

# CSED353 Network Term Project

20170684 강덕형

## 1. 개요 및 전반적인 Flow

나는 Raspberry Pi와 같은 single board computer를 가지고 있지 않기 때문에, topic B 중 “Packet Capturing program”을 구현하였다. 구현 과정에서 libpcap이라는 라이브러리를 사용하였으며, 각 network단의 message(Ethernet, IP, TCP 등등)의 header 형식을 사용하기 위하여 <net/Ethernet.h>, <netinet/ip.h> 등의 헤더 파일을 include하였다.

프로그램은 우선 pcap\_findalldevs()함수를 call하여 local machine에 연결되어 있는 interface 정보를 받아오는 것으로부터 시작된다. 이후 사용자가 선택한 interface를 pcap\_open\_live() 함수를 이용하여 open하고 해당 interface에 감지되는 packet을 tcp/udp/icmp message의 세 종류로 나눠서 도착 순서대로 보여준다.

## 2. 구현 설명

### 2.1. 헤더파일 include 및 전역 변수 정의

우선 packet capture를 수행하는 기능을 지원해주는 pcap library를 include해주었다. 또한 network packet structure를 이용하여 프로그램 실행 시 해당 packet에서 원하는 정보를 parsing 할 수 있게끔 <netinet/ip.h>, <netinet/tcp.h>, <netinet/udp.h>, <netinet/ip\_icmp.h> 등의 header를 include해주었다. initial\_sec / intial\_usec 변수는 프로그램이 첫번째 packet을 감지했을 때의 시간을 기록하기 위함이다. 해당 프로그램은 프로그램이 시작된 뒤 packet이 도착한 상대적 시간을 출력해야된다. 그러나 packet에 기록된 시간은 unix epoch(절대적 시간)이기에 상대적 시간을 구하기 위해서는 첫번째 packet의 도착 시간을 기록해두어야 한다.

### 2.2. Local machine에 연결된 device(interface) list 가져오기

pcap\_findalldevs(&alldevs\_list, errbuf)를 call하여 local machine의 device list를 가져올 수 있다. 해당 함수는 error가 발생할 때 -1을 리턴하고 해당 경우에 error message를 출력하고 종료되게 하였다. 만약 error가 발생하지 않았다면, alldevs\_list에 device list가 저장된다. alldevs\_list의 각 e원소는 device 정보를 담고 있는 pcap\_if\_t type의 구조체이다. pcap\_if\_t type 구조체는 name, description등의 member를 가지고 있다. 우리는 해당 alldevs\_lists를 순회하면서 각 device들의 name과 description등을 print한다. 각 device(interface)는 1부터 시작하는 interface number를 차례대로 부여 받게 된다. 유저는 이후 “Select the interface number ” 라는 message를 통해 자신이 분석하고자 할 interface를 선택하게 된다. 적합한 범위가 아닌 숫자가 입력되었을 경우에는, “Typed interface number is not in valid range. Type again” 메시지를 출력, 유저가 다시 입력하도록 유도한다.

## 2.3. Device(Interface) 열기

이후 사용자가 선택한 interface number에 맞게 `alldevs_list`를 다시 순회하며 해당하는 `pcap_if_t` type의 device 구조체 `dev`를 가져온다. 이후 **`pcap_open_live(dev->name, bufsize, 1, 1000, errbuf)`**를 call하여 device를 연다. `pcap_open_live`의 parameter 중 `dev->name`은 사용자가 선택한 device의 name이고, `bufsize`는 패킷당 저장할 바이트 수를 의미한다. 이 경우에는 65536으로 설정하여 packet에 탑재된 다양한 데이터들을 모두 가져올 수 있도록 하였다. 1은 PROMISCUOUS 여부를 나타낸다. 이는 네트워크 디바이스에 오는 모든 패킷을 받겠다는 의미이다. 1000은 packet buffer timeout을 의미하며, 단위는 ms이다. 에러가 발생한 경우 `errbuf`에 에러메시지가 저장된다.

device를 여는 것이 실패할 경우 error message를 출력하고 프로그램이 종료된다. 성공할 경우 "Selected device is available"이라는 message와 함께 얼마만큼의 packet을 처리하고 싶은지 물어보는 메시지가 출력된다. 만약 사용자가 0 혹은 음수를 입력하였을 경우, 프로그램은 에러가 발생할 때까지 계속 도착하는 packet을 처리하고 출력하게 된다. 양수를 입력하였을 경우 해당 숫자의 packet을 처리하고 이후 프로그램은 종료되게 된다. packet이 도착할때마다 **`pcap_loop(handle, cnt, got_packet, NULL)`**;이 call되게 된다. `pcap_loop()`는 callback function으로 packet이 도착할 때 마다 `got_packet` 함수를 실행한다.

