

# CPSC Project 1: Packet Sniffing and Spoofing

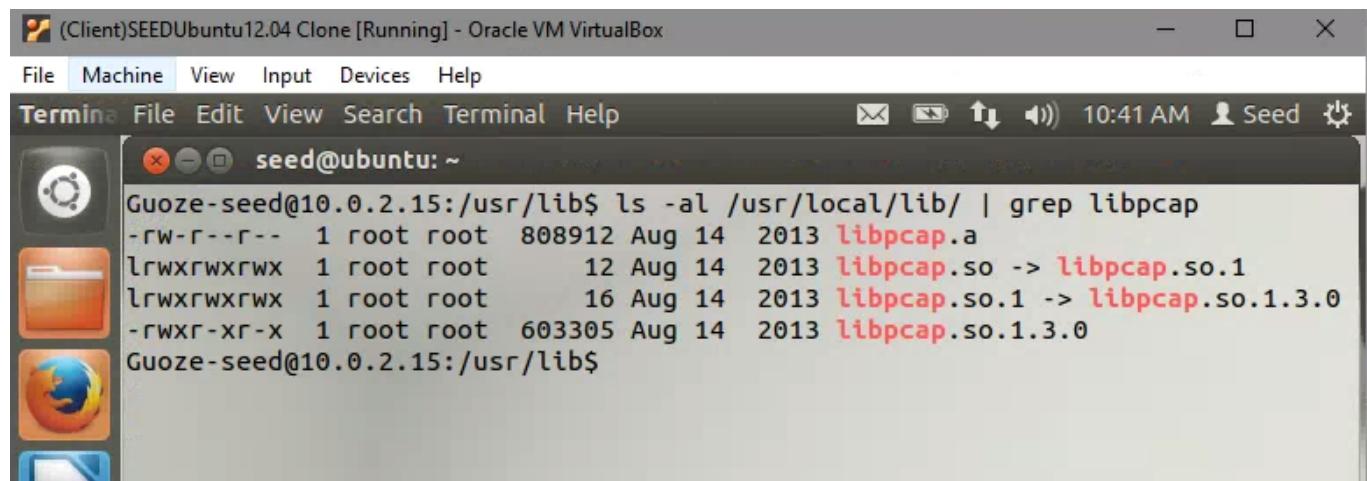
CPSC 8580 Project1  
Guoze Tang  
09/19/2018

## Task 1: Writing Packet Sniffing Program

I just download the `sniffex.c` file from the [Programming with pcap](#) website.

### Install the `libpcap`

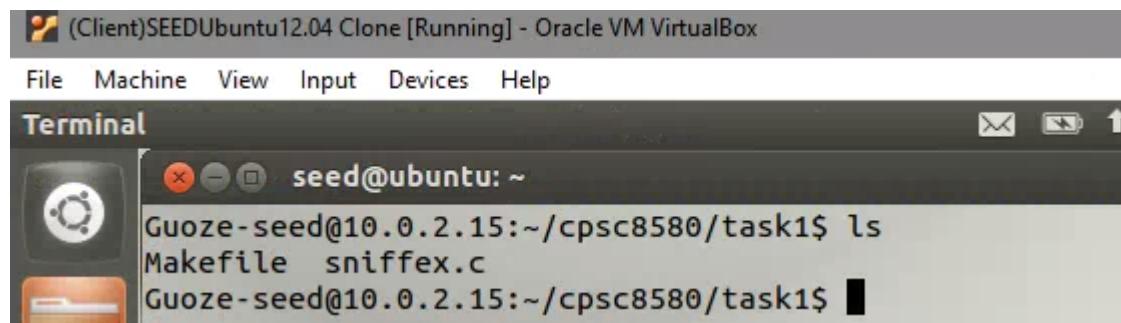
Firstly, we need install the `libpcap` on the Ubuntu. I don't need install it anymore because the VM from the SEED have installed the `libpcap` lib. I just check the lib file to make sure that I can use the `libpcap` lib in the next work.



```
Guoze-seed@10.0.2.15:/usr/lib$ ls -al /usr/local/lib/ | grep libpcap
-rw-r--r-- 1 root root 808912 Aug 14 2013 libpcap.a
lrwxrwxrwx 1 root root      12 Aug 14 2013 libpcap.so -> libpcap.so.1
lrwxrwxrwx 1 root root      16 Aug 14 2013 libpcap.so.1 -> libpcap.so.1.3.0
-rwxr-xr-x 1 root root 603305 Aug 14 2013 libpcap.so.1.3.0
Guoze-seed@10.0.2.15:/usr/lib$
```

