《计算机网络协议开发》实验报告

第 10 次实验

RIP 协议实验

姓名：元玉慧
学号：101220151
10 级 计算机 系 4 班
邮箱：njucsyyh@gmail.com
时间：2013/05/05

一、 实验目的

本实验可以假声学生深入理解 RIP 协议的工作原理，了解 RIP 协议分组的接收和发送流程，以及路由表的维护，进一步掌握计算机网络的核心技术，路由技术，并培养路由协议编程开发能力。

RIP 协议是 AS 自治系统内部选路协议，用于确定一个自治系统内执行选路的方式，AS 内部选路协议又称为内部网关协议。

二、实验设计背景

RIP 协议的产生：

20 世纪 80 年代，加州大学伯克利分校在开发 Unix 系统的同时，在 routed 程序中设计实现了 RIP 协议软件。routed 程序被绑定在 BSD Unix 系统中一起推出，被广泛的应用于早期网络中的机器之间交换路由信息。尽管 RIP/routed 没有非常突出的优点，但是由于 Unix 操作系统的普及，RIP/routed 也逐渐被推广出来，为许多人所接受，成为了中小型网络中最基本的路由协议/程序。

RIP 简述：

RIP 协议的 RFC 文本在 1988 年 6 月被正式推出，它综合了实际应用中许多实现版本的特点，同时为版本的兼容互通性提供了可靠的依据。由于 RIPv1 中存在着一些缺陷，再加上网络技术的发展，有必要对 RIP 版本进行相应的改进。1994 年 11 月，RFC1723 对 RIPv1 的报文结构进行了扩展，增加一些新的网络功能。1998 年 11 月，RIPv2 的标准 RFC 文本被正式提出，它在协议报文的路由表项中增加了子网掩码信息，同时增加了安全认证、不同路由协议交互等功能。

随着 OSPF、IS-IS 等域内路由协议的出现，许多人认为 RIP 协议软件已经过时。尽管 RIP 在协议性能和网络适应能力上远远落后于后来提出的路由协议，但是 RIP 仍然具有自身的特点。首先，在小型的网络环境中，从使用的网络带宽以及协议配置和管理复杂程度上看，RIP 的运行开销很小；其次，与其他路由协议相比，RIP 使用简单的距离-向量算法，实现更容易；最后，由于历史的原因，RIP 的应用范围非常广，在未来的几年中仍然会使用在各种网络环境中。因此，在路由器的设计中，RIP 协议是不可缺少的路由协议之一，RIP 协议的实现效率高低对路由器系统的路由性能起着重要的作用。

RIP 协议的基本特点：

RIP 协议使用 UDP 的 520 端口进行路由信息的交互，交互的 RIP 协议信息分组主要是 2 种类型：请求分组和响应分组，请求分组用来向相邻运行 RIP 协议的路由器请求路由信息，响应分组用来发送本地路由器的路由信息，RIP 协议使用距离向量路由算法，因此发送的路由信息可以使用<vector , distance>来表示，在实际分组中，vector 用路由的目的地址 address 来表示，而 distance 用该路由的距离度量值 metric 来表示，metric 的值规定为从本机到达目的网络路径上

经过的路由器的数目。Metric 的有效值是 1-16；其中 16 表示网络不可到达，可见 RIP 协议支持的网络规模是有限的。

当系统启动时，RIP 协议处理模块在所有 RIP 协议配置运行的接口处发出 request 分组，然后 RIP 协议就近路了循环等待的状态，等待外部 RIP 协议分组的到来；而接收到 request 分组的路由器应当发出包含他们的路由表信息的 response 报文。

当发出请求的路由器接收到一个 response 报文后，它会逐一处理接收到的路由表项内容。如果报文中的表项为新的路由表项，那么就会向路由表中加入该表项。如果该报文表项已经在路由表中存在，那么首先判断这个收到的路由更新信息是哪个路由器发送过来的。如果就是这个表项的源路由器（即当初发送相应路由信息从而导致这个路由表项的路由器），则无论该现有表项的距离度量值（metric）如何，都需要更新该表项；如果不是，那么只有当更新表项的 metric 值小于路由表中相应表项 metric 值时才需要替代原来的表项。

