路由器开发实验指导

**第一部分搭建路由器**

利用现有软件搭建路由器。路由协议采用Quagga，转发引擎利用Linux内核。

**1实验目的**

1）了解路由器的结构

2）熟悉软件路由器的搭建方法

3）熟悉路由器静态路由配置

4）熟悉路由器协议RIP的配置

**2实验内容**

**2.1实验一：软件下载、安装及配置**

**（1）实验环境**

如图1.1所示，所有学生机通过网络与测试服务器连接。



图1.1路由开发环境

**（2）实验操作**

从测试服务器（或互联网）下载Quagga软件，编译安装软件，并进行配置，从而完成软件路由器的搭建，可以进行路由器操作（如显示接口状态、配置接口地址等）。

在CentOS-6.5系统64位机上安装quagga-0.99.21过程（**以下安装为完整过程，若系统中已存在相关软件模块，可省略相关步骤**）：

1）安装g++：

yum install gcc yum install gcc-c++

2）安装gawk：

yuminstall gawk

3）安装ncurses：

yum install ncurses-libs

yum install ncurses-devel

4）安装libreadline:

第一种方法：yum install readline-devel

如果第一种方法在编译quagga-0.99.21时出现vtysh这个目录下不能make过的情况，请使用第二种方法安装readline。

第二种方法：

下载readline-6.2.tar.gz

wget -c ftp://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz

解压readline-6.2.tar.gz

tarxzf readline-6.2.tar.gz

编译安装

进入解压出来的目录中

./configure

make&& make install

ldconfig

5）安装quagga-0.99.21

1.解压压缩包：

tarxzf quagga-0.99.21\_20131218.tar.gz

2.编译安装

./configure--enable-vtysh --enable-zebra --enable-ripd--disable-doc --disable-babeld --enable-isisd=no --disable-bgpd --disable-ospfd --disable-ospf6d --disable-ospfclient--enable-user=root --enable-group=root -enable-vty-group=root

make&& make install

加载库文件路径

ldconfig

创建log目录

mkdir /var/log/quagga/

编辑配置文件

cp/usr/local/etc/zebra.conf.sample /usr/local/etc/zebra.conf

cp/usr/local/etc/ripd.conf.sample/usr/local/etc/ripd.conf

进入/usr/local/etc/zebra.conf文件：

vim /usr/local/etc/zebra.conf进行修改

将最后一行改成 log file /var/log/quagga/zebra.log

6）启动程序

**zebra启动：**

zebra -d 启动 zebra

zebrad -d启动后用ps–ef | grep zebra 能看到已经启动的zebra程序

**vtysh启动：**

zebra启动后输入vtysh启动vtysh

**2.2实验二：转发功能测试**

**（1）实验环境**

如图1.2所示，学生机的两个网卡分别通过网络与测试服务器连接。



图1.2转发功能环境

**（2）实验操作**

学生机的网卡1与测试服务器的一个网卡相连，且配置同一网段IP地址；学生机的网卡2与测试服务器另一个网卡相连，且配置同一网段IP地址。学生机两个网卡的IP地址不在同一网段。运行zebra，在学生机上配置静态路由，下一跳为网卡2直连的网络接口IP地址，目的IP为自己指定的另一网段IP地址，测试服务器向学生机发送目的IP地址ping包，在测试服务器和网卡2相连的网络接口使用抓包工具，若学生机转发成功，可以抓到测试服务器发出的ping包。

**2.3实验三：路由协议RIP配置**

**（1）实验环境**

如图1.3所示，学生机通过网络与测试服务器连接；学生机和测试服务器上都运行路由协议RIP。



图1.3RIP配置环境

**（2）实验操作**

学生机上先执行Ctrl+D，跳出到Linux界面，执行ripd-d，后台运行RIP。在测试服务器上也配置上RIP。**要求：**学生机能够与测试服务器建立邻居关系，并且能够相互学习到对方路由，在测试服务器上配置一条静态路由，学生机可以通过RIP学到这条路由。

**第二部分开发转发引擎**

路由协议软件采用Quagga，转发引擎需要自己开发。

**1实验目的**

1）了解转发表的结构

2）了解路由查找基本原理

3）了解IP数据包的数据格式

4）熟悉报文转发基本流程

**2实验流程图**

图2.1转发流程图

**3实验内容**

**3.1实验一：构建全局路由表**

**（1）实验环境**

在学生机上实现。

**（2）实验操作**

构建全局路由表，编写插入函数，通过链表（或其它数据结构）存储的方式将自定义的路由表项存储在路由表里。

插入函数：

int insert\_route(unsigned long ip4prefix, unsigned int prefixlen, char \* ifname, unsigned int ifindex, unsigned long nexthopaddr)

