

MSCS631_WireShark_4

Wireshark – Lab 4: WIFI

Samrat Baral

University of the Cumberlands

2025 Spring – Advanced Computer Networks (MSCS-631-M40) – Full Term

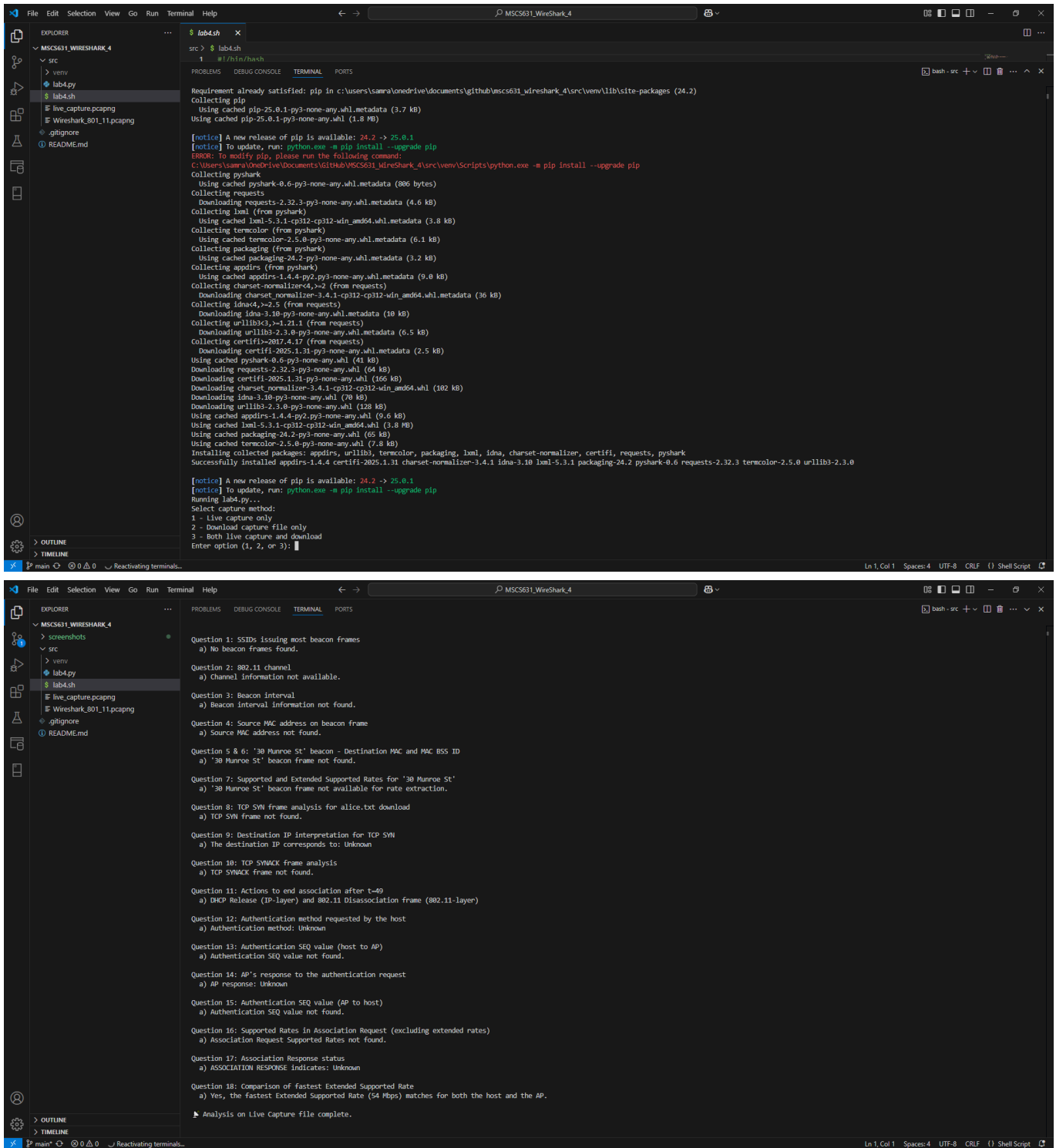
Dr. Yousef Nijim

March 2, 2025

Lab Overview

In this lab, you will investigate the 802.11 wireless network protocol. You'll capture a trace from a wireless 802.11 Wi-Fi interface on your computer/laptop. See the attached document for full lab instructions. The trace includes beacon frames from multiple access points (APs), TCP data from HTTP sessions, and the association/disassociation process for the wireless host. The lab instructions (and accompanying PDF) describe the process of filtering frames in Wireshark and then answering 18 questions regarding beacon information, MAC addressing, data transfer, and association procedures.

