

## 实验二 RAW SOCKET 编程与以太网帧分析基础

141220132 银琦 141220132@smail.nju.edu.cn

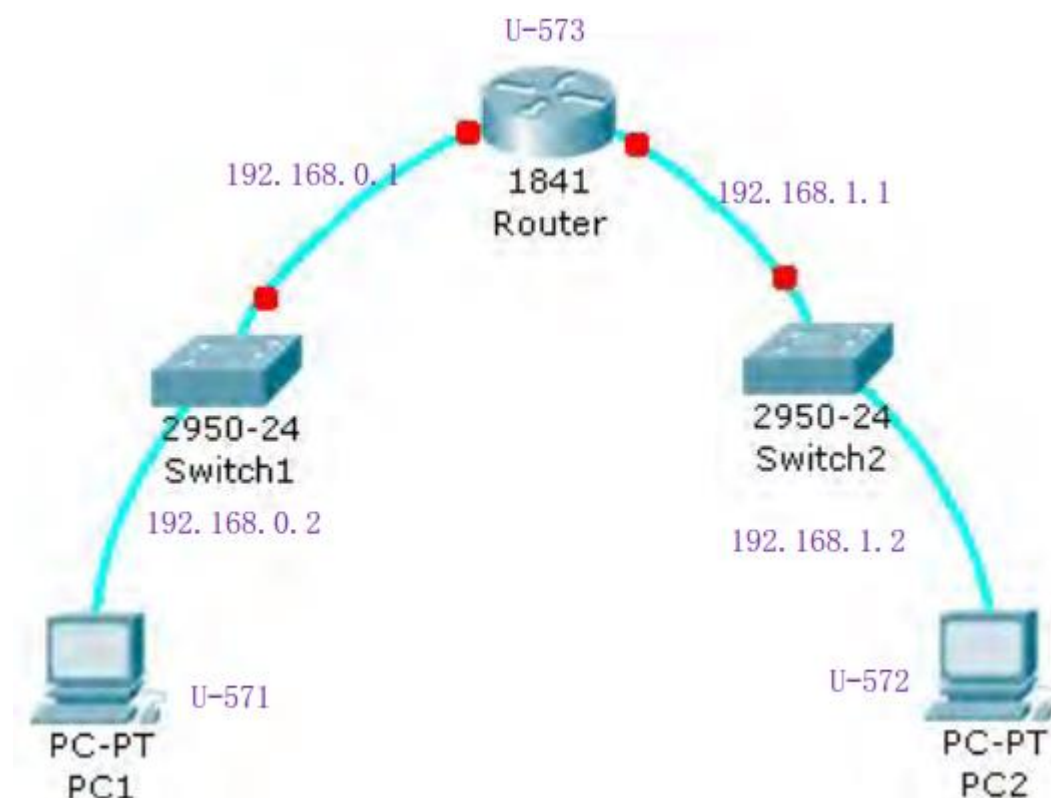
### 实验目的

本实验主要目的是让学生熟悉 Linux 环境下基本的 raw socket 编程，对以太网帧进行初步分析，可通过程序完成不同层次 PDU 的字段分析，修改和重新发送。

### 虚拟机配置

节点名	虚拟设备名	Ip	Netmask
Router	U-573	Eth0: 192.168.0.1	255.255.255.0
		Eth1: 192.168.1.1	255.255.255.0
PC1	U-571	192.168.0.2	255.255.255.0
PC2	U-572	192.168.1.2	255.255.255.0

