

Practical 1

Watch the network and you will find out what is going on there

Alejandro Pérez

March 28th, 2023

Table of Contents

- Part 1 Link and Network Layers
 - Exercise 1
 - Exercise 2
 - Exercise 3
- Part 2 Transport Layer
 - Exercise 1
 - Exercise 2
 - Exercise 3
- Part 3 Application Layer
 - Exercise 1
 - Exercise 2
 - Exercise 3

Part 1 Link and Network Layers

Exercise 1

Click over a specific frame from the capture you have done with Wireshark. Check that it includes the Ethernet header with data from the link layer. Show via a screenshot the Ethernet header contents. Respond to the following questions:

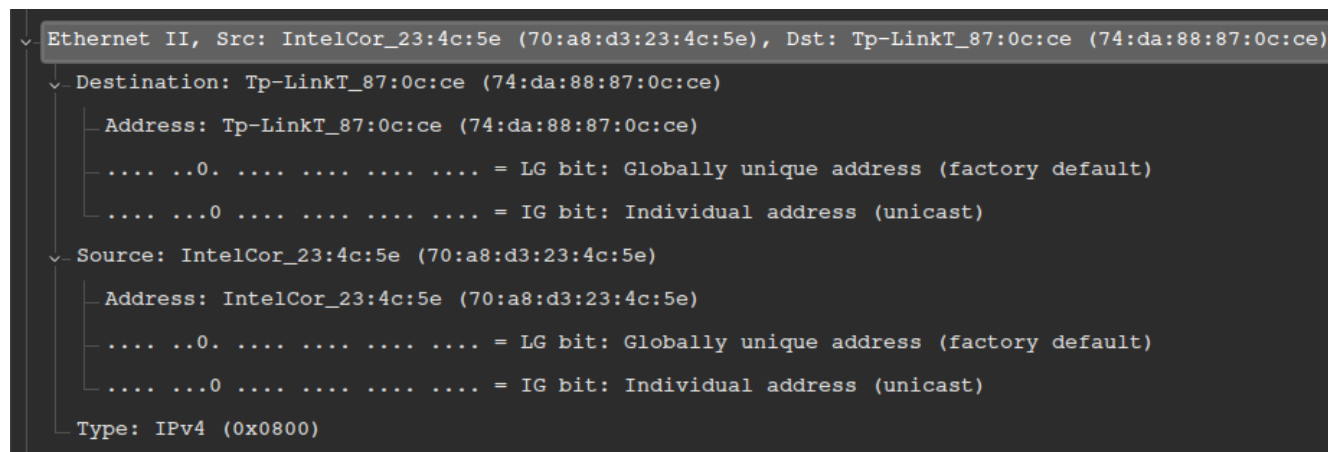


Figure 1: Ethernet header contents

a) **What are source and destination addresses? Who do you think they belong to?** src address: 70:a8:d3:23:4c:5e belongs to my network adapter.

dst address: 74:da:88:87:0c:ce belongs to my Wi-Fi router.

b) **Who assigns these addresses?** In both cases it is the hardware manufacturer that assigns these addresses. In the case of my PC it is assigned to my Network Interface Controller (NIC).

c) **What does type field mean?** The *type* field defines the type of network-layer protocol to use in a given Ethernet frame. This field is analogous to the *protocol* field of the network-layer datagram and to the *port-number* field from the transport-layer segment.

d) **Which function has the CRC field? Respond in a theoretical way.** The Cyclic Redundancy Check or *CRC* is an error-detection mechanism based on polynomial codes that ensures the integrity of a given piece of data.

Consider the following:

- A set of d -bits from the Data D
- A Generator bit pattern G which consists of $r + 1$ bits to form R known by both the sender and the receiver. The most significant bit of G will always be a 1.