参数1是目的地址，参数2是掩码长度，参数3是接口名，参数4是接口索引值，参数5是下一跳IP地址。

**3.2实验二：校验和计算**

**（1）实验环境**

学生机上实现。

**（2）实验操作**

* 校验和验证函数

int check\_sum(unsigned short \* iphd, int len, unsigned short checksum);

参数1是指向IP数据包头的指针ip\_recvpkt，参数2是IP数据包头长度，参数3是IP数据包的校验和

* 重新计算校验和函数

unsigned short count\_check\_sum(unsigned short \* iphd);

参数是指向IP数据包头的指针ip\_recvpkt。

经过本机路由器转发的数据包，ttl减1，重新计算校验和，以便封装IP数据包头时使用。

**3.3实验三：编写查表函数和撤销路由函数**

**（1）实验环境**

学生机上实现。

**（2）实验操作**

提取接收IP数据包的目的IP地址查找路由表，获取出接口和下一跳IP地址；撤销路由函数用来撤销失效的路由表项。

* 查表函数

Int lookup\_route(structin\_addr dstaddr, struct nextaddr \* nexthopinfo);

参数1是目的IP地址ip\_recvpkt->ip\_dst，参数2是存储下一跳和出接口信息的结构体。

* 撤销路由函数

Int delete\_route(struct in\_addr dstaddr, unsigned int prefixlen);

参数1是目的IP地址，参数2是掩码长度。

**3.4实验四：接收静态路由信息并存储**

**（1）实验环境**

学生机上实现。

**（2）实验操作**

创建tcp socket服务器端，调用线程使用监听接收函数来接收quagga传送的静态路由信息，（监听函数int static\_route\_get(struct selfroute \*selfrt)，参数selfrt是存储路由表所需信息的结构体），再调用插入函数将其存储在路由表里，撤销静态路由时调用撤销路由函数删除在路由表的表项；

从quagga路由管理模块获取静态路由信息具体如下：

zebra\_rib.c文件里的rib\_install\_kernel函数下发路由信息到Linux内核。

rib\_install\_kernel函数-->

kernel\_add\_ipv4函数-->

netlink\_route\_multipath函数

struct prefix存放前缀信息，struct rib存放下一跳信息。

可以在这里用tcp socket创建客户端，将路由信息发送给作为服务器端的路由转发处理程序。

**3.5实验五：重新封装数据包**

**（1）实验环境**

学生机上实现。

**（2）实验操作**

* 填充以太网包头

将获取到的下一跳接口信息存储到存储接口信息的结构体ifreq里，通过ioctl获取出接口的MAC地址作为以太网包头的源MAC地址，目的MAC地址为下一跳IP地址对应的MAC地址，目的MAC地址可以通过ARP表项获取，据获取的信息封装以太网包头，以太网类型eth\_header->ether\_type = htons(ETHERTYPE\_IP)；

* 填充IP数据包头，进行校验；
* 将剩余的IP数据包信息填充到后面，通过raw socket将重新封装的数据包发送出去。

**4主要实现功能说明**

（1）创建全局路由表链表，调用插入函数向路由表中插入表项

路由表结构体如下：

struct route

{

struct route \* next; /\* Link list. \*/

struct in\_addr ip4prefix; /\* ip prefix \*/

unsigned int prefixlen; /\* ipprefixlen \*/

struct nexthop \* nexthop; /\* Nexthop structure \*/

};

下一跳信息结构体如下：

struct nexthop

{

struct nexthop \* next;

char \* ifname;

unsigned int ifindex; /\* Interface index. \*/

struct in\_addr nexthopaddr; /\* Nexthop address \*/

}

（2）创建tcp socket服务器端，调用线程来监听接收quagga传送的静态路由信息，将其存储在路由表里。

quagga路由管理模块获取静态路由信息如下：

zebra\_rib.c文件里的netlink\_route\_multipath函数下发路由信息到Linux内核。

struct prefix存放前缀信息，struct rib存放下一跳信息。

可以在这里用tcp socket，将quagga路由管理模块作为客户端，将路由信息发送给作为服务器端的路由转发处理程序，收到路由消息后存储进路由表。

（3）编写撤销路由函数，以便撤销失效的路由

（4）通过rawsocket套接字在数据链路层捕获IP数据包

int recvfd ;

ssize\_t recvlen;

recvfd=socket(AF\_PACKET, SOCK\_RAW, htons(ETH\_P\_IP));

recvlen=recv(recvfd, skbuf, sizeof(skbuf), 0);

