. Explore the potential risks and ethical considerations involved in intercepting and decrypting encrypted traffic, as well as the laws and regulations governing such activities.

👍 1

**mabrook** Today at 8:13 PM

Great, Guys now lets drop a summary of evidence as of what we have learned today in this active class.

## Summary of what we have discussed in todays active session.

**mabrook** Today at 8:13 PM
Great, Guys now lets drop a summary of evidence as of what we have learned today in this active class.

**ipiot** Today at 8:14 PM
This activity aims to understand the application layer protocols, particularly HTTP, and analyze HTTP traffic using Wireshark.

Activity 1: Understanding the Application Layer using HTTP

Discusses the need for application layer protocols and provides examples.
Explores the principal operation of HTTP through a role-play activity, with group members acting as clients and servers, exchanging HTTP messages.
Introduces the concept of a web cache/proxy server and repeats the role-play to understand its role.

Activity 2: Analyzing HTTP in Wireshark

Opens a web browser, clears the cache, and captures packets using Wireshark while accessing a website over HTTP.
Filters the captured traffic for HTTP and analyzes various aspects of the HTTP requests and responses, such as message sequences, connection persistence, message details (GET, version, IP addresses, status codes, last modified time).
Repeats the analysis by accessing a different web page.
Discusses what happens when using HTTPS and whether the responses can be analyzed.
The activities provide a hands-on approach to understanding the application layer, specifically the HTTP protocol, and how to analyze HTTP traffic using a packet sniffer like Wireshark. It covers essential concepts such as client-server communication, message formats, connection types, and the role of web caches/proxy servers.
💯 2 🔥 2 🐄 1

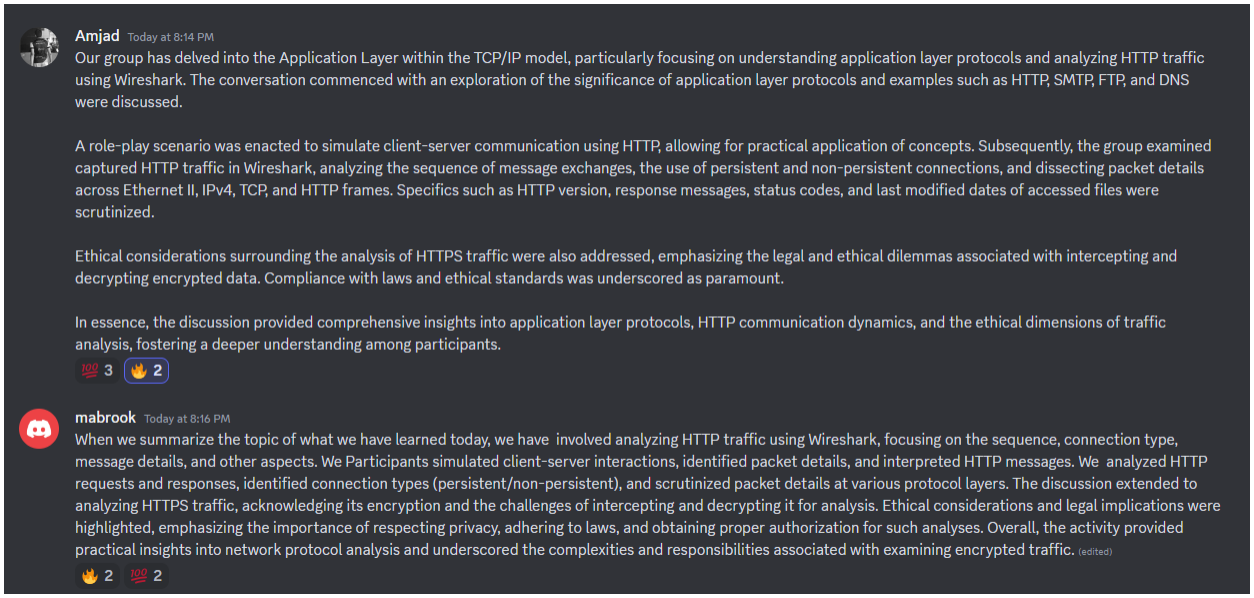
**Baxy** Today at 8:14 PM
Summary of Learning:
Throughout this activity, I have deepened my understanding of the application layer protocols, particularly focusing on HTTP (Hypertext Transfer Protocol), and practiced analyzing HTTP interactions using Wireshark. Here's how I've achieved each of the learning objectives:

