计算机网络实验lab4

静态路由编程

1.数据结构声明

结构声明如下

```
struct route_t {

        struct in_addr dest_addr;
        struct in_addr gateway;
        struct in_addr mask;
        char interface[iface_len]; // #define IFNAMSIZ 8
};


// HWADDR=, 其中 以AA:BB:CC:DD:EE:FF形式的以太网设备的硬件地址
struct device_t // 记录mac地址
{

        char interface[iface_len];
        unsigned char hwaddr [ETH_ALEN]; // define ETH_ALEN 6 {AA : BB : CC : DD : EE :FF }
};

struct arp_t  // 记录主机ip地址和mac地址的对应关系
{

    struct in_addr ip_addr;
    unsigned char hwaddr[ETH_ALEN];
};
```

2.路由器编程思路

1.建立socket套接字，读取数据包全部部分

```
int sockfd = socket(AF_PACKET,SOCK_RAW,htons(ETH_P_ALL));
```

2.接受链路层数据

```
        struct sockaddr_ll addr;
        socklen_t addr_len = sizeof(addr);
        int rev_result = recvfrom(sockfd,buffer,BUF_LEN,0,(struct sockaddr *) &addr,& addr_len);
```

3.查询arp表，获得下一跳的mac地址，修改eth头

```
        // 获得下一跳的IP地址 ,遍历arp_table
         struct in_addr next_addr;
         next_addr.s_addr = ip_head->daddr;
         //
         //next_addr =
         int arp_i;
         for( arp_i = 0; arp_i < arp_tbl_len;arp_i++)
         {
             if(arp_tbl[arp_i].ip_addr.s_addr == next_addr.s_addr)
                 break;

         }
         if(arp_i == 2)
             arp_i = 1;
          printf("arp_i : %d\n",arp_i);
       //  struct in_addr temp_addr
      //   next_addr = route_tbl[index].gateway;
         //修改报头，准备重新发送数据
         memcpy(eth_head->h_source, eth_head->h_dest,ETH_ALEN);
         memcpy(eth_head->h_dest,arp_tbl[arp_i].hwaddr,ETH_ALEN);
```

4.查询route表，获得发送网卡信息，然后从网卡发送数据

```
         int sockf_inter = socket(AF_PACKET,SOCK_DGRAM,htons(ETH_P_IP));
         struct ifreq req;
         memset(&req,0,sizeof(req));
         strncpy(req.ifr_name,route_tbl[index].interface,IFNAMSIZ - 1);
         ioctl(sockf_inter,SIOCGIFINDEX,&req);
         int if_index = req.ifr_ifindex;
         printf("index name : %d \n",if_index);
         //重新发送数据
         int sock2send = socket(AF_PACKET, SOCK_DGRAM,htons(ETH_P_IP));
         struct sockaddr_ll dest2addr = {
             .sll_family = AF_PACKET,
             .sll_protocol = htons(ETH_P_IP),
             .sll_halen = ETH_ALEN,
             .sll_ifindex = if_index,
         };
         ioctl(sockf_inter,SIOCGIFHWADDR,&req);
         memcpy(dest2addr.sll_addr,arp_tbl[arp_i].hwaddr,ETH_ALEN);
         int result = sendto(sockfd,buffer,rev_result,0,(struct sockaddr
*)&dest2addr,sizeof(dest2addr));
```

3.ping主机部分思路

1.建立套接字,接受所有数据

```
sockfd = Socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
```

2.设置发送网卡

```
    struct ifreq ifreq_i;
    memset(&ifreq_i,0,sizeof(ifreq_i));
    //giving name of Interface
    strncpy(ifreq_i.ifr_name, IFACE_NAME, IFNAMSIZ-1);
    //getting Interface Index
    Ioctl(sockfd, SIOCGIFINDEX, &ifreq_i);
    sadr_ll.sll_ifindex = ifreq_i.ifr_ifindex; // index of interface

    sadr_ll.sll_halen = ETH_ALEN;
    sadr_ll.sll_family = AF_PACKET;
    memcpy(sadr_ll.sll_addr, NEXT_HWADDR, ETH_ALEN);
```

