# Lab 1: Wireshark and your local network

The purpose of today's lab is to help you with the background and the "nuts & bolts" of internet technologies, and to give some empirical experience in "peeling back layers" of the internet.

This should hopefully be a fun bit of poking around with what your computer is actually doing all the time — for better or worse, I always find something new every time I look at the firehose of packets coming in and out of my machine.

—

# PreLab

First, make a copy of the prelab report document from the assignment in canvas.

## A. Install Wireshark

We will use Wireshark, [available here](). Follow the installation instructions to set up wireshark on your machine.

> Sometimes, you can run into some permission headaches getting wireshark access to your network traffic. The modern installer is pretty good at getting all the permissions it needs, but if you have issues, Google is going to be a better bet for debugging than staff probably.

> It is not a good idea to run Wireshark as root — it'll get all the packets, sure, but that's really opening yourself up for trouble.
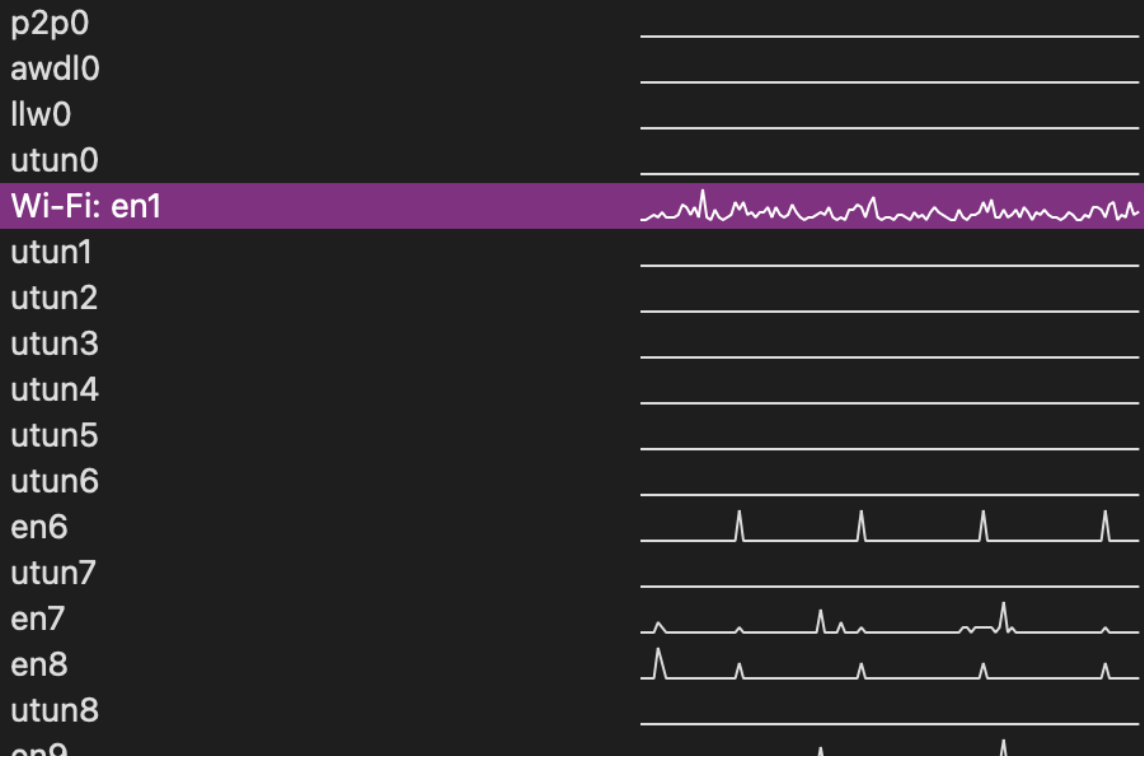
**There is no deliverable for this section.**

## B. Get comfortable with Wireshark

When you start Wireshark, you have to tell it where to capture packets from. In practice, this is a list of cryptic, short names; e.g. here's what I see:

## Capture

**...using this filter:** [🔖] Enter a capture filter ...

- p2p0
- awdl0
- llw0
- utun0
- **Wi-Fi: en1**
- utun1
- utun2
- utun3
- utun4
- utun5
- utun6
- en6
- utun7
- en7
- en8
- utun8
- en9

*I have active VMs on this machine, can you tell?*

**PreLab Q1:** What interfaces are available on your machines?

**PreLab Q2:** Where's WiFi?

**PreLab Q3:** What's all this other stuff?

# In-Lab

First, make a copy of the postlab report to fill in as you work. Some questions are tagged *PostLab*. Don't try to do these during the lab session or you will fall behind.

> **Warning:** In general, be careful when sniffing traffic. It can be illegal to monitor communications you were not supposed to have access to.