此外，为了保证路由的有效性，RIP 协议规定：每隔 30 秒，重新广播一次路由信息；若连续三次没有收到 RIP 广播的路由信息，则相应的路由信息失效。

水平分割

水平分割是一种避免路由环的出现和加快路由汇聚的技术。由于路由器可能收到它自己发送的路由信息，而这种信息是无用的，水平分割技术不反向通告任何从终端收到的路由更新信息。

RIPv2 的分组结构：

RIPv2 的报文结构如上图所示。每个报文都包括一个报文命令字段、一个报文版本字段、一个路由域字段、一个地址类字段、一个路由标记字段以及一些路由信息项（一个 RIP 报文中最多允许 25 个路由信息项），其中每个字段后括号中的数字表示该字段所占的字节数。RIP 报文的最大长度为 4+20*25=504 字节，加上 UDP 报头的 8 字节，一共是 512 字节。如果路由表的路由表项数目大于 25 时，那么就需要多个 RIP 报文来完成路由信息的传播过程。　　下面对报文字段进行逐一介绍：

(1) 命令字段 Command

表示 RIP 报文的类型，目前 RIP 只支持两种报文类型，分别是请求报文（request 1）和响应（response 2）报文

(2) 版本字段 Version

表示 RIP 报文的版本信息，RIPv2 报文中此字段为 2

(3) 路由域字段 Must Be zero

保留字段，取值为 0

(4) 地址类字段 Address Family Identifier

表示路由信息所属的地址族，目前 RIP 中规定此字段必须为 2，表示使用 IPv4 地址族

(5) IP 地址字段 IP Address

表示路由信息对应的目的地 IP 地址，可以是网络地址、子网地址以及主机地址

(6) 子网掩码字段 Subnet Mask

子网掩码。应用于 IP 地址产生非主机部分地址，为 0 时表示不包括子网掩码部分，使得 RIP 能够适应更多的环境

(7) 下一站 IP 地址字段 Next Hop

下一跳 IP 地址，如果存在的话，它标识一个比通告路由器的地址更好的下一跳地址。即它指出的下一跳地址。其度量值比在同一个子网上的通告路由器更靠近目的地。如果这个字段设置为全 0（0.0.0.0），说明通告路由器的地址是最优的下一跳地址。

(8) 度量值字段 Metric

表示从本路由器到达目的地的距离。目前 RIP 将路由路径上经过路由器的个数作为距离度量值。

一般来说，RIP 发送的请求报文和响应报文都符合上图的报文结构格式，但是当需要发送请求对方路由器全部路由表信息的请求报文时，RIP 使用另一种报文结构，此报文结构中路由信息项的地址族标识符字段为 0，目的地址字段为 0，距离度量字段为 16

三、实验设计要点

理解 RIP 协议，根据 RIP 协议的流程设计 RIP 协议的报文处理和超时处理函数。能够实现如下功能：

(1) RIP 报文有效性检查 对客户端接收到的 RIP 协议报文进行合法性检查，丢弃存在错误的报文并指出错误原因

(2) 处理 Request 报文 正确解析并处理 RIP 协议的 Request 报文，并能够根据报文的内容以及本地路由表组成相应的 Response 报文，回复给 Request 报文的发送者，并实现水平分割

(3) 处理 Response 报文 正确解析并处理 RIP 协议的 Response 报文，并根据报文中携带的路由信息更新本地路由表

(4) 路由表项超时删除 处理来自系统的路由表项超时消息，并能够删除指定的路由

(5) 由表项定时发送 实现定时对本地的路由进行广播的功能，并实现水平分割 。

四、实验内容

实验中用到的数据结构：

RIPv2 分组的头部结构：

```
typedef struct RIP{
    unsigned char command;
    unsigned char version;
    unsigned short TAG;
}* head;
```

RIPv2 分组携带的路由条目结构：

```
typedef struct Route{
    unsigned short addr_family_id;
    unsigned short route_id;
    unsigned int ip_add;
    unsigned int mask;
    unsigned int nexthop;
    unsigned int metric;
}* Route_item;
```