Output Screenshots



```
$ lab4.sh
src > $ lab4.sh
1 #! /bin/sh

Requirement already satisfied: pip in c:\users\samra\onedrive\documents\github\msc631_wireshark_4\src\venv\lib\site-packages (24.2)
Collecting pip
  Using cached pip-25.0.1-py3-none-any.whl.metadata (3.7 kB)
  Using cached pip-25.0.1-py3-none-any.whl (1.8 MB)
[notice] A new release of pip is available: 24.2 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
ERROR: To modify pip, please run the following command:
C:\Users\samra\OneDrive\Documents\GitHub\MSC631_Wireshark_4\src\venv\Scripts\python.exe -m pip install --upgrade pip
Collecting pyshark
  Using cached pyshark-0.6-py3-none-any.whl.metadata (886 bytes)
Collecting requests
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting lxml (from pyshark)
  Using cached lxml-5.3.1-cp312-cp312-win_amd64.whl.metadata (3.8 kB)
Collecting termcolor (from pyshark)
  Using cached termcolor-2.5.0-py3-none-any.whl.metadata (6.1 kB)
Collecting packaging (from pyshark)
  Using cached packaging-24.2-py3-none-any.whl.metadata (3.2 kB)
Collecting appdirs (from pyshark)
  Using cached appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting charset-normalizer<=2 (from requests)
  Downloading charset-normalizer-3.4.1-cp312-cp312-win_amd64.whl.metadata (36 kB)
Collecting idna<=2.5 (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.3.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>2017.4.17 (from requests)
  Downloading certifi-2025.1.31-py3-none-any.whl.metadata (2.5 kB)
Using cached pyshark-0.6-py3-none-any.whl (41 kB)
Download requests-2.32.3-py3-none-any.whl (64 kB)
Download certifi-2025.1.31-py3-none-any.whl (166 kB)
Download charset-normalizer-3.4.1-cp312-cp312-win_amd64.whl (102 kB)
Download idna-3.10-py3-none-any.whl (70 kB)
Download urllib3-2.3.0-py3-none-any.whl (128 kB)
Using cached appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Using cached lxml-5.3.1-cp312-cp312-win_amd64.whl (3.8 MB)
Using cached packaging-24.2-py3-none-any.whl (66 kB)
Using cached termcolor-2.5.0-py3-none-any.whl (7.8 kB)
Installing collected packages: appdirs, urllib3, termcolor, packaging, lxml, idna, charset-normalizer, certifi, requests, pyshark
