CSC203L Computer Networks Lab

Submitted by:

Umair Arshad 2024-SE-38

Submitted to:

Prof. Noman Munir

Dated: November 21, 2025

Department of Computer Science

University of Engineering and Technology, New Campus

Wire Shark

1. HTTP GET

1.1 HTTP GET request

HTTP (HyperText Transfer Protocol) is an **application-layer protocol** used for communication between a web browser (client) and a web server. When a user enters a URL in the browser, the browser sends an **HTTP GET request** to the server to request a specific resource (such as an HTML file).

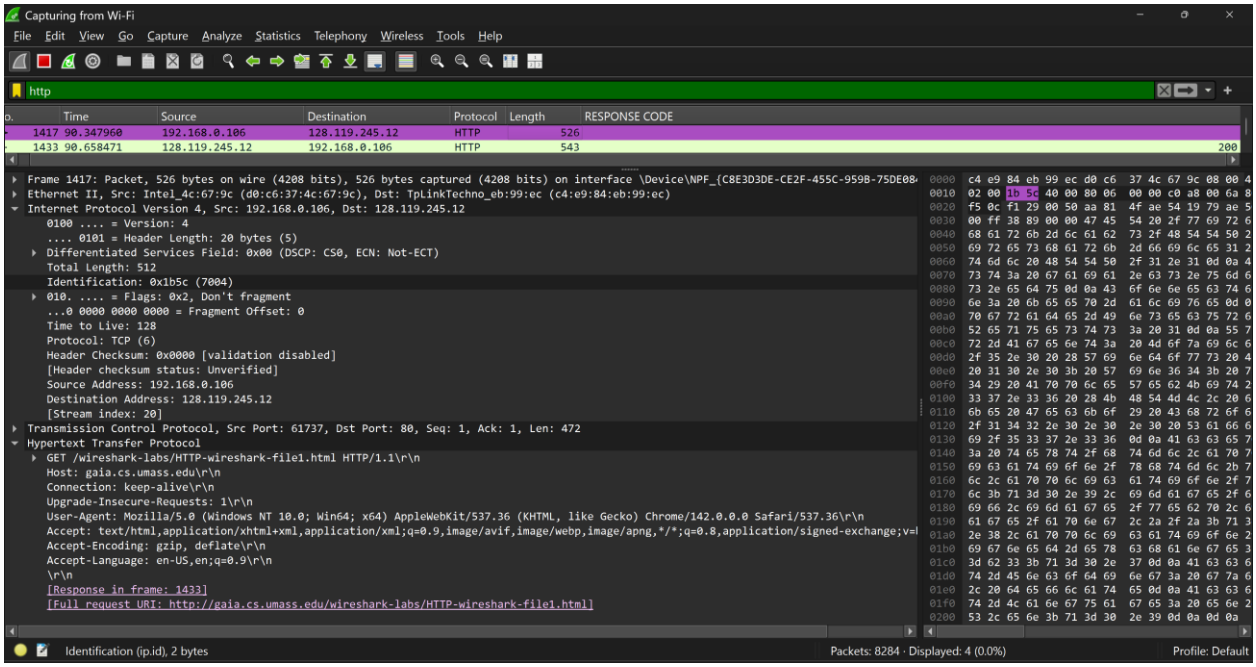
The GET request contains:

- The requested resource path
- HTTP version
- Request headers (Host, Accept-Language, User-Agent, etc.)

The server processes the request and sends back an **HTTP response**, which includes:

- A **status line** (version, status code, phrase)
- **Response headers**
- The requested data (HTML content)

Data Packet for reference:



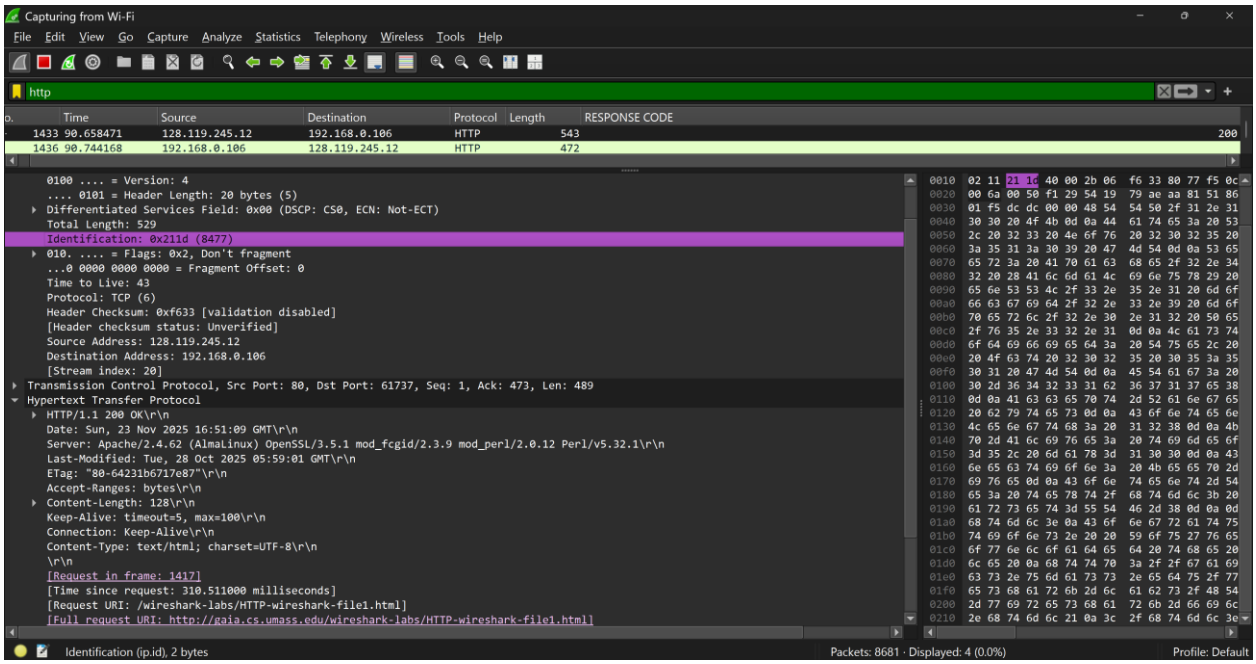
1.2 HTTP Get Response

An HTTP response message consists of three main parts:

1. **Status Line**
   Indicates whether the request was successful (e.g., 200 OK)
2. **Header Fields**
   Provide metadata such as:
   - Content-Length → size of the data
   - Last-Modified → last update time of the resource
   - Content-Type → type of content being sent
3. **Message Body**
   Contains the actual requested content (HTML file)

Wireshark allows verification of whether the full content is delivered by observing the presence of payload bytes and the Content-Length header.