分离出来的实现把 RIPV2 分组的所有路由条目发送的函数：
参数是：接收该分组的接口号，以及 RIPv2 分组携带的路由条目的数目；
主要处理路由项个数小于 25 和大于 25 的情况。

```
void Get_Route(UINT8 iNo, int num){
    int count = num/25;
    int remain = num%25;
    stud_rip_route_node* p = g_rip_route_table;
    //大于25的话，需要先将封装好的数据包发送
    if(count > 0){
       for(int i=0; i<count; i++){
                unsigned char* buffer = new unsigned char[504];
                memset(buffer, 0, 504);
                buffer[0] = 2;
                buffer[1] = 2;
                Route_item temp = (Route_item)(buffer+4);
                for(int j=0; p!=NULL && j<25; p=p->next){
                    if(p->if_no != iNo){
                        temp[j].addr_family_id = htons(2);
                        temp[j].route_id = htons(0);
                        temp[j].ip_add = htonl(p->dest);
                        temp[j].mask = htonl(p->mask);
                        temp[j].nexthop = htonl(p->nexthop);
                        temp[j].metric = htonl(p->metric);
                        j++;
                    }
                }
                rip_sendIpPkt(buffer, 504, 520, iNo);
       }
    }
```

```
    unsigned char* buffer = new unsigned char[20*remain+4];
    memset(buffer, 0, 20*remain+4);
    buffer[0] = 2;
    buffer[1] = 2;
    Route_item temp = (Route_item)(buffer+4);
    for(int j=0; p!=NULL && j<remain; p=p->next){
        if(p->if_no != iNo){
            temp[j].addr_family_id = htons(2);
            temp[j].route_id = htons(0);
            temp[j].ip_add = htonl(p->dest);
            temp[j].mask = htonl(p->mask);
            temp[j].nexthop = htonl(p->nexthop);
            temp[j].metric = htonl(p->metric);
            j++;
        }
    }
    rip_sendIpPkt(buffer, 20*remain+4, 520, iNo);
}
```

水平分割：是一种避免路由环路出现和加快路由汇聚的技术。

由于路由器可能收到自己发送的路由信息，而这些信息是无用的，水平分割技术不反向通告任何从终端收到的路由更新信息；

水平分割的规则和原理：

路由器从某个接口接收到的更新信息不允许再从这个接口发回去。

能够避免路由环路的产生；减少路由器更新信息占用的链路带宽资源。

在路由信息传送过程中，不再把路由信息发送到接收到此路由信息的接口上。

每一个路由表项都记录了当前路由表项是从哪个接口接收到的。

具体体现在如下代码：

```
for(int j=0; p!=NULL && j<25; p=p->next){
    if(p->if_no != iNo){
```

RIP 分组处理函数：

合法性判断：

```
head buf = (head)pBuffer;
if(buf->version != 2){
    ip_DiscardPkt(pBuffer, STUD_RIP_TEST_VERSION_ERROR);
    return -1;
}
if(buf->command != 1 && buf->command != 2){
    ip_DiscardPkt(pBuffer, STUD_RIP_TEST_COMMAND_ERROR);
    return -1;
}
```

处理 Request 分组，response 分组不包括来自该来源接口的路由：

```
//请求报文
if(buf->command == 1){
    printf("\nRequest\n");
    int num = 0;
    //过滤掉来自该源接口的路由，获取有效路由表项的数目
    stud_rip_route_node* p = g_rip_route_table;
    while(p != NULL){
        if(p->if_no != iNo)
            num++;
        p = p->next;
    }
    Get_Route(iNo, num);
}
```

处理 Response 分组，需要提取出该分组携带的所有路由信息：

获取路由表项的个数：

```
else if(buf->command == 2){
    Route_item temp = (Route_item)(pBuffer+4);
    int Route_count = (bufferSize-4)/20;
```

查找路由表项，若找到路由表项则需要判断条数并更新：

若匹配后的路由项通知 metric 条数>= 16，则需要特殊处理，表示此路由通路已经断开；

否则，按照 metric+1 后与路由表项的 metric 比较，若更小，则需要更新路由表；

```
//查找路由表中是否存在路由表项
while(p!=NULL){
    if (ntohl(temp[i].ip_add)==p->dest && ntohl(temp[i].mask)==p->mask){
        if(metric >= 16){
            p->metric = 16;
        }
        else if(metric+1 < p->metric){
            p->nexthop = srcAdd;
            p->metric = metric+1;
            p->if_no = iNo;
        }
        break;
    }
    p = p->next;
}
```

