Noah Barrall
Dr. Bibighaus

Wireshark Assignment

DHCP

In figure 1.1, this is an example of a DHCP resolution. What DHCP does is that it automatically assigns IP addresses and other essential network settings, making it easier to manage devices on a network. In figure 1.1, there are 5 parts to it, but the DHCP release is not part of the official DHCP process, effectively making the process four steps. The first part is the release of the DHCP lease, from the command “ipconfig /release”. This command forces the immediate release of the DHCP lease, marking the address as available. The DHCP discover packet happens when a device connects to a network, it will send out a broadcast message attempting to find a DHCP server. The DHCP offer packet is when the DHCP server responds back with an available IP address and other network settings. Fourth, is the DHCP request packet, which will be sent back to the DHCP server asking to use the IP address it was offered. Finally, is the DHCP acknowledgement, where the server will send an acknowledge message back to the client, confirming that this IP address is now assigned.

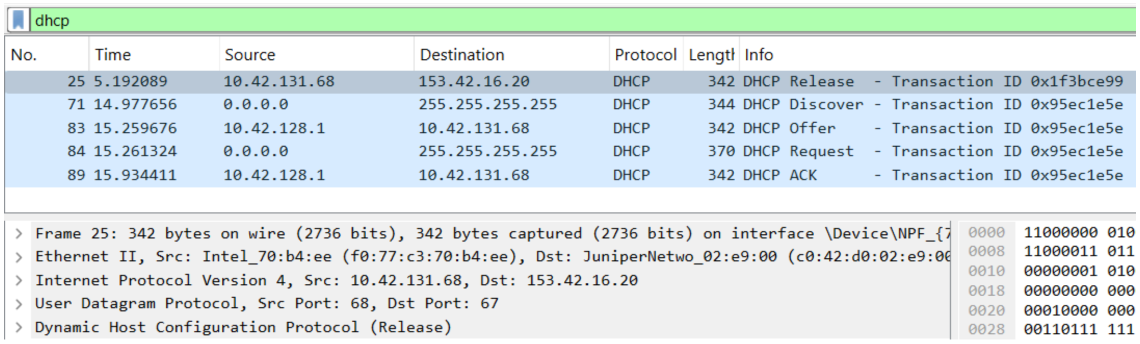
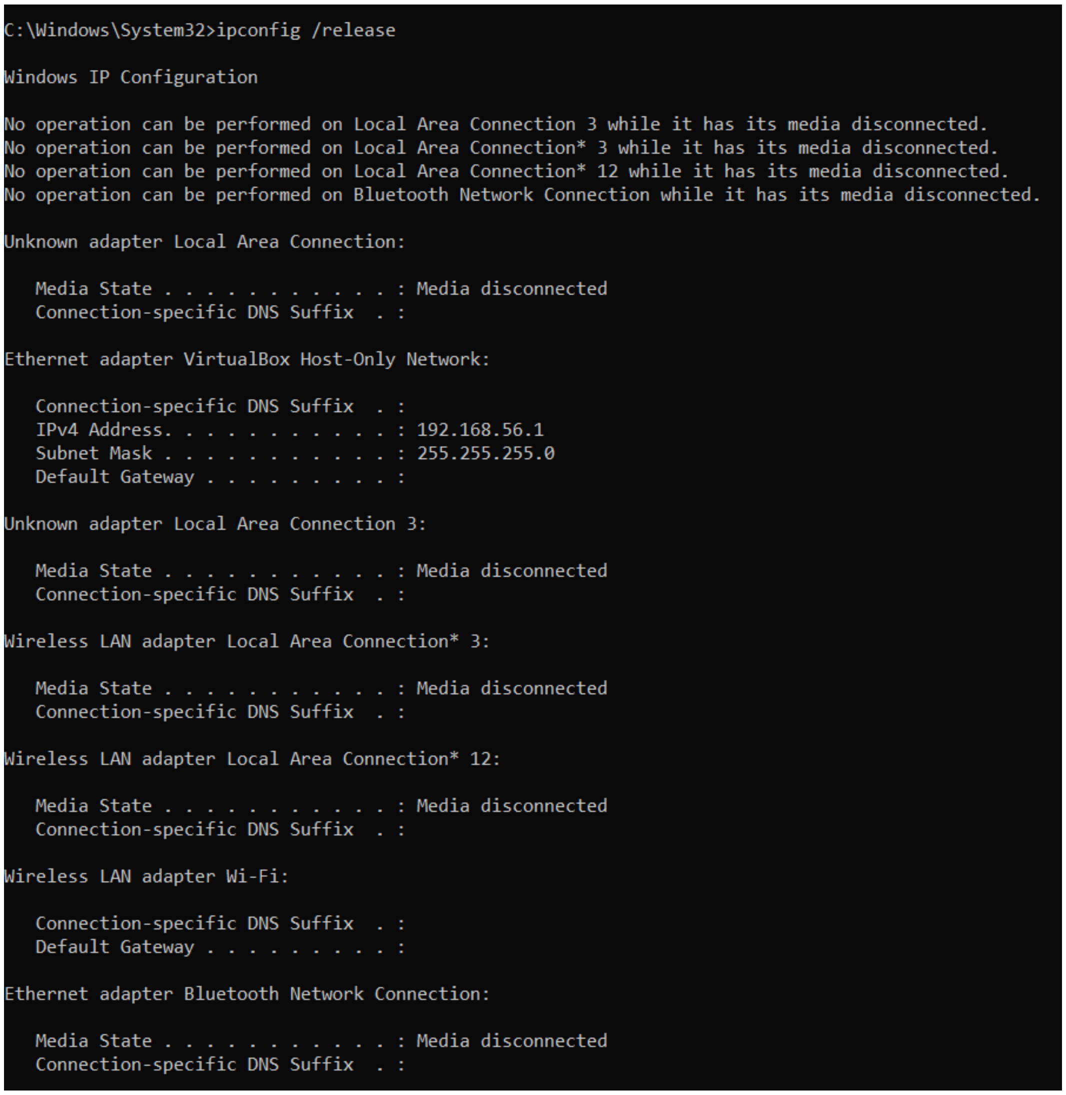

```
dhcp
No.   Time        Source          Destination       Protocol Lengtl Info
