

一. 实验目的

设计和实现一个简单的静态路由机制，用以取代 Linux 实现的静态路由方式，进而加深对二三层协议衔接及静态路由的理解。

二. 数据结构说明

```
struct ICMPHeader
{
    unsigned char type;//icmp type
    unsigned char code;//icmp code
    unsigned short checksum;//check sum
    struct{
        unsigned short id;
        unsigned short sequence;
    }echo;
    unsigned char data[0];//ICMP data parts
};
```

这是 icmp 包头结构体

成员解释：type 表示 icmp 类型，code 默认为 0，checksum 校验和，id 和 sequence 表示验证码和序列号、data 表示数据区

```
//ip head
struct IPHeader
{
    unsigned char headerLen_version;
    unsigned char tos; //service type
    unsigned short totalLen; //total length
    unsigned short id; //tag
    unsigned short flagOffset; //3 bits flag+13 bits offset
    unsigned char ttl; //time to live
    unsigned char protocol; //protocol
    unsigned short checksum; //check sum
    unsigned int srcIP; //source ip address
    unsigned int dstIP; //destination ip address
};
```

这是 ip 报文头结构

成员解释：headerlen_version 4 个字节为长度，4 个字节为版本号，tos 为服务类型、totalLen 为 ip 包总长，id 为标记号、flagOffset 为段号和偏移量（不分段）、ttl 生存期、protocol 为所携带的协议、checksum 为校验和、srcIP 为源 ip、dstIP 为目的 ip

```
typedef struct Arphdr{
    unsigned short hardware_type;
    unsigned short protocol_type;
    unsigned char hardware_address_len;
    unsigned char protocol_address_len;
    unsigned short operation_field;
    unsigned char src_mac_addr[6];
    unsigned char src_ip_addr[4];
    unsigned char dest_mac_addr[6];
    unsigned char dest_ip_addr[4];
}ARP_HEADER;
```

这是 arp 包头,

成员解释: hardware_type 表示硬件类型、protocol_type 表示协议类型、hardware_address_len 表示硬件地址长度、protocol_address_len 表示协议地址长度、operation_field 表示操作类型、src_mac_addr[6]表示源 mac 地址、src_ip_addr 表示源网络地址、dest_mac_addr 表示目的 mac 地址, dest_ip_addr 表示目的网络地址

```
typedef struct MACADDR{
    uint8_t mac[6];
}MacAddr;
```

表示 mac 地址的结构体, 共 6 个字节, uint_8 的数组表示, 数组长为 6

```
struct ARPITEM{
    uint32_t ip;
    MacAddr mac_addr;
}ArpTable[MAX_ARP_NUM];
```

表示 ARP 映射表的结构体, 每个表项由 32 位的 ip 和 mac 结构体构成, 表是一个数组

```
typedef struct LOCALINFO{
    uint32_t ip;
    uint32_t netmask;
    uint32_t gateway;//ip of gateway
    char interface[10];
}LocalInfo;
```

本机信息的结构体, 由本机 ip、子网掩码、网关 ip、网卡端口名称构成

三. 程序设计思路

答: 程序分成四个源文件, 分别为 ping.cpp、router1.c、router2.c、reply.c 分别运行于 pc0、router1、router2、pc1 四个虚拟机上。

- (1)pc0 需要完成的工作是: 根据输入的目标 ip 发送结构内容均正确的 ping 请求包
- (2)router1 需要完成的工作是: 正确接收有效的以太网帧, 合理修改包中个别内容并转发到正确端口
- (3)router2 需要完成的工作同 router1, 分别为修改包, 并转发

(4)pc1 需要完成的工作为收到 ping 请求包，发送对应的 ping 回应包

一次 ping 的过程的具体的细节如下：

假设网络拓扑为主机 A-R1-R2-主机 B

A: 192.168.2.1

R1: 192.168.2.2 192.168.3.1

R2: 192.168.3.2 192.168.4.1

B: 192.168.4.2

由主机 A 发送到主机 B，且路由器已经填充正确的路由表，为了简化，ARP 表都已经初始化

(1) 主机 A 封装报文，发现目的地址与自身不在同一网段，将报文转发给默认网关，查 ARP 表，封装报文，目的 IP 为主机 B，目的 MAC 为 R1 的 MAC 地址。