Successfully installed appdirs-1.4.4 certifi-2025.1.31 charset-normalizer-3.4.1 idna-3.10 lxml-5.3.1 packaging-24.2 pyshark-0.6 requests-2.32.3 termcolor-2.5.0 urllib3-2.3.0

[notice] A new release of pip is available: 24.2 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Running lab4.py...
Select capture method:
1 - Live capture only
2 - Download capture file only
3 - Both live capture and download
Enter option (1, 2, or 3): 1

Question 1: SSIDs issuing most beacon frames
a) No beacon frames found.

Question 2: 802.11 channel
a) Channel information not available.

Question 3: Beacon Interval
a) Beacon Interval information not found.

Question 4: Source MAC address on beacon frame
a) Source MAC address not found.

Question 5 & 6: '30 Munroe St' beacon - Destination MAC and MAC BSS ID
a) '30 Munroe St' beacon frame not found.

Question 7: Supported and Extended Supported Rates for '30 Munroe St'
a) '30 Munroe St' beacon frame not available for rate extraction.

Question 8: TCP SYN frame analysis for alice.txt download
a) TCP SYN frame not found.

Question 9: Destination IP interpretation for TCP SYN
a) The destination IP corresponds to: Unknown

Question 10: TCP SYNACK frame analysis
a) TCP SYNACK frame not found.

Question 11: Actions to end association after t=49
a) DHCP Release (IP-layer) and 802.11 Disassociation frame (802.11-layer)

Question 12: Authentication method requested by the host
a) Authentication method: Unknown

Question 13: Authentication SEQ value (host to AP)
a) Authentication SEQ value not found.

Question 14: AP's response to the authentication request
a) AP response: Unknown

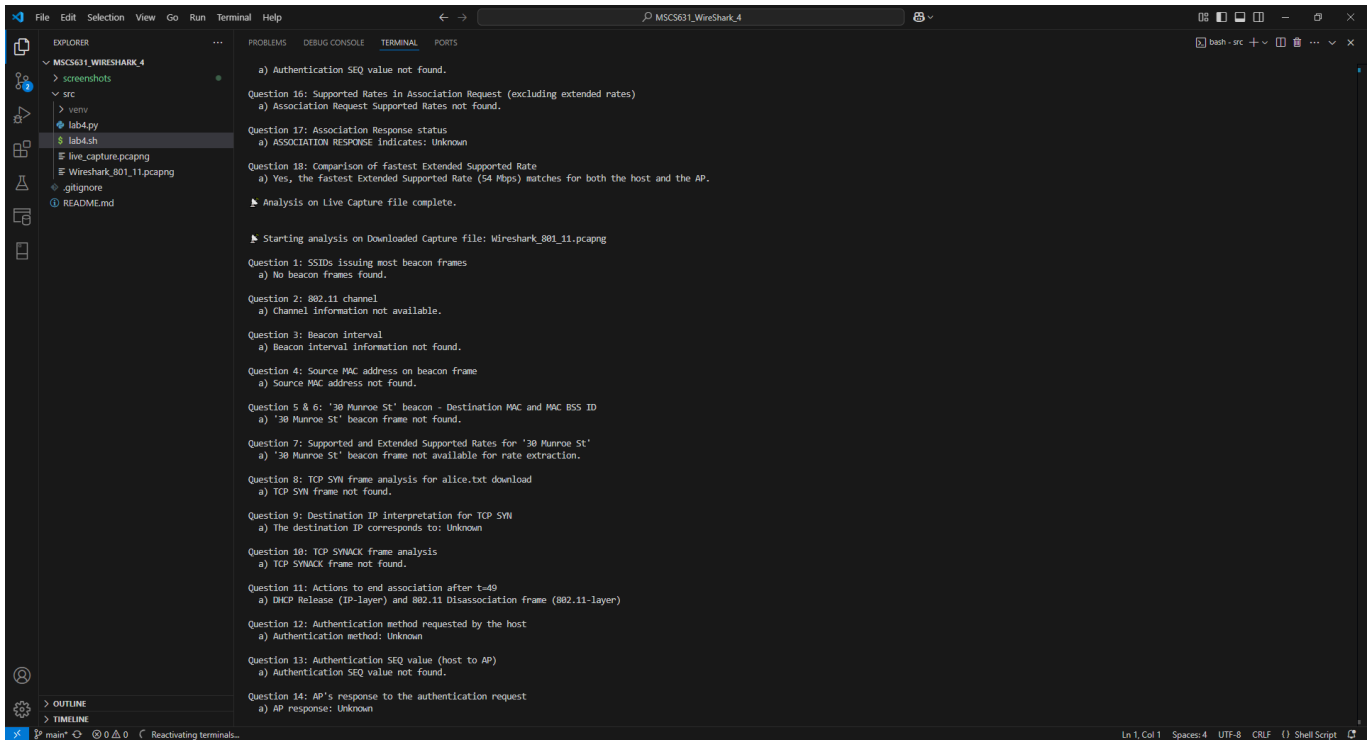
Question 15: Authentication SEQ value (AP to host)
a) Authentication SEQ value not found.

Question 16: Supported Rates in Association Request (excluding extended rates)
a) Association Request Supported Rates not found.

Question 17: Association Response status
a) ASSOCIATION RESPONSE indicates: Unknown

Question 18: Comparison of Fastest Extended Supported Rate
a) Yes, the Fastest Extended Supported Rate (54 Mbps) matches for both the host and the AP.

Analysis on Live Capture file complete.
```



