



EECE5155: Wireless Sensor Networks and the Internet of Things Laboratory Assignment 3 Report

Author (s):

- Nisha Goel
- Srishti Raju
- Group number on Canvas: 8

Date: 11/11/2024

Task 1 (Part 3)

1. Experimental setup

To accomplish this assignment, we set up Colosseum, a wireless network emulator. We started by setting up the Cisco VPN client to log into the Colosseum website and created our private/ public key pair. The SSH public key was uploaded onto the Colosseum portal, connecting our system and the Software Radio Nodes (SRNs) reserved in Colosseum. Next, we made a reservation for 2 SRNs with the webinar-interactive-v1 image. We opened a terminal for each SRN and SSH'd into the root using the host name provided by Colosseum.

Our first task was to run and analyze an RF emulator scenario with a 1 GHz center frequency. One terminal was used for transmitting a tone at the preset frequency, and the other terminal was used for receiving the tone and displaying it on a spectrum analyzer.

2. Results

We were successfully able to set up the RF emulator. The tone was initially set at 1.01 GHz, and we could see the peak at this frequency in Figure 1, where the peak is shown. We then modified the frequency, shown in Figures 2 and 4 to 0.99 GHz and 0.9 GHz respectively, and observed the change in the peak frequency in Figure 3; there was no peak frequency at 0.9 GHz in Figure and we analyze why.

- 1) The tone is sent at 1.01 GHz, which, when observed in the *uhd_tx_tone.sh* and *uhd_rx_fft.sh* scripts, is what it was initially set to. We modify this frequency for the following parts. In Figure 1, we can see the peak at this frequency, confirming our conclusions.

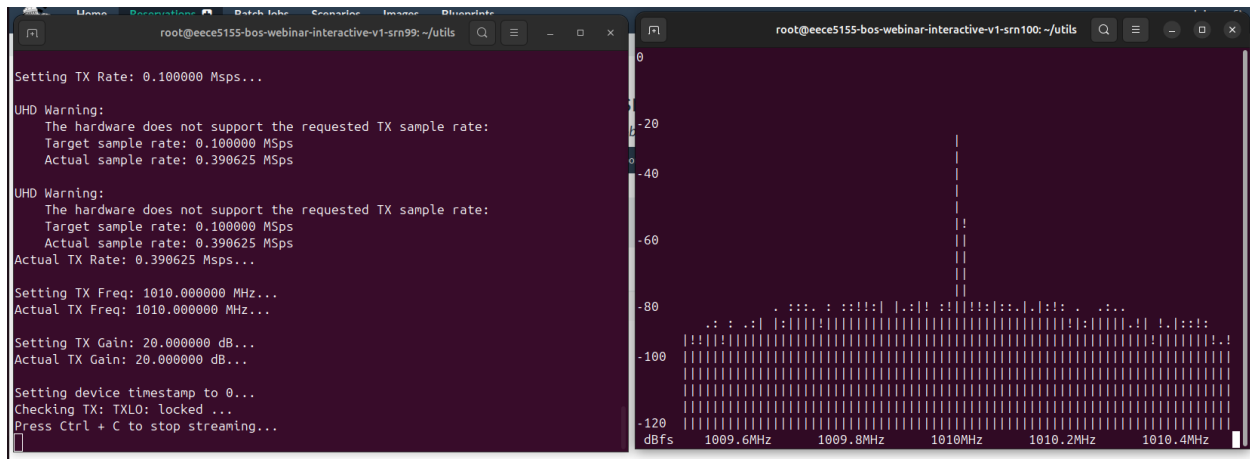


Figure 1: Spectrum at pre-set frequency

2) We updated the frequency to 0.99 GHz in the `uhd_tx_tone.sh` and `uhd_rx_fft.sh` scripts, as shown in Figure 2. As seen in Figure 3, the peak is updated to reflect this new frequency, peaking at 990 MHz.