(2) R1 接收到 IP 报文，查路由表，发现须经 R2 转发，查 ARP 表，转发 ip 报文，修改 IP 报文源 MAC 地址为自身，目的 MAC 地址为 R2，并将报文转发至 R2。

(3) R2 收到报文，查路由表，发现就是下一跳，查 ARP 表，R2 转发 ip 报文，修改 IP 报文源 MAC 地址为自身，目的 MAC 地址为主机 B，并将报文转发至主机 B

(4) 主机 B 收到报文，解析报文，发现是 icmp 请求包，发送 icmp 回应包给 R2，目的 ip 为请求包的源 ip，目的 MAC 为 icmp 请求包的 MAC

(5) R2 收到报文，查路由表，下一跳是 R1,查 ARP 表，R2 转发 ip 报文，修改 IP 报文源 MAC 地址为自身，目的 MAC 地址为 R1，并将报文转发至 R1

(6) R1 接收到 IP 报文，查路由表，发现就是下一跳，查 ARP 表，转发 ip 报文，修改 IP 报文源 MAC 地址为自身，目的 MAC 地址为 A，并将报文转发至 A。

(7) A 在有限时间内接收到回应包，完成一次 ping 的过程

四．程序运行流程说明

[主机 A]

- 1.主机 A 初始化 arp 表、本地信息
- 2.主机 A 判断目的 ip 192.168.4.2 和本机 192.168.2.1 不在一个子网
- 3.主机 A 查本地信息获得网关 ip 和网卡接口，查 arp 表获得网关 mac
- 4.进入 while 循环发送请求包
- 5.配置 sockaddr_ll 的网卡出口名称、获取源 mac 地址等本地网卡信息
- 6.依次填写以太网帧、ip 报文、icmp 包，并发出，记录发出的时间
- 7.进入 receive_icmp 函数在发出时间后等待一段时间（非阻塞），检查有无接受包
- 8.检查到正确的回应包后解析字段，否则输出未收到包
- 9.循环执行 4-8 步骤，反复发出包

[路由器]

1. 初始化路由表、arp 表
2. 循环接受包，如目的 mac 是自己或广播，则进行 3
3. 解析包的目的 ip，查路由表获得下一跳 ip，查 arp 表获得下一跳 mac，修改源 mac 为自己，目的 mac 为下一跳 mac，把 ttl 减一并重新计算校验和，转发
4. 重复 2-3 步骤

[B 主机]

1. 初始化路由表、arp 表
2. 循环接受包，如目的 mac 是自己则进行 3
3. 解析包的类型，记录抓包所在网卡名称，如果目的 ip 是自己且是 icmp 请求包则进行 4
4. 填 icmp 回应包，目的 mac 为请求包源 mac，目的 ip 为请求包源 ip，icmp 回应类型为 reply，计算校验和并发送给之前的网卡
5. 重复 2-4 步骤

五．运行结果截图

(一) 程序效果

(1) Pc0:

```
user1@ubuntu:~/woozie/lab4$ sudo ./ping 192.168.4.2
PING 192.168.4.2 56(84) bytes of data.
64 bytes from 2.0.0.0: icmp_seq=0 ttl=253 time=1.014 ms
64 bytes from 2.0.0.0: icmp_seq=1 ttl=253 time=0.765 ms
64 bytes from 2.0.0.0: icmp_seq=2 ttl=253 time=3.690 ms
64 bytes from 2.0.0.0: icmp_seq=3 ttl=253 time=4.416 ms
64 bytes from 2.0.0.0: icmp_seq=4 ttl=253 time=3.779 ms
64 bytes from 2.0.0.0: icmp_seq=5 ttl=253 time=1.956 ms
64 bytes from 2.0.0.0: icmp_seq=6 ttl=253 time=0.971 ms
64 bytes from 2.0.0.0: icmp_seq=7 ttl=253 time=2.486 ms
64 bytes from 2.0.0.0: icmp_seq=8 ttl=253 time=1.225 ms
64 bytes from 2.0.0.0: icmp_seq=9 ttl=253 time=1.047 ms
64 bytes from 2.0.0.0: icmp_seq=10 ttl=253 time=1.107 ms
64 bytes from 2.0.0.0: icmp_seq=11 ttl=253 time=2.854 ms
64 bytes from 2.0.0.0: icmp_seq=12 ttl=253 time=2.740 ms
64 bytes from 2.0.0.0: icmp_seq=13 ttl=253 time=2.444 ms
64 bytes from 2.0.0.0: icmp_seq=14 ttl=253 time=3.045 ms
64 bytes from 2.0.0.0: icmp_seq=15 ttl=253 time=6.194 ms
64 bytes from 2.0.0.0: icmp_seq=16 ttl=253 time=5.251 ms
64 bytes from 2.0.0.0: icmp_seq=17 ttl=253 time=2.782 ms
64 bytes from 2.0.0.0: icmp_seq=18 ttl=253 time=5.767 ms
64 bytes from 2.0.0.0: icmp_seq=19 ttl=253 time=6.453 ms
^A64 bytes from 2.0.0.0: icmp seq=20 ttl=253 time=2.720 ms
```

