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 分组收发/转发实验				
姓名	李世轩		院系	软件工程	
班级	2037102		学号	120L022109	
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	2022 年 10 月 21 日	
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 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。

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) 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。

3) 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。

4) 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”。

5) 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收发实验

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。

```
int stud_ip_recv(char *pBuffer,unsigned short length)
{
    byte tempByte = pBuffer[0];
    byte version = (tempByte&0xf0)>>4;
    byte headLength = tempByte & 0x0f;
    byte TTL = pBuffer[8];
    unsigned int destAddr = (pBuffer[16]<<24)+(pBuffer[17]<<16)+(pBuffer[18]<<8)+pBuffer[19];
    unsigned int checksum = (pBuffer[10]<<8)+pBuffer[11];
    if((version^0x04)!=0x00) {
        ip_DiscardPkt(pBuffer, type: STUD_IP_TEST_VERSION_ERROR);
        goto error;
    }
    if(!(headLength>=0x5&&headLength<=0xc)){
        ip_DiscardPkt(pBuffer, type: STUD_IP_TEST_HEADLEN_ERROR);
        goto error;
    }
    if(TTL <= 0x0){
        ip_DiscardPkt(pBuffer, type: STUD_IP_TEST_TTL_ERROR);
        goto error;
    }
    if(getIpv4Address()!=destAddr){
        ip_DiscardPkt(pBuffer, type: STUD_IP_TEST_DESTINATION_ERROR);
        goto error;
    }
    if(checksum!=0){
        ip_DiscardPkt(pBuffer, type: STUD_IP_TEST_CHECKSUM_ERROR);
        goto error;
    }
    ip_SendtoUp( pBuffer: pBuffer+headLength, length: length-headLength);
    return 0;
    error:
```

首先，通过一些位运算取出版本号（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。另外这里还有一个函数用来计算检验和。

```
    return 0;
}
unsigned int getCheckSum(char *head)
{
    unsigned char *p = (unsigned char*)head;
    unsigned int checksum = 0;
    for(int i = 0; i < 19; i+=2)
    {
        checksum += (p[i]<<8) + (p[i+1]);
    }
    checksum = (checksum >> 16) + (checksum & 0xffff);
    return (~checksum) & 0xffff;
}
```

接下来，通过获取的信息来判断这个分组的处理方式。

如果版本号不为4，则丢弃包，并报告错误类型为STUD_IP_TEST_VERSION_ERROR。

如果头部长度不在5-12之间，则丢弃包，并报告错误类型为STUD_IP_TEST_HEADLEN_ERROR。

如果TTL小于等于0，则丢弃包，并报告错误类型为STUD_IP_TEST_TTL_ERROR。

如果目的地址不为本机地址，则丢弃包，并报告错误类型为STUD_IP_TEST_DESTINATION_ERROR。

如果计算完的检验和不为0，则丢弃包，并报告错误类型为STUD_IP_TEST_CHECKSUM_ERROR。

此时若所有判断均未出错，则将包传给上层协议。

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。

```
{
    unsigned short length=len+20;
    byte temp[length];
    memset( Dst: temp, Val: 0, Size: length);
    temp[0]=0x45;
    temp[2]=(length&0xff00)>>8;
    temp[3]=length&0xff;
    temp[4]=0;
    temp[5]=0;
    temp[6]=0;
    temp[7]=0;
    temp[8]=ttl;
    temp[9]=protocol;
    temp[10]=0;
    temp[11]=0;
    temp[12]=(srcAddr&0xff000000)>>24;
    temp[13]=(srcAddr&0xff0000)>>16;
    temp[14]=(srcAddr&0xff00)>>8;
    temp[15]=(srcAddr&0xff);
    temp[16]=(dstAddr&0xff000000)>>24;
    temp[17]=(dstAddr&0xff0000)>>16;
    temp[18]=(dstAddr&0xff00)>>8;
    temp[19]=(dstAddr&0xff);

    unsigned int checksum = getCheckSum( head: temp);
    temp[10] = (checksum&0xff00)>>8;
    temp[11]=checksum&0xff;

    char buffer[65535];
    memcpy( Dst: buffer,  Src: (char *)temp,  Size: 24);
    memcpy( Dst: buffer+20,  Src: pBuffer,  Size: strlen( Str: pBuffer)+1);

    ip_SendtoLower( pBuffer: buffer,length);
    return 0;
```