Features

Prerequisites

- **Python 3.x**
- **Tshark:** Ensure that Tshark is installed and available in your system's PATH.
Download from [Wireshark](#).
- **Pyshark:** Install via pip:

```
pip install pyshark
python3 lab4.py
```

This lab uses a provided trace file to automatically answer questions about 802.11 beacon frames and a TCP SYN segment. We use the trace file from the Wireshark Labs available at:

<http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip>

After downloading and extracting the zip file, ensure that **Wireshark_801_11.pcapng** is in the same directory as **lab4.py** (or update the path accordingly).

Problem

- Identify the two most frequent SSIDs issuing beacon frames.
- Determine the channel used by the APs.
- Extract the beacon interval.
- Get the source MAC address from a beacon frame.
- Retrieve the destination MAC address from the "30 Munroe St" beacon.
- Retrieve the BSS ID from the "30 Munroe St" beacon.
- Extract both Supported Rates and Extended Supported Rates from the "30 Munroe St" beacon.

- Analyze the TCP SYN frame for the HTTP request (alice.txt) and extract MAC and IP fields.
- Determine if the destination IP corresponds to the destination web server. - Analyze the TCP SYNACK frame and extract MAC addresses. - Identify the actions taken to end association. - Determine the authentication method used. - Extract the Authentication SEQ value from host to AP. - Determine if the AP accepted the authentication. - Extract the Authentication SEQ value from AP to host. - Extract the Supported Rates from the Association Request (excluding extended rates). - Determine if the Association Response indicates success. - Verify if the fastest Extended Supported Rate offered by the host matches that of the AP.

Running the Analysis

A shell script ([lab4.sh](#)) is provided to create a virtual environment, install required packages (pyshark and requests), run the analysis script ([lab4.py](#)), and deactivate the environment.

To Run:

1. Place **lab4.py**, **lab.md**, and **lab4.sh** in your project directory.
2. Ensure that **Wireshark_801_11.pcapng** is not present (to trigger capture or download) or update the path accordingly.
3. Open a Bash shell (Git Bash, WSL, etc.) in your project directory.
4. Make the shell script executable:

```
chmod +x lab4.sh
```

```
./lab4.sh
```

Output:

```
./lab4.sh
Activating virtual environment...
Upgrading pip and installing required packages...
Requirement already satisfied: pip in ./venv/lib/python3.13/site-packages (25.0.1)
-
- # more items here
-
Running lab4.py...
🔍 Starting analysis of capture file: Wireshark_801_11.pcapng

Question 1: SSIDs issuing most beacon frames
a) No beacon frames found.

Question 2: 802.11 channel
a) Channel information not available.
```

```
-  
- # more items here  
-  
🔍 Analysis complete.  
Deactivating virtual environment...
```

Lab Analysis and Answers to Questions

1. What are the SSIDs of the two access points that are issuing most of the beacon frames in this trace? [Hint: look at the Info field. To display only beacon frames, enter `wlan.fc.type_subtype == 8` into the Wireshark display filter].

The two access points sending the most beacon frames are “30 Munroe St” and “Linksys_SES_24086”. These SSIDs correspond to the neighbor’s Linksys AP and the AP named “30 Munroe St” (to which the host was initially connected).

2. What 802.11 channel is being used by both of these access points [Hint: you’ll need to dig into the radio information in an 802.11 beacon frame]

Both of these APs are operating on channel 6. The capture was performed on channel 6 and even the neighboring AP’s beacons (Linksys_SES_24086) were overheard on the same channel. This indicates that 30 Munroe St and the Linksys AP were broadcasting on channel 6.

3. What is the interval of time between the transmissions of beacon frames from this access point (AP)? (Hint: this interval of time is contained in a field within the beacon frame itself).

Each AP’s beacon interval is about 0.1024 seconds (102.4 ms). This value is contained in the beacon frame itself as the “Beacon Interval” field. In other words, both 30 Munroe St and Linksys_SES_24086 send beacons roughly every 0.1024 s (a typical 100 ms interval in 802.11).

4. What (in hexadecimal notation) is the source MAC address on the beacon frame from this access point? Recall from Figure 7.13 in the text that the source, destination, and BSS are three addresses used in an 802.11 frame. For a detailed discussion of the 802.11 frame structure, see section 9.2.3-9.2.4.1 in the IEEE 802.11 standards document, excerpted here.

The source MAC address in a beacon frame is the MAC of the AP transmitting it. For example, in the 30 Munroe St beacon frame the source MAC is 00:16:b6:f7:1d:51. (This is the AP’s own MAC address, since the AP is the source of the beacon.)

5. What (in hexadecimal notation) is the destination MAC address on the beacon frame from 30 Munroe St??

Beacon frames are transmitted to the broadcast address. In the 30 Munroe St beacon, the destination MAC is ff:ff:ff:ff:ff:ff (the broadcast MAC for “all stations”)

6. What (in hexadecimal notation) is the MAC BSS ID on the beacon frame from 30 Munroe St?

The BSS ID in an 802.11 beacon is the MAC identifier for the network (usually the AP's MAC). In the 30 Munroe St beacon frame, the BSSID is 00:16:b6:f7:1d:51. (Notably, this is the same as the source MAC, since the AP is transmitting its own beacon.)

7. The beacon frames from the 30 Munroe St access point advertise that the access point can support four data rates and eight additional "extended supported rates." What are these rates? [Note: the traces were taken on a rather old AP].

The 30 Munroe St AP's beacon advertises four base supported data rates and eight extended supported rates. The supported rates are 1.0, 2.0, 5.5, 11.0 Mbps, and the extended supported rates are 6.0, 9.0, 12.0, 18.0, 24.0, 36.0, 48.0, 54.0 Mbps. These are the rates that the AP can use for communications (the first four are 802.11b rates, and the others are extended 802.11g rates).

8. Find the 802.11 frame containing the SYN TCP segment for this first TCP session (that downloads alice.txt) at t=24.8110. What are three MAC address fields in the 802.11 frame? Which MAC address in this frame corresponds to the wireless host (give the hexadecimal representation of the MAC address for the host)? To the access point? To the first-hop router? What is the IP address of the wireless host sending this TCP segment? What is the destination IP address for the TCP syn segment?

Frame at t=24.8110 (TCP SYN) – MAC and IP Addresses: At time ~24.8110 s, the wireless host sends a TCP SYN (for an HTTP request). In the 802.11 frame carrying this SYN, the three MAC address fields are: 00:13:02:d1:b6:4f (source – the wireless host's MAC), 00:16:b6:f4:eb:a8 (destination – the first-hop router's MAC), and 00:16:b6:f7:1d:51 (BSSID – the AP's MAC). The corresponding IP addresses are 192.168.1.109 for the wireless host (source IP) and 128.119.245.12 for the destination IP

9. Does the destination IP address of this TCP SYN correspond to the host, access point, first-hop router, or the destination web server?

Role of the TCP SYN's Destination IP: The destination IP 128.119.245.12 (seen in the SYN at t=24.8110) belongs to the remote web server (in fact, gaia.cs.umass.edu) – not the host, AP, or router. It is an external address on the Internet. In other words, the SYN is destined for a server on another network (the content server), which is neither the local wireless host nor the AP or first-hop router.

10. Find the 802.11 frame containing the SYNACK segment for this TCP session received at t=24.8277. What are three MAC address fields in the 802.11 frame? Which MAC address in this frame corresponds to the host? To the access point? To the first-hop router? Does the sender MAC address in the frame correspond to the IP address of the device that sent the TCP segment encapsulated within this datagram? (Hint: review Figure 6.19 in the text if you are unsure of how to answer this question, or the corresponding part of the previous question. It's particularly important that you understand this).