(2) Router (兼抓包功能)

```
(Ip analyse)
head length of ip:0x14(20) bytes
version id:0x4
tos:0
total len:0x54
id:0x0
flag:0x0
ttl:0xff
protocol:0x1 icmp)
check sum:0x5534
IP:192.168.2.1 ==> 192.168.4.2
*****
 icmp analysis)
type:0x8
code:0x0
check sum:0xf0de
identity:0x6ea
sequent num:0x37
*****
=====
<end>

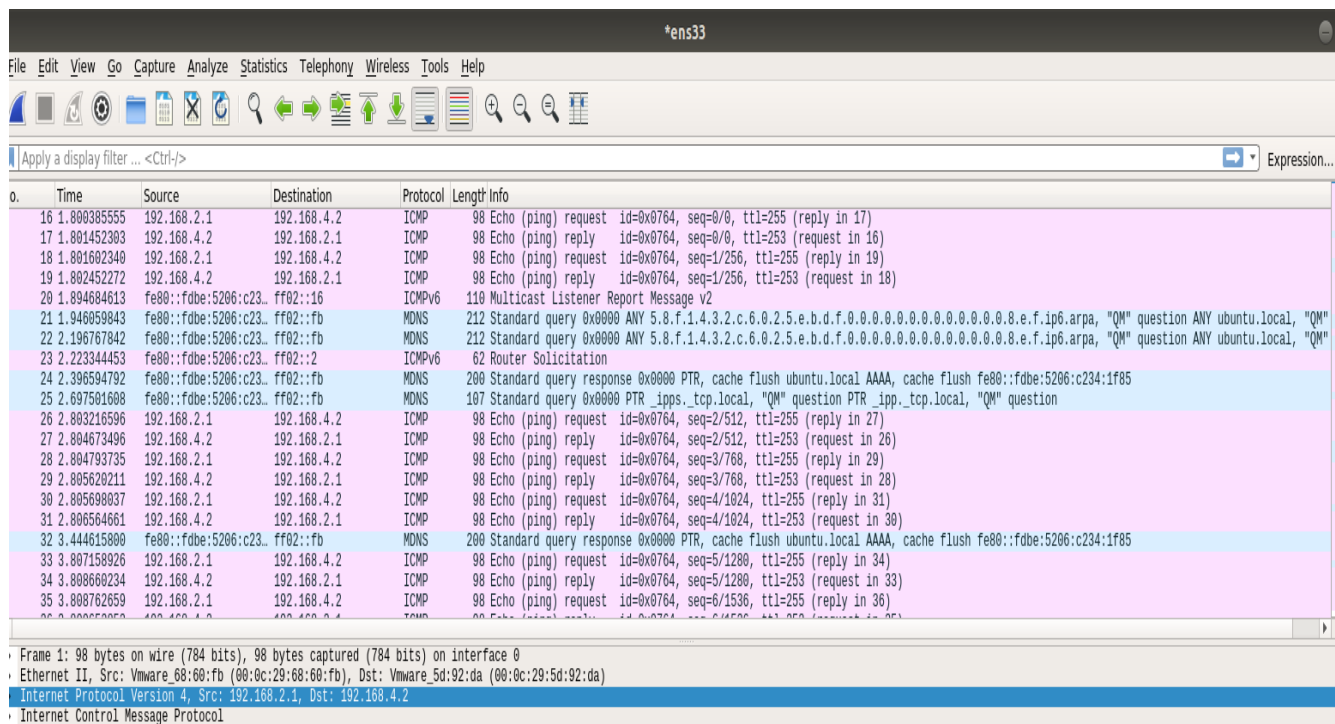
<start>
=====
*****
an ether frame from interface ens38
*****
(Ip analyse)
head length of ip:0x14(20) bytes
version id:0x4
tos:0
total len:0x54
id:0x0
flag:0x0
ttl:0xfe
protocol:0x1 icmp)
check sum:0x5535
IP:192.168.4.2 ==> 192.168.2.1
```

