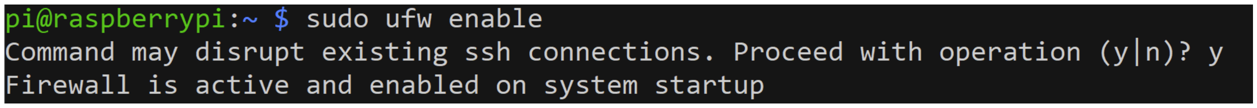
CS437: Internet of Things
Lab 3

Name: Nishant Sheikh, Molly Yang
NetID: nsheikh2, tyy2
Late days used: 0
Demo video (for Router Configuration):
https://drive.google.com/file/d/1BmvCxkO1MaNafpUalQ9XFYPNKWU8TEoX/view?usp=sharing

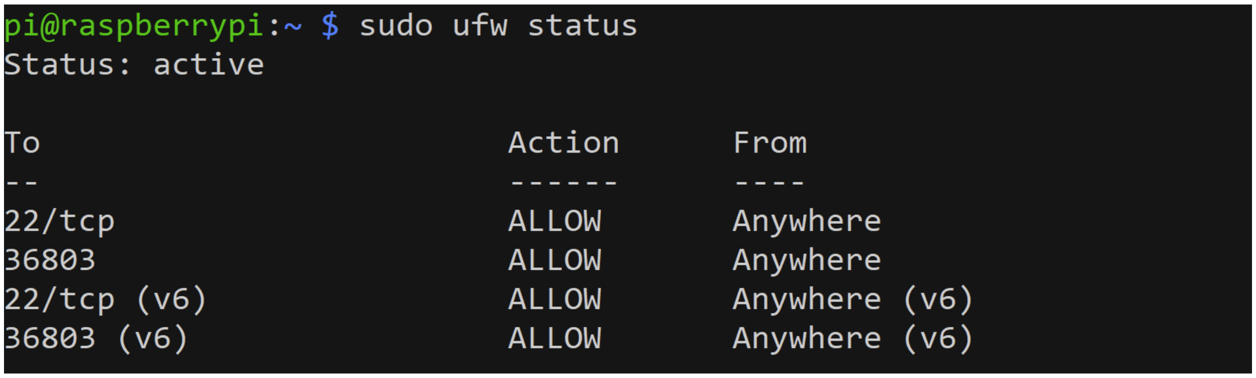
Firewall

- ☐ Read Firewall (computing) - Wikipedia
- ☐ Read Raspberry Pi Documentation - Configuration



```
pi@raspberrypi:~ $ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

Figure 1. Firewall installed and enabled



```
pi@raspberrypi:~ $ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
36803                      ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
36803 (v6)                 ALLOW       Anywhere (v6)
```

Figure 2. Firewall status showing port 22/tcp ssh is allowed, as well as the port used for lab 2

Firewalls

What sorts of firewall technologies do you feel are helpful for IoT? What layer(s) of firewall do you feel are most appropriate to protect your device?
Threshold-based filtering to block packets at the IP layer before any connection is established. Application layer would also be appropriate to identify unwanted services regardless of using a standard port or any abuse in an established protocol.

Raspberry Pi OS Firewalls

What are your thoughts on these security practices? Can you think of any possible attacks not covered by these configuration practices?
With ufw and Fail2Ban, Raspberry Pi OS security practices would be sufficient for most applications. However, if anyone's life is on the line, for medical devices or automotives, the security measures should also be implemented at the application level. If someone has enough patience and resources, they could find an allowed port, rules used to design attacks to bypass the checks and brute force their way in. It is especially vulnerable when the firewall mechanisms are widely known.

Findings and Applications

When the firewall is enabled on the raspberry pi, the VSCode remote ssh would not work since it uses another port other than port 22. The port that VSCode wants to access can be found in the log when VSCode tries to connect. It was interesting to find the port being blocked in action, and how the connection went right through after adding the allowed ports. This can be applied to any projects we come across in the future. After setting up all the necessary components in an IoT system or any other platform, the firewall can be set up to only allow traffic through the ports we will utilize. This adds an additional layer of security and minimizes chances of any malicious attempts.

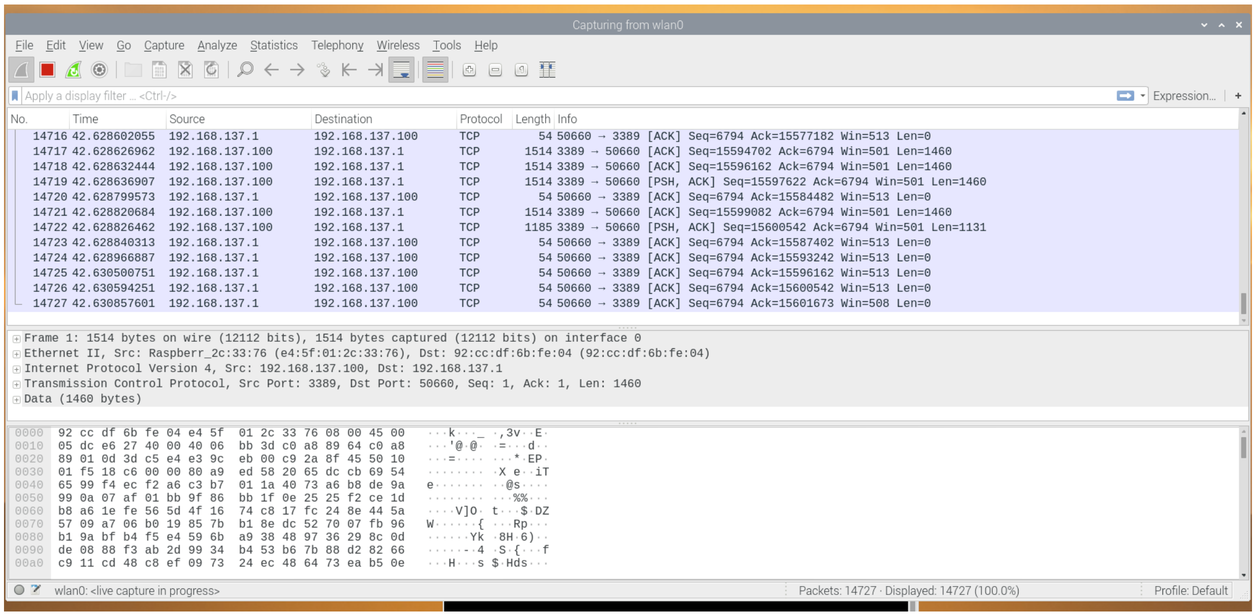
Packet Inspection

- ☐ Read https://en.wikipedia.org/wiki/Pcap, https://www.comparitech.com/net-admin/pcap-guide/
- ☐ Read https://en.wikipedia.org/wiki/Wireshark, https://www.youtube.com/watch?v=TkCSr30UojMr

pcap

What is pcap? What sorts of information does pcap capture about packets?
PCAP (packet captures) captures wired and wireless live packet data to monitor or record network traffic and translate to a human-readable format. PCAP captures bandwidth usage, resolution, and detects any suspicious activities. Information from layer 2 to 7 (everything except for the physical layer) can be detected by PCAP including source, destination, protocol used, payload.
Can you think of some example problems you could diagnose with a pcap trace?
When there are multiple devices talking to each other and sending packets over the network, PCAP trace could see if the packets reach the destination successfully and where the packets are dropped in case of an unsuccessful connection. Another use case would be tracking the malicious traffic to find areas impacted if there was a malware breach.
Can you think of problems for which pcap would not be appropriate?
PCAP would not be able to detect hardware-based attacks, therefore there should be other security measures in place. Placing the PCAP sniffer at the edge of the network would also compromise the visibility it has.

Wireshark


Capturing from wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…