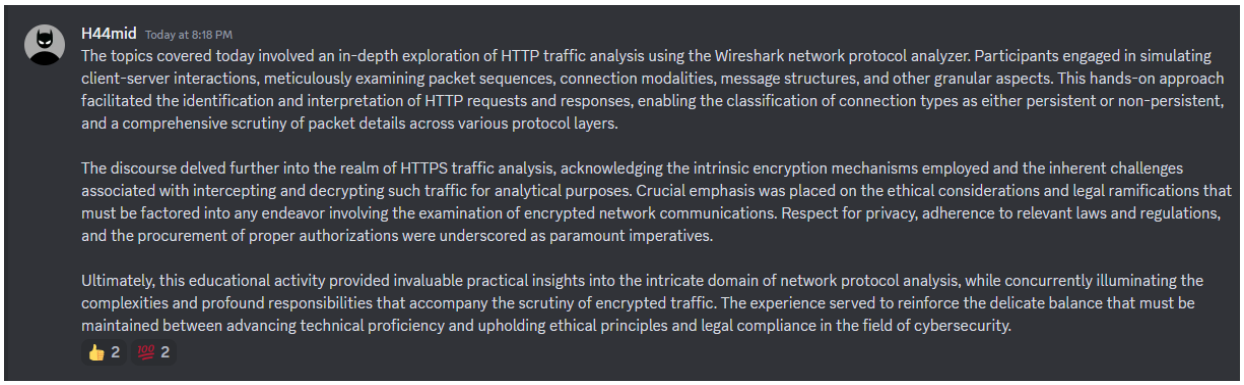
Explain the role of application layer protocols play in computer networking:
Through discussions and role plays with my peers, I have learned that application layer protocols serve as the interface for communication between software applications over a network. These protocols define the rules and conventions for data exchange, ensuring compatibility and interoperability. We discussed examples such as HTTP, SMTP, FTP, and DNS, understanding their importance in facilitating various internet applications.

Demonstrate the understanding of HTTP:
By participating in role plays and analyzing HTTP interactions using Wireshark, I have gained a deeper understanding of how HTTP operates. I learned that HTTP is a client-server protocol used for fetching resources from web servers. It operates over TCP/IP and follows a request-response model, where clients send requests for resources, and servers respond with the requested content. Analyzing HTTP messages in Wireshark provided insights into the structure of HTTP requests and responses, including details such as HTTP version, status codes, headers, and content.

Evidence of Learning:
These screenshots serve as evidence of my understanding of application layer protocols and HTTP.

Overall, this activity has been instrumental in enhancing my knowledge of computer networking concepts, particularly regarding the role of application layer protocols and the operation of HTTP. Through collaborative learning and practical analysis, I feel better equipped to apply these concepts in real-world scenarios.
👍 2 🔥 2




**Overall, we as a group of 5 discussed the 2 activities using discord and analyzed a specific http and https website by capturing the packets using Wireshark tool and explained each questions and relevant answers in depth and how these often works.**

**Members of the group.**

- **Nirosh**
- **Iflal**
- **Mabrook**
- **Amjad**
- **Haamid**

**Nirosh Ravindran**
**BSCP|CS|63|134**

# Module 2: Evidence of Learning

## Evidence: Active Class 3: It's always DNS, No It's 192.168.1.2

**Activity 1**:

Activity 1 was about answering 4 questions about the DNS server in transport layer of the TCP/IP model. We need to discuss the questions in the discord online group. As we go down further, we were asked to analyse our browser send HTTP traffic to the DNS server, deakin.edu.au. for this particular activity we need to do a roleplay on how the browser is sending and receiving the HTTP traffic through the DNS server to get us the browser page loaded.

**Activity 2:**

In activity 2 we need to send DNS queries to different webservers in different continents to analyse the DNS queries and response using nslookup and Wireshark. I had to use the Wrexham.ac.uk and Instagram.com. Then answered few questions regarding the capture and analysis of those traffic.

**Baxy** Today at 9:03 PM
Bonjour Amigos! Who here gets along with their DNS provider? No? Well, buckle up, because today we're going to explore the wonderful world of DNS in a way that WON'T put you to sleep! In fact, we might even have some laughs along the way.

So, whether you're a seasoned network pro or just starting to learn the ropes , join us as we delve into the fascinating world of DNS! Get ready for some fun, facts, and maybe even a few surprises.

First, we will conduct a group discussion and a role play to understand the role and the process of DNS. Then, you will use the nslookup and Wireshark to analyse the DNS protocol.

