Summer 99

Wireshark CA#2

COMPUTER NETWORKS

SAHAND KHOSHDEL – ST ID: 810196607

TABLE OF CONTENTS

INTRODUCTION

- In this computer assignment we will have a closer look at the DNS and HTTP protocols. Two important and common protocols running on Application Layer.
    - The First part consists of capturing DNS messages, Recognizing and Understanding what Query and Response messages are and how a Recursive Query is been done and what are the components involved in this IP address lookup procedure. We will also become familiar with DNS caches on the client side.
    - In the Second part we will have a brief look at GET messages and their responses, in the framework of the HTTP protocol. Both Requests and Responses will be divided into parts and will be discussed in this Report.

PART I: DNS (Domain Name System)

- **What does the command “ipconfig/flushdns” actually do?**
- **Why is flushing the DNS cache, necessary?**

During the course, We were informed that the main purpose of local DNS servers was saving (hostname, IP address) pairs for decreasing look up time and computation load for higher servers (TLD’s & Root DNS servers), Each computer can do something similar using its “DNS cache”, in order to reduce the whole translation process. There’s no need to fetch the IP address from your local DNS every single time, especially when the address is called repeatedly.

By saving the translation pair in its DNS cache, the host computer **only needs to look up its own translation table, in the DNS cache**. So, your computer stores the combinations of domain names and their IP in a local cache to avoid fetching from the Domain name Server (DNS) every time you use same domain name URL.

Each of the points (DNS servers) in the total hierarchical structure has **a DNS cache (record route)** for the same reason, **which is used to speed up the name resolution process.**

- It also holds another crucial information called **"Timeout"** which indicates the valid time for the IP and domain name pairs, when this time arrives, the pair is expired and your computer re-fetches the pair from the DNS and stores it in it’s local cache again.
- The whole hierarchical structure of look-up process, based on caches helps average webpage loading rates by a great factor!

```
dns.google.com
----------------------------------------
Record Name . . . . . : dns.google.com
Record Type . . . . . : 1
Time To Live  . . . . : 417
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 8.8.4.4


Record Name . . . . . : dns.google.com
Record Type . . . . . : 1
Time To Live  . . . . : 417
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 8.8.8.8
```

Figure 1-1: DNS record examples (ipconfig/displaydns)

- DNS flushing is the mechanism which the user can manually make all the entries in the cache invalid, so the host's computer re-fetches new pairs from now on, whenever it needs and stores it in the local cache.
- For example, if a website has recently moved servers, the old website might be displayed for a while (until all DNS caches in the hierarchy update the new IP address). Flushing the DNS cache might help in these situations.

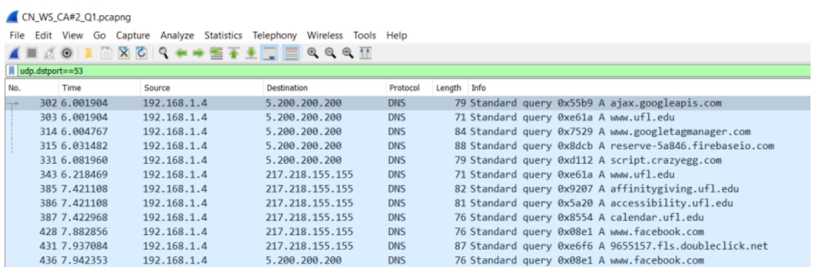
```
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ASUS>ipconfig/flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.

C:\Users\ASUS>
```

Figure 1-2: Flushing DNS cache

Part I Questions:

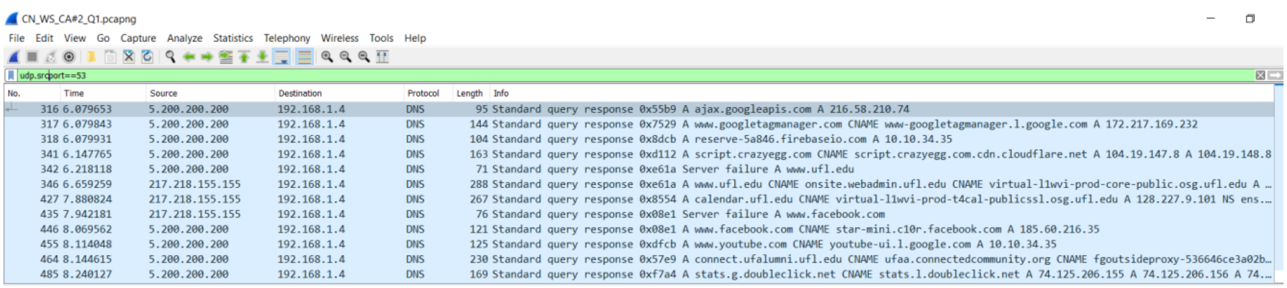
Q1)

- The DNS query packets are shown in the figure below:
  These packets contain requests to the DNS servers for translating a host name to an IP address:



No.	Time	Source	Destination	Protocol	Length	Info
302	6.001904	192.168.1.4	5.200.200.200	DNS	79	Standard query 0x55b9 A ajax.googleapis.com
303	6.001904	192.168.1.4	5.200.200.200	DNS	71	Standard query 0xe61a A www.ufl.edu
314	6.004767	192.168.1.4	5.200.200.200	DNS	84	Standard query 0x7529 A www.googletagmanager.com
315	6.031482	192.168.1.4	5.200.200.200	DNS	88	Standard query 0x8dcb A reserve-5a846.firebaseio.com
331	6.081960	192.168.1.4	5.200.200.200	DNS	79	Standard query 0xd112 A script.crazyegg.com
343	6.218469	192.168.1.4	217.218.155.155	DNS	71	Standard query 0xe61a A www.ufl.edu
385	7.421108	192.168.1.4	217.218.155.155	DNS	82	Standard query 0x9207 A affinitygiving.ufl.edu
386	7.421108	192.168.1.4	217.218.155.155	DNS	81	Standard query 0x5a20 A accessibility.ufl.edu
387	7.422968	192.168.1.4	217.218.155.155	DNS	76	Standard query 0x8554 A calendar.ufl.edu
428	7.882856	192.168.1.4	217.218.155.155	DNS	76	Standard query 0x08e1 A www.facebook.com
431	7.937084	192.168.1.4	217.218.155.155	DNS	87	Standard query 0xe6f6 A 9655157.fls.doubleclick.net
436	7.942353	192.168.1.4	5.200.200.200	DNS	76	Standard query 0x08e1 A www.facebook.com

Figure 1-3: DNS query packets (udp.dstport==53)