这个函数要做的就是将参数中信息放入一个ip分组中，这里不赘述，需要注意的就是，在计算检验和时，应该先将对应字段置为0，再进行计算。

IPv4转发实验

路由表维护

需要完成下列分组接收处理步骤：

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。

完成这两个函数需要完成对路由表的设计，在这里使用C++提供的键值对容器map。它相对与链表查询速度是非常快的，这对路由表的高性能要求是非常符合的。

另外，map内部的实现自建一颗红黑树，这颗树具有对数据自动排序的功能。在map内部所有的数据都是有序的。

这个设计方式需要维护一个全局的静态变量。

```
map<unsigned int,unsigned int> routeTable;
```

stud_Route_Init ( )函数：

```
void stud_Route_Init()
{
    routeTable.clear();
}
```

stud_route_add ( )函数：

```
void stud_route_add(stud_route_msg *proute)
{
    unsigned int maskedDest = (ntohl( netlong: proute->dest))
            &(0xffffffff<<(32-htonl( hostlong: proute->masklen)));
    unsigned int nextHop = (ntohl( netlong: proute->nexthop));
    routeTable.insert( x: map<unsigned int, unsigned int>::value_type( &: maskedDest, &: nextHop));

}


int stud_fwd_deal(char *pBuffer, int length)
```

在这函数中将路由信息的目的IP地址和掩码长度处理后（获取网络号）作为键，将下一跳的信息作为值，存入路由表map中。

转发处理流程

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。

```
int stud_fwd_deal(char *pBuffer, int length)
{
    int IHL = pBuffer[0]&0xf;
    int TTL = (int)pBuffer[8];
    int destIp = ntohl( netlong: *(unsigned int*)(pBuffer+16));
    if(destIp == getIpv4Address()){
        fwd_LocalRcv(pBuffer,length);
        return 0;
    }
    if(TTL<=0){
        fwd_DiscardPkt(pBuffer, type: STUD_FORWARD_TEST_TTLERROR);
        return 1;
    }
    std::map<unsigned int,unsigned int>::iterator iter;
    iter = routeTable.find( x: destIp);
    if(iter!=routeTable.end()){
        char *buffer = new char[length];
        memcpy( Dst: buffer, Src: pBuffer, Size: length);
        //TTL - 1
        buffer[8]--;
        unsigned short int checkSumNew = getChecksum( pBuffer: buffer,IHL);
        memcpy( Dst: buffer+10, Src: &checkSumNew, Size: sizeof(unsigned short int));
        fwd_SendtoLower( pBuffer: buffer,length, nexthop: iter->second);
        return 0;
    }else{
        fwd_DiscardPkt(pBuffer, type: STUD_FORWARD_TEST_NOROUTE);
        return 1;
    }
}
```