### Compile the file

In order to compile the `sniffex.c`, I write a `Makefile` to help me compile it. The `Makefile` file just like the followed.



```
Guoze-seed@10.0.2.15:~/cpsc8580/task1$ ls
Makefile  sniffex.c
Guoze-seed@10.0.2.15:~/cpsc8580/task1$
```

```
CXX = gcc

# Warnings frequently signal eventual errors:
CXXFLAGS= -Wall

OBJS = \
```

```
        sniffex.o
EXEC = run

%.o: %.c
    $(CXX) $(CXXFLAGS) -c $< -o $@

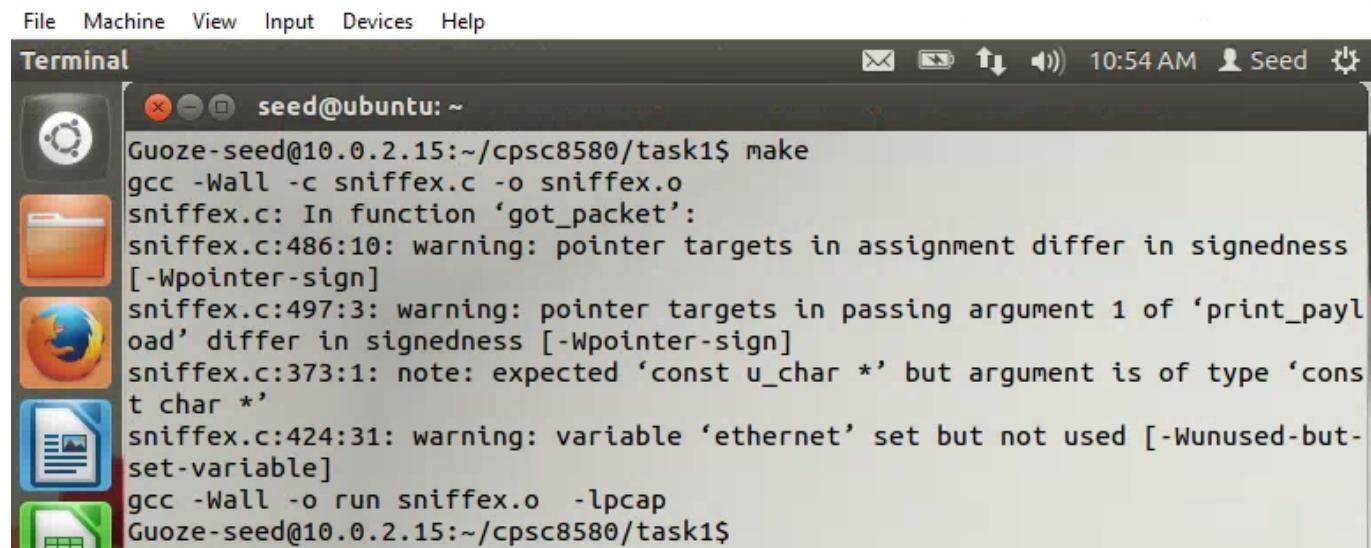
$(EXEC): $(OBJS)
    $(CXX) $(CXXFLAGS) -o $@ $(OBJS) $(LDFLAGS) -lpcap

sniffex.o: sniffex.c

clean:
    rm -rf $(OBJS)
    rm -rf $(EXEC)
```

I use `-lpcap` to assign the library with the `sniffex.o` file.