No.	Time	Source	Destination	Protocol	Length	Info
316	6.079653	5.200.200.200	192.168.1.4	DNS	95	Standard query response 0x55b9 A ajax.googleapis.com A 216.58.210.74
317	6.079843	5.200.200.200	192.168.1.4	DNS	144	Standard query response 0x7529 A www.googletagmanager.com CNAME www-googletagmanager.l.google.com A 172.217.169.232
318	6.079931	5.200.200.200	192.168.1.4	DNS	104	Standard query response 0x8dcb A reserve-5a846.firebaseio.com A 10.10.34.35
341	6.147765	5.200.200.200	192.168.1.4	DNS	163	Standard query response 0xd112 A script.crazyegg.com CNAME script.crazyegg.com.cdn.cloudflare.net A 104.19.147.8 A 104.19.148.8
342	6.218118	5.200.200.200	192.168.1.4	DNS	71	Standard query response 0xe61a Server failure A www.ufl.edu
346	6.659259	217.218.155.155	192.168.1.4	DNS	288	Standard query response 0xe61a A www.ufl.edu CNAME onsite.webadmin.ufl.edu CNAME virtual-l1wvi-prod-core-public.osg.ufl.edu A …
427	7.880824	217.218.155.155	192.168.1.4	DNS	267	Standard query response 0x8554 A calendar.ufl.edu CNAME virtual-l1wvi-prod-t4cal-publicssl.osg.ufl.edu A 128.227.9.101 NS ens.…
435	7.942181	217.218.155.155	192.168.1.4	DNS	76	Standard query response 0x08e1 Server failure A www.facebook.com
446	8.069562	5.200.200.200	192.168.1.4	DNS	121	Standard query response 0x08e1 A www.facebook.com CNAME star-mini.c10r.facebook.com A 185.60.216.35
455	8.114048	5.200.200.200	192.168.1.4	DNS	125	Standard query response 0xdfcb A www.youtube.com CNAME youtube-ui.l.google.com A 10.10.34.35
464	8.144615	5.200.200.200	192.168.1.4	DNS	230	Standard query response 0x57e9 A connect.ufalumni.ufl.edu CNAME ufaa.connectedcommunity.org CNAME fgoutsideproxy-536646ce3a02b…
485	8.240127	5.200.200.200	192.168.1.4	DNS	169	Standard query response 0xf7a4 A stats.g.doubleclick.net CNAME stats.l.doubleclick.net A 74.125.206.155 A 74.125.206.156 A 74.…

Figure 1-4: DNS query packets (udp.srcport==53)

- **What does "A" stand for?**
- The prefix "A" behind all the queries indicates a (Hostname – IPV4) translation record.

- **CDN's:**

  - The first query packet is sent to the address (URL): https://ajax.googleapis.com. This is the address of a "CDN server" associated to google. CDN stands for "content delivery network". CDN's are like the agent servers hired by the main server in different countries, providing their information for their nearby hosts, resulting on less network traffic (due to the main server bottle-neck), higher speed and more security.

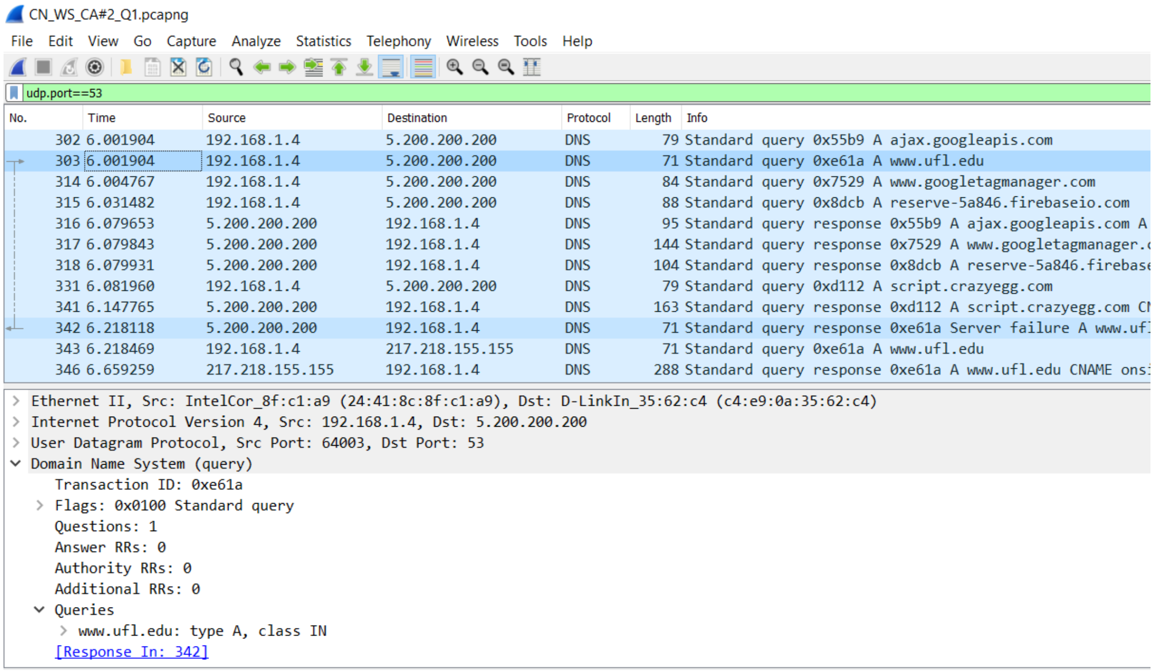


Figure 1-5: DNS query packets (udp.srcport==53)

- **Transaction ID:** The Transaction ID is a random number generated by the nameserver initiating the query. When the answering nameserver responds with an answer, it will set the same transaction ID. (Figure 1-5 shows the request and response query for ufl.edu)

- **DNS cache poisoning:** DNS cache poisoning occur if the transaction is non-random or predictable. **It happens when an attacker tricks a resolver into accepting faked data for a given query** and then having the **"poisoned" resolver** hand that faked data out to its clients.

