

Part A:

1. There are HTTP, DNS, TCP, etc.

212	17.306586	192.168.0.245	192.168.0.1	DNS	81	Standard query 0xf28d A a2047.dscb.akamai.net
213	17.314127	128.119.245.12	192.168.0.245	TCP	66	80 → 57457 [ACK] Seq=1 Ack=513 Win=30080 Len=0 TSval=16
214	17.314128	128.119.245.12	192.168.0.245	HTTP	504	HTTP/1.1 200 OK (text/html)
215	17.314128	192.168.0.1	192.168.0.245	DNS	225	Standard query response 0xf28d A a2047.dscb.akamai.net
216	17.314129	192.168.0.1	192.168.0.245	DNS	155	Standard query response 0x39d4 HTTPS a2047.dscb.akamai.
217	17.314311	192.168.0.245	128.119.245.12	TCP	66	57457 → 80 [ACK] Seq=513 Ack=439 Win=131328 Len=0 TSval=1
218	17.315821	192.168.0.245	23.55.220.43	TCP	78	55447 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
219	17.317450	104.81.241.159	192.168.0.245	TCP	74	443 → 55446 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=
220	17.317749	192.168.0.245	104.81.241.159	TCP	66	55446 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=76
221	17.317751	192.168.0.245	104.81.241.159	TLSv1...	583	Client Hello
222	17.323691	23.55.220.43	192.168.0.245	TCP	74	443 → 55447 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=
223	17.323931	192.168.0.245	23.55.220.43	TCP	66	55447 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=31
224	17.323932	192.168.0.245	23.55.220.43	TLSv1...	583	Client Hello
225	17.329749	23.55.220.43	192.168.0.245	TCP	66	443 → 55447 [ACK] Seq=1 Ack=518 Win=30080 Len=0 TSval=6
226	17.333523	104.81.241.159	192.168.0.245	TCP	66	443 → 55446 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=5

2. From the below image we can see the GET arrival time is 10:54:03.550489000, the OK arrival time is 10:54:03.585651000, so it takes $0.585651000 - 0.550489000 = 0.035162\text{secs}$

197	17.278966	192.168.0.245	128.119.245.12	HTTP	578	GET /wiresh
214	17.314128	128.119.245.12	192.168.0.245	HTTP	504	HTTP/1.1 20
260	17.399320	192.168.0.245	128.119.245.12	HTTP	524	GET /favico
261	17.428942	128.119.245.12	192.168.0.245	HTTP	550	HTTP/1.1 40

Frame 197: 578 bytes on wire (4624 bits), 578 bytes captured (4624 bits) on interface e

> Interface id: 0 (en0)

Encapsulation type: Ethernet (1)

Arrival Time: Jan 11, 2022 10:54:03.550489000 CST

[Time shift for this packet: 0.000000000 seconds]

Epoch Time: 1641920043.550489000 seconds

[Time delta from previous captured frame: 0.000119000 seconds]

197	17.278966	192.168.0.245	128.119.245.12	HTTP	578	GET /wiresh
214	17.314128	128.119.245.12	192.168.0.245	HTTP	504	HTTP/1.1 20
260	17.399320	192.168.0.245	128.119.245.12	HTTP	524	GET /favico

Frame 214: 504 bytes on wire (4032 bits), 504 bytes captured (4032 bits) on interface en

> Interface id: 0 (en0)

Encapsulation type: Ethernet (1)

Arrival Time: Jan 11, 2022 10:54:03.585651000 CST

[Time shift for this packet: 0.000000000 seconds]

Epoch Time: 1641920043.585651000 seconds

[Time delta from previous captured frame: 0.000001000 seconds]

[Time delta from previous displayed frame: 0.035162000 seconds]

3. From the image below, my address (the source) is 192.168.0.245, the website's address (the destination) is 128.119.245.12

197	17.278966	192.168.0.245	128.119.245.12	HTTP	578	GET /wireshark-labs/INTRO-wiresh
214	17.314128	128.119.245.12	192.168.0.245	HTTP	504	HTTP/1.1 200 OK (text/html)

Frame 197: 578 bytes on wire (4624 bits), 578 bytes captured (4624 bits) on interface en0, id 0

Ethernet II, Src: Apple_78:ef:89 (f4:d4:88:78:ef:89), Dst: Tp-LinkT_4f:78:9f (c0:c9:e3:4f:78:9f)

