计算机网络LAB3——IPV4转发实验
1400012849 李岩昊

1. 实验概述

在前面 IPv4 分组收发实验的基础上，增加分组转发功能。具体来说，对于每一个到达本机的 IPv4 分组，根据其目的 IPv4 地址决定分组的处理行为，对该分组进行如下的几类操作:

1) 向上层协议上交目的地址为本机地址的分组;

2) 根据路由查找结果，丢弃查不到路由的分组;

3) 根据路由查找结果，向相应接口转发不是本机接收的分组。

实验内容:

1) 设计路由表所采用的数据结构。要求能够根据目的 IPv4 地址来确定分组处理行为(转发情况下需获得下一跳的 IPv4 地址)。

2) IPv4 分组的接收和发送。对前面实验(IP实验)中所完成的代码进行修改，在路由器协议栈的 IPv4 模块中能够正确完成分组的接收和发送处理。具体要求不做改变，参见“IP 实验”。

3) IPv4 分组的转发。对于需要转发的分组进行处理，获得下一跳的 IP 地址，然后调用发送接口函数做进一步处理。

2. 实验实现思路

这次实验的数据结构非常简单，用Vector维护一个路由表。其中只需要保存目的地址和下一跳地址即可。具体实现如下图:

```
// implemented by students
struct item{
    int dst;
    int nexthop;
};
vector<item> table; // Router Table
```

1. 路由表的初始化和添加元素

stud_Route_Init()中只需将路由表清空，调用vector::clear()即可。stud_route_add中只需按照handout中给出的数据结构分析提取出对应的字段，再根据masklen的定义提取出目的地址并将元素push进vector尾部即可

具体实现细节如下——

```
void stud_Route_Init()
{
    table.clear();
    return;
}

void stud_route_add(stud_route_msg *proute)
{
    item tmp;
    tmp.dst = ntohl(proute->dest) & (0xffffffff << (32 - ntohl(proute->masklen)));
    tmp.nexthop = ntohl(proute->nexthop);
    table.push_back(tmp);
    return;
}
```

2. 路由器转发处理

按照上一次的收发实验提取对应字段，若目的地为本机地址则发送到上层，若TTL小于0则丢弃。否则遍历路由表确定转发的动作目的地。若找到表项，将TTL字段值-1，并重新计算校验和。若找不到表项则将该包丢弃。由于handout中没有给出对version，ihl错误的处理方法，所以忽略不计。具体实现细节如下——

```
int stud_fwd_deal(char *pBuffer, int length)
{
    // same as the previous lab
    int version = pBuffer[0] >> 4;
    int ihl = pBuffer[0] & 0xf;
    int ttl = (int)pBuffer[8];
    int checkSum = ntohs(*(short unsigned int*)(pBuffer + 10));
    int dstip = ntohl(*(unsigned int*)(pBuffer + 16));
    if (dstip == getIpv4Address()){
        fwd_LocalRcv(pBuffer, length);
        return 0;
    }
    if (ttl <= 0){
        fwd_DiscardPkt(pBuffer, STUD_FORWARD_TEST_TTLERROR);
        return 1;
    }
    vector<item>::iterator it;
    for (it = table.begin(); it != table.end(); it++){
        if (it->dst == dstip){
            char* buf = new char[length];
            memcpy(buf, pBuffer, length);
            buf[8]--;
            unsigned short int newchecksum = 0;
            for (int i = 0; i < 2 * ihl; ++i){
                if (i == 5)
                    continue;
                newchecksum += (buf[i*2] << 8) + (buf[i*2+1]);
            }
            newchecksum = htons(0xffff-newchecksum);
            memcpy(buf + 10, &newchecksum, sizeof(unsigned short int));
            fwd_SendtoLower(buf, length, it->nexthop);
            return 0;
        }
    }
    fwd_DiscardPkt(pBuffer, STUD_FORWARD_TEST_NOROUTE);
    return 1;
}
```

3. 实验中遇到的困难

1. 没困难。

4. 附录代码

见提交tar包中附的代码以及在线测试系统上的代码