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	IPV4 收发和转发实验				
姓名	方烨		院系	计算学部人工智能	
班级	1903601		学号	1190202418	
任课教师	李全龙		指导教师	李全龙	
实验地点	格物 207		实验时间	2021.11.13	
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：

1. IPV4 分组收发实验

IPv4 协议是互联网的核心协议，它保证了网络节点（包括网络设备和主机）在网络层能够按照标准协议互相通信。IPv4 地址唯一标识了网络节点和网络的连接关系。在我们日常使用的计算机的主机协议栈中，IPv4 协议必不可少，它能够接收网络中传送给本机的分组，同时也能根据上层协议的要求将报文封装为 IPv4 分组发送出去。

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。

2. IPv4 分组转发实验

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。

实验内容：

1. 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。

2. 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。

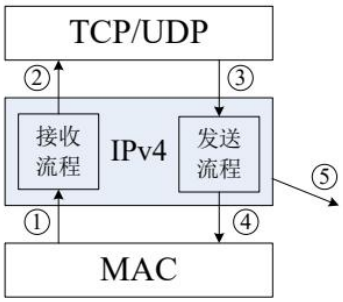
实验过程：

1.IPv4分组收发实验：

1.1 整体处理流程

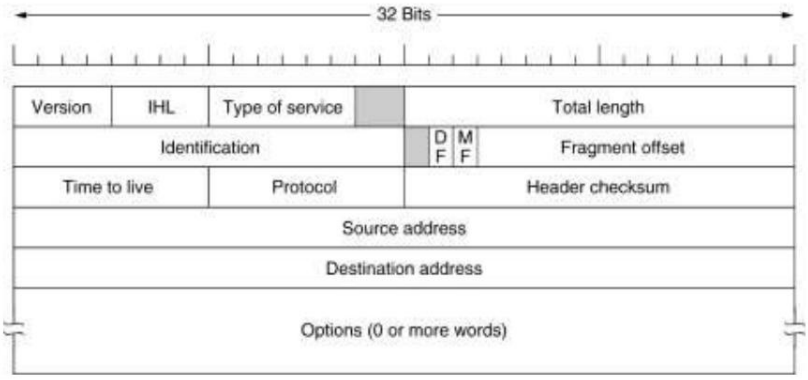
客户端接收到测试服务器发送来的 IPv4 分组后，调用接收接口函数stud_ip_recv( )（图 4-9 中接口函数1）。接收处理完成后，调用接口函数 ip_SendtoUp( ) 将需要上层协议进一步处理的信息提交给上层协议（图 4-9 中接口函数2）；或者调用函数 ip_DiscardPkt( )丢弃有错误的分组并报告错误类型（图 4-9中函数5）。

在上层协议需要发送分组时，会调用发送接口函数 stud_ip_Upsend( )（图 4-9 中接口函数3）。根据所传参数完成 IPv4 分组的封装，之后调用接口函数ip_SendtoLower( )把分组交给下层完成发送（图 4-9 中接口函数4）。



实验接口函数示意图

1.2 IPv4 分组头部格式



IPv4分组头部格式

1.3 接收流程

1.3.1 接口函数处理步骤：

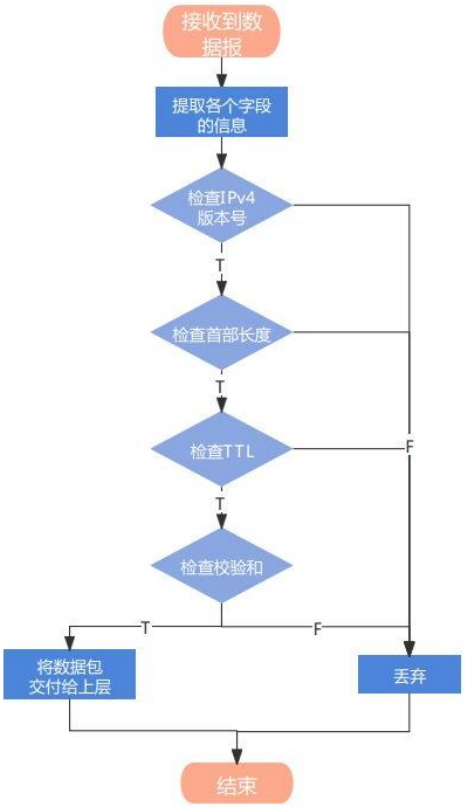
在接口函数 stud_ip_recv( )中，需要完成下列处理步骤：

① 检查接收到的 IPv4 分组头部的字段，包括版本号（Version）、头部长度（IP Head length）、生存时间（Time to live）以及头校验和（Header checksum）字段。对于出错的分组调用 ip_DiscardPkt( )丢弃，并说明错误类型。

