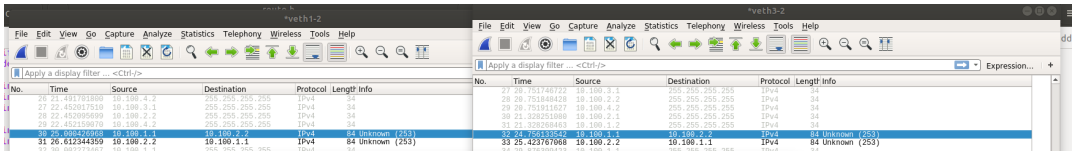
Codelist for each programming task

Link-layer: Packet I/O On Ethernet

- Functions for manage devices are in `inc/device.h`
- Functions for sending and receiving Ethernet frames are in `inc/packetio.h`
- Helper functions are in `inc/packetio.h`
- Test files are in `test`
  - `print_my_mac.cpp` is used to print one device's mac address
  - `mac_client.cpp` and `mac_server.cpp` should be run on veth1-2 and veth2-1 respectively in the example network, they implement a simple 'echo function' to output all the received packets as they are
- Run `./compile.sh` could generate binary files in the `test` folder

Network-layer: IP Protocol

- Functions related to routing are in `inc/route.h`
- Functions related to IP packets processing are in `inc/ip.h`
- Test files are in `test`
  - `print_my_ip.cpp` is used to print one device's ip address
  - `ip_client.cpp` and `ip_middle.cpp` and `ip_server.cpp` should be run on host 1, 2, 3 respectively, the server echoes the messages sent from client to server and the result shows below:
  - 

  - **Notes after finishing transport layer:** as I modified interface in the IP layer, this program cannot run correctly now, but with some modifications it can still work.

Transport-layer: TCP Protocol

- Functions related to TCP state machine are in `inc/tcp.h`
- Functions related to POSIX interface are in `inc/socket.h`
- test1
  - `tcp_client.cpp` and `tcp_server.cpp` can be used to test the POSIX interface
  - We tried to run client on ns1, ns2 almost concurrently and server on ns3
  - Message from ns3

```
root@ubuntu:~/Desktop/lab2/out# ./tcp_server.o veth3-0 veth3-2 veth3-4
==========================================
Added device: veth3-0  ID: 0
MAC address: be:4d:8c:c3:d1:c1
IP address:  10.100.  4.  2
fd: 4
==========================================
==========================================
Added device: veth3-2  ID: 1
MAC address: c6:81:01:2d:a3:92
IP address:  10.100.  2.  2
fd: 5
==========================================
==========================================
Added device: veth3-4  ID: 2
MAC address: d6:44:81:1a:cf:fb
IP address:  10.100.  3.  1
fd: 6
==========================================
[INFO] Connected to client  10.100.  2.  1
[INFO] Server received 200 bytes
[INFO] Server echoed 200 bytes
[INFO] Connection closed
[INFO] Connected to client  10.100.  1.  1
[INFO] Server received 200 bytes
[INFO] Server echoed 200 bytes
[INFO] Connection closed
^C
```

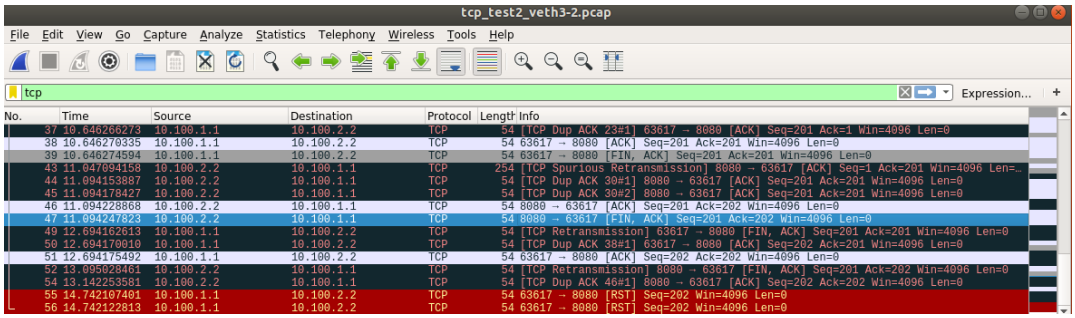
- Message from ns1

```
root@ubuntu:~/Desktop/lab2/out# ./tcp_client.o veth1-2
==========================================
Added device: veth1-2  ID: 0
MAC address: ae:40:fd:f0:cc:9a
IP address:  10.100.  1.  1
fd: 4
==========================================
[INFO] Successfully sent 100 bytes
[INFO] Successfully sent 100 bytes
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 00 01 02 03 04 05 06 07 08 09 0a 0b
0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b
1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b
2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b
3c 3d 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b
4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b
5c 5d 5e 5f 60 61 62 63
[INFO] Successfully received 200 bytes
[INFO] Closing ... Press Ctrl + C to shutdown
^C
```

- Message from ns2

```
root@ubuntu:~/Desktop/lab2/out# ./tcp_client.o veth2-1 veth2-3
==========================================
Added device: veth2-1  ID: 0
MAC address: 2a:ec:9b:28:9c:d0
IP address:  10.100.  1.  2
fd: 4
==========================================
==========================================
Added device: veth2-3  ID: 1
MAC address: 72:75:3f:04:e3:92
IP address:  10.100.  2.  1
fd: 5
==========================================
[INFO] Successfully sent 100 bytes
[INFO] Successfully sent 100 bytes
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 00 01 02 03 04 05 06 07 08 09 0a 0b
0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b
1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b
2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b
3c 3d 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b
4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b
5c 5d 5e 5f 60 61 62 63
[INFO] Successfully received 200 bytes
[INFO] Closing ... Press Ctrl + C to shutdown
^C
```

- We also saved the pcap files for two tries `tcp_test1_*.pcap` and `tcp_test2_*.pcap`, the variety of data can show the robustness of our program. A screenshot is shown below:



- test2
  - `tcp_client_1.cpp` and `tcp_server_2.cpp` are used to test super large information sent across network which are easily lost and over buffer size
  - They are run on host ns1 and ns3 respectively, while `ip_middle.cpp` is run on ns2 as a router
  - the result is as below:

```
root@ubuntu:~/Desktop/lab2/out# ./tcp_server_2.o veth3-0 veth3-2 veth3-4
==========================================
Added device: veth3-0  ID: 0
MAC address: be:4d:8c:c3:d1:c1
IP address:  10.100.  4.  2
fd: 4
==========================================
==========================================
Added device: veth3-2  ID: 1
MAC address: c6:81:01:2d:a3:92
IP address:  10.100.  2.  2
fd: 5
==========================================
==========================================
Added device: veth3-4  ID: 2
MAC address: d6:44:81:1a:cf:fb
IP address:  10.100.  3.  1
fd: 6
==========================================
[INFO] Connected to client  10.100.  1.  1
[INFO] Server received 3000 bytes
[INFO] Server echoed 3000 bytes
[INFO] Connection closed
^C
```

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 00 01 02 03 04 05 06 07
08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17
18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27
28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47
48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57
58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 66 67
68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77
[INFO] Successfully received 3000 bytes
[INFO] Closing ... Press Ctrl + C to shutdown
^C
```

- The pcap files are also saved and named as `tcp_test3_*.pcap`
- A wireshark screen shot is as follow: