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	HTTP 代理服务器的设计与实现				
姓名	王子奕		院系	计算学部	
班级	1937101		学号	1190200121	
任课教师	李全龙		指导教师	李全龙	
实验地点	G207		实验时间	2021.11.11	
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：

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。

IPv4 分组收发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。

IPv4 分组转发实验需要将实验模块的角色定位从通信两端的主机转移到作为中间节点的路由器上，在 IPv4 分组收发处理的基础上，实现分组的路由转发功能。

网络层协议最为关注的是如何将 IPv4 分组从源主机通过网络送达目的主机，这个任务就是由路由器中的 IPv4 协议模块所承担。路由器根据自身所获得的路由信息,将收到的 IPv4 分组转发给正确的下一跳路由器。如此逐跳地对分组进行转发,直至该分组抵达目的主机。IPv4 分组转发是路由器最为重要的功能。

IPv4 分组转发实验设计模拟实现路由器中的 IPv4 协议，可以在原有 IPv4 分组收发实验的基础上，增加 IPv4 分组的转发功能。对网络的观察视角由主机转移到路由器中,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。本实验中也会初步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：

在 IPv4 分组收发实验中，我们需要：

1) 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。

2) 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。

在 IPv4 分组转发实验中，我们需要：

1) 设计路由表数据结构。

设计路由表所采用的数据结构。要求能够根据目的 IPv4 地址来确定分组处理行为（转发情况下需获得下一跳的 IPv4 地址）。路由表的数据结构和查找算法会极大的影响路由器的转发性能，有兴趣的同学可以深入思考和探索。

2) IPv4 分组的接收和发送。

对前面实验（IP 实验）中所完成的代码进行修改，在路由器协议栈的 IPv4 模块中能够正确完成分组的接收和发送处理。具体要求不做改变，参见“IP 实验”。

3) IPv4 分组的转发。

对于需要转发的分组进行处理，获得下一跳的 IP 地址，然后调用发送接口函数做进一步处理。

实验过程：

一、IPv4分组收发实验

（1） 获取头部信息并检验

```
int stud_ip_recv(char *pBuffer, unsigned short length) {
  us version = pBuffer[0] >> 4;      // 版本号
  us head_length = pBuffer[0] & 0xf; // 头部长度
  us ttl = pBuffer[8];               // 生存时间
  us checksum = ntohs(*(unsigned short *)(pBuffer + 10)); // 校验和
  ui dest = ntohl(*(unsigned int *)(pBuffer + 16)); // 目的地址
  if(check_error(pBuffer,version,ttl,head_length,dest))
    return 1;

  ip_SendtoUp(pBuffer, length);
  return 0;
}
```

图表 1 按字节读取对应信息

```
bool check_error(char *pBuffer,us version,us ttl,us head_length,ui dest)
{
    if (version != 4) {
    // 只支持IPV4
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_VERSION_ERROR);
    return 1;
  } else if (ttl <= 0) {
    // TTL 值出错
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_TTL_ERROR);
    return 1;
  } else if (head_length < 5) {
    //头部长度错
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_HEADLEN_ERROR);
    return 1;
  } else if (dest != getIpv4Address()) {
    //目的地址错
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_DESTINATION_ERROR);
    return 1;
  }


  if (getipSum(pBuffer,head_length) != 0xffff) {
    // IP 校验和出错
    ip_DiscardPkt(pBuffer, STUD_IP_TEST_CHECKSUM_ERROR);
    return 1;
  }
  return 0;
}
```

图表 2 检查信息是否正常

具体实现：可以利用位运算来获取版本等头部信息。版本号和头部长度的信息在第一个字节。生存时间字段在 IPv4 数据包的头部第八个字节。源地址和目的地址在 IPv4 数据包的第十二和第十六个字节开始。此外需要检验版本号是否为 4，头部长度是否大于 5（因为最少的 IPv4 数据包的头部信息为 20 字节），生存时间是否为正（如果 TTL 非正则说明其已经过期），目的地址是否为本机地址。

（2） 头部校验和字段检验

```
us getipSum(char *pBuffer, us head_length)
{
  ul sum = 0;
  for (int i = 0; i < head_length * 2; ++i) {
    sum += (uc)pBuffer[i<<1] << 8;
    sum += (uc)pBuffer[(i<<1)|1];
  }
  return (sum & 0xffff)+(sum >> 16);
}
```

只需将整体相加，由于头部校验和为其他位之和取反，所以只需判断 两部分和是否为全一。若非全一则说明头部校验和错误。

二、IPv4分组转发实验

（1） 初始化路由操作

（2） 向路由表添加信息

将网络字节转化为本地字节再对应赋值即可。

```
void stud_route_add(stud_route_msg *proute) {
  stud_route_msg temp;
  temp.dest = ntohl(proute->dest);
  temp.masklen = ntohl(proute->masklen);
  temp.nexthop = ntohl(proute->nexthop);
  routes.push_back(temp);
}
```

（3） 处理转发

利用stud_fwd_deal函数查找路由表对应地址并转发。分为以下步骤

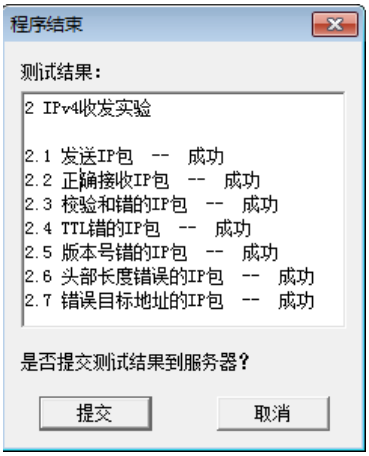
1. 判断TTL是否超时

2. 目的地址是否为本机地址，若是则调用接收函数，不是则继续判断
3. 路由表中是否还有表象，若是则继续匹配直至最长长度然后记录
4. 若当前路由表象与目的地址匹配，则TTL-1将checksum重新计算并转发。若不匹配则返回丢包相应参数结束

实验结果：

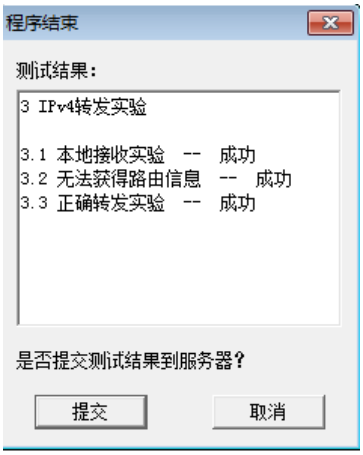
1. IPv4 收发实验的测试

在 Windows 7 客户端下测试的截图为：



2. IPv4 转发实验的测试

在 Windows 7 客户端下测试的截图为：



问题讨论：

如何提高路由表的查找效率？

当我们使用一些容器式的数据结构存储路由信息时，查找的复杂度为O(n)，当路由表中信息数量增多时效率变低。如果我们利用字典树存储路由表，为了实现最长前缀匹配，从最长的掩码开始匹配，每一个掩码都有一个哈希表，目的IP地址哈希到这些哈希表的特定的桶中，然后遍历其冲突链表得到最终结果。这样就能使复杂度降为O(1)。

心得体会：

通过本次实验我深刻的理解了数据报在互联网中的转发路由过程，辨析了IPV4协议的构成，为后面进行更为深入复杂的实验奠定良好的基础。