## 2.4. Callback 함수

callback함수인 `got_packet`함수는 크게 4부분으로 나뉜다. packet arrival time 계산부/MAC info 처리부/IP header부/IP protocol 부로 나뉜다.

`got_packet` 함수는 아래와 같은 paramter를 가진다.

```
void got_packet(u_char *param, const struct pcap_pkthdr *hdr, const u_char *packet)
```

해당 함수는 packet이 도착할때마다 `pcap_loop()`에 의해 call되며, 위와 같은 parameter를 가진다. `pcap_pkthdr *hdr`는 **`ts(time stamp)`**라는 구조체를 member 변수로 가지며, 해당 구조체는 `tv_sec`, `tv_usec`이라는 변수를 member로 가진다. `tv_sec`은 해당 packet이 도착한 second를 `unix_epoch` 형식으로 표시한 것이고, `tv_msec`은 해당 packet이 도착한 microsecond를 표시한 것이다.(0~999999) 프로그램이 실행되고 packet이 처음 도달할 경우 해당 packet의 `tv_sec`과 `tv_usec` 값은 전역변수인 `initial_sec`과 `initial_usec`에 기록된다. 이후 이러한 값과 packet의 도착 시간의 차를 이용하여 우리는 프로그램이 시작된 후 packet이 도착한 상대적 시간을 계산해낼 수 있다. 계산된 값은 `second.microsecond` (e.g. 0.000000) 형식이며 이후 string으로 변환된 뒤 최종 msg에 concatenate된다.

이후 우리는 packet을 **`ether_header type`**으로 형변환한다.

```
ethernet_header = (struct ether_header *)packet;
```

이후 **concat\_mac()**을 call하여 msg에 Ethernet source address, destination address를 concatenate한다. **concat\_mac()**에서는 2진수 형태의 address를 Ethernet address의 format에 맞게 16진수로 변환하고 주소 구분자인 :를 삽입한다.

이후 Ethernet header에서 member 변수인 **ether\_type**을 가져온다. 이때

```
ether_type = ntohs(ether_header->ether_type);
```

와 같이 **ntohs** 함수를 사용한다. **ntohs** 함수는 네트워크 바이트 정렬 방식의 데이터를 호스트 바이트 정렬 방식으로 변환시켜준다. 운영체제에 따라 바이트 정렬 방식(little endian, big endian)이 다르기에, 여러 운영체제에서 호환 가능한 코드를 작성하기 위하여 위와 같은 함수를 네트워크 바이트로 구성된 멤버 변수의 값을 참조하는 부분등에 적용하여 사용하였다. 이후 **ether\_type**이 IP일 경우 packet으로부터 **ip\_header** 부분을 parsing하여 struct ip type으로 형변환하여 **ip\_header** 변수가 해당 부분을 가리키도록 하였다.

```
ip_header = (struct ip *)(packet + sizeof(struct ether_header));
```

이후 **concat\_ip()**함수를 호출하여 ip datagram의 source와 destination address 정보를 msg에 concatenate시켜준다. **ip\_header**에는 **ip\_p**라는 protocol을 나타내는 멤버 변수가 있는데, 해당 변수의 값에 맞게 서로 다른 처리를 수행해 주었다. **ip\_p**값이 **IPPROTO\_TCP**인 경우에는 protocol이 TCP인 경우이다.

```
tcp_header = (struct tcphdr *)(packet + sizeof(struct ether_header) + ip_header->ip_hl * 4);
```

**packet**으로부터 **tcp\_header**를 parsing하고 이후 **tcp\_header**의 **th\_sport**, **th\_dport**, **th\_seq**, **th\_ack** 멤버 변수의 값을 참조하여 source port number, destination port number, sequence number, ack number등의 값을 가져올 수 있었다. 이후 이러한 값들을 msg에 concatenate 시켜 주었다.

```
printf("\033[1;36m %s\n",msg);
```

TCP인 경우에는 message를 출력할 때 font color가 **Bold Cyan**이게 처리해주었다.

**ip\_p**값이 **IPPROTO\_UDP**인 경우에도 마찬가지로 **packet**으로부터 **udp\_header**를 parsing하고, **udp\_header**로부터 **uh\_sport**, **uh\_dport**등의 멤버 변수를 참조하여 source port number, destination port number 값을 가져왔다. 이후 이러한 값을 msg에 concatenate 시켜주었다.

```
printf("\033[1;32m %s\n",msg);
```

UDP인 경우에는 message를 출력할 때 font color가 **Bold Green**이게 처리해주었다.