Q2,3)

The first access attempt was unsuccessful and the host accessed ufl.edu with the second attempt. Both attempts are analyzed in this question:

Unsuccessful Query & Response:

The details of the query message to the destination website is shown below:

```
Wireshark · Packet 303 · CN_WS_CA#2_Q1.pcapng

> Frame 303: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface \Device\NPF_{426425D2-BAF7-4EF8-BD64-F0869CC389DD}, id 0
> Ethernet II, Src: IntelCor_8f:c1:a9 (24:41:8c:8f:c1:a9), Dst: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4)
> Internet Protocol Version 4, Src: 192.168.1.4, Dst: 5.200.200.200
> User Datagram Protocol, Src Port: 64003, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0xe61a
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.ufl.edu: type A, class IN
    [Response In: 342]
```

Figure 1-6 Failed Query message detail

- The **Flag's first bit indicates that it's a query message**. Other sections were described earlier and **other flags will be described in the response part.**
- **Type:** A (IPV4 translation)
- **Class:** IN (Internet Network)

```
Wireshark · Packet 342 · CN_WS_CA#2_Q1.pcapng

> Frame 342: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface \Device\NPF_{426425D2-BAF7-4EF8-BD64-F0869CC389DD}, id 0
> Ethernet II, Src: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4), Dst: IntelCor_8f:c1:a9 (24:41:8c:8f:c1:a9)
> Internet Protocol Version 4, Src: 5.200.200.200, Dst: 192.168.1.4
> User Datagram Protocol, Src Port: 53, Dst Port: 64003
v Domain Name System (response)
    Transaction ID: 0xe61a
  v Flags: 0x8182 Standard query response, Server failure
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0010 = Reply code: Server failure (2)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.ufl.edu: type A, class IN
    [Request In: 303]
    [Time: 0.216214000 seconds]
```

Figure 1-7 Response message detail

```
v Flags: 0x8182 Standard query response, Server failure
    1... .... .... .... = Response: Message is a response
    .000 0... .... .... = Opcode: Standard query (0)
    .... .0.. .... .... = Authoritative: Server is not an authority for domain
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... 1... .... = Recursion available: Server can do recursive queries
    .... .... .0.. .... = Z: reserved (0)
    .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... .... ...0 .... = Non-authenticated data: Unacceptable
    .... .... .... 0010 = Reply code: Server failure (2)
```

Figure 1-8 Describing response Flags

- **First bit** indicates whether the message is **a query or a response**.
- The **Opcode** identifies the **Request type:** In Our case it's a Query. (standard request)
    - Opcode Types:
        - **QUERY:** Standard request
        - **IQUERY:** Inverse request (obsoleted by RFC3425)
        - **STATUS:** Server status query
        - **NOTIFY:** Database update notification (RFC1996)
        - **UPDATE:** Dynamic database update (RFC2136)

- **Authoritative DNS**: The DNS server is an Authoritative DNS for the domain and involves a copy of its domain's information. This information can be passed to the DNS server by an administrator or the upper DNS server.

- **Not Truncated**: The message isn't shortened. Truncations happens when the message is longer than the standard limit issued for the Transport Layer protocol. TCP messages are length-unlimited but UDP messages have a maximum size of 512 bytes and messages longer than this size should be truncated.

- **DNS Access: Recursive Method** (Going up to the TLD, Root DNS servers, going down to the local DNS server holding destinations IP address, and moving backwards to deliver the IP to the clients local DNS). The Client request a Recursion Method using the "Recursion Desired bit" and the Server replies whether it supports the method by the "Recursion Available bit" or not.

- **Z bit:** reserved for future use.

- **Answer Authentication**: Indicates whether the answer/authority is authenticated by the DNS server or not.

- **Data Authentication:** Answer is **not authenticated** by the server and the reply code indicates **Server Failure.**

- **Reply Code:** The four most common reply codes, returned with virtually all DNS queries, are NOERROR, NXDOMAIN, SERVFAIL, and REFUSED.

But why are reply codes so important?

  - **Reply codes** play a main role in **troubleshooting DNS problems**.

  - By **incorporating response data** into our workflow, we can **ask better questions** to **conduct a more complete forensic analysis**. Reply codes help us answer the following crucial questions:

    - Does the answer lead me to the right spot, or is it leading me to a dangerous area?

    - Suppose instead of directing the query to a good IP address, we discover that it's being sent to an IP address we don't recognize. Ok, then let's investigate what DNS server answered.

    - Often, DNS hijacks, compromise domain registrar accounts and modify the domain name records, resulting a malicious connection for the host as soon as the URL is entered. This modification can be shown with reply codes

Successful Query & Response:

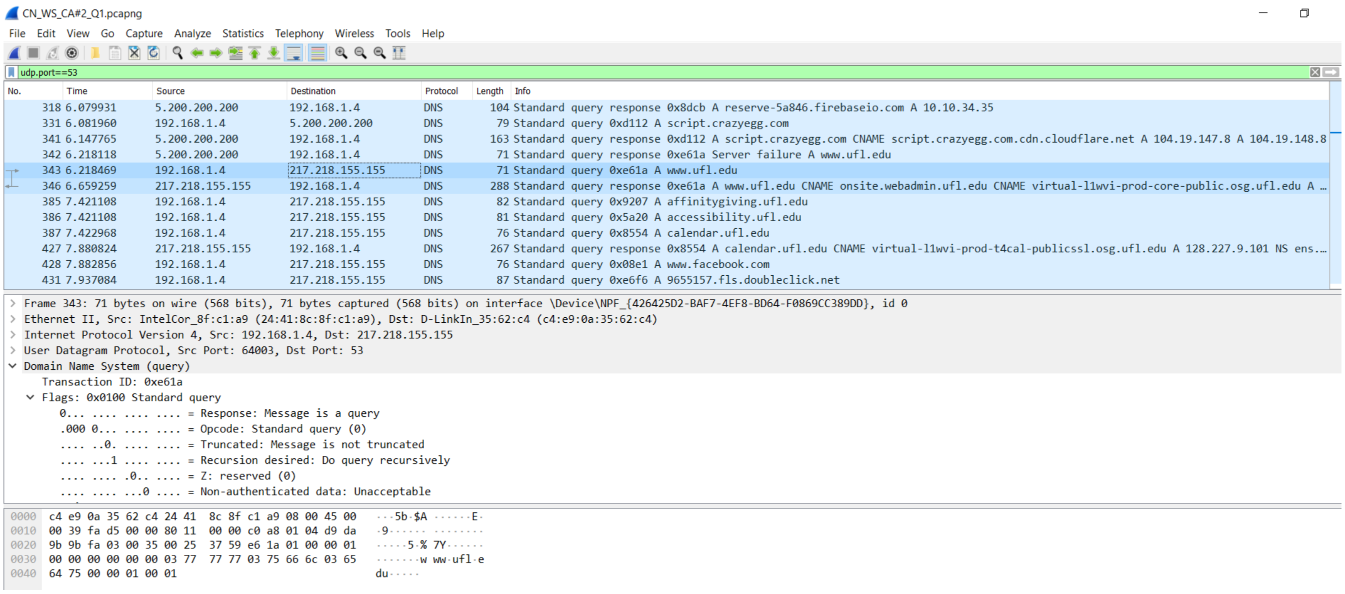


Figure 1-9: Successful Query and Response

- Query message details are exactly the same as the previous part. But the response message has a successful authentication flag and reply code.