(3) Pc1

:

```
<start>
=====
*****
an ether frame from interface ens39
*****
(ip analyse)
head length of ip:0x14(20) bytes
version id:0x4
tos:0
total len:0x54
id:0x0
flag:0x0
ttl:0xfd
protocol:0x1 icmp
check sum:0x5536
IP:192.168.2.1 ==> 192.168.4.2
*****
 icmp analysis)
type:0x8
code:0x0
check sum:0xf0db
identity:0x6ea
sequent num:0x3a
*****
=====
<end>
```

(二) wireshark 抓包效果

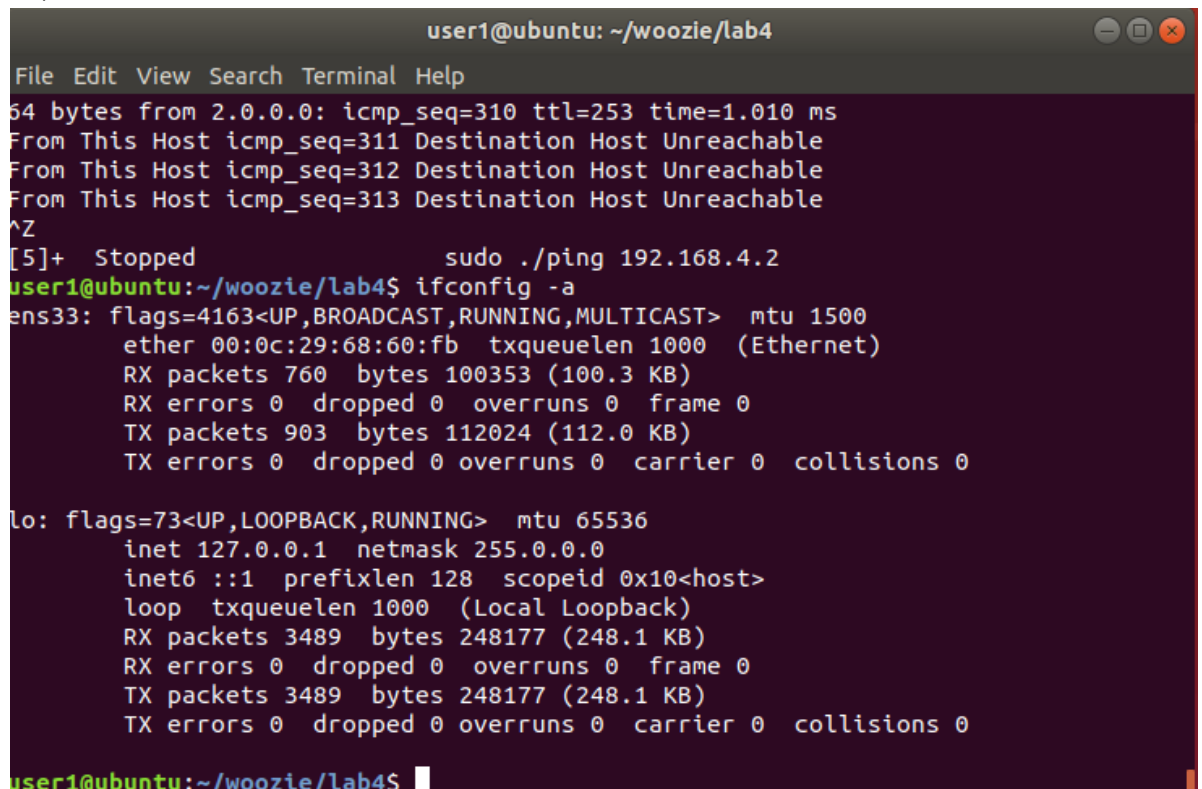


No.	Time	Source	Destination	Protocol	Length	Info
16	1.800385555	192.168.2.1	192.168.4.2	ICMP	98	Echo (ping) request id=0x0764, seq=0/0, ttl=255 (reply in 17)
17	1.801452383	192.168.4.2	192.168.2.1	ICMP	98	Echo (ping) reply id=0x0764, seq=0/0, ttl=253 (request in 16)
18	1.801602340	192.168.2.1	192.168.4.2	ICMP	98	Echo (ping) request id=0x0764, seq=1/256, ttl=255 (reply in 19)
19	1.802452272	192.168.4.2	192.168.2.1	ICMP	98	Echo (ping) reply id=0x0764, seq=1/256, ttl=253 (request in 18)
20	1.894684613	fe80::fdba:5206:c23...ff02::16		ICMPv6	110	Multicast Listener Report Message v2
21	1.946059843	fe80::fdba:5206:c23...ff02::fb		MDNS	212	Standard query 0x0000 ANY 5.8.f.1.4.3.2.c.6.0.2.5.e.b.d.f.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa, "QM" question ANY ubuntu.local, "QM"
22	2.196767842	fe80::fdba:5206:c23...ff02::fb		MDNS	212	Standard query 0x0000 ANY 5.8.f.1.4.3.2.c.6.0.2.5.e.b.d.f.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa, "QM" question ANY ubuntu.local, "QM"
23	2.223344453	fe80::fdba:5206:c23...ff02::2		ICMPv6	62	Router Solicitation
24	2.396594792	fe80::fdba:5206:c23...ff02::fb		MDNS	200	Standard query response 0x0000 PTR, cache flush ubuntu.local AAAA, cache flush fe80::fdba:5206:c23:1f85
25	2.697581608	fe80::fdba:5206:c23...ff02::fb		MDNS	107	Standard query 0x0000 PTR _ipps_tcp.local, "QM" question PTR _ipp_tcp.local, "QM" question
26	2.803216596	192.168.2.1	192.168.4.2	ICMP	98	Echo (ping) request id=0x0764, seq=2/512, ttl=255 (reply in 27)
27	2.804673496	192.168.4.2	192.168.2.1	ICMP	98	Echo (ping) reply id=0x0764, seq=2/512, ttl=253 (request in 26)