Let's begin our adventure >>>>
**Activity 1**
So let's begin our adventure by answering some questions (edited)
1. What is the core Internet function provided by DNS?

**ipiot** Today at 9:07 PM
core funcinality of DNS is resolving domain names into ip addresses, simply it acts as a translator in the internet.
4

**Baxy** Today at 9:07 PM
The core internet function provided by DNS is address translation. Computers communicate on the internet using numerical IP addresses (142.251.10.93), but humans find it easier to remember names like "YouTube.com" DNS acts like a giant phonebook for the internet, translating these domain names into the corresponding IP addresses that allow your computer to connect to the right website. (edited)
3 3

**mabrook** Today at 9:11 PM
DNS (Domain Name System) acts as the internet's address book, converting human-readable website names (like google.com) into machine-readable IP addresses. When users type a website name into their browser, DNS servers translate it to the corresponding IP address, facilitating communication between devices on the internet. This system, distributed globally, stores extensive databases linking domain names to IP addresses, ensuring smooth navigation online. Essentially, DNS enables seamless access to websites and services, simplifying the user experience on the internet. (edited)
4 4

**Amjad** Today at 9:13 PM
Domain Name System helps computers find websites. It changes website names into numbers (like 172.217.3.46) that computers can understand. So when you type a website name in your browser, DNS helps your computer know where to go. It's like a phonebook for the internet. Without DNS, you'd have to remember lots of numbers to visit websites, which would be hard. But with DNS, you can just remember the easy names, and it takes care of the rest, making it simple for everyone to use the internet. (edited)
4

**H44mid** Today at 9:17 PM
DNS, or Domain Name System, provides the core function of translating human-readable domain names (like "xhorse.com") into numerical IP addresses that computers use to communicate over the Internet. This translation is crucial for users to access websites and services easily without needing to remember long strings of numbers.
2 1

**Baxy** Today at 9:18 PM
Lets look into the next ques
2. Why do we need DNS?

**H44mid** Today at 9:19 PM
Think of DNS like a phonebook for the internet. Without DNS, we would have to remember long strings of numbers (IP addresses) to visit websites or use online services, which would be very inconvenient. DNS simplifies this by allowing us to use easy-to-remember domain names (like "google.com") instead of numbers. It's like how you don't need to remember someone's phone number if you have their name in your contacts. DNS makes it easier for us to navigate the internet and find what we're looking for quickly.

**ipiot** Today at 9:19 PM
it is easy for a user to type a domain name other than an ip address
DNS increase performace by loadbalancing the traffic towards a website.
Using DNS we could implement security features like blacklisting, so that we couyld prevent melicious actors.
2

**Baxy** Today at 9:20 PM
Remembering long strings of numbers (IP addresses) would be incredibly difficult. DNS translates these complex IP addresses into user-friendly domain names we can easily remember and type.

Websites can change servers (locations) without affecting users. DNS automatically updates the address translation so users always reach the correct destination.

The internet is a massive, decentralized network. DNS helps route information efficiently by translating domain names regardless of the physical location of the server.
3

**mabrook** Today at 9:21 PM
We need DNS because it changes website names into numbers that computers can understand. Without it, we'd have to remember lots of numbers to visit websites, which would be hard. DNS makes using the internet easier by letting us remember easy names instead of numbers.
2

**Amjad** Today at 9:23 PM
DNS (Domain Name System) is crucial because it acts as the internet's directory, translating easy-to-remember website names into the numerical IP addresses required for communication between devices. Without DNS, navigating the internet would be akin to trying to find locations without addresses, making it significantly more challenging and less user-friendly.
3 3

**Amjad** Today at 9:28 PM
Using just one DNS server for the whole internet is risky because if something happens to that server, the whole internet could stop working. It's like having only one road to get to a place—if it's blocked, you're stuck.

A better idea is to have lots of DNS servers spread out in different places. This way, if one server has a problem, the others can still help people find websites. It's like having many different roads to get to the same place, so if one road is closed, you can still use another one to reach your destination.

👍 3

**mabrook** Today at 9:29 PM
Depending solely on one DNS server for the entire network is little dangerous. If that server fails, all connections could be lost, like a single light switch controlling all the lights in a house—if it breaks, there's no light. An alternative is a redundant system with multiple DNS servers. These servers work together, ensuring that if one fails, others are available to take over. It's like having backup generators—if one fails, another kicks in to keep the power running. This setup increases reliability and ensures continuous internet access even if one server encounters problems. (edited)

🔥 3 ✌ 2 💯 2

**Baxy** Today at 9:30 PM
As we have now completed the Q & A part, we shall now lets move into the next section.

Discuss the steps involved in your browser to send a HTTP request message to the Web server, deakin.edu.au. Assume that this is the first time you access this webpage.