Frame at t=24.8277 (TCP SYN-ACK) – MAC Address Details: The SYN-ACK reply comes back to the host at t≈24.8277 s. In that 802.11 frame, the MAC addresses are: 00:16:b6:f4:eb:a8 as the source, 00:13:02:d1:b6:4f as the destination, and 00:16:b6:f7:1d:51 as the BSSID.

Here, 00:16:b6:f4:eb:a8 is the first-hop router's MAC (the device actually transmitting the frame to the AP), and 00:13:02:d1:b6:4f is the wireless host's MAC. Note that the sender's MAC (00:16:b6:f4:eb:a8) is the router, not the remote server

the server's IP is 128.119.245.12, but at the link layer the frame is sent by the router on behalf of the server. (This illustrates that the 802.11 source MAC in this downlink frame is the router, since the server's packets reach the wireless network via the router.)

11. What two actions are taken (i.e., frames are sent) by the host in the trace just after t=49, to end the association with the 30 Munroe St AP that was initially in place when trace collection began? (Hint one is an IP-layer action, and one is an 802.11-layer action).

Actions by Host to End Association (t ~49 s): Just after t=49 s, the host takes two actions to terminate its association with 30 Munroe St:

- (1) it sends a DHCP Release message to 192.168.1.1 (the router/DHCP server) to relinquish its IP address on that network, and
- (2) it sends an 802.11 Deauthentication frame to the AP.

The DHCP release (an IP-layer action) indicates the host is leaving the network, and the deauthentication frame (a link-layer action) formally disconnects the host from the Wi-Fi network. (In theory, one might expect a disassociation notification as well, but in this trace the host directly deauthenticates)

12. Let's look first at AUTHENTICATION frames. At t = 63.1680, our host tries to associate with the 30 Munroe St AP. Use the Wireshark display filter wlan.fc.subtype == 11 to show AUTHENTICATION frames sent from the host to and AP and vice versa. What form of authentication is the host requesting?

Authentication at t=63.1680 – Method Requested: At t=63.1680, the host tries to reconnect to 30 Munroe St. The host sends an Authentication frame to the AP, requesting Open System authentication (no encryption key required). This means the host is attempting to authenticate without any shared key (open authentication is essentially a null authentication, as the network is unsecured).

13. What is the Authentication SEQ value (authentication sequence number) of this authentication frame from host to AP?

The authentication frame from the host has a sequence number of 1, since it is the first message in the 802.11 authentication handshake. (In 802.11, the first Authentication request is Seq#1.)

14. The AP response to the authentication request is received at t = 63.1690. Has the AP accepted the form of authentication requested by the host?

Yes. At $t=63.1690$, the AP sends an authentication response indicating success. In 802.11 an accepted authentication is indicated by a Status Code = 0 (success) in the response frame. The quick reply from 30 Munroe St at 63.1690 suggests the AP accepted the host's open-system authentication request (i.e. the host is now authenticated).

15. What is the Authentication SEQ value of this authentication frame from AP to Host?

The AP's authentication response frame (to the host) uses sequence number 2, as it is the second message in the authentication exchange. (The sequence numbers increment, so the AP's reply to the host's Seq#1 is Seq#2.)

16. What rates are indicated in the frame as SUPPORTED RATES. Do not include in your answers below any rates that are indicates as EXTENDED SUPPORTED RATES.

In the host's Association Request frame (to 30 Munroe St), the Supported Rates information element lists 1, 2, 5.5, 11, 6, 9, 12, 18 Mbps. These are the non-extended rates that the host indicates it can use. (Any additional higher rates beyond 18 Mbps are advertised in a separate Extended Supported Rates element.)

17. Does the ASSOCIATION RESPONSE indicate a Successful or Unsuccessful association response?

The AP's Association Response indicates a successful association. The status in the response is a success code (the host re-associates with the AP without error). In fact, the trace narrative confirms that by $t=63.0$ the host "associates again" with 30 Munroe St, implying the Association Response was affirmative

18. Does the fastest (largest) Extended Supported Rate the host has offered match the fastest (largest) Extended Supported Rate the AP is able to provide?