```
No.   Time         Source           Destination      Protocol Length Info
14716 42.628602055 192.168.137.1    192.168.137.100  TCP        54 50660 → 3389 [ACK] Seq=6794 Ack=15577182 Win=513 Len=0
14717 42.628626962 192.168.137.100  192.168.137.1    TCP      1514 3389 → 50660 [ACK] Seq=15594702 Ack=6794 Win=501 Len=1460
14718 42.628632444 192.168.137.100  192.168.137.1    TCP      1514 3389 → 50660 [ACK] Seq=15596162 Ack=6794 Win=501 Len=1460
14719 42.628636907 192.168.137.100  192.168.137.1    TCP      1514 3389 → 50660 [PSH, ACK] Seq=15597622 Ack=6794 Win=501 Len=1460
14720 42.628799573 192.168.137.1    192.168.137.100  TCP        54 50660 → 3389 [ACK] Seq=6794 Ack=15584482 Win=513 Len=0
14721 42.628820684 192.168.137.100  192.168.137.1    TCP      1514 3389 → 50660 [ACK] Seq=15599082 Ack=6794 Win=501 Len=1460
14722 42.628826462 192.168.137.100  192.168.137.1    TCP      1185 3389 → 50660 [PSH, ACK] Seq=15600542 Ack=6794 Win=501 Len=1131
14723 42.628840313 192.168.137.1    192.168.137.100  TCP        54 50660 → 3389 [ACK] Seq=6794 Ack=15587402 Win=513 Len=0
14724 42.628966887 192.168.137.1    192.168.137.100  TCP        54 50660 → 3389 [ACK] Seq=6794 Ack=15593242 Win=513 Len=0
14725 42.630500751 192.168.137.1    192.168.137.100  TCP        54 50660 → 3389 [ACK] Seq=6794 Ack=15596162 Win=513 Len=0
14726 42.630594251 192.168.137.1    192.168.137.100  TCP        54 50660 → 3389 [ACK] Seq=6794 Ack=15600542 Win=513 Len=0
14727 42.630857601 192.168.137.1    192.168.137.100  TCP        54 50660 → 3389 [ACK] Seq=6794 Ack=15601673 Win=508 Len=0

Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
Ethernet II, Src: Raspberr_2c:33:76 (e4:5f:01:2c:33:76), Dst: 92:cc:df:6b:fe:04 (92:cc:df:6b:fe:04)
Internet Protocol Version 4, Src: 192.168.137.100, Dst: 192.168.137.1
Transmission Control Protocol, Src Port: 3389, Dst Port: 50660, Seq: 1, Ack: 1, Len: 1460
Data (1460 bytes)

0000  92 cc df 6b fe 04 e4 5f  01 2c 33 76 08 00 45 00   ···k···_ ·,3v··E·
0010  05 dc e6 27 40 00 40 06  bb 3d c0 a8 89 64 c0 a8   ···'@·@· ·=···d··
0020  89 01 0d 3d c5 e4 e3 9c  eb 00 c9 2a 8f 45 50 10   ···=···· ···*·EP·
0030  01 f5 18 c6 00 00 80 a9  ed 58 20 65 dc cb 69 54   ········ ·X e··iT
0040  65 99 f4 ec f2 a6 c3 b7  01 1a 40 73 a6 b8 de 9a   e······· ··@s····
0050  99 0a 07 af 01 bb 9f 86  bb 1f 0e 25 25 f2 ce 1d   ········ ···%%···
0060  b8 a6 1e fe 56 5d 4f 16  74 c8 17 fc 24 8e 44 5a   ····V]O· t···$·DZ
0070  57 09 a7 06 b0 19 85 7b  b1 8e dc 52 70 07 fb 96   W······{ ···Rp···
0080  b1 9a bf b4 f5 e4 59 6b  a9 38 48 97 36 29 8c 0d   ······Yk ·8H·6)··
0090  de 08 88 f3 ab 2d 99 34  b4 53 b6 7b 88 d2 82 66   ·····-·4 ·S·{···f
00a0  c9 11 cd 48 c8 ef 09 73  24 ec 48 64 73 ea b5 0e   ···H···s $·Hds···