首先从报文中获取IHL（头部长度），TTL，目标IP地址的信息。

如果目的IP地址为本机地址，则想上层协议转发。

如果TTL小于等于0，则说明其为不合法的报文，将其丢弃，报告信息STUD_FORWARD_TEST_TTLERROR。

如果报文合法，从路由表中获取下一跳地址。

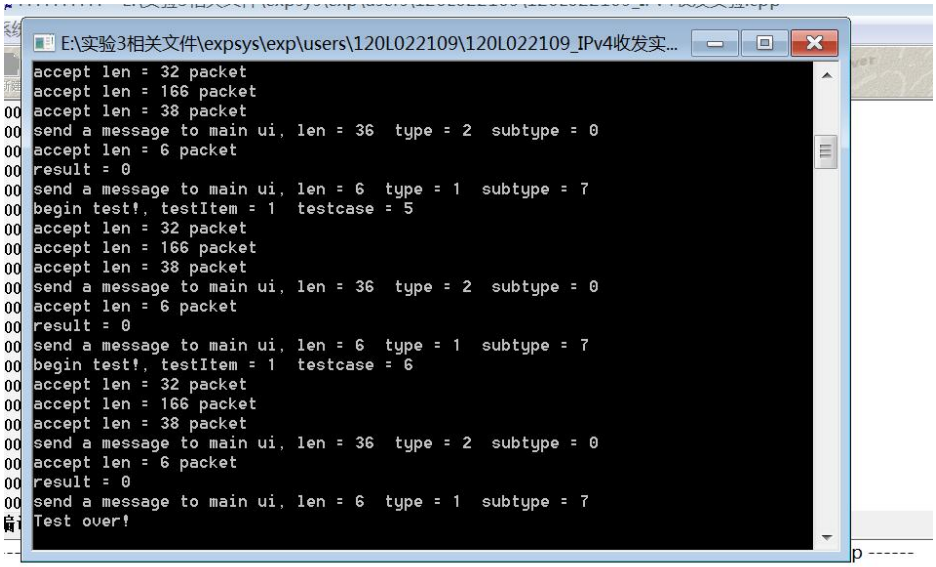
如果未找到下一跳路由，则将其丢弃，报告信息STUD_FORWARD_TEST_NOROUTE。

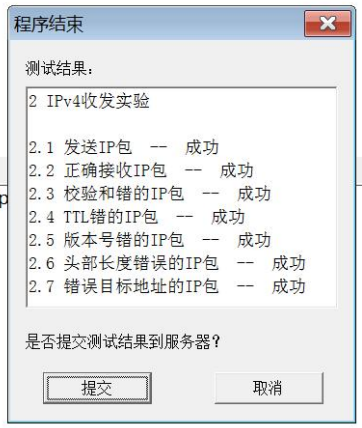
如果找到下一跳路由，将TTL减一并重新计算检验和。并向下层协议发送报文。

```
unsigned short getChecksum(char *pBuffer,int IHL){
    unsigned short int checkSumNew = 0;
    unsigned int sum = 0;
    for(int i = 0; i < 2*IHL; i++)
    {
        if(i!=5)
        {
            sum += ((unsigned char)pBuffer[2*i]<<8)+((unsigned char)pBuffer[i*2+1]);
        }
    }
    while((sum&0xffff0000)!=0){
        sum = (sum&0xffff)+((sum>>16)&0xffff);
    }
    checkSumNew = (short unsigned int)sum;
    checkSumNew = htons( hostshort: ~checkSumNew);
    return checkSumNew;
}
```

