

Wireshark CA#1

COMPUTER NETWORKS

SAHAND KHOSHDEL - ST ID: 810196607

TABLE OF CONTENTS

INTRODUTION	
PART 1	3
PART 2	8
PART 3	15
REFERENCES	

INTRODUCTION

In this Computer Assignment the goal is to explore 3 important Network Protocols :

- HTTP in the Application Layer
- ARP in the Data link layer
- DHCP Also running in the Application Layer
- In Each Part we will analyze the Source and Destination Addresses (MAC, IP), Message Types, headers, timing parameters, etc.

Part 1

1.1:

The source address indicates the address which my PC has been given according to the Internet Protocol .

The destination address is the address of "ece.ut.ac.ir" according to the Internet Protocol .

Source IP address	Destination IP address
192.168.1.8	80.66.179.158

Table 1.1

	terms terms						
htt	l http						
No.	Time	Source	Destination	Protocol	Length	Info	
-	528 3.911100	192.168.1.8	80.66.179.158	HTTP	504	GET /documents/70819125/2017cca1-b036-41de-bcce-f7376699275b HTTP/1.1	
4	544 3.968776	80.66.179.158	192.168.1.8	HTTP	280	HTTP/1.1 302	
> F	> Frame 528: 504 bytes on wire (4032 bits), 504 bytes captured (4032 bits) on interface \Device\NPF_{89CD229C-CEE1-406B-A2B1-4BE85F267A7E}, id 0						
> E	Ethernet II, Src: AzureWav_34:c6:af (f0:03:8c:34:c6:af), Dst: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4)						
> I	> Internet Protocol Version 4, Src: 192.168.1.8, Dst: 80.66.179.158						
> T	Transmission Control Protocol, Src Port: 1335, Dst Port: 80, Seq: 1, Ack: 1, Len: 450						
> H	Hypertext Transfer Protocol						

Figure 1.1

1.2:

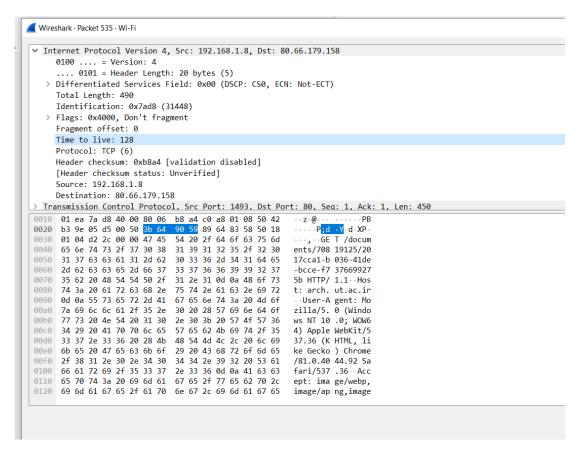


Figure 1.2

Time to live: 128

- Time to live (TLL) also known as "Hop Limit" is a kind of mechanism which limits the lifespan of data in a computer network. Whenever this time is reached the data is either discarded or revalidated.
- TLL can be implemented in data packets as a counter attached to data frames or embedded in the data.
- Under the Internet Protocol, TLL is an **8-bit field** in the **IPv4 header**.¹

¹ In theory TLL (under the IPv4 protocol) is mentioned in seconds

• The purpose of the TTL field is to avoid a situation in which an undeliverable datagram keeps circulating on an Internet system. The TTL field is set by the sender of the datagram, and reduced by every router on the route to its destination. If the TTL field reaches zero before the datagram arrives at its destination, then the datagram is discarded and an Internet Control Message Protocol (ICMP) error datagram is sent back to the sender.