若没有匹配到路由表项则需要添加新的路由表项到路由表中：

```
//若分组的表项是新的路由表项，则插入路由表中
if(p==NULL){
    stud_rip_route_node* node = new stud_rip_route_node;
    node->dest = ntohl(temp[i].ip_add);
    node->mask = ntohl(temp[i].mask);
    node->nexthop = srcAdd;
    node->metric = (metric+1 <= 16)? metric+1 : 16;
    node->if_no = iNo;
    //在链表头部插入路由表项
    node->next = g_rip_route_table;
    g_rip_route_table = node;
}
```

RIP 超时处理函数：

若消息类型为路由信息广播，则会在每一个接口上广播自己的路由信息；

否则，若到特定路由目的地址的路由信息更新超时了，则会设置其对应的路由通路的 metric 为 16；

```
void stud_rip_route_timeout(UINT32 destAdd, UINT32 mask, unsigned char msgType)
{
    stud_rip_route_node *p = g_rip_route_table;
    if(RIP_MSG_DELE_ROUTE == msgType){
        while(p != NULL){
            if(p->dest == destAdd && p->mask == mask)
                p->metric = 16;
            p = p->next;
        }
    }
    else if (RIP_MSG_SEND_ROUTE == msgType){
        int num1 = 0, num2 = 0;
        p = g_rip_route_table;
        while(p != NULL){
            if(p->if_no != 1)
                num1++;
            if(p->if_no != 2)
                num2++;
            p = p->next;
        }
        Get_Route(1, num1);
        Get_Route(2, num2);
    }
}
```

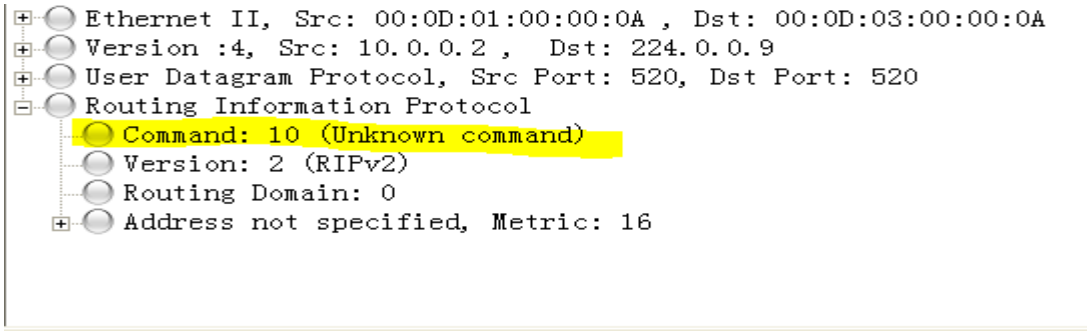
五、实验结果

1	Fri May 03 18:03:2...	10.0.0.2	224.0.0.9	UDP	RIP	6.1 RIP报文有效性检查
2	Fri May 03 18:03:3...	10.0.0.2	224.0.0.9	UDP	RIP	6.2 Request报文处理
3	Fri May 03 18:03:3...	10.0.0.1	10.0.0.2	UDP	RIP	6.2 Request报文处理
4	Fri May 03 18:03:3...	10.0.0.2	224.0.0.9	UDP	RIP	6.3 Response报文处理
5	Fri May 03 18:03:4...	10.0.0.1	10.0.0.2	UDP	RIP	6.5 路由表项定时发送
6	Fri May 03 18:03:4...	20.0.0.1	20.0.0.2	UDP	RIP	6.5 路由表项定时发送

【1】 RIP 分组的处理：

1 号报文：

分析：报文信息可见；

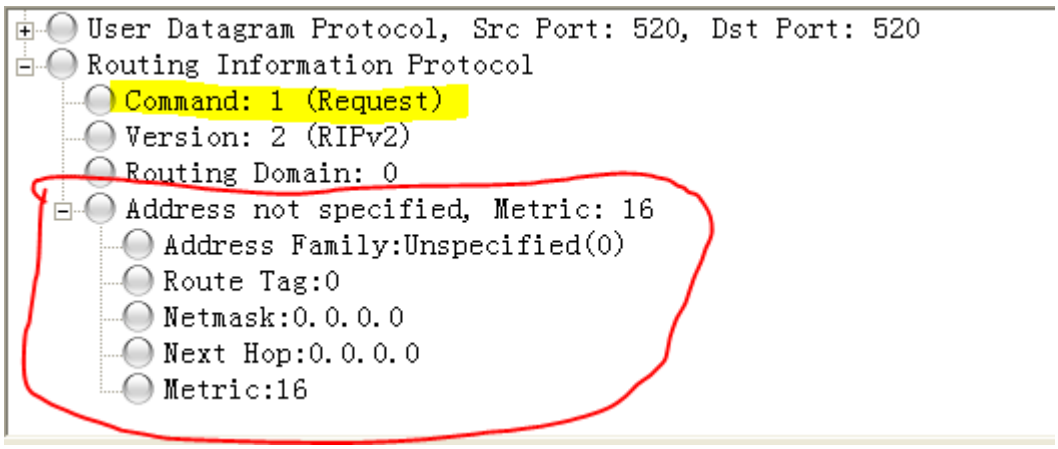


Command 为 10，是一种 unknown command， 所以该分组无法被识别，将会被丢弃。

2 号报文：

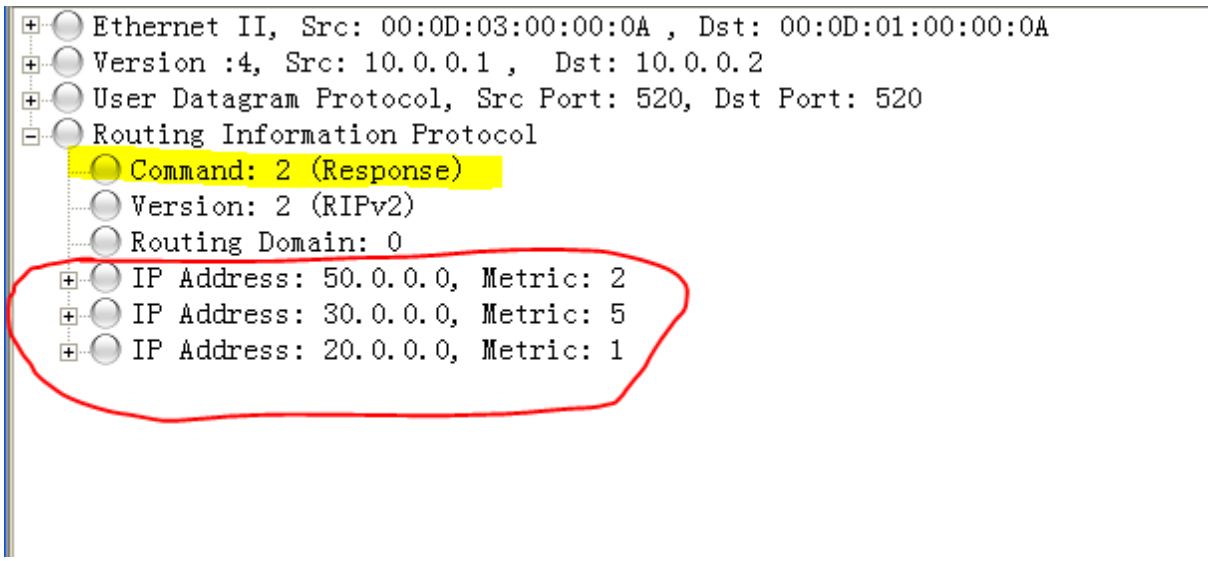
分析：以条 request 报文，地址族标识符字段为 0，目的地址字段为 0， 度量字段为 16，即为请求路由器全部路由表信息。