创建套接字，然后接收IP数据包，IP数据包存储在skbuf里，struct ip \* ip\_recvpkt;让ip\_recvpkt指向skbuf中的IP数据包头；

（5）对IP数据包进行校验处理，校验和正确则继续，错误则丢弃该数据包

（6）取出IP数据包的目的IP地址，根据此信息查询路由表，获取路由表中相匹配的出接口信息和下一跳IP地址；

（7）通过查询ARP表得到下一跳的MAC地址

获取下一跳MAC地址函数:

Int arpGet(struct arpmac \* srcmac, char \* ifname, char \* ipStr);

参数1是存储下一跳MAC和出接口索引值的结构体，参数2是下一跳接口名称，参数3是下一跳IPv4地址。

（8）将IP数据包包头的TTL减1，再重新计算校验和

（9）重新封装数据包，通过raw socket从查询路由表获取的出接口经数据包发送出去

* 将获取到的下一跳接口信息存储到存储接口信息的结构体ifreq里，通过ioctl获取出接口的MAC地址作为以太网包头的源MAC地址，目的MAC地址为下一跳IP地址的MAC地址，根据获取的信息封装以太网包头，以太网类型eth\_header->ether\_type = htons(ETHERTYPE\_IP)；
* 再填充IP数据包头，进行校验；
* 通过raw socket将重新封装的数据包发送出去。

**第三部分开发RIP协议**

RIP路由协议采用Quagga，需要开发RIP报文的发送和接收处理，以及路由计算。

**1实验目的**

1）了解RIP协议原理

2）了解RIP协议报文交互过程

3）了解RIP协议报文处理过程

4）熟悉RIP协议报文的发送和接收

**2实验原理**

RIP是一种内部网关协议（IGP），是一种动态路由选择协议，用于自治系统（AS）内的路由信息的传递。RIP协议基于距离向量算法，使用“跳数”(即metric)来衡量到达目标地址的路由距离。在默认情况下，RIP使用一种非常简单的度量制度：距离就是通往目的站点所需经过的链路数，取值为1~15，数值15表示路径无限长。RIP进程使用UDP的520端口来发送和接收RIP分组。RIP分组分为两种：请求分组和响应分组。

启动RIP之后，在配置network之后，会向直连主机组播发送RIP请求报文，请求一份完整的路由表，然后开始接收RIP报文，直连主机在接收到request请求报文后，会查询自身的RIP路由表，获取RIP路由表信息，然后修改命令为response，再组播发送回去，自身路由发送变化时，会通过更新路由组播发送给直连路由器。如图3.1所示。



图3.1 RIP协议流程图

RIP报文格式如下：报文头部是4个字节，包括了“版本”字段2个字节和“命令”字段2个字节。路由表项是16个字节, 其中地址族标识占2个字节，路由标记占1个字节，度量值占1个字节，子网掩码、下一跳、IP地址各自占4个字节。一个完整的RIP报文可以有多个路由表项，但报文的总长度不能超过udp报文长度。详见表3.1。

|  |  |  |
| --- | --- | --- |
| 版本version(2) | 命令command(2) | |
| 地址族标识AF(2) | 路由标记Tag(1) | 度量值metric(1) |
| 子网掩码(4) | | |
| 下一跳(4) | | |
| IP地址(4) | | |