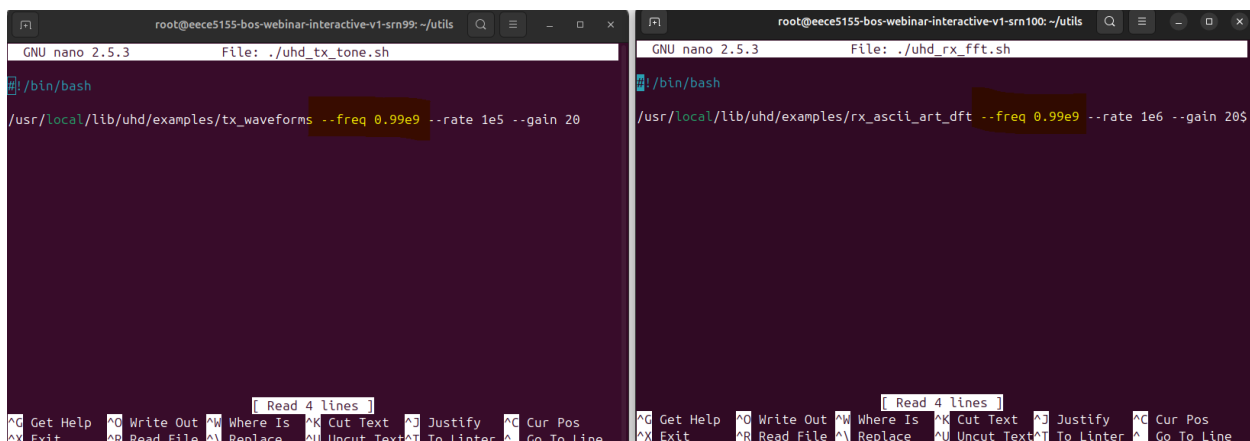


Figure 2: Frequency is modified to be 0.99 GHz

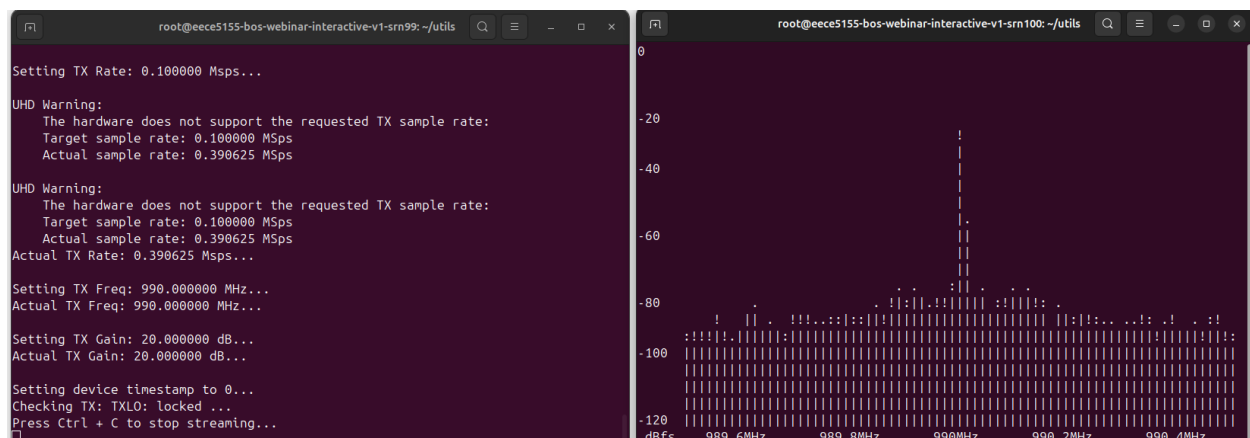


Figure 3: Spectrum at 0.99 GHz

3) We updated the frequency to 0.9 GHz, or 900 MHz, in the `uhd_tx_tone.sh` `uhd_rx_fft.sh` scripts as shown in Figure 4. Since the center frequency is 1000.0 MHz, and the max scenario bandwidth is 80.0 MHz, that means that the operational frequency range that is between 920-1080 MHz (<https://colosseumneu.freshdesk.com/support/solutions/articles/61000277641-test-scenario-all-paths-0-db-1009>). As 900 MHz is just outside that range, it makes sense that we cannot see a peak in Figure 5.

```

root@eece5155-bos-webinar-interactive-v1-srn99: ~/utils
GNU nano 2.5.3 File: ./uhd_tx_tone.sh Modified
#!/bin/bash
/usr/local/lib/uhd/examples/tx_waveforms --freq 0.9e9 --rate 1e5 --gain 20

root@eece5155-bos-webinar-interactive-v1-srn100: ~/utils
GNU nano 2.5.3 File: ./uhd_rx_fft.sh Modified
#!/bin/bash
/usr/local/lib/uhd/examples/rx_ascii_art_fft --freq 0.9e9 --rate 1e6 --gain 20 $