## 虚拟机的配置图



在设置每个虚拟机之前输入 `sudo service network-manager stop`

## 设置 IP 及路由规则

在 U-571 中输入 `sudo ifconfig eth0 192.168.0.2 netmask 255.255.255.0`

`sudo route add default gw 192.168.0.1`

在 U-572 中输入 `sudo ifconfig eth0 192.168.1.2 netmask 255.255.255.0`

`sudo route add default gw 192.168.1.1`

在 U-573 中输入 `sudo ifconfig eth0 192.168.0.1 netmask 255.255.255.0`

`sudo ifconfig eth0 192.168.1.1 netmask 255.255.255.0`

`sudo vim /etc/sysctl.conf`

修改 `net.ipv4.ip_forward = 1`

## 数据结构说明

实验主要使用的结构体为 IP 和 ICMP。

IP 数据结构参看：<http://www.oschina.net/code/explore/glibc-2.9/sysdeps/generic/netinet/ip.h>

IP 报头格式数据结构定义在<netinet/ip.h>中，其定义可以对 ip 进行多方面的计算和处理，在实验中主要用到了 ip 中各数据段的含义，并对 ip 进行了检验。

ICMP 数据结构参看：[http://www.oschina.net/code/explore/glibc-2.9/sysdeps/gnu/netinet/ip\\_icmp.h](http://www.oschina.net/code/explore/glibc-2.9/sysdeps/gnu/netinet/ip_icmp.h)

ICMP 的数据结构定义在<netinet/ip\_icmp.h>中，设置 icmp 报头，发送 ICMP 报文，对 ICMP 包进行校验，接收等等。

## 程序设计的思路以及运行流程

### 1. MyRaw\_socket 程序设计：

在给定代码的基础上进行简单修改。

首先建立一个简单的链路层 socket，然后不断地接收另一台机器所发送的包，并显示包的部分内容，并且根据包头类型域的值来显示包的类型。在程序中添加关于 ARP 和 RARP 包的判定，即数据帧的 13、14 字节中 ARP 为 0x0800，而 RARP 为 0x8035。若为 ARP 或 RARP 包，进行相应的模仿 WireShark 输出，若为其他包，即根据给定程序进行修改模仿 WireShark 输出。