   25 5.192089    10.42.131.68    153.42.16.20      DHCP     342 DHCP Release  - Transaction ID 0x1f3bce99
   71 14.977656   0.0.0.0         255.255.255.255   DHCP     344 DHCP Discover - Transaction ID 0x95ec1e5e
   83 15.259676   10.42.128.1     10.42.131.68      DHCP     342 DHCP Offer    - Transaction ID 0x95ec1e5e
   84 15.261324   0.0.0.0         255.255.255.255   DHCP     370 DHCP Request  - Transaction ID 0x95ec1e5e
   89 15.934411   10.42.128.1     10.42.131.68      DHCP     342 DHCP ACK      - Transaction ID 0x95ec1e5e

> Frame 25: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface \Device\NPF_{7   0000  11000000 010
> Ethernet II, Src: Intel_70:b4:ee (f0:77:c3:70:b4:ee), Dst: JuniperNetwo_02:e9:00 (c0:42:d0:02:e9:00  0008  11000011 011
> Internet Protocol Version 4, Src: 10.42.131.68, Dst: 153.42.16.20                                    0010  00000001 010
> User Datagram Protocol, Src Port: 68, Dst Port: 67                                                   0018  00000000 000
> Dynamic Host Configuration Protocol (Release)                                                        0020  00010000 000
                                                                                                       0028  00110111 111
```