Internet Protocol Version 4, Src: 192.168.0.245, Dst: 128.119.245.12

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 564

Identification: 0x0000 (0)

> Flags: 0x40, Don't fragment

...0 0000 0000 0000 = Fragment Offset: 0

Time to Live: 64

Protocol: TCP (6)

Header Checksum: 0x01a3 [validation disabled]

[Header checksum status: Unverified]

Source Address: 192.168.0.245

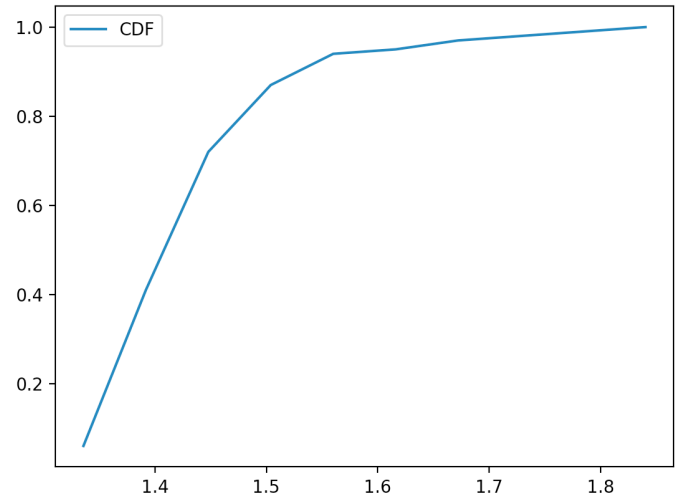
Destination Address: 128.119.245.12

Part B:

1. In circuit-switched networks, the resources are limited but reserved. Therefore the people on the phone can transfer voice messages to each other at the guaranteed constant rate and expect the minimum delay under the dedicated channel. However, packet-switching will be a nightmare for constant voice transfer. If the voice packets are lost during high-traffic times, the conversation no longer makes sense. And it doesn't have an end-to-end & dedicated connection between two hosts.
2. Because web browsing includes large pieces of data to be transferred and it asks for efficiency. Since packet-switching doesn't require a dedicated channel, it can use all bandwidth and make good use of space to transfer data at high speed. Also, it allows many users to 'communicate' at the same time (as opposed to two people talking on the phone). And if data fails to get transferred, it will be re-routed and re-sent, which is very convenient.
3. 1) It can happen due to congestion. If the current data exceeds the capacity, packets at the end of the queue might be dropped or missed.
2) It can also happen due to hardware problems. Say, the network work is down, and the many network switches and routers were out of power, then the packets can be missed.
4. 121 minutes = 7260 secs
4K-resolution time: $13\text{GB}/10\text{Mbps} = 13 \times 8 \times 10^9 \text{ bits} / 10^7 \text{ bps} = 10400 \text{ secs} > 7260$
HD-resolution time: $6\text{GB}/10\text{Mbps} = 6 \times 8 \times 10^9 \text{ bits} / 10^7 \text{ bps} = 4800 \text{ secs} < 7260$
SD-resolution time: $2\text{GB}/10\text{Mbps} = 2 \times 8 \times 10^9 \text{ bits} / 10^7 \text{ bps} = 1600 \text{ secs} < 7260$
So the best version is with HD resolution.
5. Fiber-optic connection is much faster than the ADSL connection at home. It helps reduce chances of internet congestion and greatly decreases delay time. Though the average bandwidth on campus is less than that at home, the data travels much faster than that at home.
6. a) It may happen because of network congestion. The traffic can be terrible if there are too many simultaneous active users sharing the same network service. So the aggregate arrival rate of packets exceeds the output capacity of the link. It may also be due to some outage in the area so that the data fails to/slowly get transferred.
b) Some certain websites might be very popular, but the web server is old and can only deal with a certain number of visitors→bad performance. It can also happen if the web server is located far away from where we are (like in another continent). The third possibility is that your router is improperly set.
7. I chose RFC 9156: <https://www.rfc-editor.org/rfc/rfc9156.pdf> to explain.
Almost every activity on the internet involves DNS queries. So it's very important to keep the privacy during information exchange. DNS queries can be something like "what are the QTYPE records of the QNAME? However, when the recursive resolver and name servers exchange information multiple rounds, the DNS queries reveal more information than needed even though the specific name server doesn't need to know the full QNAME. This is highly risky. This article is trying to minimize QNAME to improve privacy performance.