② 检查 IPv4 分组是否应该由本机接收。如果分组的目的地址是本机地址或广播地址，则说明此分组是发送给本机的；否则调用ip_DiscardPkt( )丢弃，并说明错误类型。

③ 如果 IPV4 分组应该由本机接收，则提取得到上层协议类型，调用 ip_SendtoUp( )接口函数，交给系统进行后续接收处理。

1.3.2 接收流程示意图



1.4 发送流程

1.4.1 接口函数处理步骤：

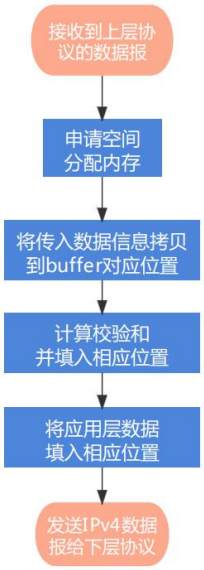
在接口函数 stud_ip_Upsend( )中，需要完成下列处理步骤：

① 根据所传参数（如数据大小），来确定分配的存储空间的大小并申请分组的存储空间。

② 按照 IPv4 协议标准填写 IPv4 分组头部各字段，标识符（Identification）字段可以使用一个随机数来填写。（注意：部分字段内容需要转换成网络字节序）

③ 完成 IPv4 分组的封装后，调用 ip_SendtoLower( )接口函数完成后续的发送处理工作，最终将分组发送到网络中。

1.4.2 发送流程示意图



1.5 检测原理

①检查IPv4版本号

版本号在第0个字节的高4位，因此需要先进行移位；

再将其和4比较即可。若版本号不正确，则丢包，并传递错误原因参数。

```
int version = pBuffer[0] >> 4; //IP 版本号
//检查 IPv4 版本号
if (version != 4){
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_VERSION_ERROR) ;
    return 1;
}
```

②检查首部长度

首部长度信息存储在第0字节的后4位，所以通过“&”运算进行提取低位信息。

```
int headLen = pBuffer[0] & 0xf; //首部长度
//检查头部长度，<20 字节有误需丢弃
if (headLen < 5){
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_HEADLEN_ERROR) ;
    return 1;
}
```

③检查TTL

TTL存储在第8个字节，直接读取即可。

```
short ttl = (unsigned short)pBuffer[8]; //TTL
//检查 TTL，TTL=0 则丢弃
if (ttl == 0){
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_TTL_ERROR) ;
    return 1;
}
```

④检查校验和

检验和字段保存在IP数据报的第10个字节，直接读取即可，需要注意的是此处要将网络字节序转换为本地字节序。

```
short checksum = ntohs(*(unsigned short *)(pBuffer + 10)); //校验和
//如果校验和有误，丢弃
if(calChecksum(pBuffer, headLen) != 0xffff){
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_CHECKSUM_ERROR);
    return 1;
}
```

⑤检查目的地址

目的地址保存在IP数据报的第16个字节，读取后将其转换为本地字节序。

```
int desIP = ntohl(*(unsigned int *)(pBuffer + 16)); //目的 IP 地址
//检查目的地址，不是本机地址或广播地址则丢弃
if(desIP != getIpv4Address() && desIP != 0xffff){
```

```
        ip_DiscardPkt(pBuffer, STUD_IP_TEST_DESTINATION_ERROR);
        return 1;
    }
```

2.IPv4分组转发实验：

1.1 整体处理流程

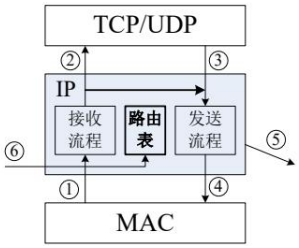
在下层接收接口函数 Stud_fwd_deal( )中（图中接口函数1），实现分组接收处理。主要功能是根据分组中目的 IPv4 地址结合对应的路由信息对分组进行处理。