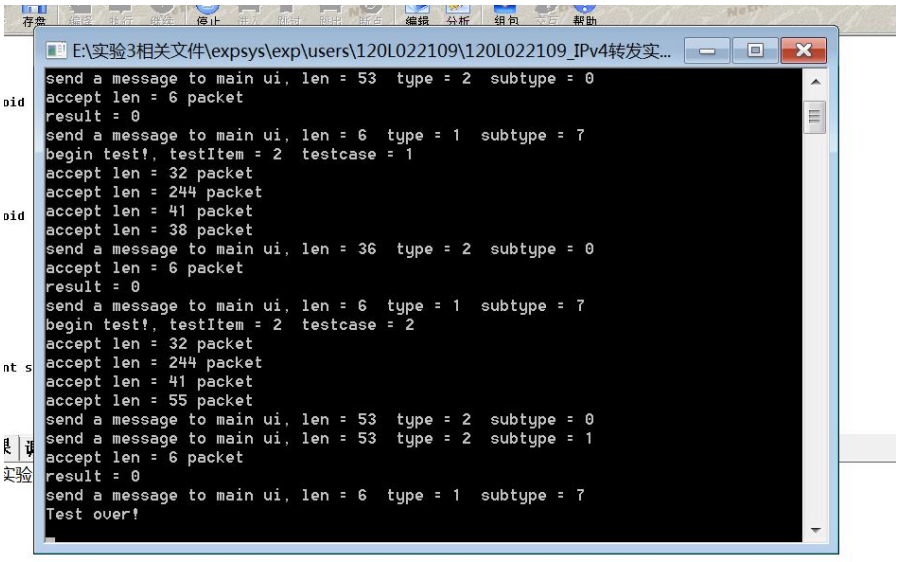
实验结果：

IPv4收发实验



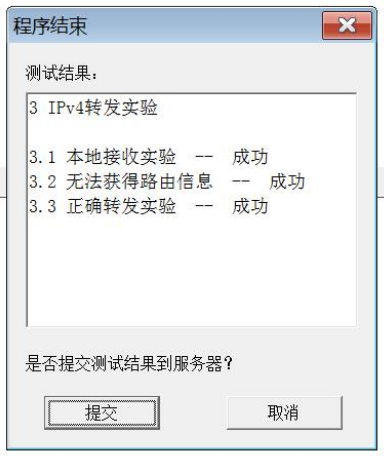


IPv4转发实验



```
l(proute->masklen)));

est,nextHop));
```

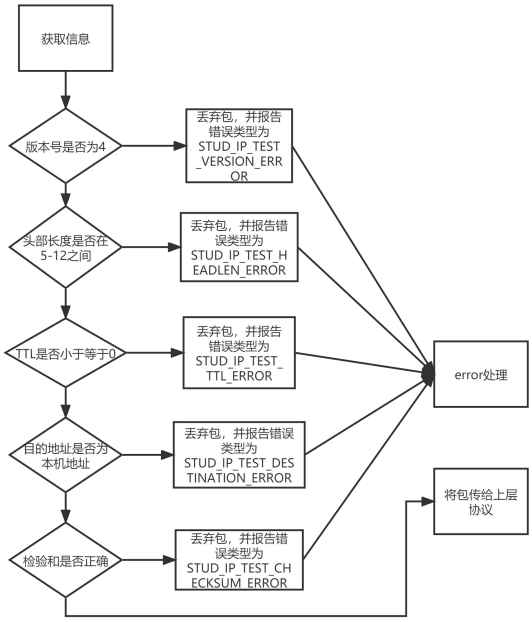



```
.cpp ------ up-to-date.
```

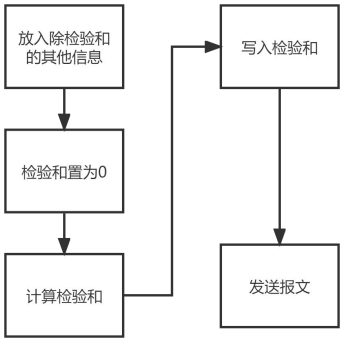
问题讨论：

1) 要求给出发送和接收函数的实现程序流程图；

接收函数



发送函数



2) 要求给出版本号（Version）、头部长度（IP Head length）、生存时间（Time to live）以及头校验和（Header checksum）字段的错误检测原理，并根据实验具体情况给出错误的具体数据，例如如果为头部长度错，请给出收到的错误的 IP 分组头部长度字段值为多少。