For this discussion lets do a role play

**Baxy** Today at 9:40 PM
Now lets decide our roles in this

but before that lets see the scenario

a. One group member can act as the web server, another as DNS servers (you need to decide how many DNS serves will be involved), and the remaining members can be the clients.

b. First, Client 1 needs to view deakin.edu.au and initiate the conversation saying the right message to the right device (to the person who is acting as the right device). Use your knowledge about web browsing that we covered in Module 1 and the first part of Module 2. Assume that there is no DNS caching available.

c. All the devices need to respond to each other with the correct messages in the right sequence. Make sure you record all the steps as you will need those notes to complete the activity 2.

d. Now, after Client 1 accessed deakin.edu.au, Client 2 also needs to view deakin.edu.au. Discuss the steps involved in Client 2's web browser to be able to send a HTTP request message to the Web server.

e. You can show all the steps in a timing diagram with the end systems and numbering the sequence of steps (similar to the activity you did in Active Class (edited)

**ipiot** Today at 9:44 PM
Ill go with the web server as my role play.

**Baxy** Today at 9:44 PM
I'm root dns server (edited)

**mabrook** Today at 9:44 PM
ill pick the DNS role (edited)

**H44mid** Today at 9:45 PM
I'll take client 1 (edited)

**Baxy** Today at 9:46 PM
so let's begin

**H44mid** Today at 9:50 PM
**Client 1** *(To DNS Server)*: I want to access the Deakin University website. (edited)

**mabrook** Today at 9:51 PM
**DNS SERVER 1** (to root DNS server) : I don't have the IP address for "deakin.edu.au" in my cache. I need to forward the request to the root DNS server. (edited)
Please provide the IP address for "deakin.edu.au".

**Baxy** Today at 9:56 PM
**ROOT DNS Server** (*to DNS Server 1*)**:** Ohhh my poor little DNS, you still have to learn a lot, ok will provide you with the IP Address of "deakin.edu.au", which is 123.45.67.89 (edited)

**mabrook** Today at 9:58 PM
**DNS SERVER 1** (to client 1) : I received the IP address for "deakin.edu.au". Here is the IP address 123.45.67.89. (edited)

**H44mid** Today at 9:59 PM
**Client 1** *(To Web Server)*: Please send me the webpage content for deakin.edu.au at IP address 123.45.67.89. (edited)

**ipiot** Today at 10:09 PM
**Web Server** *(to Client 1)* Responding with the web page for Client 1 (edited)

**H44mid** Today at 10:10 PM
**Client 1**: Received the web page from the webserver and Displaying the web page content

**Baxy** Today at 10:19 PM
Now lets move on to the next activity

**Activity 2**

in this activity we need to analyze the DNS traffic for different websites using the nslookup and WIreshark (edited)

**ipiot** Today at 10:26 PM
These are the outputs i recived from the nslookup command,
while comparing both the screenshot i get few more authoritative answers the american website which is the "stanford.edu". (edited)

```
nslookup deakin.edu.au
;          192.168.1.1
;          192.168.1.1#53

thoritative answer:
  deakin.edu.au
s: 128.184.20.21
  deakin.edu.au
s: 128.184.204.21
  deakin.edu.au
s: 2402:6940:1201:2022:128:184:233:77
  deakin.edu.au
s: 2402:6940:1401:2024:128:184:237:77
  deakin.edu.au
s: 2402:6940:1201:2023:128:184:235:77
  deakin.edu.au
s: 2402:6940:1401:2025:128:184:239:77
```

```
ookup -type=ns stanford.edu
      192.168.1.1
      192.168.1.1#53

itative answer:
du    nameserver = ns5.dnsmadeeasy.com.
du    nameserver = avallone.stanford.edu.
du    nameserver = atalante.stanford.edu.
du    nameserver = ns7.dnsmadeeasy.com.
du    nameserver = ns6.dnsmadeeasy.com.
du    nameserver = argus.stanford.edu.

ive answers can be found from:
deeasy.com     internet address = 208.94.14
deeasy.com     internet address = 208.80.12
deeasy.com     internet address = 208.80.12
deeasy.com     has AAAA address 2600:1000:5
deeasy.com     has AAAA address 2600:1801:6
deeasy.com     has AAAA address 2600:1802:7
```