分组 A. 需要上交，则调用接口函数 Fwd_LocalRcv( )（图中接口函数 2）；

B. 需要丢弃，则调用函数 Fwd_DiscardPkt( )（图中函数5）；

C. 需要转发，则进行转发操作。转发操作的实现要点包括，TTL值减1，然后重新计算头校验和，最后调用发送接口函数 Fwd_SendtoLower( )（图中接口函数4）将分组发送出去。

注意：接口函数Fwd_SendtoLower( )比前面实验增加了一个参数 pNxtHopAddr，要求在调用时传入下一跳的 IPv4 地址，此地址是通过查找路由表得到的。



1.2 转发处理流程

1.2.1 路由表维护：

1) stud_Route_Init ( )函数中，对路由表进行初始化。

2) stud_route_add ( )函数中，完成路由的增加。

1.2.2 接口函数处理步骤：

在 stud_fwd_deal ( )函数中，需要完成下列分组接收处理步骤：

1) 查找路由表。根据相应路由表项的类型来确定下一步操作，错误分组调用函数 fwd_DiscardPkt ( )进行丢弃，上交分组调用接口函数 fwd_LocalRcv ( )提交给上层协议继续处理，转发分组进行转发处理。注意，转发分组还要从路由表项中获取下一跳的 IPv4地址。

2) 转发处理流程。对 IPv4 头部中的 TTL 字段减 1，重新计算校验和，然后调用下层接口 fwd_SendtoLower ( )进行发送处理。

1.2.3 数据结构说明：

1) 路由表项结构体

```
// 路由表项结构体
struct routeTable{
    unsigned int dstIP;    //目的 IP 地址
    unsigned int mask;     //掩码
    unsigned int maskLen; //掩码长度
    unsigned int nexthop; //下一跳
};
```

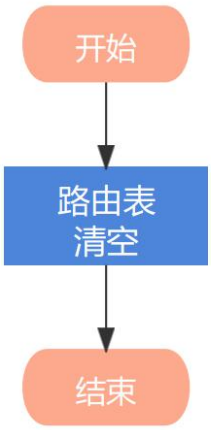
2) 路由表

由路由表项组成的容器结构

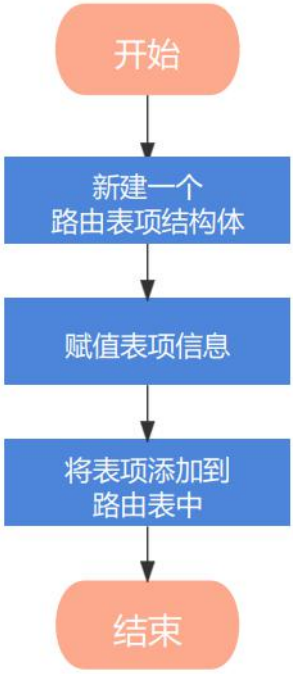
```
1.  //路由表
2.  vector<routeTable> route;
```

