实验六 vpn 设计、实现分析

一．实验目的

设计和实现一个简单的虚拟专用网络的机制， 并与已有的标准实现（如 PPTP）进行比较， 进而让学生进一步理解 VPN 的工作原理和内部实现细节。

二．数据结构说明

```
struct VPNITEMP{
    uint32_t ip;
    uint32_t vpn_server;
    uint32_t netmask;
}VpnTable[MAX_TABLE_NUM];
```

这是一个描述 vpn 表项的数据结构

ip 表示被服务的目的主机，vpn_server 表示服务这个主机的 vpn 服务器，netmask 表示 ip 子网掩码

```
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

```
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

```
//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

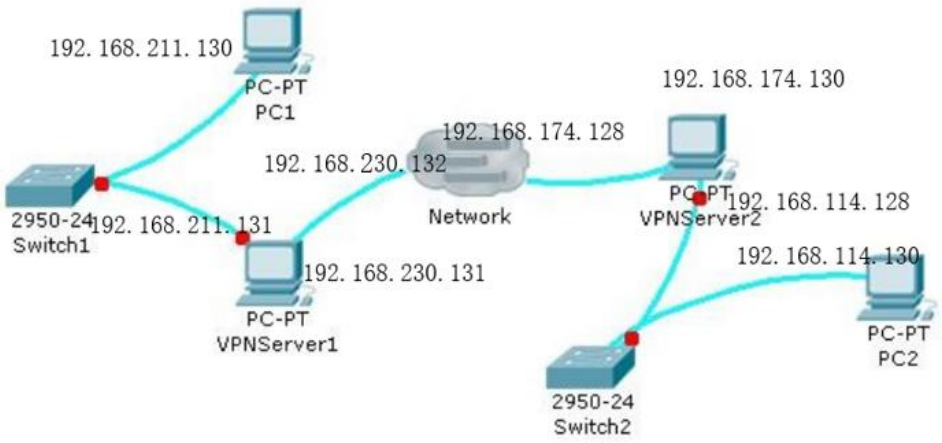
```
struct ARPITEM{
uint32_t ip;
MacAddr mac_addr;
}ArpTable[MAX_ARP_NUM];
```

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．拓扑结构



四．程序运行流程说明

1. 接受一个包
2. 分析包的数据链路层，分析出源 mac，接受端口号，负载类型

```
int proto_choice=AnalyseEth(eth_head,&interface_num,&src_mac);
```

如果负载类型是 ip 类型则 proto_choice 为 1，源 mac 填进 src_mac，interface 是表示端口号的字符串

3. 分析包的 ip 层

```
int proto=AnalyseIp(ip_head,&src_ip,&dst_ip);
```

填写源目的 ip 进入 src_ip 和 dst_ip

4. 根据接受端口号和目的 ip 判断如何处理包

```
if(interface_num==0){
    printf("in to out\n");
    //in to out, repacked
    uint32_t vpn_index;
    if((vpn_index=SearchVpn(dst_ip))==-1){
        printf("check the vpn table!\n");
        return 0;
    }

    Repacked(VpnTable[vpn_index].vpn_server,(char*)ip_head);
    proto=AnalyseIp(ip_head,&src_ip,&dst_ip);
}

else if(interface_num==1){
    printf("out to in\n");
    //out to in, unpacked
    if(ip_head->proto!=IPPROTO_IPIP)
        continue;

    if(SearchServer(src_ip)==-1)
    continue;

    else{
        unpacked((char*)ip_head);
        proto=AnalyseIp(ip_head,&src_ip,&dst_ip);
    }
}
```

如果 interface_num 是 0，则是 ens33 端口，则包要从 ens38 转发出去，要先查 vpn 表，获得 dst_ip 所对应的 vpn 服务器 ip，然后调用 repacked 将 ip 包前面再套一个 ip 头（依次右移 20 字节腾出位置就可以填入外层 ip 头），目的 ip 是目的 vpn 服务器，源 ip 是自己 vpn 服务器，类型是 ipip 类型

如果 interface_num 是 1，则是 ens38 端口，则包要从 ens33 转发进去，检查是否为 ipip 类型，不是 ipip 类型就不处理，检查源 ip 是否在 vpn 表项中能找到，找不到则说明该包不应被服务，continue 即可，如果找到了则说明是要被服务的 ipip 包，则调用 unpacked，即将外层 ip 头去掉（依次左移 20 字节）

5. 处理完包后，根据外层目的 ip 查路由表，得到下一跳的 ip 后查 arp 表获得下一跳的 mac 地址

6. 发送包，注意在发送的时候会修改 ttl，重新计算 checksum
   如果是从 ens33 端口进入的包，会调用 Forwarding 函数对 ipip 包从 ens38 进行转发
   如果是从 ens38 进入的包，会调用 Forwarding2 函数对 ip 包从 ens33 进行转发