3.设置eth头

```
    struct ethhdr *eth_head = (struct ethhdr *)(send_buffer);
    memcpy(eth_head->h_source,PC_HWADDR,ETH_ALEN);
    memcpy(eth_head->h_dest,NEXT_HWADDR,ETH_ALEN);
    eth_head->h_proto = htons(ETH_P_IP);
```

4.设置ip头

```
    struct iphdr *ip_head = (struct iphdr *)(send_buffer + sizeof(ethhdr));
    ip_head->version = 4;
    ip_head->ihl = 5;
    ip_head->ttl = 64;
    ip_head->protocol = 1;  // ICMP
    ip_head->saddr = inet_addr(LOCAL_IP);
    ip_head->daddr = inet_addr(DST_IP);
    ip_head->tot_len = htons(sizeof(struct iphdr) + ICMP_PACKET_LEN);
    ip_head->check = cal_chksum((void *)ip_head, sizeof(struct iphdr));
```

4.设置eth头

```
    struct icmphdr *icmp_head = (struct icmphdr *)(send_buffer + sizeof(ethhdr) +
sizeof(iphdr));

    icmp_head->type  = ICMP_ECHO; //8 - Echo Request
    icmp_head->code  = 0;
    // icmp_hdr->checksum = 0;
    //icmp_head->un.echo.id = htons(getpid());
    icmp_head->icmp_id = uid;
    icmp_head->un.echo.sequence = htons(sequence);

    gettimeofday((struct timeval *)((char *)icmp_head+sizeof(struct icmphdr)),NULL);
    icmp_head->checksum =cal_chksum((unsigned short*)icmp_head, ICMP_PACKET_LEN);

    /* Send Ethernet frame */
    int total_len = sizeof(struct ethhdr) + sizeof(struct iphdr) + ICMP_PACKET_LEN;
```

5.发送数据包

```
int send_end = sendto(sockfd, sendbuf, total_len, 0,
(const struct sockaddr*)&sadr_ll,sizeof(struct sockaddr_ll)
```

4.reply主机部分思路

1.建立套接字

```
sock_r = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_IP));
```

2.收取数据包

```
int  result = recvfrom(sock_r,buffer,BUFFSIZ,0, NULL, NULL);
```

3.验证各部分信息是否匹配

```
struct ethhdr *eth = (struct ethhdr *)(buffer);
if (strncmp((char *)eth->h_source, (char *)LOCAL_HWADDR, ETH_ALEN) &&
strncmp((char *)eth->h_source, (char *)BROADCAST_HWADDR, ETH_ALEN))
continue;
printf("Ethernet header checked\n");


// Extracting the IP header
struct iphdr *ip = (struct iphdr*)(buffer + sizeof(struct ethhdr));
struct in_addr local_in_addr, dst_in_addr;
inet_aton(LOCAL_IP, &local_in_addr);
dst_in_addr.s_addr = ip->daddr;
if (local_in_addr.s_addr != dst_in_addr.s_addr || ip->protocol != 1)
continue;
printf("IP header checked\n");

// Extracting the ICMP header
struct icmphdr *icmp_hdr =
(struct icmphdr *)(buffer + sizeof(struct ethhdr) + sizeof(struct iphdr));
if (icmp_hdr->type  != ICMP_ECHO)
    continue;
printf("ICMP header checked\n");
```

4.修改源mac和目的mac,源ip和目的ip

```
memcpy(eth->h_source, eth->h_dest, ETH_ALEN);
memcpy(eth->h_dest, NEXT_HOP_HWADDR, ETH_ALEN);

// Change the source and destination IP address

ip->daddr = ip->saddr;
```