wlan0: <live capture in progress>    Packets: 14727 · Displayed: 14727 (100.0%)    Profile: Default
```

Figure 3. Wireshark interface showing packets captured from wlan0

What is Wireshark? How is it related to or different from pcap? What are some situations you might want to use Wireshark?
Wireshark is an open source tool implemented to use PCAP that runs on most platforms. Wiresharks provide a well rounded overview and essential details on the network. Wireshark is great for a general survey, debugging, or network monitoring without the need to look into any specialized areas. It is also great for learning about networking.

(set up Wireshark) What do you notice? What do you see in the contents of packet headers, is it what you expected? Also do you see any packets you don't expect?
Opening up Wireshark, traffic between the router and the pi through TCP was observed. It captures messages with ACK tags, which was expected. There were a couple ACK PSH tags that were unexpected. PSH (push) tags are used to tell the client/server to send the data it has to the receiving end immediately even when the buffer is not filled up.

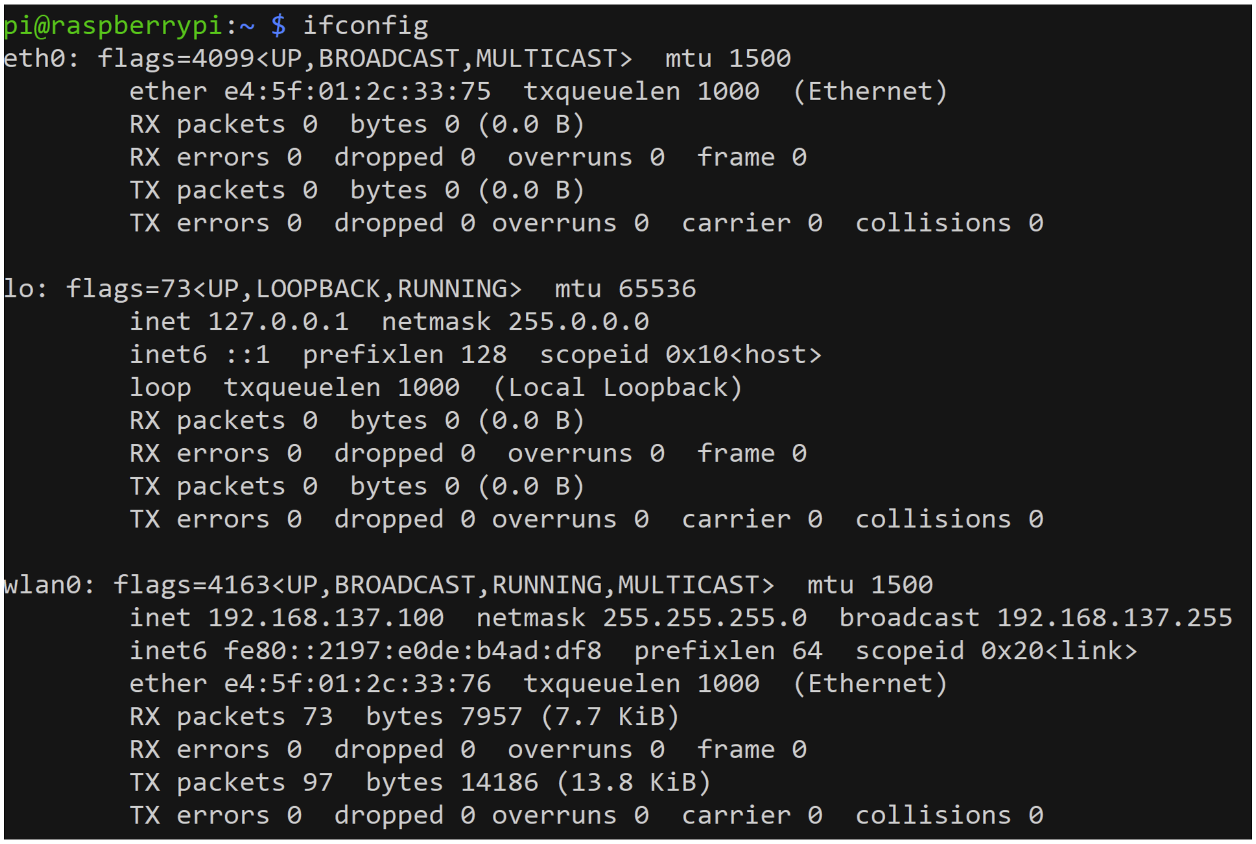
Think about how to use packet inspection to deal with various scenarios. If you believed you were under attack, and an adversary had infiltrated your system, how would you use Wireshark to figure out what happened?
Wireshark can trace through the packets and find the sources and destination the attack packets has been through. This can help identify the areas infected and how to contain the spread and minimize damage. Looking through the sources and destinations, we can find out when, where and how (through what protocol) the system was infiltrated.

Findings and Applications

Wireshark is a very versatile and well-rounded tool great for anyone learning or working with networking. While there are may functionalities available, I imagine it could not only be applied to tracing packets to check that the system is functional, but also finding the loopholes in the systems before any attacks could be launched.

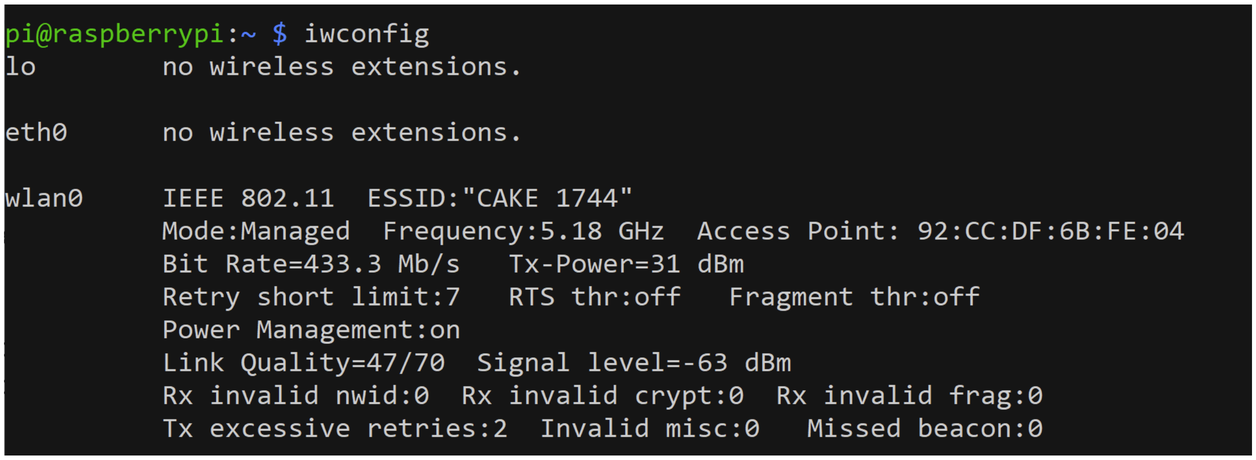
Wireless Configuration

- [ ] Read https://www.geeksforgeeks.org/iwconfig-command-in-linux-with-examples/ and https://linux.die.net/man/8/iwconfig



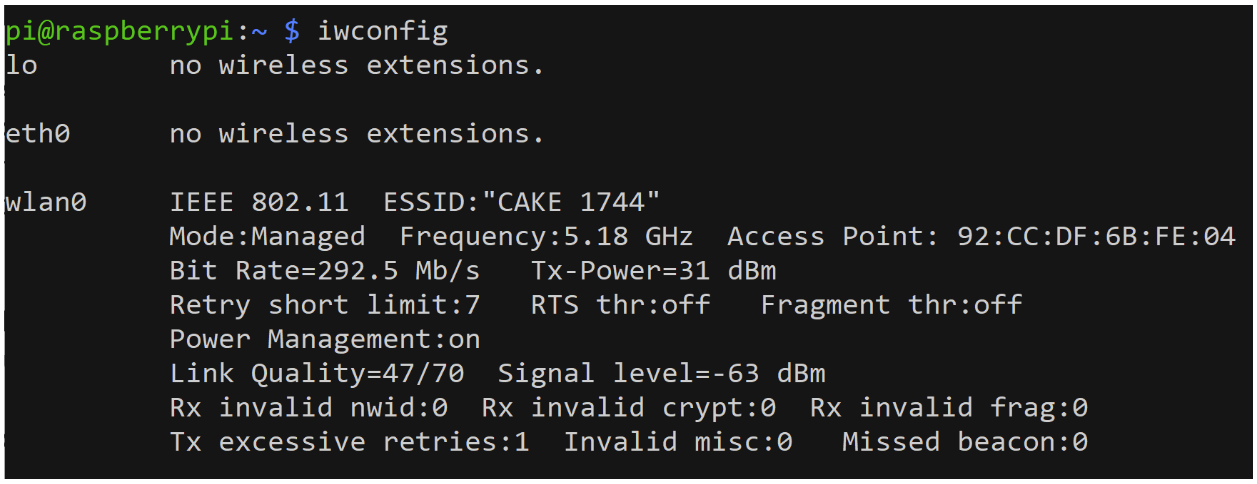
```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether e4:5f:01:2c:33:75  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.137.100  netmask 255.255.255.0  broadcast 192.168.137.255
        inet6 fe80::2197:e0de:b4ad:df8  prefixlen 64  scopeid 0x20<link>
        ether e4:5f:01:2c:33:76  txqueuelen 1000  (Ethernet)
        RX packets 73  bytes 7957 (7.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 97  bytes 14186 (13.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4. Running `ifconfig`



