**实验 4：IPv4 分组转发实验**

|  |  |
| --- | --- |
| **学号：** | 1130310128 |
| **姓名：** | 杨尚斌 |
| **专业：** | 计算机科学与技术 |
| **指导老师：** | 聂兰顺 |

1. **实验目的**

通过前面的实验，我们已经深入了解了 IPv4 协议的分组接收和发送处理流程。本实验需要将实验模块的角色定位从通信两端的主机转移到作为中间节点的路由器上，在 IPv4 分组收发处理的基础上，实现分组的路由转发功能。

网络层协议最为关注的是如何将IPv4分组从源主机通过网络送达目的主机，这个任务就是由路由器中的 IPv4 协议模块所承担。路由器根据自身所获得的路由信息，将收到的IPv4分组转发给正确的下一跳路由器。如此逐跳地对分组进行转发，直至该分组抵达目的主机。IPv4 分组转发是路由器最为重要的功能。

本实验设计模拟实现路由器中的 IPv4 协议，可以在原有 IPv4 分组收发实验的基础上，增加 IPv4 分组的转发功能。对网络的观察视角由主机转移到路由器中，了解路由器是如何为分组选择路由，并逐跳地将分组发送到目的主机。本实验中也会初步接触路由表这一重要的数据结构，认识路由器是如何根据路由表对分组进行转发的。

1. **实验环境**

* 接入 Internet 的实验主机
* Windows XP
* 开发语言：C++

1. **实验内容**

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。

1. **实验要点**

**1) 设计路由表数据结构。**

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）。路由表的数据结构和查找算法会极大的影响路由器的转发性能，有兴趣的同学可以深入思考和探索。

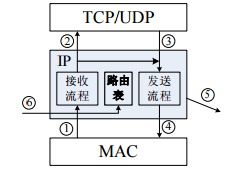
**2) IPv4 分组的接收和发送。**

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

1. **实验结果**



**路由表结构体：**

struct routeTable

{

unsigned int destIP;

unsigned int mask;

unsigned int masklen;

unsigned int nexthop;

};

路由表的实现模仿 struct stud\_route\_msg 结构体的实现，为了方便运算，在路由表的数据结构里面加入了 mask，这个值可以由 mask\_len 来得到

**路由表初始化函数：**

vector<routeTable> m\_table;

void stud\_Route\_Init()

{

m\_table.clear();

return;

}

声明 route\_Table 类型的m\_table，路由表初始化的时候清空这个 vector

**路由表增加函数：**

void stud\_route\_add(stud\_route\_msg \*proute)

{

routeTable newTableItem;

newTableItem.masklen = ntohl(proute->masklen);

newTableItem.mask = (1<<31)>>(ntohl(proute->masklen)-1);

newTableItem.destIP = ntohl(proute->dest)&newTableItem.mask;

newTableItem.nexthop = ntohl(proute->nexthop);

m\_table.push\_back(newTableItem);

return;

}

int stud\_fwd\_deal(char \*pBuffer, int length)

{

int IHL = pBuffer[0] & 0xf;

int TTL = (int)pBuffer[8];

int headerChecksum = ntohl(\*(unsigned short\*)(pBuffer+10));

int destIP = ntohl(\*(unsigned int\*)(pBuffer+16));

if(destIP == getIpv4Address())

{

fwd\_LocalRcv(pBuffer, length);

return 0;

}

if(TTL <= 0)

{

fwd\_DiscardPkt(pBuffer, STUD\_FORWARD\_TEST\_TTLERROR);

return 1;

}

bool isMatch = false;

unsigned int longestMatchLen = 0;

int bestMatch = 0;

for(int i = 0; i < m\_table.size(); i ++)

{

if(m\_table[i].masklen > longestMatchLen && m\_table[i].destIP == (destIP & m\_table[i].mask))

{

bestMatch = i;

isMatch = true;

longestMatchLen = m\_table[i].masklen;

}

}

if(isMatch)

{

char \*buffer = new char[length];

memcpy(buffer,pBuffer,length);

buffer[8]--; //TTL - 1

int sum = 0;

unsigned short int localCheckSum = 0;

for(int j = 0; j < 2 \* IHL; j ++)

{

if (j != 5) {

sum = sum + (buffer[j\*2]<<8) + (buffer[j\*2+1]);

}

}

while((unsigned(sum) >> 16) != 0)

sum = unsigned(sum) >> 16 + sum & 0xffff;

localCheckSum = htons(0xffff - (unsigned short int)sum);

memcpy(buffer+10, &localCheckSum, sizeof(unsigned short));

fwd\_SendtoLower(buffer, length, m\_table[bestMatch].nexthop);

return 0;

}

else

{

fwd\_DiscardPkt(pBuffer, STUD\_FORWARD\_TEST\_NOROUTE);

return 1;

}

return 1;

}

如果目标地址是本地 Ip 地址，则直接交接给上层，如果不是的情况下，则检验 TTL，如果大于 0 ，则使用最长子网掩码算法在路由表中查找，找到之后修改校验和然后转发，如果没有则抛出错误。