哈尔滨工业大学

**<<计算机网络>>**

**实验报告**

**(2018年度春季学期)**

|  |  |
| --- | --- |
| **姓名：** | **李博** |
| **学号：** | **1150310116** |
| **学院：** | **计算机科学与技术学院** |
| **教师：** | **聂兰顺** |

## 实验一 HTTP代理服务器的设计与实现

## 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。

## 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。

## 三、实验过程及结果

3.1 发送/接受函数流程图

接受函数



发送函数



3.2 函数说明

3.2.1 接收接口: int stud\_ip\_recv(char \* pBuffer, unsigned short length)

* 参数:

pBuffer: 指向接收缓冲区的指针，指向 IPv4 分组头部

length: IPv4 分组长度

* 返回值:

0: 成功接收 IP 分组并交给上层处理

1:IP 分组接收失败

* 实现:

int stud\_ip\_recv(char \*pBuffer,unsigned short length)

{

unsigned short version = pBuffer[0] >> 4;

if(version != 4)

{

ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_VERSION\_ERROR);

return 1;

}

unsigned short headlen = pBuffer[0] & 15;

if(headlen != 5)

{

ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_HEADLEN\_ERROR);

return 1;

}

unsigned short ttl = pBuffer[8];

if(ttl == 0)

{

ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_TTL\_ERROR);

return 1;

}

unsigned int packet\_destination = ntohl(((unsigned int\*)pBuffer)[4]);

if(getIpv4Address() != packet\_destination)

{

ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_DESTINATION\_ERROR);

return 1;

}

int checksum = ((unsigned short \*)pBuffer)[5];

if(checksum != \_checksum(pBuffer))

{

ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_CHECKSUM\_ERROR);

return 1;

}

ip\_SendtoUp(pBuffer, length);

return 0;

}

3.2.2 发送接口int stud\_ip\_Upsend(char\* pBuffer, unsigned short len, unsigned int srcAddr, unsigned int dstAddr ,byte protocol, byte ttl)

* 参数:

pBuffer:指向发送缓冲区的指针，指向 IPv4 上层协议数据头部 len:IPv4 上层协议数据长度

srcAddr:源 IPv4 地址

dstAddr:目的 IPv4 地址

protocol:IPv4 上层协议号

ttl:生存时间(Time To Live)

* 返回值

0: 成功发送 IP 分组

1: 发送 IP 分组失败

* 实现:

int stud\_ip\_Upsend(char \*pBuffer,unsigned short len,unsigned int srcAddr,

unsigned int dstAddr,byte protocol,byte ttl)

{

unsigned short totallen = len + 20;

char \*pSend = (char\*)malloc(sizeof(char)\*(totallen));

//version headlength

pSend[0] = 'E';

//total length

unsigned short nslen = htons(totallen);

memcpy(pSend + 2, &nslen, sizeof(unsigned short));

//time to live

pSend[8] = ttl;

//protocal

pSend[9] = protocol;

//source address

unsigned int source\_add = htonl(srcAddr);

memcpy(pSend + 12, &source\_add, sizeof(unsigned int));

//destination address

unsigned int dest\_add = htonl(dstAddr);

memcpy(pSend + 16, &dest\_add, sizeof(unsigned int));

//checksum

unsigned short checksum = \_checksum(pSend);

memcpy(pSend + 10, &checksum, sizeof(short));

//data

memcpy(pSend + 20, pBuffer, len);

