哈爾濱工業大學
Harbin Institute of Technology

计算机网络
课程实验报告

实验名称	ip 分组收发实验和转发实验				
姓名	姜思琪		院系	计算机科学与技术学院	
班级	1603109		学号	1160300814	
任课教师	李全龙		指导教师	李全龙	
实验地点	格物 213		实验时间	2018.11.10 周六 3、4 节	
实验课表现	出勤、表现得分(10)		实验报告得分(40)	实验总分	
	操作结果得分(50)				
教师评语					

计算机科学与技术学院 SINCE 1956...
School of Computer Science and Technology

实验目的：

IPv4 分组收发实验：

本实验通过设计实现主机协议栈中的 IPv4 协议，让学生深入了解网络层协议的基本原理，学习 IPv4 协议基本的分组接收和发送流程。

另外，通过本实验，学生可以初步接触互联网协议栈的结构和计算机网络实验系统，为后面进行更为深入复杂的实验奠定良好的基础。

IPv4 分组转发实验：

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。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。

实验内容：

IPv4 分组收发实验：

1) 实现 IPv4 分组的基本接收处理功能。对于接收到的IPv4分组，检查目的地址是否为本地地址，并检查IPv4 分组头部中其它字段的合法性。提交正确的分组给上层协议继续处理， 丢弃错误的分组并说明错误类型。

2) 实现 IPv4 分组的封装发送。根据上层协议所提供的参数，封装 IPv4 分组，调用系统提供的发送接口函数将分组发送出去。

IPv4 分组转发实验：

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。

实验过程：

IPv4 分组收发实验：

1) 实现 IPv4 分组的基本接收处理功能。

计算机网络实验系统所提供的上下层接口函数和协议中分组收发的主要流程，设计实现一个简单的 IPv4 分组收发模块。能够检测出接收到的 IP 分组是否存在如下错误：校验和错、TTL 错、版本号错、头部长度错、错误目标地址。

首先，要设计IP v4分组的**头部格式**。根据教材设计一个结构体，包括版本号、协议类型、总长度、标志符、偏移量、TTL、协议、首部校验和、源地址和目标地址。

结构代码如下：

```
struct Ipv4{
    char version_ihl;
    char type_of_service;
    short total_length;
    short identification;
    short fragment_offset;
    char time_to_live;
    char protocol;
    short header_checksum;
    unsigned int source_address;
    unsigned int destination_address;
    Ipv4() {memset(this,0,sizeof(Ipv4));}
```

根据已知知识，我们知道IP v4的**工作流程**：

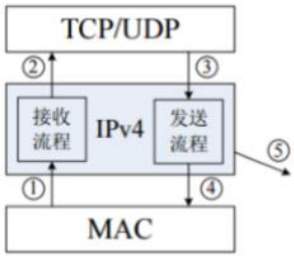
发送：

首先确定分配的存储空间的大小并申请分组的存储空间；再按照协议填写头部；然后封装，发送。发送需要调用ip_SendtoLower()接口函数。

接受：

收到分组要依次检查各错误，同时要判断是否应该由本机接收，出错和不是发送给本机的分组，调用 ip_DiscardPkt()丢弃，说明错误类型。如果是由本机接受，则调用 ip_SendtoUp()接口函数，交给系统进行后续接收处理。

流程图如下：



下面解释一下错误检测原理：

（1） 版本号错

版本号是前四位里，第一个字节右移 4 位和 0xF 取和运算，如果结果依旧是 4， 则代表版本号正确。

```
int version = 0xf & ((ipv4->version_ihl)>> 4);
if(version != 4)  {
ip_DiscardPkt(pBuffer,STUD_IP_TEST_VERSION_ERROR);
    return 1;
}
```

（2） 头部长度错

头部长度在第一个字节的后 4 位里面，将第一个字节和 0xF 取和，如果结果为 5，代表头部长度没有问题。

```
int ihl = 0xf & ipv4->version_ihl;
if(ihl < 5) {
ip_DiscardPkt(pBuffer,STUD_IP_TEST_HEADLEN_ERROR);
    return 1;
}
```