8. Left hand side is a screenshot of the output. Right hand side is CDF

```
64 bytes from 129.105.0.1: icmp_seq=67 ttl=252 time=1.37 ms
64 bytes from 129.105.0.1: icmp_seq=68 ttl=252 time=1.43 ms
64 bytes from 129.105.0.1: icmp_seq=69 ttl=252 time=1.33 ms
64 bytes from 129.105.0.1: icmp_seq=70 ttl=252 time=1.40 ms
64 bytes from 129.105.0.1: icmp_seq=71 ttl=252 time=1.41 ms
64 bytes from 129.105.0.1: icmp_seq=72 ttl=252 time=1.38 ms
64 bytes from 129.105.0.1: icmp_seq=73 ttl=252 time=1.36 ms
64 bytes from 129.105.0.1: icmp_seq=74 ttl=252 time=1.47 ms
64 bytes from 129.105.0.1: icmp_seq=75 ttl=252 time=1.40 ms
64 bytes from 129.105.0.1: icmp_seq=76 ttl=252 time=1.39 ms
64 bytes from 129.105.0.1: icmp_seq=77 ttl=252 time=1.41 ms
64 bytes from 129.105.0.1: icmp_seq=78 ttl=252 time=1.45 ms
64 bytes from 129.105.0.1: icmp_seq=79 ttl=252 time=1.39 ms
64 bytes from 129.105.0.1: icmp_seq=80 ttl=252 time=1.42 ms
64 bytes from 129.105.0.1: icmp_seq=81 ttl=252 time=1.44 ms
64 bytes from 129.105.0.1: icmp_seq=82 ttl=252 time=1.38 ms
64 bytes from 129.105.0.1: icmp_seq=83 ttl=252 time=1.51 ms
64 bytes from 129.105.0.1: icmp_seq=84 ttl=252 time=1.39 ms
64 bytes from 129.105.0.1: icmp_seq=85 ttl=252 time=1.39 ms
64 bytes from 129.105.0.1: icmp_seq=86 ttl=252 time=1.39 ms
64 bytes from 129.105.0.1: icmp_seq=87 ttl=252 time=1.39 ms
64 bytes from 129.105.0.1: icmp_seq=88 ttl=252 time=1.44 ms
64 bytes from 129.105.0.1: icmp_seq=89 ttl=252 time=1.58 ms
64 bytes from 129.105.0.1: icmp_seq=90 ttl=252 time=1.67 ms
64 bytes from 129.105.0.1: icmp_seq=91 ttl=252 time=1.46 ms
64 bytes from 129.105.0.1: icmp_seq=92 ttl=252 time=1.51 ms
64 bytes from 129.105.0.1: icmp_seq=93 ttl=252 time=1.42 ms
64 bytes from 129.105.0.1: icmp_seq=94 ttl=252 time=1.51 ms
```



Part C:

1. Running HTTP 1.1

- ▼ Hypertext Transfer Protocol
 - ▼ GET /gts1c3/MFcvVaADAgEAME4wTDBKMAkGBSs0AwIaBQAEMcueY
 - > [Expert Info (Chat/Sequence): GET /gts1c3/MFcvVaADAgEAME4wTDBKMAkGBSs0AwIaBQAEMcueY]
Request Method: GET
Request URI: /gts1c3/MFcvVaADAgEAME4wTDBKMAkGBSs0AwIaBQAEMcueY
Request Version: HTTP/1.1

2. en-US (US English)

```
User-Agent: com.apple.trustd/2.1\r\n
```

```
Accept-Language: en-US,en;q=0.9\r\n
```

```
Accept-Encoding: gzip, deflate\r\n
```

3. From the screenshot below, my IP address is 192.168.0.245, the server's is 128.119.245.12

5718	455.767501	192.168.0.245	128.119.245.12	HTTP	577 GET /wireshark-labs/HTTP-wir
5720	455.797694	128.119.245.12	192.168.0.245	HTTP	552 HTTP/1.1 200 OK (text/html)

Source Address: 192.168.0.245

Destination Address: 128.119.245.12

4. The status code is 200 OK

5718	455.767501	192.168.0.245	128.119.245.12	HTTP	577	GET /wireshark-labs/HTTP-v
5720	455.797694	128.119.245.12	192.168.0.245	HTTP	552	HTTP/1.1 200 OK (text/html)

5. From the screenshot below, it's last modified on Fri, 14 Jan 2022 06:59:01 GMT

5720	455.797694	128.119.245.12	192.168.0.245	HTTP	552	HTTP/1.1 200
------	------------	----------------	---------------	------	-----	--------------

