哈爾濱工業大學
Harbin Institute of Technology

计算机网络
课程实验报告

实验名称	IPv4 分组收发、转发实验				
姓名	王丙昊		院系	计算机科学与技术学院	
班级	1603108		学号	1160300302	
任课教师	聂兰顺		指导教师	聂兰顺	
实验地点	格物 207		实验时间	2018.11.14	
实验课表现	出勤、表现得分(10)		实验报告得分(40)	实验总分	
	操作结果得分(50)				
教师评语					

计算机科学与技术学院 SINCE 1956...
School of Computer Science and Technology

实验目的：

通过设计实现主机协议栈中的 IPv4 协议，来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、发送流程。并初步接触互联网协议栈的结构和计算机网络实验系统，为后面进行更深入复杂的实验奠定良好的基础。

在原有的 IPv4 收发实验的基础上，增加 IPv4 分组的转发功能。了解路由器是如何为分组选择路由，并逐跳地将分组发送到目的主机。实验中也会接触路由表这一重要的数据结构，认识路由器是如何根据路由表对分组进行转发的。

实验内容：

1） 实现IPv4分组的基本**接收处理**功能，能够检查出接收到的IP分组是否存在如下错误：校验和错、TTL错、版本号错、头部长度错和错误目的地址

2） 实现IP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。

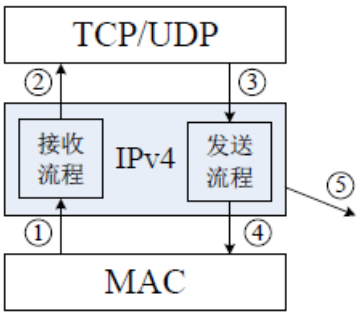
3） 实现IPv4分组的**转发**：对每一个到达本机的IPv4分组，根据其目的IPv4地址决定分组的处理行为，设计路由表的结构，根据路由表查询结果，向上层协议交付、丢弃和转发。

实验过程：

一、 IPv4分组收发

1. 程序流程

程序流程示意图：



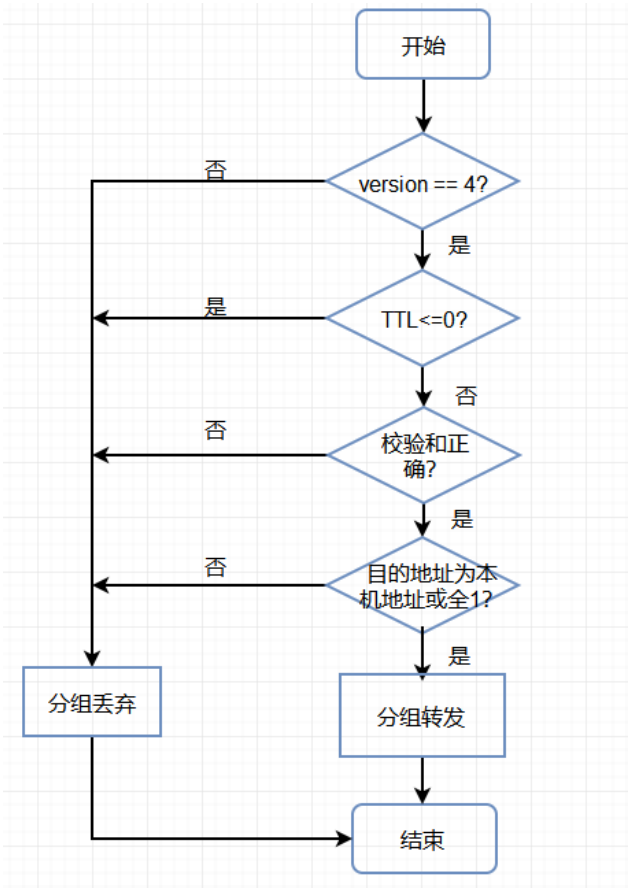
1） 接收流程

接收时调用函数stud_ip_recv()：

① 检查接收到的IPv4分组头部地字段，包括版本号(Version)、头部长度(header_len)、生存时间(TTL)以及首部校验和(header_checksum)。对于出错的分组调用ip_DiscardPkt()丢弃，并说明错误类型。