Figure 1.1

To perform a DHCP resolution to be able to capture the whole process, I chose to manually release and renew my DHCP lease. In order to do this, I used the command “ipconfig /release” to force a release of my DHCP lease, as shown in figure 1.2. This disconnected my wireless connection from my device, causing wireshark to stop capturing data until I used the command “ipconfig /renew”, as shown in figure 1.4, to force a renewal of a DHCP lease to an available IP address. This caused wireshark to begin capturing data again and that is what caused this DHCP process to happen.



```
C:\Windows\System32>ipconfig /release

Windows IP Configuration

No operation can be performed on Local Area Connection 3 while it has its media disconnected.
No operation can be performed on Local Area Connection* 3 while it has its media disconnected.
No operation can be performed on Local Area Connection* 12 while it has its media disconnected.
No operation can be performed on Bluetooth Network Connection while it has its media disconnected.

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Unknown adapter Local Area Connection 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 12:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Default Gateway . . . . . . . . . :

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Figure 1.2

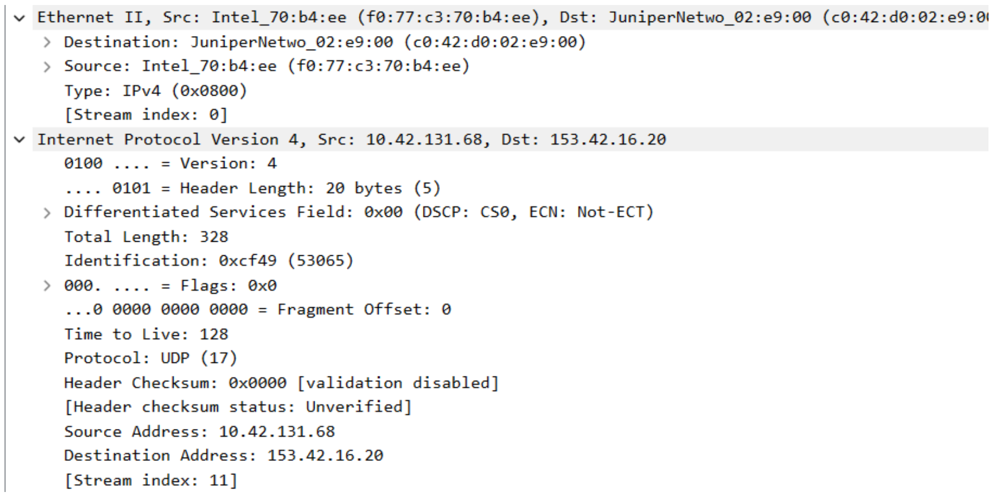
When considering what parts of the DHCP process is part of layer 2 and which is part of layer 3, there is a lot that goes into it. In general, the Ethernet II header mostly deals with Layer 2 (data link layer), and the IPv4 header deals with layer 3 (network layer).

Starting the DHCP process is the release. When the client no longer needs the IP, there is a layer 3 unicast packet sent from the client's IP to the DHCP server IP. This initiates the release of the IP address. Next in the process is the DHCP Discover, which operates at both layer 2 and layer 3. The discover packet is broadcast to all devices on the local network at layer 2 using the source and destination MAC addresses. It is also used on layer 3, by being sent as a broadcast packet using source IP address 0.0.0.0, as shown in figure 1.1, and the destination address as 255.255.255.255.