Data Packet for Reference:



1.3 Questions and Answers

---

1. What is the IP address of your computer? Of the gaia.cs.umass.edu server?

- IP address of my computer: 192.168.0.106
- IP address of gaia.cs.umass.edu: 128.119.245.12

---

2. What HTTP version is your browser running? What version of HTTP is the server running?

- Browser HTTP version: HTTP/1.1
  (seen in GET line: GET /... HTTP/1.1)
- Server HTTP version: HTTP/1.1
  (seen in response: HTTP/1.1 200 OK)

---

3. What is the status code and phrase returned from the server to your browser?

- Status Code: 200
- Status Phrase: OK

---

4. What languages does your browser indicate to the server that it can accept? Which header line is used to indicate this information?

- Languages: en-US, en;q=0.9
- Header line: Accept-Language

---

5. When was the HTML file last modified at the server? Which header line is used to indicate this information?

- Last Modified Date: Tue, 28 Oct 2025 05:59:01 GMT
- Header line: Last-Modified

---

6. How many bytes of content (size of file) are returned to your browser? Which header line is used to indicate this information?

- Content Size: 1287 bytes
- Header line: Content-Length

2. HTTP Conditional GET

A **Conditional GET** is used to reduce unnecessary data transfer. After a resource is downloaded once, the browser stores it in cache. When requesting the same resource again, the browser sends an **If-Modified-Since** header containing the timestamp of the cached version.

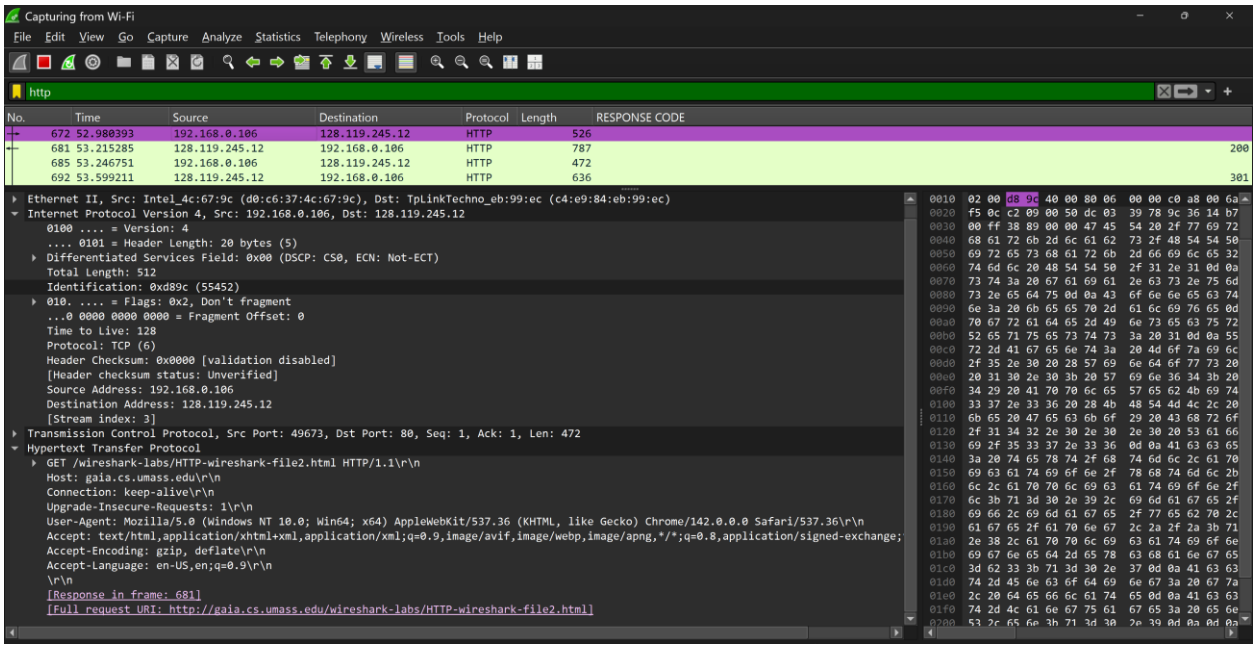
- If the file **has changed**, the server responds with 200 OK and sends the updated content.
- If the file **has not changed**, the server responds with 304 Not Modified and sends **no data**.

This mechanism improves:

- Network efficiency
- Page load speed
- Server performance

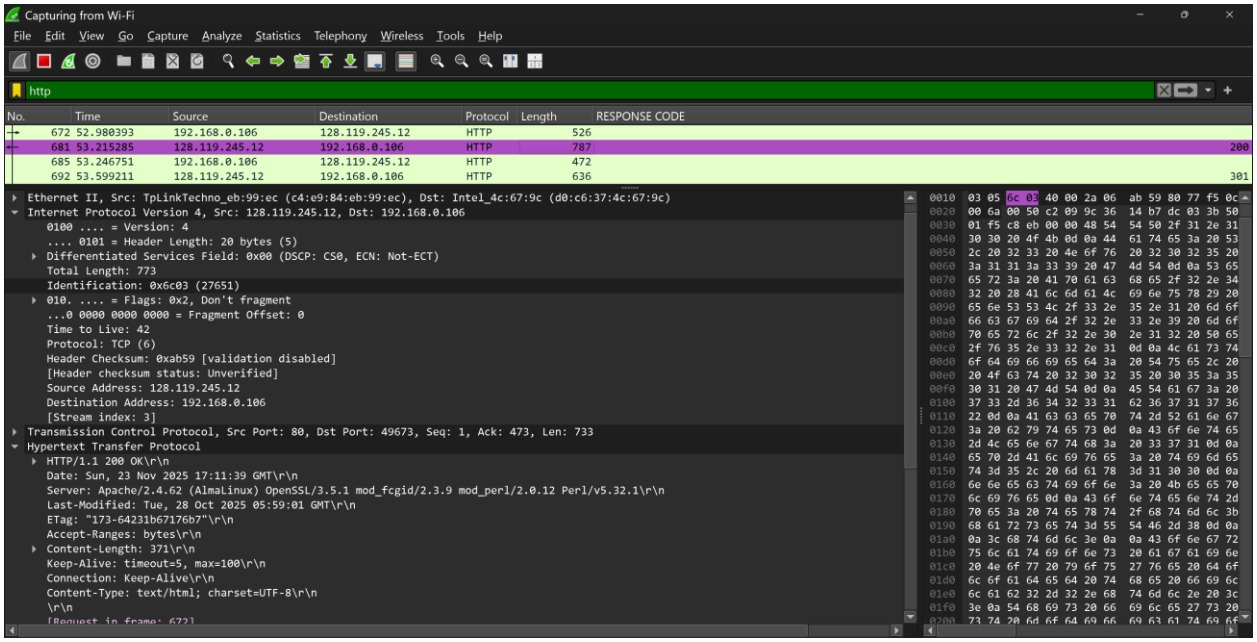
2.1 HTTP First GET Request

Data Packet for Reference:



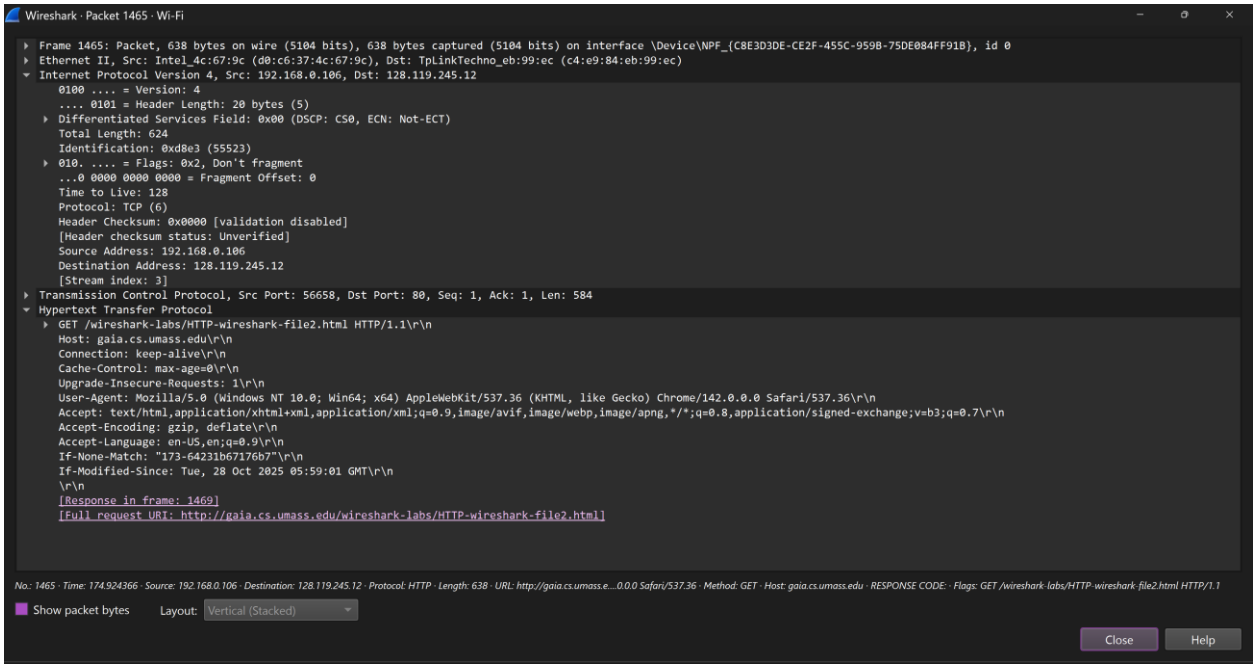
2.2 HTTP First GET Response

Data Packet for Reference:



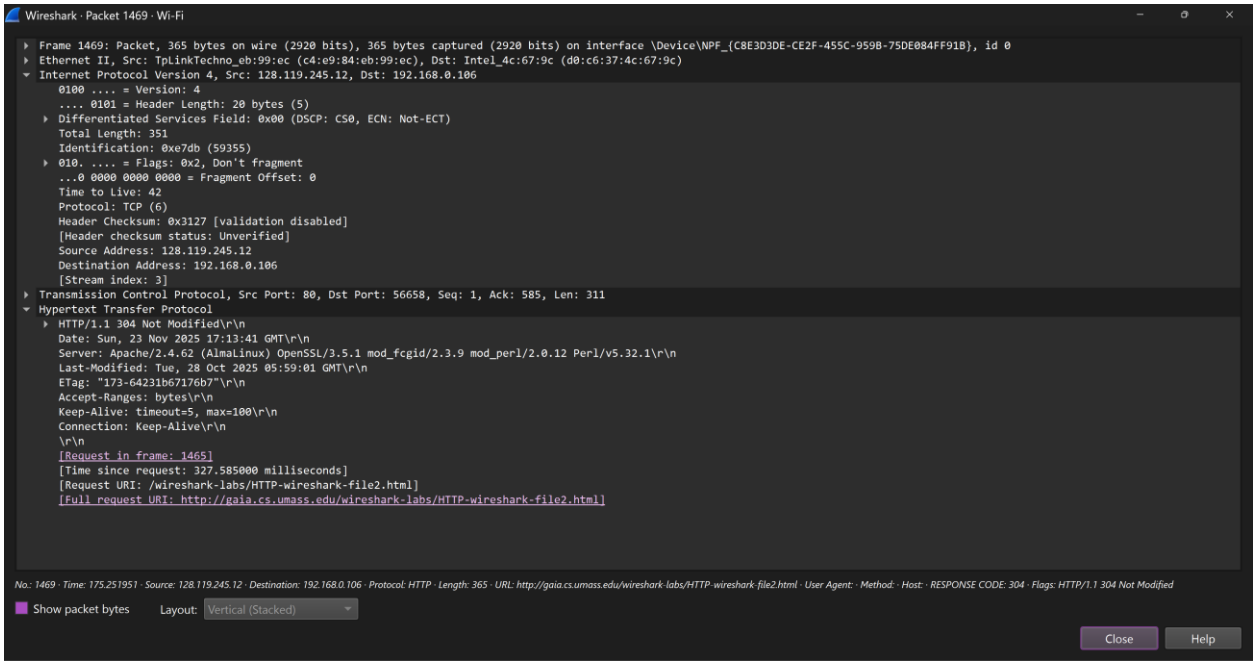
2.3 HTTP Second GET Request

Data Packet for Reference:



2.4 HTTP Second GET Response

Data Packet for Reference



2.5 Questions/Answers

---

1. Inspect the contents of the first HTTP GET request from your browser to the server. Is there an “IF-MODIFIED-SINCE” header line in the HTTP GET message? Why or why not?