（3） 校验和错

校验和在第11-12个字节，根据mooc反码求和，先将校验和位均置0，再将所有的字节加起来，然后用 ffff 减去，如果和一开始的校验和一样，则没错。

```
int header_checksum = ntohs(ipv4->header_checksum);
int sum = 0;
for(int i = 0; i < ihl*2; i++) {
    if(i!=5)
    {
        sum += (int)((unsigned char)pBuffer[i*2] << 8);
        sum += (int)((unsigned char)pBuffer[i*2+1]);
    }
}

while((sum & 0xffff0000) != 0) {
    sum = (sum & 0xffff) + ((sum >> 16) & 0xffff);
}
unsigned short int ssum = (~sum) & 0xffff;
if(ssum != header_checksum) {
    ip_DiscardPkt(pBuffer,STUD_IP_TEST_CHECKSUM_ERROR);
    return 1;
}
```

（4） 错误目标地址

在首部的第 17 -20 个字节中，取出和本地 Ip 地址做比较，如果不等于本地地址也不等于 0xffffffff，表示目标地址出错。

```
int destination_address = ntohl(ipv4->destination_address);
if(destination_address != getIpv4Address() && destination_address != 0xffffffff) {
    ip_DiscardPkt(pBuffer,STUD_IP_TEST_DESTINATION_ERROR);
        return 1;
}
```

（5） TTL 错

在第 9 个字节，按照数据结构的定义（char + char + short + short + short），存在于 time_to_live 里面，按照规定，如果 ttl 的值是 0，则代表生命周期结束，要抛弃这个包。

```
int ttl = (int)ipv4->time_to_live;
if(ttl == 0) {
    ip_DiscardPkt(pBuffer,STUD_IP_TEST_TTL_ERROR);
    return 1;
}
```

2) 实现 IPv4 分组的封装发送。

关键代码如下图：

```
int stud_ip_Upsend(char *pBuffer,unsigned short len,unsigned int srcAddr, unsigned int dstAddr,byte protocol,byte ttl){
    char *pack_to_sent = new char[len+20];
    memset(pack_to_sent,0,len+20);
    *((Ipv4*)pack_to_sent) = Ipv4(len,srcAddr,dstAddr,protocol,ttl);
    memcpy(pack_to_sent+20,pBuffer,len);
    ip_SendtoLower(pack_to_sent,len+20);
    delete[] pack_to_sent;

    return 0;
}
```

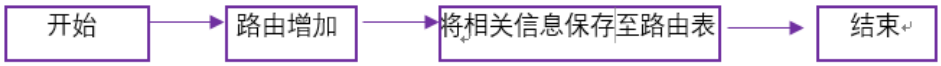
根据上层协议所提供的参数，封装 IPv4 分组，调用系统提供的发送接口函数将分组发送出去。

IPv4 分组转发实验：

1) 给出路由表初始化、路由增加、路由转发三个函数的实现流程图。

路由表初始化：只需要清空路由表即可。

路由增加：根据系统已经规定的参数进行传入，将其相关的信息保存到路由表中，比如地址、子网掩码的长度、子网掩码的值以及下一跳的位置。



（子网掩码的值可以通过移位运算和子网掩码的长度计算得到）

路由转发：获取到 IP 报文后：如果不是本地IP地址，则

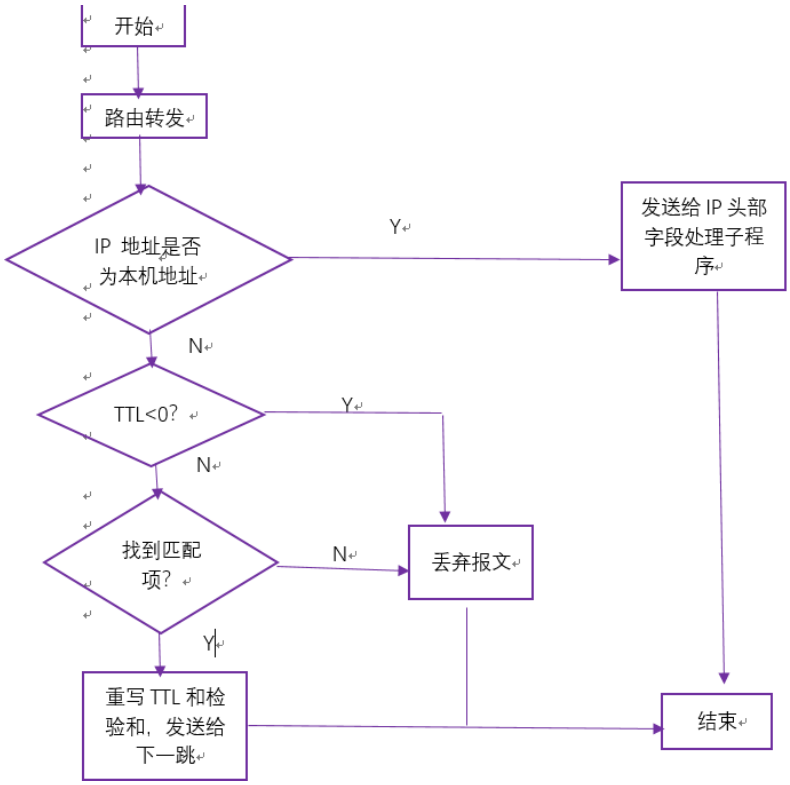
（1） 判断目的 IP 地址是否为本机地址，如果是本机地址，则调用本地 IP 的处理判断子程序 fwd_LocalRcv()进行其头部的深度判断；

