网络原理：RIP 协议实验报告

黄家晖　2014011330

2016 年 12 月 8 日

1　实验目标

1. 深入理解 RIP 工作原理，了解 RIP 分组接收和发送流程；
2. 熟悉路由表的维护，深入掌握路由技术；
3. 培养路由协议变成开发能力。

2　实验原理

本实验要求实现 RIP 路由协议（全称为 Routing Information Protocol）v2 版本。该协议基于 UDP，各个网络节点向周围节点按一定的策略每隔特定时间发送自己的路由表，启动时还会主动请求周围节点的路由信息。

在进行实际路由时，路由根据路由表项利用距离向量算法利用最短路径进行路由，并维护每个表项的超时计数器，一旦计数器超时或综合得到的目的地跳数过大，则认为目的地不可达，路由表项失效。

3　程序设计

RIP 报文有效性检查　实验中此部分要求比较简单，仅需检查包的版本号和 command 字段是否合法即可，如果不合法则对包进行丢弃。

处理 Request 报文　RIP 协议规定当某个接口收到 Request 报文时应该发送路由表。根据水平分裂算法的规定，所有由该端口发送来的路由表项均不应该被回送，需要特殊判断。

处理 Response 报文　对于 reponse 报文中的每一个表项，首先检查该表项是否存在于本地路由表中。如果不存在且 metric 值在范围内，则添加本地路由表项；如果存在，则对比该路由表项的来源与本地路由表来源是否相同（即检查本地该项的下一条是否是表项来源地址），如果相同，则说明该本地项本身就应该由此路径更新，否则，比较当前表项和新表项的 metric 值，选择最短路径进行路由。

路由表项超时删除　超时时，需要把超时表项的 metric 值置为 16。

路由表项定时发送 RIP 协议规定，每隔 30s 应该重新广播一次路由信息，此时要向所有端口发送自己的本地路由表项。注意也要满足水平分裂算法的要求。

4 实验中遇到的问题

实验书中 P162 页对于 Response 分组的处理描述有问题，也致使我很长时间无法理解 RIP 算法的含义。经过同学的说明得到了解决，RIP 的详细描述请见上一节。

5 思考题

1 当 metric 值变为 16 的时候，即视作节点不可达。如果此时直接丢弃此路由表项和不向外广播，则对端路由很可能不清楚这个节点不可达的情况。在某些特殊情况下，当水平分裂算法失效时，路由不可达的消息不会主动发送，造成慢收敛。

相反地，如果向外广播 metric 值为 16 的表项，则对端路由就能够根据这个信息更新自己的路由表，确定目的地不可达，尽早发现通路问题。

2 相同点：RIP 和 OSPF 协议都属于路由协议，均工作在同一个自治域内。

不同点：1. RIP 以跳数来衡量到达目的节点的远近，且最大仅支持 16 跳，而 OSPF 则根据带宽和延迟衡量，没有最大距离限制；2. RIP 采用距离向量算法作为基础，内部存储本地路由表，每 30 秒进行更新，收敛速度比较慢；OSPF 采用链路状态算法作为基础，每个路由器维护整个网络拓扑结构，对存储和计算能力要求很高，但通过专门的报文通告拓扑变化，收敛速度快；3. RIP 需要定时发送整个路由表，相比 OSPF 占用更大带宽；4. RIP 基于 UDP 协议，工作在应用层，而 OSPF 工作在链路层。