Answer:
No, the first HTTP GET request does **not** contain an If-Modified-Since header.
This is because the browser is requesting the file **for the first time**, so it does not yet have a cached copy and therefore has no timestamp to send to the server.

---

2. Inspect the contents of the server response. Has the server explicitly returned the contents of the file? How can you tell?

Answer:
Yes, the server has explicitly returned the full file.
This is indicated by:

- The status code **200 OK**, which means the resource is being delivered normally.
- The header **Content-Length: 371**, showing that the file’s entire payload is included.
- The response packet size is large (full HTML page content visible).

---

3. Now inspect the contents of the second HTTP GET request from your browser to the server. Is there an “IF-MODIFIED-SINCE:” header line in the HTTP GET message? If so, what information follows it?

Answer:
Yes, the second GET request contains an If-Modified-Since header.
The exact value included is:

If-Modified-Since: Tue, 28 Oct 2025 05:59:01 GMT

This timestamp matches the Last-Modified time that the server sent in its response to the first GET request.

---

4. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Has the server explicitly returned the contents of the file? Explain.

Answer:
The server returns:

304 Not Modified

This means the file **has not changed** since the time provided in the If-Modified-Since header. Because of this, the server **does NOT send the file again**.

You can tell because:

- 304 Not Modified responses contain **no message body**.
- The packet size is very small.
- No Content-Length with a file size appears—only headers are returned.

