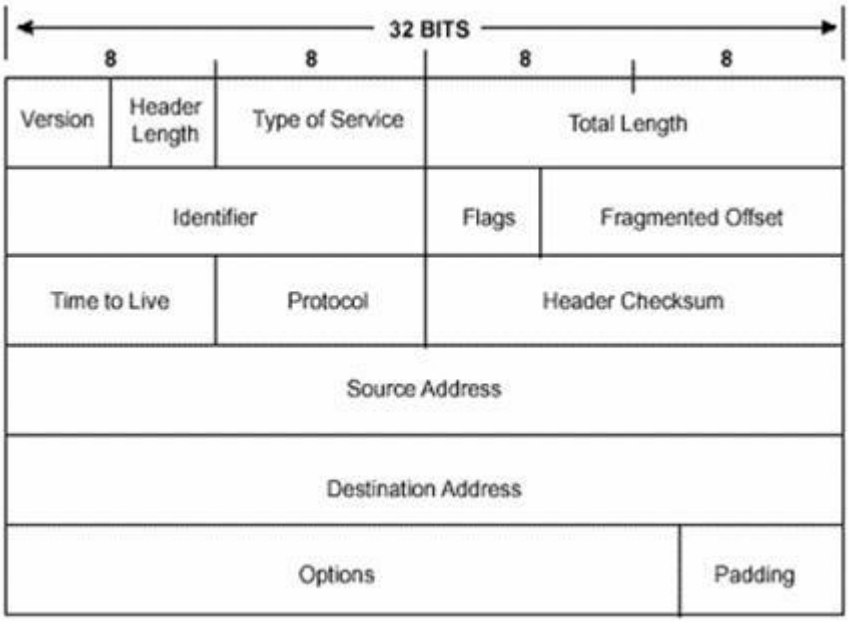
IPV4 收发实验报告

一、实验目的

通过本实验能够了解 IPV4 这个网络核心协议收发包的基本流程，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。

二、实验要求

IPv4分组头部格式如下图所示：



本实验要求学生根据NetRiver实验系统提供的上下层接口函数和IPv4协议分组收发的主要流程，独立设计实现一个简单的IPv4分组收发模块。实现的主要功能包括IPv4分组的基本接收处理和IPv4分组的封装发送，具体要求如下。

（1） 实现IPv4分组的基本接收处理功能

对于接收到的IPv4分组，首先应该检查IPv4分组的头部校验和以及其他字段的合法性，如果没有问题，则检查目的地址是否为本机地址。然后把正确分组中的数据提交给上层协议继续处理，或者丢弃错误的分组并说明错误类型。

（2） 实现IPv4分组的封装和发送

根据上层协议提供的参数，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） 不要求实现IPv4协议中的选项和分片处理功能。

三、实验接口

（1）需要学生实现的接口函数说明

需要学生实现的接口函数包括接收接口函数和发送接口函数。接收接口函数在接收IPv4分组时由系统调用，发送接口函数在发送数据时由系统调用。

1） 接收接口函数：

```
Int stud_ip_recv(char * pBuffer, unsigned short length)
```

参数说明：

pBuffer：指向接收缓冲区的指针，指向所接收到的IPv4分组。

Length：IPv4分组长度。

返回值：

1：IP分组接收失败。

0：成功接收IP分组并交给上层处理。

接收接口函数实现的主要功能是：接收到IPv4分组后，首先检查IPv4分组头部中各字段的合法性，并检查目的地址是否为本机地址。对于正确的分组，提交给上层协议继续处理，否则，丢弃错误的分组并说明错误类型。

2） 发送接口函数：

Int stud_ip_Upsend(char * pBuffer, unsigned short len, unsigned int srcAddr, unsigned int dstAddr, byte protocol, byte ttl)

参数说明：

pBuffer：指向发送缓冲区的指针，即指向IPv4上层协议数据（不含IPv4分组头部）。

len：待发送的上层协议数据长度。

srcAddr：待发送数据的源IPv4地址。

dstAddr：待发送数据的目的IPv4地址。

Protocol：IPv4上层协议号。

Ttl：生存时间。

返回值：

1：发送IPv4分组失败。

0：发送IPv4分组成功。

发送接口函数的主要功能是：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）系统提供的接口函数说明

本实验系统中提供了丢弃分组函数、发送分组函数、上层接收函数和获取本机IPv4地址函数。

1） 丢弃分组函数：

void ip_DiscardPkt(char * pBuffer, int type)