本接收以后，看到接口号为 1，在本地路由表中查找接口号为 2 的路由条目信息，组织成 response 报文并发送出去。



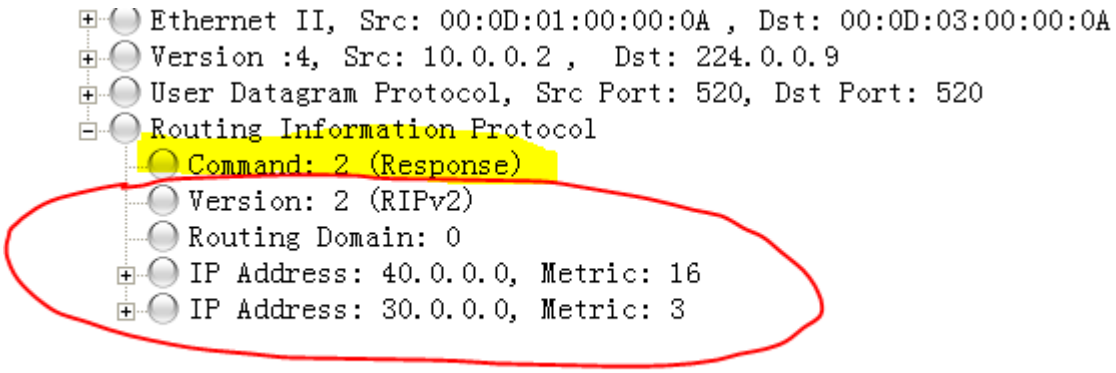
3 号报文：

根据 2 号报文的分析，得知 3 号报文是对全部路由信息请求报文的应答报文，2 号报文从 1 号接口发来，所以 3 号报文发送的路由信息为接口为 2 的路由表



4 号报文：

收到一个 response 报文，路由条目只有一项，其 IP address 为

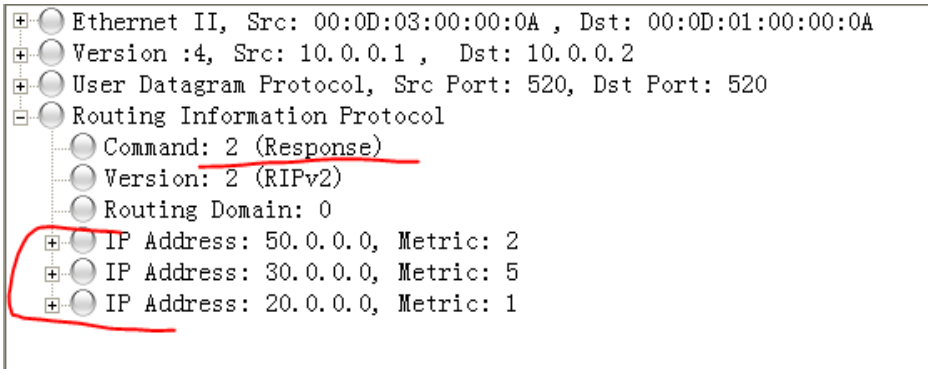


路由条目中只有一项，其地址为 40.0.0.0 ， NetMask 为 255.255.0.0. 查找路由表，没有该表项，将修改路由信息并添加到路由表中；

【2】 RIP 超时处理：

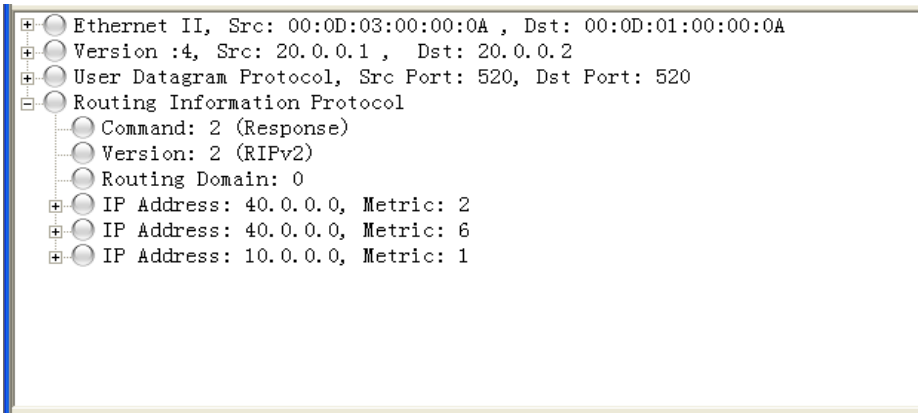
5 号报文：

这是一个定时发送的报文，因为有 2 个接口，这个报文是从 1 号接口发送出去的，由于要进行水平分割，所以只需要发送路由表中接口信息为 2 的路由条目，10.0.0.2 为组播地址。