**ip\_p**값이 **IPPROTO\_ICMP**인 경우에도 마찬가지로 **packet**에서 **icmp\_header**를 parsing하고, 해당 header로부터 type, code등의 멤버 변수를 참조하여 icmp type, code 값을 가져왔다. 이후 이러한 값을 msg에 concatenate 시켜주었다.

```
printf("\033[1;33m %s\n",msg);
```

ICMP인 경우에는 message를 출력할 때 font color가 **Bold Yellow**이게 처리해주었다.

### 3. 실행 예제

```
kang@kang-VirtualBox: ~/바탕화면/network_termproject
kang@kang-VirtualBox:~/바탕화면/network_termproject$ ls
main.c
kang@kang-VirtualBox:~/바탕화면/network_termproject$ gcc -o main main.c -lpcap
kang@kang-VirtualBox:~/바탕화면/network_termproject$ ls
main  main.c
kang@kang-VirtualBox:~/바탕화면/network_termproject$ sudo ./main
[sudo] kang의 암호:
1: enp0s3 (No description available)
2: lo (No description available)
3: any (Pseudo-device that captures on all interfaces)
4: bluetooth-monitor (Bluetooth Linux Monitor)
5: nflog (Linux netfilter log (NFLOG) interface)
6: nfqueue (Linux netfilter queue (NFQUEUE) interface)
Select the interface number (1-6):
```

① 프로그램이 실행되면, 해당 local machine에 연결된 device들의 list를 가져오고 description과 함께 보여준다. 유저는 분석하고자 하는 interface의 number를 입력할 수 있다.

```
kang@kang-VirtualBox: ~/바탕화면/network_termproject
kang@kang-VirtualBox:~/바탕화면/network_termproject$ ls
main.c
kang@kang-VirtualBox:~/바탕화면/network_termproject$ gcc -o main main.c -lpcap
kang@kang-VirtualBox:~/바탕화면/network_termproject$ ls
main  main.c
kang@kang-VirtualBox:~/바탕화면/network_termproject$ sudo ./main
[sudo] kang의 암호:
1: enp0s3 (No description available)
2: lo (No description available)
3: any (Pseudo-device that captures on all interfaces)
4: bluetooth-monitor (Bluetooth Linux Monitor)
5: nflog (Linux netfilter log (NFLOG) interface)
6: nfqueue (Linux netfilter queue (NFQUEUE) interface)
Select the interface number (1-6): 0
Typed interface number is not in valid range. Type again.
Select the interface number (1-6):
```

② 만약 사용자가 입력한 interface number가 invalid range에 있는 경우, 사용자에게 잘못 입력하였다는 메시지를 출력하고 재입력하게 한다.

```
kang@kang-VirtualBox: ~/바탕화면/network_termproject
kang@kang-VirtualBox:~/바탕화면/network_termproject$ ls
main.c
kang@kang-VirtualBox:~/바탕화면/network_termproject$ gcc -o main main.c -lpcap
kang@kang-VirtualBox:~/바탕화면/network_termproject$ ls
main main.c
kang@kang-VirtualBox:~/바탕화면/network_termproject$ sudo ./main
[sudo] kang의 암호:
1: enp0s3 (No description available)
2: lo (No description available)
3: any (Pseudo-device that captures on all interfaces)
4: bluetooth-monitor (Bluetooth Linux Monitor)
5: nflog (Linux netfilter log (NFLOG) interface)
6: nfqueue (Linux netfilter queue (NFQUEUE) interface)
Select the interface number (1-6): 0
Typed interface number is not in valid range. Type again.
Select the interface number (1-6): 1
Selected device enp0s3 is available
How many packet do you want to process?
If you want to process infinity packets, type 0 : 1
```

