

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验3：IPv4 分组收发 和 分组转发 实验 | | | | | |
| 姓名 | 陈翔 | | 院系 | 计算机科学与技术学院 | | |
| 班级 | 1603109 | | 学号 | 1161800218 | | |
| 任课教师 | 李全龙 | | 指导教师 | 李全龙 | | |
| 实验地点 | 格物213 | | 实验时间 | 2018.11.10 | | |
| 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 | | | | | | |
|  | | | | | | |

****

|  |
| --- |
| 实验目的： |
| **IPv4分组收发实验：**  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。  **IPv4分组转发实验**  通过前面的实验，我们已经深入了解了 IPv4 协议的分组接收和发送 处理流程。本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。  本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： |
| **IPv4分组收发实验：**  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。  **IPv4分组转发实验**  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： |
| **IPv4分组收发实验：**  1)要求给出发送和接收函数的实现程序流程图；  接收流程  ① 检查接收到的 IPv4 分组头部的字段，包括版本号（Version）、头 部长度（IP Head length）、生存时间（Time to live）以及头校验 和（Header checksum）字段。对于出错的分组调用 ip\_DiscardPkt( ) 丢弃，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。    发送流程  ① 根据所传参数（如数据大小），来确定分配的存储空间的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，标识符 （Identification）字段可以使用一个随机数来填写。（注意：部分 字段内容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    2)给出自己所新建的数据结构的说明（如果有）；  按照 IPv4 首页的顺序构造结构体。  结构体的构造符合 IPv4 首部   |  | | --- | | struct Ipv4  {  char version\_ihl; // 版本号  char type\_of\_service; .// 协议类型  short total\_length; // 总长度  short identification; // 标志符  short fragment\_offset; // 偏移量  char time\_to\_live; // TTL  char protocol; // 协议  short header\_checksum; // 首部校验和  unsigned int source\_address; // 源地址  unsigned int destination\_address; // 目标地址  } |   3)要求给出版本号（Version）、头部长度（IP Head length）、生存时间（Time to live）以及头校验和（Header checksum）字段的错误  检测原理，并根据实验具体情况给出错误的具体数据，例如如果为头部长度错，请给出收到的错误的 IP 分组头部长度字段值为多少。  版本号校验  版本号在第一个字节的前4位，version\_ihl 是结构体的第一个字节，前4位代表版本号，右移4位，和 0xF 取和运算，如果结果依旧是4，则代表版本号正确   |  | | --- | | int version = 0xf & ((ipv4->version\_ihl)>> 4);  if(version != 4) {  ip\_DiscardPkt(pBuffer,STUD\_IP\_TEST\_VERSION\_ERROR);  return 1;  } |   错误的版本号，例如1    头部长度校验  头部长度在第一个字节的后4位里面，则 version\_ihl和 0xF 取和，如果结果为5，则代表头部长度没有问题   |  | | --- | | int ihl = 0xf & ipv4->version\_ihl;  if(ihl < 5) {  ip\_DiscardPkt(pBuffer,STUD\_IP\_TEST\_HEADLEN\_ERROR);  return 1;  } |   错误的头部长度    TTL 校验  TTL 存在于第9个字节里面，存在于 time\_to\_live 里面，按照规定，如果 ttl 的值是0，则代表生命周期结束，要抛弃这个包   |  | | --- | | int ttl = (int)ipv4->time\_to\_live;  if(ttl == 0) {  ip\_DiscardPkt(pBuffer,STUD\_IP\_TEST\_TTL\_ERROR);  return 1;  } |   错误的TTL，例如0，即“00000000”    目标地址校验  目标地址存在于首部的第 17 -20 个字节中，和本地 ip 地址做比较，如果不等于本地地址，并且也不等于0xffffffff，则表示目标地址出错。   |  | | --- | | int destination\_address = ntohl(ipv4->destination\_address);  if(destination\_address != getIpv4Address() && destination\_address != 0xffffffff) {  ip\_DiscardPkt(pBuffer,STUD\_IP\_TEST\_DESTINATION\_ERROR);  return 1;  } |   错误的目标地址，例如“192.166.77.9”    校验和  校验和存在于11-12 个字节，校验和检测的规则如下： 16 进制反码求和，也就是说是将所有的字节加起来（校验和部分忽略，即为0），然后用 ffff 减去   |  | | --- | | int header\_checksum = ntohs(ipv4->header\_checksum);  int sum = 0;  for(int i = 0; i < ihl\*2; i++) {  if(i!=5)  {  sum += (int)((unsigned char)pBuffer[i\*2] << 8);  sum += (int)((unsigned char)pBuffer[i\*2+1]);  }  }  while((sum & 0xffff0000) != 0) {  sum = (sum & 0xffff) + ((sum >> 16) & 0xffff);  }  unsigned short int ssum = (~sum) & 0xffff;  if(ssum != header\_checksum) {  ip\_DiscardPkt(pBuffer,STUD\_IP\_TEST\_CHECKSUM\_ERROR);  return 1;  } | |  | |  |   错误的校验和：0x00C8应该是0x2E19    **IPv4分组转发实验**   1. 路由表初始化、路由增加、路由转发三个函数的实现流程图；   路由表初始化：stud\_Route\_Init ( )函数  清空vector    路由表增加：stud\_route\_add ( )函数  从stud\_route\_msg结构中取得dest, masklen, nexthop，转为网络字节序之后经过处理，构建结构体route，并添加到vector中。    路由转发：stud\_fwd\_deal ( )函数  在 stud\_fwd\_deal ( )函数中，需要完成下列分组接收处理步骤：  查找路由表。根据相应路由表项的类型来确定下一步操作，错误分组调用函数 fwd\_DiscardPkt ( )进行丢弃，上交分组调用接口函数 fwd\_LocalRcv ( )提交给上层协议继续处理，转发分组进行转发处理。注意，转发分组还要从路由表项中获取下一跳的 IPv4 地址。  转发处理流程。对 IPv4 头部中的 TTL 字段减 1，重新计算校验和，然后调用下层接口 fwd\_SendtoLower ( )进行发送处理。     1. 所新建数据结构的说明；  |  | | --- | | struct route  {  uint low,high;  uint masklen;  uint nextIP;  route(uint low, uint high, uint masklen, uint nextIP)  {  this->low = low;  this->high = high;  this->masklen = masklen;  this->nextIP = nextIP;  }  }; |   low: 子网最小的IP地址  high：子网最大的IP地址  masklen：子网掩码长度  nextIP：下一调IP地址  3)在存在大量分组的情况下如何提高转发效率，如果代码中有相关功能实现，请给出具体原理说明。  1. 树形结构匹配路由表项：路由表存储结构由线性结构改为树形结构，提高匹配效率。  2. 并行检查：每次在转发分组时，都要检测数据合法性，计算校验和等操作，我们可以并行操作，就能提高转发效率，由硬件来实现。  3. 缓存分组：经过路由器的前后分组间的相关性很大，具有相同目的地址和源地址的分组往往连续到达，快速转发过程中，缓存分组，如果该分组的目的地址和源地址与转发缓存中的匹配，则直接根据转发缓存中的下一网关地址进行转发，减轻了路由器的负担，提高路由器吞吐量。 |
| 实验结果； |
|  |
| 心得体会： |
| 通过本次实验，我有以下几点收获：  ① 深入了解网络层协议的基本原理；   1. 学习了IPv4协议基本的分组接收和发送流程； 2. 了解IP报文的结构； 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，为后面进行更深入复杂的实验奠定基础； 4. 了解路由器是如何为分组选择路由，并逐跳地发送到目的主机； 5. 深入理解了路由表的数据结构，理解路由器是如何根据路由表对分组进行转发的。 |