```

Figure 4: Frequency is modified to be 0.9 GHz

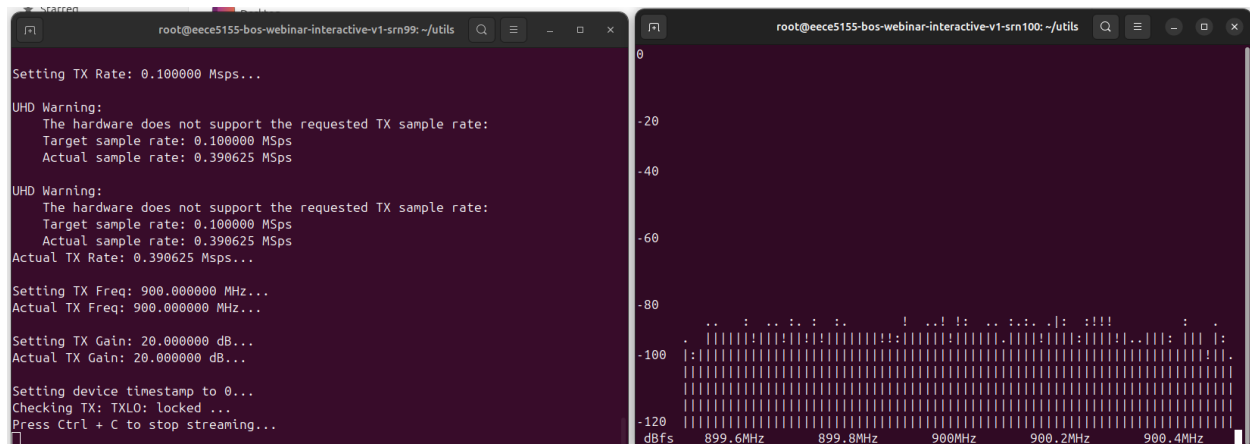


Figure 5: Spectrum at 0.9 GHz

3. Learnt Lessons

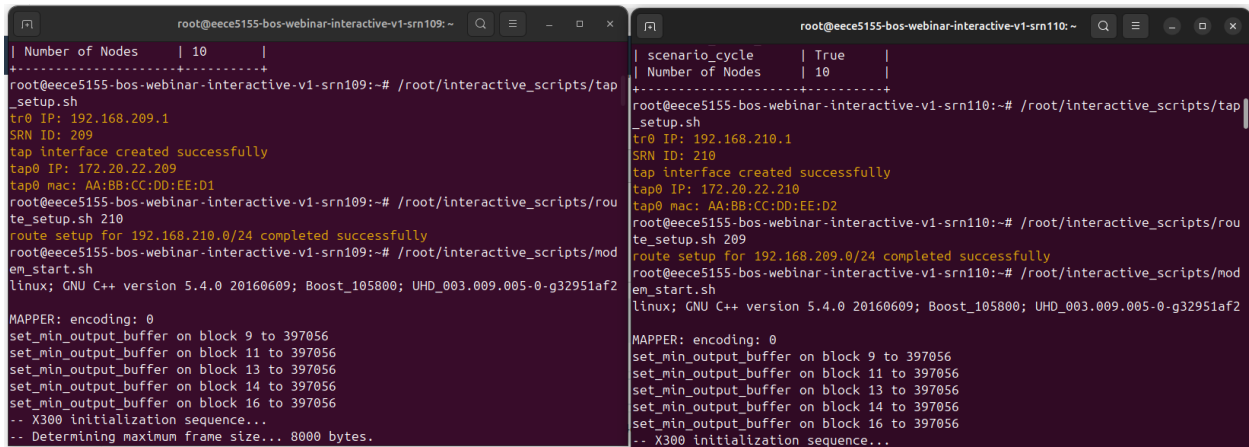
In this assignment, we learned how to set up and connect to the Colosseum wireless network emulator to observe the traffic between WiFi nodes. For this first task, we learned how to verify if the RF emulator is set up properly. We then adjusted the frequencies for the tone and analyzed the receiving spectrum, and learned why we get the result we do when adjusting the frequency from 0.99 GHz to 0.9 GHz.

Task 2 (Part 4)

1. Experimental setup

In this task, we use the SRNs to emulate Wi-Fi nodes and test wireless communication between them. Each SRN is assigned a unique ID and tr0 IP address, enabling traffic routing via virtual tap interfaces configured through scripts (*tap_setup.sh* for tap interfaces and *route_setup.sh* for routing tables). After setting up these interfaces, we activate the Wi-Fi modems on each SRN using *modem_start.sh* to simulate real Wi-Fi connections. We then open two new terminals while the WiFi terminals remain open in the background. In the new terminals, we use traffic generation tools like ping to create and measure data flows between nodes (using the tr0 IP for the other SRN), allowing us to analyze network metrics such as throughput, latency, and packet loss and assess Wi-Fi performance and protocol behavior in a controlled, virtual environment.