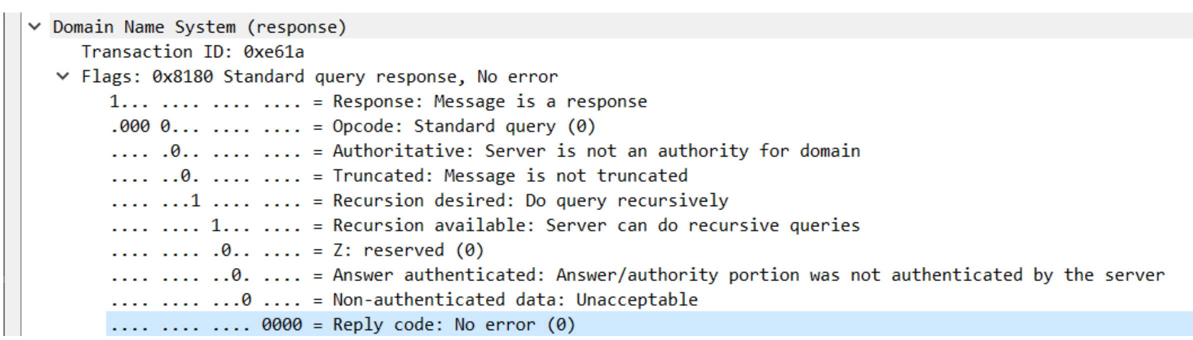


Figure 1-10: Successful Response message detail

The "Answer Part" the response message would be further discussed in Question 6

Q4)

TTL stands for **“Time to Live”**. It’s a metric for **the time a DNS cache holds a record** before discarding it in order to prevent slow cache access and high load. After the **expiration** of a record’s TTL, it should be **discarded or refreshed**.

TTL can be found in the **Answers** part of a **Response message** as shown below:

(4’th row of each answer)

```
v Answers
  v www.ufl.edu: type CNAME, class IN, cname onsite.webadmin.ufl.edu
      Name: www.ufl.edu
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
      Time to live: 300 (5 minutes)
      Data length: 18
      CNAME: onsite.webadmin.ufl.edu
  v onsite.webadmin.ufl.edu: type CNAME, class IN, cname virtual-l1wvi-prod-core-public.osg.ufl.edu
      Name: onsite.webadmin.ufl.edu
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
      Time to live: 900 (15 minutes)
      Data length: 37
      CNAME: virtual-l1wvi-prod-core-public.osg.ufl.edu
  v virtual-l1wvi-prod-core-public.osg.ufl.edu: type A, class IN, addr 128.227.9.98
      Name: virtual-l1wvi-prod-core-public.osg.ufl.edu
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 900 (15 minutes)
      Data length: 4
      Address: 128.227.9.98
> Authoritative nameservers
> Additional records
  [Request In: 343]
  [Time: 0.440790000 seconds]
```

Figure 1-11: Time to live – Answers

- ## Failover Enable:

For example, let’s say the **primary IP address** for the domain we are looking for is **unavailable**. This domain also has **Failover enabled**, which would point users to a **backup IP address** when the primary is down.

This could be handled two ways. If the **record has a high TTL**, users will still be **pointed to the primary IP address** until the resolver’s **cache expires**. If the **record has a low TTL**, they have a higher chance of **being pointed** to the **correct** endpoint **sooner.**

- ## Time and Money Cost:

  - The shorter the TTL, the more often the authoritative name servers will have to answer queries, and your monthly bill goes up.
  - Longer TTL’s also cut resolution times. Every time a query has to ask an authoritative name server, it adds an additional lookup, which could add precious milliseconds.

When to keep TTL short?	When to use a long TTL?
• When the endpoint IP is dynamically changing (Failover, Load Balancing, …) • Critical records (30 sec – 5min) (Min TTL affecting resolving name server: 30 sec) • Advance settings like Geo-DNS	• Top records that should have longer TTL’s: - MX record (points to your mail server) - DKIM and SPF (usually configured with MX records) - TXT record • A and CNAME records point to a web server, typically require a longer TTL, since they are rarely changed. (12 h – 1 Day)

Figure 1-12: How long should a TTL be?

Q5)

Recursive Query: These websites are actually the DNS servers placed in the path of our queries.3 types of queries are used in DNS messages. Recursive, Iterative & Non-Recursive queries. Recursive Queries are used in this CA. A recursive query is initiated by the DNS resolver checking the DNS local cache for finding the corresponding IP address to the hostname the client has requested. If the pair isn't found in the local cache, The DNS resolver starts a recursive process, contacting the local DNS, TLD's & Root DNS and vice versa in the destination side until it finds the Authoritative Name Server holding the corresponding IP address for the destination and returns it to the client. The procedure ends up by storing the recent accessed pair in the clients local DNS cache.

What are the Queries with another destination website?

If we try to access the websites, Error 400 will appear, indicating a malformed or illegal request that probably shows they don't have HTML data to present and are DNS servers placed in the query path.

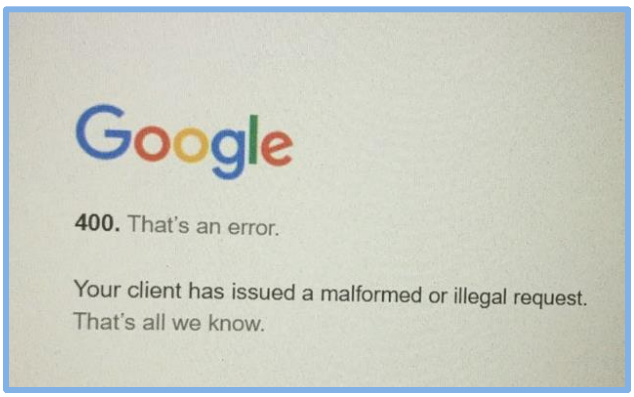


Figure 1-13: 400 error caused by accessing DNS servers

Q6)

The response message contains 3 answers. The first one is the answer for accessing ufl.edu and the other two are answers for accessing UFL's subdomains.