Date: Sat, 15 Jan 2022 01:16:52 GMT\r\n

```
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.25 mod_perl/2.0.11 Perl/v5.
```

Last-Modified: Fri, 14 Jan 2022 06:59:01 GMT\r\n

6. 128 bytes

5720	455.797694	128.119.245.12	192.168.0.245	HTTP	552	HTTP/1.1	200	OK
------	------------	----------------	---------------	------	-----	----------	-----	----

Accept-Ranges: bytes\r\n

✓ Content-Length: 128\r\n

7. No they match exactly

8. No there is not.

9. Yes. I can tell from the “Line-based text data”.

455	4.342433	192.168.0.245	128.119.245.12	HTTP	577	GET /wireshark-labs/HTTP-wires
457	4.374090	128.119.245.12	192.168.0.245	HTTP	796	HTTP/1.1 200 OK (text/html)
756	28.410814	192.168.0.245	128.119.245.12	HTTP	689	GET /wireshark-labs/HTTP-wires
758	28.442690	128.119.245.12	192.168.0.245	HTTP	306	HTTP/1.1 304 Not Modified

Frame 457: 796 bytes on wire (6368 bits), 796 bytes captured (6368 bits) on interface en0, id 0
 Ethernet II, Src: Tp-LinkT_4f:78:9f (c0:c9:e3:4f:78:9f), Dst: Apple_78:ef:89 (f4:d4:88:78:ef:89)
 Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.0.245
 Transmission Control Protocol, Src Port: 80, Dst Port: 65397, Seq: 1, Ack: 512, Len: 730
 Hypertext Transfer Protocol
 Line-based text data: text/html (10 lines)

```

  \n
  <html>\n
  \n
  Congratulations again! Now you've downloaded the file lab2-2.html. <br>\n
  This file's last modification date will not change. <p>\n
  Thus if you download this multiple times on your browser, a complete copy <br>\n
  will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE<br>\n
  field in your browser's HTTP GET request to the server.\n
  \n
  </html>\n
  
```

10. Yes, the second request has “IF-MODIFIED-SINCE”. What follows is the last time that I access the page.

455	4.342433	192.168.0.245	128.119.245.12	HTTP	577	GET /wireshark-labs/HTTP-wires
457	4.374090	128.119.245.12	192.168.0.245	HTTP	796	HTTP/1.1 200 OK (text/html)
756	28.410814	192.168.0.245	128.119.245.12	HTTP	689	GET /wireshark-labs/HTTP-wires
758	28.442690	128.119.245.12	192.168.0.245	HTTP	306	HTTP/1.1 304 Not Modified

dnt: 1\r\n
 sec-gpc: 1\r\n
 If-None-Match: "173-5d5855675417f"\r\n
 If-Modified-Since: Fri, 14 Jan 2022 06:59:01 GMT\r\n

11. HTTP status code is: 304 Not modified. It didn't return the explicit content because the content is stored in cache. And we can just retrieve it from the cache.

12. Only one. Packet 242. (shown in the screenshot below problem 14)

13. Packet 247 (screenshot shown below problem 14)

14. 200 OK

242	22.128845	192.168.0.245	128.119.245.12	HTTP	472	GET /wireshark-labs/HTTP-wireshark-file3.
247	22.162243	128.119.245.12	192.168.0.245	HTTP	583	HTTP/1.1 200 OK (text/html)

> Frame 242: 472 bytes on wire (3776 bits), 472 bytes captured (3776 bits) on interface en0, id 0
 > Ethernet II, Src: Apple_78:ef:89 (f4:d4:88:78:ef:89), Dst: Tp-LinkT_4f:78:9f (c0:c9:e3:4f:78:9f)

15. 5 TCP segments

247	22.162243	128.119.245.12	192.168.0.245	Destination address	583	HTTP/1.1 200 OK (text/html)
-----	-----------	----------------	---------------	---------------------	-----	-----------------------------

> [Timestamps]
 > [SEQ/ACK analysis]
 TCP payload (517 bytes)
 TCP segment data (517 bytes)
 > [4 Reassembled TCP Segments (4861 bytes): #244(1448), #245(1448), #246(1448), #247(517)]

16. Sent three messages. They are 128.119.245.12 (the web page itself), 128.119.245.12(the pearson.png), 178.79.137.164(8e_cover_small.jpg). (Screenshot shown below problem 17)

17. My browser downloaded them serially. I can tell from the time of the response that the Pearson image comes first. And the second image is requested even after the first image request gets responded.