② 检查IPv4分组是否应该由本机接收。如果分组的目的地址是本机地址**或广播地址，**则说明此分组是发送给本机的；否则也调用ip_DiscardPkt()丢弃，并说明错误类型。

③ 如果IPv4分组应该由本机接收，则提取得到上层协议类型，调用ip_SendtoUp()接口函数，交给系统进行后续接收处理。



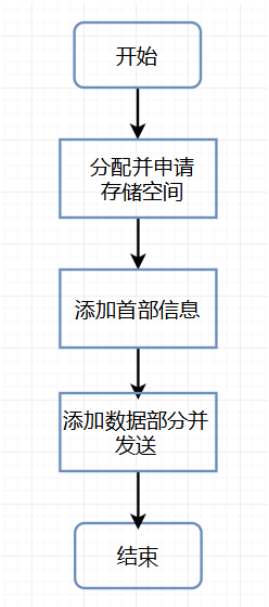
2） 发送流程

发送时调用函数stud_ip_Upsend()：

① 根据所传参数，来确定分配的存储空间大小并申请分组的存储空间。

② 按照IPv4协议标准填写IPv4分组头部各字段，对于没有给出的参数可以置0。(注意部分字段**需要转换成网络字节序)**

③ 完成IPv4分组的封装后，调用ip_SendtoLower()接口函数完成后续的发送处理工作，最终将分组发送到网络中。

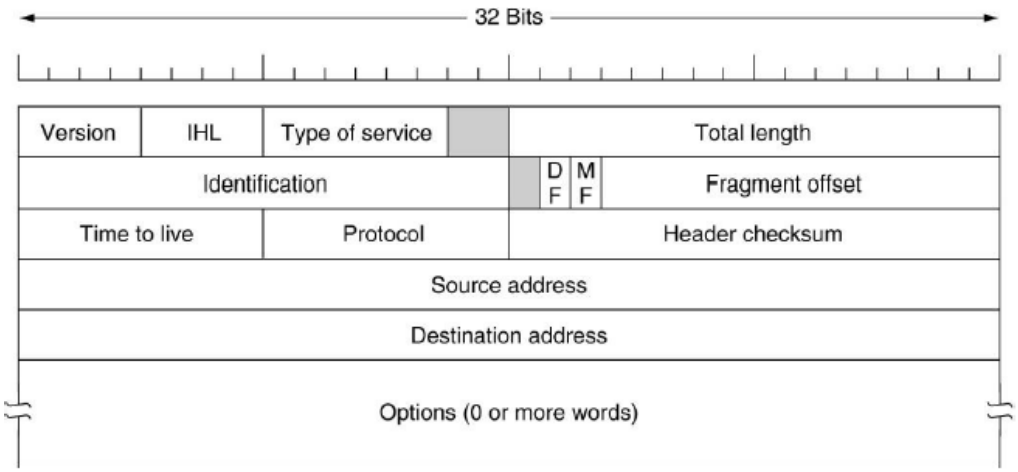


发送过程实现如下：

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

    byte *pkt_send = new byte[len + 20];
    memset(pkt_send, 0, len+20);

    pkt_send[0] = 0x45;
    pkt_send[1] = 0x80;
    pkt_send[8] = ttl;
    pkt_send[9] = protocol;

    *(short *)(pkt_send + 2) = htons(20 + len);
    *(int *)(pkt_send + 12) = ntohl(srcAddr);
    *(int *)(pkt_send + 16) = ntohl(dstAddr);
    *(short *)(pkt_send + 10) = htons(compute_checksum(pkt_send, 5));

    memcpy(pkt_send + 20, pBuffer, len);

    ip_SendtoLower(pkt_send, len + 20);
    return 0;
}
```

2. 数据结构说明

根据IPv4分组头部格式来设计数据接口：



数据结构定义如下：

