**《计算机网络》实验报告3**

IPv4 分组收发实验

**童超宇**

院 （系）：计算机科学与技术学院 专 业：计算机科学与技术

学　　号 ：1152130106 指导教师：刘亚维

**2018年5月**

# 目录

[1 实验目的 3](#_Toc307274513)

[2 实验内容 3](#_Toc843702061)

[3 程序流程图 3](#_Toc112365768)

[3.1 发送函数 3](#_Toc1115211661)

[3.2 接收函数 4](#_Toc1315863866)

[4 错误检测 4](#_Toc1240030037)

[4.1 版本号 4](#_Toc176138292)

[4.2 头部长度 4](#_Toc1755458720)

[4.3 生存时间 4](#_Toc289539420)

[4.4 目的地址 4](#_Toc220365483)

[4.5 头部校验和字段 4](#_Toc177507389)

[5 错误示例 5](#_Toc1246552443)

[6 源代码（有注释） 5](#_Toc296442823)

# 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。

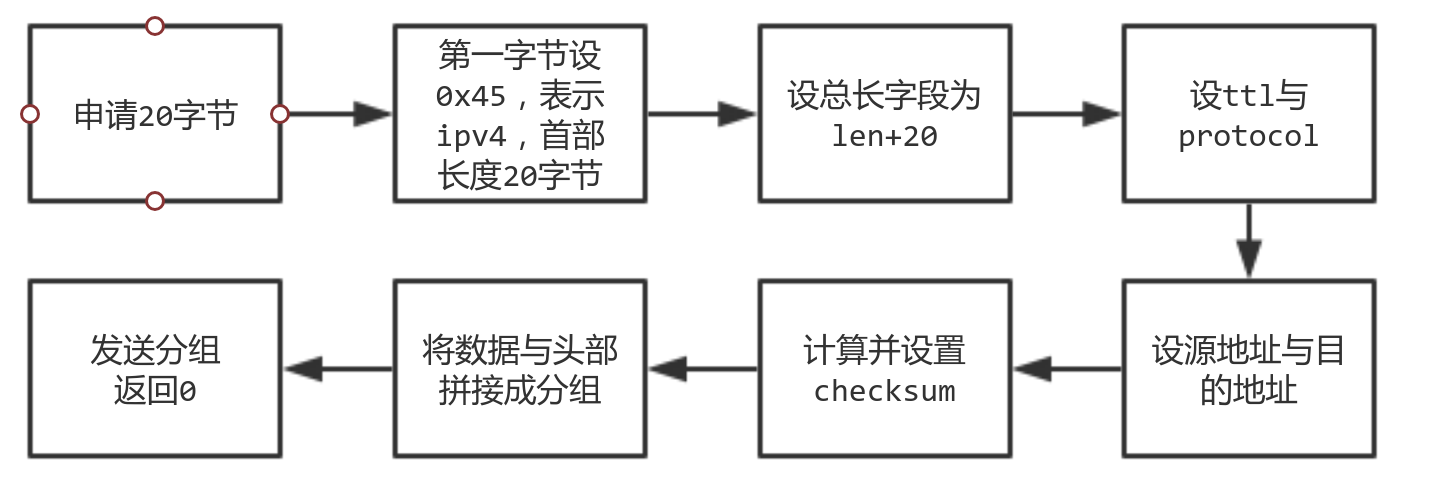
