CS4150: Computer Networks Lab

Lab10

111901030
Mayank Singla

Q1. You are given the virtual network in Fig. 1. This network has two subnets, and one node common to both these subnets. Your first exercise is to write a simple packet sniffer at node r1. This packet sniffer should print the MAC addresses (source and destination), and IP addresses (source and destination) of all IP packets sent and received by r1.
Refer to raw-sockets-c-code-linux and using-raw-sockets

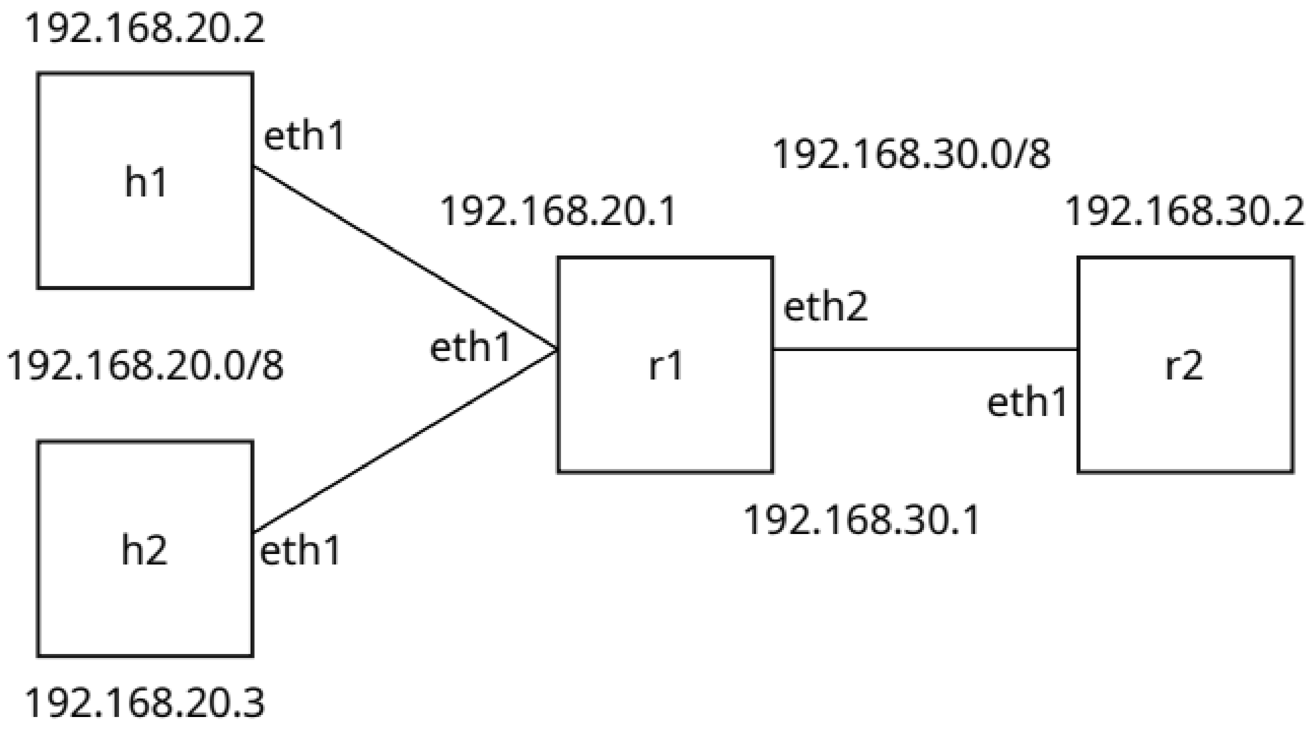
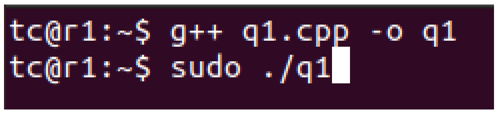


Figure 1: A network with 2 subnets.

I created the function **printMACAddress()** which prints the MAC address in a pretty format using the given array of unsigned characters.
I created the function **printEthernetHeader()** which takes the packet in form of a buffer and extracts the source and destination MAC addresses from that packet. For this purpose, I simply used the built-in `struct ethhdr` and typecast the buffer to a pointer of this struct.
I created the function **printIpHeader()** which takes the packet in form of a buffer and extracts the source and destination IP addresses from that packet. For this purpose, I simply used the built-in