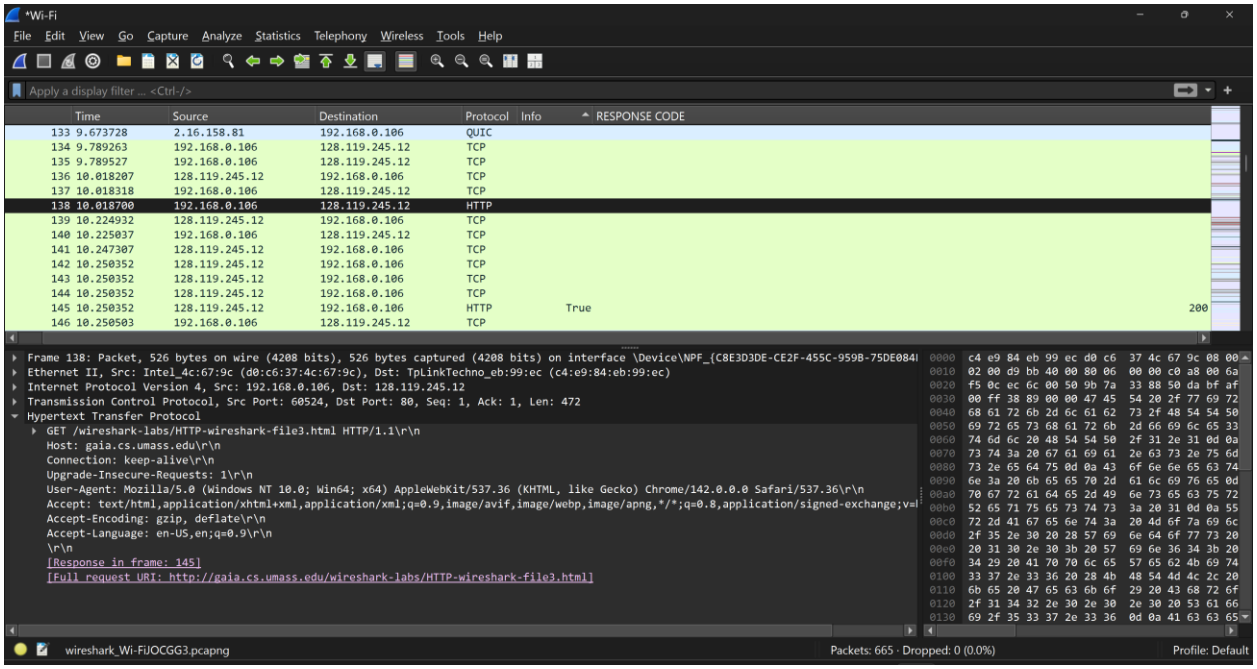
2.6 Retrieving Long Documents

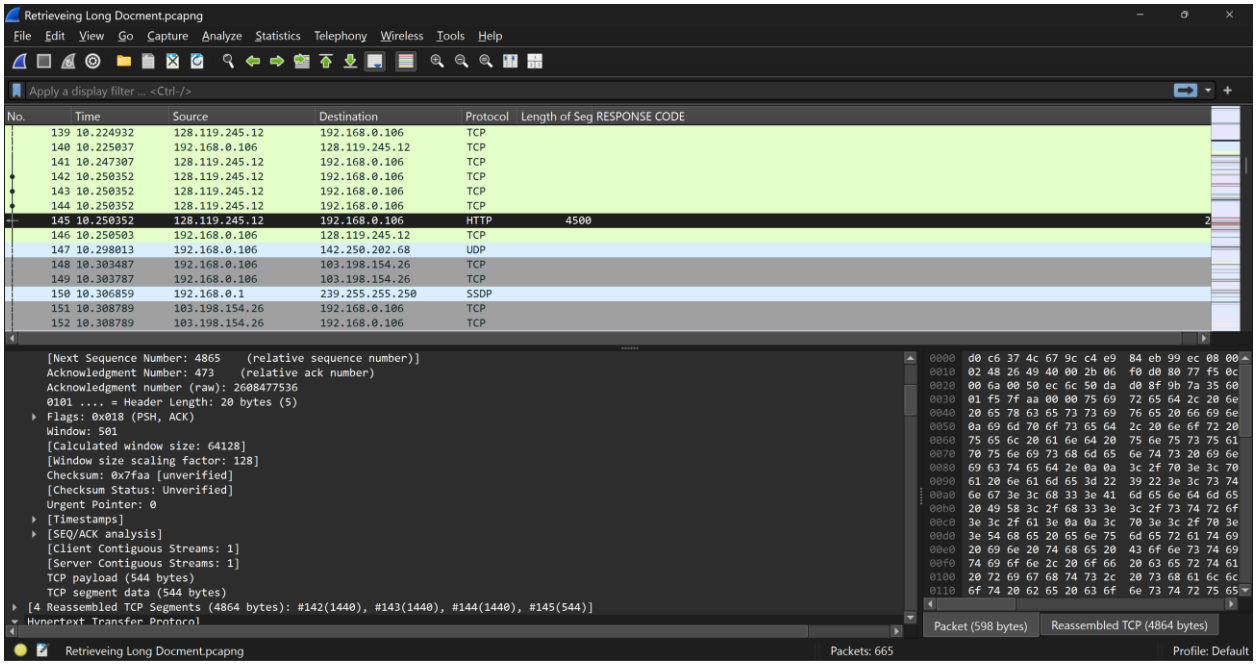
When an HTTP response is larger than the **Maximum Segment Size (MSS)** of TCP, the data is split into multiple TCP segments.
Key points:

- HTTP is unaware of segmentation
- TCP handles splitting and reassembly
- Wireshark shows “Reassembled TCP segments” for large files

Each TCP segment carries part of the HTTP payload, and the receiver reassembles them using sequence numbers.

Screen Shots of Data Packets
We need data packets from 138 to 145





11. How many HTTP GET request messages has your browser sent? Which packet contains the request?

Your browser sent **1 HTTP GET request**.
The GET request for the Bill of Rights is in **packet 138**.

---

12. Which packet contains the status code and phrase? What is the status code and phrase?

The HTTP status line appears in **packet 145**.
The status code and phrase are:

HTTP/1.1 200 OK

13. How many TCP segments carry the HTTP response text, and how many text bytes are in each?

Wireshark shows the response reassembled from **4 TCP segments**:

- **Packet 142:** 1440 bytes of text
- **Packet 143:** 1440 bytes of text
- **Packet 144:** 1440 bytes of text
- **Packet 145:** 544 bytes of text

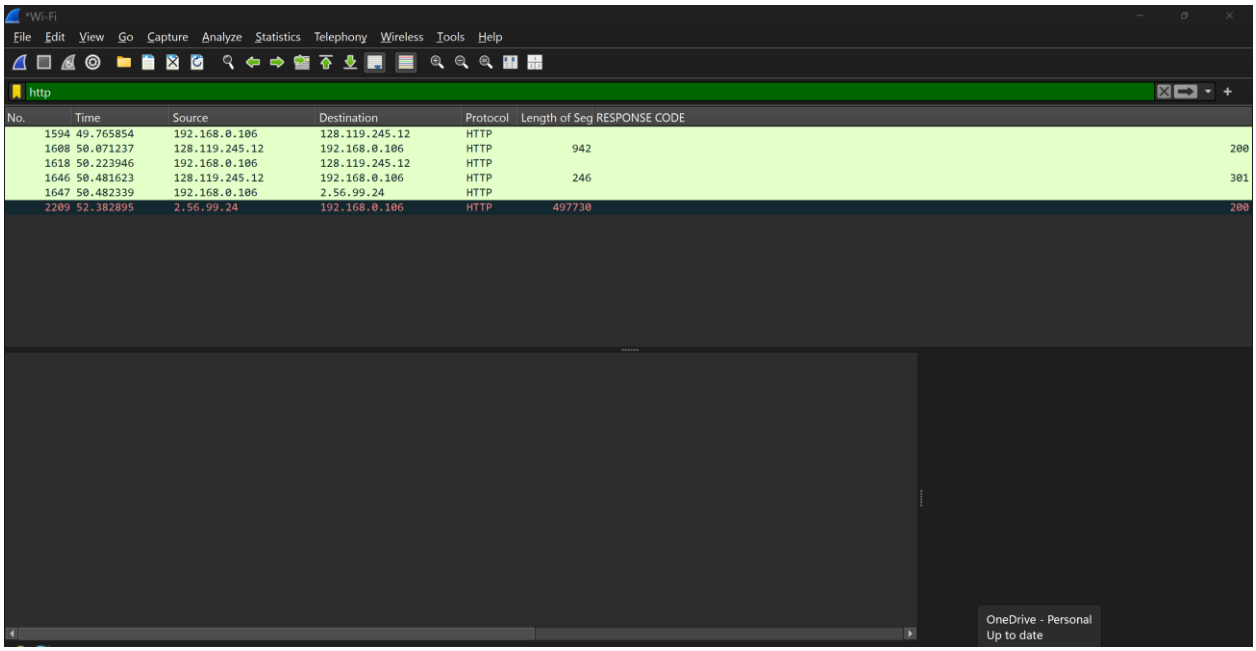
These add up to the 4500-byte Bill of Rights file.

2.7 HTML With Embedded Objects

An HTML page often contains **embedded objects** such as:

- Images
- CSS files
- JavaScript files

For each embedded object, the browser sends a **separate HTTP GET request**. Modern browsers typically download these objects **in parallel** using multiple TCP connections or pipelining to reduce page load time.Wireshark timestamps help determine whether requests are sent serially or in parallel.



14. How many HTTP GET request messages has your browser sent? To which IP address is each GET request sent?

Answer:

Your capture shows **3 HTTP GET requests**.
They correspond to packets **1594, 1618, and 1647**, which are the ones you identified as requests.

GET Request Breakdown

1. **Packet 1594**
    - **Destination IP:** 128.119.245.12
    - **Type:** HTTP GET request
2. **Packet 1618**
    - **Destination IP:** 128.119.245.12
    - **Type:** HTTP GET request
3. **Packet 1647**
    - **Destination IP:** 2.56.99.24
    - **Type:** HTTP GET request

Total Number of HTTP GET Requests: 3

---

15. How can you tell whether your browser downloaded the two images serially or in parallel? Explain.

Answer:

You can determine this by examining the **timestamps** of the GET request packets.