```
struct Ipv4{
    char version_headlen;       // 版本号与首部长度
    char TOS;                   // 服务类型
    short total_len;            // 报文总长度
    short identification;       // 标识
    short flag_offset;          // 标志位以及片偏移
    char TTL;                   // 生存时间
    char protocol;              // 上层协议类型
    short header_checksum;      // 首部校验和
    unsigned int source_addr;   // 源IP地址
    unsigned int dest_addr;     // 目的IP地址
```

3. 错误检测原理

1） 版本号错误

从IPv4分组报文中提取出版本号字段，如果该字段不为4，说明版本号错误。

```
int version = 0xf0 & ipv4->version_headlen;

if(version != 0x40)  {
ip_DiscardPkt(pBuffer, STUD_IP_TEST_VERSION_ERROR);
    return 1;
}
```

2） 头部长度错误

首部长度是以**4个字节**为单位的，一般由于选项字段为空，首部长度为20个字节，所以该字段应该为**5**

```
int header_len = 0xf & ipv4->version_headlen;

if(header_len < 0x05) {
ip_DiscardPkt(pBuffer, STUD_IP_TEST_HEADLEN_ERROR);
    return 1;
}
```

3） 生存时间错误

按照规定，如果TTL的值为0，则代表生命周期结束，丢弃该分组。为了防止伪造TTL为负值IPv4分组，也**需要判断TTL是否为负值**并丢弃。

```
if(ttl <= 0) {
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_TTL_ERROR);
    return 1;
}
```

4） 首部校验和错误

当主机收到IP分组时，重新计算首部校验和，将其与分组中的首部校验和进行比较，如果**不相同，说明传输过程中一定发生了错误**。校验和计算的规则：16进制反码求和，也就是将所有的**除了校验和字段之外的字节**加起来，然后用0xffff减去

5） 目的地址错误

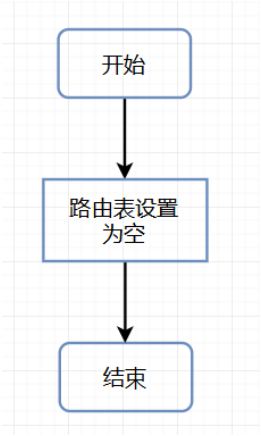
可以通过getIpv4Address()接口函数来获得本机的IP地址，通过与IPv4分组中的dest_addr字段进行比较，如果**不相同并且dest_addr不是广播地址**，说明本机不能接收该IP分组，需要丢弃。

```
if(dest_addr != local_addr && dest_addr != 0xffffffff) {
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_DESTINATION_ERROR);
    return 1;
}
```

二、 IPv4分组转发

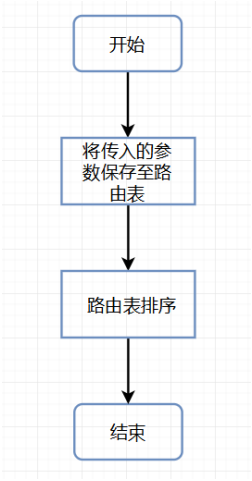
1. 程序流程

1) void stud_Route_Init()



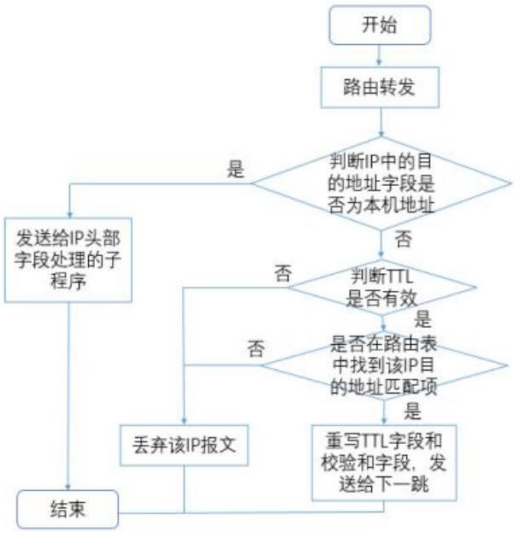