The next step in this process is the DHCP Offer, where the DHCP server proposes an available IP address and other network information to the client. The information includes the subnet mask, lease time, default gateway, DNS server addresses, and IP address of the DHCP server itself. As you can see in figure 1.1, the server has now assigned me my destination address as 10.42.131.68, and it is coming from the IP of the server I will now communicate with. This transaction operates on layer 3 since it is dealing with IP addresses, IP communication, and routing information.

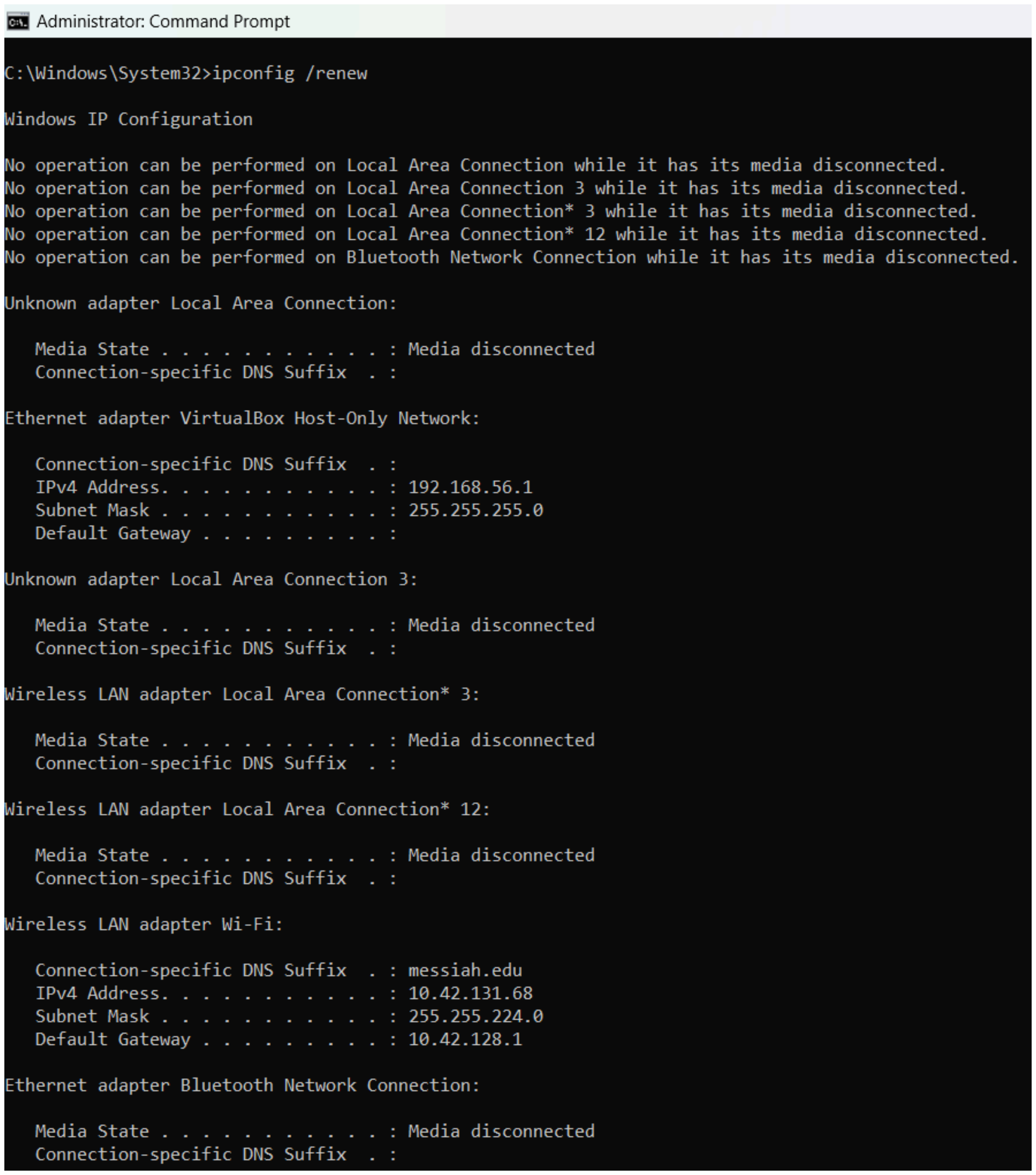
Fourthly, is the DHCP Request. The request operates on both layer 2 and layer 3. For layer 2, the DHCP Request packet will be broadcasted at Layer 2 using the source MAC address. This particular message is encapsulated in an ethernet frame for transmission over the network, and contains both the source and destination MAC addresses. Regarding layer 3, the actual DHCP request message is part of a UDP/IP packet. The message itself is a broadcast message sent with a destination IP of 255.255.255.255, as shown in figure 1.1.

The final step to the DHCP process is DHCP Acknowledgement (ACK). The ACK response confirms the lease of the IP address and any other network settings that were requested by the client. The client can now use the newly assigned IP address. As shown in figure 1.1, the source is the server's IP address, and the destination is now the newly assigned IP address to the client. DHCP ACK operates at both layer 2 and layer 3, but primarily functions at layer 3. At layer 2, the ACK request uses both the source and destination MAC addresses, to allow these packets to be transmitted on the local network. The core functions of layer 3 include confirming the assignment of IP addresses and DHCP server sending the ACK using IP addresses. As well as some of the background settings such as configurations for the subnet mask, default gateway, and DNS servers, and routing if the client and servers are on different subnets.