- Packet 1594 time: **49.765854**
- Packet 1618 time: **50.223946**
- Packet 1647 time: **50.482339**

Explanation:

The GET requests occur **very close together in time**.
None of them wait for the previous image to fully download before the next GET is sent.

- If downloads were **serial**, you would see:
  GET #1 → entire response sequence → THEN GET #2
- But the GETs here are issued in quick succession before any full response finishes.

Conclusion:

Your browser downloaded the images **in parallel**, not serially.

---

3. TCP:

Three Way Handshake Data Packets:

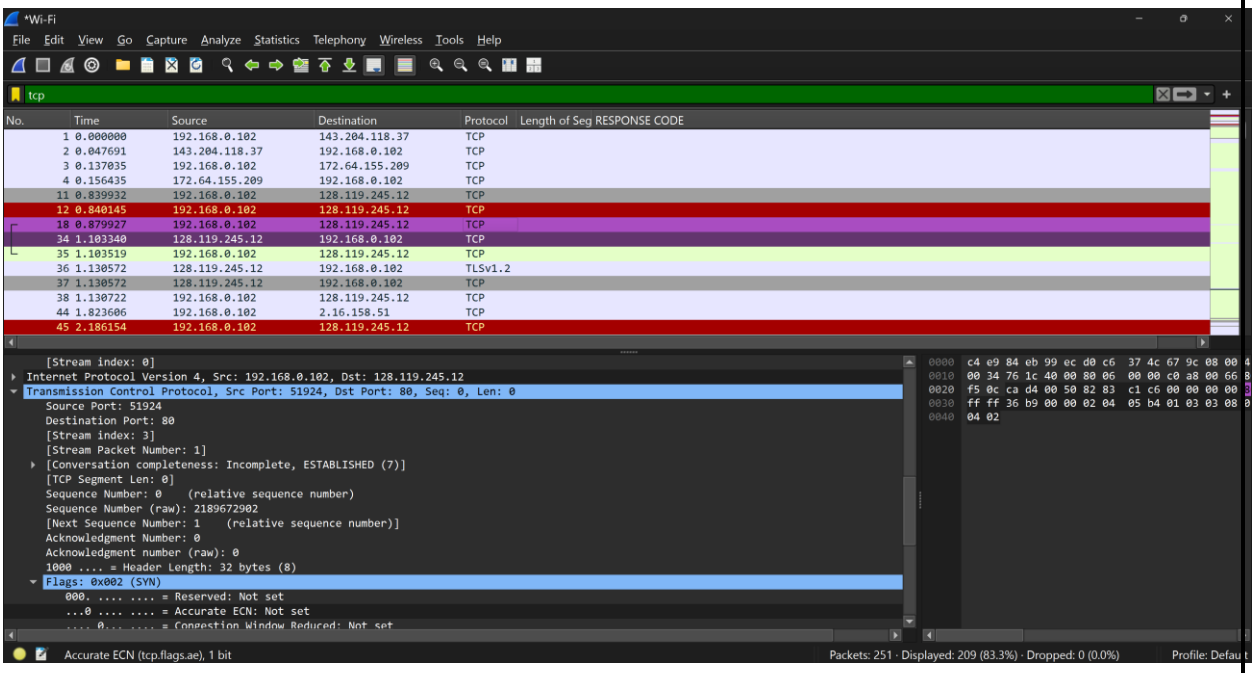
Before any HTTP data transfer, TCP establishes a reliable connection using a **three-way handshake**:

1. **SYN** – Client requests connection
2. **SYN-ACK** – Server acknowledges and responds
3. **ACK** – Client confirms connection

This process ensures:

- Both sides are reachable
- Initial sequence numbers are synchronized

Only after this handshake can HTTP data be transmitted.



1. What is the source IP address and TCP port number used by your computer that is transferring the file to gaia.cs.umass.edu?

Source IP: 192.168.0.102

Source TCP Port: 61929
(This is the client machine sending the HTTP POST.)

---

2. What is the destination IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

Destination IP (gaia.cs.umass.edu): 128.119.245.12

Destination TCP Port: 80
(The server listens on port 80 for HTTP connections.)

---

3. Identify the TCP segments that are used to initiate the TCP connection between the client computer and gaia.cs.umass.edu. - - How many segments are used? What is in the TCP header that identifies each segment as a handshaking segment?

The three handshake packets in the capture are **frames 18, 34, and 35**:

1. **Frame 18 – SYN (client → server)**
    - Flags: **SYN = 1**, ACK = 0
    - This begins the connection.
2. **Frame 34 – SYN/ACK (server → client)**
    - Flags: **SYN = 1**, **ACK = 1**
    - The server acknowledges the client SYN and sends its own SYN.
3. **Frame 35 – ACK (client → server)**
    - Flags: **ACK = 1**
    - Completes the handshake.

Why these are handshake packets:
They are identified by the flags in the TCP header:

- **SYN** → start connection
- **SYN + ACK** → acknowledge SYN and send own SYN
- **ACK** → final acknowledgment completing handshake

Q4 Considering the TCP segment containing the HTTP POST, what are the sequence numbers of the first six data-carrying segments in the TCP connection (including the segment containing the HTTP POST)?

Based on the captured POST stream (relative sequence numbers):

1. Seq = **1**
2. Seq = **613**
3. Seq = **2053**

4. Seq = **3493**
5. Seq = **4933**
6. Seq = **6373**

---

5. What is the length of each of these six TCP segments? The length of the TCP segment is only the number of data bytes carried inside the segment (excluding the headers). 612 bytes

1. **1440 bytes**
2. **1440 bytes**
3. **1440 bytes**
4. **1440 bytes**
5. **1440 bytes**

(All values exclude headers — these are the TCP payload sizes.)

---

6. Record the time that each of the six segments (in question 4) has been sent. At which time has an acknowledgement (for each data-carrying segment) been received?