1.3,1.4:

The 48-bit address of user's computer is indicated in Hexa-Decimal (12 hexa-decimal digits seperated pairwise, The Second 12 digits of the first row) – Displayed in **Figure 1.3**

Figure 1.3

The 48-bit address of destination is aslo indicated in Hexa-Decimal (12 hexa-decimal digits seperated pairwise, The First 12 digits of the first row) Displayed in **Figure 1.4**

Wireshark Packet 3122 · Wi-Fi > Frame 3122: 504 bytes on wire (4032 bits), 504 bytes captured (4032 bits) on interface \Devic > Ethernet II, Src: AzureWav_34:c6:af (f0:03:8c:34:c6:af), Dst: D-LinkIn_35:62:c4 (c4:e9:0a:35: > Destination: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4) > Source: AzureWav_34:c6:af (f0:03:8c:34:c6:af) Type: IPv4 (0x0800) > Internet Protocol Version 4, Src: 192.168.1.8, Dst: 80.66.179.158 > Transmission Control Protocol, Src Port: 12246, Dst Port: 80, Seq: 1, Ack: 1, Len: 450 > Hypertext Transfer Protocol

Figure 1.4

...5b. .. .4....E

--@-@----PB

·····GF T /docum

1.5: Header Size : 20 bytes (Both TCP & IP headers are 20 bytes long)

0020 b3 9e 2f d6 00 50 8a cb f4 f2 e3 0a 4e 49 50 18 0030 01 04 cf 93 00 00 47 45 54 20 2f 64 6f 63 75 6d

```
■ Wireshark · Packet 3122 · Wi-Fi

> Frame 3122: 504 bytes on wire (4032 bits), 504 bytes captured (4032 bits) on inte
Ethernet II, Src: AzureWav 34:c6:af (f0:03:8c:34:c6:af), Dst: D-LinkIn 35:62:c4 (
Internet Protocol Version 4, Src: 192.168.1.8, Dst: 80.66.179.158
     0100 .... = Version: 4
     .... 0101 = Header Length: 20 bytes (5)
   > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
     Total Length: 490
     Identification: 0x4093 (16531)
   > Flags: 0x4000, Don't fragment
     Fragment offset: 0
     Time to live: 128
     Protocol: TCP (6)
     Header checksum: 0xf2e9 [validation disabled]
      [Header checksum status: Unverified]
     Source: 192.168.1.8
                                                          ---5b-----<u>E</u>-
 0000 c4 e9 0a 35 62 c4 f0 03 8c 34 c6 af 08 00 <mark>45</mark> 00
 0010 01 ea 40 93 40 00 80 06 f2 e9 c0 a8 01 08 50 42
                                                          ..@.@... PB
 0020 b3 9e 2f d6 00 50 8a cb f4 f2 e3 0a 4e 49 50 18
                                                          ../..P....NIP.
 0030 01 04 cf 93 00 00 47 45 54 20 2f 64 6f 63 75 6d
                                                          ·····GE T /docum
0040 65 6e 74 73 2f 37 30 38 31 39 31 32 35 2f 32 30
                                                          ents/708 19125/20
```

Figure 1.5

Wireshark · Packet 3122 · Wi-Fi

```
> Internet Protocol Version 4, Src: 192.168.1.8, Dst: 80.66.179.158
Transmission Control Protocol, Src Port: 12246, Dst Port: 80, Seq: 1, Ack: 1, Len: 45
     Source Port: 12246
     Destination Port: 80
     [Stream index: 13]
     [TCP Segment Len: 450]
     Sequence number: 1
                           (relative sequence number)
     Sequence number (raw): 2328622322
                                    (relative sequence number)]
     [Next sequence number: 451
     Acknowledgment number: 1
                                 (relative ack number)
     Acknowledgment number (raw): 3809103433
     0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
     Window size value: 260
     [Calculated window size: 66560]
0020 b3 9e 2f d6 00 50 8a cb f4 f2 e3 0a 4e 49 50 18
                                                          ../..P.. ....NI<mark>P</mark>.
0030 01 04 cf 93 00 00 47 45 54 20 2f 64 6f 63 75 6d
                                                          ·····GE T /docum
0040 65 6e 74 73 2f 37 30 38 31 39 31 32 35 2f 32 30
                                                          ents/708 19125/20
      24 27 62 62 64 24 24 62
                               20 22 26 24 24 24 64 66
                                                          47---4 L ASC 443-
                                      Figure 1.6
```

1.6:

The "O" in the "OK" response consumes the first 52 bytesnod the Ethernet frame, and after 14 Ethernet frame bytes, IP Header comes in 20 bytes and TCP Header in 20 bytes also. Then the Data (HTTP) comes in the frame.