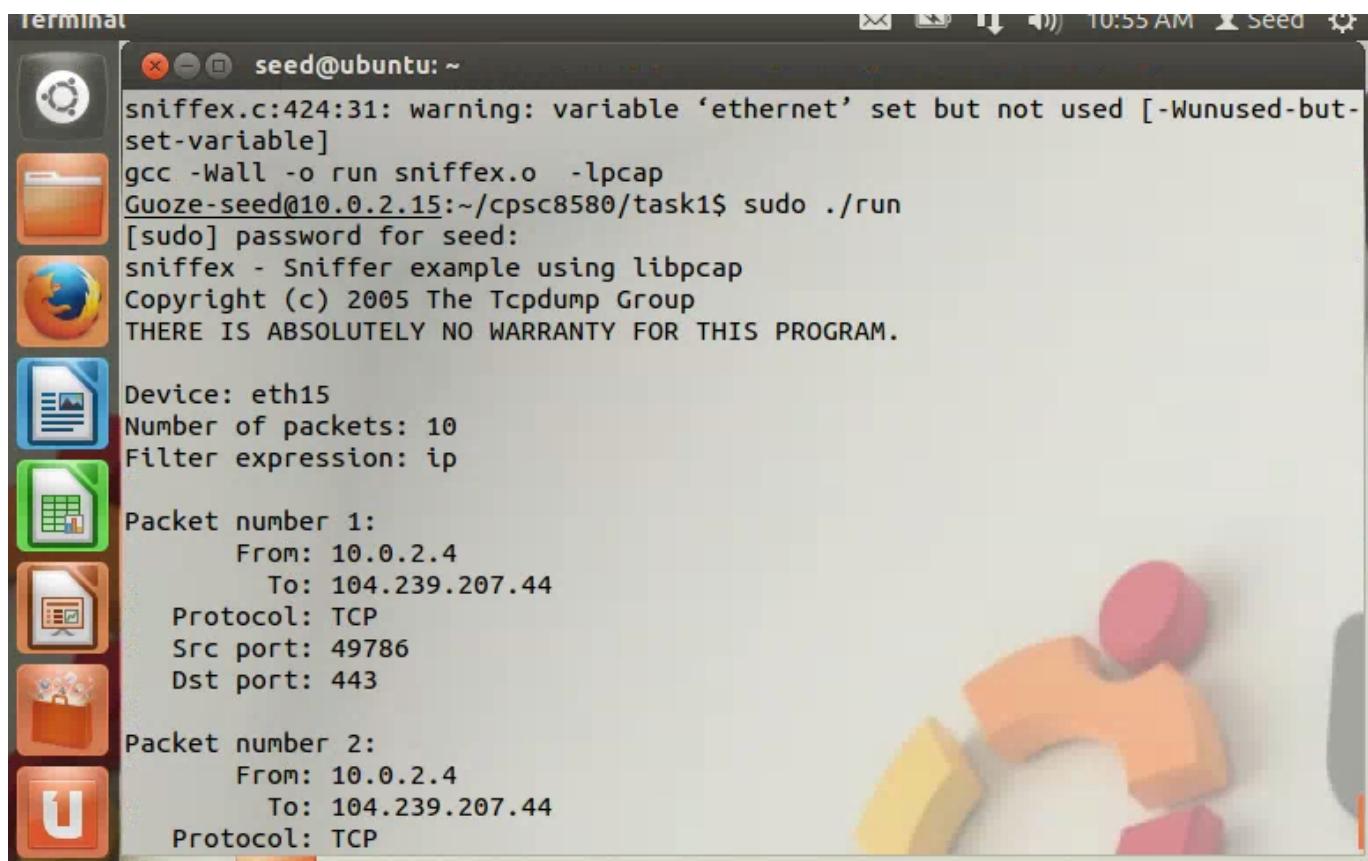
## Output of the compile

A screenshot of a Linux desktop environment, specifically Ubuntu. A terminal window titled "seed@ubuntu: ~" is open. The window shows the command "make" being run in the directory "/cpsc8580/task1". The output of the compilation process is displayed, including several warning messages from the compiler (gcc) about pointer assignments and signedness issues. The terminal window has a standard black background with white text and is located on a desktop with other icons visible in the background.

```
seed@ubuntu: ~
Guoze-seed@10.0.2.15:~/cpsc8580/task1$ make
gcc -Wall -c sniffex.c -o sniffex.o
sniffex.c: In function 'got_packet':
sniffex.c:486:10: warning: pointer targets in assignment differ in signedness
[-Wpointer-sign]
sniffex.c:497:3: warning: pointer targets in passing argument 1 of 'print_payload' differ in signedness [-Wpointer-sign]
sniffex.c:373:1: note: expected 'const u_char *' but argument is of type 'const char *'
sniffex.c:424:31: warning: variable 'ethernet' set but not used [-Wunused-but-set-variable]
gcc -Wall -o run sniffex.o -lpcap
Guoze-seed@10.0.2.15:~/cpsc8580/task1$
```

## Execute the program

Use `sudo ./run` to execute the sniffing program.



A screenshot of a Ubuntu desktop environment. On the left, there's a vertical dock with icons for the Dash, Home, Applications, and a few others. The main window is a terminal window titled 'terminal' with the command 'seed@ubuntu: ~'. The terminal displays the following output:

```
sniffex.c:424:31: warning: variable 'ethernet' set but not used [-Wunused-but-set-variable]
gcc -Wall -o run sniffex.o -lpcap
Guoze-seed@10.0.2.15:~/cpsc8580/task1$ sudo ./run
[sudo] password for seed:
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth15
Number of packets: 10
Filter expression: ip

Packet number 1:
    From: 10.0.2.4
        To: 104.239.207.44
    Protocol: TCP
    Src port: 49786
    Dst port: 443

Packet number 2:
    From: 10.0.2.4
        To: 104.239.207.44
    Protocol: TCP
```

Then, the program will print the information about the device, the number of packets and the information about the packet.

```
Device: eth15
Number of packets: 10
Filter expression: ip
```

These printing information just tell the user about how many packets this sniffing program will be capture. And it just capture the packets from the **Device: eth15**. It will be stop after it have captured 10 packets.

```
Packet number 1:
    From: 10.0.2.4
        To: 104.239.207.44
    Protocol: TCP
    Src port: 49786
    Dst port: 443
```

And, this is the packet information to give us the summary about the **Protocol**, **Src IP**, **Dst IP**, **Sro port**, and **Dst port**.

---

### Problem 1:

Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not detailed explanation like the one in the tutorial.

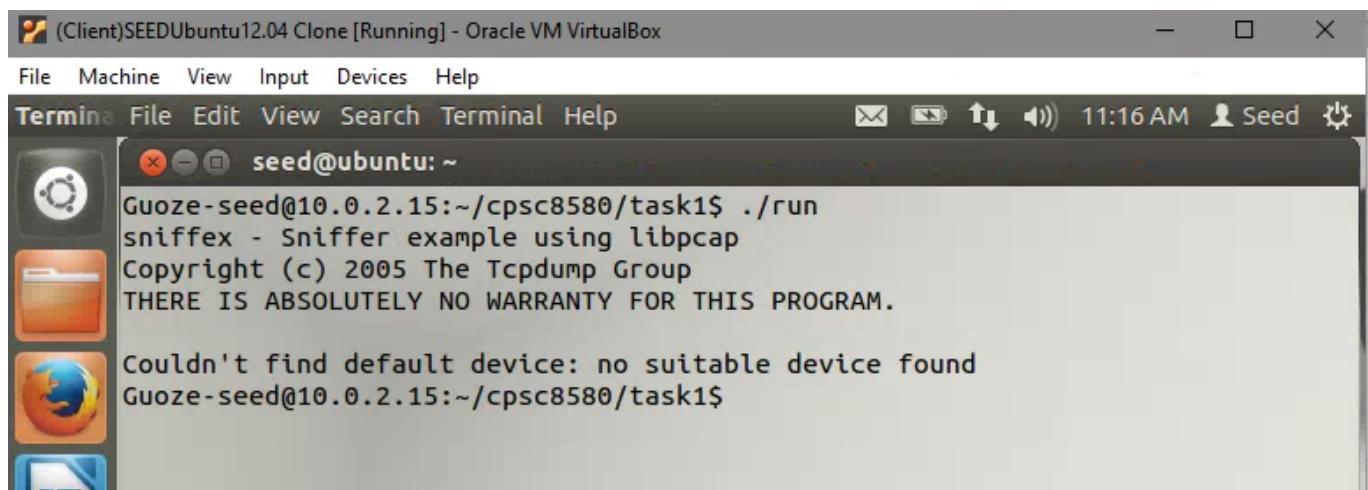
In my opinion, the sequence of the library calls that are essential for sniffer program include the followed steps.