From these two values we extract the following formula:

$$R = \text{remainder} \frac{D \cdot 2^r}{G}$$

Figure 2: CRC formula


e) **What is the purpose of the preamble in an Ethernet frame?** The preamble consists of *8 bytes* of data in which the first seven bytes contain 10101010 and the eighth one has 10101011. This helps the receiving adapters to realize that data is about to come in, as well as helps sync the sender's and receiver's clocks so that the receiver can 'lock' onto the sender. The last two bits of the preamble (11) are the final wake-up call to warn that actual data is about to be received next.

f) **What data does this Ethernet frame carry?** The Ethernet frame consists of the following:

- Preamble
- Destination address
- Source address
- Type field
- Data payload and padding
- CRC

Exercise 2

In the previous frame, notice that the detail of an IP header is also included. Show via a screenshot the IP header contents. Respond to the following questions



```
Internet Protocol Version 4, Src: 192.168.1.115, Dst: 142.250.200.131
- 0100 .... = Version: 4
- .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 60
- Identification: 0xfdeb (65003)
> 010. .... = Flags: 0x2, Don't fragment
- ...0 0000 0000 0000 = Fragment Offset: 0
- Time to Live: 64
- Protocol: TCP (6)
- Header Checksum: 0x2337 [validation disabled]
- [Header checksum status: Unverified]
- Source Address: 192.168.1.115
- Destination Address: 142.250.200.131
```

Figure 3: IP header contents

a) **What are source and destination addresses? Who do you think they belong to?** src address: 192.168.1.115 is the IP address of my computer on my home network

dst address: 142.250.200.131 is the public address of the destination website I want to access (google.com)

b) Why does the packet have an identifier? Because of fragmentation in IPv4, packets are identified with an identifier, which is set by the sender and typically incremented for every packet. This is done so that the receiver can verify that the packets all arrive and do so in the proper order, as well as to determine which fragments belong to a given datagram.

c) Which flags are active in this packet? The only active flag is 0x2 Don't fragment

d) Explain what the TTL value means in the analyzed packet. The Time to Live or TTL for this particular frame is 64. This means that the data in this frame will travel up to 64 nodes in the network before reaching my computer, otherwise it will be discarded by the router.

e) Why is a checksum field needed again in the IP protocol? In order to verify the integrity of the transmitted frames.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
IP Protocol Types	63949				0,0469	100%	1,9500	518,162
UDP	1384				0,0010	2,16%	0,1600	1111,902
TCP	62555				0,0459	97,82%	1,9500	518,162
NONE	10				0,0000	0,02%	0,0100	121,039

Figure 4: IP protocol types

f) Go to menu Statistics > IPv4 statistics > IP protocol types. Which is the protocol sending more packets? Why? Clearly TCP is sending more packets because it is proven to be reliable, secure, and has error detection and correction features. Other Internet protocols such as UDP have no handshake, and can be less secure or reliable. One good use case of UDP could be video calls.

Exercise 3

Finally, without applying any filters, go to the statistics menu Statistics > Protocol Hierarchy. Show in a screenshot the results and comment them, relating them with packet encapsulation and de-encapsulation, a fundamental pillar of network communications.

We can see from the statistics above that IPv4 is still the most widely used version of the IP protocol, whereas IPv6 only accounts for 2% of my computer's traffic. Focusing on IPv4, we see that about 95% of the traffic goes to the *TCP* (Transmission Control Protocol) whereas only 2% is related to *UDP* (User Datagram Control). This is largely due to the fact that TCP has mechanisms for ensuring security between two communication ends, using a handshake between the two parties before transmitting any payload. Also, thanks to TCP's packet fragmentation, TCP packets can scale very well on slower networks, ensuring that every fragment of a packet is received by the receiver in the right order and has little odds of bit corruption.