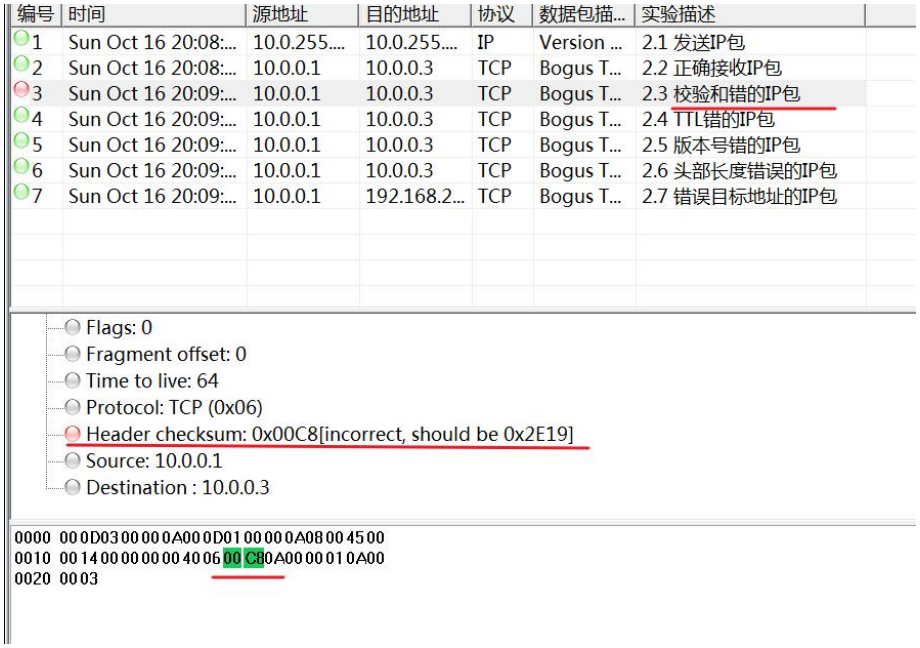
如果版本号不为4，则丢弃包，并报告错误类型为STUD_IP_TEST_VERSION_ERROR。

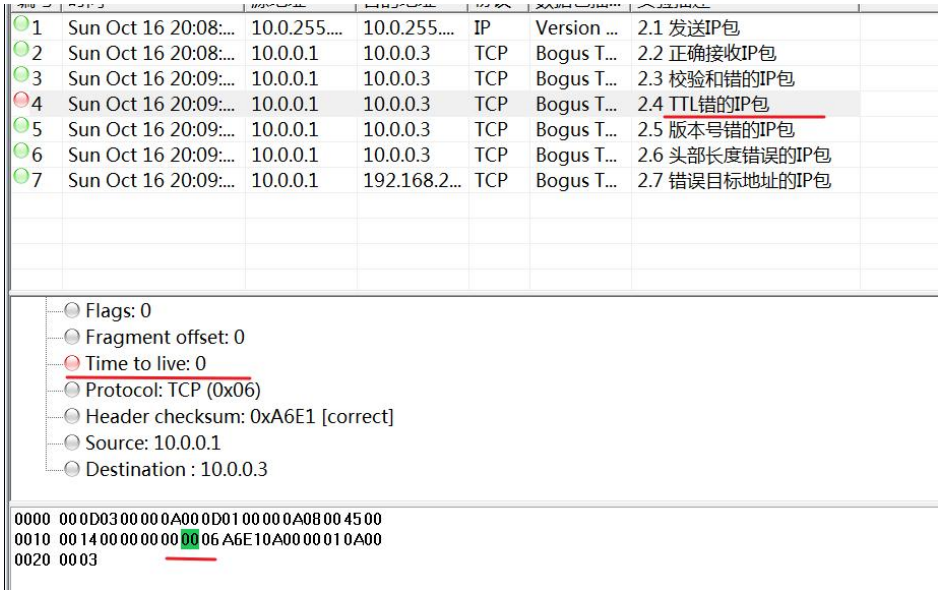
如果头部长度不在5-12之间，则丢弃包，并报告错误类型为STUD_IP_TEST_HEADLEN_ERROR。