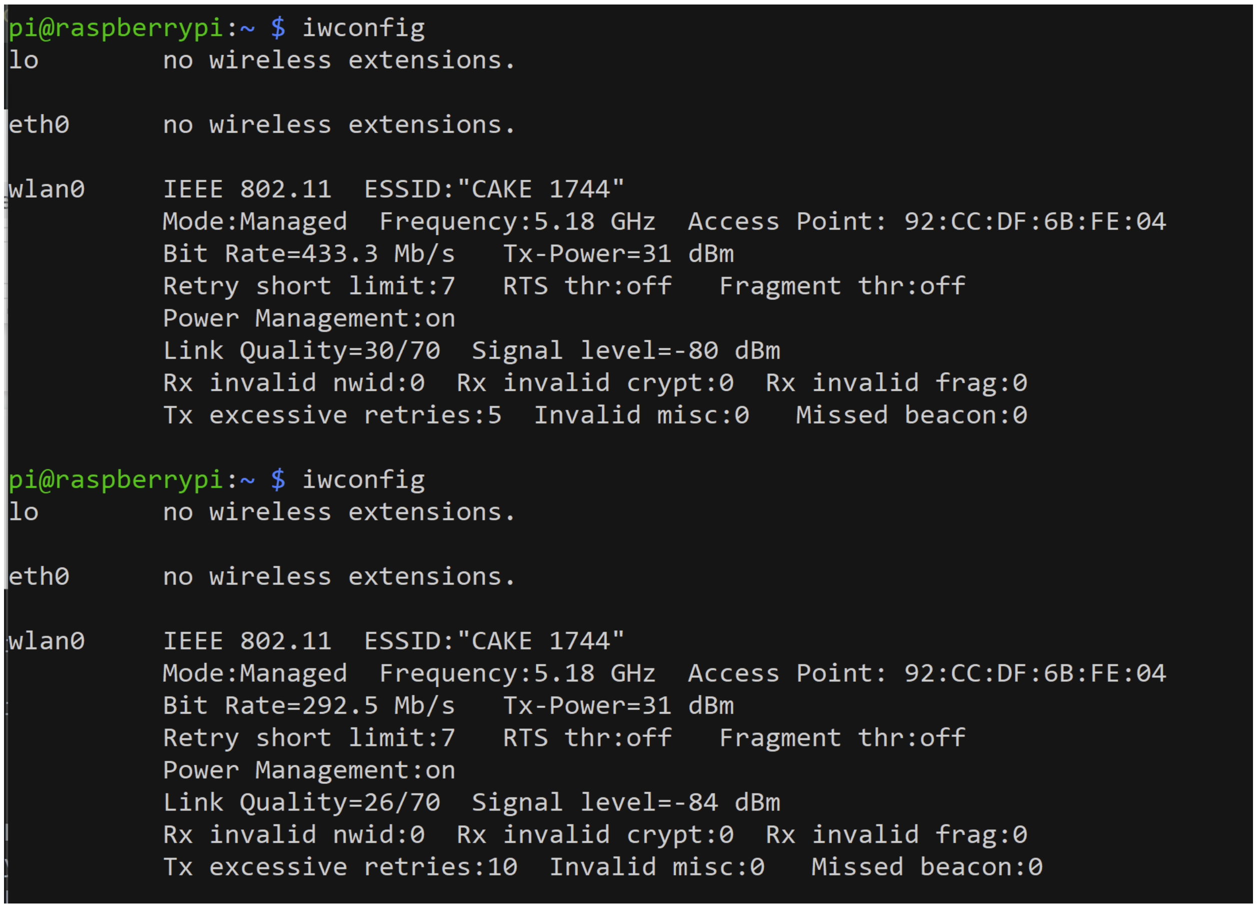
```
pi@raspberrypi:~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"CAKE 1744"
          Mode:Managed  Frequency:5.18 GHz  Access Point: 92:CC:DF:6B:FE:04
          Bit Rate=433.3 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=47/70  Signal level=-63 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:2  Invalid misc:0   Missed beacon:0
```

Figure 5. Running `iwconfig` right next to the router



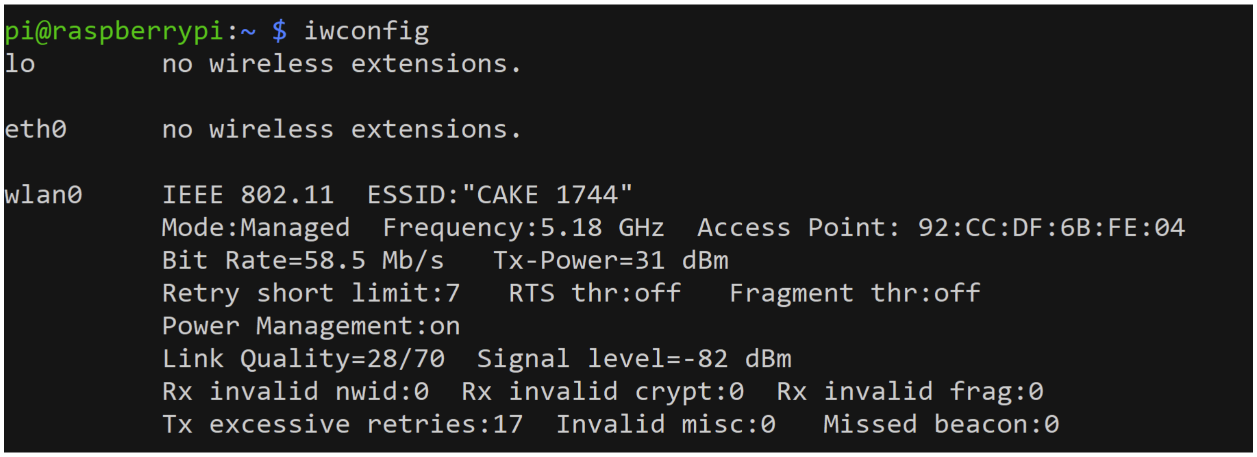
```
pi@raspberrypi:~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"CAKE 1744"
          Mode:Managed  Frequency:5.18 GHz  Access Point: 92:CC:DF:6B:FE:04
          Bit Rate=292.5 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=47/70  Signal level=-63 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:1  Invalid misc:0   Missed beacon:0
```

Figure 6. Running `iwconfig` in the room farthest from the router. Note the bit rate lowered ~ 140 Mb/s



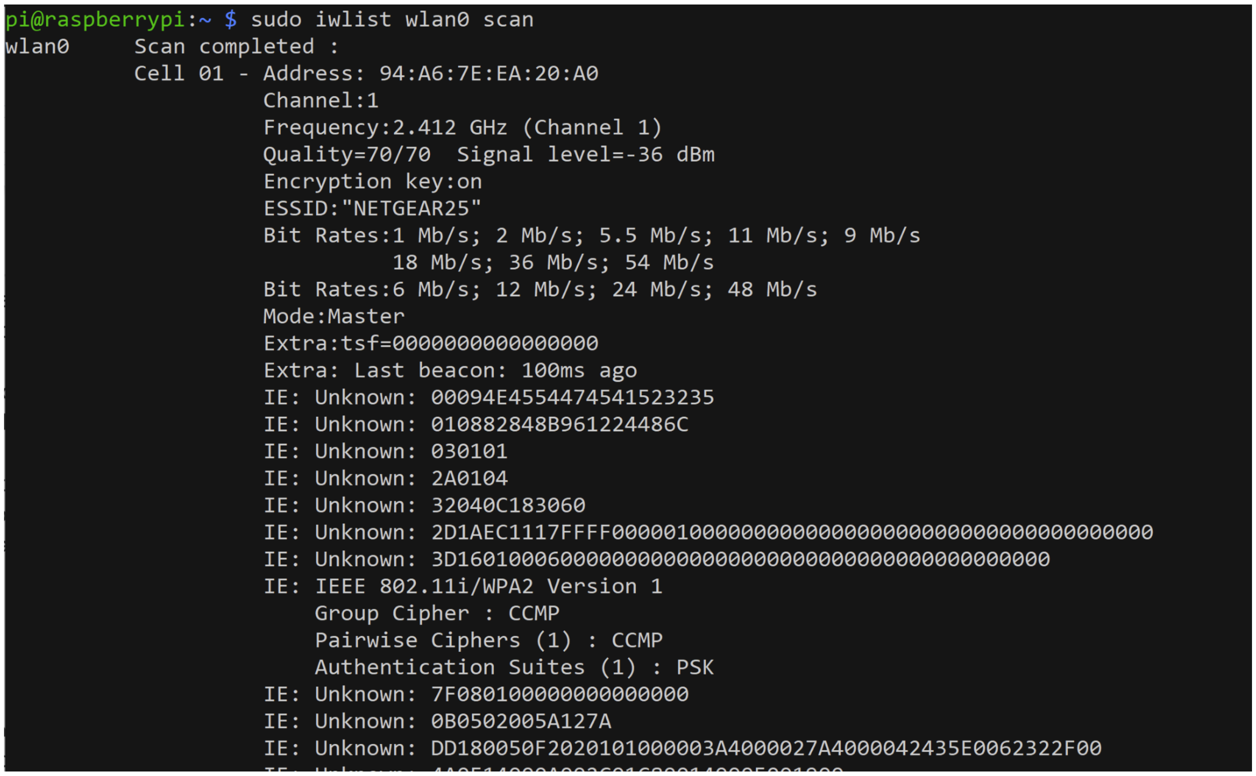
```
pi@raspberrypi:~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"CAKE 1744"
          Mode:Managed  Frequency:5.18 GHz  Access Point: 92:CC:DF:6B:FE:04
          Bit Rate=433.3 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=30/70  Signal level=-80 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:5  Invalid misc:0   Missed beacon:0

pi@raspberrypi:~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"CAKE 1744"
          Mode:Managed  Frequency:5.18 GHz  Access Point: 92:CC:DF:6B:FE:04
          Bit Rate=292.5 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=26/70  Signal level=-84 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:10  Invalid misc:0   Missed beacon:0
```

