Network Packet Analysis Using Wireshark **Name**: Spencer Stein

1. DHCP Transaction Analysis

The DHCP process consists of four stages: **Discover**, **Offer**, **Request**, and **Acknowledge**, commonly known as **DORA**.

First, my device broadcasts a **Discover** message to locate DHCP servers. The server then responds with a DHCP **Offer**, proposing an available IP address and configuration settings. However, since I started and stopped Wireshark multiple times while capturing DHCP traffic, my device already had an IP lease, so I was unable to capture the **Discover** and **Offer** stages.

The client then replies with a **Request**, selecting one of the offers and formally asking to lease the IP address. Finally, the server sends an **Acknowledgement (ACK)** to confirm the lease and provide the client with its network configuration, completing the process.

Here are the two packets that are able to be seen:

No.	Time	Source	Destination	Protocol	Length	Info
558	96.362263	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0xc0b2744a
562	96.386441	10.42.128.1	10.42.157.111	DHCP	342	DHCP ACK - Transaction ID 0xc0b2744a

DHCP Request:

```
Dynamic Host Configuration Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xc0b2744a
    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: 1e:9e:e3:d3:2b:73 (1e:9e:e3:d3:2b:73)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Request)
  > Option: (55) Parameter Request List
  > Option: (57) Maximum DHCP Message Size
  > Option: (61) Client identifier
  > Option: (50) Requested IP Address (10.42.157.111)
  > Option: (51) IP Address Lease Time
  > Option: (255) End
    Padding: 00000000000000000000000000000000
```

The DHCP Request packet is sent by the client (my IP) to confirm or renew its IP address with the DHCP server. If the client is accepting a new IP offer, the Request packet finalizes that choice. If it is renewed, the client asks to extend the lease on its current IP. The

packet contains the client's MAC address and the requested IP, making sure the client has the correct network settings.

On my machine, the client is requesting a new IP address from the DHCP server. It sends a mac address and a requested IP address. It also requests maximum DHCP message size, client identifier, and the IP address lease time. It needs these parts to confirm or renew the IP address.

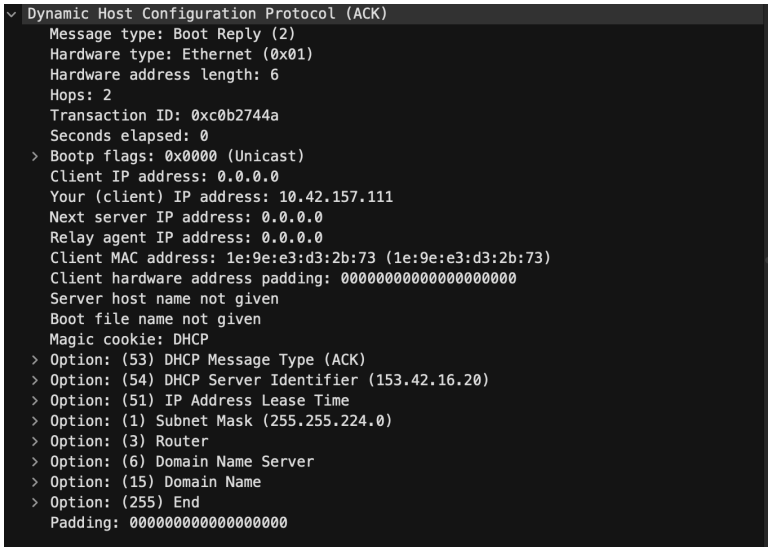
Layer 2:

- **Source MAC Address**: The MAC address of your computer.
- **Destination MAC Address**: The MAC address of the DHCP server or a broadcast address

Layer 3:

- **Source IP Address**: Typically 0.0.0.0 for new requests.
- **Destination IP Address**: Broadcast address (255.255.255.255) to reach the DHCP server.
- **Requested IP Address**: The IP the client wishes to obtain or renew.
- **Other:** Includes maximum message size, client identifier, and lease time.

2. DHCP ACK



The **DHCP ACK** packet is the server's response to the Request, confirming the IP assignment or lease renewal. It contains the final network configuration, including the IP address and lease duration. This packet makes sure the client has network access and allows continued use of the assigned IP for a period of time.