表3.1 RIP报文格式

version:RIP报文版本，默认情况下为2；

command:定义了报文类型(response或者request)；

AF:Address family identifier,发送请求报文时地址族默认为0；

Tag:外部路由标记；

metric:路由跳数，请求完整的RIP报文时该值为16；

子网掩码:用于区分子网；

Next Hop:下一跳地址；

IP地址:目标地址IP。

**3实验内容**

**3.1实验一：RIP请求报文的发送和接收以及RIP更新报文的发送**

**（1）实验环境**

如图3.2所示，学生机通过网络与测试服务器连接；利用学生机上的网卡1和测试服务器相连接，测试服务器上运行路由协议RIP。



图3.2RIP请求报文开发环境

**（2）实验操作**

**步骤1:获取RIP代码，搭建环境**

* 获取支持RIP的quagga代码，在PC机上构建运行和开发环境。
* 熟悉和了解RIP协议的原理，为开发RIP协议的模块做准备。

**步骤2:实现RIP发送处理模块**

**2.1:RIP发送请求报文，实现rip\_request\_interface\_send函数。**

* 判断RIP协议版本是否为RIPv2并且支持组播；
* 填充请求完整的RIP报文内容，command字段为RIP\_REQUEST；
* 调用rip\_send\_packet函数将报文发送给每一个直连网络。

**2.2:RIP发送更新报文，实现rip\_update\_process函数。**

**2.2.1发送更新给每一个直连网络：**

* 使用ALL\_LIST\_ELEMENTS\_RO函数遍历每一个接口；
* 查看是否是loopback口，是的话则continue；
* 获取RIP接口信息，当指定被动接口时，抑制响应；
* 使用ALL\_LIST\_ELEMENTS遍历每一个直连网络，使用rip\_update\_interface发送更新到每一个直连网络，其地址族需为AF\_INET。

**2.2.2发送更新给每一个邻居：**

* 使用route\_top和route\_next来遍历每一个邻居；
* 使用if\_lookup\_address和connected\_lookup\_address来确保邻居是连接网络的；
* 设置目的地址和端口，使用rip\_output\_process发送更新报文给邻居。

**步骤3:实现RIP接收处理模块**

**3.1:RIP接收数据处理，实现rip\_read函数。**

* 使用THREAD\_FD函数获取套接字，使用recvfrom接收数据包；
* if\_check\_address函数检查是否从自己接口发出，是则忽略该数据包；
* 根据if\_lookup\_address函数丢弃不正确接口的数据包；
* 根据if\_lookup\_address和connected\_lookup\_address找出ipv4地址；
* RIP协议版本的检验；
* 请求报文或者响应报文处理。

**3.2:RIP请求报文处理，实现rip\_request\_process函数。**

* 使用if\_is\_loopback函数，判断是否不响应loopback接口；
* 检查接口是否enable；
* 当指定被动接口时，抑制响应；
* 更新RIP邻居rip\_peer\_update；
* 如果是请求完整的RIP路由表,调用rip\_output\_process响应请求发送路由；
* 若不是请求完整的RIP路由表，则按照条目处理请求报文，调用rip\_send\_packet函数发送。

**3.2实验二：RIP响应报文的接收**

**（1）实验环境**

如图3.3所示，学生机通过网络与测试服务器连接；利用学生机上的网卡1和测试服务器相连接，测试服务器上运行路由协议RIP。



图3.3 RIP响应报文开发环境

**（2）实验操作**

**步骤1:获取RIP代码，搭建环境**

* 获取支持RIP的quagga代码，在PC机上构建运行和开发环境。
* 熟悉和了解RIP协议的原理，为开发RIP协议的模块做准备。

**步骤2:实现RIP接收处理模块**

参考实验三中第一个小实验步骤3中的3.1 rip\_read函数。

**步骤3:实现RIP响应报文处理模块**

**3.1:RIP响应报文处理，实现rip\_response\_process函数。**

* 检查包是否是从RIP 接口（520）接收的，不是则丢弃；
* 使用if\_lookup\_address函数检验数据报的ipv4源地址是否是直连的邻居；
* 使用rip\_peer\_update函数，更新RIP邻居；
* 获得路由条目，遍历路由条目，检验每一个条目：

检验过程：

* 地址族校验，RIP仅支持AF\_INET；
* 使用rip\_destination\_check函数判断目的地址是否可靠；
* 使用if\_lookup\_address函数查看下一跳是否存在；
* 如果下一跳是组播地址，则忽略该条目；使用if\_lookup\_address查询下一跳地址是否存在,存在返回非空,使用route\_node\_match\_ipv4在路由表中匹配下一跳地址,如果匹配到,若路由的类型为ZEBRA\_ROUTE\_RIP并且路由的sub类型为RIP\_ROUTE\_RTE，则设置下一跳为源地址；若不是，下一跳设为0；如果匹配不到，下一跳也设为0。
* 使用rip\_rte\_process函数更新RIP路由表以及发送内核。