**Baxy** Today at 10:42 PM
This is the output for the website **wrexham.ac.uk**
The IP address for wrexham.ac.uk is 194.82.118.123.
There are 3 Name Servers I got:
--> dns01.glyndwr.ac.uk
--> attis.glyndwr.ac.uk
--> dns02.glyndwr.ac.uk

Additionally I received this as well

responsible mail addr = hostmaster.glyndwr.ac.uk
serial = 2013060482
refresh = 14400 (4 hours)
retry = 3600 (1 hour)
expire = 864000 (10 days)
default TTL = 86400 (1 day)

The authoritative nameservers for wrexham.ac.uk are not shown (non-authoritative answer).

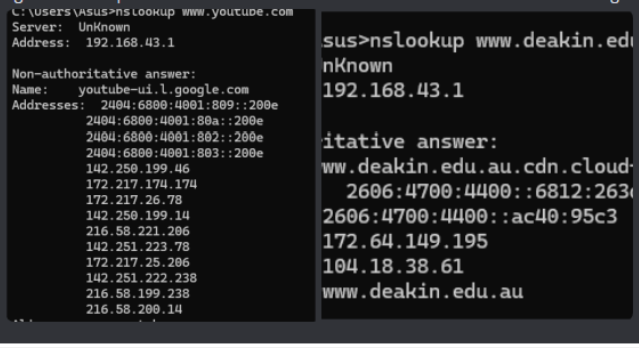
**Authoritative vs. Non-authoritative Answers:**
Authoritative answers come directly from the DNS servers responsible for a specific domain (like the registrar's servers for wrexham.ac.uk). They are the most reliable source.

Non-authoritative answers come from caching servers that have temporarily stored information about a domain lookup. While they can be faster, they might not be the most up-to-date information.

```
C:\Users\baxy2>nslookup -type=ns wrexham.ac.uk
Server:  UnKnown
Address:  fe80::1

Non-authoritative answer:
wrexham.ac.uk   nameserver = dns01.glyndwr.ac.uk
wrexham.ac.uk   nameserver = attis.glyndwr.ac.uk
wrexham.ac.uk   nameserver = dns02.glyndwr.ac.uk

attis.glyndwr.ac.uk     internet address = 194.82.118.123
dns01.glyndwr.ac.uk     internet address = 194.82.119.5

C:\Users\baxy2>nslookup -type=ns www.wrexham.ac.uk
Server:  UnKnown
Address:  fe80::1

wrexham.ac.uk
        primary name server = dns01.glyndwr.ac.uk
        responsible mail addr = hostmaster.glyndwr.ac.uk
        serial  = 2013060482
        refresh = 14400 (4 hours)
        retry   = 3600 (1 hour)
        expire  = 864000 (10 days)
        default TTL = 86400 (1 day)

C:\Users\baxy2>
```

**mabrook** Today at 10:43 PM

I got these outputs for the Australian website and the American websites using nslookup and we could see the difference of the address




**Baxy** Today at 10:48 PM

this is the other website, instagram.com

```
C:\Users\baxy2>nslookup -type=SOA instagram.com
Server:  UnKnown
Address:  fe80::1

Non-authoritative answer:
instagram.com
        primary name server = a.ns.instagram.com
        responsible mail addr = dns.facebook.com
        serial  = 2395763829
        refresh = 14400 (4 hours)
        retry   = 1800 (30 mins)
        expire  = 604800 (7 days)
        default TTL = 3600 (1 hour)

C:\Users\baxy2>
```

**Amjad** Today at 10:53 PM

I have provided the 2 different websites from different countries, with the non authorative answers.

**H44mid** Today at 10:53 PM

Based on the additional information you provided and the screenshots, here's an updated analysis:

For the domain "www.harvard.edu", the nslookup command shows both non-authoritative and authoritative answers:

Non-authoritative answer:
Aliases: map.fastly.net
Addresses: 2a04:4e42:48::e45
199.232.46.133
Aliases: www.harvard.edu

Authoritative answer:
Name: www.harvard.edu
Addresses: 2607:fb90:180e:cbaa:d6e0:d5c7:f61d:6e9
199.232.46.133