```
void stud_Route_Init()
{
    route_table.clear();
}
```

2) void stud_route_add(stud_route_msg *proute)



根据系统已经规定的参数进行传入，将其相关的信息保存到路由表中，包括目标地址、掩码长度以及下一条地址。

3) int stud_fwd_deal(char *pBuffer, int length)



首先判断目的地址**是否为本机地址和TTL是否大于0**，如果是本机地址，则调用fwd_LocalRcv()进行其头部的进度判断，如果TTL小于等于0，则丢弃该分组；如果不是本机地址，**查找路由表**，如果路由表中有匹配的表项，TTL减一并重新计算校验和，将数据分组转发到下一条地址；如果没有路由表中没有匹配，则丢弃该分组。

2. 路由表数据结构说明

数据结构定义如下，路由表中的每一项包括目的地址、掩码长度和下一条地址。

```
typedef struct route_node
{
    unsigned int dest;      // 目标地址
    unsigned int masklen;   // 掩码长度
    unsigned int nexthop;   // 下一条地址
}Node;

vector<Node> route_table;
```

3. 大量分组时提高转发效率

1） 路由聚合

通过路由聚合，能够选择一个地址通告众多网络，旨在缩小路由器路由表的规模，以节省内存，并缩短路由表查询进行转发所需要的时间。

2） 数据结构的改进

可以将路由表的线性存储结构改为树形结构，来提高匹配效率。

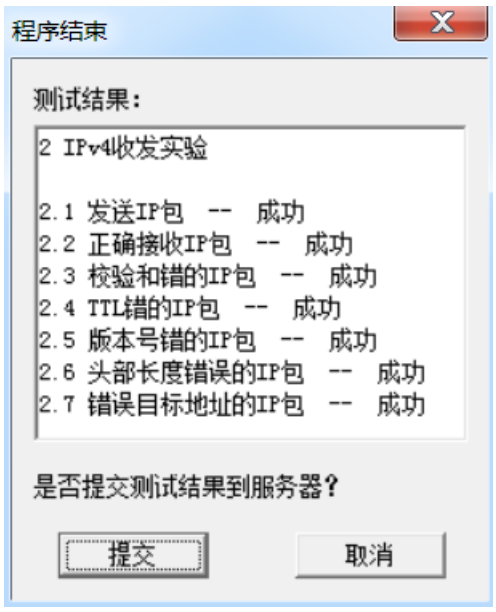
4. 最长前缀匹配的实现

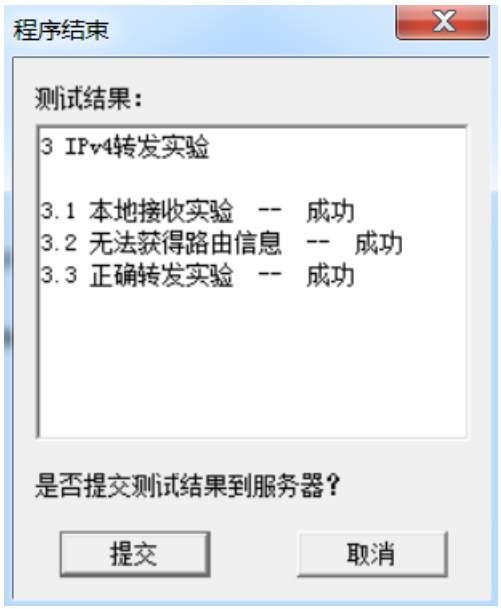
在向路由表中添加项时，实现最长前缀匹配。如果一个目的地址可以与多个表项进行匹配，则让其与子网掩码最长的一个匹配，即在添加路由表时，每当一个表项被添加时，对该路由表进行排序，使IP地址较大且掩码长度大的位于前面。

在**void stud_route_add(stud_route_msg *proute)**中实现

```
void stud_route_add(stud_route_msg *proute)
{
    Node *new_node = new Node();
    new_node->dest = ntohl(proute->dest);
    new_node->masklen = ntohl(proute->masklen);
    new_node->nexthop = ntohl(proute->nexthop);
    route_table.push_back(*new_node);
    sort(route_table.begin(), route_table.end(), cmp);
    return;
}
```

实验结果：





心得体会：

1. 深入了解了网络协议的基本原理；
2. 学习了IPv4协议分组的收发、转发基本流程；
3. 初步接触了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现奠定了基础。