Exercise 4

Running a P4 program

In this exercise we look at how a P4 program responds to packets sent to the Raspberry Pi.

Firstly, let's look at the Python code for sending packets:

send.py
~/CWM-ProgNets/assignment4

```python
#!/usr/bin/python

from scapy.all import Ether, IP, sendp, get_if_hwaddr, get_if_list, TCP, Raw, UDP
import sys
import random, string


def randomword(length):
    return ''.join(random.choice(string.ascii_lowercase) for i in range(length))

def send_random_traffic(num_packets, interface, src_ip, dst_ip):
    dst_mac = "BB:BB:BB:BB:BB:BB"
    src_mac= "AA:AA:AA:AA:AA:AA"
    total_pkts = 0
    port = 1024
    for i in range(num_packets):
            data = randomword(22)
            p = Ether(dst=dst_mac,src=src_mac)/IP(dst=dst_ip,src=src_ip)
            p = p/UDP(sport= 50000, dport=port)/Raw(load=data)
            sendp(p, iface = interface, inter = 0.01)
            # If you want to see the contents of the packet, uncomment the line below
            # print(p.show())
            total_pkts += 1
    print("Sent %s packets in total" % total_pkts)

if __name__ == '__main__':
    if len(sys.argv) < 5:
        print("Usage: python send.py number_of_packets interface_name src_ip_address dst_ip_address")
        sys.exit(1)
    else:
        num_packets = sys.argv[1]
        interface = sys.argv[2]
        src_ip = sys.argv[3]
        dst_ip = sys.argv[4]
        send_random_traffic(int(num_packets), interface, src_ip, dst_ip)
```

Python 2 Tab Width: 8 Ln 11, Col 65 INS

Python code

We send packets with source MAC address AA:AA:AA:AA:AA:AA and destination address BB:BB:BB:BB:BB:BB, using the UDP protocol.

Below is the relevant section of the P4 program code:

```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {

    action swap_mac_addresses() {
       macAddr_t tmp_mac;
       tmp_mac = hdr.ethernet.dstAddr;
       hdr.ethernet.dstAddr = hdr.ethernet.srcAddr;
       hdr.ethernet.srcAddr = tmp_mac;

       //send it back to the same port
       standard_metadata.egress_spec = standard_metadata.ingress_port;
    }

    action drop() {
        mark_to_drop(standard_metadata);
    }

    table src_mac_drop {
        key = {
            hdr.ethernet.srcAddr: exact;
        }
        actions = {
            swap_mac_addresses;
            drop;
            NoAction;
        }
        size = 1024;
        default_action = swap_mac_addresses();
    }

    apply {
        if (hdr.ethernet.isValid()) {
            src_mac_drop.apply();
        }
    }
}
```