The server's IP address is 153.42.16.20 and the subnet mask is 255.255.224.0

Layer 2:

- **Source MAC Address**: The MAC address of the DHCP server
- **Destination MAC Address**: My MAC address

Layer 3:

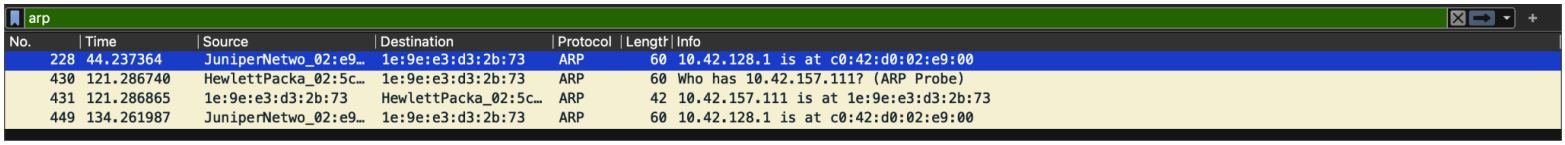
- **Source IP Address**: The IP address of the DHCP server (**153.42.16.20**)
- **Destination IP Address**: My IP address
- **Assigned IP Address**: The IP assigned to the client.
- **Subnet Mask**: Defines the network segment (**255.255.224.0**).
- **Lease Duration**: The time the assigned IP is valid.

2. ARP Analysis

ARP Cache - The ARP cache is a table that stores the mapping between IP addresses (Layer 3) and their corresponding MAC addresses (Layer 2)

```
? (10.42.128.1) at c0:42:d0:2:e9:0 on en0 ifscope [ethernet]
? (10.42.159.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
mdns.mcast.net (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet
]
```

ARP Transaction - a protocol used to map or translate an IP address to a physical machine (MAC) address in a local network



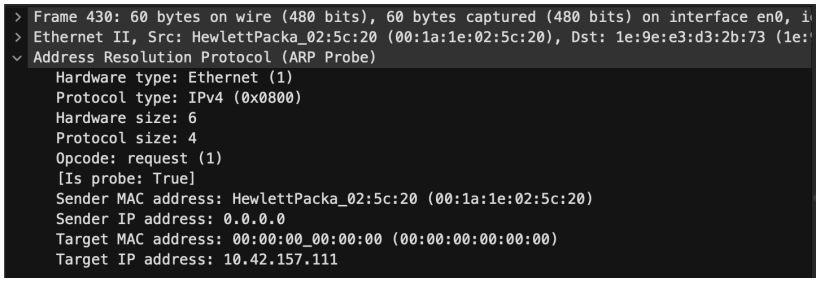
ARP Request - Tells the network to find the MAC address for a specific IP address, using a broadcast to ensure all devices hear the request.

Layer 2:

- **Source MAC Address**: Your computer's MAC address.
- **Destination MAC Address**: Set to **FF:FF:FF:FF:FF** (broadcast).

Layer 3:

- **Source IP Address**: Your computer’s IP address.
- **Destination IP Address**: The IP address being queried.



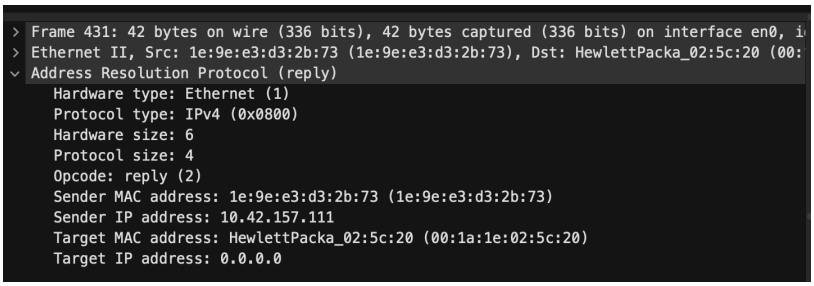
ARP Reply - Responds with the MAC address for the requested IP address, sent directly to the requesting device, enabling it to establish communication over the network.