45	2.627042	192.168.0.245	128.119.245.12	HTTP	472	GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
47	2.658747	128.119.245.12	192.168.0.245	HTTP	1367	HTTP/1.1 200 OK (text/html)
49	2.662204	192.168.0.245	128.119.245.12	HTTP	498	GET /pearson.png HTTP/1.1
53	2.695405	128.119.245.12	192.168.0.245	HTTP	781	HTTP/1.1 200 OK (PNG)
58	2.762343	192.168.0.245	178.79.137.164	HTTP	465	GET /8E_cover_small.jpg HTTP/1.1
60	2.862551	178.79.137.164	192.168.0.245	HTTP	237	HTTP/1.1 301 Moved Permanently

18. It says "401 Unauthorized"

90	8.637323	192.168.0.245	128.119.245.12	HTTP	488	GET /wireshark-labs/protected_pages/HTTP-wireshark-labs-HTTP-wireshark-file4.html HTTP/1.1
92	8.669100	128.119.245.12	192.168.0.245	HTTP	783	HTTP/1.1 401 Unauthorized (text/html)
147	21.589696	192.168.0.245	128.119.245.12	HTTP	547	GET /wireshark-labs/protected_pages/HTTP-wireshark-labs-HTTP-wireshark-file4.html HTTP/1.1
149	21.624807	128.119.245.12	192.168.0.245	HTTP	556	HTTP/1.1 200 OK (text/html)

19. It sends with an Authorization Field:

90	8.637323	192.168.0.245	128.119.245.12	HTTP	488	GET /wireshark-labs/protected_pages/HTTP-wireshark-labs-HTTP-wireshark-file4.html HTTP/1.1
92	8.669100	128.119.245.12	192.168.0.245	HTTP	783	HTTP/1.1 401 Unauthorized (text/html)
147	21.589696	192.168.0.245	128.119.245.12	HTTP	547	GET /wireshark-labs/protected_pages/HTTP-wireshark-labs-HTTP-wireshark-file4.html HTTP/1.1
149	21.624807	128.119.245.12	192.168.0.245	HTTP	556	HTTP/1.1 200 OK (text/html)
154	21.699183	192.168.0.245	128.119.245.12	HTTP	445	GET /favicon.ico HTTP/1.1
156	21.729439	128.119.245.12	192.168.0.245	HTTP	551	HTTP/1.1 404 Not Found (text/html)

```
Host: gaia.cs.umass.edu\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.2 Safari/605.1.15\r\n
Accept-Language: en-US,en;q=0.9\r\n
Accept-Encoding: gzip, deflate\r\n
Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcmcs=\r\n
\r\n
```

Part D:

1. Chose one in Korea: www.aiit.or.kr : the IP address is 192.168.0.1

```
meimeisun@Meimeisuns-MacBookpro ~ % nslookup www.aiit.or.kr
Server:          192.168.0.1
Address:         192.168.0.1#53
```

```
Non-authoritative answer:
Name:   www.aiit.or.kr
Address: 58.229.6.225
```

2. This is an example of Cambridge

```
[meimeisun@Meimeisuns-MacBookpro ~ % nslookup -type=NS cam.ac.uk
Server:          192.168.0.1
Address:         192.168.0.1#53
```

```
Non-authoritative answer:
cam.ac.uk      nameserver = ns3.mythic-beasts.com.
cam.ac.uk      nameserver = ns2.ic.ac.uk.
cam.ac.uk      nameserver = ns1.mythic-beasts.com.
cam.ac.uk      nameserver = auth0.dns.cam.ac.uk.
cam.ac.uk      nameserver = dns0.eng.cam.ac.uk.
cam.ac.uk      nameserver = dns0.cl.cam.ac.uk.
```

```
Authoritative answers can be found from:
ns1.mythic-beasts.com  internet address = 45.33.127.156
ns3.mythic-beasts.com  internet address = 185.24.221.32
dns0.cl.cam.ac.uk      internet address = 128.232.0.19
dns0.eng.cam.ac.uk      internet address = 129.169.8.8
auth0.dns.cam.ac.uk     internet address = 131.111.8.37
ns1.mythic-beasts.com   has AAAA address 2600:3c00:e000:19::1
ns3.mythic-beasts.com   has AAAA address 2a02:2770:11:0:21a:4aff:febe:759b
dns0.cl.cam.ac.uk       has AAAA address 2001:630:212:200::d:a0
auth0.dns.cam.ac.uk     has AAAA address 2001:630:212:8::d:a0
```