③ 사용자가 valid range에 있는 interface number를 입력한 경우, 이후 얼마만큼의 packet을 처리할 것인지 질의하는 메시지를 출력한다.

```
select the interface number (1-6): 1
selected device enp0s3 is available
How many packet do you want to process?
If you want to process infinity packets, type 0 : 0
0.0: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->168.126.63.1] UDP Src Port: 59211, Dst Port: 53
0.272: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->168.126.63.1] UDP Src Port: 59145, Dst Port: 53
0.4729: [52:54:00:12:35:02->00:00:27:ba:de:74][168.126.63.1->10.0.2.15] UDP Src Port: 53, Dst Port: 59211
0.5704: [52:54:00:12:35:02->00:00:27:ba:de:74][168.126.63.1->10.0.2.15] UDP Src Port: 53, Dst Port: 59145
0.6180: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 0
0.97590: [52:54:00:12:35:02->00:00:27:ba:de:74][103.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.97630: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.100855: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.104087: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.196121: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.196132: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.196991: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.196998: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.220398: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.220631: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.221331: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.221484: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.221576: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.221977: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.313200: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.313284: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.313745: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.313768: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.313750: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.316032: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 4555, Ack Num: 10432
0.359193: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.359476: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.450701: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.452098: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.452315: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.481319: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.481400: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.481886: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.819106: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.821156: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->103.102.166.224] TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.821456: [52:54:00:12:35:02->00:00:27:ba:de:74][003.102.166.224->10.0.2.15] TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.821584: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->168.126.63.1] UDP Src Port: 39203, Dst Port: 53
0.825819: [00:00:27:ba:de:74->52:54:00:12:35:02][00:0.2.15->168.126.63.1] UDP Src Port: 52744, Dst Port: 53
0.831383: [52:54:00:12:35:02->00:00:27:ba:de:74][168.126.63.1->10.0.2.15] UDP Src Port: 53, Dst Port: 52744
0.831398: [52:54:00:12:35:02->00:00:27:ba:de:74][168.126.63.1->10.0.2.15] UDP Src Port: 53, Dst Port: 39203
```

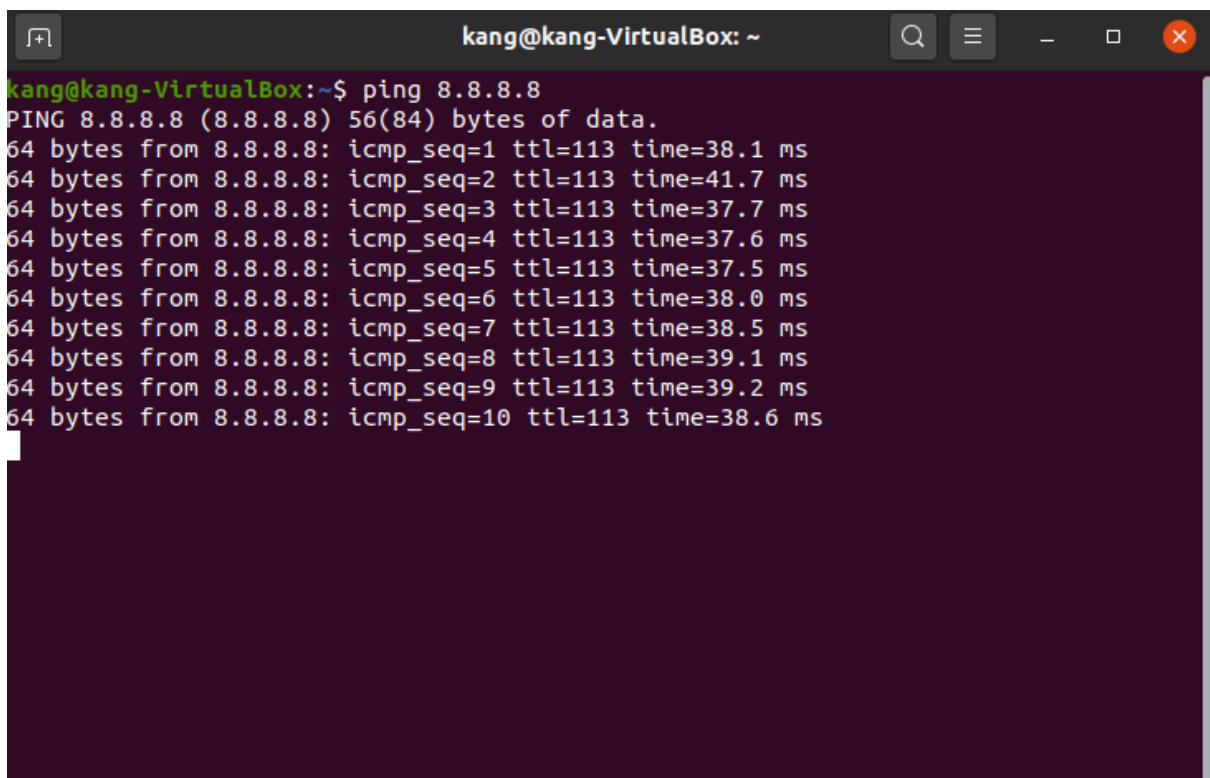
④ 0을 입력하였기에 error가 발생할때까지 device로 도착하는 모든 packet을 처리한다. TCP/UDP에 따라 다른 색깔의 메시지가 출력되는 것을 확인할 수 있으며 모두 메시지의 시작 부분에 packet 도착 시간을 출력하고 있음을 확인할 수 있다. 이후 src MAC address, dst MAC address,