1.2.4 函数流程图

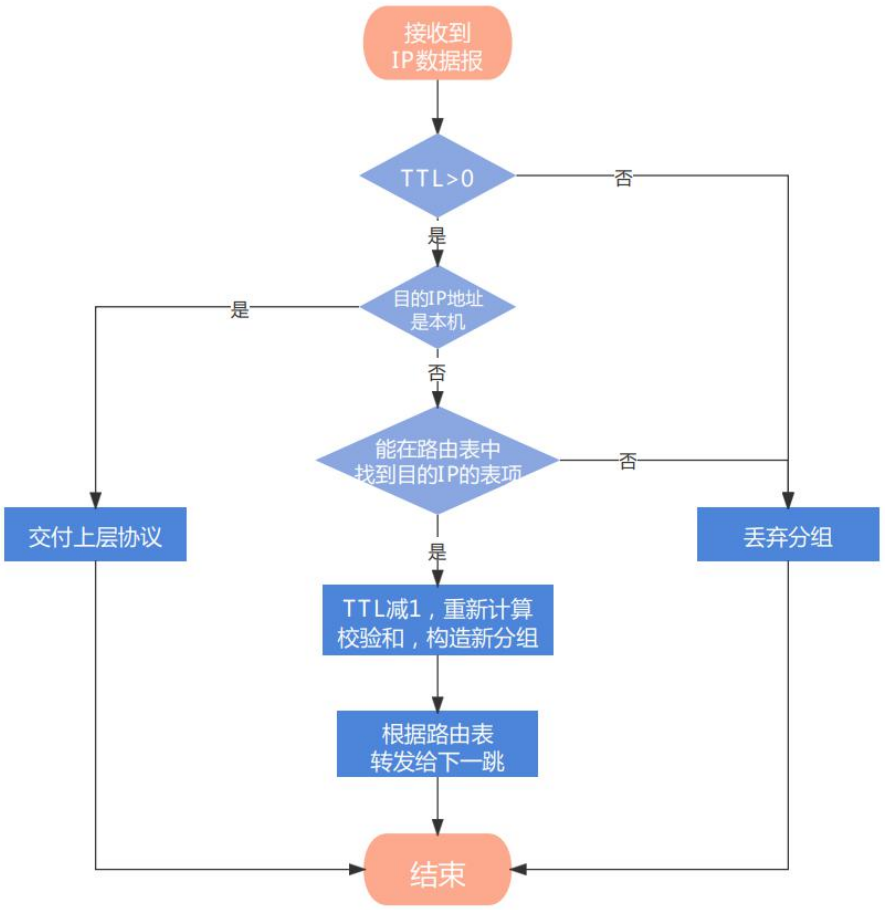
1) 路由表初始化



2) 路由增加



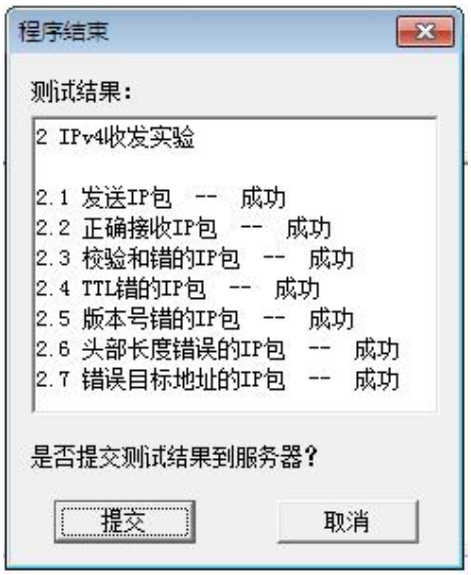
3) 路由转发



实验结果：

1.IPv4分组收发实验：

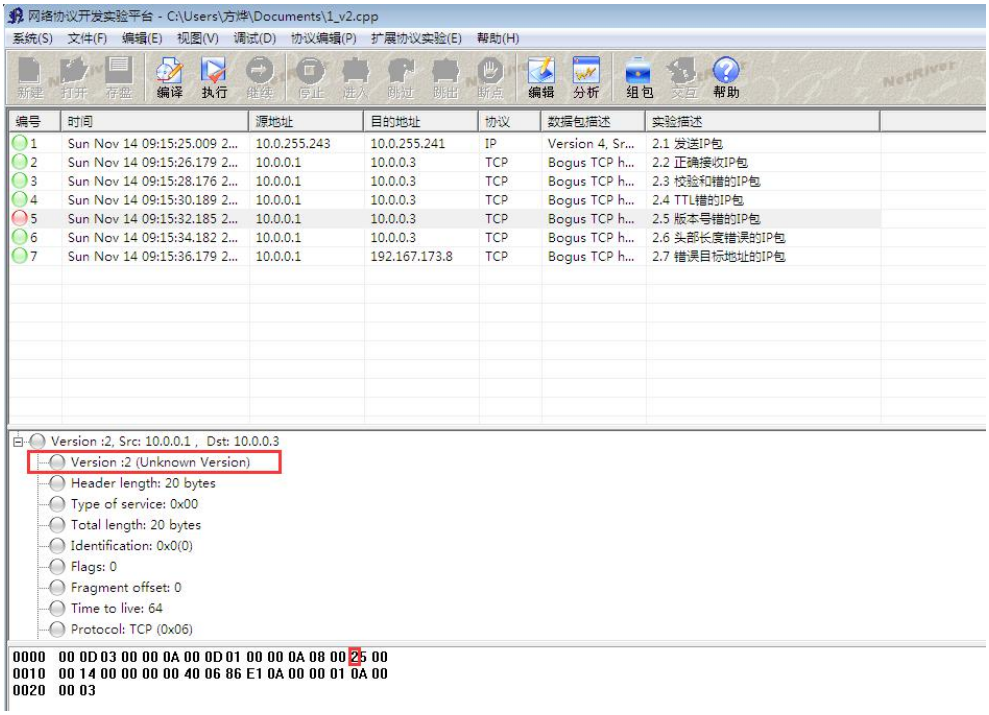
1.1 测试结果



1.2 测试结果分析

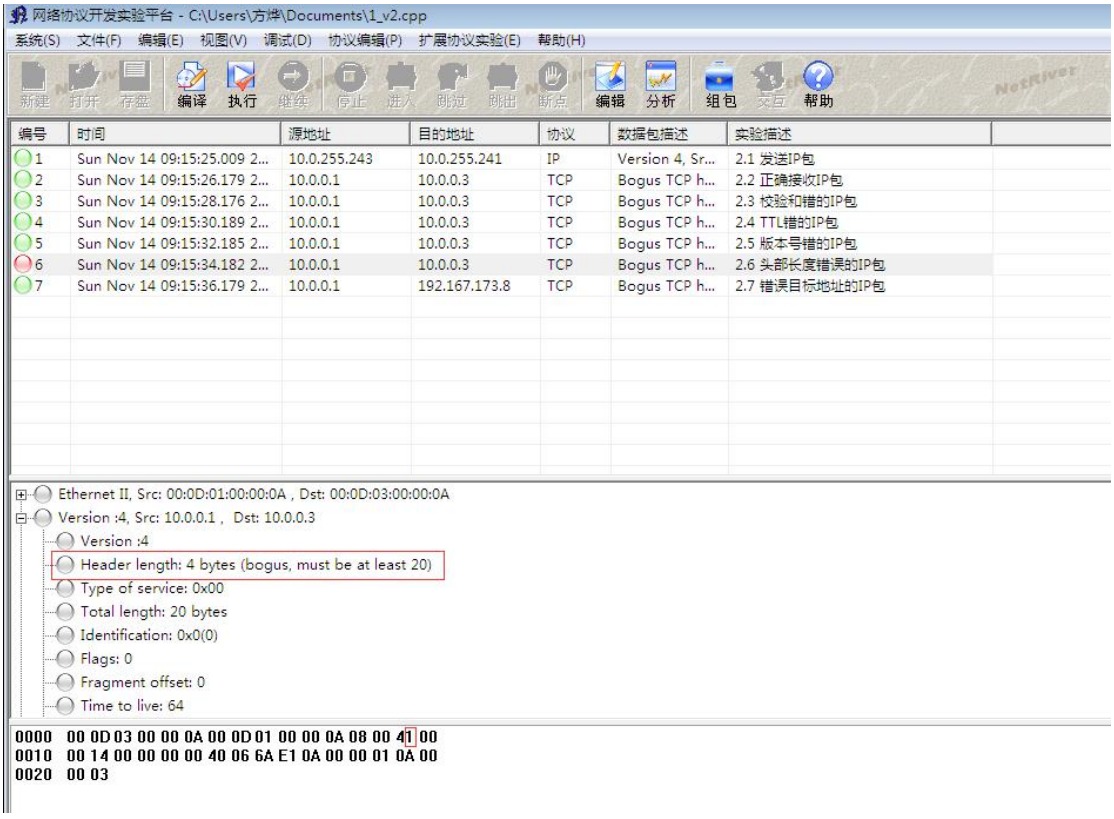