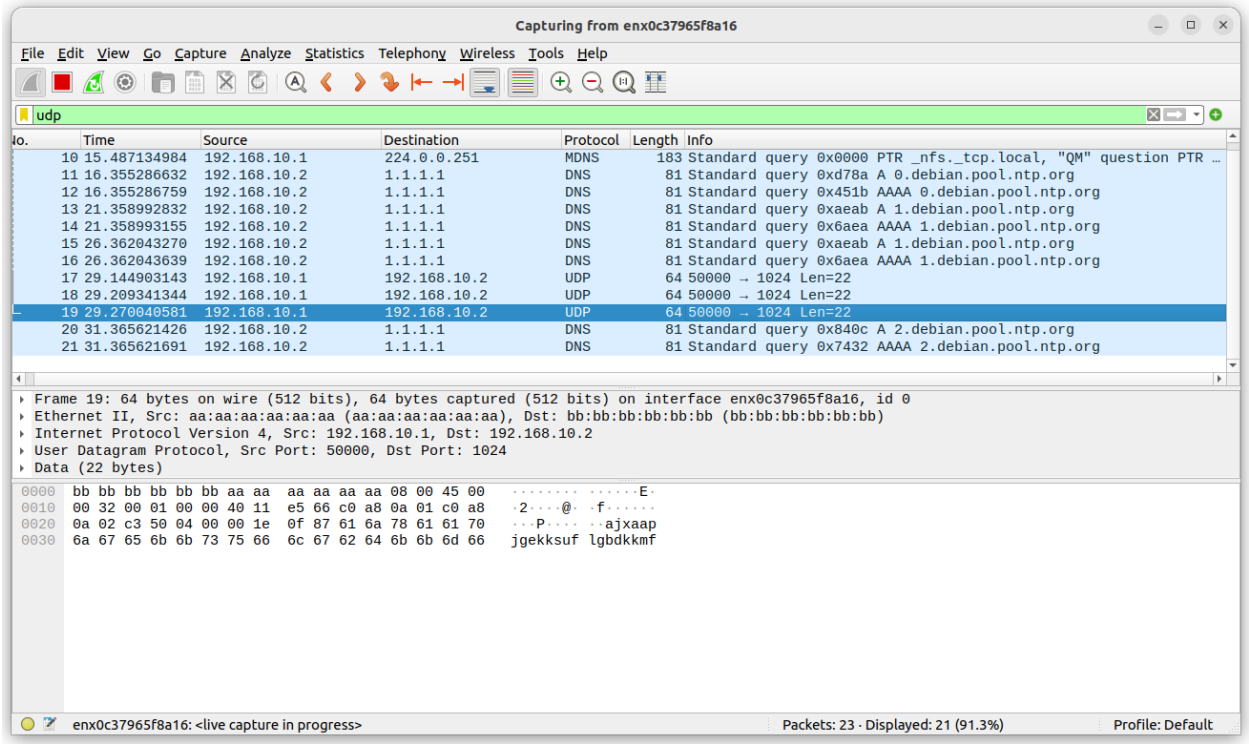
Ingress processing in the P4 code

We note that the default action on each packet is to swap the MAC addresses and return the packet.

We can use the following command in the terminal to send 3 packets:

sudo python3 send.py 3 enx0c37965f8a16 192.168.10.1 192.168.10.2

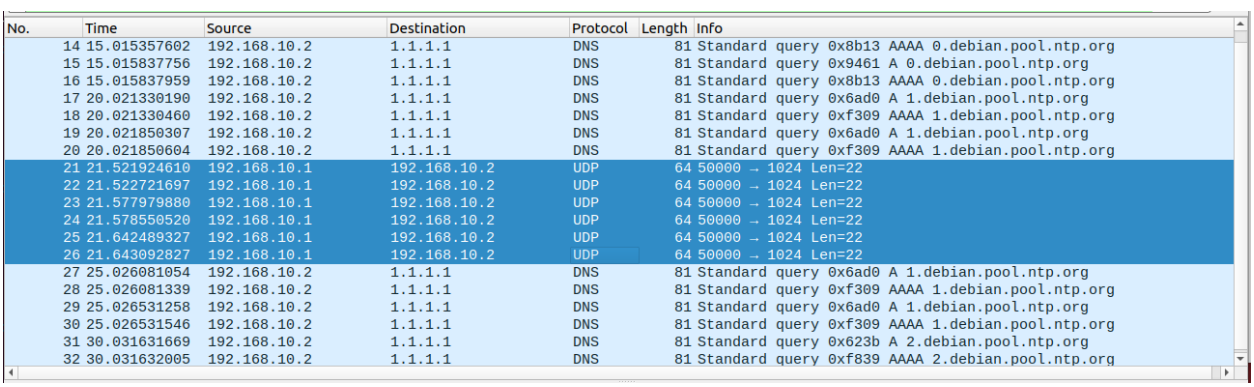
Let's see what happens when the P4 program isn't running:



View of WireShark, the tool with which we will inspect packets

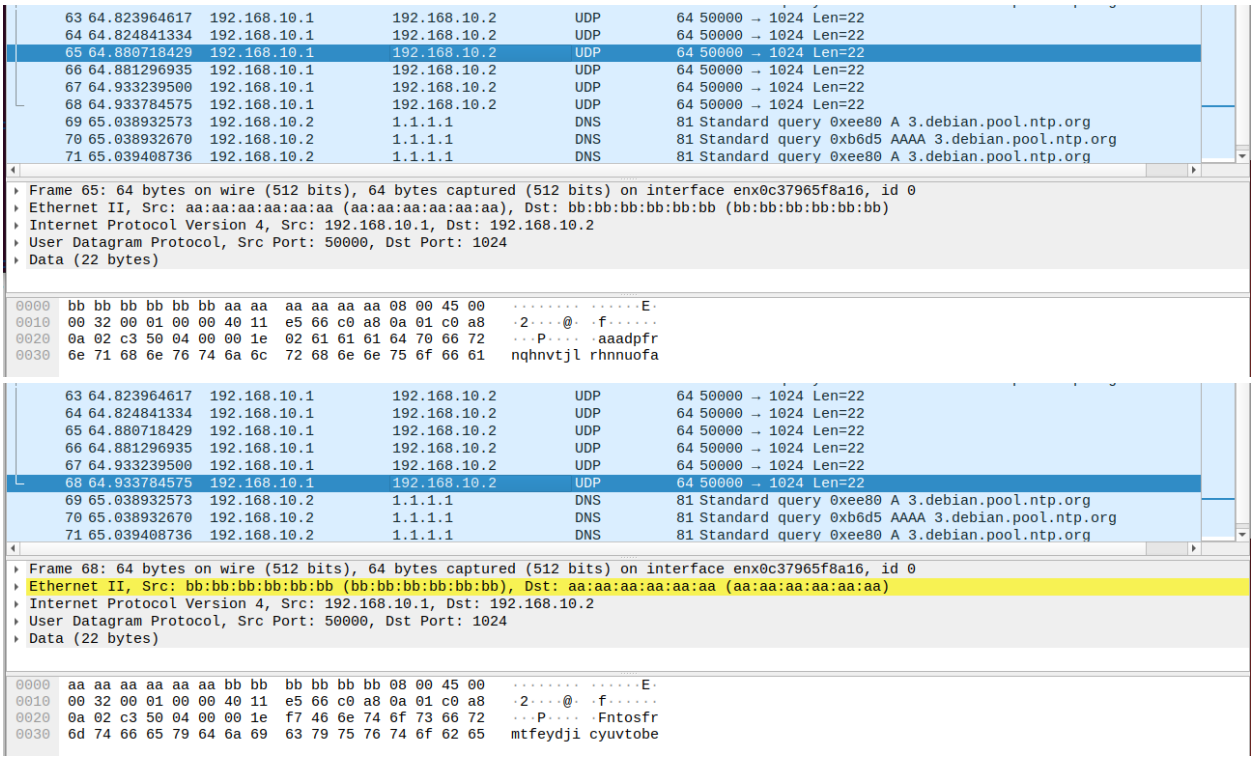
We see that the packet's header does indeed match what we said in the Python program.