src ip address, dst ip address가 출력되는 것을 확인할 수 있다.

```
0.0: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->168.126.63.1) UDP Src Port: 59211, Dst Port: 53
0.272: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->168.126.63.1) UDP Src Port: 59145, Dst Port: 53
0.4729: [52:54:00:12:35:02->08:00:27:ba:de:74](168.126.63.1->10.0.2.15) UDP Src Port: 53, Dst Port: 59211
0.5704: [52:54:00:12:35:02->08:00:27:ba:de:74](168.126.63.1->10.0.2.15) UDP Src Port: 53, Dst Port: 59145
0.6180: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 0
0.97590: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.97630: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.103855: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.104087: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.196121: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.196132: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.196991: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.196998: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.220398: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.220631: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.221331: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.221494: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
0.221576: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.221977: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.312820: [52:54:00:12:35:02->08:00:27:ba:de:74](103.102.166.224->10.0.2.15) TCP Src Port: 443, Dst Port: 37476, Seq Num: 10432, Ack Num: 4555
0.312834: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->103.102.166.224) TCP Src Port: 37476, Dst Port: 443, Seq Num: 4555, Ack Num: 10432
```

⑤TCP의 경우 메시지가 Bold Cyan으로 출력되며, TCP src port number, dst port number, seq number, ack number가 출력됨을 확인할 수 있다. UDP의 경우 Bold Green으로 출력되며, UDP src port number, dst port number가 출력됨을 확인할 수 있다.



```
kang@kang-VirtualBox: ~  
kang@kang-VirtualBox:~$ ping 8.8.8.8  
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=38.1 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=41.7 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=37.7 ms  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=37.6 ms  
64 bytes from 8.8.8.8: icmp_seq=5 ttl=113 time=37.5 ms  
64 bytes from 8.8.8.8: icmp_seq=6 ttl=113 time=38.0 ms  
64 bytes from 8.8.8.8: icmp_seq=7 ttl=113 time=38.5 ms  
64 bytes from 8.8.8.8: icmp_seq=8 ttl=113 time=39.1 ms  
64 bytes from 8.8.8.8: icmp_seq=9 ttl=113 time=39.2 ms  
64 bytes from 8.8.8.8: icmp_seq=10 ttl=113 time=38.6 ms
```

⑥ICMP message를 수신하기 위해 8.8.8.8(Google DNS server)로 ping message를 보냈다.

```

46.981511: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
47.19572: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
47.983154: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
48.24803: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
48.985117: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
49.22848: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
49.986261: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
50.23796: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
50.987883: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
51.25412: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
51.989576: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
52.27583: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
52.990753: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
53.29281: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
53.992924: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
54.32011: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
54.993253: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
55.32409: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
55.995427: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
56.33996: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
56.996813: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
57.35613: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
57.998301: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
58.37153: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
59.540: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
59.38697: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
60.1810: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
60.40639: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
61.3684: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
61.56682: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
62.5302: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
62.44189: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
63.6724: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
63.45630: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
This packet is not an IP packet.
This packet is not an IP packet.
64.7949: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0
64.46936: [52:54:00:12:35:02->08:00:27:ba:de:74](8.8.8.8->10.0.2.15) ICMP Type: 0, Code: 0
65.9155: [08:00:27:ba:de:74->52:54:00:12:35:02](10.0.2.15->8.8.8.8) ICMP Type: 2048, Code: 0

```

⑦ICMP message가 도착한 것을 확인할 수 있다. ICMP message는 Bold Yellow로 출력된다. 메시지의 Type이 0이고 Code가 0인 것으로 보아, Echo reply 메시지이다. Interface로 도착하는 packet 중 몇몇 packet은 IP packet이 아니다. 이 경우에는 "This packet is not an IP packet"이라는 메시지를 출력하도록 하였다.

## 4. 결론 및 발전방향

이번 Term Project를 통해 libpcap을 활용하여 간단한 packet capturing program을 구현할 수 있었다. 현재는 packet의 일부 header 정보만을 활용하여 packet분석을 진행한다. 이에 추후에는 사용자가 특정 packet을 지목하면 해당 packet의 모든 header 정보들을 활용하여 packet 분석을 진행할 수 있는 기능을 구현하고자 한다.