Figure 7. Running `iwconfig` outside the fridge vs inside



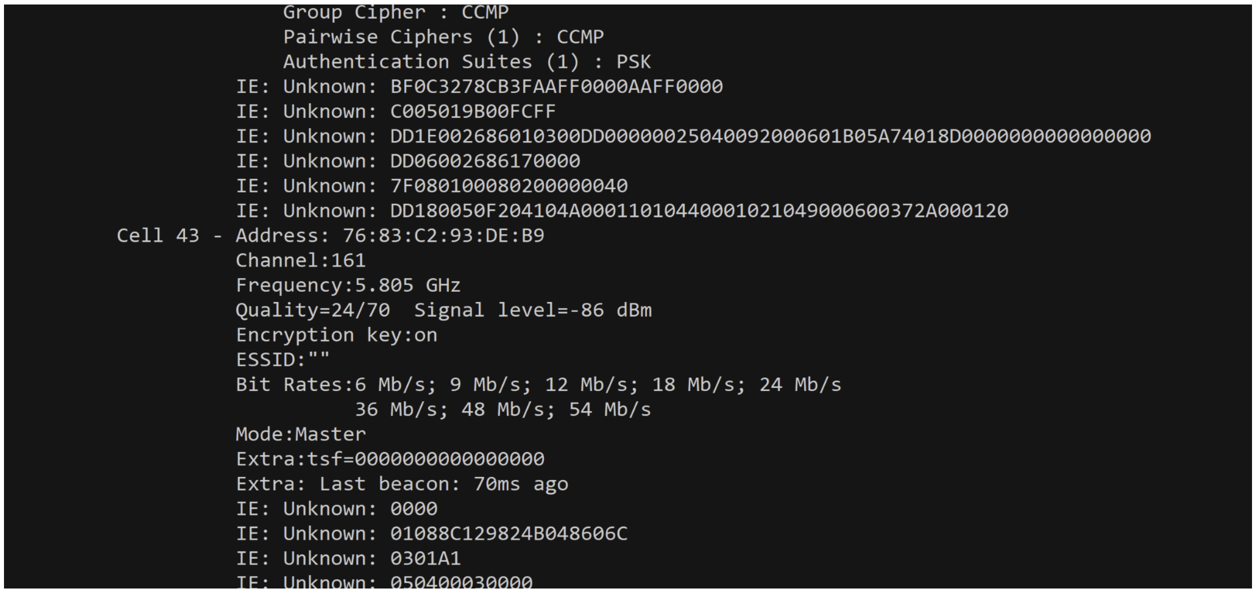
```
pi@raspberrypi:~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"CAKE 1744"
          Mode:Managed  Frequency:5.18 GHz  Access Point: 92:CC:DF:6B:FE:04
          Bit Rate=58.5 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=28/70  Signal level=-82 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:17  Invalid misc:0   Missed beacon:0
```

Figure 8. Running `iwconfig` after the car has been in the fridge for a little longer. The bit rate decreased significantly



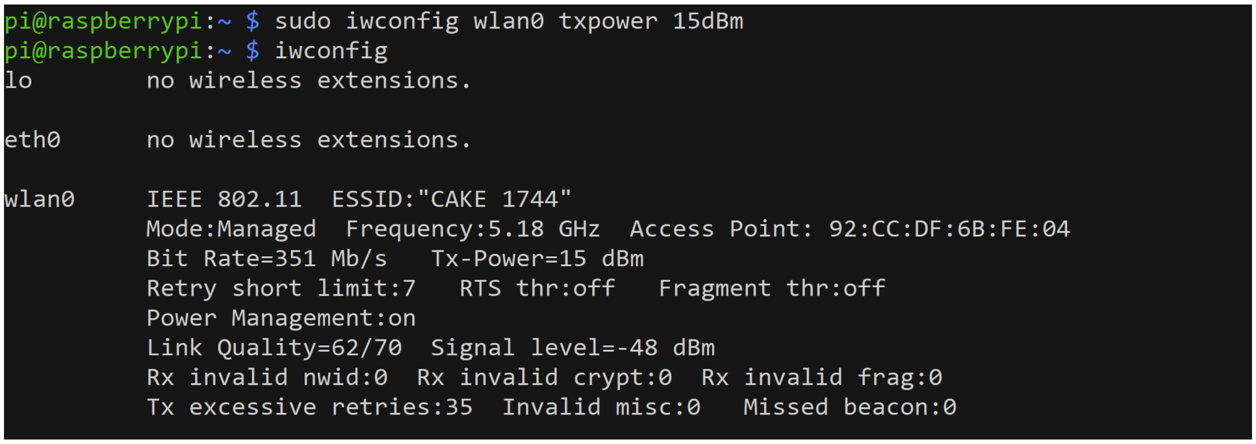
```
pi@raspberrypi:~ $ sudo iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 94:A6:7E:EA:20:A0
                    Channel:1
                    Frequency:2.412 GHz (Channel 1)
                    Quality=70/70  Signal level=-36 dBm
                    Encryption key:on
                    ESSID:"NETGEAR25"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 9 Mb/s
                              18 Mb/s; 36 Mb/s; 54 Mb/s
                    Bit Rates:6 Mb/s; 12 Mb/s; 24 Mb/s; 48 Mb/s
                    Mode:Master
                    Extra:tsf=0000000000000000
                    Extra: Last beacon: 100ms ago
                    IE: Unknown: 00094E4554474541523235
                    IE: Unknown: 010882848B961224486C
                    IE: Unknown: 030101
                    IE: Unknown: 2A0104
                    IE: Unknown: 32040C183060
                    IE: Unknown: 2D1AEC1117FFFF000001000000000000000000000000000000000000
                    IE: Unknown: 3D160100060000000000000000000000000000000000000000
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 7F080100000000000000
                    IE: Unknown: 0B0502005A127A
                    IE: Unknown: DD180050F2020101000003A4000027A4000042435E0062322F00
```

Figure 9. Running `sudo iwlist wlan0 scan` (a long list of access points and signal strength was observed)



```
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: BF0C3278CB3FAAFF0000AAFF0000
                    IE: Unknown: C005019B00FCFF
                    IE: Unknown: DD1E00268601030 0DD000000250400920006 01B05A74018D0000000000000000
                    IE: Unknown: DD06002686170000
                    IE: Unknown: 7F080100080200000040
                    IE: Unknown: DD180050F204104A000110104400010210490006 00372A000120
          Cell 43 - Address: 76:83:C2:93:DE:B9
                    Channel:161
                    Frequency:5.805 GHz
                    Quality=24/70  Signal level=-86 dBm
                    Encryption key:on
                    ESSID:""
                    Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
                              36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=0000000000000000
                    Extra: Last beacon: 70ms ago
                    IE: Unknown: 0000
                    IE: Unknown: 01088C129824B048606C
                    IE: Unknown: 0301A1
                    IE: Unknown: 050400030000
```

Figure 10. Running `sudo iwlist wlan0 scan` showed 43 access points



```
pi@raspberrypi:~ $ sudo iwconfig wlan0 txpower 15dBm
pi@raspberrypi:~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"CAKE 1744"
          Mode:Managed  Frequency:5.18 GHz  Access Point: 92:CC:DF:6B:FE:04
          Bit Rate=351 Mb/s   Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=62/70  Signal level=-48 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:35  Invalid misc:0   Missed beacon:0
```

Figure 11. Running `sudo iwconfig wlan0 txpower 15dBm` to reduce the Tx-power default setting by half

while true; do sleep 0.5; iwconfig wlan0 | grep "Link Quality"; done

iwconfig

What properties would you inspect?
The bit rate, tx-power, and link quality were inspected.