3. nslookup -type=MX mail.yahoo.com

4. As shown in the screenshot

```
meimeisun@Meimeisuns-MacBookpro ~ % nslookup mail.yahoo.com ns1.mythic-beasts.com
Server:          ns1.mythic-beasts.com
Address:         45.33.127.156#53
```

```
** server can't find mail.yahoo.com: REFUSED
```

5. Sent over via UDP

569	152.730685	192.168.0.245	192.168.0.1	DNS	72	Standard query 0x0028 A www.ietf.org
571	153.118677	192.168.0.1	192.168.0.245	DNS	149	Standard query response 0x0028 A www.ietf.org CNAME
572	153.118678	192.168.0.1	192.168.0.245	DNS	187	Standard query response 0x01eb HTTPS www.ietf.org C
718	153.459539	192.168.0.245	192.168.0.1	DNS	78	Standard query 0x648d HTTPS analytics.ietf.org

Frame 569: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface en0, id 0
Ethernet II, Src: Apple_78:ef:89 (f4:d4:88:78:ef:89), Dst: Tp-LinkT_4f:78:9f (c0:c9:e3:4f:78:9f)
Internet Protocol Version 4, Src: 192.168.0.245, Dst: 192.168.0.1
User Datagram Protocol, Src Port: 59646, Dst Port: 53

6. From the above screenshot, we see the Destination port is 53, the source port is 59646
7. From the above screenshot, the destination is 192.168.0.1. The two IP addresses are the same.

8. It's type A, without any answers.

569	152.730685	192.168.0.245	192.168.0.1	DNS	72	Standard query 0x0028 A www.ietf.c
571	153.118677	192.168.0.1	192.168.0.245	DNS	149	Standard query response 0x0028 A w
572	153.118678	192.168.0.1	192.168.0.245	DNS	187	Standard query response 0x01eb HTT
718	153.459539	192.168.0.245	192.168.0.1	DNS	78	Standard query 0x648d HTTPS analvt

Authority RRs: 0
Additional RRs: 0
Queries
 > www.ietf.org: type A, class IN

9. Three answers. One answer contains the canonical name of the ietf website, and two other answers contain the ip addresses of the website which is the canonical version of ietf

569	152.730685	192.168.0.245	192.168.0.1	DNS	72	Standard query 0x0028 A www.ietf
571	153.118677	192.168.0.1	192.168.0.245	DNS	149	Standard query response 0x0028 A
572	153.118678	192.168.0.1	192.168.0.245	DNS	187	Standard query response 0x01eb H
718	153.459539	192.168.0.245	192.168.0.1	DNS	78	Standard query 0x648d HTTPS anal
719	153.459939	192.168.0.245	192.168.0.1	DNS	78	Standard query 0xcef5 A analytic
1057	153.619389	192.168.0.1	192.168.0.245	DNS	94	Standard query response 0xcef5 A
1125	153.915988	192.168.0.1	192.168.0.245	DNS	131	Standard query response 0x648d H

 > www.ietf.org: type A, class IN
Answers
 > www.ietf.org: type CNAME, class IN, cname www.ietf.org.cdn.cloudflare.net
 > www.ietf.org.cdn.cloudflare.net: type A, class IN, addr 104.16.44.99
 > www.ietf.org.cdn.cloudflare.net: type A, class IN, addr 104.16.45.99
[\[Request In: 569\]](#)

10. The destination IP address is 104.16.44.99. Yes, it matches with the one provided previously.

569	152.730685	192.168.0.245	192.168.0.1	DNS	72	Standard query 0x0028 A www.ietf.org
570	152.854559	Tp-LinkT_4f:78:9f	Apple_78:ef:89	ARP	42	192.168.0.1 is at c0:c9:e3:4f:78:9f
571	153.118677	192.168.0.1	192.168.0.245	DNS	149	Standard query response 0x0028 A www.ietf.org CNAME ww
572	153.118678	192.168.0.1	192.168.0.245	DNS	187	Standard query response 0x01eb HTTPS www.ietf.org CNAME
573	153.123550	192.168.0.245	104.16.44.99	TCP	78	61013 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
574	153.124074	192.168.0.245	104.16.44.99	TCP	78	61014 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
575	153.132025	104.16.44.99	192.168.0.245	TCP	66	443 → 61013 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
576	153.132025	104.16.44.99	192.168.0.245	TCP	66	80 → 61014 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=