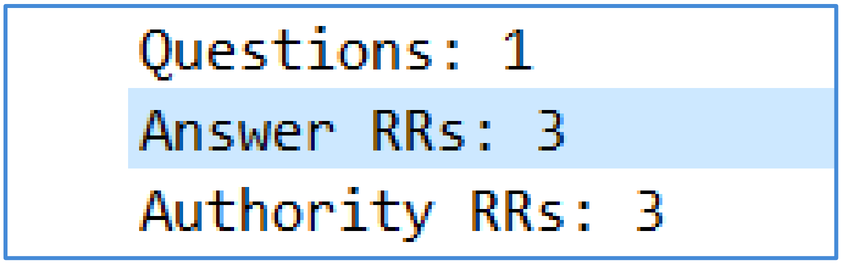


Figure 1-14: Answer RR's

RR's (Resource records):

> A resource record, commonly referred to as an RR, is the unit of information entered in DNS zone files; RRs are the basic building blocks of host-name and IP information and are used to resolve all DNS queries. Resource records come in a fairly wide variety of types in order to provide extended name-resolution services.
>
> Different types of RRs have different formats, as they contain different data. Many RRs share a common format. Each DNS Server contains RRs for the portion of the name space for which it is authoritative

Each answer contains a Name, Type, TTL, Class, Data length & CNAME (only if the answer Type is CNAME)

They were previously shown in Figure 1-11.

- (CNAME refers to the domain name that is queried in order to resolve the original DNS query)

Q7)

NS-lookup basics

- NSlookup is a network **administration command-line** tool available for many computer operating systems.

- It is **used for querying** the Domain Name System (**DNS**) to obtain domain name or IP address mapping information.

- The main use of **nslookup** is for troubleshooting DNS related problems.

- Nslookup can be use in **interactive** and **non-interactive** mode.

    - To use in interactive mode, we can **type nslookup** at the command line and hit return.

    - We will use non-interactive mode in this CA

- NSlookup is used for the following cases:

    - Find the IP address of a host.
    - Find the domain name of an IP address.
    - Find mail servers for a domain.

```
C:\Users\ASUS>nslookup ufl.edu
Server:  UnKnown
Address:  5.200.200.200

Non-authoritative answer:
Name:    ufl.edu
Address:  128.227.9.98
```

Figure 1-15: DNS server and dest. domain name & IP

- A **Non-authoritative** answer, shows that the resolver **didn’t fetch** the answer from an authoritative DNS server, particularly responsible for the domain, but **got it from a cache record stored in some DNS server** along the path.

```
C:\Users\ASUS>nslookup -type=NS ufl.edu
Server:  UnKnown
Address:  5.200.200.200

Non-authoritative answer:
ufl.edu nameserver = ns.name.ufl.edu
ufl.edu nameserver = ens.name.ufl.edu
ufl.edu nameserver = rns.name.ufl.edu

C:\Users\ASUS>
```

Figure 1-16: results for “nslookup -type=NS ufl.edu”

- Command prompt returns the DNS servers which resolver accessed through nslookup for hostname – IP translation.

PART II: HTTP (Hypertext Transfer Protocol)

- By accessing the URL mentioned in the CA, the website containing 1 HTML text message and 2 images (1 PNG (The logo) and 1 JPG (The book cover)) appears:

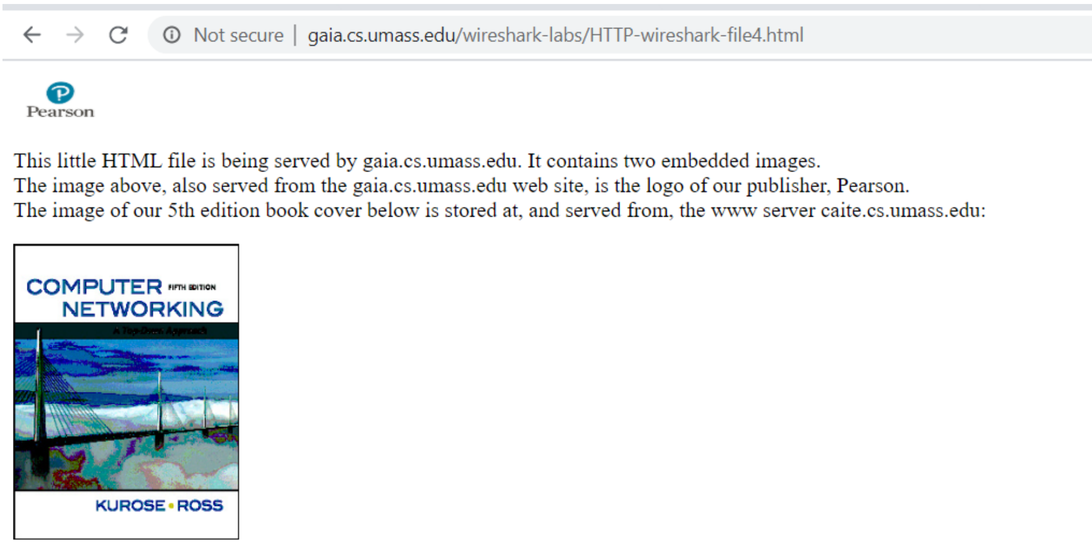


Figure 2-1: Destination website content

HTTP basics:

An HTTP client sends a request message to an HTTP server. The server, in turn, returns a response message. In other words, HTTP is a *pull protocol*, the client *pulls* information from the server (instead of server *pushes* information down to the client).

The **browser** turns the **URL** into a ***request message*** and sends it to the HTTP server. The **HTTP server interprets** the **request message**, and **returns** you an **appropriate response message**, which is either the resource you requested or an error message. This process is illustrated below:

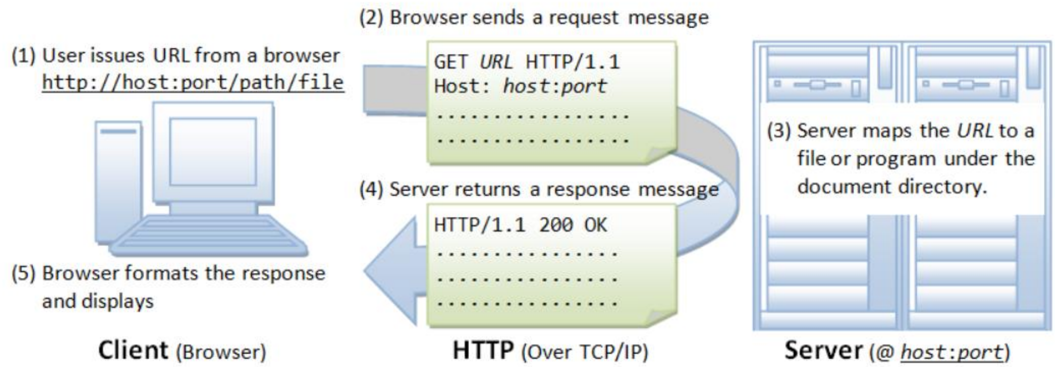
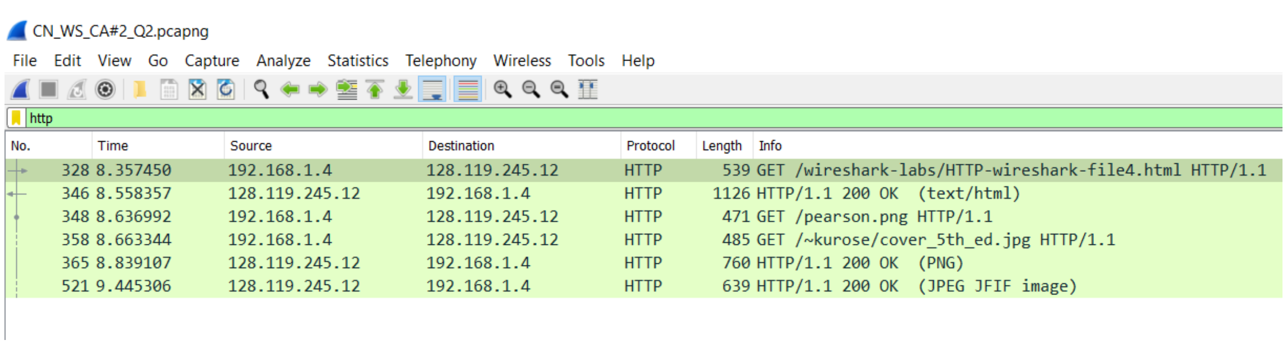