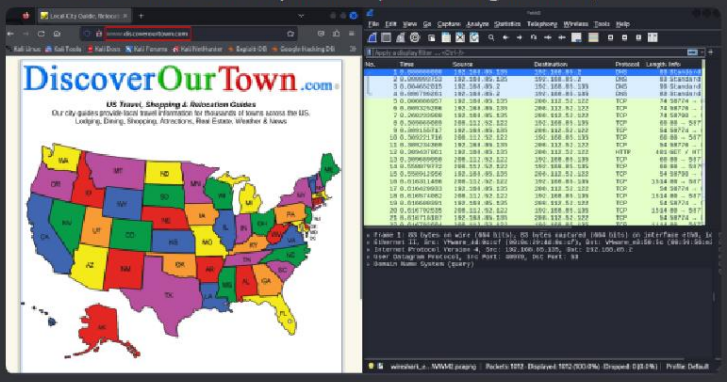
The authoritative answer is provided directly by the authoritative DNS servers for the harvard.edu domain, giving the canonical IP addresses associated with www.harvard.edu.

```
sers\cex> nslookup abc.ne
  UnKnown
:  fe80::1

horitative answer:
  abc.net.au
:  23.195.152.93
```

```
ers\cex> nslookup www.harvard.
  UnKnown
  fe80::1

horitative answer:
  pantheon-systems.map.fastly.n
es:  2a04:4e42:48::645
  199.232.46.133
  www.harvard.edu
```

**ipiot** Today at 10:55 PM

Let's trace DNS in Wireshark now,
Below is website loaded and captured the packets using wireshark,




**Baxy** Today at 10:56 PM

now lets analyze, for that we need to answer a few questions to reach our objective

a. Find the DNS query and response messages. Which transport layer protocol they have used? UDP or TCP?

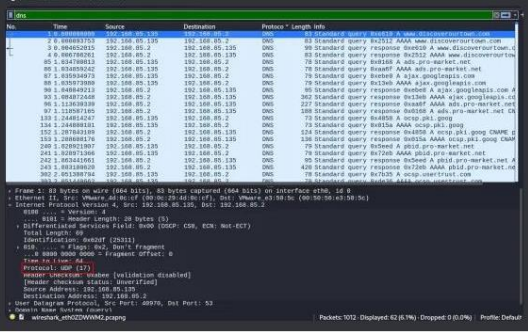
**H44mid** Today at 11:00 PM

The main section shows a list of captured packets, with details such as the time, source and destination IP addresses, protocol (DNS in this case), length, and additional information about the DNS queries and responses being made.

Some of the DNS queries seen are for domains like "discoverourtown.com", "pro-market.net", "googleapis.com", "ocsp.pki.goog", and others. These queries are likely being made by applications or services on the local machine to resolve domain names to IP addresses for establishing connections.

The bottom section provides more detailed information about one specific packet, displaying the IP and UDP headers, flags, and other protocol-level details.

Overall, this output from a network analyzer tool allows administrators or security professionals to inspect and analyze the DNS traffic and other network activity on a system or network.



**Baxy** Today at 11:01 PM

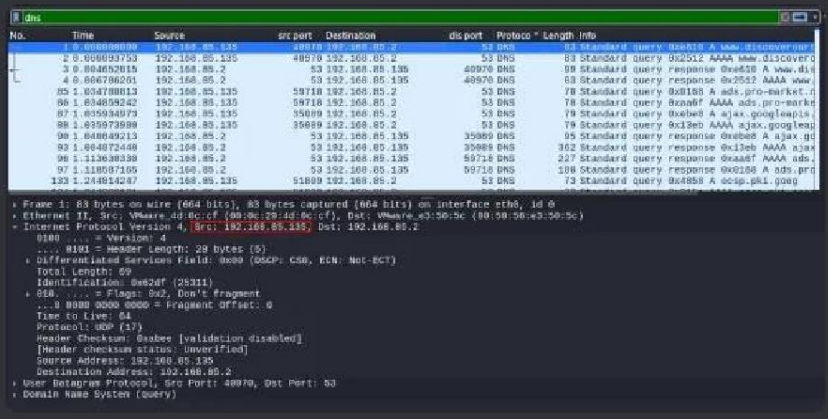
b. What are the destination port of the DNS query message and the source port of DNS response message?

The destination port for the DNS query message is 53 and the source port of the DNS response message is also 53. This is the standard configuration for DNS communication. Port 53 is commonly known as the DNS port.


**Amjad** Today at 11:03 PM

c. What is the IP address that the DNS query message was sent?

The ip address is 192.168.85.135 (edited)




**mabrook** Today at 11:06 PM

d. Identify the IP address of your local DNS server using the terminal/command prompt. Are these two IP addresses identified in c and d the same?