11. No there's not.

12. The destination port is 53, the source port is 54944.

31	6.610233	192.168.0.245	192.168.0.1	DNS	71	Standard query 0xfc2e A www.mit.edu
32	6.647380	192.168.0.1	192.168.0.245	DNS	163	Standard query response 0xfc2e A ww

Frame 31: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface en0, id 0
Ethernet II, Src: Apple_78:ef:89 (f4:d4:88:78:ef:89), Dst: Tp-LinkT_4f:78:9f (c0:c9:e3:4f:78:9f)
Internet Protocol Version 4, Src: 192.168.0.245, Dst: 192.168.0.1
User Datagram Protocol, Src Port: 54944, Dst Port: 53
 Source Port: 54944
 Destination Port: 53

13. The IP address is 192,168.0.1. Yes, the same IP address.

14. Type A, no answers.

31	6.610233	192.168.0.245	192.168.0.1	DNS	71	Standard query 0xfc2e A www.mit.edu
32	6.647380	192.168.0.1	192.168.0.245	DNS	163	Standard query response 0xfc2e A ww

User Datagram Protocol, Src Port: 54944, Dst Port: 53

Domain Name System (query)

Transaction ID: 0xfc2e

> Flags: 0x0100 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 0

Queries

> www.mit.edu: type A, class IN

15. Three answers are provided: The first answer contains the CNAME of the second answer, the second answer contains the CNAME for the third answer, the third answer provides the corresponding IP address for the website.

31	6.610233	192.168.0.245	192.168.0.1	DNS	71	Standard query 0x
32	6.647380	192.168.0.1	192.168.0.245	DNS	163	Standard query re

Answer RRs: 3

Authority RRs: 0

Additional RRs: 0

Queries

> www.mit.edu: type A, class IN

Answers

> www.mit.edu: type CNAME, class IN, cname www.mit.edu.edgekey.net

> www.mit.edu.edgekey.net: type CNAME, class IN, cname e9566.dscb.akamaiedge.net

> e9566.dscb.akamaiedge.net: type A, class IN, addr 23.79.197.77

16. It is sent to my IP address: 192.168.0.1

20	14.239629	192.168.0.245	192.168.0.1	DNS	89	Standard query 0x1624 A addons-pa.clients6.google.com
21	14.248766	192.168.0.1	192.168.0.245	DNS	105	Standard query response 0x1624 A addons-pa.clients6.g
41	17.529293	192.168.0.245	192.168.0.1	DNS	67	Standard query 0x3ab6 NS mit.edu
42	17.541965	192.168.0.1	192.168.0.245	DNS	402	Standard query response 0x3ab6 NS mit.edu NS ns1-37.ak
45	20.227203	192.168.0.245	192.168.0.1	DNS	75	Standard query 0xf70d A play.google.com
54	20.233002	192.168.0.1	192.168.0.245	DNS	91	Standard query response 0xf70d A play.google.com A 14

17. Type NS query with no answers

21	14.248766	192.168.0.1	192.168.0.245	DNS	105	Standard query response 0x1624 /
41	17.529293	192.168.0.245	192.168.0.1	DNS	67	Standard query 0x3ab6 NS mit.edu
42	17.541965	192.168.0.1	192.168.0.245	DNS	402	Standard query response 0x3ab6 N
45	20.227203	192.168.0.245	192.168.0.1	DNS	75	Standard query 0xf70d A play.goc
54	20.233002	192.168.0.1	192.168.0.245	DNS	91	Standard query response 0xf70d /

Authority RRs: 0

Additional RRs: 0

Queries

> mit.edu: type NS, class IN

18. The servers can be seen from the screenshot below. As you can see from the first example, it doesn't provide any IP addresses.