2. Results



```
root@eece5155-bos-webinar-interactive-v1-srn109:~# /root/interactive_scripts/tap_setup.sh
+-----+
| Number of Nodes | 10 |
+-----+
root@eece5155-bos-webinar-interactive-v1-srn109:~# /root/interactive_scripts/tap_setup.sh
tr0 IP: 192.168.209.1
SRN ID: 209
tap interface created successfully
tap0 IP: 172.20.22.209
tap0 mac: AA:BB:CC:DD:EE:D1
root@eece5155-bos-webinar-interactive-v1-srn109:~# /root/interactive_scripts/route_setup.sh 210
route setup for 192.168.210.0/24 completed successfully
root@eece5155-bos-webinar-interactive-v1-srn109:~# /root/interactive_scripts/modem_start.sh
linux; GNU C++ version 5.4.0 20160609; Boost_105800; UHD_003.009.005-0-g32951af2
MAPPER: encoding: 0
set_min_output_buffer on block 9 to 397056
set_min_output_buffer on block 11 to 397056
set_min_output_buffer on block 13 to 397056
set_min_output_buffer on block 14 to 397056
set_min_output_buffer on block 16 to 397056
-- X300 initialization sequence...
-- Determining maximum frame size... 8000 bytes.

root@eece5155-bos-webinar-interactive-v1-srn110:~# /root/interactive_scripts/tap_setup.sh
+-----+
| scenario_cycle | True |
| Number of Nodes | 10 |
+-----+
root@eece5155-bos-webinar-interactive-v1-srn110:~# /root/interactive_scripts/tap_setup.sh
tr0 IP: 192.168.210.1
SRN ID: 210
tap interface created successfully
tap0 IP: 172.20.22.210
tap0 mac: AA:BB:CC:DD:EE:D2
root@eece5155-bos-webinar-interactive-v1-srn110:~# /root/interactive_scripts/route_setup.sh 209
route setup for 192.168.209.0/24 completed successfully
root@eece5155-bos-webinar-interactive-v1-srn110:~# /root/interactive_scripts/modem_start.sh
linux; GNU C++ version 5.4.0 20160609; Boost_105800; UHD_003.009.005-0-g32951af2
MAPPER: encoding: 0
set_min_output_buffer on block 9 to 397056
set_min_output_buffer on block 11 to 397056
set_min_output_buffer on block 13 to 397056
set_min_output_buffer on block 14 to 397056
set_min_output_buffer on block 16 to 397056
-- X300 initialization sequence...
```

Figure 6: Setting up the WiFi interfaces and routes for both SRNs

```
root@eece5155-bos-webinar-interactive-v1-srn109: ~  
Subtype: Data  
seq nr: 632  
mac 1: aa:bb:cc:dd:ee:d2  
mac 2: aa:bb:cc:dd:ee:d1  
mac 3: 42:42:42:42:42:42  
instantaneous fer: 0.999756  
.....E..Tv<..@.....v..F.2.8-g.....r.....!#$%&'()*+,-./01234567  
new mac frame (length 88)  
=====  
duration: 00 00  
frame control: 00 08 (DATA)  
Subtype: Data  
seq nr: 634  
mac 1: 33:33:0:0:0:2  
mac 2: aa:bb:cc:dd:ee:d2  
mac 3: 42:42:42:42:42:42  
instantaneous fer: 0.5  
.....U.....  
unknown ether type  
new mac frame (length 88)  
=====  
duration: 00 00  
frame control: 00 08 (DATA)  
Subtype: Data  
seq nr: 633  
mac 1: 33:33:0:0:0:2  
mac 2: aa:bb:cc:dd:ee:d1  
mac 3: 42:42:42:42:42:42  
instantaneous fer: 0.999756  
.....W.....  
root@eece5155-bos-webinar-interactive-v1-srn110: ~  
Subtype: Data  
seq nr: 632  
mac 1: aa:bb:cc:dd:ee:d2  
mac 2: aa:bb:cc:dd:ee:d1  
mac 3: 42:42:42:42:42:42  
instantaneous fer: 0.999756  
.....E..Tv<..@.....v..F.2.8-g.....r.....!#$%&'()*+,-./01234567  
unknown ether type  
new mac frame (length 88)  
=====  
duration: 00 00  
frame control: 00 08 (DATA)  
Subtype: Data  
seq nr: 634  
mac 1: 33:33:0:0:0:2  
mac 2: aa:bb:cc:dd:ee:d2  
mac 3: 42:42:42:42:42:42  
instantaneous fer: 0.5  
.....U.....  
new mac frame (length 88)  
=====  
duration: 00 00  
frame control: 00 08 (DATA)  
Subtype: Data  
seq nr: 633  
mac 1: 33:33:0:0:0:2  
mac 2: aa:bb:cc:dd:ee:d1  
mac 3: 42:42:42:42:42:42  
instantaneous fer: 0.999756  
.....W.....
```

Figure 7: Wi-Fi process to run in the background

4) From the above Figure 6, the image displays the initialization process, where each SRN executes *tap_setup.sh*, *route_setup.sh*, and *modem_start.sh* scripts to create tap interfaces, set up routing, and start the Wi-Fi modem, respectively. This setup enables both SRNs to exchange data packets over the Wi-Fi network.

