IP转发实验报告

1800017781 程芷怡 元培学院

一、实验要求和接口

这个实验要求我们完成正确的本地接收ip报文和实现ip报文转发的工作，重点是实现路由表的数据结构和查找。
系统提供的接口有：

```
extern void fwd_SendtoLower(char *pBuffer, int length, unsigned int nexthop);
extern void fwd_LocalRcv(char *pBuffer, int length);
extern void fwd_DiscardPkt(char *pBuffer, int type);
extern unsigned int getIpv4Address();
```

其中。 `fwd_SendtoLower` 将 `pBuffer` 指向的报文传到下一层协议层发送，参数 `length` 是报文长度， `nexthop` 是下一跳。 `fwd_LocalRcv` 在本地接收 `pBuffer` 指向的报文。 `fwd_DiscardPkt` 出现错误丢弃报文， `getIpv4Address` 获取本机ip地址。
需要实现的接口有：
`void stud_Route_Init()` 初始化路由表。
`void stud_route_add(stud_route_msg *proute)` 往路由表中添加项。
`int stud_fwd_deal(char *pBuffer, int length)` 对付来的ip包。

其中， `stud_route_msg` 结构为：

```
struct stud_route_msg {
    unsigned int dest;
    unsigned int masklen;
    unsigned int nexthop;
};
```

二、数据结构设计

数据结构十分简单，直接用了一个 `vector` 存储指向结构体的指针。

三、函数过程

```
void stud_Route_Init() {}
```

不需要初始化。

```
void stud_route_add(stud_route_msg *proute) {
    stud_route_msg *to_add = new stud_route_msg();
    to_add->dest = proute->dest;
    to_add->masklen = proute->masklen;
    to_add->nexthop = proute->nexthop;

    route_map.push_back(to_add);
}
```

添加路由表项。

添加了 route_find 函数和最长匹配所使用的 count_prefix_zero_bit 函数：
（注意容易错的一个点是网络序和主机序的转换，必须是主机序的比较才能从高位开始算zero_bit）。

```
int count_prefix_zero_bit(unsigned int num) {
    int cnt = 0;
    unsigned int mask = (1 << 31);
    while (mask) {
        if (mask & num == 0) {
            cnt++;
            mask >>= 1;
        } else {
            return cnt;
        }
    }
    return 32;
}

stud_route_msg *route_find(unsigned int dest) {
    int maxlen = 0, maxidx = -1;
    for (int i = 0; i < route_map.size(); i++) {
        if (count_prefix_zero_bit(ntohl(dest) ^ ntohl(route_map[i]->dest)) >= route_map[i]->masklen) {
            if (route_map[i]->masklen >= maxlen) {
                maxlen = route_map[i]->masklen;
                maxidx = i;
            }
        }
    }
    if (maxidx == -1) {
        return NULL;
    }
    return route_map[maxidx];
}
```

stud_fwd_deal 函数首先检查目标地址是否是本机地址，如果是直接接受，否则在路由表中查找。如果表中不存在对应目标地址的表项则丢弃报错，否则检查ttl。如果ttl为0，则丢弃，否则重新设置校验和，注意这里需要跳过，不要把原来的校验和加上。代码如下：

```
int stud_fwd_deal(char *pBuffer, int length) {
    unsigned int dest_addr = ((unsigned long *)pBuffer)[4];
    printf("dest_addr: %x\n", dest_addr);
    if (ntohl(dest_addr) == getIpv4Address()) { //
        fwd_LocalRcv(pBuffer, length);
        return 0;
    }
    stud_route_msg *message = route_find(dest_addr);
    if (message == NULL) {
        fwd_DiscardPkt(pBuffer, STUD_FORWARD_TEST_NOROUTE);
        return 1;
    }
    if ((unsigned int)pBuffer[8]== 0) {
        fwd_DiscardPkt(pBuffer, STUD_FORWARD_TEST_TTLERROR);
        return 1;
    }
    pBuffer[8] -= 1;
    unsigned int sum = 0;
    for (int i = 0; i < ((unsigned int)pBuffer[0]&0xf)*4; i += 2) {
        if (i == 10) continue; //
        sum += (*(unsigned short *)(pBuffer + i));
    }
    unsigned short sum2 = (sum & 0xffff) + (sum >> 16);
    sum2 = ~sum2;
    ((unsigned short *)pBuffer)[5] = (sum2);
    if (dest_addr == getIpv4Address()) {
        fwd_LocalRcv(pBuffer, length);
    } else {
        fwd_SendtoLower(pBuffer, length, message->nexthop);
    }
    return 0;
}
```