1. Choose the source of the packet. (From device or file) This step uses the `pcap_lookupdev` function in the lib.
2. Open the source device or file and create a sniffing session to record the packets. This step uses the `pcap_t *pcap_open_live` function in the lib.
3. Configure the sniffing session to filter the traffic. This step uses the `int pcap_compile` or `int pcap_setfilter` functions in the lib.

### Problem 2:

Why do you need the root privilege to run sniffex? Where does the program fail if executed without the root privilege?

If we don't use root privilege to run sniffex, the output is `Couldn't find default device: no suitable device found`. Because if you don't use the root privilege, this program can't get the device information from the system by use the call `pcap_lookupdev`.

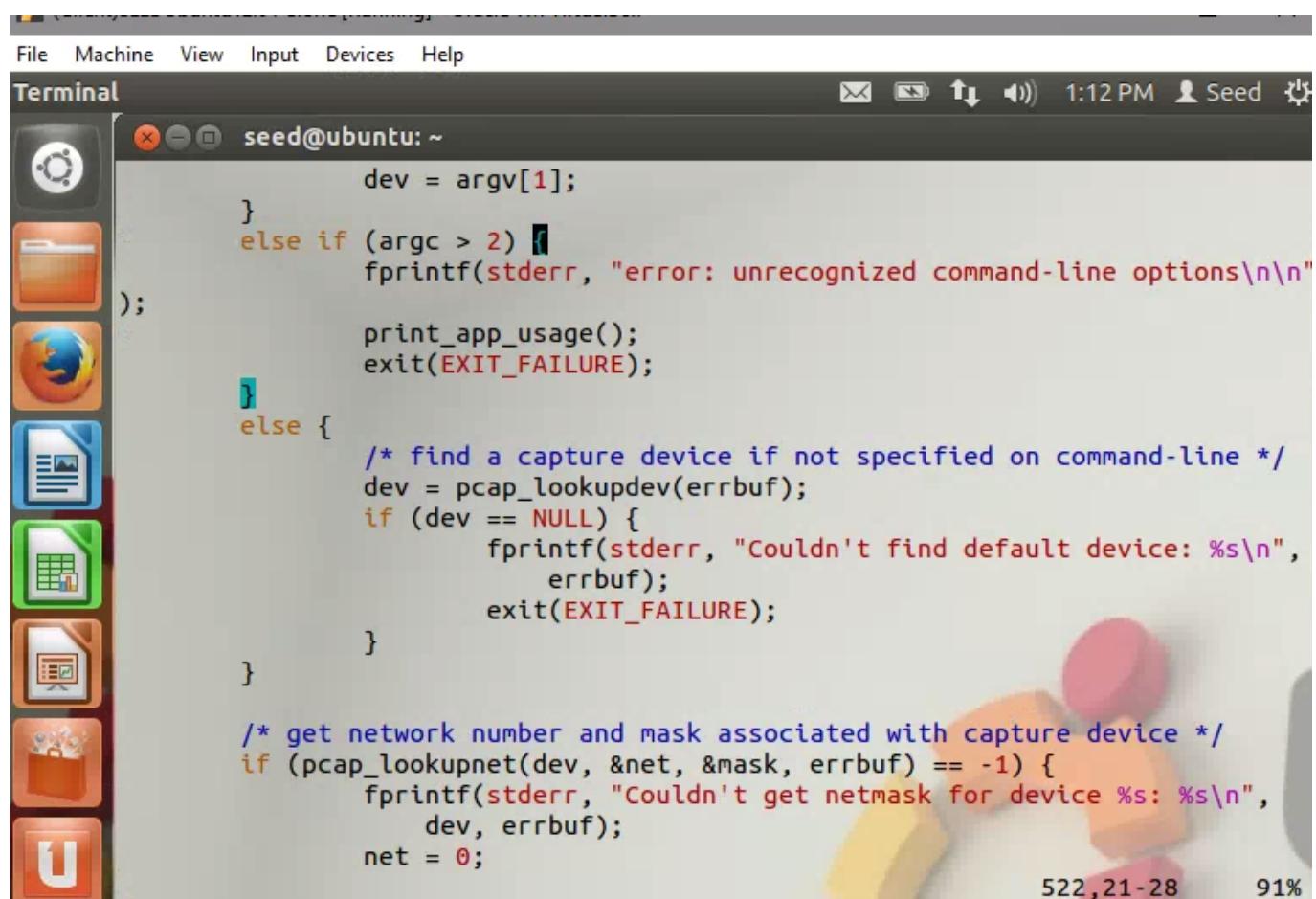


The screenshot shows a terminal window titled "seed@ubuntu: ~". The window contains the following text:

```
Guoze-seed@10.0.2.15:~/cpsc8580/task1$ ./run
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Couldn't find default device: no suitable device found
Guoze-seed@10.0.2.15:~/cpsc8580/task1$
```