## A. [10 min] Finding and inspecting your own traffic

1. Open a terminal window and run:
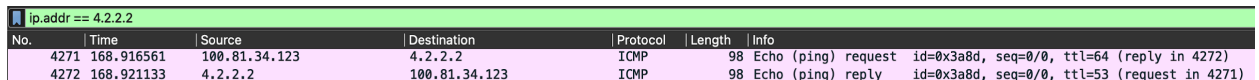
   ```
   ping 4.2.2.2
   ```

2. Open wireshark, and start collecting traffic on your default interface.

3. Add a filter:

   ```
   ip.addr==4.2.2.2
   ```

4. You should see something like this:

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 4271 | 168.916561 | 100.81.34.123 | 4.2.2.2 | ICMP | 98 | Echo (ping) request  id=0x3a8d, seq=0/0, ttl=64 (reply in 4272) |
| 4272 | 168.921133 | 4.2.2.2 | 100.81.34.123 | ICMP | 98 | Echo (ping) reply    id=0x3a8d, seq=0/0, ttl=53 (request in 4271) |

ip.addr == 4.2.2.2

5. Explore a bit in Wireshark, what can you learn about the packet you are observing?

6. Stop the capture. Go to `File → SaveAs` and [save a copy of this capture](#) to use during your postlab writeup.

**PostLab: Document your results.**

## B: [10 min] Insecure Chat

For this section, we will use the [netcat (nc) utility](#). If using MacOS or Linux, this is probably already built-in to your machine and can be run from a terminal window. If using Windows, you can get the `ncat` tool by installing the [nmap package](#). To run `ncat` you will need to run Command Prompt as administrator.

1. Ensure you are connected to eduroam or wahoo.

2. Set up a "listen" (`nc -l [PORT]`) or (`ncat -l [PORT]`) endpoint. This tells your machine to listen for incoming connections on a specific port. Ports are 16 bits, and the lower numbers are typically reserved. Choose your preferred port number.

   **NOTE**: If you are on Windows, you will need to disable the windows firewall for public networks to receive incoming connections.

3. List your IP address and port on our class [discovery board](#).

4. Open a new terminal, choose another IP address from the board, and try to connect (`nc [IP] [PORT]`). You may want to instruct netcat to use a timeout so that it will fail if the listener already has a connection. On my machine that looks like (`nc [IP] [PORT] -w 500`) or (`ncat [IP] [PORT] -w 500ms`). You may need to view the options in case the flag is different on your machine.

5. You should be able to chat! Introduce yourself, and find your virtual connection in real life. With whom did you connect?

```
$ nc -l 5000                                    $ nc localhost 5000
Hello from sender                               Hello from sender
Why hello there, this is the listener!          Why hello there, this is the listener!
```

   *What is "localhost"? Why can't you use that?*

6. Start a wireshark capture.

7. After sending a few messages back and forth, stop and save your wireshark capture.

8. Add a filter so that you are only looking at traffic from your chat session. (*Hint: what unique number did you control when you set this chat up?*)

9. Which machines can you see your chat traffic in Wireshark? Why? Can you see other chat traffic?

10. You might want to remove your IP address from the discovery board when you are finished with this section.

**PostLab: Document your results.**


## C: [10 min] Discover WiFi Networks Around You

Computers discover WiFi routers using a variety of *probe packets*. These probe packets contain information such as the name of the WiFi network (SSID), signal strength information, security capabilities, etc.

1. You will need a trace of raw WiFi packets. If you happen to have a Mac, you should be able to find a tool called `Wireless Diagnostics`. Once you open the tool go to `Window -> Sniffer`. Here you should see an option to set the `Channel` and `Channel Width`. Choose `Channel` as 5, leave width unchanged and press `Start`. The tool will run a wireless capture till you press `Stop`. The capture output pcap file can be found in `/var/tmp/`. This pcap file can be directly opened in Wireshark and analyzed.

   If you don't have a Mac, you can use the first pcap file that we recorded, available on canvas.

2. We are running a test network in lab. *Without connecting to it*, can you find it?

3. If you are capturing, let this run for about thirty seconds, then stop and save the capture.

4. What are the types of probe packets you can see from our test network?

5. Filter for only packets from our test network. What filter did you use?

6. What is the MAC address of the router for the test network?


**PostLab: Document your results.**



## D: [10 min] Snooping Connection Formation

Next, let's try to capture connection formation between a computer and the wireless router.