Figure 7 shows packet data exchanged between the two nodes. Specifically, each node appears to be transmitting and receiving Media Access Control (MAC) frames indicating that the SRNs are exchanging data frames over the network. These frames contain information such as MAC addresses and sequence numbers, which are used in Wi-Fi protocols to manage and verify data transmission. The frames shown also have details about frame control and instantaneous error values, suggesting that both nodes are capturing details of received packets to analyze performance metrics like transmission errors and latency.

Key findings are:

- **MAC Frame Analysis:** The data frames being exchanged between the nodes include MAC addresses in the format expected for Wi-Fi communication. This indicates that the nodes are successfully exchanging Wi-Fi data frames at the MAC layer.
- **Instantaneous Error Measurements:** Both nodes are logging instantaneous error values, which can be useful for performance analysis. High error values might suggest interference, weak signals, or issues in the network configuration.
- **Successful Configuration:** Since each SRN can receive and process MAC frames from the other, it implies that the Wi-Fi interfaces and routing tables were correctly configured, allowing successful communication between nodes.

This setup appears to be functioning as expected for Wi-Fi communication testing, where nodes are exchanging data frames to assess network behavior and performance.

```

root@eece5155-bos-webinar-interactive-v1-srn109: ~
64 bytes from 192.168.210.1: icmp_seq=296 ttl=64 time=14.4 ms
64 bytes from 192.168.210.1: icmp_seq=297 ttl=64 time=14.7 ms
64 bytes from 192.168.210.1: icmp_seq=298 ttl=64 time=14.3 ms
64 bytes from 192.168.210.1: icmp_seq=299 ttl=64 time=14.5 ms
64 bytes from 192.168.210.1: icmp_seq=300 ttl=64 time=14.3 ms
64 bytes from 192.168.210.1: icmp_seq=301 ttl=64 time=14.5 ms
64 bytes from 192.168.210.1: icmp_seq=302 ttl=64 time=14.7 ms
64 bytes from 192.168.210.1: icmp_seq=303 ttl=64 time=14.4 ms
64 bytes from 192.168.210.1: icmp_seq=304 ttl=64 time=14.6 ms
64 bytes from 192.168.210.1: icmp_seq=305 ttl=64 time=14.6 ms
64 bytes from 192.168.210.1: icmp_seq=306 ttl=64 time=14.7 ms
64 bytes from 192.168.210.1: icmp_seq=307 ttl=64 time=14.4 ms
64 bytes from 192.168.210.1: icmp_seq=308 ttl=64 time=14.5 ms
64 bytes from 192.168.210.1: icmp_seq=309 ttl=64 time=14.6 ms
64 bytes from 192.168.210.1: icmp_seq=310 ttl=64 time=14.5 ms
64 bytes from 192.168.210.1: icmp_seq=311 ttl=64 time=14.5 ms
64 bytes from 192.168.210.1: icmp_seq=312 ttl=64 time=14.7 ms
64 bytes from 192.168.210.1: icmp_seq=313 ttl=64 time=14.4 ms
64 bytes from 192.168.210.1: icmp_seq=314 ttl=64 time=14.4 ms
^C
--- 192.168.210.1 ping statistics ---
314 packets transmitted, 314 received, 0% packet loss, time 313310ms
rtt min/avg/max/mdev = 13.427/18.128/64.354/11.020 ms
root@eece5155-bos-webinar-interactive-v1-srn109: ~

root@eece5155-bos-webinar-interactive-v1-srn110: ~
64 bytes from 192.168.209.1: icmp_seq=288 ttl=64 time=19.3 ms
64 bytes from 192.168.209.1: icmp_seq=289 ttl=64 time=18.3 ms
64 bytes from 192.168.209.1: icmp_seq=290 ttl=64 time=17.4 ms
64 bytes from 192.168.209.1: icmp_seq=291 ttl=64 time=16.4 ms
64 bytes from 192.168.209.1: icmp_seq=292 ttl=64 time=15.3 ms
64 bytes from 192.168.209.1: icmp_seq=293 ttl=64 time=14.7 ms
64 bytes from 192.168.209.1: icmp_seq=294 ttl=64 time=14.4 ms
64 bytes from 192.168.209.1: icmp_seq=295 ttl=64 time=14.7 ms
64 bytes from 192.168.209.1: icmp_seq=296 ttl=64 time=14.8 ms
64 bytes from 192.168.209.1: icmp_seq=297 ttl=64 time=14.8 ms
64 bytes from 192.168.209.1: icmp_seq=298 ttl=64 time=14.5 ms
64 bytes from 192.168.209.1: icmp_seq=299 ttl=64 time=14.8 ms
64 bytes from 192.168.209.1: icmp_seq=300 ttl=64 time=14.6 ms
64 bytes from 192.168.209.1: icmp_seq=301 ttl=64 time=14.7 ms
64 bytes from 192.168.209.1: icmp_seq=302 ttl=64 time=14.6 ms
64 bytes from 192.168.209.1: icmp_seq=303 ttl=64 time=14.7 ms
64 bytes from 192.168.209.1: icmp_seq=304 ttl=64 time=14.4 ms
64 bytes from 192.168.209.1: icmp_seq=305 ttl=64 time=14.8 ms
64 bytes from 192.168.209.1: icmp_seq=306 ttl=64 time=14.8 ms
^C
--- 192.168.209.1 ping statistics ---
306 packets transmitted, 306 received, 0% packet loss, time 305305ms
rtt min/avg/max/mdev = 13.420/18.300/65.193/11.437 ms
root@eece5155-bos-webinar-interactive-v1-srn110: ~