As a result, I read the `sniffex.c` in order to find the detailed information about this notification. I found the reason is the `pcap_lookupdev` can't get the device information from the Linux system.



```

File Machine View Input Devices Help
Terminal seed@ubuntu: ~
    dev = argv[1];
}
else if (argc > 2) {
    fprintf(stderr, "error: unrecognized command-line options\n\n");
    print_app_usage();
    exit(EXIT_FAILURE);
}
else {
    /* find a capture device if not specified on command-line */
    dev = pcap_lookupdev(errbuf);
    if (dev == NULL) {
        fprintf(stderr, "Couldn't find default device: %s\n",
                errbuf);
        exit(EXIT_FAILURE);
    }

    /* get network number and mask associated with capture device */
    if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1) {
        fprintf(stderr, "Couldn't get netmask for device %s: %s\n",
                dev, errbuf);
        net = 0;
}
522,21-28 91%

```

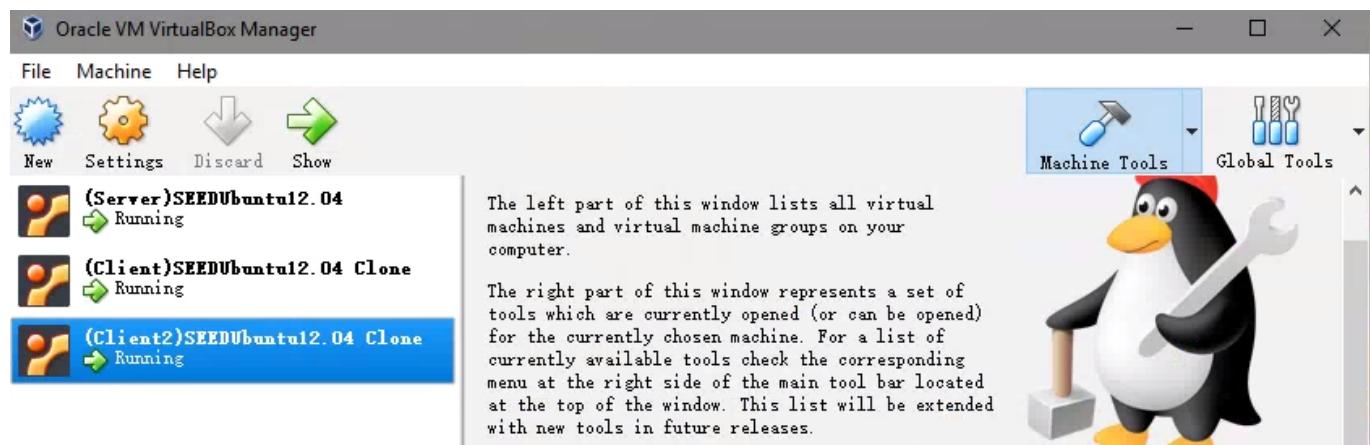
### Problem 3:

Please turn on and turn off the promiscuous mode in the sniffer program. Can you demonstrate the difference when this mode is on and off? Please describe how you demonstrate this.

What is the promiscuous mode?

In a network, promiscuous mode allows a network device to intercept and read each network packet that arrives in its entirety. (From: <https://searchsecurity.techtarget.com/definition/promiscuous-mode>)

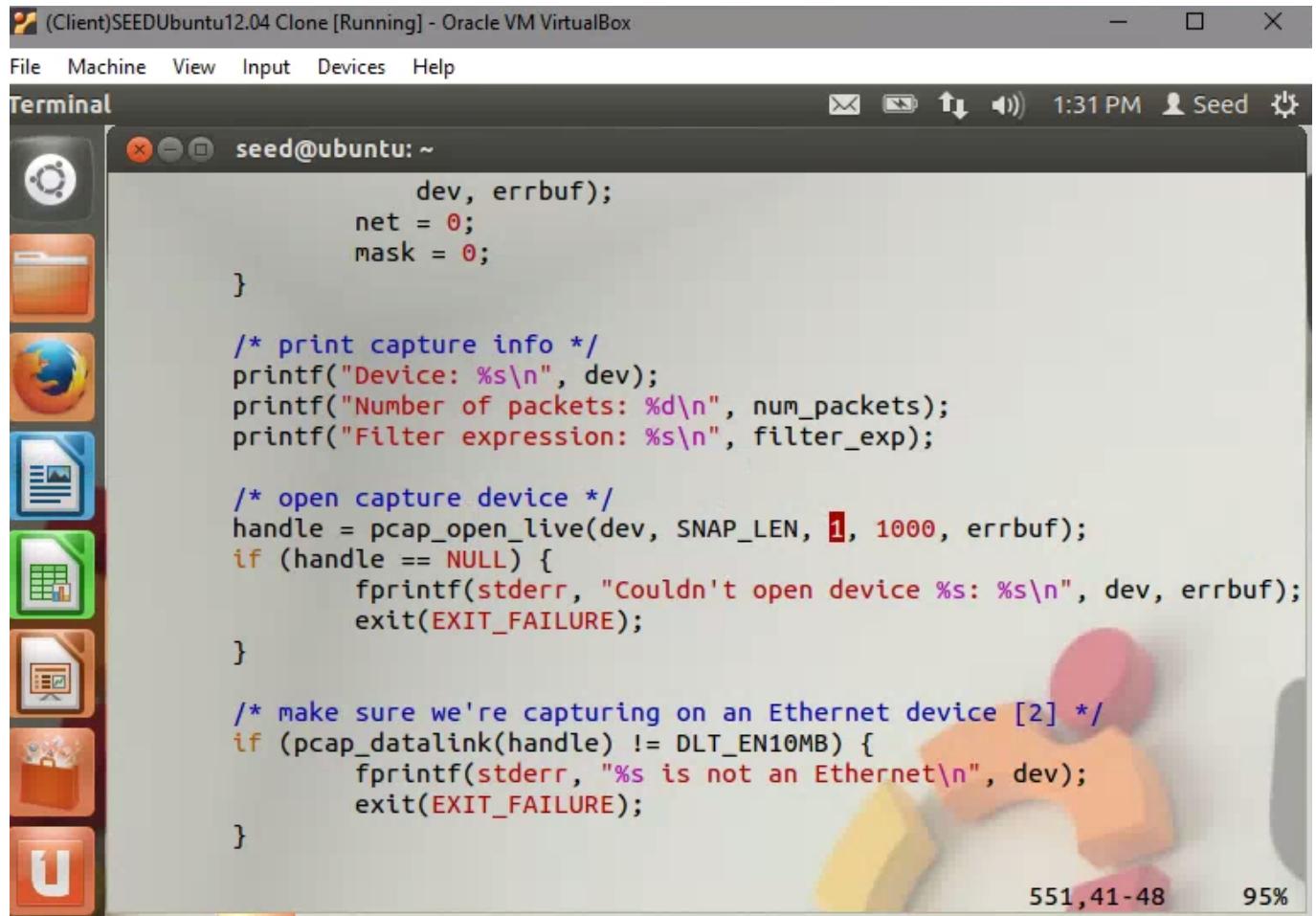
Firstly, I build a local network use three VM Ubuntu. They are the **Server**, **Client**, and **Client2**. Like followed:



And the IP address for these VM are:

```
Server IP: 10.0.2.15  
Client2 IP: 10.0.2.5  
Client IP: 10.0.2.4
```

On the Client VM, I changed the code in the `sniffex.c` to turn on the promiscuous mode.



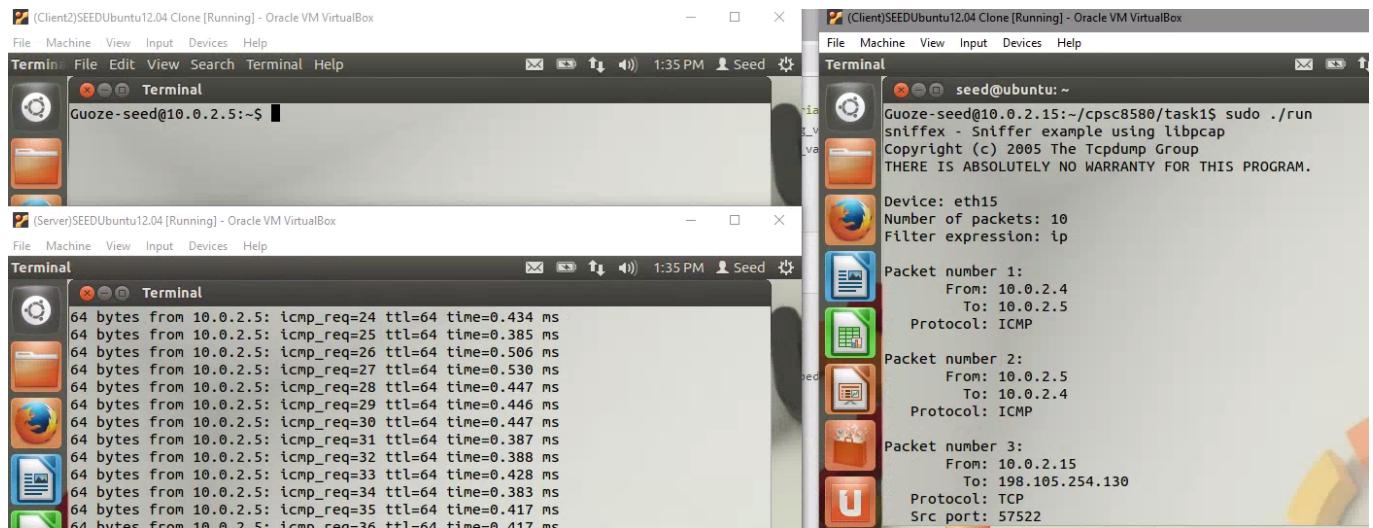
```
seed@ubuntu: ~  
        dev, errbuf);  
    net = 0;  
    mask = 0;  
}  
  
/* print capture info */  
printf("Device: %s\n", dev);  
printf("Number of packets: %d\n", num_packets);  
printf("Filter expression: %s\n", filter_exp);  
  
/* open capture device */  
handle = pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf);  
if (handle == NULL) {  
    fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);  
    exit(EXIT_FAILURE);  
}  
  
/* make sure we're capturing on an Ethernet device [2] */  
if (pcap_datalink(handle) != DLT_EN10MB) {  
    fprintf(stderr, "%s is not an Ethernet\n", dev);  
    exit(EXIT_FAILURE);  
}
```

Just need to change the `third` parameter in the `pcap_open_live` function.

```
handle = pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf);
```

Change it to `1` from `0`.

After that, I use the `Server` to ping the `Client2` VM.



Then, I can find the **Client** VM can capture the network traffic even if only the **Client2** and **Server** VM have a conversation in the network.