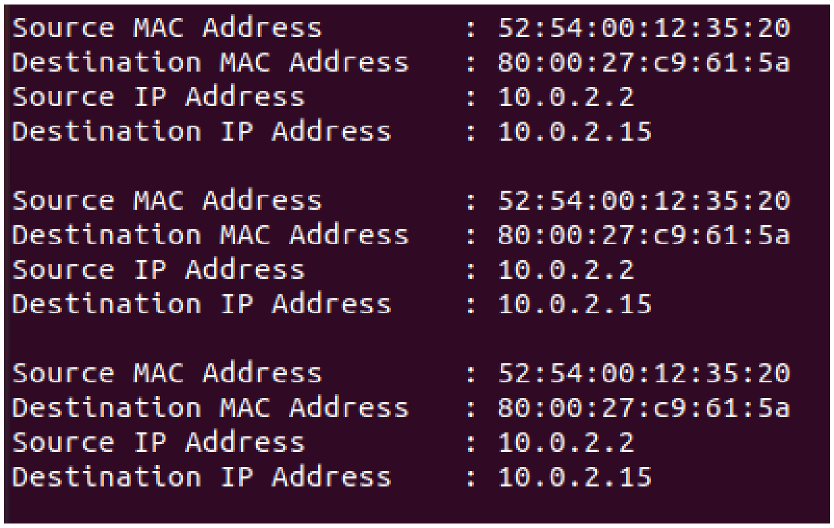
`struct ip` and typecast the buffer to a pointer of this struct leaving the size of `struct ethhdr` from the start to skip the ethernet header.
In my **main** function, I created a raw socket with **ETH_P_IP** as the protocol to listen to all the IP packets. Then, I put it in a **recvfrom** loop, receive data on it, and print their source and destination MAC and IP addresses. A raw socket when put in a **recvfrom** loop receives all incoming packets. This is because it is not bound to a particular address or port).

Here are some of the observations. I found that when doing the **tcpdump** command on router **r1**, there were a lot of incoming packets because of **ssh** which was flooding the incoming packets.



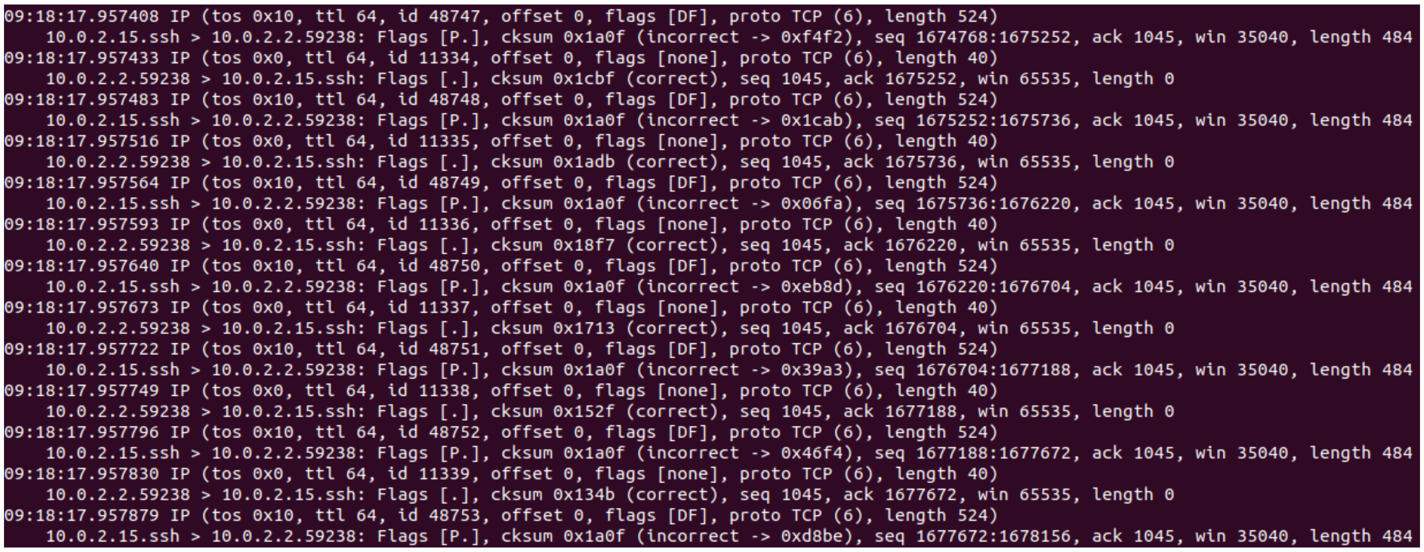
```
tc@r1:~$ g++ q1.cpp -o q1
tc@r1:~$ sudo ./q1
```

(Code Output)



```
Source MAC Address        : 52:54:00:12:35:20
Destination MAC Address   : 80:00:27:c9:61:5a
Source IP Address         : 10.0.2.2
Destination IP Address    : 10.0.2.15

Source MAC Address        : 52:54:00:12:35:20
Destination MAC Address   : 80:00:27:c9:61:5a
Source IP Address         : 10.0.2.2
Destination IP Address    : 10.0.2.15

Source MAC Address        : 52:54:00:12:35:20
Destination MAC Address   : 80:00:27:c9:61:5a
Source IP Address         : 10.0.2.2
Destination IP Address    : 10.0.2.15
```

(tcpdump command output)