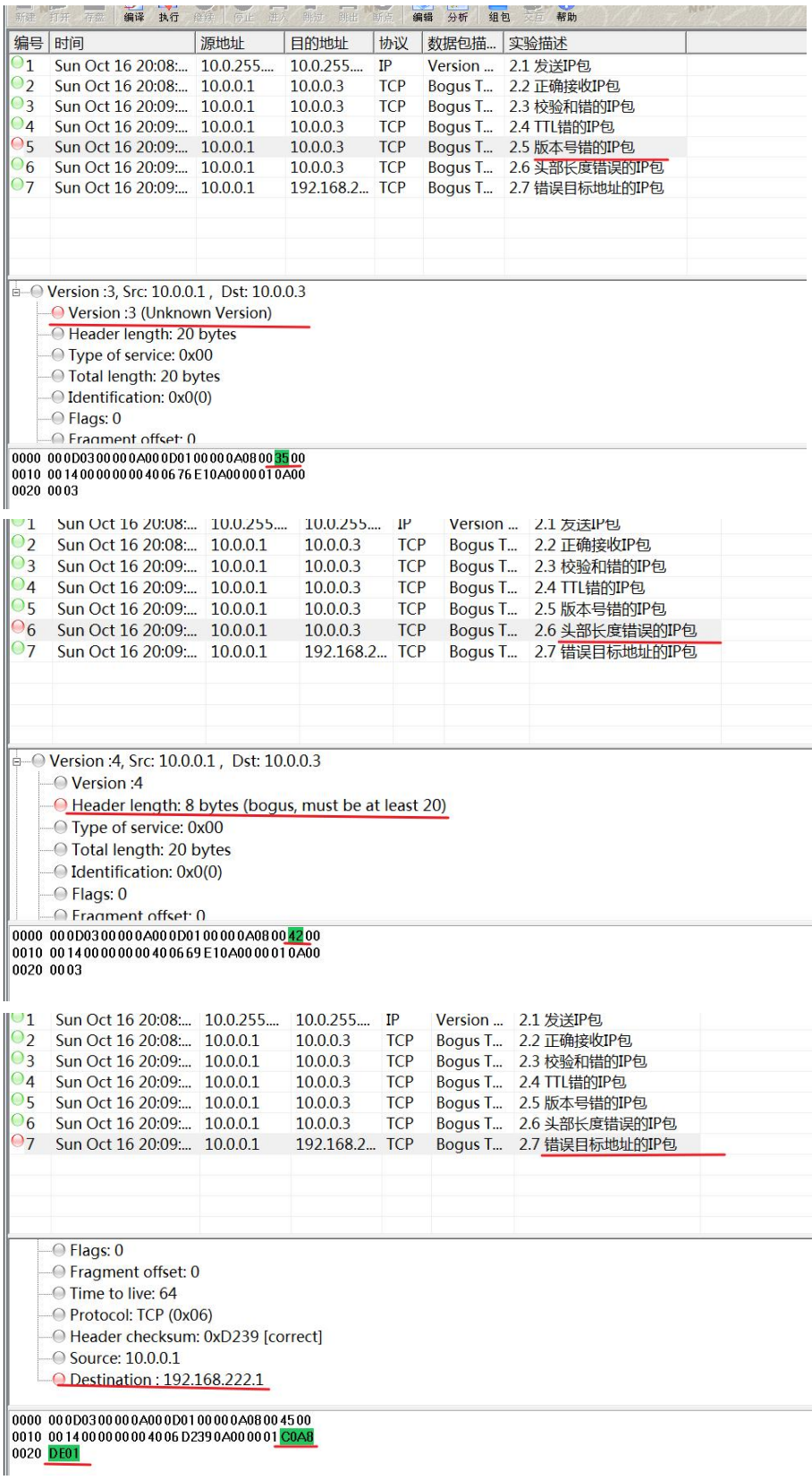
如果TTL小于等于0，则丢弃包，并报告错误类型为STUD_IP_TEST_TTL_ERROR。

