

【2022 Advanced Computer Networks Homework 7】

Rules:

1. Please finish this homework under Ubuntu 22.04 OS system.
2. You must use C language to complete this homework with **makefile**.
3. You are not allowed to copy the other's homework. Otherwise, you will fail this course.
4. Compress this assignment to a tar file, and name it "StudentID_TCPIP_HW7.tar.gz" (e.g., M063040001_TCPIP_HW7.tar.gz) and submit it to the Cyber University system <http://cu.nsysu.edu.tw/> before the deadline.
5. If you have any question, please send email to TA at net_ta@net.nsysu.edu.tw or drop by Room EC5018 in the duration of 11:00 to 17:00. However, TA will not help you to debug program.
6. You have to deeply understand what your program do because TA will ask you about your program during demo.
7. Deadline: **2022/12/5 23:59**. **No delay is accepted**. If you have any trouble, please notify us in advance by email.

Homework 7: subnet IP scanner.

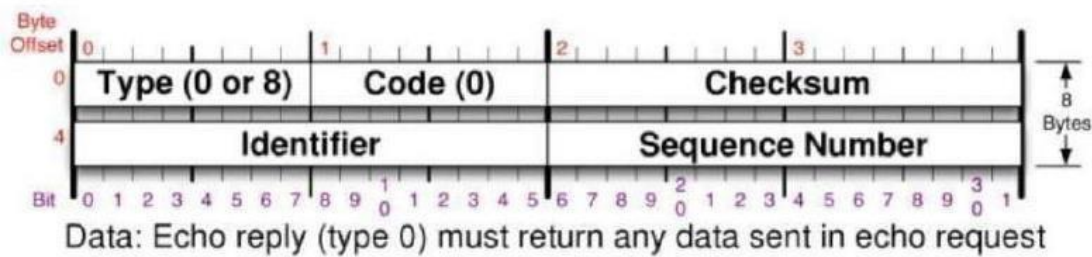
Request:

The scanner sends ICMP echo request to all the other subnet IP addresses. For example, let your host's IP address is 140.117.171.148 and netmask is 255.255.255.0. ICMP echo request (type 8) is sent to 140.117.171.1 ~ 140.117.171.254 (except itself) respectively, and the ICMP echo reply (type 0) is caught if the host is alive on the subnet. This is assuming that ICMP is enabled (Ubuntu 20.04 enable by default). Then the hosts will automatically send ICMP echo reply when ICMP echo requests are received. The IP header TTL (Time-to-Live) field of ICMP echo request must set to 1. Thus, the ICMP echo request will be confined within its subnet.

Send ICMP echo request packet :

Fill the IP header according to the following format:

1. Header length = calculate by yourself
2. Total length = calculate by yourself
3. Id = 0
4. Flag = don't fragment
5. TTL = 1
6. Protocol = ICMP
7. Checksum (You can let OS do it.)



Fill the ICMP packet according to the following format:

1. Checksum (You can let OS do it.)
2. ID: process id
3. Sequence number: Starting from 1, increase one by one.
4. Data : Your Student ID. Note that data size must correspond to IP header.

Receive ICMP echo reply packet :

When receiving ICMP echo reply, you can use the standard socket like the one in the ARP homework or use libpcap (*sudo apt-get install libpcap-dev*) to implement it. libpcap is used in Wireshark and tcpdump. You can set filter to catch expected packets from all incoming ones. You can use tcpdump to test your filter rules. The rules you use on tcpdump basically can also use on libcap. Maybe there is a little difference, but it looks similar. You should use the filter to minimize the number of packets received.

For ICMP echo reply, please check the following fields:

1. The source in IP header
2. ICMP type
3. The ID in ICMP message is matching what you have set.
4. The sequence number in ICMP message is the same as echo request.

Part of libcap initialization procedure is already in the file we provide. You just need to set the suitable filter rules.

A sent packet must use the socket which is like the one in the following figure.

```
if((sockfd = socket(AF_INET, SOCK_RAW , IPPROTO_RAW)) < 0)
{
    perror("socket");
    exit(1);
}

if(setsockopt( sockfd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0)
{
    perror("setsockopt");
    exit(1);
}

/*
 * Use "sendto" to send packets, and use "pcap_get_reply"(in pcap.c)
 * or use the standard socket like the one in the ARP homework
 * to get the "ICMP echo response" packets
 * You should reset the timer every time before you send a packet.
 */
if(sendto(sockfd, packet, PACKET_SIZE, 0, (struct sockaddr *)&dst, sizeof(dst)) < 0)
{
    perror("sendto");
    exit(1);
}

free(packet);
```

File provide :

fill_packet.c fill_packet.h

pcap.c pcap.h

main.c

ping.c (Original Ping program on Ubuntu, you can reference it. Complete package :

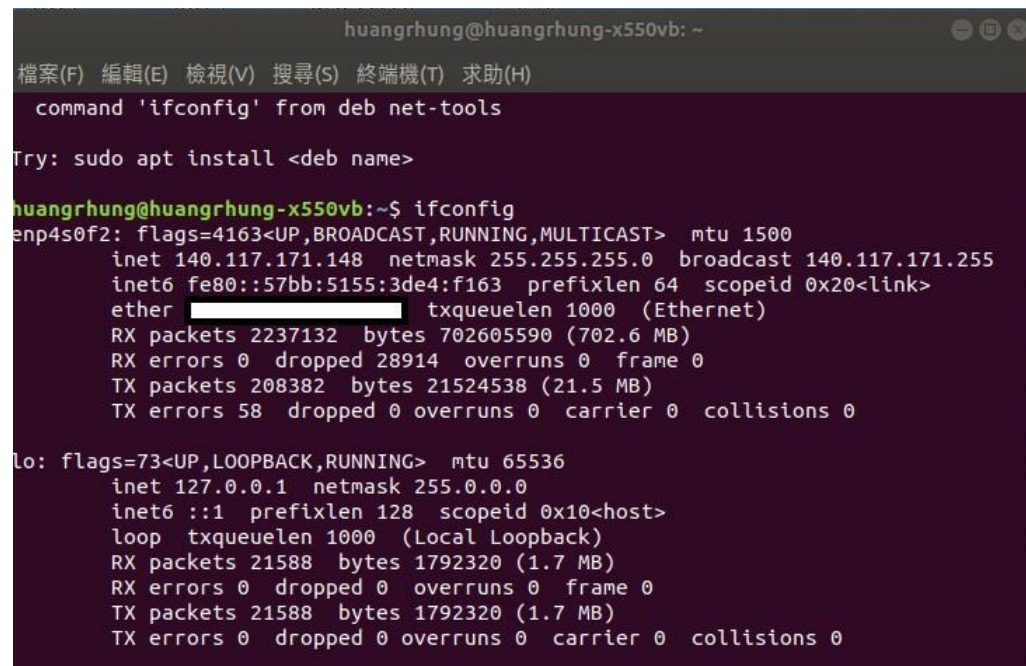
<http://www.skbuff.net/iputils>)

Input Usage

usage:

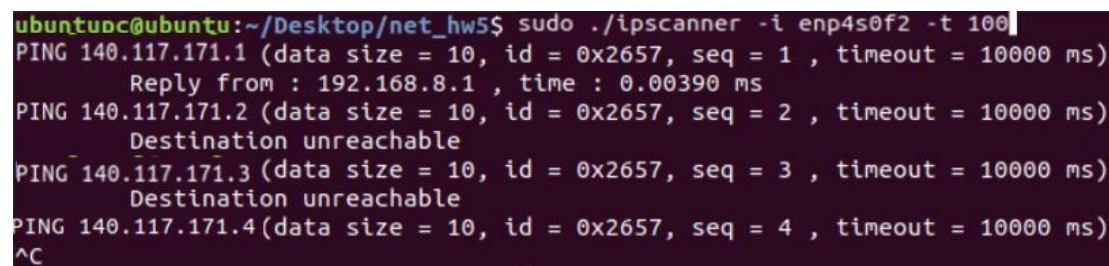
```
# sudo ./ipscanner -i [Network Interface Name] -t [timeout(ms)]
```