Layer 2:

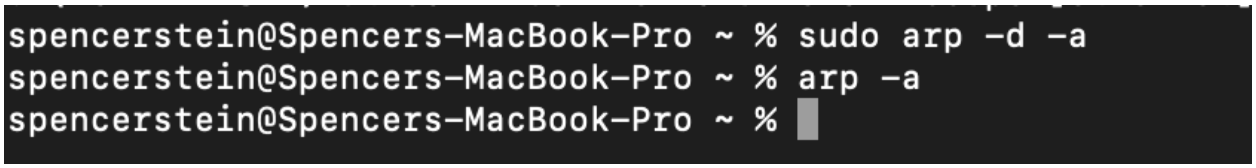
- **Source MAC Address**: The MAC address of the responding device.
- **Destination MAC Address**: The MAC address of your computer.

Layer 3:

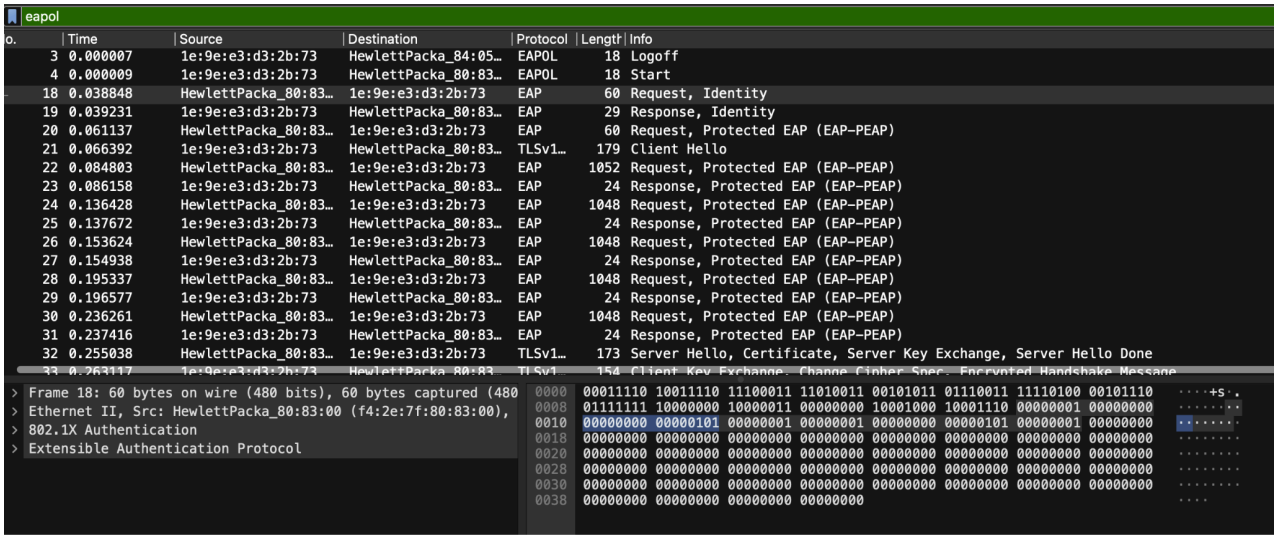
- **Source IP Address**: The IP address of the responding device.
- **Destination IP Address**: Your computer's IP address.



New ARP Cache - once deleted, there was nothing in it. After pinging Luke Wilson, I saw his IP (10.42.158.191). Now the ARP cache saves the MAC address associated with the IP address.