# 实验内容

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。

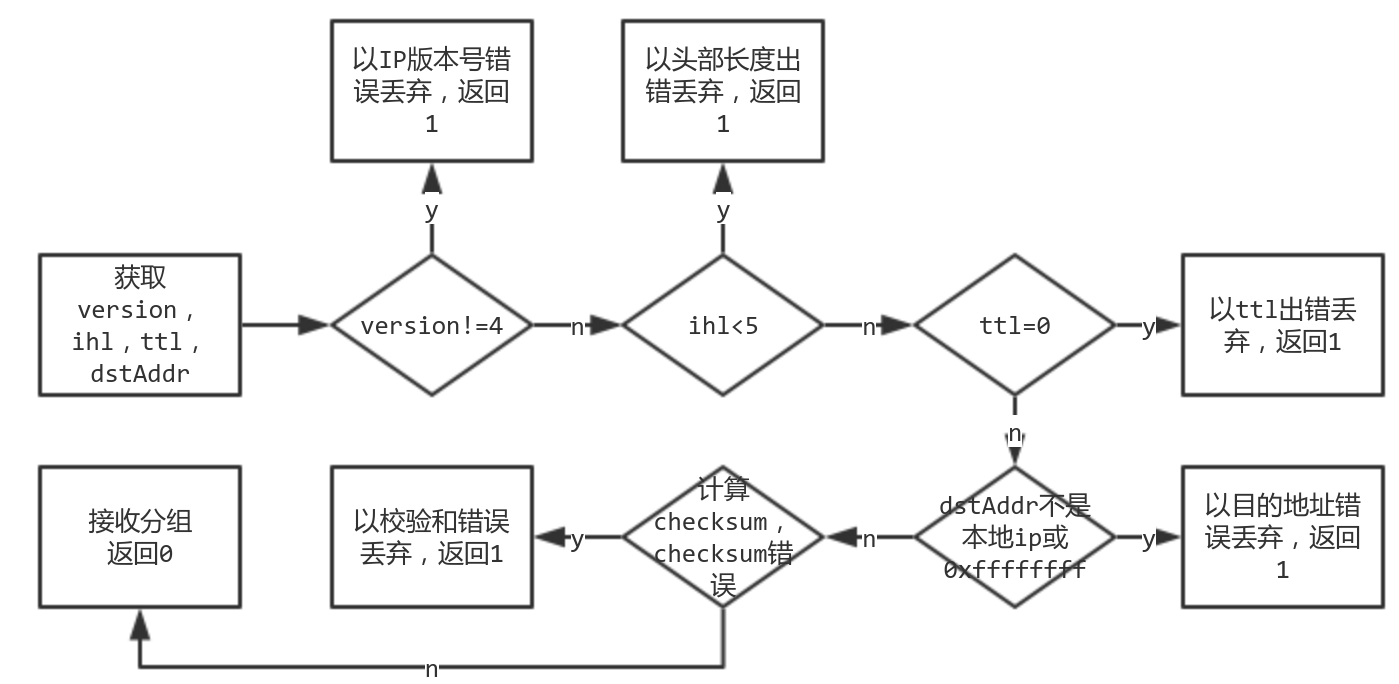
实现IPv4分组的封装发送。根据上层协议所提供的参数，封装IPv4分组，调用系统提供的发送接口函数将分组发送出去。

# 程序流程图

## 发送函数



## 接收函数



**注：程序无类或结构体，对于传入的pBuffer直接取对应字段的数据**

# 错误检测

## 版本号

版本号必须是4，表示ipv4，否则以ip版本号出错丢弃分组，返回1

## 头部长度

头部长度字段必须>=5，表示头部长度>=20字节，否则以头部长度出错丢弃分组，返回1

## 生存时间

生存时间必须>0，表示还能转发，否则以生存时间出错丢弃分组，返回1

## 目的地址

目的地址应为本机地址或0xffffffff，否则以目的地址出错丢弃分组，返回1

## 头部校验和字段

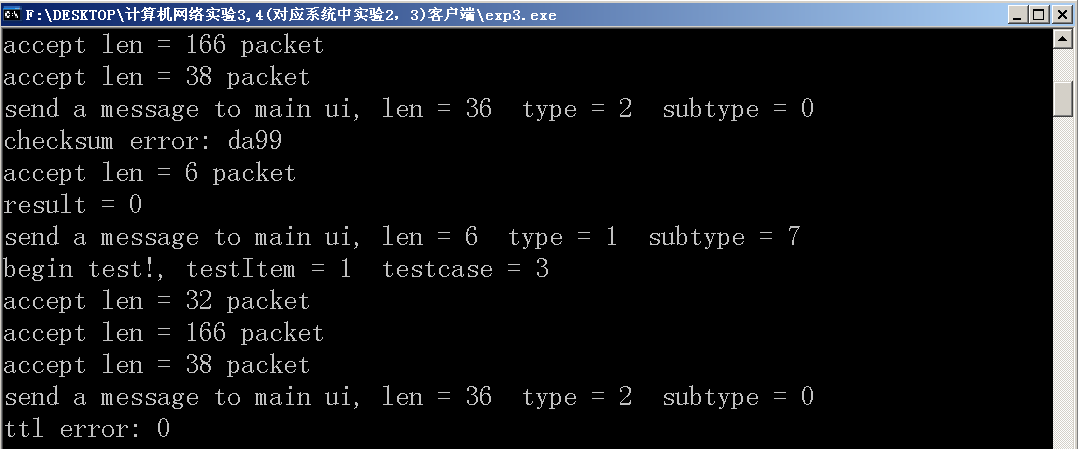
用两个循环计算校验和，第一个循环按双字节将头部字段累加，第二个循环反复将结果高16位加到低16位上直到高16位全0