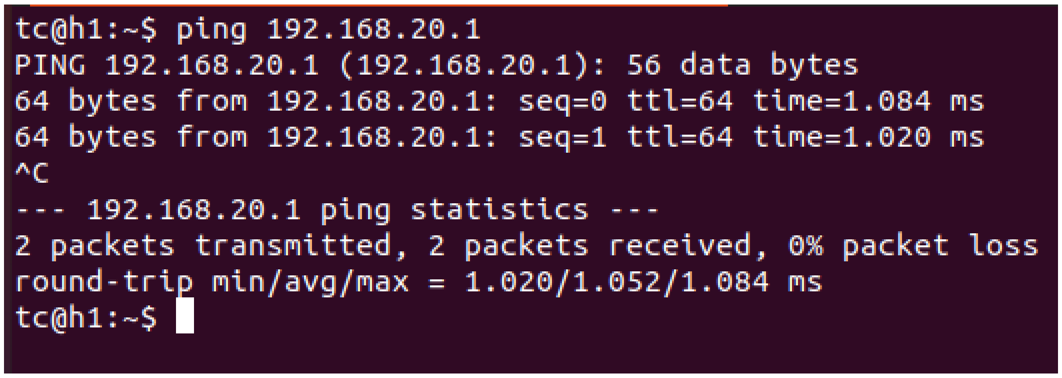


```
09:18:17.957408 IP (tos 0x10, ttl 64, id 48747, offset 0, flags [DF], proto TCP (6), length 524)
    10.0.2.15.ssh > 10.0.2.2.59238: Flags [P.], cksum 0x1a0f (incorrect -> 0xf4f2), seq 1674768:1675252, ack 1045, win 35040, length 484
09:18:17.957433 IP (tos 0x0, ttl 64, id 11334, offset 0, flags [none], proto TCP (6), length 40)
    10.0.2.2.59238 > 10.0.2.15.ssh: Flags [.], cksum 0x1cbf (correct), seq 1045, ack 1675252, win 65535, length 0
09:18:17.957483 IP (tos 0x10, ttl 64, id 48748, offset 0, flags [DF], proto TCP (6), length 524)
    10.0.2.15.ssh > 10.0.2.2.59238: Flags [P.], cksum 0x1a0f (incorrect -> 0x1cab), seq 1675252:1675736, ack 1045, win 35040, length 484
09:18:17.957516 IP (tos 0x0, ttl 64, id 11335, offset 0, flags [none], proto TCP (6), length 40)
    10.0.2.2.59238 > 10.0.2.15.ssh: Flags [.], cksum 0x1adb (correct), seq 1045, ack 1675736, win 65535, length 0
09:18:17.957564 IP (tos 0x10, ttl 64, id 48749, offset 0, flags [DF], proto TCP (6), length 524)
    10.0.2.15.ssh > 10.0.2.2.59238: Flags [P.], cksum 0x1a0f (incorrect -> 0x06fa), seq 1675736:1676220, ack 1045, win 35040, length 484
09:18:17.957593 IP (tos 0x0, ttl 64, id 11336, offset 0, flags [none], proto TCP (6), length 40)
    10.0.2.2.59238 > 10.0.2.15.ssh: Flags [.], cksum 0x18f7 (correct), seq 1045, ack 1676220, win 65535, length 0
09:18:17.957640 IP (tos 0x10, ttl 64, id 48750, offset 0, flags [DF], proto TCP (6), length 524)
    10.0.2.15.ssh > 10.0.2.2.59238: Flags [P.], cksum 0x1a0f (incorrect -> 0xeb8d), seq 1676220:1676704, ack 1045, win 35040, length 484
09:18:17.957673 IP (tos 0x0, ttl 64, id 11337, offset 0, flags [none], proto TCP (6), length 40)
    10.0.2.2.59238 > 10.0.2.15.ssh: Flags [.], cksum 0x1713 (correct), seq 1045, ack 1676704, win 65535, length 0
09:18:17.957722 IP (tos 0x10, ttl 64, id 48751, offset 0, flags [DF], proto TCP (6), length 524)
    10.0.2.15.ssh > 10.0.2.2.59238: Flags [P.], cksum 0x1a0f (incorrect -> 0x39a3), seq 1676704:1677188, ack 1045, win 35040, length 484
09:18:17.957749 IP (tos 0x0, ttl 64, id 11338, offset 0, flags [none], proto TCP (6), length 40)
    10.0.2.2.59238 > 10.0.2.15.ssh: Flags [.], cksum 0x152f (correct), seq 1045, ack 1677188, win 65535, length 0
09:18:17.957796 IP (tos 0x10, ttl 64, id 48752, offset 0, flags [DF], proto TCP (6), length 524)
    10.0.2.15.ssh > 10.0.2.2.59238: Flags [P.], cksum 0x1a0f (incorrect -> 0x46f4), seq 1677188:1677672, ack 1045, win 35040, length 484
09:18:17.957830 IP (tos 0x0, ttl 64, id 11339, offset 0, flags [none], proto TCP (6), length 40)
    10.0.2.2.59238 > 10.0.2.15.ssh: Flags [.], cksum 0x134b (correct), seq 1045, ack 1677672, win 65535, length 0
09:18:17.957879 IP (tos 0x10, ttl 64, id 48753, offset 0, flags [DF], proto TCP (6), length 524)
    10.0.2.15.ssh > 10.0.2.2.59238: Flags [P.], cksum 0x1a0f (incorrect -> 0xd8be), seq 1677672:1678156, ack 1045, win 35040, length 484
```

Q2. Modify the packet sniffer in the previous exercise to print the MAC addresses (source and destination), and IP addresses (source and destination) of only those IP packets received by r1 on interface *eth1.*