适用范围：RIP 适用于小型网络（节点数大约不超过 10 个），且各个节点计算能力有限；OSPF 更适用于大型自治域内网络，由专门的大型路由器节点组成。

6 附录：源代码

```
1  #include "sysinclude.h"
2
3  extern void rip_sendIpPkt(unsigned char *pData, UINT16 len,unsigned short dstPort,UINT8 iNo);
4  extern void ip_DiscardPkt(char *pBuffer, int type);
5
6  extern struct stud_rip_route_node *g_rip_route_table;
7
8  #define RIP_UDP_PORT 520
9  #define INTERFACE_COUNT 2
10
11 struct rip_header
12 {
13   unsigned char command;
14   unsigned char version;
15   unsigned short must_be_zero;
```

```
16 };
17
18 struct rip_item
19 {
20   unsigned short afi;
21   unsigned short route_tag;
22   stud_rip_route_node info;
23 };
24
25 inline int add_and_truncate(int target) {
26   return (target == 16) ? 16 : target + 1;
27 }
28
29 inline void send_rip_items(UINT8 iNo)
30 {
31   unsigned char* data = new unsigned char[1024];
32   unsigned int data_ptr = 0;
33   // Create header
34   rip_header* response_header = (rip_header*) data;
35   response_header->command = 2;
36   response_header->version = 2;
37   response_header->must_be_zero = 0;
38   data_ptr += 4;
39   // Create Body.
40   stud_rip_route_node *route_table = g_rip_route_table;
41   while (route_table) {
42     // Hori-split:
43     if (route_table->if_no != iNo) {
44       rip_item* tp = (rip_item*) (data + data_ptr);
45       tp->afi = htons(2);
46       tp->route_tag = 0;
47       tp->info.dest = htonl(route_table->dest);
48       tp->info.mask = htonl(route_table->mask);
49       tp->info.nexthop = htonl(route_table->nexthop);
50       tp->info.metric = htonl(route_table->metric);
51       data_ptr += 20;
52     }
53     route_table = route_table->next;
54   }
55   rip_sendIpPkt(data, data_ptr, RIP_UDP_PORT, iNo);
56 }
57
58 int stud_rip_packet_recv(char *pBuffer,int bufferSize,UINT8 iNo,UINT32 srcAdd)
59 {
60   rip_header* header = (rip_header*) pBuffer;
61   // Packet Validation.
62   if (header->version != 2) {
```

```
63      ip_DiscardPkt(pBuffer, STUD_RIP_TEST_VERSION_ERROR);
64      return -1;
65    }
66    if (header->command == 1) {
67      // Request Command. -> Issue response
68      send_rip_items(iNo);
69    } else if (header->command == 2) {
70      // Response Command. -> Update Local RIP table
71      unsigned int data_ptr = 8;
72      while (data_ptr < bufferSize) {
73        stud_rip_route_node* current_item = (stud_rip_route_node*) (pBuffer + data_ptr);
74        // Check all local rip items.
75        stud_rip_route_node* route_table = g_rip_route_table;
76        bool found_in_local = false;
77        int current_metric = ntohl(current_item->metric);
78        while (route_table) {
79          if (route_table->dest == ntohl(current_item->dest) &&
80            route_table->mask == ntohl(current_item->mask)) {
81            if (route_table->nexthop == srcAdd) {
82              route_table->metric = add_and_truncate(current_metric);
83              // Mark that this item is from iNo.
84              route_table->if_no = iNo;
85            } else {
86              if (route_table->metric > current_metric) {
87                route_table->metric = add_and_truncate(current_metric);
88                route_table->nexthop = srcAdd;
89                route_table->if_no = iNo;
90              }
91            }
92            found_in_local = true;
93            break;
94          }
95          route_table = route_table->next;
96        }
97        if (!found_in_local) {
98          // Add to routing table.
99          stud_rip_route_node* new_route_table = new stud_rip_route_node();
100         new_route_table->dest = ntohl(current_item->dest);
101         new_route_table->mask = ntohl(current_item->mask);
102         new_route_table->nexthop = srcAdd;
103         new_route_table->metric = add_and_truncate(current_metric);
104         new_route_table->if_no = iNo;
105         new_route_table->next = g_rip_route_table;
106         g_rip_route_table = new_route_table;
107       }
108       data_ptr += 20;
109     }
```

```
110     } else {
111       ip_DiscardPkt(pBuffer, STUD_RIP_TEST_COMMAND_ERROR);
112       return -1;
113     }
114     return 0;
115   }
116
117   void stud_rip_route_timeout(UINT32 destAdd, UINT32 mask, unsigned char msgType)
118   {
119     if (msgType == RIP_MSG_SEND_ROUTE) {
120       // send responses.
121       for (UINT8 i = 0; i < INTERFACE_COUNT; ++ i)
122         send_rip_items(i + 1);
123     } else if (msgType == RIP_MSG_DELE_ROUTE) {
124       // delete an item.
125       stud_rip_route_node *route_table = g_rip_route_table;
126       while (route_table) {
127         if (route_table->dest == destAdd && route_table->mask == mask) {
128           route_table->metric = 16;
129           return;
130         }
131         route_table = route_table->next;
132       }
133     }
134   }
```