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

一、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。

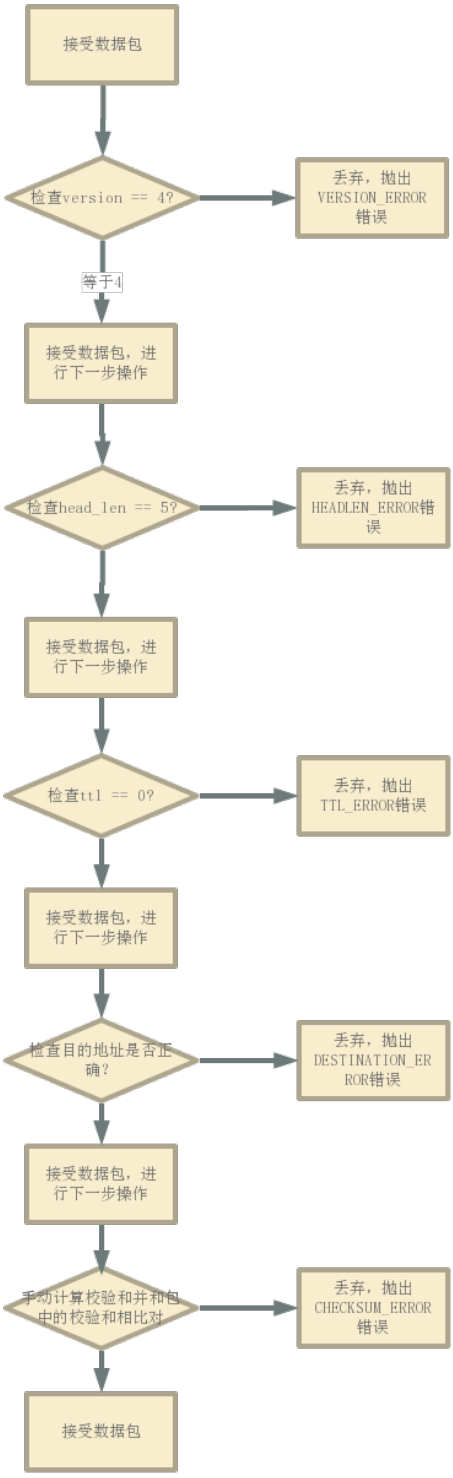
实验内容

a) 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。
b) 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。

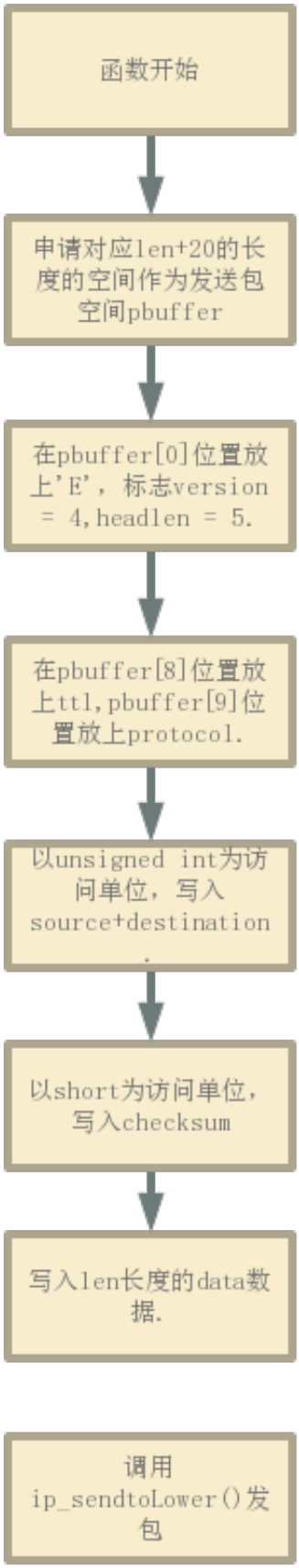
三、实验过程及结果

3.1 发送/接受函数流程图

接受函数


接受数据包
检查version == 4?
丢弃，抛出VERSION_ERROR错误
等于4
接受数据包，进行下一步操作
检查head_len == 5?
丢弃，抛出HEADLEN_ERROR错误
接受数据包，进行下一步操作
检查ttl == 0?
丢弃，抛出TTL_ERROR错误
接受数据包，进行下一步操作
检查目的地址是否正确?
丢弃，抛出DESTINATION_ERROR错误
接受数据包，进行下一步操作
手动计算校验和并和包中的校验和相比对
丢弃，抛出CHECKSUM_ERROR错误
接受数据包

发送函数


函数开始
申请对应len+20的长度的空间作为发送包空间pbuffer
在pbuffer[0]位置放上'E'，标志version = 4,headlen = 5.
在pbuffer[8]位置放上ttl,pbuffer[9]位置放上protocol.
以unsigned int为访问单位，写入source+destination.
以short为访问单位，写入checksum
写入len长度的data数据.
调用ip_sendtoLower()发包

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：

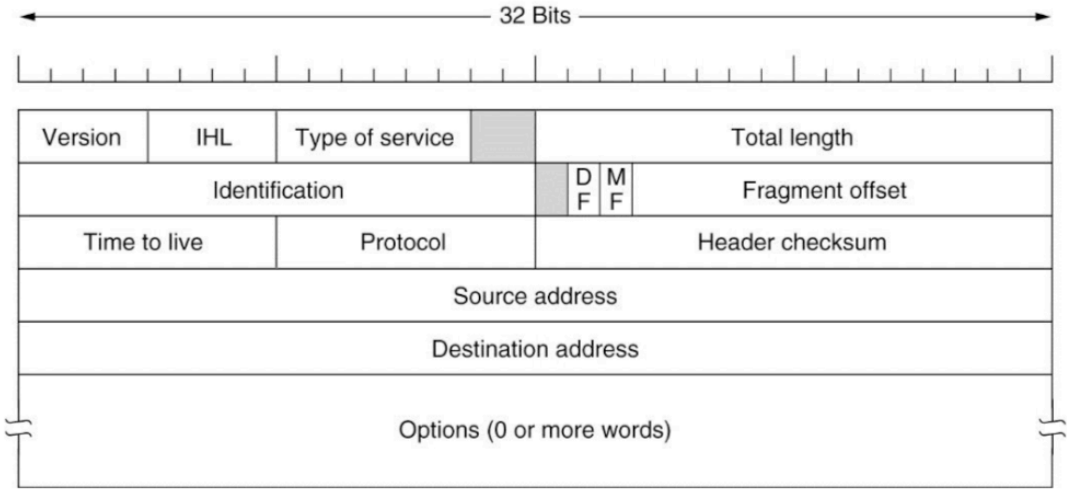


图 4-10 IPv4 分组头部格式

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。