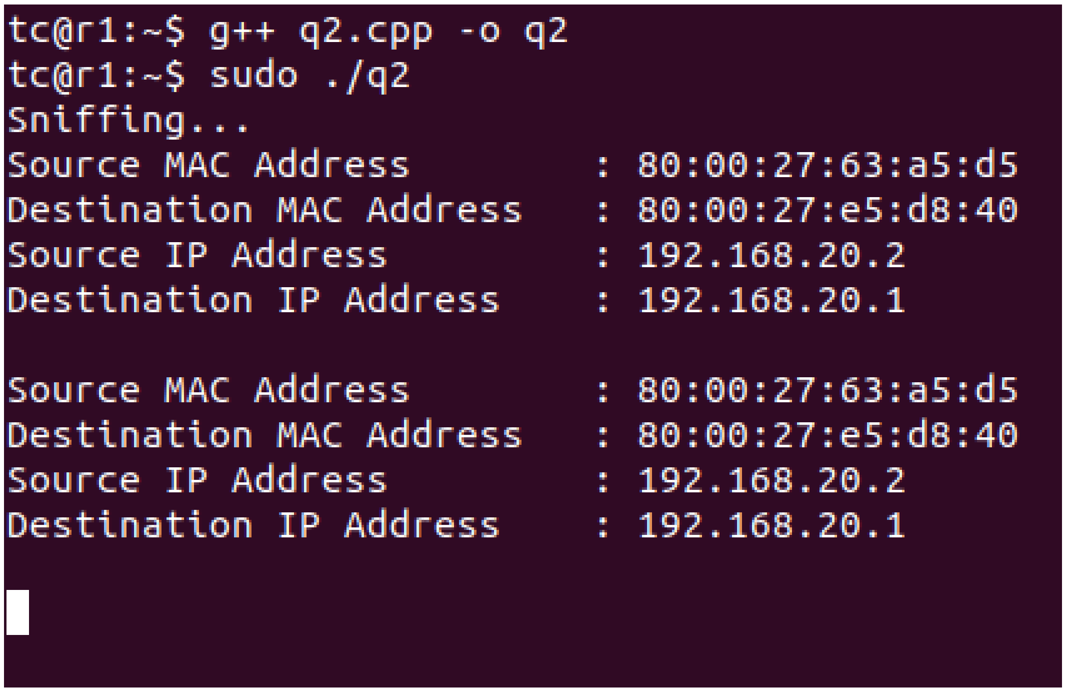
I have the same set of functions as in the previous question. This time, after creating the socket, I am also binding the socket to the interface **eth1** using the **bind** system call, so that it only listens to the incoming packets on the given interface. Also, while receiving the packets, I am only printing the details of the packet which are not outgoing as we are supposed to only print the details of the IP packets received.

Here are some of the observations. This time, because we are only listening to the IP packets on interface **eth1**, the flooding of the packets was not there and so I pinged the IP address of the interface which is **192.168.20.1** from various machines to send packets to that interface.

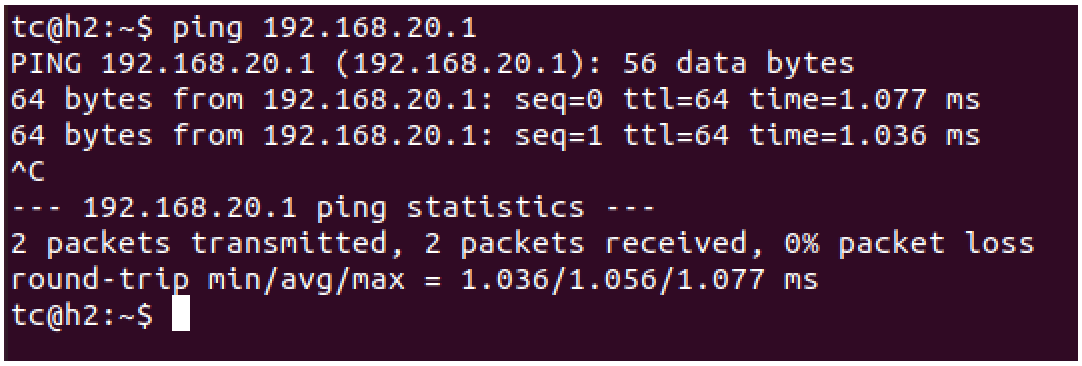


```
tc@h1:~$ ping 192.168.20.1
PING 192.168.20.1 (192.168.20.1): 56 data bytes
64 bytes from 192.168.20.1: seq=0 ttl=64 time=1.084 ms
64 bytes from 192.168.20.1: seq=1 ttl=64 time=1.020 ms
^C
--- 192.168.20.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1.020/1.052/1.084 ms
tc@h1:~$
```

(Ping eth1 from h1)