```

Figure 8: Pinging the tr0 interface of the other SNR for each node

5)

1) Average Round-Trip Time (RTT):

Node 1: 18.128 ms

Node 2: 18.300 ms

This indicates a relatively low and stable round-trip time between the two nodes, showing that the network is performing efficiently.

2) One-Way Delay:

Node 1: $(18.128/2) = 9.064$ ms

Node 2: $(18.300/2) = 9.15$ ms

3) Packet Loss

- Both nodes report 0% packet loss, indicating that all packets sent during the ping test were successfully received by the other node.
- This zero-packet loss suggests a stable connection with no significant interference or congestion, which is ideal for testing and indicates that the Wi-Fi setup is functioning well.

The results indicate a well-functioning Wi-Fi network between the two SRNs with low latency, no packet loss, and stable round-trip times. This setup is performing effectively for experimental purposes, with minimal delays and no loss, making it suitable for further network performance testing.

3. Learnt Lessons

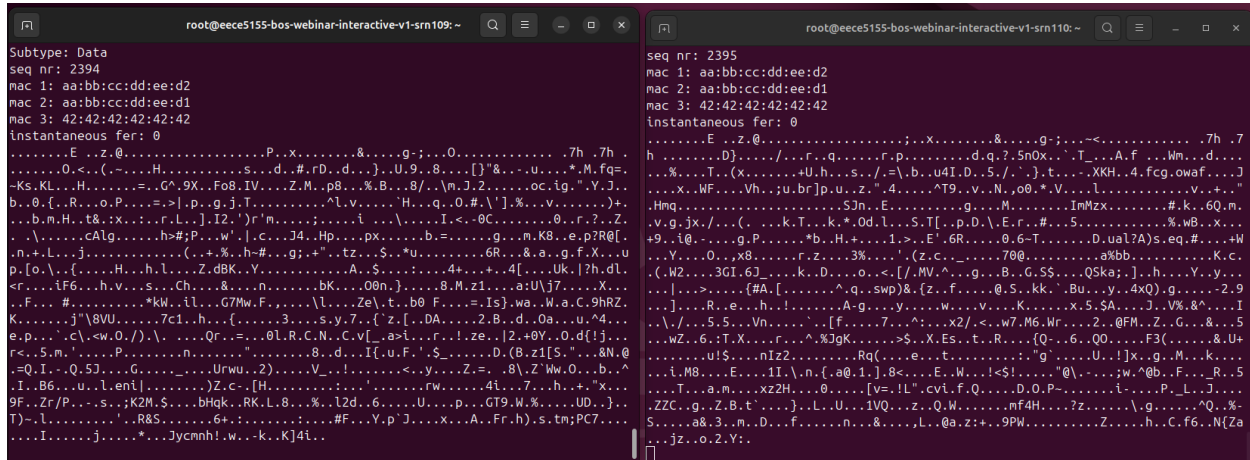
For this second task, we learned how to start the WiFi nodes for both SRNs, and verify if done correctly. We ran into a couple issues, when pinging to tr0 of the other SRN, where we were getting “connect: Network is unreachable.” We later realized that for some reason, our nodes were unable to connect to EEPROM, and it was a node issue (for nodes 101 and 102), and not an issue with how we set up the nodes. It was interesting to experience this issue though, and we learned that creating a new reservation with different nodes helped.

