router 大作业报告

白承桓 2017080338

December 5, 2021

1 handlepacket 流程

1.1 以太网头部判断与分配进程

从主函数 handlepacket 出发，首先会使用布尔函数 bool checkEther 判断以太网头部是 Arp 或是 IPv4 格式，若皆不是跳出 error，此后再利用 bool judgeMACtype 判断此包的目的 MAC 地址是否为此路由器还是广播或者是未知 MAC 地址。此后若是 IPv4，则进入 void incomeIPv4 函数进入处理 IPv4 进程，若是 Arp，则进入 void incomeArp 函数进入处理 Arp 进程。

1.2 IPv4 进程的处理

在确定收到 IPv4 数据报后，首先通过 bool checkIPv4 检查其校验和等合法性，然后若目的 interface 为此路由器，通过查看 pIPv4->ip_p，即协议，若协议为 tcp,udp，则根据 pdf 的要求执行 void sendReplyIcmpPortUnreachable 函数，发送 Icmp 包告诉 host 无法到达。

1.2.1 sendReplyIcmpPortUnreachable 函数重点部分展示

```
// refer to RFC 792
  pReplyIcmpT3->icmp_type = 3;
  pReplyIcmpT3->icmp_code = 3;
  pReplyIcmpT3->icmp_sum = 0;
```

若协议为 Icmp 协议，执行 void incomeIcmp 函数

1.2.2 void incomeIcmp 函数流程与重点部分展示

首先调用 bool checkICMP 判断 ICMP 是否合法，若 pIcmp->icmp_type 不为 8 或 0 或校验和有误则返回错误，然后会查看路由表和 arp 表准备回复，

```
// refer to RFC 792
  pReplyICMP->icmp_type = 0;
  pReplyICMP->icmp_code = 0;
  pReplyICMP->icmp_sum = 0;
```

若目的 interface 不是路由器，则需要进行转发，所以需要先确定生命周期是否过期，若过期，则通过执行 void sendReplyIcmpTimeExceeded 函数发送超时 Icmp 消息。

1.2.3 void sendReplyIcmpTimeExceeded 重点部分展示

```
// refer to RFC 792
 pReplyIcmpT3->icmp_sum = 0;
 pReplyIcmpT3->icmp_type = 11;
 pReplyIcmpT3->icmp_code = 0;
```

查看路由表后再对照 arp 表，若有，则执行 void sendIpv4Packet 函数

1.2.4 void sendIpv4Packet 重点部分展示

```
// ip 头部设置
 pForwardIPv4->ip_ttl --;  //生命周期减一
 pForwardIPv4->ip_sum = cksum(pForwardIPv4, sizeof(struct ip_hdr));
```

1.3 Arp 进程的处理

在确定收到 Arp 数据报后，与 ipv4 流程类似，检查合法性后，若 pArp->arp_op，即 Opcode==1 则执行 void sendArpReply 函数进行 arp request，若 Opcode==2 则执行 void incomeArpReply 函数进行 arp reply.

1.3.1 void sendArpReply 重点部分展示

```
  // set MAC dst and src
 memcpy(pReplyEther->ether_dhost, pEther->ether_shost, ETHER_ADDR_LEN);
 memcpy(pReplyEther->ether_shost, iface->addr.data(), ETHER_ADDR_LEN);
 pReplyArp->arp_op = htons(0x0002);
```

1.3.2 void incomeArpReply 重点部分展示

```
if (arpRequest == nullptr) {
   std::cerr << "No queued request to remove" << std::endl; // never requaired
 } else {                                    //
   for (auto pendingPacket: arpRequest->packets) {
     handlePacket(pendingPacket.packet, pendingPacket.iface);     //main
   }
   m_arp.removeRequest(arpRequest);
```

2 路由表 looking up

核心思想为最长前缀匹配，先获取所有入口，为了方便计算，对掩码进行去反后会变成 65535 或 4294967295 等二的幂次方，此后每向右 shift 一位，len+1，直到经变换后的掩码 <=0，此后再用 ip 长度 32-len 得到此入口的前缀长度，依次循环最长的 len 所对应的入口即是查看表的结果。

2.0.1 lookup 重点部分展示

```
unsigned rmask_minus_ntohl = ~ntohl(e.mask); // 先取反，一直shift到=0，shift越多，
    while (rmask_minus_ntohl > 0) { //取反的取反
      len ++;
      rmask_minus_ntohl /= 2;}
    len = IP_MAX_SIZE - len;
    if (max_len > len){}
    else { //最大前缀匹配
      max_len = len;
      res = e;
    }
```

3 periodicCheckArpRequestsAndCacheEntries 说明

参考 arp-cache.hpp 中的提示，对已超过 5 次的请求加入 invalidRequests 中，然后对每个请求队列进行 hendle_arpreq，对已超过 5 次请求的队列中的每个包，都执行 void sendReplyIcmpHostUnreachable 函数发送无法到达 Icmp 包。此后再对每个 invalidRequests 的请求进行移除操作并删除过期缓存条目。

3.0.1 void sendReplyIcmpHostUnreachable 重点部分展示

```
  // refer to RFC 792
pReplyIcmpT3->icmp_type = 3;
pReplyIcmpT3->icmp_code = 1;
pReplyIcmpT3->icmp_sum = 0;
```

4 难点

难点：

1. 阅读并遵循 RFC 792 协议。
2. 对各种已经写好的函数与变量的含义的了解与使用。
3. arp-cache 中对 requests 的理解。
4. 各种意外处理
5. 各种进程之间的先后顺序的理解

5 感想

通过本次大作业对报文的传输，icmp 协议，与 arp 表，路由表等只在理论中了解的机制具体在实际中是怎么互相作用的有了很大的了解。收获颇大。