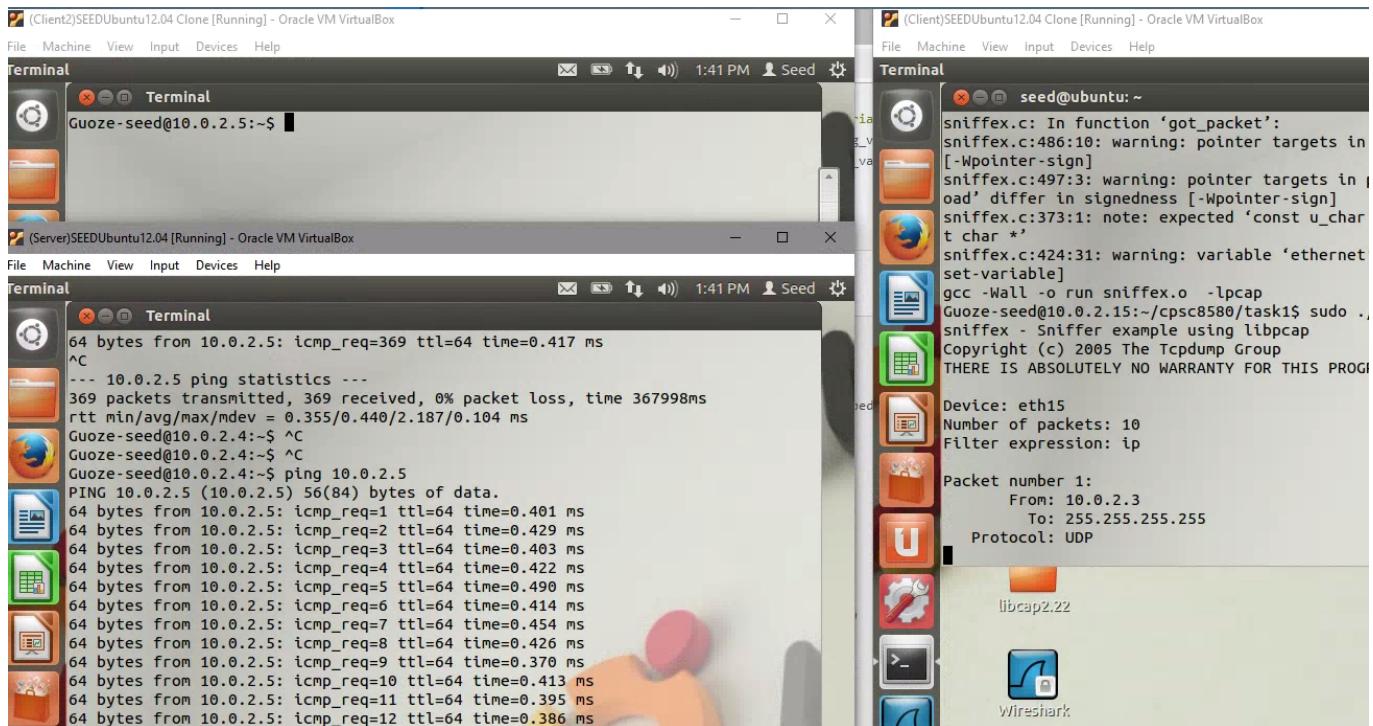
So the promiscuous mode just like you can capture all the network packets in the local network even these packets weren't sent to you.

### Turn of promiscuous mode.

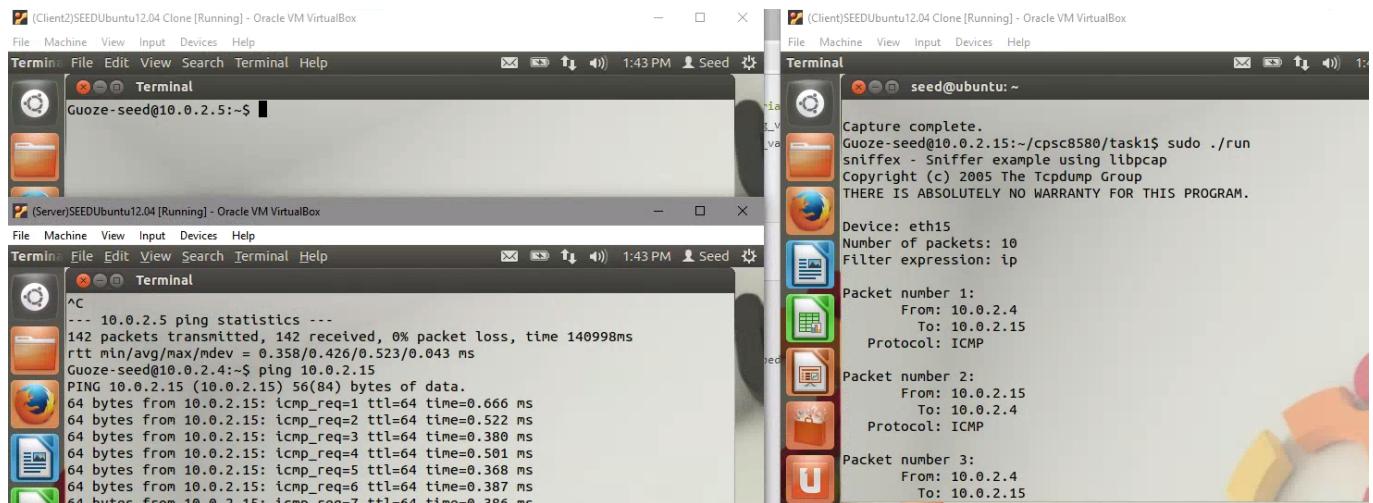
Change to 0 from 1.

```
handle = pcap_open_live(dev, SNAP_LEN, 0, 1000, errbuf);
```

And do the same work again.