Figure 2-3: From entering URL to accessing destination info

When this request message reaches the server, the server can take either one of these actions:

1. The server interprets the request received, maps the request into a *file* under the server's document directory, and returns the file requested to the client.
2. The server interprets the request received, maps the request into a *program* kept in the server, executes the program, and returns the output of the program to the client.
3. The request cannot be satisfied, the server returns an error message.



No.	Time	Source	Destination	Protocol	Length	Info
328	8.357450	192.168.1.4	128.119.245.12	HTTP	539	GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
346	8.558357	128.119.245.12	192.168.1.4	HTTP	1126	HTTP/1.1 200 OK (text/html)
348	8.636992	192.168.1.4	128.119.245.12	HTTP	471	GET /pearson.png HTTP/1.1
358	8.663344	192.168.1.4	128.119.245.12	HTTP	485	GET /~kurose/cover_5th_ed.jpg HTTP/1.1
365	8.839107	128.119.245.12	192.168.1.4	HTTP	760	HTTP/1.1 200 OK (PNG)
521	9.445306	128.119.245.12	192.168.1.4	HTTP	639	HTTP/1.1 200 OK (JPEG JFIF image)

Figure 2-2: HTTP messages sent and received by the browse

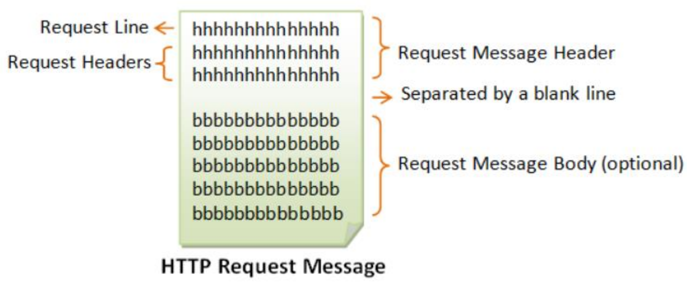
Part II Questions:

Q1)

As shown in Figure 2-2, The browser has sent 3 GET messages corresponding to the 3 (1 text and 2 images) data files it has accessed. It has also received their response from the destination website.

Q2)

- The format of an HTTP request message is as below:



GET message details:

GET is the most common HTTP request method. A client can use the GET request method to request for a piece of resource from an HTTP server.

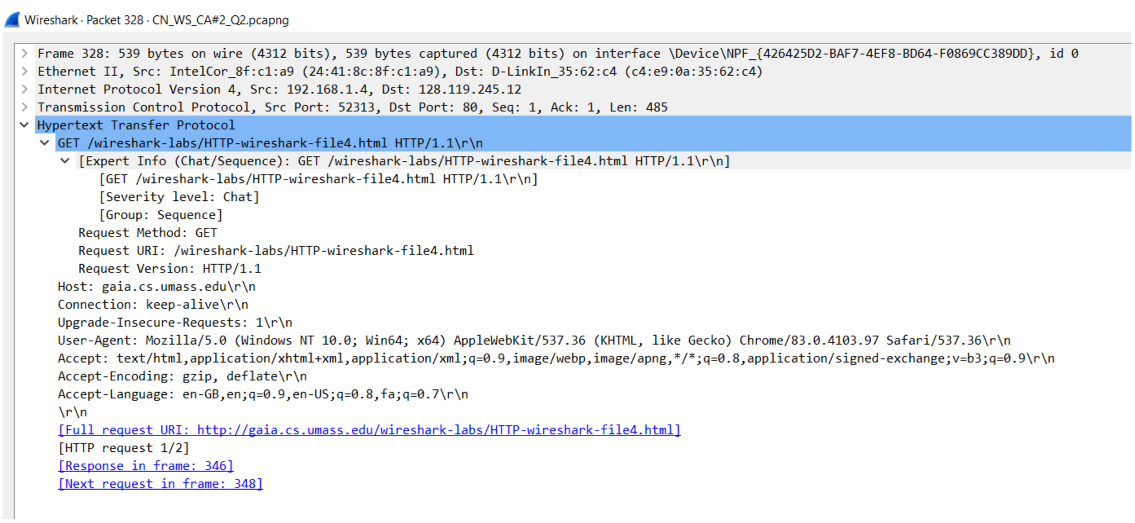


Figure 2-3: GET message for HTML content

- ***Request-URI***: specifies the path of resource requested, which must begin from the root "/" of the document base directory.

- ***HTTP-version***: Either HTTP/1.0 or HTTP/1.1. This client *negotiates* the protocol to be used for the current session. For example, the client may request to use HTTP/1.1. If the server does not support HTTP/1.1, it may inform the client in the response to use HTTP/1.0.