Here The command "search localdomain nameserver 192.168.85.2" is being executed to configure the local DNS server address in the /etc/resolv.conf file. This IP address 192.168.85.2 matches the DNS server IP address seen in the previous network packet capture, which is responding to DNS queries from the device with IP 192.168.85.135.

Therefore yes, the IP addresses 192.168.85.2 identified in this command and in the previous packet capture are the same, referring to the local DNS server that the device is using to resolve domain names. (edited)

```
└─$ cat /etc/resolv.conf
# Generated by NetworkManager
search localdomain
nameserver 192.168.85.2
```

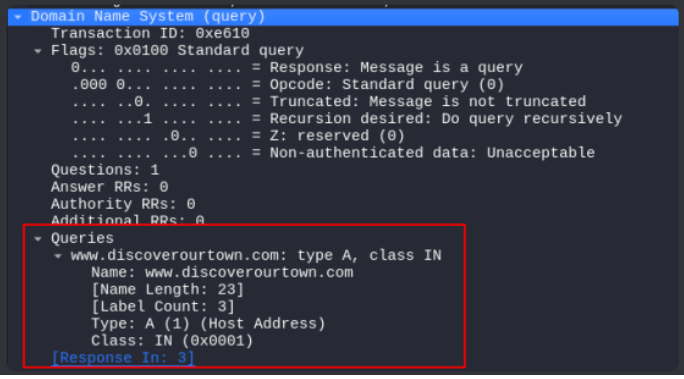
**ipiot** Today at 11:07 PM

e. You can further explore the DNS query message. Can you identify the “Type” of DNS query? What does the query message contain?

Based on information of the packet which is on the screenshot, the "Type" of DNS query is "A".

The query message contains:
www.discoverourtown.com: type A, class IN

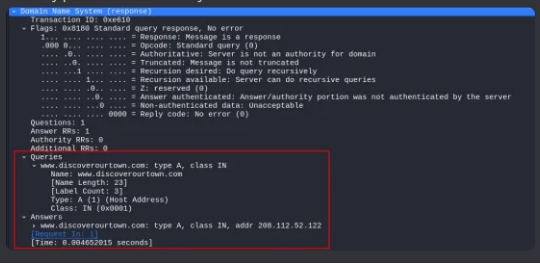
"www.discoverourtown.com" is the domain name being queried.
"type A" indicates that this is an IPv4 address record (A record) query.
"class IN" means the query is for the Internet (IN) class, which is the most common class used for public DNS queries. (edited)




**H44mid** Today at 11:10 PM

You can further explore the DNS response message. Can you identify the “Type” of DNS response?

The response, identified by the Transaction ID 0xe610, is a standard query response with no errors, indicating a successful resolution of a domain name to its corresponding IP address. The specific domain name being queried is "www.discoverourtown.com", and the DNS server has provided the IPv4 address 208.112.52.122 as the answer for this domain in the "Answers" section. Additional information is also provided, such as the query type (A record for IPv4), the response time (0.004652015 seconds), and various flags and fields related to the DNS protocol and the nature of the query and response. This detailed output allows network administrators and security professionals to analyze and troubleshoot DNS resolutions and traffic on their networks.



**Baxy** Today at 11:13 PM

g. Are there any “answers” in the response message? If so, what are these answers?

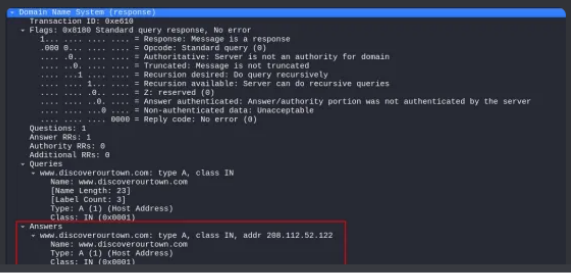
Yes, there are "Answers" sections in the response message shown in the image. The first "Answer" section contains the line:

"www.discoverourtown.com: type A, class IN, addr 208.112.52.122"