41	17.529293	192.168.0.245	192.168.0.1	DNS	67	Standard query 0x3ab6 NS mit.edu
42	17.541965	192.168.0.1	192.168.0.245	DNS	402	Standard query response 0x3ab6 NS mit.edu NS ns1-37.aka
45	20.227203	192.168.0.245	192.168.0.1	DNS	75	Standard query 0xf70d A play.google.com

Queries

> mit.edu: type NS, class IN

Answers

> mit.edu: type NS, class IN, ns ns1-37.akam.net

Name: mit.edu

Type: NS (authoritative Name Server) (2)

Class: IN (0x0001)

Time to live: 1800 (30 minutes)

Data length: 17

Name Server: ns1-37.akam.net

> mit.edu: type NS, class IN, ns asia1.akam.net

> mit.edu: type NS, class IN, ns use5.akam.net

> mit.edu: type NS, class IN, ns use2.akam.net

> mit.edu: type NS, class IN, ns eur5.akam.net

> mit.edu: type NS, class IN, ns asia2.akam.net

> mit.edu: type NS, class IN, ns usw2.akam.net

> mit.edu: type NS, class IN, ns ns1-173.akam.net

19. It is sent to 8.8.8.8. No, it's not the one for my default DNS server. It corresponds to Google's server.

11	10.645800	192.168.0.245	192.168.0.1	DNS	90	Standard query 0xcd2 A google-public-dns-a.google.com
12	10.652476	192.168.0.1	192.168.0.245	DNS	106	Standard query response 0xcd2 A google-public-dns-a.go
13	10.654447	192.168.0.245	8.8.8.8	DNS	74	Standard query 0x2321 A www.aiit.or.kr
14	11.467354	8.8.8.8	192.168.0.245	DNS	90	Standard query response 0x2321 A www.aiit.or.kr A 58.22

20. Type A. No answer

11	10.645800	192.168.0.245	192.168.0.1	DNS	90	Standard query 0xcd2 A google-p
12	10.652476	192.168.0.1	192.168.0.245	DNS	106	Standard query response 0xcd2 A
13	10.654447	192.168.0.245	8.8.8.8	DNS	74	Standard query 0x2321 A www.aiit
14	11.467354	8.8.8.8	192.168.0.245	DNS	90	Standard query response 0x2321 A

Transaction ID: 0x2321

Flags: 0x0100 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 0

Queries

> www.aiit.or.kr: type A, class IN

--

21. There's one answer, containing the IP address of the website.

13	10.654447	192.168.0.245	8.8.8.8	DNS	74	Standard query 0x2321 A ww
14	11.467354	8.8.8.8	192.168.0.245	DNS	90	Standard query response 0x

> www.aiit.or.kr: type A, class IN

Answers

> www.aiit.or.kr: type A, class IN, addr 58.229.6.225

[\[Request In: 13\]](#)

Part E:

1. a) Query Name is a sequence of one or more labels, each beginning with a 1-byte count that specifies the number of bytes that follow. So the name is terminated with a byte of 0.

b) DNS header(12 bytes) + one question (15 bytes domain name + 2 bytes qtype + 2 bytes qclass) + one answer (16 bytes using pointer) = 47 bytes in total
2. When doing VO-IP, errors such as packet loss have minimal impacts on the audio output. UDP cares less about packet loss (at most some unnoticeable silence). However, TCP makes sure the packets are all delivered (not even a single loss). But in that case, users will experience intolerable delay and lag of the audio sent over.
3. When: Often used by default; often happens especially when the web server is very busy so that users can experience less delay.
Why: The web server and the client don't need to spend time establishing TCP, which consumes resources and results in significant delay. What's more, requests between the same server and the same client can be made back to back, without waiting for replies of pending requests.
4. Normally two parties must establish an SMTP connection to continue to send mail messages. Now, we can combine SMTP and the mail sending process. I.e., each mail message carries an SMTP connection request. I.e., sending a message and SMTP handshake happens at the same time. In this way, clients will experience more delays. But now SMTP is stateless.
5. Practically, the two will navigate you to the same website. But there is some underlying difference: when we enter the non-www address, we are actually being redirected to the one with www.
Two different ways:
 - a) After you enter the URL, the browser will look up the IP address of the domain (northwestern.edu) with the help of DNS. Then, TCP connection is established. Then the browser sends HTTP a request to the web server and the server sends back a response.
 - b) You visited the website before. So the IP address is cached, and you will directly jump to the TCP connection. The rest is the same as procedure a).
6. A website is often replicated over multiple servers, each having a different IP address. When doing requests during maintenance time, we can tell DNS to avoid the affected/rebooted web server and at the same time perform load distribution to unaffected servers.
7. Like mentioned in question 6, a large and busy website like Google often has multiple web servers. When a client makes a DNS query for Google, it responds with a set of IP addresses, but rotates the order of the addresses in response. (because a client normally first sends requests to the primary address) In this way, the traffic is fairly distributed to multiple servers.