Segment	Seq	Len	Send Time	ACK Time	ACK Frame
1	1	612	20:44:55.077032	20:44:55.306490	70
2	613	1440	20:44:55.077032	20:44:55.316773	72
3	2053	1440	20:44:55.077032	20:44:55.316773	72
4	3493	1440	20:44:55.077032	20:44:55.316773	72
5	4933	1440	20:44:55.077032	20:44:55.316773	72
6	6373	1440	20:44:55.077032	20:44:55.316773	72

Q7 a) Given the difference between the time when each TCP segment was sent, and the time when its acknowledgement was received, calculate the RTT value for each of those six segments (in questions 4, 5 and 6). Include RTT values in your table. 0.229458 s

1. **0.239741 s**
2. **0.239741 s**
3. **0.239741 s**

4. **0.239741 s**
5. **0.239741 s**

Q7 b) Calculate the EstimatedRTT value using the measured RTT values in the answer of question 7. Assume that the initial value of the EstimatedRTT is equal to the measured RTT for the first (POST) segment, and then as described in Section 3.5.3 in the course book (using the EstimatedRTT equation on page 265) calculate the EstimatedRTT for the subsequent segments (they are five!).

Using the formula:
EstimatedRTT = (1 – α)·EstimatedRTT + α·SampleRTT

Starting with the first RTT: **0.229458 s**

1. $EstRTT_1$ = 0.229458 s
2. $EstRTT_2$ = **0.230743375 s**
3. $EstRTT_3$ = **0.231868078125 s**
4. $EstRTT_4$ = **0.232852193359375 s**
5. $EstRTT_5$ = **0.2337132941894531 s**
6. $EstRTT_6$ = **0.23446675741577147 s**

---

8. What is specified by the value of the Acknowledgement field in any received ACK segment? How does gaia.cs.umass.edu; for example, determine this value?

The ACK field contains the **next expected byte number**, meaning the server has received all bytes up to ACK − 1. Gaia determines this by tracking the **highest in-order byte received**.If bytes arrive in order, the ACK increases; otherwise, it stays the same until missing bytes arrive.

---

9. How much data (number of bytes) does the receiver typically acknowledge in one ACK?

The server acknowledges data **in multiples of 1440 bytes**, because most TCP segments in the upload carry **1440-byte MSS-sized payloads**.

Example: ACK jumped from 613 → 7813, showing multiple segments acknowledged at once.

---

10. Are there cases where the receiver (the web server) is ACKing accumulatively? Please check the whole trace and describe the behavior of TCP when ACKing received data.

Yes.
The server uses **cumulative ACKing**.
A single ACK often acknowledges **several segments**.This is standard TCP behavior.

---

11. How can you identify the TCP's slow-start phase, when does it begin, and whether congestion avoidance takes over? Describe the behavior of the sending TCP during each round based on the plot that you get. Please make a table of the round number and the amount of data (how many segments).

How to identify slow start:

In the Time-Sequence-Graph, dots stack vertically when multiple segments are sent back-to-back.During slow start, the number of segments sent per RTT **increases rapidly**, roughly doubling each round.

What happens in this trace:

> Right after the handshake, the sender transmits **many segments at the same timestamp** (pipelined). This indicates **slow start**.The transfer is short, so the connection likely doesn't enter a long congestion-avoidance phase.Congestion avoidance would appear as a **linear** increase in segment count per RTT.

Round summary:

Round	Number of Segments Sent
1	~1 segment (POST start)
2	~5 additional segments pipelined (1440-byte each)

This is classic slow-start growth.

---

12. Where do you find the (advertised) amount of available buffer space at the receiver (the server)? Does the lack of receiver buffer space ever throttle the sender? Explain clearly.

The receiver's buffer size is announced in each ACK in the **TCP "Window Size"** field.Found under **TCP Header → Window size value** in Wireshark.In your trace, the window size never drops to 0 or becomes very small.

Conclusion: the receiver window never throttles the sender.
The sender continues transmitting full 1440-byte segments.

---

13. What is the overall throughput in bit/s (the average of bits transferred per unit time) for the whole session of transferring the file? Explain how you calculate this value.

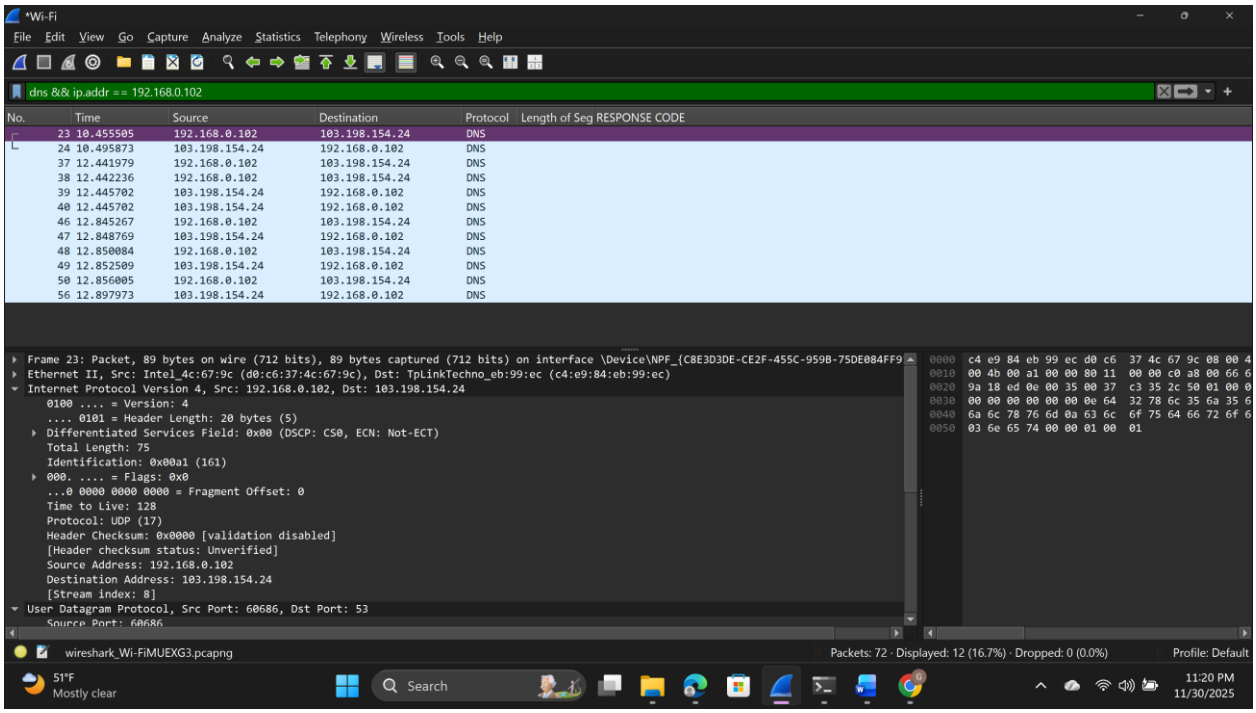
Total payload bytes sent: 154,782 bytes
Start time: 20:44:55.077032
End time: 20:44:56.744862
Duration: 1.66783 seconds