Task 3 (Part 5)

1. Experimental setup

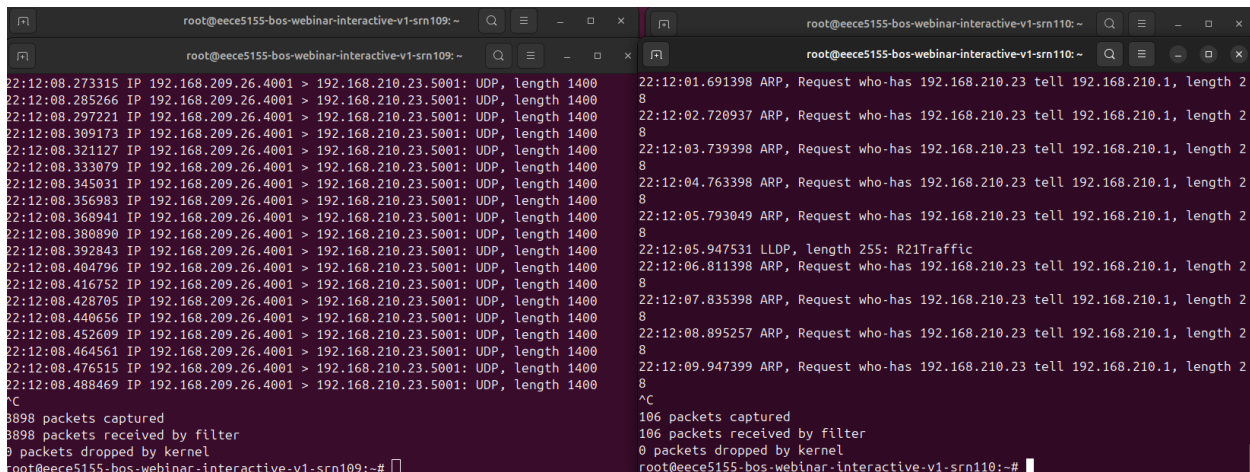
The Colosseum Traffic Generator (TGEN) is initiated to create packet flows between Wi-Fi nodes (SRNs) according to a predefined traffic scenario. Using the command *colosseumcli tg start 10090*, TGEN simulates network traffic with specific parameters like packet size and transmission rate to emulate real-world Wi-Fi conditions. After allowing a few minutes for TGEN to start, we monitor traffic on each SRN's *tr0* interface with *tcpdump -i tr0*, capturing details on packet characteristics and flow rates. The setup ensures that packets received on one SRN are forwarded to the other via the RF channel emulator, enabling bidirectional communication. By analyzing the captured data, we assess network performance and traffic patterns, providing insights into latency, packet size, and throughput.

2. Results



```
root@eece5155-bos-webinar-interactive-v1-srn109: ~
Subtype: Data
seq nr: 2394
mac 1: aa:bb:cc:dd:ee:d2
mac 2: aa:bb:cc:dd:ee:d1
mac 3: 42:42:42:42:42:42
instantaneous fer: 0
.....E..z@.....P..X.....&.....g.....0.....7h..7h..
.....O<.....(.....H.....S.....d.....rD.....d.....).....U9.....8.....[.....].....*.....M.....f.....q.....
-Ks.....KL.....H.....=.....G^.....9X.....Fo8.....IV.....Z.....M.....p8.....%.....B.....8/.....\n.....J.....2.....oc.....ig.....Y.....J.....
b.....0.....{.....R.....o.....P.....=.....>.....].....p.....g.....J.....T.....l.....v.....^.....l.....v.....^.....H.....q.....o.....#.....\.....].....%.....v.....).....+.....
...b.....m.....H.....t8.....x.....t.....r.....L.....].....I2.....').....r.....m.....;.....l.....\.....\.....I.....<.....-.....0C.....m.....0.....r.....?.....Z.....
.....\.....\.....cAlg.....h.....>.....P.....w.....'.....c.....J4.....Hp.....px.....b.....=.....g.....m.....K8.....e.....p.....RQ[.....
.....n.....+.....L.....j.....j.....(.....+.....%.....h.....#.....g.....+.....tz.....$.....*.....u.....6R.....&.....a.....g.....f.....X.....u.....
p.....[.....o.....\.....{.....H.....h.....L.....Z.....d.....BK.....Y.....A.....S.....$.....4.....+.....+.....4[.....UK.....].....7h.....dl.....
<.....r.....iF6.....h.....v.....s.....Ch.....&.....n.....bK.....00n.....}.....8.....M.....z1.....a.....Uj7.....X.....
.....F.....#.....#.....*.....k.....i.....l.....G7Mw.....F.....[.....L.....Ze.....t.....b0.....F.....=.....Is.....}.....wa.....W.....a.....C.....9h.....R.....Z.....
K.....K.....j.....\8VU.....).....7c1.....h.....(.....3.....s.....y.....7.....{.....z.....[.....DA.....2.....B.....d.....0a.....u.....4.....
e.....p.....'.....c.....\w.....O...../.....\.....Qr.....=.....0.....l.....R.....C.....N.....C.....v.....[.....a.....>.....l.....r.....!.....ze.....].....2.....+.....0V.....o.....d.....[.....j.....
r.....c.....5.....m.....'.....P.....n.....n.....n.....8.....d.....I.....(.....u.....F.....'.....$.....Z.....D.....(.....B.....z1.....[.....5.....".....&.....N.....@.....
=.....Q.....I.....-.....Q.....J.....5.....G.....G.....Urwu.....).....2.....).....V.....!.....!.....c.....y.....Z.....=.....8.....Z.....W.....O.....b.....^.....
.I.....B6.....u.....l.....ent.....].....).....Z.....c.....[.....H.....(.....r.....w.....4.....i.....7.....h.....+.....".....x.....
9F.....Zr/P.....s.....;.....k2M.....$.....bHq.....RK.....L.....8.....%.....l2d.....6.....U.....p.....p.....GT9.....W.....%.....UD.....).....
T.....-.....l.....S.....R&S.....+.....+.....+.....+.....#.....F.....Y.....p.....J.....X.....A.....Fr.....h.....).....s.....tm.....PC7.....
.....I.....j.....j.....*.....Jycnnh.....!.....w.....-.....K.....Kj4.....
```