28	2.804793735	192.168.2.1	192.168.4.2	ICMP	98	Echo (ping) request id=0x0764, seq=3/768, ttl=255 (reply in 29)
29	2.805620211	192.168.4.2	192.168.2.1	ICMP	98	Echo (ping) reply id=0x0764, seq=3/768, ttl=253 (request in 28)
30	2.805698037	192.168.2.1	192.168.4.2	ICMP	98	Echo (ping) request id=0x0764, seq=4/1024, ttl=255 (reply in 31)
31	2.806564061	192.168.4.2	192.168.2.1	ICMP	98	Echo (ping) reply id=0x0764, seq=4/1024, ttl=253 (request in 30)
32	3.444615800	fe80::fdba:5206:c23...ff02::fb		MDNS	200	Standard query response 0x0000 PTR, cache flush ubuntu.local AAAA, cache flush fe80::fdba:5206:c23:1f85
33	3.807158926	192.168.2.1	192.168.4.2	ICMP	98	Echo (ping) request id=0x0764, seq=5/1280, ttl=255 (reply in 34)
34	3.808660234	192.168.4.2	192.168.2.1	ICMP	98	Echo (ping) reply id=0x0764, seq=5/1280, ttl=253 (request in 33)
35	3.808762659	192.168.2.1	192.168.4.2	ICMP	98	Echo (ping) request id=0x0764, seq=6/1536, ttl=255 (reply in 36)

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: Vmware 68:00:fb (00:0c:29:68:60:fb), Dst: Vmware 5d:92:da (00:0c:29:5d:92:da)
Internet Protocol Version 4, Src: 192.168.2.1, Dst: 192.168.4.2
Internet Control Message Protocol

(三) 网卡配置截图

(1) pc0



```
user1@ubuntu: ~/woozie/lab4
File Edit View Search Terminal Help

64 bytes from 2.0.0.0: icmp_seq=310 ttl=253 time=1.010 ms
From This Host icmp_seq=311 Destination Host Unreachable
From This Host icmp_seq=312 Destination Host Unreachable
From This Host icmp_seq=313 Destination Host Unreachable
^Z
[5]+  Stopped                  sudo ./ping 192.168.4.2
user1@ubuntu:~/woozie/lab4$ ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 00:0c:29:68:60:fb  txqueuelen 1000  (Ethernet)
        RX packets 760  bytes 100353 (100.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 903  bytes 112024 (112.0 KB)
        TX errors 0  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 3489  bytes 248177 (248.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3489  bytes 248177 (248.1 KB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

user1@ubuntu:~/woozie/lab4$
```