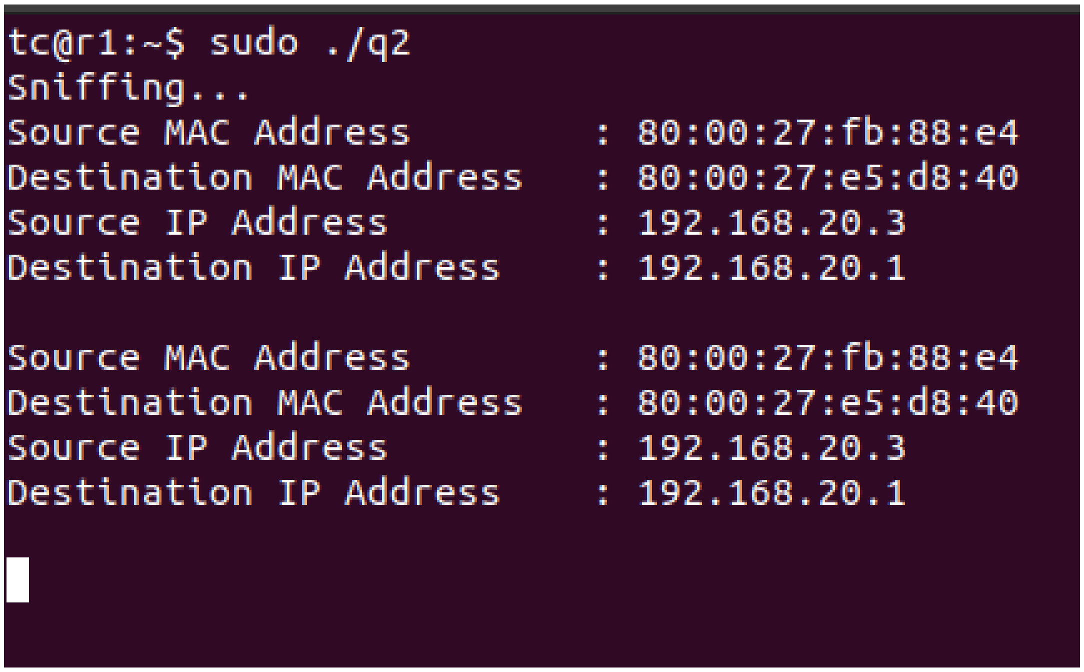


```
tc@r1:~$ g++ q2.cpp -o q2
tc@r1:~$ sudo ./q2
Sniffing...
Source MAC Address          : 80:00:27:63:a5:d5
Destination MAC Address     : 80:00:27:e5:d8:40
Source IP Address           : 192.168.20.2
Destination IP Address      : 192.168.20.1

Source MAC Address          : 80:00:27:63:a5:d5
Destination MAC Address     : 80:00:27:e5:d8:40
Source IP Address           : 192.168.20.2
Destination IP Address      : 192.168.20.1
```

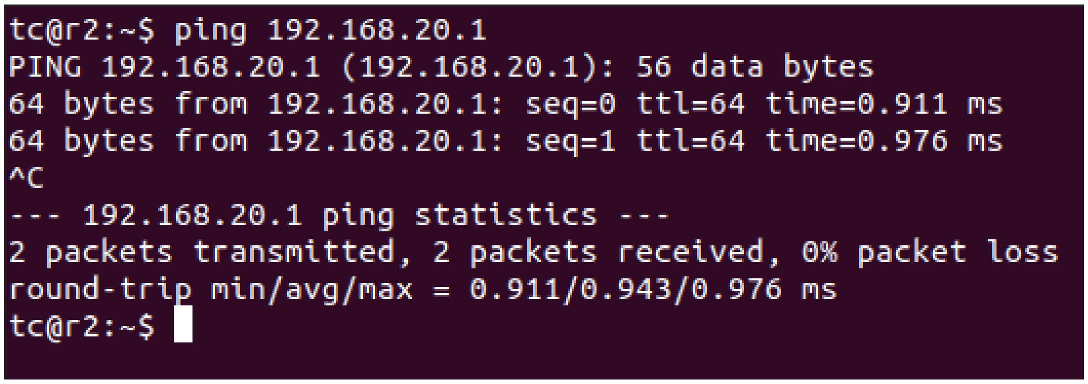


```
tc@h2:~$ ping 192.168.20.1
PING 192.168.20.1 (192.168.20.1): 56 data bytes
64 bytes from 192.168.20.1: seq=0 ttl=64 time=1.077 ms
64 bytes from 192.168.20.1: seq=1 ttl=64 time=1.036 ms
^C
--- 192.168.20.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1.036/1.056/1.077 ms
tc@h2:~$
```

(Ping eth1 from h2)

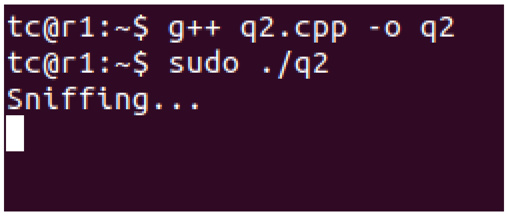


```
tc@r1:~$ sudo ./q2
Sniffing...
Source MAC Address          : 80:00:27:fb:88:e4
Destination MAC Address     : 80:00:27:e5:d8:40
Source IP Address           : 192.168.20.3
Destination IP Address      : 192.168.20.1

Source MAC Address          : 80:00:27:fb:88:e4
Destination MAC Address     : 80:00:27:e5:d8:40
Source IP Address           : 192.168.20.3
Destination IP Address      : 192.168.20.1
```



```
tc@r2:~$ ping 192.168.20.1
PING 192.168.20.1 (192.168.20.1): 56 data bytes
64 bytes from 192.168.20.1: seq=0 ttl=64 time=0.911 ms
64 bytes from 192.168.20.1: seq=1 ttl=64 time=0.976 ms
^C
--- 192.168.20.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.911/0.943/0.976 ms
tc@r2:~$
```