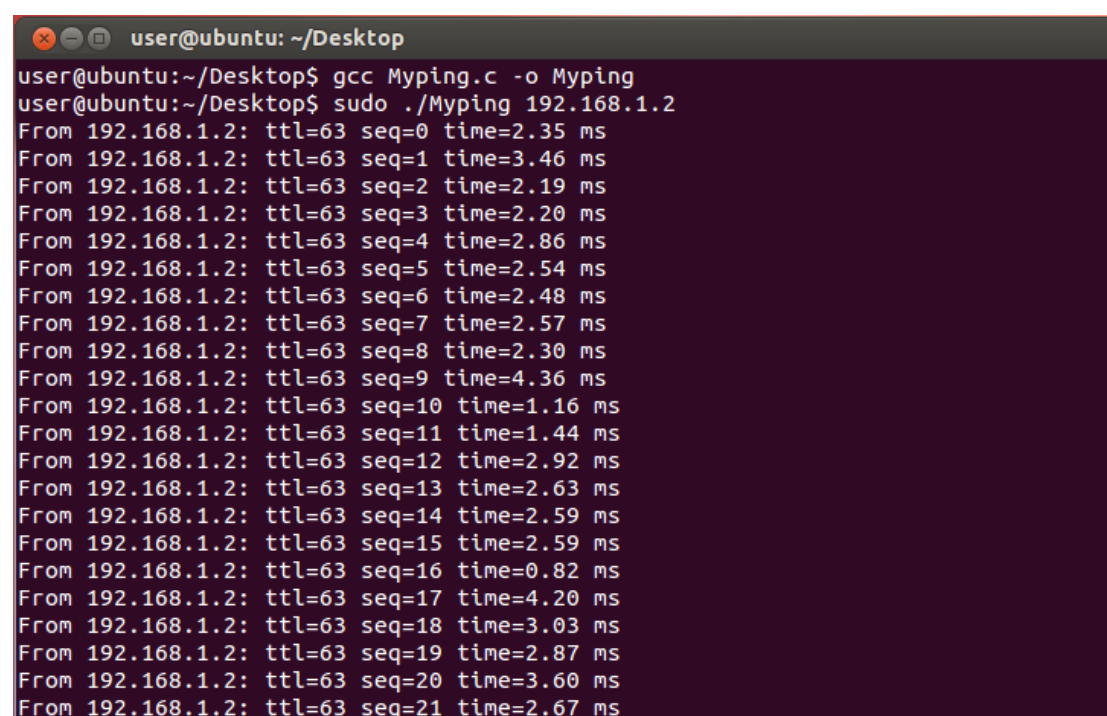
### 2. Myping 程序设计：

在网络程序的基础上删改函数完成：

首先建立链路层 socket，之后通过实验教材给出的 socket (AF\_INET, SOCK\_RAW, IPPROTO\_TCP|IPPROTO\_UDP|IPPROTO\_ICMP) 进行发包动作，对数据包进行内容填充，使得其符合 ICMP 格式，然后计算其校验和，通过函数对发包过程模仿 ping 进行输出，达到类似于 ping 动作的目的。