Part 2

Introducing the ARP protocol:

The ARP protocol functionally consists of 2 main parts:

- 1: The part which determines a physical address when sending a packet
- 2: The Other part answers requests from other machines

So ARP provides method for hosts to send message to destination address on a physical network.

Ethernet hosts must convert a 32-bit IP address into a 48-bit Ethernet address. The host checks its ARP cache to see if address mapping from IP to physical address is known:

- If mapping is known, physical address is placed in frame and sent (Destination is recognized)
- If mapping is not known, broadcast message is sent and awaits a reply
- Target machine, recognizing IP address matches its own, returns answer

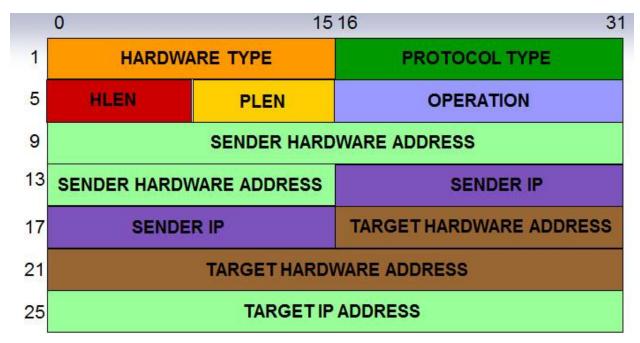


Figure 2.1: ARP Protocol Format

2.1:

The first column shows the IP addresses in the ARP cache. The second column shows the physical (MAC) addresses (48 bit Ethernet Address) and the last column shows the type of ARP entries:

ARP entries can be **Dynamic** or **Static**:

Dynamic:

Which means that the ARP entry (the Ethernet MAC to IP address link) has been learned (usually from the default gateway) and is kept on a device for some period of time, as long as it is being used.

Static:

A **static** ARP entry is the opposite of a dynamic ARP entry. With a static ARP entry, the computer is manually entering the link between the Ethernet MAC address and the IP address. Software in your computer will predefine these static entries such as multicast addresses and broadcast addresses.

```
Command Prompt
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\Asus>arp -a
Interface: 192.168.1.8 --- 0xd
                        Physical Address
 Internet Address
                                              Type
 192.168.1.1
                        c4-e9-0a-35-62-c4
                                              dynamic
 192.168.1.2
                        3c-dc-bc-98-04-e9
                                              dynamic
                        24-fd-52-8f-6f-51
 192.168.1.5
                                              dynamic
 192.168.1.255
                        ff-ff-ff-ff-ff
                                              static
 224.0.0.2
                        01-00-5e-00-00-02
                                              static
  224.0.0.22
                        01-00-5e-00-00-16
                                              static
  224.0.0.251
                        01-00-5e-00-00-fb
                                              static
                        01-00-5e-00-00-fc
  224.0.0.252
                                              static
                        01-00-5e-7f-ff-fa
 239.255.255.250
                                              static
  255.255.255.255
                        ff-ff-ff-ff-ff
                                              static
C:\Users\Asus>
```

Figure 2.2

2.2:

a,b)

The Hexadecimal Values of the Source and Destination are shown in Figure 2.4:

ARP is a protocol which belongs to the Datalink layer and it saves the Translations (Mappings) of IP address(Network Layer) (The Upper Layer) to MAC(Physical) address (Physical Layer) (The Layer below), So ARP corresponds to IP Protocols.

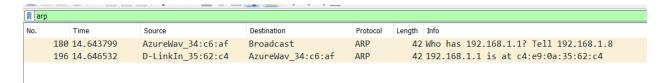


Figure 2.3

Figure 2.4

c)

The value of the opcode field within the ARP-Payload part of the Ethernet frame: 0001

```
Address Resolution Protocol (request)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: request (1)

Sender MAC address: AzureWav_34:c6:af (f0:03:8c:34:c6:af)

Sender IP address: 192.168.1.8

Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)

Target IP address: 192.168.1.1
```

Figure 2.5

d)