I can find, the sniffing program can't capture the packets if these packets don't send to **Client** VM. And It only can capture these packets which was sent to the **Client** VM.



#### Problem 4:

Please write filter expressions to capture each of the followings. In your lab reports, you need to include screendumps to show the results of applying each of these filters.

- Capture the ICMP packets between two specific hosts.
- Capture the TCP packets that have a destination port range from to port 10 - 100.

In this case, I use **Client**, **Client2**, and **Server** VM to do this work.

Server IP: 10.0.2.15  
 Client2 IP: 10.0.2.5  
 Client IP: 10.0.2.4

The sniffing program is executing on the **Client** VM. Use the **Client2** and **Server** VM call **ping** function to send some **ICMP** packets to **Client**.

#### Filter the Src IP

Filter the Src IP: 10.0.2.5(Client2) and Dst IP: 10.0.2.15 (Client) and the **ICMP** packets.

```

seed@ubuntu: ~
    * Print payload data; it might be binary, so don't just
    * treat it as a string.
    */
if (size_payload > 0) {
    printf("    Payload (%d bytes):\n", size_payload);
    print_payload(payload, size_payload);
}

return;
}

int main(int argc, char **argv)
{
    char *dev = NULL;                      /* capture device name */
    char errbuf[PCAP_ERRBUF_SIZE];          /* error buffer */
    pcap_t *handle;                        /* packet capture handle */

    char filter_exp[] = "icmp and src host 10.0.2.5 and dst host 10.0.2.15
";                                /* filter expression [3] */
    struct bpf_program fp;                /* compiled filter program (expression) */
    bpf_u_int32 mask;                     /* subnet mask */

```

I get the output:

```

Guoze-seed@10.0.2.5:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_req=1 ttl=64 time=0.703 ms
64 bytes from 10.0.2.15: icmp_req=2 ttl=64 time=0.456 ms
64 bytes from 10.0.2.15: icmp_req=3 ttl=64 time=0.470 ms
64 bytes from 10.0.2.15: icmp_req=4 ttl=64 time=0.437 ms
64 bytes from 10.0.2.15: icmp_req=5 ttl=64 time=0.467 ms
64 bytes from 10.0.2.15: icmp_req=6 ttl=64 time=0.405 ms
64 bytes from 10.0.2.15: icmp_req=7 ttl=64 time=0.367 ms
64 bytes from 10.0.2.15: icmp_req=8 ttl=64 time=0.420 ms
64 bytes from 10.0.2.15: icmp_req=9 ttl=64 time=0.467 ms
64 bytes from 10.0.2.15: icmp_req=10 ttl=64 time=0.420 ms
64 bytes from 10.0.2.15: icmp_req=11 ttl=64 time=0.422 ms

```

```

set-variable]
gcc -Wall -fPIC -o sniffex.o -lpcap
Guoze-seed@10.0.2.15:~/cpsc8580/task1$ sudo ./run
Sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth15
Number of packets: 10
Filter expression: icmp and src host 10.0.2.5 and dst host 10.0.2.15

Packet number 1:
    From: 10.0.2.5
    To: 10.0.2.15
    Protocol: ICMP

Packet number 2:
    From: 10.0.2.5
    To: 10.0.2.15
    Protocol: ICMP

Packet number 3:
    From: 10.0.2.5
    To: 10.0.2.15

```

Even through the **Server** VM send some packets to **Client**, but the sniffing program doesn't capture these packets.

**Capture the TCP packets that have a destination port range from to port 10 - 100.**

Filter the **dst portrange 10-100** and the **TCP** packets.

```

printf("  Payload (%d bytes):\n", size_payload);
print_payload(payload, size_payload);
}

return;
}

int main(int argc, char **argv)
{
    char *dev = NULL;                      /* capture device name */
    char errbuf[PCAP_ERRBUF_SIZE];          /* error buffer */
    pcap_t *handle;                        /* packet capture handle */

    char filter_exp[] = "tcp dst portrange 10-100";           /* filter expr
ession [3] */
    struct bpf_program fp;                  /* compiled filter program (ex
pression) */
    bpf_u_int32 mask;                     /* subnet mask */
    bpf_u_int32 net;                      /* ip */
    int num_packets = 10;                 /* number of packets to captur
e */
-- INSERT --

```

510,47-54 87%

In this case, I need to make some **TCP** packets. So I just use the **wget** command to download a web page. And the web server will send this web page by FTP packets to me.

**wget http://guozet.me/post/C++-initialization-of-variable/**

```

seed@ubuntu:~$ wget http://guozet.me/post/C++-initialization-of-variable/
--2018-09-19 14:09:13-- http://guozet.me/post/C++-initialization-of-variable/
Resolving guozet.me (guozet.me)... 192.30.252.154, 192.30.252.153
Connecting to guozet.me (guozet.me)|192.30.252.154|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55649 (54K) [text/html]
Saving to: `index.html'

100%[=====] 55,649      --K/s   in 0.09s

2018-09-19 14:09:13 (615 KB/s) - `index.html' saved [55649/55649]

Guoze-seed@10.0.2.15:~$ 

```