Wireshark - Protocol Hierarchy Statistics - wlan0

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	74743	100.0	68677486	329 k	0	0	0	74743
Ethernet	100.0	74743	1.5	1046402	5.020	0	0	0	74743
Internet Protocol Version 6	2.1	1592	0.1	63680	305	0	0	0	1592
Internet Control Message Protocol v6	2.1	1592	0.1	80764	387	1592	80764	387	1592
Internet Protocol Version 4	97.6	72981	2.1	1459672	7.003	0	0	0	72981
User Datagram Protocol	2.3	1701	0.0	13608	65	0	0	0	1701
Network Time Protocol	0.0	10	0.0	480	2	10	480	2	10
NetBIOS Name Service	0.0	32	0.0	1996	9	32	1996	9	32
NetBIOS Datagram Service	0.0	4	0.0	804	3	0	0	0	4
SMB (Server Message Block Protocol)	0.0	4	0.0	476	2	0	0	0	4
SMB MailSlot Protocol	0.0	4	0.0	100	0	0	0	0	4
Microsoft Windows Browser Protocol	0.0	4	0.0	132	0	4	132	0	4
Dynamic Host Configuration Protocol	0.0	4	0.0	1192	5	4	1192	5	4
Domain Name System	2.1	1592	0.1	75616	362	1592	75616	362	1592
Data	0.1	59	0.0	16992	81	59	16992	81	59
Transmission Control Protocol	95.3	71243	95.9	65890800	316 k	49579	33179131	159 k	71243
Transport Layer Security	28.7	21443	96.3	66167871	317 k	21443	50696878	243 k	25326
Malformed Packet	0.0	2	0.0	0	0	2	0	0	2
Hypertext Transfer Protocol	0.3	202	0.1	99152	475	101	27221	130	202
Line-based text data	0.1	101	0.0	25030	120	101	25030	120	101
Data	0.0	17	0.0	32078	153	17	32078	153	17
Internet Group Management Protocol	0.0	13	0.0	156	0	13	156	0	13
Internet Control Message Protocol	0.0	24	0.0	10524	50	18	10008	48	24
Domain Name System	0.0	6	0.0	300	1	6	300	1	6
Address Resolution Protocol	0.2	170	0.0	4760	22	170	4760	22	170

No display filter.

Help Copy Close

Figure 5: Protocol Hierarchy

Part 2 Transport Layer

Exercise 1

First of all, you have to analyze the UDP protocol. You can filter this protocol in Wireshark and you will only see packets from this protocol. Show via a screenshot the header fields and respond to the following questions:

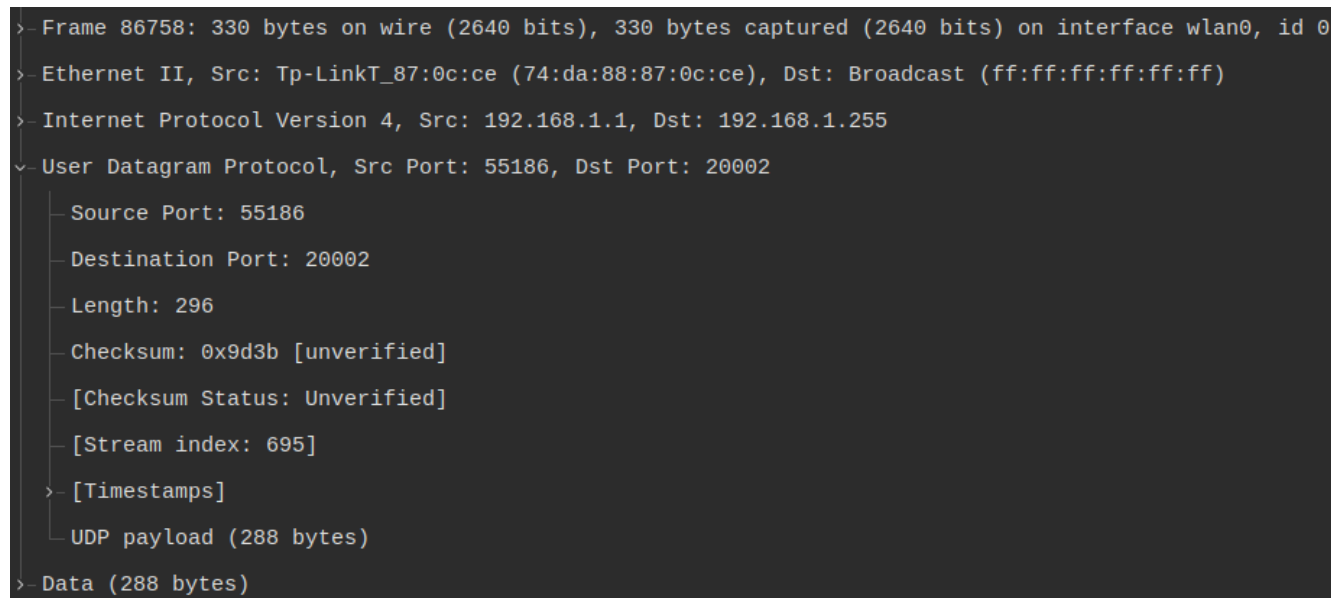


Figure 6: UDP header fields

a) **Which protocol/s from the application layer has/have generated these packets?** Judging by the destination port, it seems like this corresponds to an *HTTP* request.

b) **Why do you think UDP is used instead of TCP?** TBA

c) **Why do values have source and destination ports? What does it mean?** Ports are used in UDP to determine the type of network service to be used. The destination port refers to the desired network services, whereas the source port is typically randomized by the system, and is used to create a session for such network service.

d) **Why is the checksum field required?** Because UDP is a best-effort connectionless protocol, it does not care if packets are dropped, if they are corrupted or if they are sent in an incorrect order. However, UDP provides a checksum field that only verifies the integrity of the packages, but does not provide any sort of error correction, unlike TCP.

e) **How is it calculated?** All the segments of the payload are added up and then turned into the complementary, i.e. turning all the 1s into 0s and all the 0s into 1s. The receiver would then do the same sum and check that the calculated checksum and the original checksum match.

f) **What is the value of the length field? Check with hexadecimal values of the UDP packet that this is the value indeed.** The *length* field value is 296.

From the screenshot above, the hexadecimal code for the *length* field is 01 28 which represents 296 in decimal

Exercise 2

Next, we have to check how the TCP protocol works. Go to the statistics menu Statistics > Flow Graph, and select only TCP type, checking the option Flow type > TCP flow. Show in a screenshot the phases where the connection is established and finished. Explain the process. What is the flag PSH used for?

0000	ff ff ff ff ff ff 74 da	88 87 0c ce 08 00 45 00t.....E
0010	01 3c 00 00 40 00 40 11	b5 60 c0 a8 01 01 c0 a8	<.@.@`.....
0020	01 ff d7 92 4e 22 01 28	9d 3b 01 f0 00 01 01 10	..N"·(;.....
0030	11 00 00 00 00 00 33 3d	a3 da 81 a3 ce ab df b73=.....
0040	d8 bc 9e a4 86 f6 84 eb	89 ec ce e2 c0 a4 c5 b1
0050	d0 f2 c8 b3 91 f6 84 eb	9e ee b1 d8 bc 9e a4 86
0060	e2 d3 b7 8e b9 db e8 d9	f4 c1 f1 c5 a6 8b e8 dc
0070	eb db f6 c5 f3 cb fb d6	e5 d6 b7 d2 b6 86 b4 d6
0080	e0 d4 e1 d4 f6 da f8 95	f4 97 b5 8f ad 9a ae 83
0090	c7 86 ab 93 ab 86 be 89	a4 94 d7 fa b9 fc de f2
00a0	d0 b9 c9 eb d1 f3 c2 fb	c9 e7 d6 e0 d8 f6 c7 e9
00b0	d8 fa d6 f4 99 f6 92 f7	9b b9 83 a1 e0 92 f1 99
00c0	fc 8e ae e3 b1 87 b7 87	a5 89 ab c4 b4 d1 a3 c2
00d0	b6 df b0 de 81 ec 83 e7	82 a0 9a b8 f9 a9 8b a7
00e0	85 f5 87 e8 8c f9 9a ee	b1 c5 bc cc a9 8b b1 93
00f0	df 8b ce 89 e8 9c f9 8e	ef 96 b4 98 ba d5 bb de
0100	b3 d6 a5 cd 92 e1 94 e4	94 fb 89 fd df e5 91 e3
0110	96 f3 df fd 92 fc 99 f4	91 e2 8a d5 a7 c8 a4 c1
0120	e3 d9 fb 96 f7 84 f0 95	e7 c5 e9 cb a4 ca af c2
0130	a7 d4 bc e3 90 e5 95 e5	8a f8 8c d3 a5 c0 b2 c1
0140	a8 c7 a9 8b b1 ea db 86	fb 86