(2) router1

```
*****
 icmp analysis)
type:0x0
code:0x0
check sum:0xf765
identity:0x764
sequent num:0x136
*****
=====
<end>

^Z
[2]+  Stopped                  sudo ./router
user1@ubuntu:~/woozie/lab4$ ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 00:0c:29:5d:92:da  txqueuelen 1000  (Ethernet)
        RX packets 739  bytes 94089 (94.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 888  bytes 111860 (111.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens38: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 00:0c:29:5d:92:e4  txqueuelen 1000  (Ethernet)
        RX packets 740  bytes 95708 (95.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 900  bytes 111888 (111.8 KB)
        TX errors 0  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 3404  bytes 243645 (243.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3404  bytes 243645 (243.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

user1@ubuntu:~/woozie/lab4$
```


(3) router2

```
protocol:0x1(icmp)
check sum:0x5534
IP:192.168.4.2 ==> 192.168.2.1
*****
(icmp analysis)
type:0x0
code:0x0
check sum:0xf765
identity:0x764
sequent num:0x136
*****
=====
<end>

^Z
[2]+  Stopped                  sudo ./router
user1@ubuntu:~/woozie/lab4$ ifconfig -a
ens38: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 00:0c:29:3f:68:e4 txqueuelen 1000 (Ethernet)
        RX packets 718 bytes 91148 (91.1 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 876 bytes 107994 (107.9 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens39: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 00:0c:29:3f:68:ee txqueuelen 1000 (Ethernet)
        RX packets 754 bytes 100312 (100.3 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 881 bytes 110103 (110.1 KB)
        TX errors 0 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 3317 bytes 236425 (236.4 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 3317 bytes 236425 (236.4 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user1@ubuntu:~/woozie/lab4$
```

(4) pc1

```
head length of ip:0x14(20) bytes
version id:0x4
tos:0
total len:0x54
id:0x0
flag:0x0
ttl:0xfd
protocol:0x1 icmp)
check sum:0x5536
IP:192.168.2.1 ==> 192.168.4.2
*****
 icmp analysis)
type:0x8
code:0x0
check sum:0xef65
identity:0x764
sequent num:0x136
*****
=====
<end>

z^Z
[2]+  Stopped                  sudo ./reply
user1@ubuntu:~/woozie/lab4$ ifconfig -a
ens39: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 00:0c:29:05:f6:08  txqueuelen 1000  (Ethernet)
        RX packets 714  bytes 91939 (91.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 891  bytes 113015 (113.0 KB)
        TX errors 0  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 3316  bytes 237381 (237.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3316  bytes 237381 (237.3 KB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

user1@ubuntu:~/woozie/lab4$
```

六．参考网页说明：

- (1) <https://zhidao.baidu.com/question/240387199.html>
加深对原理理解
- (2) <https://www.cnblogs.com/h2zZhou/p/10488628.html>
对链路层原始套接字的讲解
- (3) <https://www.cnblogs.com/zhangshenghui/p/6097492.html>
对链路层 sockaddr_ll 等结构体讲解
- (4) https://github.com/Wuchenwcf/MyCode/blob/master/C%2B%2B/Linux/computer%20network/raw_socket_%E6%95%B0%E6%8D%AE%E9%93%BE%E8%B7%AF%E5%B1%82%E5%8F%91%E5%8C%85.cpp
数据链路层原始套接字发包示例

七．对比样例程序

参考点为创建链路层原始套接字时的一些配置工作，学习其接口的使用，获取本机 mac 的方法。参考后可以顺利的发出一个数据链路层包，以此为基础完成 ping

八．代码个人创新以及思考

- (1) 结构体的设计，比如 arp 表、路由表、mac 地址以及各种包头结构，提高了代码的复用性和可维护性
- (2) 层次式的填写解析流程，提高了代码的可读性和逻辑性
- (3) 路由器转发的同时打印抓包相关信息，提高了代码的可调试性