```
if(interface_num==0)
Forwarding(RouterTable[next_index].interface,next_mac,buffer);

else{
printf("next_index:%d ,interface:%5s \n",next_index,RouterTable[next_index].interface);
Forwarding2(RouterTable[next_index].interface,next_mac,buffer);

}
```

7. 转发完后打印相关包的信息便于调试

```
switch(proto){
    case IPPROTO_ICMP:{
        //icmp analysis
            icmp_head=(ICMP_HEADER*)((char*)ip_head+off);
        AnalyseIcmp(icmp_head);
        break;
    }
    case IPPROTO_IGMP:{printf("igmp\n");break;}
    case IPPROTO_IPIP:printf("ipip\n");break;
    case IPPROTO_TCP:{
        //tcp analysis
        tcp_head=(TCP_HEADER*)((char*)ip_head+off);
        AnalyseTCP(tcp_head);
        break;
    }
    case IPPROTO_UDP:{
        //udp analysis
                                udp_head=(UDP_HEADER*)((char*)ip_head+off);
                                AnalyseUDP(udp_head);
                                break;
                            }
    default:printf("Pls query yourself\n");
}
```

五．运行结果截图

1.pc1（192.168.211.130） ping pc2（192.168.114.130）

```
user1@ubuntu:~$ ping 192.168.114.130
PING 192.168.114.130 (192.168.114.130) 56(84) bytes of data.
64 bytes from 192.168.114.130: icmp_seq=1 ttl=62 time=2.57 ms
64 bytes from 192.168.114.130: icmp_seq=2 ttl=62 time=3.46 ms
64 bytes from 192.168.114.130: icmp_seq=3 ttl=62 time=5.50 ms
64 bytes from 192.168.114.130: icmp_seq=4 ttl=62 time=3.07 ms
64 bytes from 192.168.114.130: icmp_seq=5 ttl=62 time=23.4 ms
64 bytes from 192.168.114.130: icmp_seq=6 ttl=62 time=3.49 ms
64 bytes from 192.168.114.130: icmp_seq=7 ttl=62 time=3.83 ms
64 bytes from 192.168.114.130: icmp_seq=8 ttl=62 time=3.86 ms
64 bytes from 192.168.114.130: icmp_seq=9 ttl=62 time=3.74 ms
64 bytes from 192.168.114.130: icmp_seq=10 ttl=62 time=6.07 ms
64 bytes from 192.168.114.130: icmp_seq=11 ttl=62 time=4.53 ms
64 bytes from 192.168.114.130: icmp_seq=12 ttl=62 time=72.6 ms
64 bytes from 192.168.114.130: icmp_seq=13 ttl=62 time=3.95 ms
64 bytes from 192.168.114.130: icmp_seq=14 ttl=62 time=3.86 ms
```

2.vpn1 ens33 端口抓包 request

```
▸ Frame 18: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: Vmware_5d:92:da (00:0c:29:5d:92:da), Dst: Vmware_05:0f:71 (00:0c:29:05:0f:71)
  ▸ Destination: Vmware_05:0f:71 (00:0c:29:05:0f:71)
  ▸ Source: Vmware_5d:92:da (00:0c:29:5d:92:da)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.211.130, Dst: 192.168.114.130
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x10af (4271)
  ▸ Flags: 0x0000
    Time to live: 255
    Protocol: ICMP (1)
    Header checksum: 0xe3a3 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.211.130
    Destination: 192.168.114.130
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xd030 [correct]
    [Checksum Status: Good]
    Identifier (BE): 3530 (0x0dca)
    Identifier (LE): 51725 (0xca0d)

0000  00 0c 29 05 0f 71 00 0c  29 5d 92 da 08 00 45 00   ··)··q·· )]····E·
0010  00 54 10 af 00 00 ff 01  e3 a3 c0 a8 d3 82 c0 a8   ·T····· ········
0020  72 82 08 00 d0 30 0d ca  1a 05 00 00 00 00 00 00   r····0·· ········
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00                                             ··
```

3.vpn1 ens33 端口抓包 reply