Figure 7: Hex values

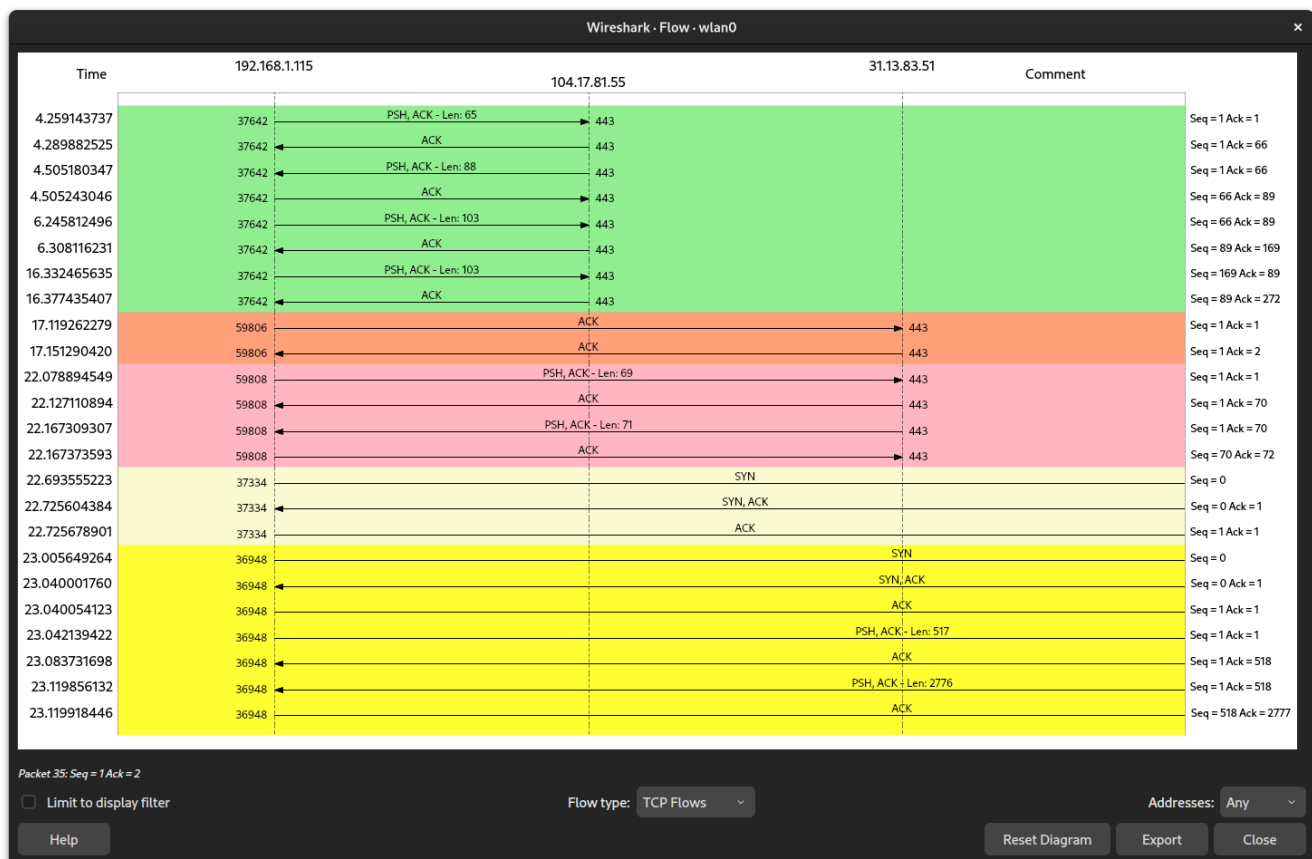


Figure 8: Tcp flow graph

The *PSH* flag is used to ensure that data is transmitted as soon as possible, rather than waiting for the buffer to store more data to fill a packet. The receiving end is also notified that this data is to be pushed to the application as soon as possible.

Exercise 3

Go to the main packets list. You can filter using TCP protocol to see only packets concerning this protocol. Select a packet and show in a screenshot the TCP header fields content and respond to the following questions:

```
> Frame 33980: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlan0, id 0
> Ethernet II, Src: Tp-LinkT_87:0c:ce (74:da:88:87:0c:ce), Dst: IntelCor_23:4c:5e (70:a8:d3:23:4c:5e)
> Internet Protocol Version 4, Src: 172.65.251.78, Dst: 192.168.1.115
> Transmission Control Protocol, Src Port: 443, Dst Port: 57302, Seq: 220, Ack: 860, Len: 0
  - Source Port: 443
  - Destination Port: 57302
  - [Stream index: 229]
  - [Conversation completeness: Complete, WITH_DATA (31)]
  - [TCP Segment Len: 0]
  - Sequence Number: 220      (relative sequence number)
  - Sequence Number (raw): 2783053281
  - [Next Sequence Number: 220      (relative sequence number)]
  - Acknowledgment Number: 860      (relative ack number)
  - Acknowledgment number (raw): 1453116194
  - 1000 .... = Header Length: 32 bytes (8)
> Flags: 0x010 (ACK)
  - Window: 8
  - [Calculated window size: 65536]
  - [Window size scaling factor: 8192]
  - Checksum: 0x4a6a [unverified]
  - [Checksum Status: Unverified]