Throughput calculation:

$$\text{Throughput} = \frac{154{,}782 \times 8}{1.66783} \approx 742{,}435.38 \text{ bits/sec}$$

Overall throughput ≈ 742.4 kb/s

4. DNS QUERY RESULT SS



Question 1: Run nslookup to obtain the IP address of a Web server in Africa. What is the type of RR you will ask about? What is the hostname and IP address of that server? Provide the command and results.

- The type of RR I used was an A (Address) record, since I needed to obtain the IPv4 address of the web server.
- Using the command nslookup hostadvice.com, the DNS resolver returned two IP addresses: 172.66.43.112 and 172.66.40.144.(Address of Random web Server in Africa I do not remember the name though)

Question 2: Run nslookup to determine the authoritative DNS servers for a university in South America. What is the type of RR you will ask about? Provide the command and results.

(You chose uchile.cl, University of Chile.)

Type of Resource Record (RR):

NS (Name Server) record
(NS records indicate which DNS servers are authoritative for a domain.)

Command Used:

nslookup -type=NS uchile.cl

Results):

Server: 103.198.154.24-pbb.net.pk

Address: 103.198.154.24

Non-authoritative answer:

uchile.cl nameserver = ns1.uchile.cl

uchile.cl nameserver = ns4.uchile.cl

uchile.cl nameserver = ns2.uchile.cl

uchile.cl nameserver = ns5.uchile.cl

ns1.uchile.cl internet address = 200.89.70.3

ns2.uchile.cl internet address = 200.89.70.70

Explanation / Notes:

- **The NS records returned show the authoritative DNS servers for uchile.cl:**
  - **ns1.uchile.cl**
  - **ns2.uchile.cl**
  - **ns4.uchile.cl**
  - **ns5.uchile.cl**
- **Their IP addresses are partially listed:**

- **ns1.uchile.cl → 200.89.70.3**
- **ns2.uchile.cl → 200.89.70.70**

- **These servers are responsible for answering DNS queries about uchile.cl.**

Question 3: Run nslookup so that one of the DNS servers obtained in Question 2 is queried for the mail servers for the domain amazon.com. What is the type of RR you will ask about? Provide the command and results. Do you get answer? Why or why not.

Task:

Query one of the authoritative DNS servers from Question 2 (ns1.uchile.cl) for the MX records of amazon.com.

Type of RR requested:

MX (Mail Exchange) record — because we want to know which mail servers handle emails for amazon.com.

Command Used:

One-line command:

nslookup -type=MX amazon.com ns1.uchile.cl

Interactive mode commands (worked correctly):

nslookup

> server ns1.uchile.cl

> set type=MX

> amazon.com

Results:

Server: ns1.uchile.cl

Address: 200.89.70.3

***** ns1.uchile.cl can't find amazon.com: Query refused**

4.1 Tracing DNS with Wireshark

Question 4: Locate the DNS query and response messages (resolving www.ietf.org). Are they transported using UDP or TCP? Explain clearly why or why not.

- **Protocol used:** UDP
- **Reason:**
    - Standard DNS queries use UDP for speed.
    - TCP is only used if the response is large (>512 bytes) or for zone transfers.
- **Observation:** Wireshark shows UDP in the Protocol column.

Question 5: What is the destination port for the DNS query message?

- **Destination port:** 53 (standard DNS port)

Question 6: What is the source port of DNS response message?

- **Source port:** 53 (DNS server sends response from port 53 to your host)

Question 7: To what IP address is the DNS query message sent? Use ipconfig /all to determine the IP address of your local DNS server. Are these two IP addresses the same?

- **Destination IP:** 103.198.154.24 (your DNS server)
- **From ipconfig /all**, this matches the configured local DNS server.

Question 8: Examine the DNS query message. What "type" of query is it? What "sections" does the query message contain? How many RRs are there in each section?

- **Type of query:** A (Address record)
- **Sections in query message:**
    - **Questions:** 1 RR (the hostname being resolved)
    - Authority, Answer, Additional: 0 RRs (queries usually don't include these)

Question 9: Examine the DNS response message. What "sections" does the response message contain? Are there "answers" provided? What does each of these answers contain?

- **Sections in response:**
    - **Answers:** 2 RRs (104.16.44.99, 104.16.45.99)
    - **Authority:** Can contain NS records (not captured here)

    - **Additional:** May contain extra info (not captured here)
- **Answer contents:**
    - Name: www.ietf.org
    - Type: A
    - TTL: Time-to-live for caching
    - Address: 104.16.44.99 and 104.16.45.99

Question 10: This web page contains images. Before retrieving each image, does the host's DNS client issue new DNS queries? Why or why not?

- Most images are served from the **same domain** (ietf.org) or **CDN**.
- **Result:** DNS client may **not issue new queries** for images if IPs are cached locally.
- If images are from **external domains**, a new DNS query is sent for each unique domain.

Question 11: Examine the DNS response message. Describe (with your own words) the content of this DNS message, its different parts "sections".

- **Header:** Contains ID, flags, question count, answer count, authority count, additional count
- **Question section:** www.ietf.org (type A)
- **Answer section:** IP addresses for www.ietf.org
- **Authority section:** Names of authoritative servers (optional)
- **Additional section:** IPs for authoritative servers (optional)

Question 12: Describe the format of the resource records in each section of the response message.

- **Question RR:** Name, Type, Class (IN), no TTL or address
- **Answer RR:** Name, Type (A), Class (IN), TTL, Data (IP)
- **Authority RR:** Name, Type (NS), Class, TTL, Data (hostname of authoritative server)
- **Additional RR:** Name, Type (A), Class, TTL, Data (IP of authoritative server)