Now let's run the compiled P4 program on the Raspberry Pi. Sending 3 packets, we get 3 back to make a total of 6, so the program is working:



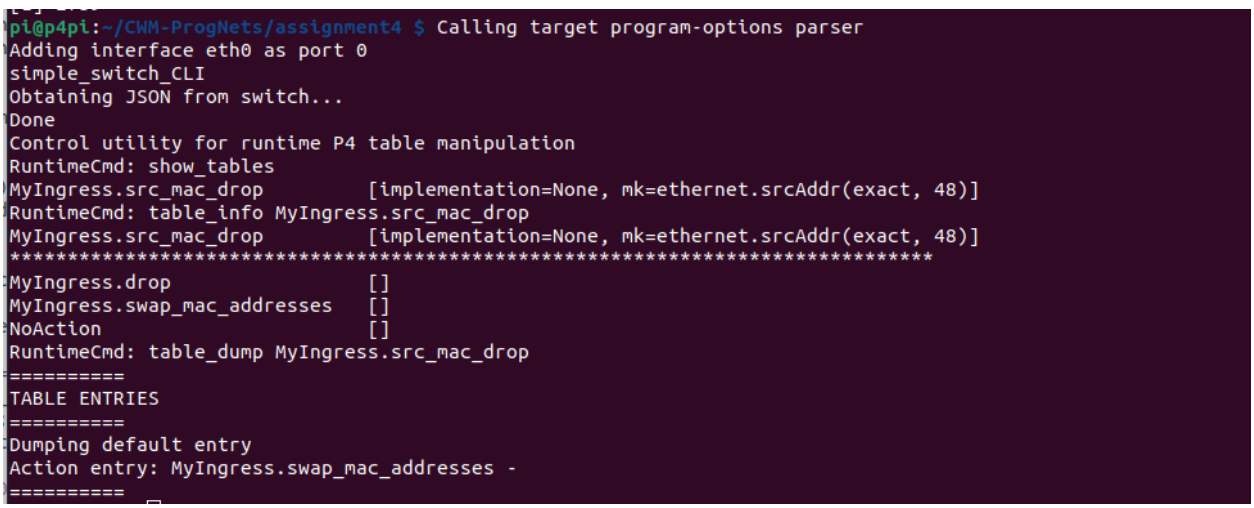
UDP packets highlighted here

We see that the MAC addresses are swapped between the two as well:



Note that we can also see the change in the top row of the packet entries

Switching to the terminal of the Raspberry Pi, we now wish to add a case to the table on the Raspberry Pi. First, let's look at its initial state:



View of SSH terminal. We inspect the table using table_dump <table name>

If we now use the following command:

table_add MyIngress.src_mac_drop MyIngress.drop AA:AA:AA:AA:AA:AA =>

then the table now has an entry with handle 0 to drop packets with source MAC address AA:AA:AA:AA:AA:AA, seen here:

```
RuntimeCmd: table_add MyIngress.src_mac_drop MyIngress.drop AA:AA:AA:AA:AA:AA =>
Adding entry to exact match table MyIngress.src_mac_drop
match key:           EXACT-aa:aa:aa:aa:aa:aa
action:              MyIngress.drop
runtime data:
Entry has been added with handle 0
RuntimeCmd: table_dump MyIngress.src_mac_drop
==========
TABLE ENTRIES
**********
Dumping entry 0x0
Match key:
* ethernet.srcAddr    : EXACT     aaaaaaaaaaaa
Action entry: MyIngress.drop -
==========
Dumping default entry
Action entry: MyIngress.swap_mac_addresses -
==========
```