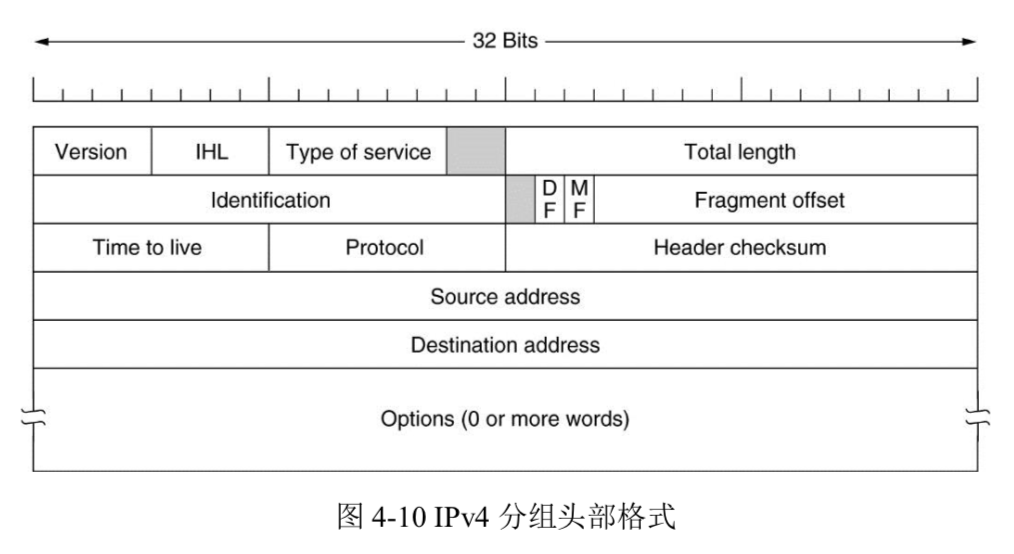
ip\_SendtoLower(pSend,totallen);

return 0;

}

3.3 IPV4数据包解析及检测原理

IPV4数据包格式如下：



首先是对于Version字段，我们通过unsigned short version = pBuffer[0] >> 4 该条语句从数据包的第一个8位的高4位，获取version的un short型数据，通过判定其是否等于4来确定是否是IPV4协议。否则我们抛出异常：ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_VERSION\_ERROR);

其次是对于IHL字段，我们通过un short headlen = pBuffer[0] & 15 从数据包的第一个8位的低4位来获取。这个字段用来确定数据的偏移量。最小值是5（二进制0101），相当于5\*4=20字节，在代码中我们判断它是否为5，在实际表示中这个值会被乘以4，否则我们抛出异常ip\_DiscardPkt(pBuffer,STUD\_IP\_TEST\_HEADLEN\_ERROR);

接下来是判定TTL字段，这个8位字段避免报文在互联网中永远存在（例如陷入路由环路）。存活时间以秒为单位，报文经过的每个路由器都将此字段减1，当此字段等于0时，报文不再向下一跳传送并被丢弃，最大值是255。常规地，一份ICMP报文被发回报文发送端说明其发送的报文已被丢弃。我们在代码中使用unsigned short ttl = pBuffer[8] 来获取这个字段，并且判定其值是否为0，如果为0，抛出异常：ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_TTL\_ERROR);

接下来是判定数据包地址是否为发送到本机的，我们使用unsigned int packet\_destination = ntohl(((unsigned int\*)pBuffer)[4])这条语句获取数据包的IP目的地址字段，使用getIpv4Address() 获取本机的IP地址，我们判定这两个值是否相等，如果不相等则丢弃这个数据包，抛出ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_DESTINATION\_ERROR);

最后我们还需要通过语句int checksum = ((unsigned short \*)pBuffer)[5] 来获取数据包里携带的checksum，然后利用checksum算法，将数据包现有的内容计算checksum并且和数据包中携带的进行对比，如果对比通过则说明数据包正确，反之则需要丢弃此数据包并且抛出ip\_DiscardPkt(pBuffer, STUD\_IP\_TEST\_CHECKSUM\_ERROR);

## 四、实验心得

IPv4是一种无连接的协议，操作在使用分组交换的链路层（如以太网）上。此协议会尽最大努力交付数据包，意即它不保证任何数据包均能送达目的地，也不保证所有数据包均按照正确的顺序无重复地到达。这些方面是由上层的传输协议（如传输控制协议）处理的。

该实验让我们亲自在netriver实验环境下完成了IPV4数据包收发的实验，虽然实验系统比较老旧，但是我们在实验的过程中，可以自己编辑数据包，可以观察发送接收端的全过程，这对于我们对IPV4协议的理解有了很好的提升。在实现接口的过程中，虽然存在着一些参数定义不太清楚的情况，但是大部分都是可以通过比对数据包来明晰的，『手动编辑数据包』这个功能让我觉得很赞。

最后则是，实验还是稍显简单，大家的可操作性并不是很强。