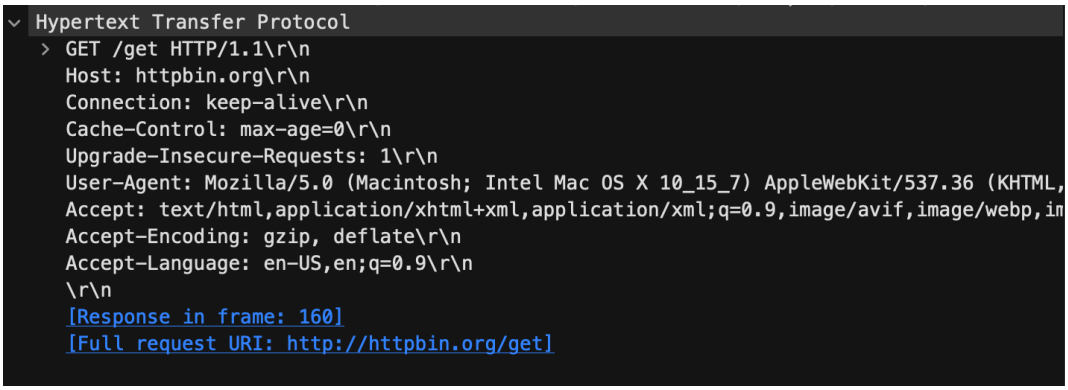
3. Wi-Fi Joining Analysis

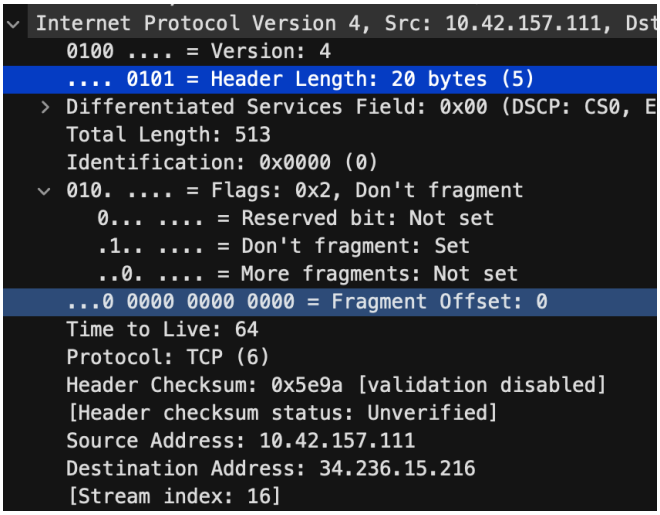


Request Bit Information:

1. **Destination MAC Address**: `f4:2e:7f:80:83:00` → Binary: `11110100 00101110 01111111 10000000 10000011 00000000`
   - This is the MAC address of the network device (e.g., access point).
2. **Source MAC Address**: `1e:9e:e3:d3:2b:73` → Binary: `00011110 10011110 11100011 11010011 00101011 01110011`
   - This is my computer
3. **EtherType**: `88 8e` → Binary: `10001000 10001110`
   - This indicates the **EtherType** is **EAPOL**, as the value `88 8e` corresponds to EAP over LAN (EAPOL).
4. **EAPOL Version**: `01` → Binary: `00000001`
   - EAPOL version 1 (WPA and WPA2 typically use version 1).
5. **EAPOL Type**: `00` → Binary: `00000000`
   - Type 0 indicates this is an **EAP packet**.
6. **EAP Length**: `00 05` → Binary: `00000000 00000101`
   - The length of the EAP message is 5 bytes.
7. **EAP Code**: `01` → Binary: `00000001`
   - EAP code 1 represents a **Request**.
8. **EAP Identifier**: `00` → Binary: `00000000`
   - The identifier of this EAP packet, used to match responses with requests.
9. **EAP Length**: `00 03` → Binary: `00000000 00000011`
   - The length of the EAP message is 3 bytes.
10. **EAP Type**: `01` → Binary: `00000001`
    - Type 1 represents a **Request for Identity**

4. Web Server





Field	Value	Description	Hex to Binary
Version	Version 4	Specifies the IP version	`4 = 0100`
Header Length	20 Bytes (5)	The length of the header in 32-bit words	`5 = 0101`
Type of Service	0x00	Specifies the priority of the packet	`00 = 00000000`
Total Length	513	The total length of the IP packet (header plus the data)	`0201= 0000001000000001`

Identification	0x0000	An identifier for the packet	`00 00 = 0000000000000000`
Flags	0x2	Control flags which determines which packet can be fragmented	`40 = 01000000`
Fragment Offset	0	The offset of this fragment in the packer	`00 = 00000000`
Time to Live (TTL)	64	The maximum time the pack can exist before being trashed	`40 = 01000000`
Protocol	TCP (6)	Indicated the transport later protocol used (TCP = 6)	`06 = 00000110`
Header Checksum	0x5e9a	Error checking for the header	`5cb4 = 0101110010110100`
Source Ip Address	10.42.157.111	The IP address of the sender (my computer)	`0a2a9d6f = 00001010001010101001110101101111`
Destination IP Address	34.236.15.216	The IP address of the receiver (the web server)	`5db8d70e = 01011101101110001101011100001110`