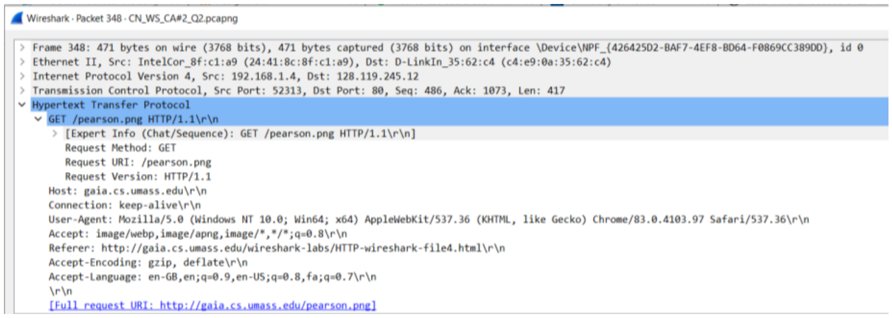


Figure 2-4: GET message for PNG content

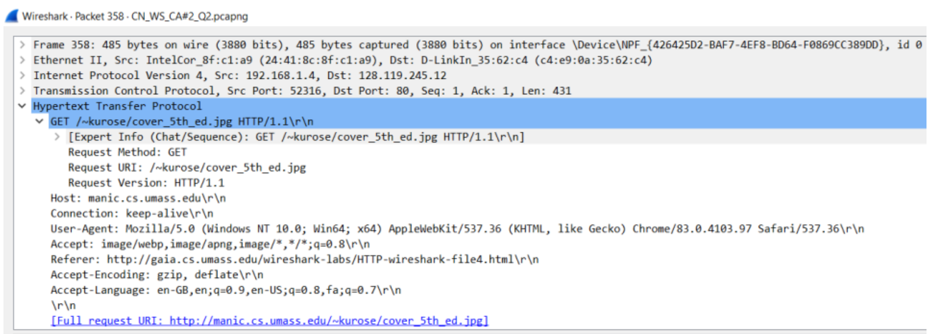


Figure 2-5: GET message for JPG content

The **server** receives the request message, **interprets** and **maps the *request-URI*** to a document under its document directory. If the **requested document is available**, the **server returns the document** with a **response status code "200 OK"**.

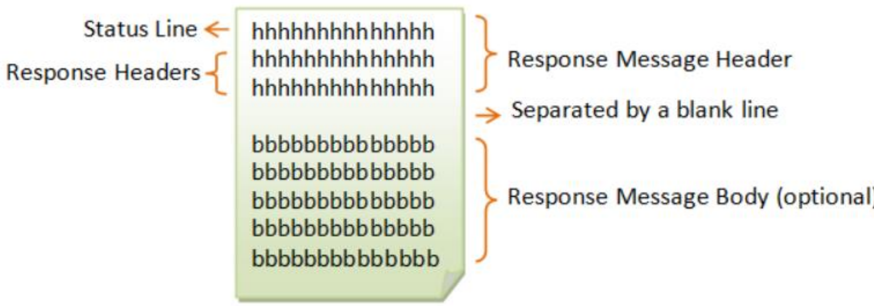
The response headers provide the necessary description of the document returned, such as the last-modified date (Last-Modified), the MIME type (Content-Type), and the length of the document (Content-Length).

The response body contains the requested document. The browser will format and display the document according to its media type (e.g., Plain-text, HTML, JPEG, GIF, and etc.) and other information obtained from the response headers.

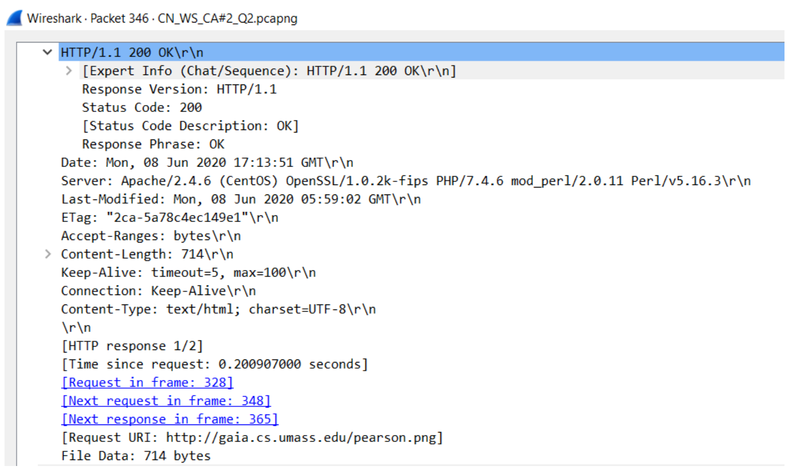
In Our Case, the 3 GET requests issued by the browser reach the server. Fetch the data and comeback with a response message that we'll explain in the next section.

Q3)

- The format of the HTTP response message is as below:

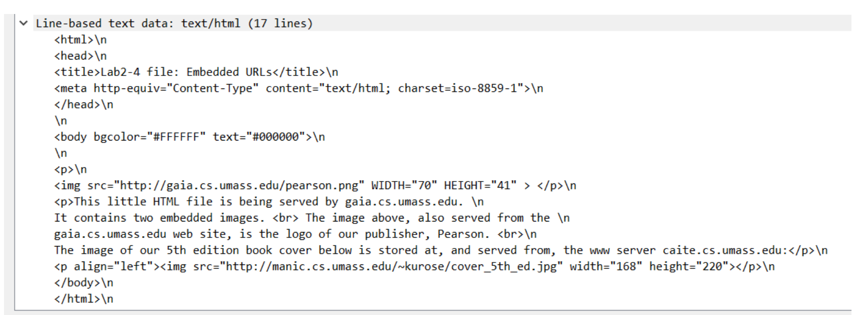


HTTP Response Message



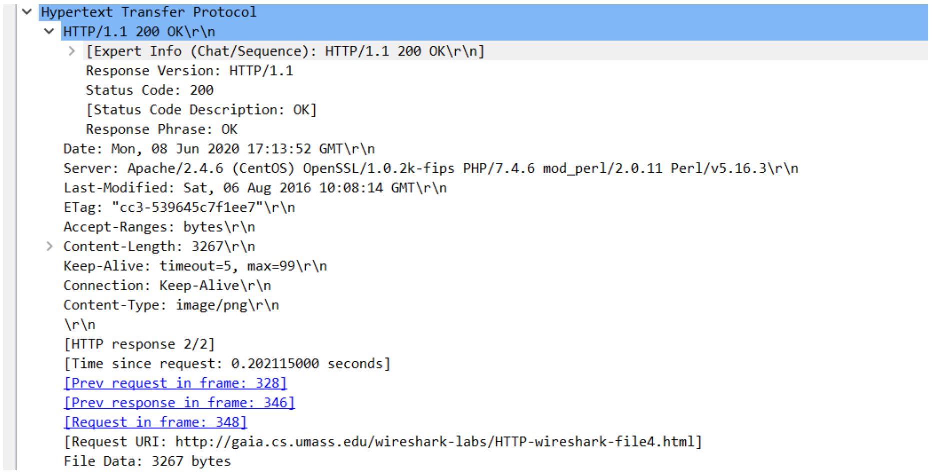
```
Wireshark · Packet 346 · CN_WS_CA#2_Q2.pcapng
v HTTP/1.1 200 OK\r\n
  > [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Date: Mon, 08 Jun 2020 17:13:51 GMT\r\n
  Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.6 mod_perl/2.0.11 Perl/v5.16.3\r\n
  Last-Modified: Mon, 08 Jun 2020 05:59:02 GMT\r\n
  ETag: "2ca-5a78c4ec149e1"\r\n
  Accept-Ranges: bytes\r\n
> Content-Length: 714\r\n
  Keep-Alive: timeout=5, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  \r\n
  [HTTP response 1/2]
  [Time since request: 0.200907000 seconds]
  [Request in frame: 328]
  [Next request in frame: 348]
  [Next response in frame: 365]
  [Request URI: http://gaia.cs.umass.edu/pearson.png]
  File Data: 714 bytes
```