1.2.1 版本号错误的数据报

下图中数据报的version字段为2，与IPv4数据报的version字段为4不符。



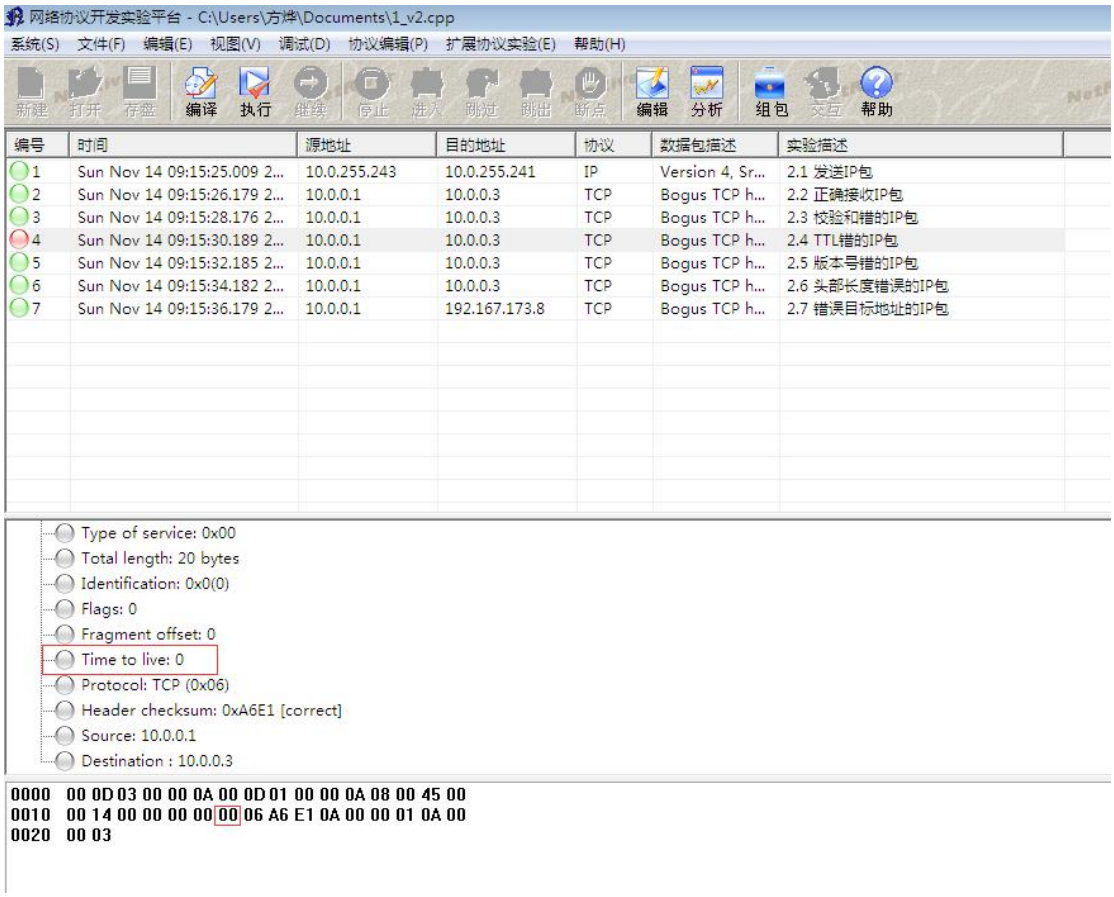
1.2.2 首部长度错误的数据报

下图中数据报首部长度字段为1，表示首部长度为4个字节<20字节，不符合协议。



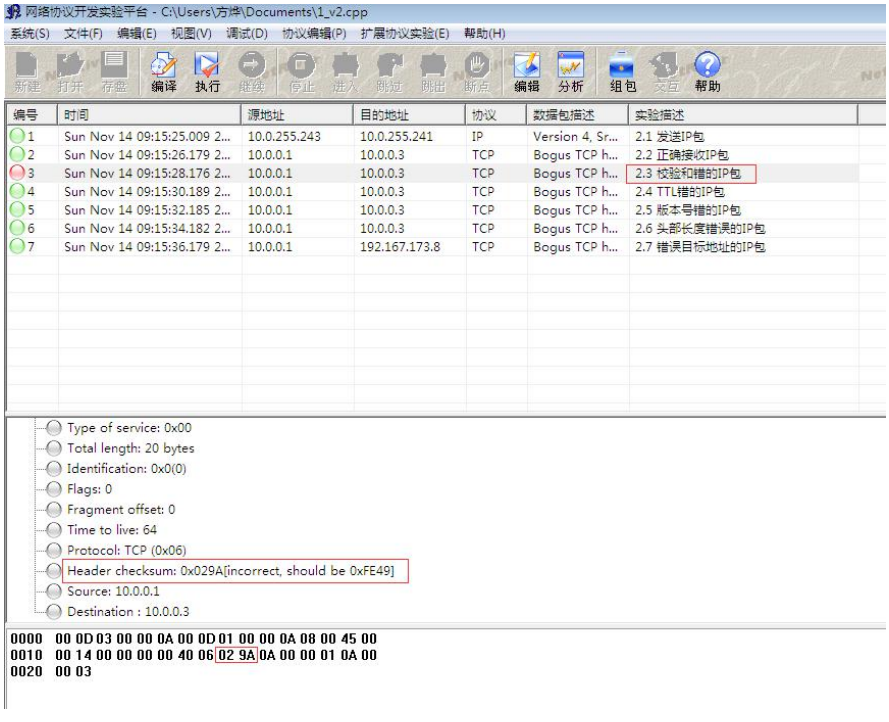
1.2.3 TTL错误的数据报