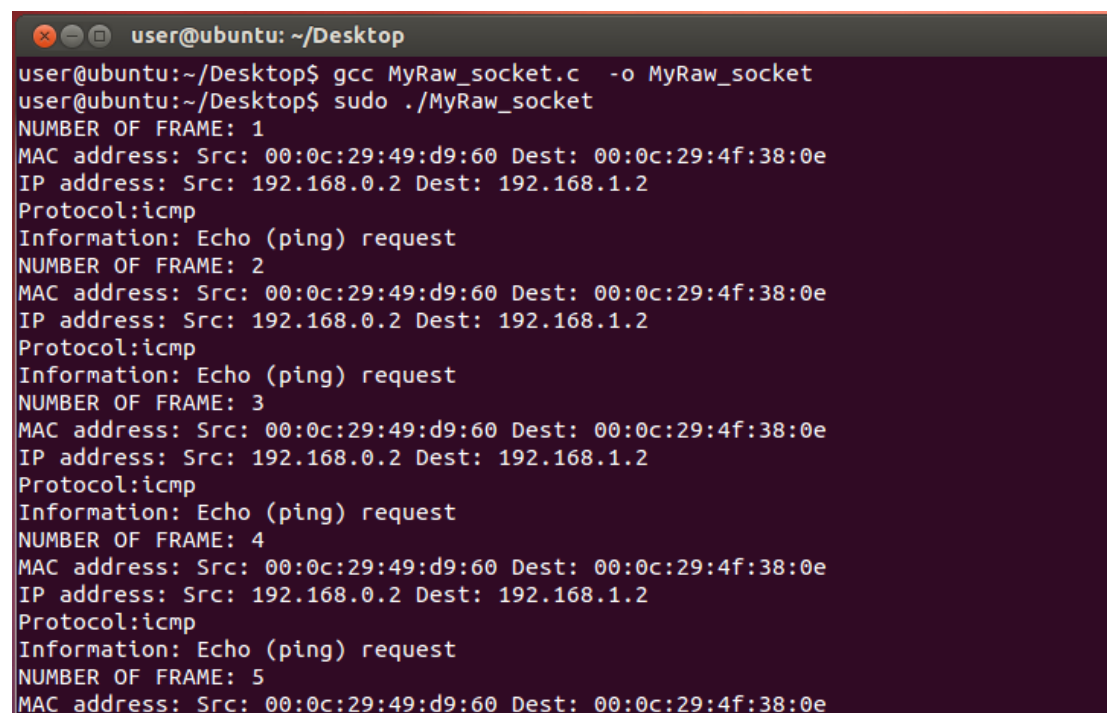
## 运行结果截图

PC1 执行自定义 Myping 程序：



```
user@ubuntu: ~/Desktop
user@ubuntu:~/Desktop$ gcc Myping.c -o Myping
user@ubuntu:~/Desktop$ sudo ./Myping 192.168.1.2
From 192.168.1.2: ttl=63 seq=0 time=2.35 ms
From 192.168.1.2: ttl=63 seq=1 time=3.46 ms
From 192.168.1.2: ttl=63 seq=2 time=2.19 ms
From 192.168.1.2: ttl=63 seq=3 time=2.20 ms
From 192.168.1.2: ttl=63 seq=4 time=2.86 ms
From 192.168.1.2: ttl=63 seq=5 time=2.54 ms
From 192.168.1.2: ttl=63 seq=6 time=2.48 ms
From 192.168.1.2: ttl=63 seq=7 time=2.57 ms
From 192.168.1.2: ttl=63 seq=8 time=2.30 ms
From 192.168.1.2: ttl=63 seq=9 time=4.36 ms
From 192.168.1.2: ttl=63 seq=10 time=1.16 ms
From 192.168.1.2: ttl=63 seq=11 time=1.44 ms
From 192.168.1.2: ttl=63 seq=12 time=2.92 ms
From 192.168.1.2: ttl=63 seq=13 time=2.63 ms
From 192.168.1.2: ttl=63 seq=14 time=2.59 ms
From 192.168.1.2: ttl=63 seq=15 time=2.59 ms
From 192.168.1.2: ttl=63 seq=16 time=0.82 ms
From 192.168.1.2: ttl=63 seq=17 time=4.20 ms
From 192.168.1.2: ttl=63 seq=18 time=3.03 ms
From 192.168.1.2: ttl=63 seq=19 time=2.87 ms
From 192.168.1.2: ttl=63 seq=20 time=3.60 ms
From 192.168.1.2: ttl=63 seq=21 time=2.67 ms
```

PC2 执行 MyRaw\_socket 接受发送帧:



```
user@ubuntu: ~/Desktop
user@ubuntu:~/Desktop$ gcc MyRaw_socket.c -o MyRaw_socket
user@ubuntu:~/Desktop$ sudo ./MyRaw_socket
NUMBER OF FRAME: 1
MAC address: Src: 00:0c:29:49:d9:60 Dest: 00:0c:29:4f:38:0e
IP address: Src: 192.168.0.2 Dest: 192.168.1.2
Protocol:icmp
Information: Echo (ping) request
NUMBER OF FRAME: 2
MAC address: Src: 00:0c:29:49:d9:60 Dest: 00:0c:29:4f:38:0e
IP address: Src: 192.168.0.2 Dest: 192.168.1.2
Protocol:icmp
Information: Echo (ping) request
NUMBER OF FRAME: 3
MAC address: Src: 00:0c:29:49:d9:60 Dest: 00:0c:29:4f:38:0e
IP address: Src: 192.168.0.2 Dest: 192.168.1.2
Protocol:icmp
Information: Echo (ping) request
NUMBER OF FRAME: 4
MAC address: Src: 00:0c:29:49:d9:60 Dest: 00:0c:29:4f:38:0e
IP address: Src: 192.168.0.2 Dest: 192.168.1.2
Protocol:icmp
Information: Echo (ping) request
NUMBER OF FRAME: 5
MAC address: Src: 00:0c:29:49:d9:60 Dest: 00:0c:29:4f:38:0e
```

## 相关资料

1. 关于 vmware 的 usb 无法识别问题解决方案:

<http://wenku.baidu.com/view/4e40aa1d59eef8c75fbfb320.html>

2. 原始套接字透析之 Raw Socket 基础

<http://wenku.baidu.com/view/ec1a81395727a5e9856a6158.html>

3. Raw Socket 介绍及编程

<http://wenku.baidu.com/view/eebccfffc8d376eeaeaa318e.html>

4. Ping 程序编写

<http://biancheng.dnbcw.info/linux/336268.html>

5. 地址解析协议 ARP

<http://baike.baidu.com/view/32698.htm?hold=redirect>

6. 反向地址解析协议 RARP

<http://baike.baidu.com/view/32772.htm>

## 对比样例程序

### 1. MyRaw\_socket 程序:

参考实验教材给定程序，其中建立链路层、发包等主题思路均相同，只是略微添加了关于包类型的判定以及优化输出。并且又通过包中位置分别 `reply` 和 `request` 的差别并进行相应的简单处理。