1. Again, if you have a Mac, you can use the Wireless Diagnostics sniffer to capture WiFi packets. You want to use channel 5 again. You can also do this in wireshark directly if you disconnect from all WiFi networks, select the gear icon in wireshark, and then check the "Monitor" box:
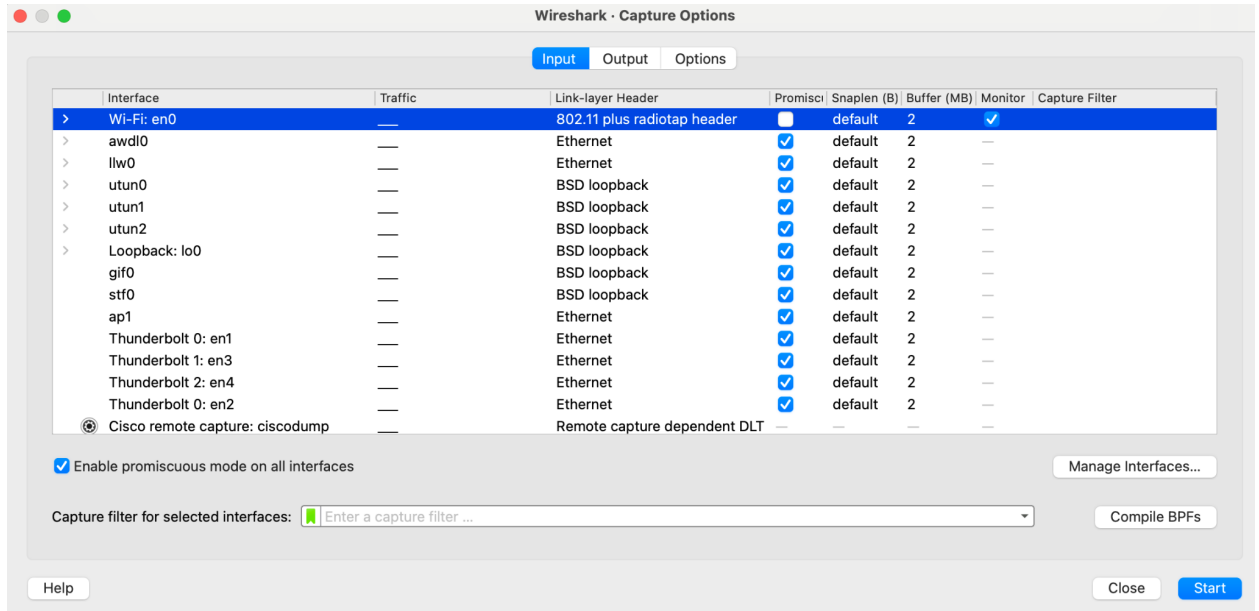
If you do not have a Mac, you can use the second pcap file on canvas, and skip to step 4.

2. While scanning, have another computer join the local WiFi network we are running. You can use your phone/tablet or work with a neighbor.

3. Once the connection is complete, wait another ten-ish seconds, then stop and save the capture.

4. With the trace:

    a. Filter the traffic to isolate the connection process.

    b. What is the first type of packet that the computer sends to the router to initiate connection formation?

    c. How do you figure out which packet is sent next? Is there some information in the packet that helps you identify this?

    d. How do you know when the connection is established?

**PostLab: Document your results.**

# E: [10 min] Inspecting Protocol Information

A main reason to use Wireshark is to be able to inspect the packets being sent. Remember, a packet is just a buffer of data (a string of 1s and 0s). You can see this in Wireshark when you select a packet:
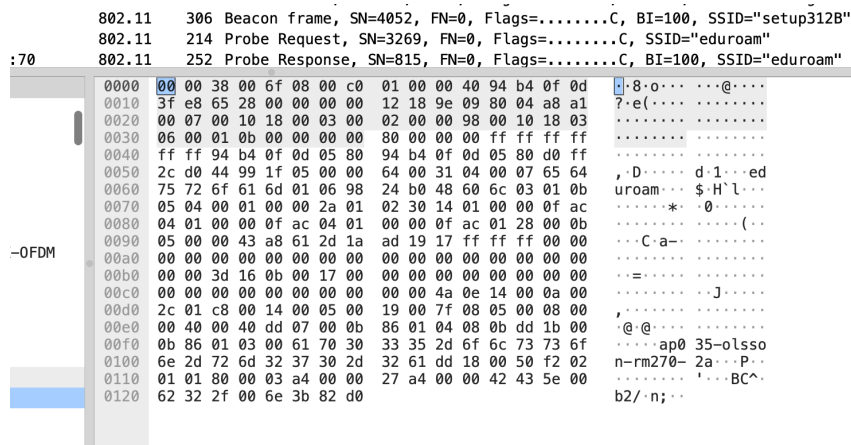


Wireshark understands virtually every network protocol, and is able to interpret that buffer of data and provide a most understandable representation for us to inspect.

1. Do a new scan with the wireless diagnostics tool (if you can), this time on channel 11. Otherwise open the pcap 3 file on canvas.

2. Look for a WiFi beacon packet. Find the field in the packet that includes the SSID. What is the Tag Number for that field? Look for another beacon packet, and find its SSID. Is the Tag Number the same for the SSID field?

3. Find a beacon packet from the eduroam network. WiFi is meant to be extensible, so the protocol permits "vendor specific" tags to be included in the packets. What is the Tag Number for vendor specific tags?

4. Look through the vendor specific tags for the eduroam network. What is the name that UVA ITS gave to the access point that sent the beacon?

**PostLab: Document your results.**

---

# PostLab

Go back and fill in the *PostLab* questions in the PostLab Report Document.

Once you are finished, submit your report via Gradescope.