View after the entry has been added

Our outgoing packets had that address, so we expect them to be dropped. Indeed, we see that we now only see the outgoing packets on WireShark:

16	15.008846247	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x59c6 AAAA 2.debian.pool.ntp.org
17	17.448167688	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
18	17.529354128	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
19	17.593610735	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
20	20.011341370	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x8847 A 3.debian.pool.ntp.org
21	20.011341654	192.168.10.2	1.1.1.1	DNS	81	Standard query 0xa08c AAAA 3.debian.pool.ntp.org
22	20.011889262	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x8847 A 3.debian.pool.ntp.org

```
▸ Frame 17: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface enx0c37965f8a16, id 0
▸ Ethernet II, Src: aa:aa:aa:aa:aa:aa (aa:aa:aa:aa:aa:aa), Dst: bb:bb:bb:bb:bb:bb (bb:bb:bb:bb:bb:bb)
▸ Internet Protocol Version 4, Src: 192.168.10.1, Dst: 192.168.10.2
▸ User Datagram Protocol, Src Port: 50000, Dst Port: 1024
▸ Data (22 bytes)

0000  bb bb bb bb bb bb aa aa  aa aa aa aa 08 00 45 00   ........ ......E·
0010  00 32 00 01 00 00 40 11  e5 66 c0 a8 0a 01 c0 a8   ·2····@· ·f......
0020  0a 02 c3 50 04 00 00 1e  e6 65 72 6f 77 64 6f 6b   ···P···· ·erowdok
0030  78 75 70 76 65 6a 6c 6c  6b 69 6f 6a 63 70 6a 61   xupvejll kiojcpja
```

Only 3 packets show up on WireShark now

Now if we go back to the Python program and change the source MAC address to CC:CC:CC:CC:CC:CC, we get reflections again:

No.	Time	Source	Destination	Protocol	Length	Info
183	306.584762986	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x5a5a AAAA 1.debian.pool.ntp.org
184	311.589793125	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x958e A 2.debian.pool.ntp.org
185	311.589793432	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x6c90 AAAA 2.debian.pool.ntp.org
186	311.590339890	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x958e A 2.debian.pool.ntp.org
187	311.590340176	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x6c90 AAAA 2.debian.pool.ntp.org
192	312.492246248	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
193	312.493073425	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
194	312.544558430	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
195	312.545219569	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
196	312.625366459	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
197	312.625964613	192.168.10.1	192.168.10.2	UDP	64	50000 → 1024 Len=22
198	316.594804050	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x958e A 2.debian.pool.ntp.org
199	316.594804320	192.168.10.2	1.1.1.1	DNS	81	Standard query 0x6c90 AAAA 2.debian.pool.ntp.org

```
▸ Frame 195: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface enx0c37965f8a16, id 0
▸ Ethernet II, Src: bb:bb:bb:bb:bb:bb (bb:bb:bb:bb:bb:bb), Dst: SiliconL_cc:cc:cc (cc:cc:cc:cc:cc:cc)
▸ Internet Protocol Version 4, Src: 192.168.10.1, Dst: 192.168.10.2
▸ User Datagram Protocol, Src Port: 50000, Dst Port: 1024
▸ Data (22 bytes)

0000  cc cc cc cc cc cc bb bb  bb bb bb bb 08 00 45 00   ........ ......E·
0010  00 32 00 01 00 00 40 11  e5 66 c0 a8 0a 01 c0 a8   ·2····@· ·f......
0020  0a 02 c3 50 04 00 00 1e  f2 68 74 68 77 68 6b 6f   ···P···· ·hthwhko
0030  6f 70 69 71 65 68 72 62  62 71 62 68 75 72 6e 6b   opiqehrb bqbhurnk
```

3 packets sent, 3 packets return again

```
8 def randomword(length):
9     return ''.join(random.choice(string.ascii_lowercase) for i in range(length))
10
11 def send_random_traffic(num_packets, interface, src_ip, dst_ip):
12     dst_mac = "BB:BB:BB:BB:BB:BB"
13     src_mac= "CC:CC:CC:CC:CC:CC"
14     total_pkts = 0
15     port = 1024
16     for i in range(num_packets):
17             data = randomword(22)
```

Edited Python code: the change in line 13 is highlighted