Take a look at the link quality using the Link Quality metric - walk around with your car and watch that change. What happens when you get closer to the access point? Can you discover any "dead zones" in your house where you have poor coverage?
The link quality did not change significantly being right next to the router or to the farthest end of the apartment. However, the bit rate decreased a little bit. A dead zone discovered was inside

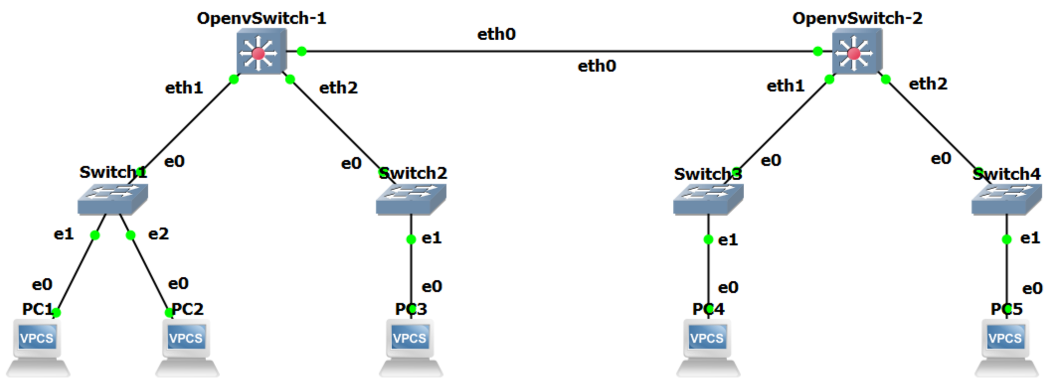
the fridge. Due to being enclosed in layers of metal, plastic, and some food (maybe being cold too), the link quality and bit rate dropped significantly compared to outside the fridge.

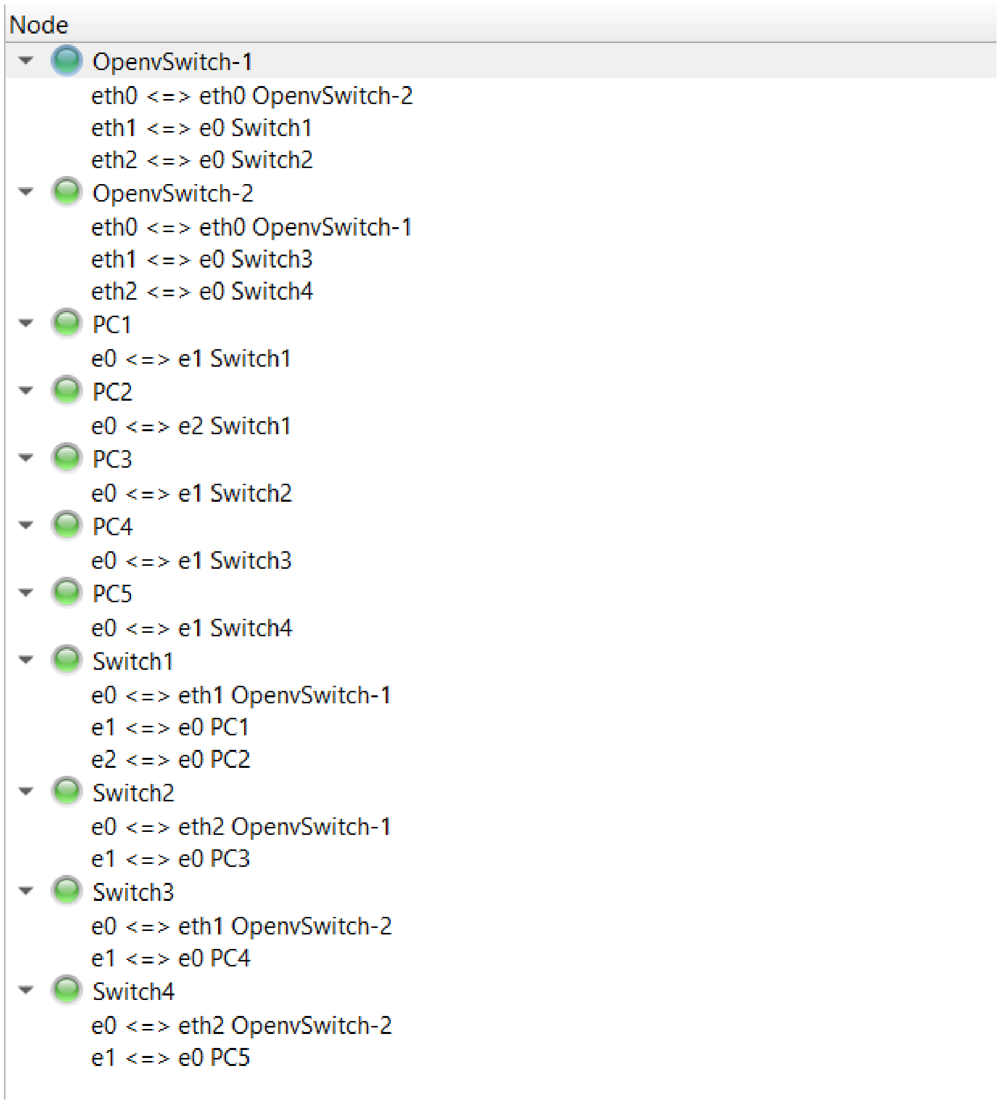
Findings and Applications

It was interesting to see that a dead zone does not necessarily mean that it is too far away from the router for the transmit power to reach, but also its environment. This can be applied to designing the enclosure of a component. The choice of material needs to be considered for its electrical properties to not hinder the electromagnetic waves being received by the antenna.

The car network surveyor was also a great learning point on how easily we can see the other access points around us - and that we need a strong password and implement security measures if we are working on anything we wouldn’t like other people to see on the network.