若结果不是0xffff，以头部校验和出错丢弃分组，返回1

# 错误示例

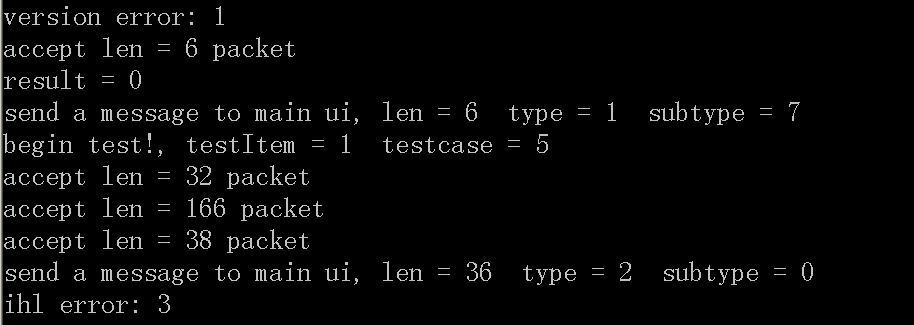
checksum error： da99

ttl error： 0



version error：1

ihl error：3



目的地址错误



# 源代码（有注释）

#include "sysInclude.h"

typedef unsigned int uint;

typedef unsigned short ushort;

extern void ip\_DiscardPkt(char \*pBuffer, int type);

extern void ip\_SendtoLower(char \*pBuffer, int length);

extern void ip\_SendtoUp(char \*pBuffer, int length);

extern uint getIpv4Address();

int stud\_ip\_recv(char \*pBuffer, ushort length) {

// resolve header information.

int version = (int)(pBuffer[0]) >> 4;

int ihl = (int)(pBuffer[0]) & 0xf;

int ttl = (int)(pBuffer[8]);

int checkSum = ntohs(\*(ushort \*)(pBuffer + 10));

uint dstAddr = ntohl(\*(uint \*)(pBuffer + 16));

if (version != 4) {

printf("version error: %d\n",version);

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

return 1;

}

if (ihl < 5) {

printf("ihl error: %d\n",ihl);

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

return 1;

}

if (ttl == 0) {

printf("ttl error: %d\n",ttl);

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

return 1;

}

if (dstAddr != getIpv4Address() && dstAddr != 0xffffffff) {

printf("dstAddr error: %d\n",dstAddr);

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

return 1;

}

int sum = 0;

// 16 bit add

for (int i = 0; i < 10; ++i) sum += (int)(\*(ushort \*)(pBuffer + i \* 2));

// add high 16 to low 16, until high 16=0x0

while (sum > 0xffff) sum = (sum & 0xffff) + (sum >> 16);

if (sum != 0xffff) {

printf("checksum error: %x\n",sum);

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

return 1;

}

ip\_SendtoUp(pBuffer, length);

return 0;

}

int stud\_ip\_Upsend(char \*pBuffer, ushort len, uint srcAddr, uint dstAddr,

byte protocol, byte ttl) {

char header[20];

memset(header, 0, 20);

// version and ihl

header[0] = 0x45;

// total length

\*(ushort \*)(header + 2) = htons((ushort)(20 + len));

header[8] = ttl;

header[9] = protocol;

// copy as uint(32 bit)

\*(uint \*)(header + 12) = htonl(srcAddr);

\*(uint \*)(header + 16) = htonl(dstAddr);

int sum = 0;

for (int i = 0; i < 10; ++i) sum += (int)(\*(ushort \*)(header + i \* 2));

while (sum > 0xffff) sum = (sum & 0xffff) + (sum >> 16);

sum = ~((ushort)sum);

\*(ushort \*)(header + 10) = (ushort)sum;

char \*msg = new char[len + 20];

memcpy(msg, header, 20);

memcpy(msg + 20, pBuffer, len);

ip\_SendtoLower(msg, len + 20);

return 0;

}