  - Urgent Pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [Timestamps]
> [SEQ/ACK analysis]
```

Figure 9: TCP header fields

a) Which is the sequence number? What is it used for? The sequence number is 220 and it is used in TCP to identify a packet to be transmitted and keep track how much data is yet to be transmitted. The *seq* (sequence number) is sent in a packet, and the receiver confirms that it has correctly received the packet by sending an *ack*.

b) And the ACK number? The *ack* (acknowledgement) is the code sent from the receiver back to the sender to indicate if the sent data is corrupt or otherwise invalid, or if it has received it correctly. If the packet has been correctly received, the *ack* is the sequence number of the next packet that is expected to be transmitted by the sender. The *ack* for this transmission is 860.

c) Which values have source and destination ports? What does it mean? Source Port: 443

Destination Port: 57302

Similarly to UDP, the source port refers to a port generated from the computer to create a session to be used in a destination port representing a specific network application.

d) Which flags are active in this packet? The only active flag for this package is the *ACK* flag, as this corresponds to an *ack* message.

e) Explain what the values relative to the window mean. The TCP window refers to the maximum number of segments that can be transmitted by the sender before an *ack* is returned.

f) Which data does this TCP packet carry? This TCP packet carries no payload, as it only confirms with an *ack* that the last packet was sent successfully.