### 2. Myping 程序:

参考网络程序:

```
unsigned short  in_cksum(uint16_t * addr, int len) {  
    int nleft = len;  
  
    uint32_t sum = 0;  
  
    uint16_t *w = addr;  
  
    uint16_t answer = 0;  
  
    /*  
    * Our algorithm is simple, using a 32 bit accumulator (sum), we add  
    * sequential 16 bit words to it, and at the end, fold back all the  
    * carry bits from the top 16 bits into the lower 16 bits.  
    */  
  
    while (nleft > 1) {  
        sum += *w++;  
  
        nleft -= 2;
```

```
    }

    /* 4mop up an odd byte, if necessary */

    if (nleft == 1) {

        *(unsigned char *)&answer = *(unsigned char *)w ;

        sum += answer;

    }

    /* 4add back carry outs from top 16 bits to low 16 bits */

    sum = (sum >> 16) + (sum & 0xffff); /* add hi 16 to low 16 */

    sum += (sum >> 16); /* add carry */

    answer = ~sum; /* truncate to 16 bits */

    return(answer);

}


void ping(int signo) {

    struct icmp* icmp = NULL;

    static char buf[1024];

    memset(buf, 0, sizeof(buf));

    icmp = (struct icmp*)buf;

    icmp->icmp_type = ICMP_ECHO;

    icmp->icmp_code = 0;

    icmp->icmp_id = id;

    icmp->icmp_seq = seq++;
```

```
gettimeofday((struct timeval*)icmp->icmp_data, NULL);

icmp->icmp_cksum = in_cksum((uint16_t *)icmp, sizeof(struct
icmp));

sendto(rawfd, icmp, sizeof(struct icmp), 0, (struct sockaddr
*)&addr, sizeof(struct sockaddr_in));

alarm(1);

return ;

}
```

```
int main(int argc , char **argv) {

    int rc;

    int len;

    char recv[1024];

    if(argc != 2){

        err_print("usage: ping <ip-address>\n");

    }

    rc = inet_pton(AF_INET, argv[1], &addr.sin_addr);

    if(rc != 1){    //!!!!

        err_print("ip address not right\n");

    }

    addr.sin_family = AF_INET;

    if(getuid() != 0){
```



```
        err_print("must be root\n");
    }

    rawfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);

    if(rawfd < 0){
        err_print("create socket error\n");
    }

    id = getpid() & 0xFFFF;

    signal(SIGALRM, ping);

    alarm(1);

    while(1) {
        len = read(rawfd, recv, sizeof(recv));

        if(len < 0){
            err_print("read error\n");
        }

        if(len == 0){
            continue;
        }

        /*parse icmp hdr*/

        icmp_parse(recv);
    }
}
```

## 代码个人创新以及思考

1. 原本的代码输出较为简单，在 `Raw socket` 代码中又简单优化了输出方式，可以模仿 `wireshark` 的输出方式进行输出。在掌握网络通信本质之后，就可以进行图形化界面的程序编程。

2. 实现了自己编写 `ping` 程序来对机器发包。

3. 在实验开始前对整个实验的内容还不能完全理解，因而上网仔细了解了 `ARP`、`RARP` 规则，对协议、数据帧、包等等概念有了进一步的了解。也从实验中切身体会了计算机网络之间的通信方式。

4. 在实验过程中遇到了 `vmware` 软件自身电脑没有 `usb` 接口选项而机房电脑不能识别 `usb` 的情况。后上网仔细进行了了解，在个人电脑上开启关于 `usb` 服务后即解决，也尝试使用了网络方法解决机房电脑非图形化界面下无法识别 `usb` 的问题，手动挂载 `U` 盘。

5. 在配置环境时，输入 `service network restart` 会报错。