如果目的地址不为本机地址，则丢弃包，并报告错误类型为STUD_IP_TEST_DESTINATION_ERROR。

如果计算（使用生成检验和的算法）完的检验和不为0，则丢弃包，并报告错误类型为STUD_IP_TEST_CHECKSUM_ERROR。



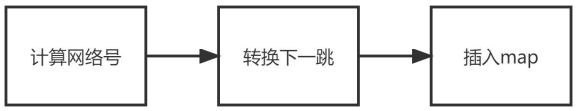



编译 执行 编辑 分析 组包 帮助
编号 时间 源地址 目的地址 协议 数据包描... 实验描述
1 Sun Oct 16 20:08:... 10.0.255.... 10.0.255.... IP Version ... 2.1 发送IP包
2 Sun Oct 16 20:08:... 10.0.0.1 10.0.0.3 TCP Bogus T... 2.2 正确接收IP包
3 Sun Oct 16 20:09:... 10.0.0.1 10.0.0.3 TCP Bogus T... 2.3 校验和错的IP包
4 Sun Oct 16 20:09:... 10.0.0.1 10.0.0.3 TCP Bogus T... 2.4 TTL错的IP包
5 Sun Oct 16 20:09:... 10.0.0.1 10.0.0.3 TCP Bogus T... 2.5 版本号错的IP包
6 Sun Oct 16 20:09:... 10.0.0.1 10.0.0.3 TCP Bogus T... 2.6 头部长度错误的IP包
7 Sun Oct 16 20:09:... 10.0.0.1 192.168.2... TCP Bogus T... 2.7 错误目标地址的IP包
Version :3, Src: 10.0.0.1 , Dst: 10.0.0.3
Version :3 (Unknown Version)
Header length: 20 bytes
Type of service: 0x00
Total length: 20 bytes
Identification: 0x0(0)
Flags: 0
Fragment offset: 0
0000 000D0300000A000D0100000A080035 00
0010 0014000000004006 76E10A000010A00
0020 0003
Version :4, Src: 10.0.0.1 , Dst: 10.0.0.3
Version :4
Header length: 8 bytes (bogus, must be at least 20)
Type of service: 0x00
Total length: 20 bytes
Identification: 0x0(0)
Flags: 0
Fragment offset: 0
0000 000D0300000A000D0100000A080042 00
0010 0014000000004006 69E10A000010A00
0020 0003
Flags: 0
Fragment offset: 0
Time to live: 64
Protocol: TCP (0x06)
Header checksum: 0xD239 [correct]
Source: 10.0.0.1
Destination : 192.168.222.1
0000 000D0300000A000D0100000A080045 00
0010 0014000000004006 D2390A000001 C0AB
0020 DE01

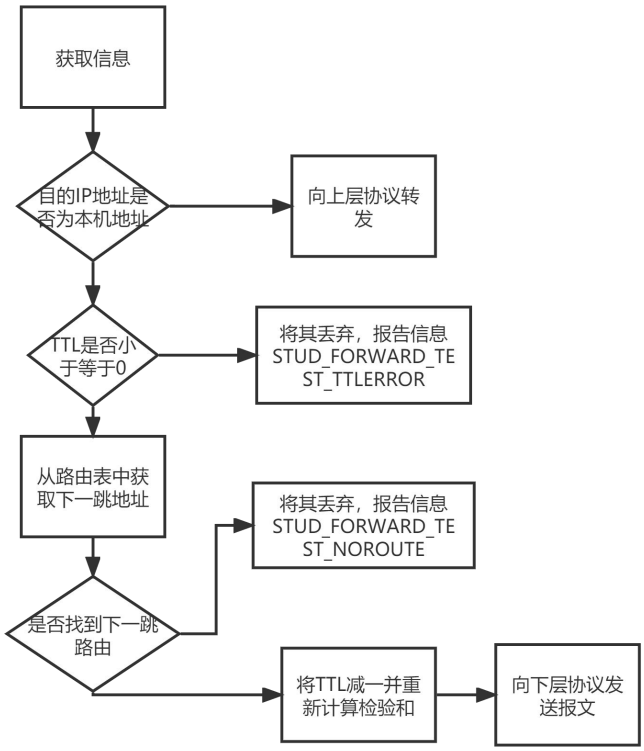
3) 要求给出路由表初始化、路由增加、路由转发三个函数的实现流程图；

路由表初始化

路由增加



路由转发



心得体会：

更加细致地了解IPv4报文的结构；

熟悉C语言的一些位操作

熟悉一些C++的网络相关的函数