(Ping eth1 from r2)



```
tc@r1:~$ g++ q2.cpp -o q2
tc@r1:~$ sudo ./q2
Sniffing...
```

(No output as it only listens to packets coming on eth1)

Q3. Enhance the program created in the previous exercise so that you transfer IP packets across the subnets in Fig. 1. In fact, you will essentially be building a rudimentary router. Some hints can be found at a-guide-to-using-raw-sockets

For this purpose, I need to create two receiving sockets that will receive the IP packets on both interfaces as in the previous question. Also, I will need to create two sending sockets that will be used to send the received packets.
I modified the **printEthernetHeader()** and **printIpHeader()** functions in the previous question to not only print the source and destination MAC and IP addresses but also return their values as strings.

I created the function **createRecvSocket()** that will create a receiving socket and bind it to the given interface. The domain and protocol used for this purpose are **AF_PACKET** and **ETH_P_IP**.
I created the function **createSendSocket()** that will create a sending socket. The domain and protocol used for this purpose are **AF_INET** and **IPPROTO_RAW**.

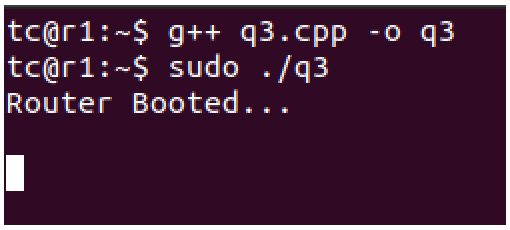
I created the function **getIfInfo()** which will return the MAC and IP address of the given interface. For this purpose, I used the **ioctl()** system call using the **ifreq** struct.

Then in the main function, I created both the receiving socket and sending sockets. Because I need to simultaneously listen on both the receiving sockets, I need to use them with the **select()** system call. So, I created the **fd_set** object and add the receiving socket descriptors to this set. I also retrieved the MAC and IP addresses of both interfaces using the **getIfInfo()** function.

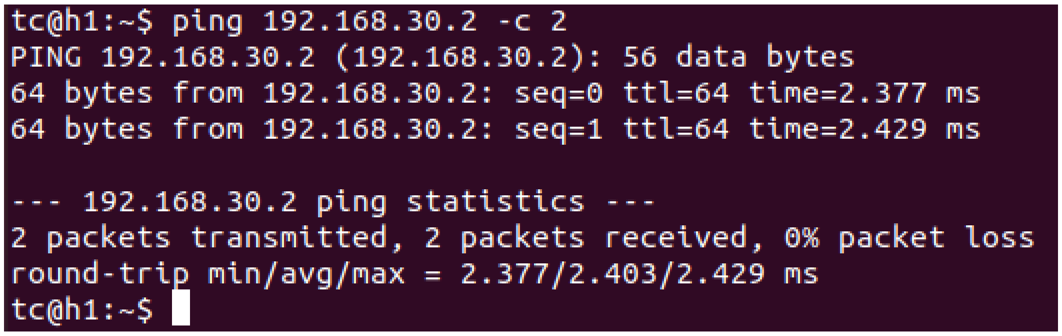
Then, in an infinite loop, I keep on waiting for any of the receiving socket descriptors to become ready. When any one of them is ready, I checked if the packet was not outgoing as in the previous question, and also checked if the packet was not destined for this interface itself by checking the destination IP address in the packet and the IP address of the interface.

Then, for routing this packet, I am not providing the ethernet header information, and the kernel and the OS will automatically fill that for me. So, I skipped the ethernet header from the received packet and send the rest of the packet using the **sendto()** function.

Here are some of the observations. If the router is working properly, then executing the ping command from any node to any other node should work as usual.