The Sender's IP address is shoen in the figure below: 192.168.1.8

```
> Frame 180: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{89CD229C-
> Ethernet II, Src: AzureWav_34:c6:af (f0:03:8c:34:c6:af), Dst: Broadcast (ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: AzureWav 34:c6:af (f0:03:8c:34:c6:af)
    Sender IP address: 192.168.1.8
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.1.1
0000 ff ff ff ff ff f0 03 8c 34 c6 af 08 06 00 01
                                                     0010 08 00 06 04 00 01 f0 03 8c 34 c6 af c0 a8 01 08
                                                     .........4...
0020 00 00 00 00 00 c0 a8 01 01
```

Figure 2.6

e)

In the field: Target MAC address (Figure 2.6): All digits are equal to zero because sender A doesn't know B (Destination's) MAC address;

So it broadcast its IP address for the whole networks, B gives response when he sees his IP and then it puts his MAC address in the response. After A (Sender) has received the MAC address it wll be saved in the ARP cache for next communications.

2.3:

Response of device B:

a)

The value of the opcode field: 2

```
Frame 196: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{89CD229C-CEE1-4068-A281-4BE85F267A7E}, id 0

Ethernet II, Src: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4), Dst: AzureWav_34:c6:af (f0:03:8c:34:c6:af)

Address Resolution Protocol (reply)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: reply (2)

Sender MAC address: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4)

Sender IP address: 192.168.1.1

Target MAC address: AzureWav_34:c6:af (f0:03:8c:34:c6:af)

Target IP address: 192.168.1.8
```

Figure 2.7

By running the "arp -a" command, we can see the list of arp entries has been extended after receiving the response message which contains the Destinations MAC address:

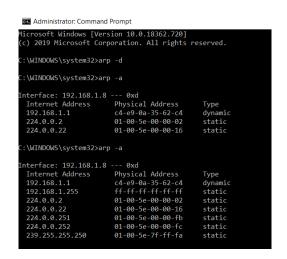


Figure 2.8

b,c)

The Answer appears in the field "Sender MAC address as its the destination's response to the message Sender has Broadcasted in the Network:

• Sender(Destination's) MAC address (the answer):

c4:e9:0a:35:62:c4

• Source(the "Target" in the response message) hexadecimal MAC address:

f0:03:8c:34:c6:af

• IP addresses for the Source(Target) in hexadecimal:

c0:a8:01:01

• IP addresses for the Destination(Sender) in hexadecimal:

c0:a8:01:08

```
> Frame 196: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\MPF_(89CD29C-CEE1-4068-A2B1-4BE85F267A7E), id 0

Ethernet II, Src: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4), Dst: AzureWay_34:c6:af (f9:03:8c:34:c6:af)

### Address Resolution Protocol (repel)

| Protocol tyne: IPv4 (0x0800)

| Hardware size: 6

| Protocol size: 4
| Opcode: reply (2)
| Sender PMC address: D-LinkIn_35:62:c4 (c4:e9:0a:35:62:c4)

| Sender IP address: 192.168.1.1
| Target IP address: 192.168.1.8

| Target IP address: 192.168.1.8

| Resolution | Protocol size: 4 (c4:e9:0a:35:62:c4)
| Target IP address: 192.168.1.8

| Resolution | Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 20:168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Sender PMC address: 192.168.1.8
| Protocol size: 4 (c4:e9:0a:35:62:c4)
| Protocol
```

Figure 2.9

Figure 2.10

Part 3

3.1:

The Timing Datagram is displayed in the Figure below:



Figure 3.1

3.2:

Option 53 indicates the values which the message type sets and are different in request and discover messages despite same Source and Destination IP addresses and other same specifications.

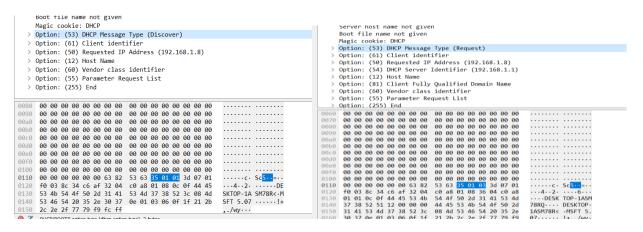


Figure 3.2

3.3:

The transaction ID (xid) is chosen by the client (usually randomly) and is copied by the server in the responses. It has a client specific function, which is usually to allow the client to recognize which "DHCP" response to which request has been associated.