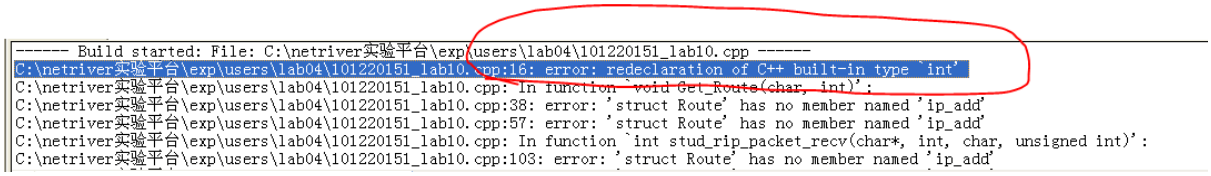
6 号报文：

这是从 2 好接口发送出去的报文，由于需要进行水平分割，所以只需要发送路由表中的接口信息为 1 的路由条目。



六、实验问题及解决方案

-1- 变量命名的问题：



问题在于：ip_addr 可能是系统的宏定义部分，所以定义为 ip_addr 为报： redeclaration 的错误：

```
typedef struct Route{
    unsigned short addr_family_id;
    unsigned short route_id;
    unsigned int ip_addr;
    unsigned int mask;
    unsigned int nexthop;
    unsigned int metric;
}* Route_item;
```

-2- 如何判断路由信息对应的路由项为新的路由项：

跟 IPv4 路由信息查询的方法一样，通过计算子网编号来确定路由信息是否相同，进行更新路由信息；

-3- RIPV1 和 RIPV2 的区别：

RIPV1 是有类别路由选择协议，RIPV2 是无类别路由选择协议，RIPV1 没有子网掩码，RIPV2 有子网掩码；

其次 RIPV1 没有下一跳和路由标记，RIPV2 有下一跳，用于路由更新的重定向，也有路由标记，用于过滤和做策略；

RIPV1 没有认证的功能，RIPV2 有，RIPV1 是广播更新，RIPV2 是组播更新；

七、思考问题

-1-静态路由选择协议和动态路由选择协议的不同

静态路由选择协议：

路由器可以以来人工变成把选择路径输入到设备中；在路由器中设置固定的路由表，静态路由的优点是：简单，高效，可靠。在所有的路由中，静态路由的优先级是最高的，当静态路由和动态路由发成冲突的时候，以静态路由为准；

动态路由选择协议：

网络中的路由器之间相互传递路由信息，利用收集到的路由信息来不断更新路由表的过程，能够实时的适应网络结构的变化，动态路由协议适用于网络规模大，网络拓扑结构复杂的网络。常见的 RIP，OSPF 都是动态路由协议；

-2-比较适量路由选择协议和链路状态路由选择协议：

RIP 是距离矢量路由选择协议，它选择路由的度量标准是跳数，最大跳数是 15 跳，如果大于 15 跳，它就会丢弃数据包。 RIP 的局限性在大型网络中使用所产生的问题：

i. RIP 的 15 跳限制，超过 15 跳的路由被认为不可达；

ii. 周期性广播整个路由表，在低速链路及广域网云中应用将产生很大问题；

iii. 收敛速度慢于 OSPF，在大型网络中收敛时间需要几分钟；

iv. RIP 没有网络延迟和链路开销的概念，路由选路基于跳数。拥有较少跳数的路由总是被选为最佳路由即使较长的路径有低的延迟和开销；

v. RIP 没有区域的概念，不能在任意比特位进行路由汇总。

一些增强的功能被引入 RIP 的新版本 RIPv2 中，RIPv2 支持 VLSM，认证以及组播更新，但 RIPv2 的跳数限制以及慢收敛使它仍然不适用于大型网络。

OSPF 协议是链路状态路由选择协议，它选择路由的度量标准是带宽，延迟。相比 RIP 而言，OSPF 更适合用于大型网络，这是因为：

i. 没有跳数的限制

ii. 使用组播发送链路状态更新，在链路状态变化时使用触发更新，提高了带宽的利用率

iii. 收敛速度快

-3-在 stud_ip_packet_recv()函数的实现中，如果接收到 Request 分组，是否直接根据本地路由信息组成 Response 分组，并考虑到水平分割算法：

不是；需要注意路由表项的个数可能大于 25，而每一个 RIPV2 分组最多只能携带 25 个路由表想，因此需要进行处理

八、 实验的启示/意见和建议

此次实验大约花费了 25+小时吧！包括写代码+调代码+写报告