**IPV4协议收发实验上机报告**

1. **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。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。学习 IPv4 协议基本的分组接收和发送流程。

1. **实验要求**

根据计算机网络实验系统所提供的上下层接口函数和协议中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。要求实现的主要功能包括：

1） IPv4 分组的基本接收处理；

2） IPv4 分组的封装发送；

注： 不要求实现 IPv4 协议中的选项和分片处理功能

1. **实验内容**
2. 实现 IPv4 分组的基本接收处理功能

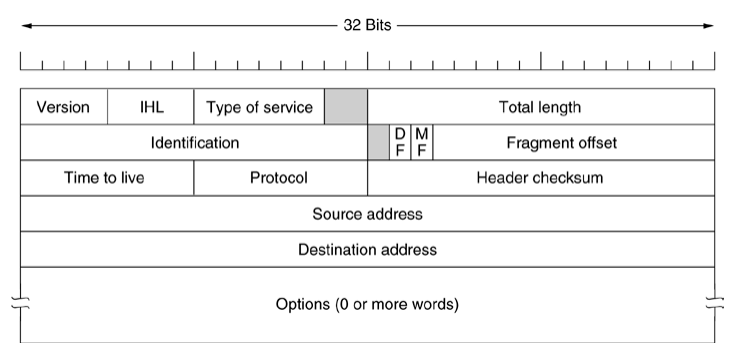
对于接收到的 IPv4 分组，检查目的地址是否为本地地址，并检查IPv4 分组头部中其它字段的合法性。提交正确的分组给上层协议继续处理，丢弃错误的分组并说明错误类型。

1. 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ᨀ供的发送接口函数将分组发送出去。

1. **实验原理**

IPv4分组头部格式如图，所以只需从头部各字节提取相应信息，对报文做出检验即可。



1. **实验代码**

//生成校验和函数

unsigned short \_checksum(char \*pBuffer)

{

int sum = 0;

for(int i = 0; i < 10; ++i)

{

if(i != 5)

{

sum += ((unsigned short\*)pBuffer)[i];

}

}

while(sum > 0xffff)

{

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

}

return (unsigned short)(0xffff - sum);

}

//接受下层报文

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;

}

//发送上层报文

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;

}