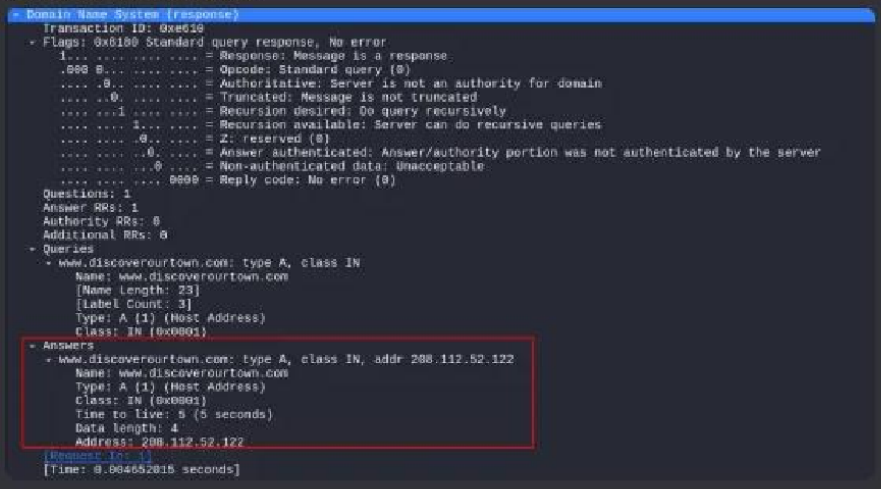
This provides the IP address 208.112.52.122 for the domain www.discoverourtown.com.

The second "Answer" section repeats the same information:

"Name: www.discoverourtown.com
Type: A (1) (Host Address)
Class: IN (0x0001)
Time to live: 5 (5 seconds)
Data length: 4
Address: 208.112.52.122"




"Name: www.discoverourtown.com
Type: A (1) (Host Address)
Class: IN (0x0001)
Time to live: 5 (5 seconds)
Data length: 4
Address: 208.112.52.122"



**ipiot** Today at 11:13 PM

h. The web page that you have accessed contains a couple of images. Does the host request new DNS queries to access each image? Explain your answer.

Based on the captured packets using wireshark it appears that no new DNS queries are being made to access individual images on the web page.
The DNS queries and responses shown in the capture seem to be related to resolving the domain names of the website itself (www.discoverourtown.com) and various other domains like ads.pro-market.net, ajax.googleapis.com, google-analytics.com, and pbid.pro-market.net.
once the domain name of the website is resolved, the web browser can request and load individual resources (like images, CSS files, JavaScript files) from the same domain without needing to perform additional DNS queries. The URLs for these resources are relative to the domain, and the browser can construct the full URLs based on the resolved IP address of the domain.

Unless the images are hosted on separate domains or subdomains, the browser would not need to perform new DNS queries specifically for each image. It can use the already resolved IP address of the main website domain to request and load the images.

```
Standard query 0x37c6 A www.discoverourtown.com
Standard query 0xe7c1 AAAA www.discoverourtown.com
Standard query response 0x37c6 A www.discoverourtown.com A 208.112.52.122 NS bdns.au…
Standard query response 0xe7c1 AAAA www.discoverourtown.com SOA bdns.aus.siteprotect…
Standard query 0x4898 A ads.pro-market.net
Standard query 0x259c AAAA ads.pro-market.net
Standard query 0xfbb5 A ajax.googleapis.com
Standard query 0x41b4 AAAA ajax.googleapis.com
Standard query 0x8f0e A pbid.pro-market.net
Standard query response 0x8f0e A pbid.pro-market.net A 107.178.240.89
Standard query 0x2c0f AAAA pbid.pro-market.net
Standard query response 0x41b4 AAAA ajax.googleapis.com AAAA 2404:6800:4003:c11::5f …
Standard query response 0xfbb5 A ajax.googleapis.com A 142.251.175.95 NS ns3.google.…
Standard query response 0x259c AAAA ads.pro-market.net CNAME ads.pro-market.net.akam…
Standard query 0x2775 A www.google-analytics.com
Standard query response 0x4898 A ads.pro-market.net CNAME ads.pro-market.net.akamaiz…
Standard query response 0x2775 A www.google-analytics.com A 142.251.175.100 A 142.25…
Standard query 0xa472 AAAA www.google-analytics.com
Standard query response 0xa472 AAAA www.google-analytics.com AAAA 2404:6800:4003:c1c…
Standard query response 0x2c0f AAAA pbid.pro-market.net AAAA 2600:1901:0:8eee:: NS n…
Standard query 0x63a8 A www.google-analytics.com
Standard query 0x54ae AAAA www.google-analytics.com
Standard query response 0x63a8 A www.google-analytics.com A 142.251.175.139 A 142.25…
Standard query response 0x54ae AAAA www.google-analytics.com AAAA 2404:6800:4003:c1c…
Standard query 0x2f44 A ocsp.pki.goog
Standard query 0xa64c AAAA ocsp.pki.goog
```