（2） 如果不是本地IP地址，则提取出 TTL 和目的 IP 地址，如果 TTL小于 0，直接调用 fwd_DiscardPkt()丢弃该 IP 报文；

（3） 如果TTL符合要求，则根据此目的地址寻找路由表看是否能够找到匹配项；

（4） 如果找到匹配项，则修改 IP 头部字段中 TTL 的值并且重写校验和，再将修改过的 IP 报文发送给数据链路层，用于在网络中传输。

（5） 如果没有在路由表中没有找到匹配项，则丢弃该 IP 报文。



2) 新建数据结构的说明:

路由表数据结构：目的 IP 地址、子网掩码、下一跳地址

```
struct RNode
{
    int dest;
    int masklen;
    int nexthop;
```

3) 大量分组的情况下提高转发效率：

①路由表存储结构由线性结构改为树形结构。

②多线程工作同时查找路由表。多线程一个特点就是同时运行，可以大大节省时间。分组到来后分配到不同线程，各线程同时搜索路由表，会提高时间效率。

③路由聚合，节省路由表空间的同时，缩短寻找下一跳的时间，就不会存在大量“前缀”相同且下一跳也相同的路由表项，造成时间和空间的浪费。在代码中实现，路由表中记录目标 IP 地址的网络地址。

实验结果：

本次实验是在老师所给的程序上运行的，实验结果截图如下。

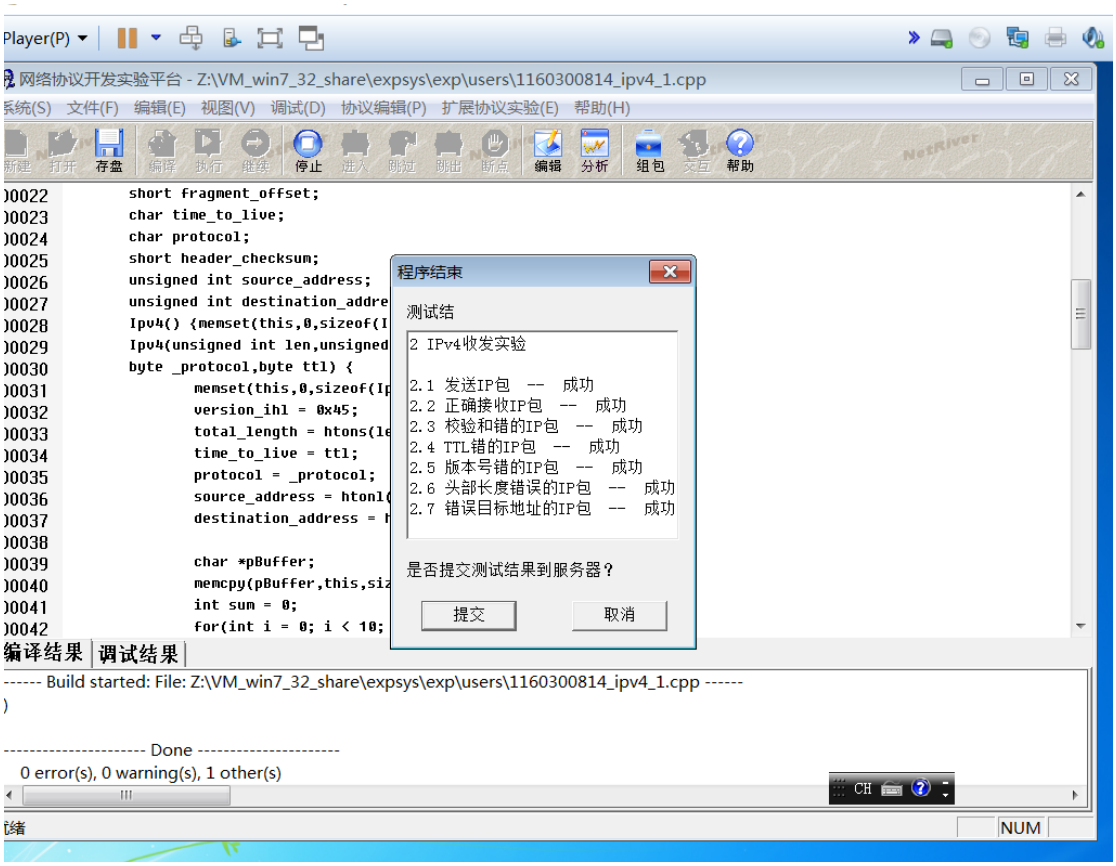
IPv4 分组收发实验：

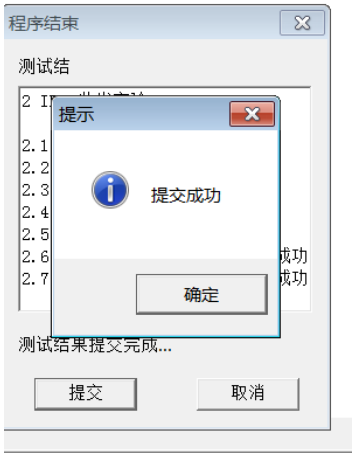
成功编译及运行结果输出如下：

```
begin test!, testItem = 1  testcase = 0
accept len = 32 packet
accept len = 166 packet
send a message to main ui, len = 53  type = 2  subtype = 1
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 1  testcase = 1
accept len = 32 packet
accept len = 166 packet
accept len = 38 packet
send a message to main ui, len = 36  type = 2  subtype = 0
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 1  testcase = 2
accept len = 32 packet
accept len = 166 packet
accept len = 38 packet
send a message to main ui, len = 36  type = 2  subtype = 0
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 1  testcase = 3
accept len = 32 packet
accept len = 166 packet
accept len = 38 packet
send a message to main ui, len = 36  type = 2  subtype = 0
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 1  testcase = 4
accept len = 32 packet
accept len = 166 packet
accept len = 38 packet
send a message to main ui, len = 36  type = 2  subtype = 0
accept len = 6 packet
result = 0
```

```
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 1  testcase = 5
accept len = 32 packet
accept len = 166 packet
accept len = 38 packet
send a message to main ui, len = 36  type = 2  subtype = 0
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 1  testcase = 6
accept len = 32 packet
accept len = 166 packet
accept len = 38 packet
send a message to main ui, len = 36  type = 2  subtype = 0
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
Test over!
```