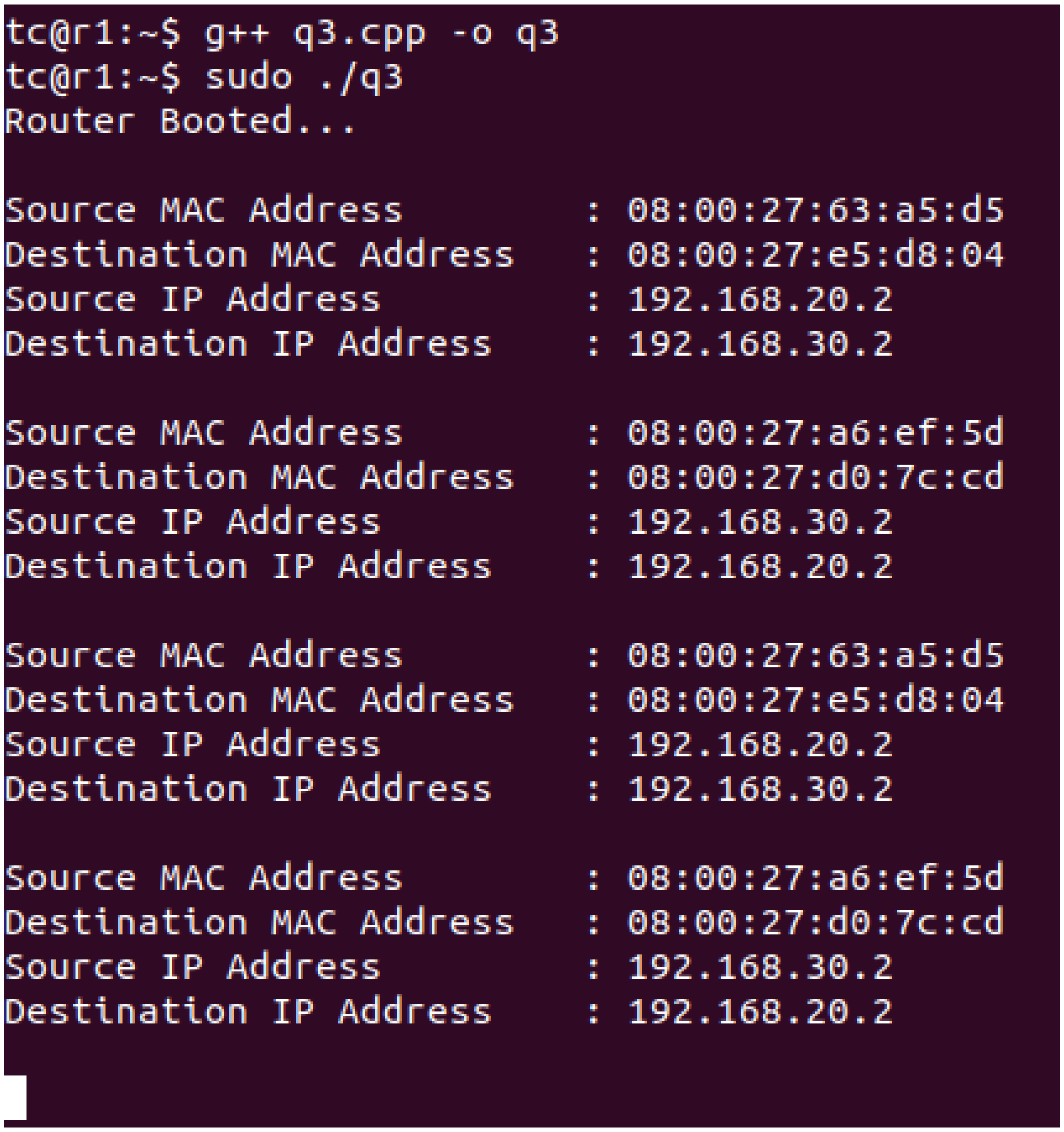


```
tc@r1:~$ g++ q3.cpp -o q3
tc@r1:~$ sudo ./q3
Router Booted...
```

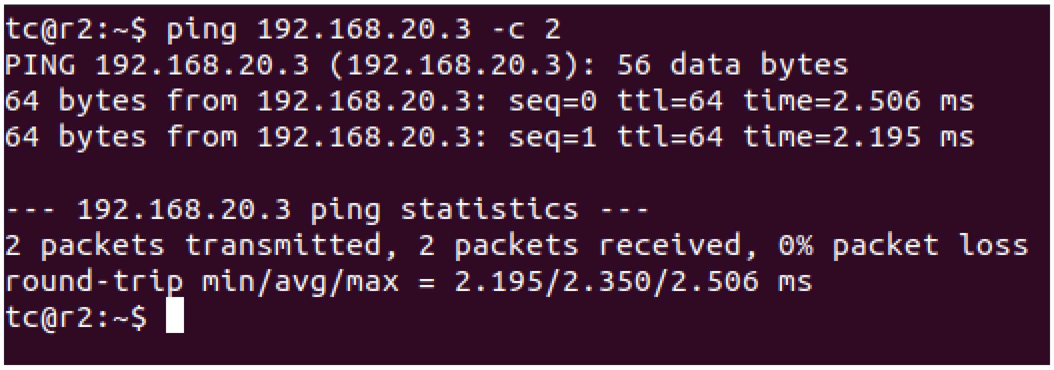


```
tc@h1:~$ ping 192.168.30.2 -c 2
PING 192.168.30.2 (192.168.30.2): 56 data bytes
64 bytes from 192.168.30.2: seq=0 ttl=64 time=2.377 ms
64 bytes from 192.168.30.2: seq=1 ttl=64 time=2.429 ms

--- 192.168.30.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2.377/2.403/2.429 ms
tc@h1:~$
```

(ping r2 from h1)