First set:

We Can see that all of the Transaction ID's of the first set are the same . that's because all are associated to the same client request

Transaction-ID for "Discover": 0x 2cf52b98

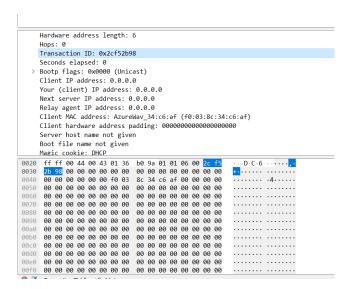


Figure 3.3

Transaction-ID for "Offer": 0x 2cf52b98

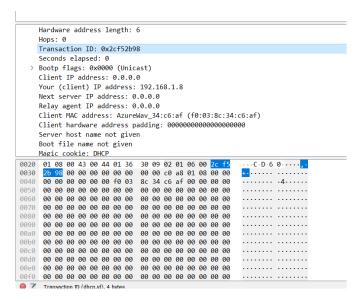


Figure 3.4

Transaction-ID for "Request": 0x 2cf52b98

```
Hardware address length: 6
   Hops: 0
   Transaction ID: 0x2cf52b98
   Seconds elapsed: 0
   Bootp flags: 0x0000 (Unicast)
   Client IP address: 0.0.0.0
   Your (client) IP address: 0.0.0.0
   Next server IP address: 0.0.0.0
   Relay agent IP address: 0.0.0.0
   Client MAC address: AzureWav 34:c6:af (f0:03:8c:34:c6:af)
   Client hardware address padding: 000000000000000000000
   Server host name not given
   Boot file name not given
   Magic cookie: DHCP
0020 ff ff 00 44 00 43 01 50
                                            -D-C-P Z----,-
                      5a 93 01 01 06 00 2c f
    00 00 00 00 00 f0 03
                      8c 34 c6 af 00 00 00 00
                                          .........4.....
0050
    0060
    00 00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
                                          . . . . . . . . . . . . . . . . . . .
0080 00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
0090
    00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
00b0
    00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
00c0
    00 00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
                                          00e0
```

Figure 3.5

Transaction-ID for "ACK": 0x 2cf52b98

```
Hardware address length: 6
   Hops: 0
   Transaction ID: 0x2cf52b98
   Seconds elapsed: 0
 > Bootp flags: 0x0000 (Unicast)
   Client IP address: 0.0.0.0
   Your (client) IP address: 192.168.1.8
   Next server IP address: 0.0.0.0
   Relay agent IP address: 0.0.0.0
   Client MAC address: AzureWav_34:c6:af (f0:03:8c:34:c6:af)
   Server host name not given
   Boot file name not given
   Magic cookie: DHCP
   01 08 00 43 00 44 01 50
                                      ...C.D.P ,....,.
                    2c d5 02 01 06 00 2c f5
    2b 98 00 00 00 00 00 00 00 00 c0 a8 01 08 00 00
                                       .........4.....
    00 00 00 00 00 00 f0 03 8c 34 c6 af 00 00 00 00
   ...... ......
0050
   00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00
0060
0070
    00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00
0090
    00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00
    ...... ......
00c0
    .....
    ...... ......
00f0
```

Figure 3.6

Second set:

We observe that all of the transaction ID's of the second set are the same too.but they differ from the TransactionID's of the first set and that's because they're all associated to a different request from the client.