Part 3. Application Layer

Exercise 1

First of all, you have to analyze a DNS header from a request and a response, so you can filter in Wireshark by DNS protocol and you will see only packets from this protocol. Show a screenshot with the header fields and respond to the following questions:

```

> Frame 32961: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface wlan0, id 0
> Ethernet II, Src: IntelCor_23:4c:5e (70:a8:d3:23:4c:5e), Dst: Tp-LinkT_87:0c:ce (74:da:88:87:0c:ce)
> Internet Protocol Version 4, Src: 192.168.1.115, Dst: 208.67.222.222
> User Datagram Protocol, Src Port: 42899, Dst Port: 53
> Domain Name System (query)
  - Transaction ID: 0xb513
  > Flags: 0x0100 Standard query
  - Questions: 1
  - Answer RRs: 0
  - Authority RRs: 0
  - Additional RRs: 0
  > Queries

```

Figure 10: DNS header

a) Which source and destination ports are used? src: 42899 (UDP)
dst: 53 (UDP)

b) What is the transaction identifier for? It is used to identify a query for a domain, which can later be remembered to prevent asking over and over for the same domain.

c) What do flags mean? The flags explain that it is a standard query, done recursively, and that the response message is not to be truncated.

```

  - 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)

```

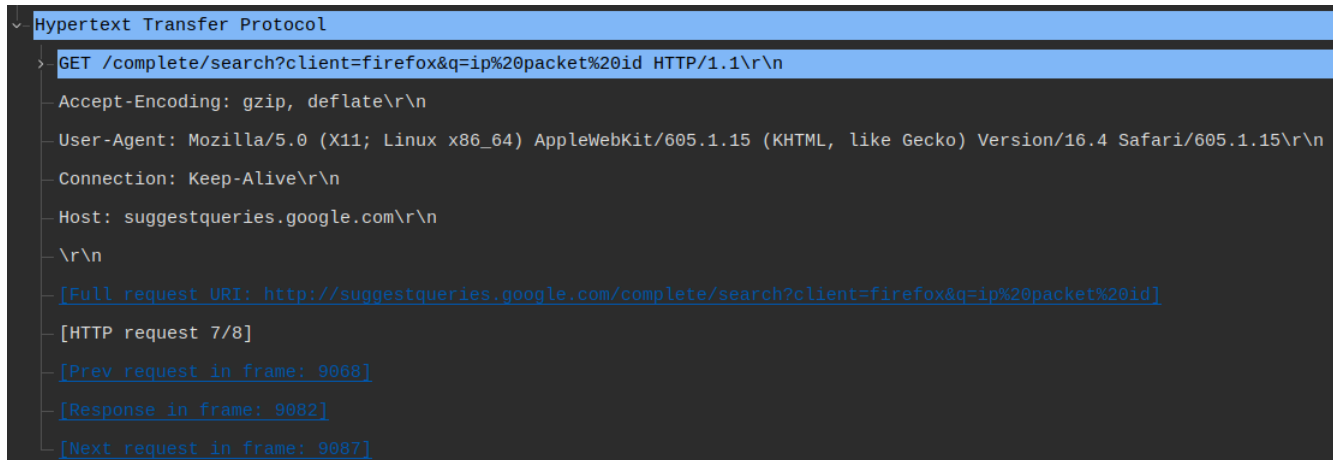
Figure 11: DNS flags

d) In the request, which domain is requested? google.com

- e) **Which kind and class this domain is? What does it mean?** Type A: Corresponds to a Host Address
Class IN : Refers to an INternet query
- f) **Can there be multiple records (RR, resource records) in the response?** Not for this query.

Exercise 2

Now analyze an HTTP header from a GET request. Show a screenshot with the header fields and respond to the following questions:



```
✓ Hypertext Transfer Protocol
  > GET /complete/search?client=firefox&q=ip%20packet%20id HTTP/1.1\r\n
  Accept-Encoding: gzip, deflate\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.4 Safari/605.1.15\r\n
  Connection: Keep-Alive\r\n
  Host: suggestqueries.google.com\r\n
  \r\n
  [Full request URI: http://suggestqueries.google.com/complete/search?client=firefox&q=ip%20packet%20id]
  [HTTP request 7/8]
  [Prev request in frame: 9068]
  [Response in frame: 9082]
  [Next request in frame: 9087]
```