下图中数据报TTL字段值为0，说明该数据报已失效。



1.2.4 校验和错误的数据报

下图中数据报校验和值为0x029A，和计算出的校验和值0xFE49不符。



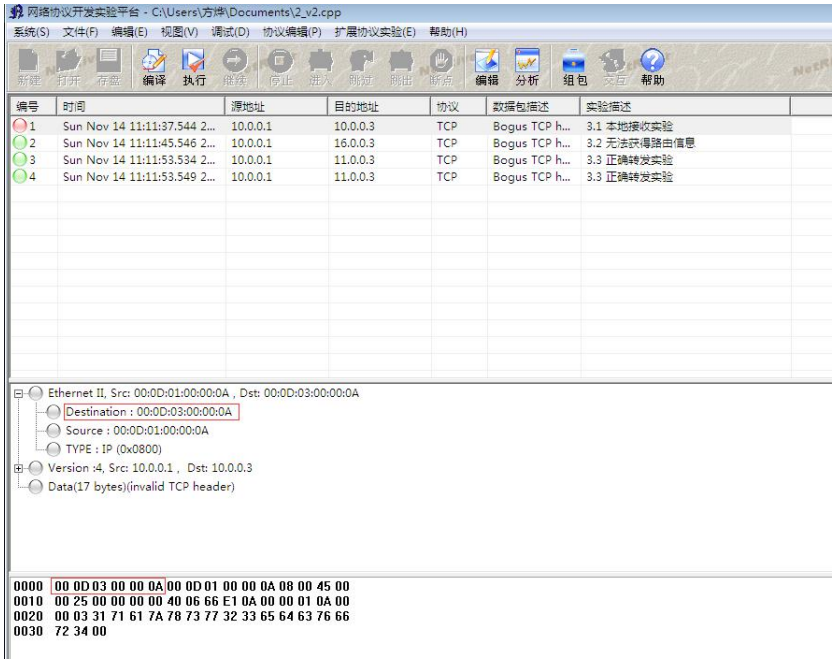
2.IPv4分组转发实验：

2.1 测试结果

2.2 测试结果分析

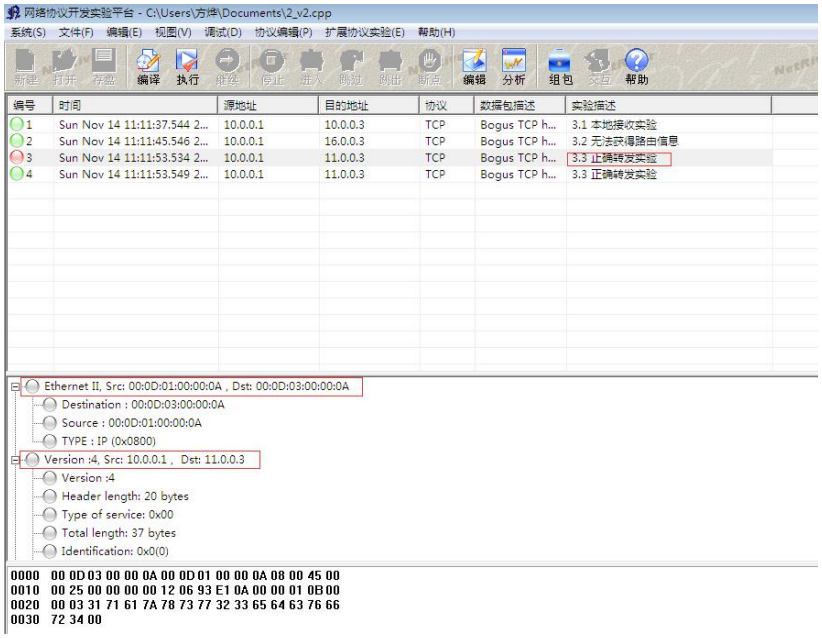
2.2.1 本地接收

目的地址是本机，直接接收



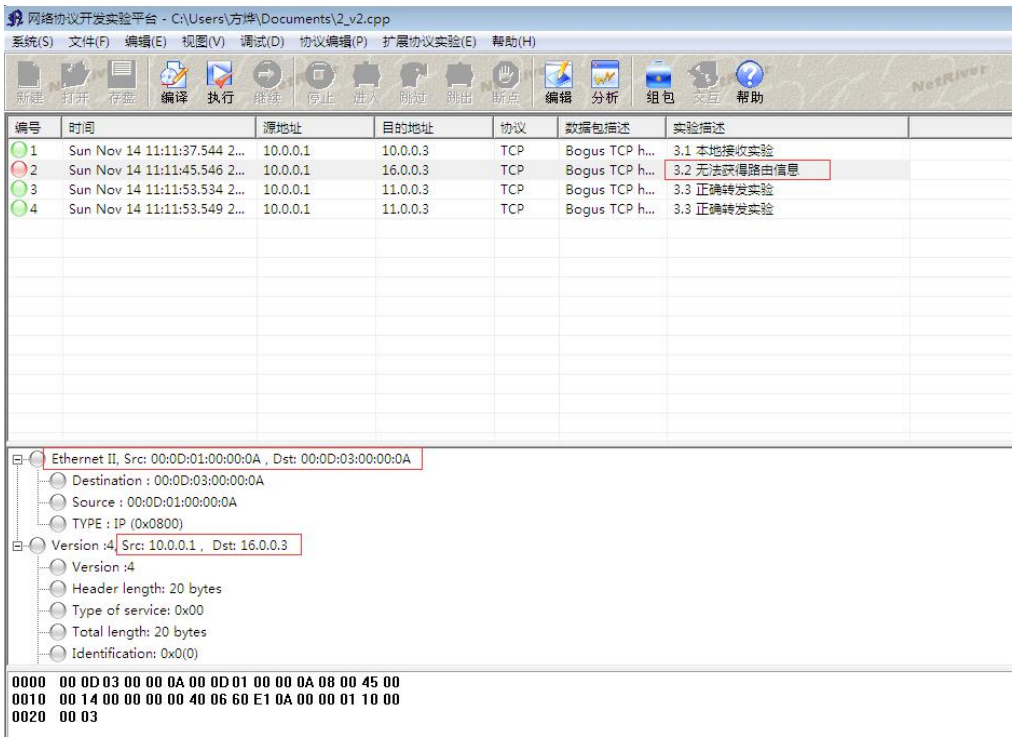
2.2.2 正确转发

路由表中存在目的地址可以匹配，且数据合法，正确转发



2.2.3 无法获得路由信息

在路由表中匹配不到相应项，丢弃



问题讨论：

分析在存在大量分组的情况下如何提高转发效率。

1.优化I/O，DMA，减少内存管理操作

2.优化中断分发

3.优化路由查找算法

1).分离路由表和转发表，路由表和转发表同步采用RCU机制

2).尽量采用线程局部数据

3).采用hash/trie方式以及DxR或者我设计的DxRPro定位结构

4.优化lock

1).查询定位局部表，无锁（甚至RW锁都没有）不禁止中断

2).临界区和内核线程关联，不禁中断，不禁抢占（其实内核编译时抢占已经关闭了）

3).优先级锁队列替换争抢模型，维持cache热度

4).采用Windows的自旋锁机制

心得体会：

本次实验个人感觉难度较前两次稍低，不过实验环境在win7虚拟机上稍微有点麻烦，主要的难点和坑点是对于位的处理，找对应的字段时需要头脑清晰避免出错。通过实验我对于IPv4的收发过程更为熟悉，也更为熟悉了IP数据段的结构。