```
v Ethernet II, Src: Intel_70:b4:ee (f0:77:c3:70:b4:ee), Dst: JuniperNetwo_02:e9:00 (c0:42:d0:02:e9:0
  > Destination: JuniperNetwo_02:e9:00 (c0:42:d0:02:e9:00)
  > Source: Intel_70:b4:ee (f0:77:c3:70:b4:ee)
    Type: IPv4 (0x0800)
    [Stream index: 0]
v Internet Protocol Version 4, Src: 10.42.131.68, Dst: 153.42.16.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 328
    Identification: 0xcf49 (53065)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.42.131.68
    Destination Address: 153.42.16.20
    [Stream index: 11]
```

Figure 1.3



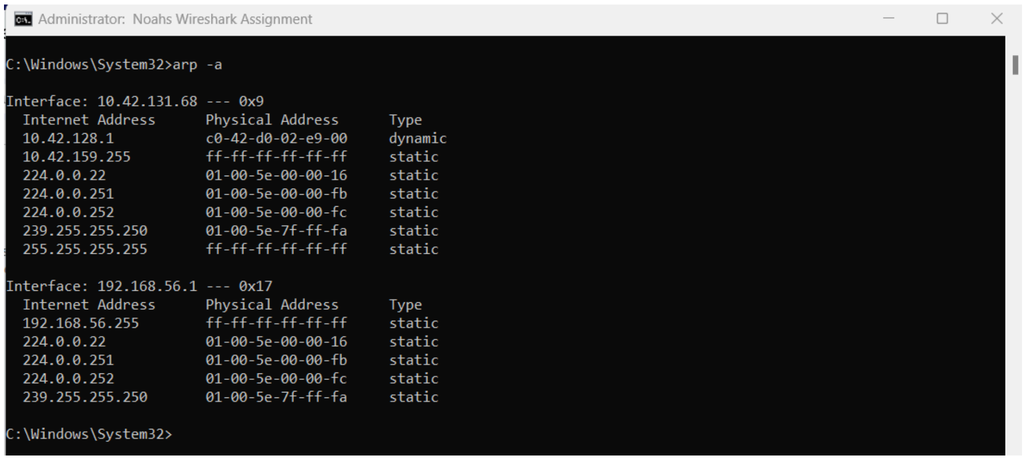
Administrator: Command Prompt