Figure 12: HTTP GET request

- a) **Which source and destination ports are used?** src: 53148 (TCP)
dst: 80 (TCP)
- b) **Which is the version of the HTTP protocol?** HTTP Version 16.4
- c) **Are persistent connections used?** Yes, as connection is set to *Keep-Alive*
- d) **Which language is used? Why is it provided?** TBA
- e) **Which kind of content could be processed?** TBA
- f) **Which coding formats are used?** TBA

Exercise 3

Analyze an HTTP header of a successful GET response. Show a screenshot with the header fields and respond to the following questions:

- a) **What does this response contain?** It contains some extra information like the date or expiration date, but mainly it contains a Javascript snippet code with the answer to the query.
- b) **Which kind of content does it contain?** Line-Based Text data.
- c) **Is the content fragmented? How do we know?** It is, because the header gives us information on the different chunks of data and the encoding for the end.

```
Hypertext Transfer Protocol
> HTTP/1.1 200 OK\r\n
- Date: Tue, 28 Mar 2023 19:56:54 GMT\r\n
- Pragma: no-cache\r\n
- Expires: -1\r\n
- Cache-Control: no-cache, must-revalidate\r\n
- Content-Type: text/javascript; charset=UTF-8\r\n
- Content-Security-Policy: object-src 'none';base-uri 'self';script-src 'nonce-JlI9P-UGJmOG7EWKV6YPEQ' 'strict-dynamic' 'report-sample'
- Content-Disposition: attachment; filename="f.txt"\r\n
- Content-Encoding: gzip\r\n
- Server: gws\r\n
- X-XSS-Protection: 0\r\n
- X-Frame-Options: SAMEORIGIN\r\n
- Transfer-Encoding: chunked\r\n
- \r\n
- [HTTP response 7/8]
- [Time since request: 0.106808260 seconds]
- [Prev request in frame: 9068]
- [Prev response in frame: 9074]
- [Request in frame: 9076]
- [Next request in frame: 9087]
- [Next response in frame: 9093]
- [Request URI: http://suggestqueries.google.com/complete/search?client=firefox&q=ip%20packet%20id]
> HTTP chunked response
- Content-encoded entity body (gzip): 140 bytes -> 268 bytes
- File Data: 268 bytes
> Line-based text data: text/javascript (1 lines)
```

Figure 13: HTTP successful GET

d) Which fields are related to dates? What do they mean? Date: date of the response

Expires: whether the response is set to automatically expire

e) Are cookies used? How do they work? TBA

f) Describe what the fields related to caches mean. TBA

Exercise 4

Go to the Wireshark statistics menu Statistics > HTTP > Packet counter. Show a screenshot where the unsuccessful responses received are seen (they have value Count > 0). Describe what each error code means and how many packets of each packet have been received. Afterwards, select an error code and look for it on the main packet capture screen. Show a screenshot with the header fields with a brief explanation.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Total HTTP Packets	671				0,0001	100%	0,0500	4807,363
Other HTTP Packets	7				0,0000	1,04%	0,0200	4416,583
HTTP Response Packets	329				0,0001	49,03%	0,0200	360,503
??? : broken	0				0,0000	0,00%	-	-
5xx: Server Error	0				0,0000	0,00%	-	-
4xx: Client Error	0				0,0000	0,00%	-	-
3xx: Redirection	1				0,0000	0,30%	0,0100	273,005
301 Moved Permanently	1				0,0000	100,00%	0,0100	273,005
2xx: Success	328				0,0001	99,70%	0,0200	360,503
200 OK	328				0,0001	100,00%	0,0200	360,503
1xx: Informational	0				0,0000	0,00%	-	-
HTTP Request Packets	335				0,0001	49,93%	0,0300	4807,363
GET	335				0,0001	100,00%	0,0300	4807,363

Display filter: Apply Copy Save as... Close

Figure 14: Packet counter