Your program must automatically get host IP address and netmask on the Network Interface which is provided by user.



```
huangrhung@huangrhung-x550vb: ~  
command 'ifconfig' from deb net-tools  
Try: sudo apt install <deb name>  
huangrhung@huangrhung-x550vb:~$ ifconfig  
enp4s0f2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 140.117.171.148 netmask 255.255.255.0 broadcast 140.117.171.255  
    inet6 fe80::57bb:5155:3de4:f163 prefixlen 64 scopeid 0x20<link>  
    ether XXXXXXXXXX txqueuelen 1000 (Ethernet)  
    RX packets 2237132 bytes 702605590 (702.6 MB)  
    RX errors 0 dropped 28914 overruns 0 frame 0  
    TX packets 208382 bytes 21524538 (21.5 MB)  
    TX errors 58 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 21588 bytes 1792320 (1.7 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 21588 bytes 1792320 (1.7 MB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

The Ubuntu host network interface enp4s0f2 with IP address 140.117.171.148 and netmask is 255.255.255.0, so the program will scan 140.117.171.1~140.117.171.254



```
ubuntu@ubuntu:~/Desktop/net_hw$ sudo ./ipscanner -i enp4s0f2 -t 100  
PING 140.117.171.1 (data size = 10, id = 0x2657, seq = 1 , timeout = 10000 ms)  
    Reply from : 192.168.8.1 , time : 0.00390 ms  
PING 140.117.171.2 (data size = 10, id = 0x2657, seq = 2 , timeout = 10000 ms)  
    Destination unreachable  
PING 140.117.171.3 (data size = 10, id = 0x2657, seq = 3 , timeout = 10000 ms)  
    Destination unreachable  
PING 140.117.171.4 (data size = 10, id = 0x2657, seq = 4 , timeout = 10000 ms)  
^C
```

There are one host, for example, with IP 140.117.171.1 which sends ICMP echo reply.

Packet Description

63 33.002971	D-LinkIn_03:9f:1e	AsustekC_b1:b1:75	ARP	60 Who has 140.117.171.1? Tell 140.117.171.148
64 33.003001	AsustekC_b1:b1:75	D-LinkIn_03:9f:1e	ARP	42 140.117.171.1 is at ac:22:0b:b1:b1:75

No.63-64 :

The socket will automatically find the MAC address of 140.117.171.1. So it sends an ARP request.140.117.171.1 sends ARP reply and make the host know its MAC address.

```

> Internet Protocol Version 4, Src: 140.117.171.148 Dst: 140.117.171.1
  v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4d5a [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  > [No response seen]
  v Data (32 bytes)
    Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
    [Length: 32]

```

The ICMP packet:

Type : 8

Id : 0x256

Seq : 1

Data : Your Student ID are Hex ,(You must change it to your student ID).

20	16.603378	140.117.171.148	140.117.171.3	ICMP	74 Echo (ping) request	id=0x0001, seq=20/5120, ttl=128 (no response found!)
37	21.464421	140.117.171.148	140.117.171.3	ICMP	74 Echo (ping) request	id=0x0001, seq=21/5376, ttl=128 (no response found!)
45	26.463745	140.117.171.148	140.117.171.3	ICMP	74 Echo (ping) request	id=0x0001, seq=22/5632, ttl=128 (no response found!)

The socket will find the MAC address of 140.117.171.1. So it sends an ARP request.

But there is no any reply and it gets an ICMP message of no response found or other reason.