```
▸ Frame 19: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: Vmware_05:0f:71 (00:0c:29:05:0f:71), Dst: Vmware_5d:92:da (00:0c:29:5d:92:da)
  ▸ Destination: Vmware_5d:92:da (00:0c:29:5d:92:da)
  ▸ Source: Vmware_05:0f:71 (00:0c:29:05:0f:71)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.114.130, Dst: 192.168.211.130
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x4812 (18450)
  ▸ Flags: 0x0000
    Time to live: 62
    Protocol: ICMP (1)
    Header checksum: 0x6d41 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.114.130
    Destination: 192.168.211.130
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xd830 [correct]
    [Checksum Status: Good]
    Identifier (BE): 3530 (0x0dca)
    Identifier (LE): 51725 (0xca0d)

0000  00 0c 29 5d 92 da 00 0c  29 05 0f 71 08 00 45 00   ··)]···· )··q··E·
0010  00 54 48 12 00 00 3e 01  6d 41 c0 a8 72 82 c0 a8   ·TH···>· mA··r···
0020  d3 82 00 00 d8 30 0d ca  1a 05 00 00 00 00 00 00   ·····0·· ········
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00                                             ··
```

4.vpn1 ens38 端口抓包 request

```
▸ Frame 5: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
▸ Ethernet II, Src: Vmware_05:0f:7b (00:0c:29:05:0f:7b), Dst: Vmware_2a:52:a3 (00:0c:29:2a:52:a3)
▸ Internet Protocol Version 4, Src: 192.168.230.131, Dst: 192.168.174.130
▸ Internet Protocol Version 4, Src: 192.168.211.130, Dst: 192.168.114.130
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x7f95 [correct]
    [Checksum Status: Good]
    Identifier (BE): 4263 (0x10a7)
    Identifier (LE): 42768 (0xa710)
    Sequence number (BE): 228 (0x00e4)
    Sequence number (LE): 58368 (0xe400)
    [Response frame: 6]
    Timestamp from icmp data: May 29, 2019 07:07:57.000000000 PDT
    [Timestamp from icmp data (relative): 2.869044240 seconds]
  ▸ Data (48 bytes)
```

5.vpn1 ens38 端口抓包 reply

```
▸ Frame 9: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
▸ Ethernet II, Src: Vmware_2a:52:a3 (00:0c:29:2a:52:a3), Dst: Vmware_05:0f:7b (00:0c:29:05:0f:7b)
▸ Internet Protocol Version 4, Src: 192.168.174.130, Dst: 192.168.230.131
▸ Internet Protocol Version 4, Src: 192.168.114.130, Dst: 192.168.211.130
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x8795 [correct]
    [Checksum Status: Good]
    Identifier (BE): 4263 (0x10a7)
    Identifier (LE): 42768 (0xa710)
    Sequence number (BE): 228 (0x00e4)
    Sequence number (LE): 58368 (0xe400)
    Timestamp from icmp data: May 29, 2019 07:07:57.000000000 PDT
    [Timestamp from icmp data (relative): 2.870648379 seconds]
  ▸ Data (48 bytes)
```

6.net any 端口抓包 request

```
Frame 6: 120 bytes on wire (960 bits), 120 bytes captured (960 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 192.168.230.131, Dst: 192.168.174.130
Internet Protocol Version 4, Src: 192.168.211.130, Dst: 192.168.114.130
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xe034 [correct]
    [Checksum Status: Good]
    Identifier (BE): 3530 (0x0dca)
    Identifier (LE): 51725 (0xca0d)
    Sequence number (BE): 2561 (0x0a01)
    Sequence number (LE): 266 (0x010a)
    [Response frame: 9]
  Data (56 bytes)
```

7.net any 端口抓包 reply

```
Frame 9: 120 bytes on wire (960 bits), 120 bytes captured (960 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 192.168.174.130, Dst: 192.168.230.131
Internet Protocol Version 4, Src: 192.168.114.130, Dst: 192.168.211.130
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xe834 [correct]
    [Checksum Status: Good]
    Identifier (BE): 3530 (0x0dca)
    Identifier (LE): 51725 (0xca0d)
    Sequence number (BE): 2561 (0x0a01)
    Sequence number (LE): 266 (0x010a)
    [Request frame: 6]
    [Response time: 1.992 ms]
  Data (56 bytes)
```

六．代码个人创新以及思考

（1）结构体的设计，比如 vpn 表、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