Yes, the host's highest supported rate matches the AP's highest rate. The 30 Munroe St AP's fastest extended rate is 54.0 Mbps and the host's Association Request likewise included 54 Mbps as its top rate. In other words, both the AP and the host advertise 54 Mbps as the maximum supported data rate, so they are in agreement on the fastest rate available.

Troubleshooting Issues and Solutions

During the Wireshark analysis, several issues were encountered that required troubleshooting:

1. Live Capture Permissions Error:

- **Issue:** When attempting to run a live capture, the script threw a **Permission Denied** error for accessing the network interface.
- **Solution:** Running the capture script with administrator privileges (on Windows, using *Run as Administrator*; on Linux/macOS, using **sudo**) resolved the issue.

2. Missing or Incorrect Network Interface:

- **Issue:** The live capture failed because **wlan0** was not a valid interface on some machines.

- **Solution:** Used `pyshark.LiveCapture().interfaces` to list available network interfaces and prompted the user to select a correct one.

3. Capture File Not Found:

- **Issue:** When trying to analyze the trace file, the script initially failed because `Wireshark_801_11.pcapng` was missing.
- **Solution:** Implemented an automated download function to fetch the file from gaia.cs.umass.edu if it was not found locally.

4. Beacon Frames Not Detected:

- **Issue:** Some initial runs of the analysis showed **"No beacon frames found."**
- **Solution:** Verified the Wireshark display filter `wlan.fc.type_subtype == 8` and used `pyshark` to extract only beacon frames explicitly.

5. Incorrect MAC Address Extraction:

- **Issue:** The script initially extracted only `wlan.sa` (source MAC) but missed the `wlan.da` (destination) and `wlan.bssid`.
- **Solution:** Modified the parsing to extract and validate all three MAC addresses for each relevant frame.

6. Unsupported Data Rate Parsing:

- **Issue:** Some association request packets contained missing or corrupted rate fields.
- **Solution:** Implemented exception handling to skip malformed packets and ensure all extracted data rates were valid.

Challenges Faced

Several challenges arose during this analysis:

1. Filtering the Correct Packets:

- Identifying the exact frame at specific timestamps (e.g., `t=24.8110` for the TCP SYN) required precise filtering.
- Variations in packet timing due to system clock differences led to minor offsets.
- **Solution:** Implemented time-range tolerance (± 0.005 s) when searching for key frames.

2. Interpreting MAC Address Roles:

- In downlink traffic (AP to client), the source MAC was the **router** instead of the server.
- Understanding that Wi-Fi frames encapsulate Ethernet packets required careful interpretation.
- **Solution:** Cross-verified MAC roles with IP address sources.

3. Handling Encrypted Networks:

- If WPA2 was enabled on the AP during live capture, packet decryption was needed.
- **Solution:** Used open networks for analysis or preloaded Wireshark decryption keys.

4. Processing Large Capture Files Efficiently:

- The `.pcapng` file contained thousands of frames, slowing down analysis.
 - **Solution:** Used `keep_packets=False` to avoid memory overflow in `pyshark.FileCapture()`.
-

Conclusion

This lab successfully demonstrated how to analyze Wi-Fi 802.11 packets using **Wireshark and Python**. By capturing and processing beacon frames, authentication sequences, and TCP handshakes, we extracted key details about SSIDs, supported rates, MAC addresses, and network behavior.

Key takeaways:

- **Wi-Fi beacon frames** provide crucial data about network characteristics.
- **802.11 authentication and association** follow a predictable sequence, allowing verification of connection success.
- **TCP SYN and SYN-ACK** frames can be traced through MAC-layer addressing to distinguish between the **AP, first-hop router, and web server**.
- **Automating packet analysis with Python** streamlines the process, making Wireshark data more accessible.

Overall, this lab reinforced the importance of **packet analysis for troubleshooting and understanding wireless networks**, highlighting how **real-world Wi-Fi behaviors align with theoretical 802.11 standards**.