Transaction-ID for "Discover": 0x 644cf48c

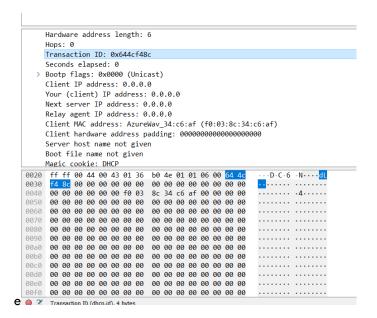


Figure 3.7

Transaction-ID for "Offer": 0x 644cf48c

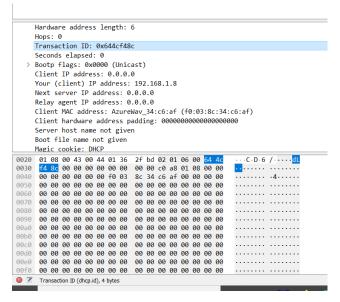


Figure 3.8

Transaction-ID for "Request": 0x 644cf48c

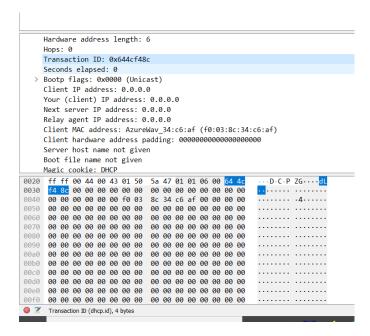


Figure 3.9

Transaction-ID for "ACK": 0x 644cf48c

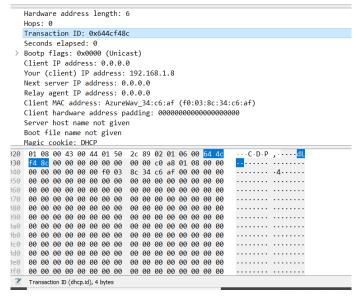


Figure 3.10

3.4:

According to the Figure below which was shown previously in Figure 3.1 as the "Timing Datagram" the Source and Destination IP adresses of the 4 DHCP messages of both sets are shown **on the Datagram** (The arrows show the message flow from Source to Destination)

Discover and Request:

Source IP: 0.0.0.0

Destination IP: 255.255.255.255 (means message is sent broadcast)

Offer and ACK:

Source IP: 192.168.1.1

Destination IP: 192.168.1.8

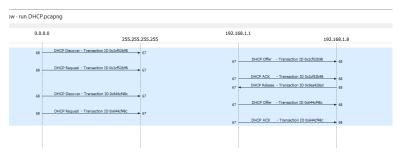


Figure 3.11

3.5:

The IP address of the DHCP server is shown in the offer message for the first time (As the request message is a broadcast message in order to find the DHCP server and request an IP from it):

)	nootp						
	Time	Source	Destination	Protocol	Length Info		
	241 46.709472	0.0.0.0	255.255.255.255	DHCP	344 DHCP Discover - Transaction ID 0x2cf52b98		
	242 46.782700	192.168.1.1	192.168.1.8	DHCP	344 DHCP Offer - Transaction ID 0x2cf52b98		
	243 46.784317	0.0.0.0	255.255.255.255	DHCP	370 DHCP Request - Transaction ID 0x2cf52b98		
	257 49.433928	192.168.1.1	192.168.1.8	DHCP	370 DHCP ACK - Transaction ID 0x2cf52b98		

Figure 3.12

DHCP IP address: 192.168.1.1

3.6:

The IP address that the DHCP server offers to the client(my computer which doesn't have IP address currently; my IP: 0.0.0.0) is shown in the "Offer" message:

Your(Client) IP address: 192.168.1.8 (Shown in the figure below)

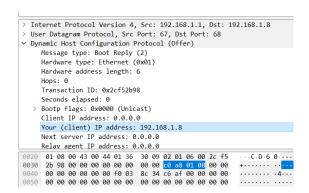


Figure 3.13

2.7:

The client accepts the offered IP address from the DHCP server:

This IP (192.168.1.8) is shown in the Option 50 of the "Request" message

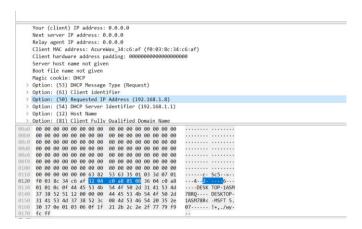


Figure 3.14

2.8:

In short, DHCP Lease Time is the amount of time in minutes or seconds a network device can use an IP Address in a network.

The IP Address is reserved for that device until the reservation expires. The DHCP server is responsible for assigning every device a unique address.

Time to lease in my experiment (in sec): 86400 sec (equals 24 hours (One Day))

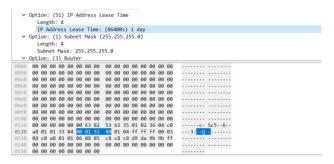


Figure 3.15

References:

- 1- https://www.cellstream.com/reference-reading
- 2- <u>https://wiki.wireshark.org/Hyper_Text_Transfer_Protocol</u>
- 3- https://www.quora.com/What-is-IP-lease-time-in-DHCP