Figure 2-6: GET message response for HTML content



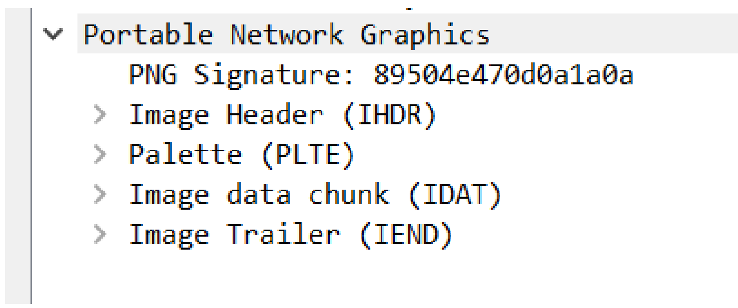
```
v Line-based text data: text/html (17 lines)
    <html>\n
    <head>\n
    <title>Lab2-4 file: Embedded URLs</title>\n
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">\n
    </head>\n
    \n
    <body bgcolor="#FFFFFF" text="#000000">\n
    \n
    <p>\n
    <img src="http://gaia.cs.umass.edu/pearson.png" WIDTH="70" HEIGHT="41" > </p>\n
    <p>This little HTML file is being served by gaia.cs.umass.edu. \n
    It contains two embedded images. <br> The image above, also served from the \n
    gaia.cs.umass.edu web site, is the logo of our publisher, Pearson. <br>\n
    The image of our 5th edition book cover below is stored at, and served from, the www server caite.cs.umass.edu:</p>\n
    <p align="left"><img src="http://manic.cs.umass.edu/~kurose/cover_5th_ed.jpg" width="168" height="220"></p>\n
    </body>\n
    </html>\n
```

Figure 2-7: HTML content as the answer embedded in the response message



```
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: Mon, 08 Jun 2020 17:13:52 GMT\r\n
  Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.6 mod_perl/2.0.11 Perl/v5.16.3\r\n
  Last-Modified: Sat, 06 Aug 2016 10:08:14 GMT\r\n
  ETag: "cc3-539645c7f1ee7"\r\n
  Accept-Ranges: bytes\r\n
  Content-Length: 3267\r\n
  Keep-Alive: timeout=5, max=99\r\n
  Connection: Keep-Alive\r\n
  Content-Type: image/png\r\n
  \r\n
  [HTTP response 2/2]
  [Time since request: 0.202115000 seconds]
  [Prev request in frame: 328]
  [Prev response in frame: 346]
  [Request in frame: 348]
  [Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html]
  File Data: 3267 bytes
```

Figure 2-8: GET message response for PNG content



```
Portable Network Graphics
    PNG Signature: 89504e470d0a1a0a
  Image Header (IHDR)
  Palette (PLTE)
  Image data chunk (IDAT)
  Image Trailer (IEND)
```

Figure 2-9: PNG content as the answer embedded in the response message

The JPEG response message has a similar response but a much larger content part

- Both responses of the two images were issued after getting both GET requests. This shows a parallel fetch for image data on the website.

Main Parts of a response message:

1. Status Line

2. Response Headers

1. Status Line

The first line is called the *status line*, followed by optional response header(s).

The status line has the following syntax:

```
HTTP-version status-code reason-phrase
```

- *HTTP-version*: The HTTP version used in this session. Either HTTP/1.0 and HTTP/1.1.
  - In Our case: HTTP/1.1
- *status-code*: a 3-digit number generated by the server to reflect the outcome of the request.
  - In Our case: 200
- *reason-phrase*: gives a short explanation to the status code.
  - In Our case: OK (message replied successfully (content fetched))
- Common status code and reason phrase are:

  "200 OK", "404 Not Found", "403 Forbidden", "500 Internal Server Error".

2. Response Headers:

The response headers are in the form (name : value) pairs:

```
response-header-name: response-header-value1, response-header-value2, ...
```

The response message body contains the resource data requested which in our case is the html text or the PNG, JPG files.

```
> Content-Length: 714\r\n
  Keep-Alive: timeout=5, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  \r\n
  [HTTP response 1/2]
  [Time since request: 0.200907000 seconds]
```

Figure 2-9: Response Headers for HTML response message

References

Websites used for PART I:

1. https://www.lifewire.com/what-is-a-dns-cache-817514

2. https://superuser.com/questions/235749/what-does-dns-flushing-do/235751#:~:text=This%20means%20that%20a%20website,for%20that%20length%20of%20time.&text=When%20you%20do%20a%20ipconfig,from%20the%20connected%20DNS%20server.

3. https://www.youtube.com/channel/UCKWaEZ-_VweaEx1j62do_vQ

4. https://ns1.com/resources/dns-types-records-servers-and-queries

5. http://www-inf.int-evry.fr/~hennequi/CoursDNS/NOTES-COURS_eng/msg.html#:~:text=Protocol%20and%20Format-,DNS%20Messages,(common%20requests%20and%20responses).

6. https://bluecatnetworks.com/blog/the-top-four-dns-response-codes-and-what-they-mean/

7. http://social.dnsmadeeasy.com/blog/long-short-ttls/

8. https://docs.microsoft.com/en-us/windows/win32/dns/managing-dns-resource-records#:~:text=A%20resource%20record%2C%20commonly%20referred,provide%20extended%20name%2Dresolution%20services.

9. http://www.steves-internet-guide.com/using-nslookup/

10. https://www.ibm.com/support/knowledgecenter/en/SS2JL7_5.1.1/com.ibm.omegamon.mn.doc_5.1.1/3270userguide/mfn_userguide_take_action_commands_nslookup_3270.htm

Websites used for PART II:

11. https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html#:~:text=GET%20is%20the%20most%20common,line)%20(optional%20request%20body)