```
C:\Windows\System32>ipconfig /renew

Windows IP Configuration

No operation can be performed on Local Area Connection while it has its media disconnected.
No operation can be performed on Local Area Connection 3 while it has its media disconnected.
No operation can be performed on Local Area Connection* 3 while it has its media disconnected.
No operation can be performed on Local Area Connection* 12 while it has its media disconnected.
No operation can be performed on Bluetooth Network Connection while it has its media disconnected.

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Unknown adapter Local Area Connection 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 12:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : messiah.edu
   IPv4 Address. . . . . . . . . . . : 10.42.131.68
   Subnet Mask . . . . . . . . . . . : 255.255.224.0
   Default Gateway . . . . . . . . . : 10.42.128.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Figure 1.4

ARP

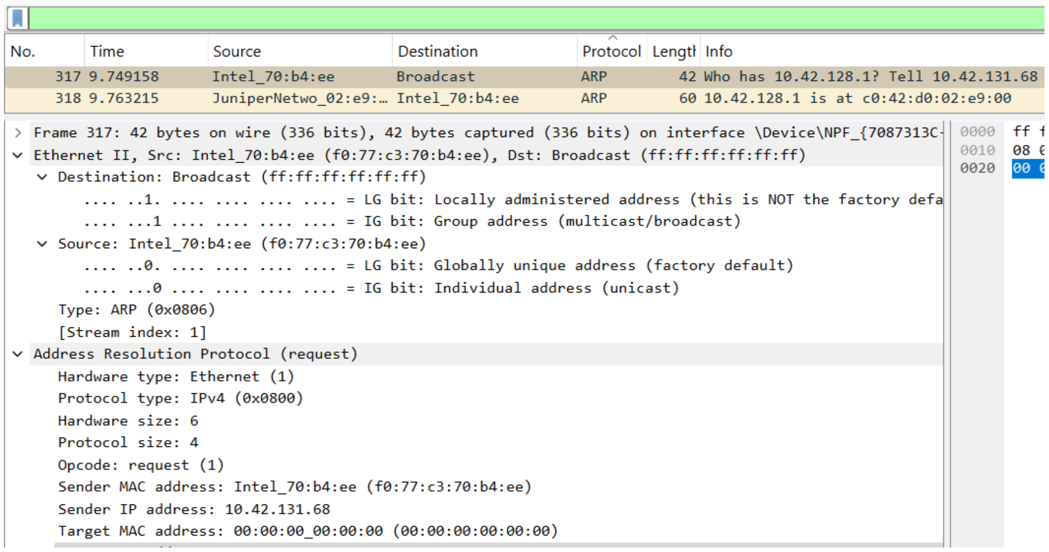
In this part of the exercise, the goal was to find our computer's arp cache. In order to do this, you need to enter the command arp -a, as shown in Figure 2.1. The arp cache is important because it allows for devices to quickly find each other without having to repeatedly ask for each other's MAC addresses.



```
Administrator: Noahs Wireshark Assignment

C:\Windows\System32>arp -a

Interface: 10.42.131.68 --- 0x9
  Internet Address      Physical Address      Type
  10.42.128.1           c0-42-d0-02-e9-00     dynamic
  10.42.159.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x17
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

C:\Windows\System32>
```

Figure 2.1

The next section, figure 2.2, is a capture of an ARP transaction. In order to capture this transaction, I chose to disconnect from the WiFi and then reconnect causing the device to generate ARP traffic while the device needs to reestablish a connection and find local devices.



```
No.     Time        Source                 Destination      Protocol  Lengtl Info
    317 9.749158    Intel_70:b4:ee         Broadcast        ARP          42 Who has 10.42.128.1? Tell 10.42.131.68
    318 9.763215    JuniperNetwo_02:e9:…   Intel_70:b4:ee   ARP          60 10.42.128.1 is at c0:42:d0:02:e9:00

> Frame 317: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{7087313C
v Ethernet II, Src: Intel_70:b4:ee (f0:77:c3:70:b4:ee), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  v Destination: Broadcast (ff:ff:ff:ff:ff:ff)
       .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory defa
       .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
  v Source: Intel_70:b4:ee (f0:77:c3:70:b4:ee)
       .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
       .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: ARP (0x0806)
    [Stream index: 1]
v Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Intel_70:b4:ee (f0:77:c3:70:b4:ee)
    Sender IP address: 10.42.131.68
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)

0000  ff f
0010  08 0
0020  00 0
```

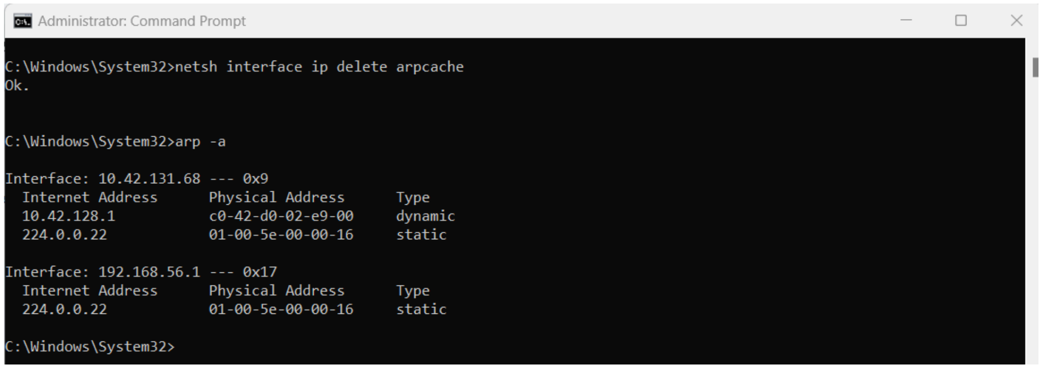
Figure 2.2

While analyzing the packets, I chose to analyze packet numbers 317 and 318, because these are the two key packets, the ARP request and the ARP reply. The first part of the packet contains the ethernet II header or the data link layer information (layer 2). The information contained in this part of the packet is the destination MAC Address indicating which address the packet is directed to, source MAC Address indicating which device this packet is coming from, and the EtherType. For the EtherType in ARP packets, it is typically set to (0x0806) indicating that the payload is an ARP transaction.