Router Configuration





```
Node
OpenvSwitch-1
    eth0 <=> eth0 OpenvSwitch-2
    eth1 <=> e0 Switch1
    eth2 <=> e0 Switch2
OpenvSwitch-2
    eth0 <=> eth0 OpenvSwitch-1
    eth1 <=> e0 Switch3
    eth2 <=> e0 Switch4
PC1
    e0 <=> e1 Switch1
PC2
    e0 <=> e2 Switch1
PC3
    e0 <=> e1 Switch2
PC4
    e0 <=> e1 Switch3
PC5
    e0 <=> e1 Switch4
Switch1
    e0 <=> eth1 OpenvSwitch-1
    e1 <=> e0 PC1
    e2 <=> e0 PC2
Switch2
    e0 <=> eth2 OpenvSwitch-1
    e1 <=> e0 PC3
Switch3
    e0 <=> eth1 OpenvSwitch-2
    e1 <=> e0 PC4
Switch4
    e0 <=> eth2 OpenvSwitch-2
    e1 <=> e0 PC5
```

Overview:

Our setup consists of the following components:

Component Type	Components
Virtual PC (via VPCS)	PC1, PC2, PC3, PC4, PC5
Level 2 switch	Switch1, Switch2, Switch3, Switch4
Router/Level 3 switch	OpenvSwitch-1, OpenvSwitch-2

For our Router/Level 3 switch appliance, we used Open vSwitch, as suggested on Campuswire (see #404). We assigned each one to a shared subnet via the eth0 interface, setting the other switch as a gateway to allow traffic to find its way across router subnets.

For each Virtual PC, we set the gateway to the relevant OpenvSwitch interface.

Breakdown:

Component	Type	Connection/Interfacing
PC1	Virtual PC	IP: 10.0.0.2/25 Gateway: 10.0.0.1
PC2	Virtual PC	IP: 10.0.0.3/25 Gateway: 10.0.0.1
PC3	Virtual PC	IP: 10.0.0.130/25 Gateway: 10.0.0.129
PC4	Virtual PC	IP: 10.2.2.2/25 Gateway: 10.2.2.1
PC5	Virtual PC	IP: 10.2.2.130/25 Gateway: 10.2.2.129
Switch1	Level 2 switch	OpenvSwitch-1 connects to this device via its interface eth1
Switch2	Level 2 switch	OpenvSwitch-1 connects to this device via its interface eth2
Switch3	Level 2 switch	OpenvSwitch-2 connects to this device via its interface eth1

Switch4	Level 2 switch	OpenvSwitch-1 connects to this device via its interface eth2
OpenvSwitch-1	Router/Level 3 switch	Interface eth0: ● IP: 10.1.1.1/24 ● Gateway: 10.1.1.2 Interface eth1: ● IP: 10.0.0.1/24 Interface eth2: ● IP: 10.0.0.129/24
OpenvSwitch-1	Router/Level 3 switch	Interface eth0: ● IP: 10.1.1.1/24 ● Gateway: 10.1.1.2 Interface eth1: ● IP: 10.0.0.1/24 Interface eth2: ● IP: 10.0.0.129/24

Instructions:

Setup:

1. Create 5 virtual PCs using the “VPCS” template.
    a. Name them PC1, PC2, PC3, PC4, and PC5
2. Create 4 Ethernet switches using the “Ethernet Switch” template.
    a. Name them Switch1, Switch2, Switch3, and Switch4
3. Add the Open vSwitch template (if not already present)
    a. In the menu bar, select File > New Template
    b. Select “Install an appliance from the GNS3 server (recommended)”, and click Next
    c. Expand the “Switches” category, select the appliance named “Open vSwitch”, and click Next
    d. Select “Install the appliance on the GNS3 VM (recommended)”, and click Next
    e. Click Finish
4. Create 2 Open vSwitch instances using the “Open vSwitch” template.
    a. Name them OpenvSwitch-1, and OpenvSwitch-2

Configuration:

1. Configure OpenvSwitch-1
    a. Right click on the object > Configure
    b. In the General Settings tab, set Adapters to 3

c. Click the Edit button for Network Configuration, and enter the following:
```
# Static config for eth0
auto eth0
iface eth0 inet static
      address 10.1.1.1
      netmask 255.255.255.0
      gateway 10.1.1.2

# Static config for eth1
auto eth1
iface eth1 inet static
      address 10.0.0.1
      netmask 255.255.255.128

# Static config for eth2
auto eth2
iface eth2 inet static
      address 10.0.0.129
      netmask 255.255.255.128
```
d. Click Save
e. Click Apply

2. Configure OpenvSwitch-2
   a. Right click on the object > Configure
   b. In the General Settings tab, set Adapters to 3
   c. Click the Edit button for Network Configuration, and enter the following in the popup window:
```
# Static config for eth0
auto eth0
iface eth0 inet static
      address 10.1.1.2
      netmask 255.255.255.0
      gateway 10.1.1.1

# Static config for eth1
auto eth1
iface eth1 inet static
      address 10.2.2.1
      netmask 255.255.255.128

# Static config for eth2
auto eth2
iface eth2 inet static
      address 10.2.2.129
      netmask 255.255.255.128
```

   d. Click Save
   e. Click Apply
3. Configure PC1
   a. Right click on object > Edit Config
   b. Enter the following in the popup window:
      ```
      set pcname PC1
      ip 10.0.0.2/25 10.0.0.1
      ```
   c. Click Save
4. Configure PC2
   a. Right click on object > Edit Config
   b. Enter the following in the popup window:
      ```
      set pcname PC2
      ip 10.0.0.3/25 10.0.0.1
      ```
   c. Click Save
5. Configure PC3
   a. Right click on object > Edit Config
   b. Enter the following in the popup window:
      ```
      set pcname PC3
      ip 10.0.0.130/25 10.0.0.129
      ```
   c. Click Save
6. Configure PC4
   a. Right click on object > Edit Config
   b. Enter the following in the popup window:
      ```
      set pcname PC4
      ip 10.2.2.2/25 10.2.2.1
      ```
   c. Click Save
7. Configure PC
   a. Right click on object > Edit Config
   b. Enter the following in the popup window:
      ```
      set pcname PC5
      ip 10.2.2.130/25 10.2.2.129
      ```
   c. Click Save

Connection:

1. Connect OpenvSwitch-1 (eth0) to OpenvSwitch-2 (eth0)
2. Connect OpenvSwitch-1 (eth1) to Switch1 (e0)
3. Connect OpenvSwitch-1 (eth2) to Switch2 (e0)
4. Connect OpenvSwitch-2 (eth1) to Switch3 (e0)
5. Connect OpenvSwitch-2 (eth2) to Switch4 (e0)
6. Connect Switch1 (e1) to PC1 (e0)
7. Connect Switch1 (e2) to PC2 (e0)
8. Connect Switch2 (e1) to PC3 (e0)
9. Connect Switch3 (e1) to PC4 (e0)
10. Connect Switch4 (e1) to PC5 (e0)

Contributions

Group Member	**Contribution**
Nishant Sheikh	Router Configuration, report
Molly Yang	Firewall, Packet Inspection, Wireless Configuration, report