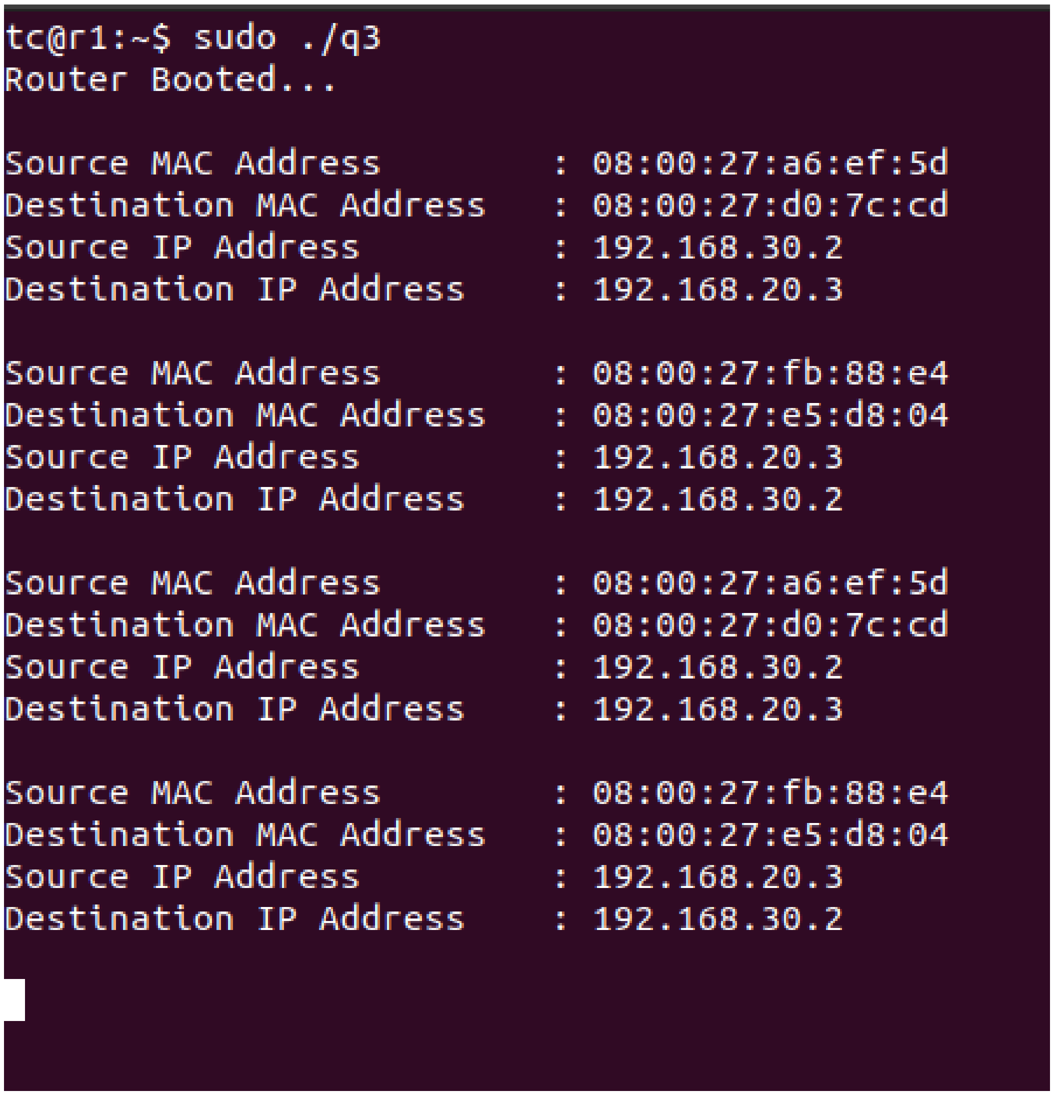


```
tc@r1:~$ g++ q3.cpp -o q3
tc@r1:~$ sudo ./q3
Router Booted...

Source MAC Address         : 08:00:27:63:a5:d5
Destination MAC Address    : 08:00:27:e5:d8:04
Source IP Address          : 192.168.20.2
Destination IP Address     : 192.168.30.2

Source MAC Address         : 08:00:27:a6:ef:5d
Destination MAC Address    : 08:00:27:d0:7c:cd
Source IP Address          : 192.168.30.2
Destination IP Address     : 192.168.20.2

Source MAC Address         : 08:00:27:63:a5:d5
Destination MAC Address    : 08:00:27:e5:d8:04
Source IP Address          : 192.168.20.2
Destination IP Address     : 192.168.30.2

Source MAC Address         : 08:00:27:a6:ef:5d
Destination MAC Address    : 08:00:27:d0:7c:cd
Source IP Address          : 192.168.30.2
Destination IP Address     : 192.168.20.2
```



```
tc@r2:~$ ping 192.168.20.3 -c 2
PING 192.168.20.3 (192.168.20.3): 56 data bytes
64 bytes from 192.168.20.3: seq=0 ttl=64 time=2.506 ms
64 bytes from 192.168.20.3: seq=1 ttl=64 time=2.195 ms

--- 192.168.20.3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2.195/2.350/2.506 ms
tc@r2:~$
```

(ping h2 from r2)



```
tc@r1:~$ sudo ./q3
Router Booted...

Source MAC Address          : 08:00:27:a6:ef:5d
Destination MAC Address     : 08:00:27:d0:7c:cd
Source IP Address           : 192.168.30.2
Destination IP Address      : 192.168.20.3

Source MAC Address          : 08:00:27:fb:88:e4
Destination MAC Address     : 08:00:27:e5:d8:04
Source IP Address           : 192.168.20.3
Destination IP Address      : 192.168.30.2

Source MAC Address          : 08:00:27:a6:ef:5d
Destination MAC Address     : 08:00:27:d0:7c:cd
Source IP Address           : 192.168.30.2
Destination IP Address      : 192.168.20.3

Source MAC Address          : 08:00:27:fb:88:e4
Destination MAC Address     : 08:00:27:e5:d8:04
Source IP Address           : 192.168.20.3
Destination IP Address      : 192.168.30.2
```