测试结果：





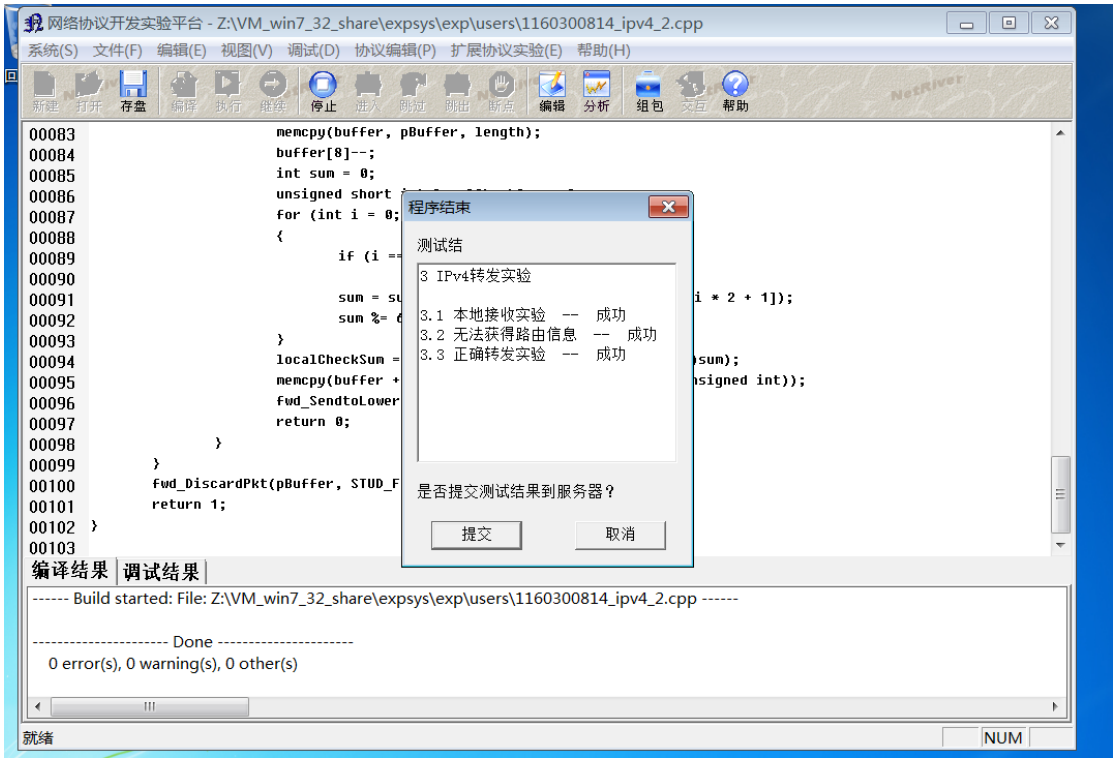
IPv4 分组转发实验：

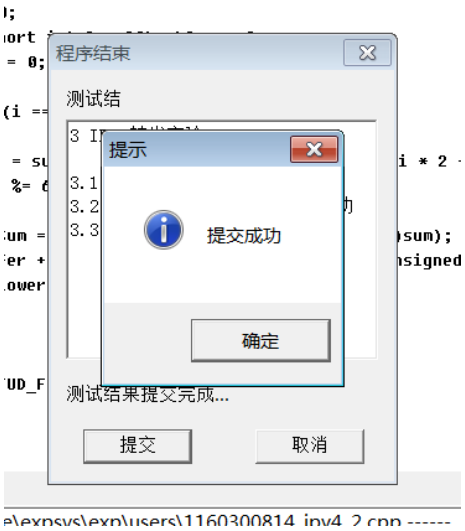
成功编译及运行结果输出如下：

Z:\VM_win7_32_share\expsys\exp\users\1160300814_ipv4_2.exe

```
begin test!, testItem = 2  testcase = 0
accept len = 32 packet
accept len = 244 packet
accept len = 41 packet
accept len = 55 packet
send a message to main ui, len = 53  type = 2  subtype = 0
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 2  testcase = 1
accept len = 32 packet
accept len = 244 packet
accept len = 41 packet
accept len = 38 packet
send a message to main ui, len = 36  type = 2  subtype = 0
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
begin test!, testItem = 2  testcase = 2
accept len = 32 packet
accept len = 244 packet
accept len = 41 packet
accept len = 55 packet
send a message to main ui, len = 53  type = 2  subtype = 0
send a message to main ui, len = 53  type = 2  subtype = 1
accept len = 6 packet
result = 0
send a message to main ui, len = 6  type = 1  subtype = 7
Test over!
```

测试结果：





问题讨论：

对如何提高转发效率，我之前并不是很懂，后来通过搜索资料才得知。目前我们举例都是几个路由，在可数范围内，才不会觉得转发的慢，但在实际应用中，却有很大的数据量，如果处理不好就不能正常应用。

如果只是完成作业，对提高效率并不需要进行深入思考，但是这正是老师鼓励我们的地方：要多多思考，多问自己为什么。而不是机械的完成作业。

心得体会：

通过本次实验，深刻理解了IP v4协议的工作原理和实际应用，同时对路由转发有了更多的理解。

学会了路由聚合，熟练应用多线程。以前一直觉得多线程很麻烦，会引起很多思考不到的问题，不过只有多加练习才能掌握多线程，了解了就不会畏惧。

理解了路由器如何根据路由表对分 组进行转发。