When expanding the Destination and Source fields, they contain additional information. For the destination address, you have the LG (Local/Global) bit and IG (Individual/Group) bits, both of which are set to 1. This indicates that for the LG bit, this is a locally administered address, and for the IG bit, it means that this is a group address, meaning this frame is meant for multiple devices on a network. This makes sense for the destination address, considering it is attempting to reach the default gateway for my device. The source address has both bits set to 0. For the LG bit, it means that this is a globally unique address, and for the IG bit, this means this is an individual or unicast address.

The second part of the packet is the address resolution protocol, dealing with both layer 2 and layer 3. There are two fields in this part of the frame that are involved in layer 2 communication. They are hardware type, and in this case mine is set to 1, meaning that we are using ethernet. As well as hardware size, which indicates the length of the hardware address, or MAC address in bytes, which would be 6. The layer 3 part of this includes protocol type, IPv4, the protocol size which indicates the length of the protocol in bytes, 4. As well as the better known areas, sender MAC and IP addresses and target MAC and IP addresses.

Figure 2.3 shows my personal devices arp cache after clearing it using the command “netsh interface ip delete arpcache”.

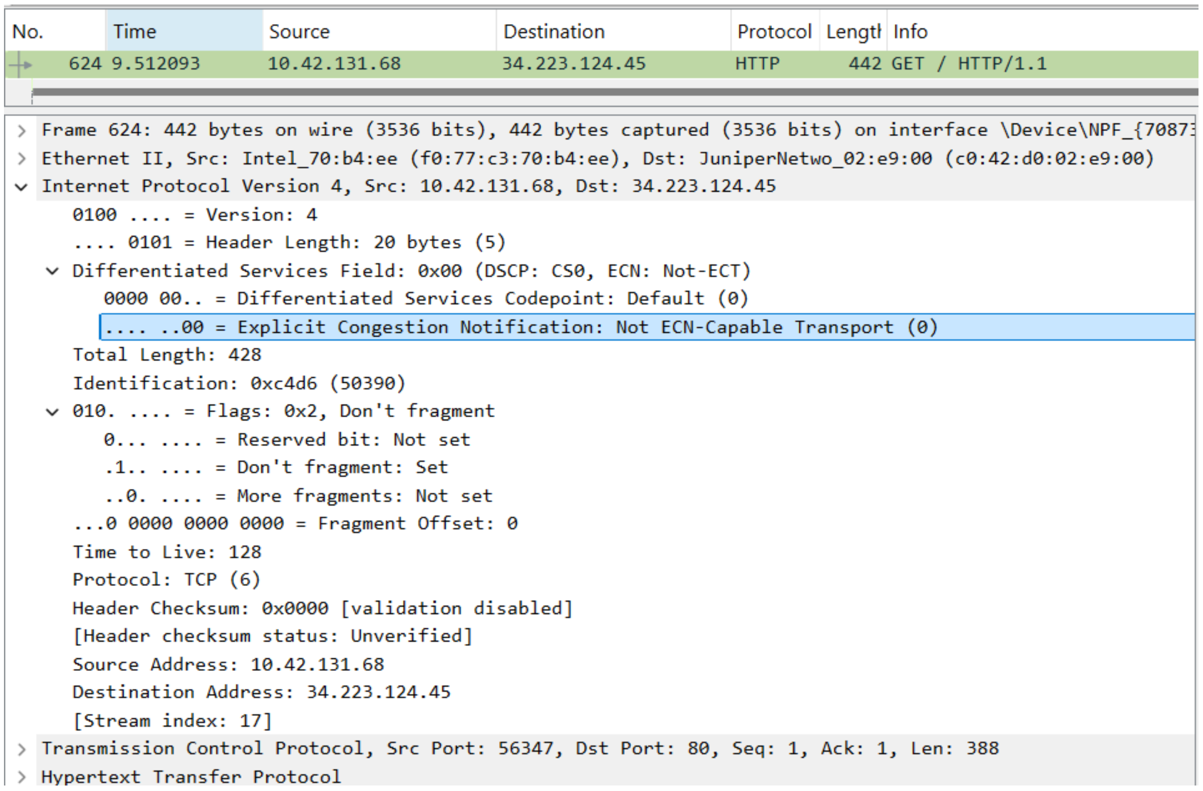


```
C:\Windows\System32>netsh interface ip delete arpcache
Ok.


C:\Windows\System32>arp -a

Interface: 10.42.131.68 --- 0x9
  Internet Address      Physical Address      Type
  10.42.128.1           c0-42-d0-02-e9-00     dynamic
  224.0.0.22            01-00-5e-00-00-16     static

Interface: 192.168.56.1 --- 0x17
  Internet Address      Physical Address      Type
  224.0.0.22            01-00-5e-00-00-16     static

C:\Windows\System32>
```