Figure 9: Traffic generator



```
root@eece5155-bos-webinar-interactive-v1-srn109: ~
22:12:08.273315 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.285266 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.297221 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.309173 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.321127 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.333079 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.345031 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.356983 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.368941 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.380890 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.392843 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.404796 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.416752 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.428705 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.440656 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.452609 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.464561 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.476515 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
22:12:08.488469 IP 192.168.209.26.4001 > 192.168.210.23.5001: UDP, length 1400
^C
3898 packets captured
3898 packets received by filter
0 packets dropped by kernel
root@eece5155-bos-webinar-interactive-v1-srn109: ~#

root@eece5155-bos-webinar-interactive-v1-srn110: ~
22:12:01.691398 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:02.720937 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:03.739398 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:04.763398 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:05.793049 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:05.947531 LLDP, length 255: R21Traffic
22:12:06.811398 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:07.835398 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:08.895257 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
22:12:09.947399 ARP, Request who-has 192.168.210.23 tell 192.168.210.1, length 28
^C
106 packets captured
106 packets received by filter
0 packets dropped by kernel
root@eece5155-bos-webinar-interactive-v1-srn110: ~#
```

Figure 10: Packet flow on the *tr0* interface for each node

6) Based on the screenshots in Figure 9, the traffic generated between the two nodes shows a mixture of UDP and ARP packets with varying packet sizes and transmission rates. Here are the observed characteristics:

1. **Packet Size:**

- a. The UDP packets appear with a length of 1400 bytes, which is typical for bulk data transfers or high-throughput scenarios.
- b. ARP packets have a smaller, consistent length, mainly used for resolving IP addresses to MAC addresses, indicating that there is some basic network discovery or address resolution happening alongside data transfers.

2. **Packet Rate:**

- a. The UDP packets are being transmitted at a high rate, as shown by multiple consecutive packets with minimal time gaps (in milliseconds) between them. This indicates a relatively high data throughput, suggesting a test of network load handling capability.
- b. The ARP requests are intermittent and less frequent, which aligns with typical network behavior where ARP packets are only sent when address resolution is needed.

The traffic generated in this scenario reflects a high-bandwidth, UDP-heavy network load, testing the Wi-Fi nodes' ability to handle large data packets with minimal delay. The consistent UDP packet size and high transmission rate are ideal for evaluating throughput and latency in a controlled Wi-Fi environment. The occasional ARP requests show normal network operation. Overall, this setup is suitable for assessing the capacity and performance of Wi-Fi nodes under heavy traffic, providing insights into their handling of large data packets and network latency under load.

3. **Learnt Lessons**

For this third task, we learned how to start the traffic generator with the WiFi nodes still running in the background. We were able to observe the packet flows on the tr0 interface and observe the packet forwarding between the SRNs. Finally, we learned about best practices, stopping the RF and TGEN scenarios when we finish the tasks. As a whole, this lab was really fun and interesting to learn about how wireless networks work, especially since we use WiFi in our daily lives!