参数说明：

pBuffer：指向被丢弃分组的指针。

type：分组被丢弃原因，可取以下值：

STUD_IP_TEST_CHECKSUM_ERROR：IP头部校验和出错

STUD_IP_TEST_TTL_ERROR：TTL值出错

STUD_IP_TEST_VERSION_ERROR：IP版本号错

STUD_IP_TEST_HEADLEN_ERROR：头部长度错

STUD_IP_TEST_DESTINATION_ERROR：目的地址出错

2） 发送分组函数：

void ip_SendtoLower(char * pBuffer, int length)

参数：

pBuffer：指向待发送的IPv4分组的指针。

Length：待发送的IPv4分组长度。

3） 上层接收函数：

void ip_SendtoUp(char * pBuffer, int length)

参数：

pBuffer：指向要上交的上层协议分组的指针。

Length：上交分组长度。

4） 获取本机IPv4地址函数：

Unsigned int getIpv4Address()

四、实验实现

```
#include "sysInclude.h"    //这是Rivernet2000 系统软件提供的库函数
extern void ip_DiscardPkt(char* pBuffer,int type); //丢弃包
extern void ip_SendtoLower(char*pBuffer,int length); //交给下层
extern void ip_SendtoUp(char *pBuffer,int length); //交给上层
extern unsigned int getIpv4Address();
// implemented by students
int stud_ip_recv(char *pBuffer,unsigned short length)    //自己实现的本机Ipv4数据包接收函数
{
//获取ip头信息
int version = pBuffer[0] >> 4;
int headlength = pBuffer[0] & 0xf;
int timetolive = (int)pBuffer[8];
int headerChecksum = ntohs(*(short unsigned int*)(pBuffer+10));
int destinationAddress = ntohl(*(unsigned int*)(pBuffer+16));
if (version != 4) //检查version
{
ip_DiscardPkt(pBuffer,STUD_IP_TEST_VERSION_ERROR);
return 1;
}
if (headlength < 5) //检查IHL
{
ip_DiscardPkt(pBuffer,STUD_IP_TEST_HEADLEN_ERROR);
return 1;
}
if (timetolive == 0) //检查TTL
{
ip_DiscardPkt(pBuffer,STUD_IP_TEST_TTL_ERROR);
return 1;
}
//检查目的地址和本机地址是否相同
if (destinationAddress != getIpv4Address() && destinationAddress != 0xffffff)
{
ip_DiscardPkt(pBuffer,STUD_IP_TEST_DESTINATION_ERROR);
return 1;
}
```

```
int sum = 0;
unsigned short int localCheckSum = 0;
unsigned short int field;
int offset;
for(int i = 1;i <= headlength*2; i++) //计算校验和
{
offset = (i-1)*2;
if(i != 6)
{
field = (pBuffer[offset])<<8;
field += pBuffer[offset+1];
sum += field;
sum %= 65535;
}
}
localCheckSum = 0xffff - (unsigned short int)sum;
if(localCheckSum != headerChecksum) //检验校验和
{
ip_DiscardPkt(pBuffer,STUD_IP_TEST_CHECKSUM_ERROR);
return 1;
}

ip_SendtoUp(pBuffer,length);
return 0;
}
int stud_ip_Upsend(char *pBuffer,unsigned short len,unsigned int srcAddr,
unsigned int dstAddr,byte protocol,byte ttl)
{
char *sendBuffer = new char(len + 20);
memset(sendBuffer, 0, len+20);
//填入各种IP头信息
sendBuffer[0] = 0x45;
unsigned short int totallen = htons(len + 20);
memcpy(sendBuffer + 2, &totallen, sizeof(unsigned short int));
sendBuffer[8] = ttl;
sendBuffer[9] = protocol;
unsigned int src = htonl(srcAddr);
unsigned int dis = htonl(dstAddr);
memcpy(sendBuffer + 12, &src, sizeof(unsigned int));
memcpy(sendBuffer + 16, &dis, sizeof(unsigned int));

int sum = 0;
unsigned short int localCheckSum = 0;
for(int i = 0;i < 10;i ++)
```

```
{
sum = sum + (sendBuffer[i*2]<<8) + (sendBuffer[i*2+1]);
sum %= 65535;
}
localCheckSum = htons(0xffff - (unsigned short int)sum);

memcpy(sendBuffer + 10, &localCheckSum, sizeof(unsigned short int));
memcpy(sendBuffer + 20, pBuffer, len);
//发送
ip_SendtoLower(sendBuffer,len+20);
return 0;
}
```

五、实验心得体会

通过本次实验，我更加深入地了解到了网络原理课程中所学到的知识，加深了对 IP 协议的理解，也对自己协议编写的能力有了显著地提高。