Figure 2.3

Web Surfing

In figure 3.1 I searched the site http://neverssl.com, to be able to get an HTTP protocol show up in wireshark. I was able to capture this GET packet.



```
No.     Time        Source          Destination     Protocol Length Info
   624  9.512093    10.42.131.68    34.223.124.45   HTTP       442  GET / HTTP/1.1

> Frame 624: 442 bytes on wire (3536 bits), 442 bytes captured (3536 bits) on interface \Device\NPF_{7087
> Ethernet II, Src: Intel_70:b4:ee (f0:77:c3:70:b4:ee), Dst: JuniperNetwo_02:e9:00 (c0:42:d0:02:e9:00)
v Internet Protocol Version 4, Src: 10.42.131.68, Dst: 34.223.124.45
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  v Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 428
    Identification: 0xc4d6 (50390)
  v 010. .... = Flags: 0x2, Don't fragment
      0... .... = Reserved bit: Not set
      .1.. .... = Don't fragment: Set
      ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.42.131.68
    Destination Address: 34.223.124.45
    [Stream index: 17]
> Transmission Control Protocol, Src Port: 56347, Dst Port: 80, Seq: 1, Ack: 1, Len: 388
> Hypertext Transfer Protocol
```

Figure 3.1

In figure 3.1, I am analyzing an HTTP GET request. An HTTP GET request is a method used by a client (usually a web browser) to request data from a server. In the context of the HTTP (Hypertext Transfer Protocol), the GET method is one of the most commonly used request types, and it is designed to retrieve resources, such as HTML pages, images, or other types of data.

Each bit in the IP header tells us a chunk of information about the packet, going from the start of bits to the end:

- 4 bits, 0100, represents the version of IP, showing us the version is IPv4
- 4 bits, 0101, represents the header length, showing us the length is 5 bit or 20 bytes
- 8 bits, 00000000, represents the Differentiated Services Field
    - 6 bits, 000000, represents the differentiated services codepoint. This specifies the different levels of service. In my case, all 0s means “Best Effort” or the default level of service

    - 2 bits, 00, represents the Explicit Congestion Notification (ECN). This reduces the likelihood of packet loss. 00 means that ECN is not being used.
- 16 bits, 00000001 10101100, represents the total length of the IP packet header and data in bytes. These 16 bits translate to 428 in decimal form.
- 16 bits, 11000100 11010110, represents the unique identification for each packet fragmentation. In this packet, the binary translates to 50390 in decimal.
- 3 bits, 010, represents the flags. The way the flags are set now means that the packets can’t be fragmented, and the fragment is either complete or final.
- 13 bits, 0 0000 00000000, represents the fragment offset. This indicates the position of the packet in the original packet. All 0s means either the packet is not fragmented or it is the first fragment of the packet.
- 8 bits, 10000000, represents the time to live. This specifies the packet's lifespan and is decremented 1 by each hop. The value is 128.
- 8 bits, 00000110, represents the Protocol. This specifies the protocol used in the portion of data, and for our packet is 6, which is TCP.
- 16 bits, 00000000 00000000, represents the header checksum. When all 16 bits are 0, this indicates that the header checksum is invalid.
- 32 bits, 00001010 00101010 10000011 01000100, represents the source IP address of the packet.
- 32 bits, 00100010 11011111 01111100 00101101, represents the destination IP address of the packet.