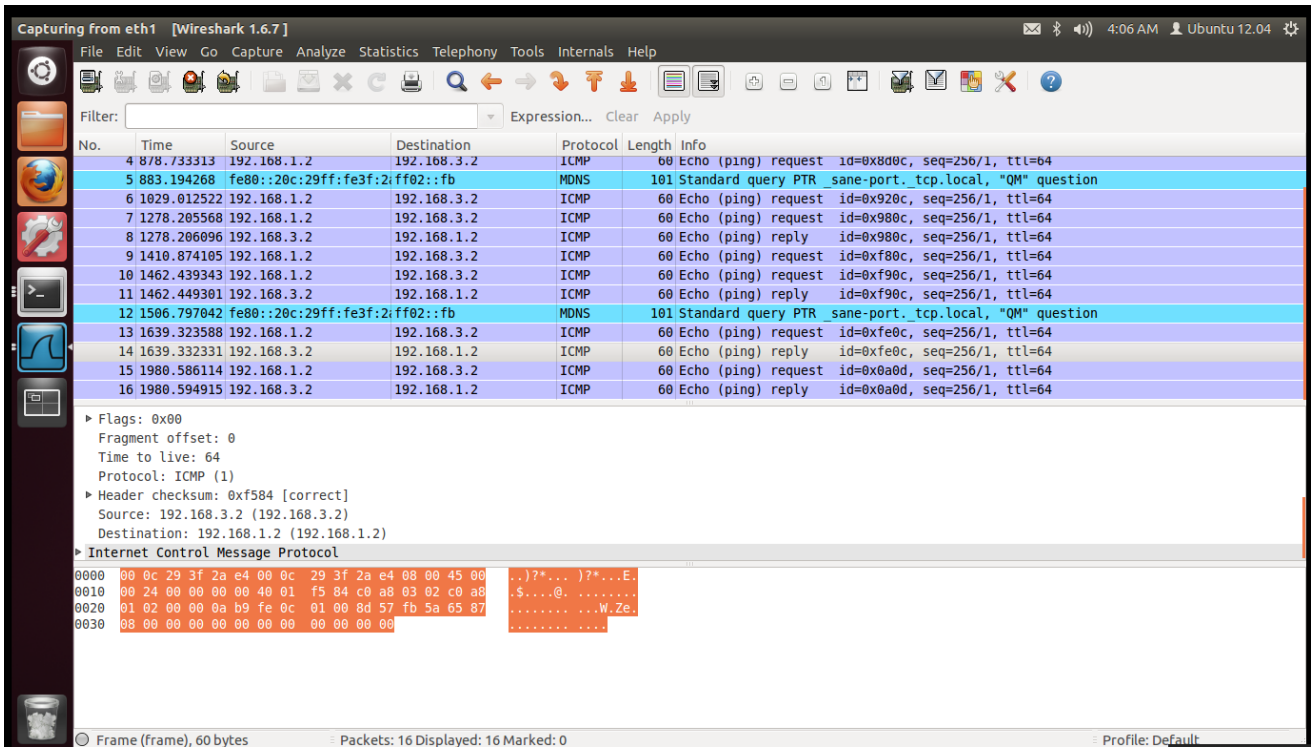
```
    ip->saddr = inet_addr(LOCAL_IP);
    ip->check = 0;
    ip->check = cal_chksum((unsigned short *)ip, sizeof(struct iphdr));

    // Change ICMP header
    icmp_hdr->type = ICMP_ECHOREPLY;
    icmp_hdr->checksum = 0;
    icmp_hdr->checksum = cal_chksum((unsigned short *)icmp_hdr,
    sizeof(struct icmphdr) + sizeof(struct timeval));
```

5.重新发送

```
int result_send = sendto(sock_r, buffer, buflen, 0,
    (const struct sockaddr*)&sadr_ll,sizeof(struct sockad
```

5.wireshark抓包截图



6.实验中遇到的问题和疑问

sendto函数出现过无法发包的情况 参数len 不宜过大，应该使用缓冲区的有效长度

```
    const char *arp_ip2 = "192.168.2.2";
    const char arp_hw2[ETH_ALEN] = {0x00,0x0c,0x29,0x3f,0x3b,0x15};

    strncpy(arp_tbl[0].hwaddr,arp_hw1,ETH_ALEN);
    strncpy(arp_tbl[1].hwaddr,arp_hw2,ETH_ALEN);
```

此段代码无法初始化hwaddr，打印出来的hwaddr仍然是空，至今无法理解

在循环中，每次recv之前都需要将buffer缓冲区初始化。

arp表和路由表的转发规则设置问题。 当目的ip不在当前路由表中的时候的时候，默认选择哪一个。

icmp 和 icmphdr是否指的是同一个结构体