**3.2:更新RIP路由表以及发送内核，实现rip\_rte\_process函数。**

* 填充prefix结构体；

struct prefix\_ipv4

{

u\_charfamily;

u\_charprefixlen;

structin\_addrprefix \_\_attribute\_\_ ((aligned (8)));

};

* 使用apply\_mask\_ipv4函数确保掩码是实用的；
* 使用rip\_offset\_list\_apply\_in函数处理偏移列表；
* 如果偏移列表不修改metric,使用接口的度量(metric)，如果度量值大于16，设置度量值为16；
* 设置下一跳指针，如果下一跳为空,设置为源地址,非空,设置为条目的下一跳；
* 使用rip\_nexthop\_check函数检查下一跳地址是否是自己，是直接返回；
* 使用route\_node\_get函数从路由表里获取前缀的索引；
* 检查路由表里是否已经有了路由：
  + **如果有路由：**
    - 将下一跳地址与数据报的路由器地址进行比较。如果该数据报来自与现有路由相同的路由器，则rip\_timeout\_update函数重新初始化超时时间；
    - 比较度量，如果数据报与现有路由来自同一路由器，并且新度量与旧度量不同；或者新度量低于旧度量，或者如果标签已经改变；或者存在具有较低管理距离的路由，或者更新实际ROU上的距离。做下列操作：
      * 根据数据报的路由，填写新的路由信息并调整下一跳的地址；
      * 如果旧的metric为无穷大，则新的metric不为无穷大,设置路类型为ZEBRA\_ROUTE\_RIP，sub类型为RIP\_ROUTE\_RTE；使用RIP\_TIMER\_OFF函数关闭定时器；调整下一跳，使用rip\_zebra\_ipv4\_add添加路由到内核;设置路由标志；
      * 如果旧的metric不为无穷大,使用rip\_zebra\_ipv4\_delete函数从内核删除原来的路由；调整下一跳,使用rip\_zebra\_ipv4\_add添加路由到内核并更改路由标志；
      * 调用rip\_event(RIP\_TRIGGERED\_UPDATE, 0);通知进程触发更新；
      * 如果新度量为无穷大，如果旧的metric不为无穷大,使用 RIP\_TIMER\_ON 和RIP\_TIMER\_OFF函数将垃圾收集计时器设置为120s,使用rip\_zebra\_ipv4\_delete从内核删除原来的路由；更改路由标志为RIP\_RTF\_FIB；
      * 如果新度量不为无穷大,使用rip\_timeout\_update函数重新计时。
  + **如果没有这样的路由，度量值也不是无穷大,则添加路由：**
    - * 设置RTE条目的目标前缀和长度；
      * 设置度量为新计算的度量，标签为条目的标签；
      * 设置下一跳地址(from)为从数据报来的路由器地址；
      * 使用rip\_timeout\_update函数停止定时器；设置路由更改标志；
      * 调用rip\_event (RIP\_TRIGGERED\_UPDATE,0);通知进程触发更新；
      * 使用rip\_distance\_apply获得距离值；
      * 使用rip\_zebra\_ipv4\_add将路由下发到内核，更改路由标志。

**第四部分综合实验**

RIP协议和转发引擎、路由管理、转发引擎性能的测试。

**1实验要求**

第一个实验通过RIP协议下发路由和配置静态路由发送转发引擎，分别验证是否可以转发，第二个实验室是路由管理优先级的测试，第三个实验通过多条有效路由对转发引擎进行测试，验证引擎的转发速率。

**2实验内容**

**2.1实验一：RIP协议和转发引擎**

**（1）实验目的**

通过实验验证静态路由的转发和Rip路由的转发。

**（2）实验环境**

如图4.1所示，学生机通过网络与测试服务器连接；利用学生机上的网卡1和网卡2与测试服务器相连接，测试服务器和学生机上运行路由协议RIP。



图4.1 RIP协议和转发引擎验证环境

**（3）实验操作**

**步骤1:搭建环境**

* 在学生机上运行自己开发好的RIP协议,和测试服务器建立RIP连接。
* 在学生机上运行自己开发好的路由转发引擎。

**步骤2:将路由信息发送转发引擎**

* 在quagga代码rt\_netlink.c文件netlink\_route\_multipath函数中实现路由信息的发送。
* cmd为24表示插入路由，为25表示删除路由。
* 创建socket套接字，连接转发引擎，发送路由信息。

**步骤3:验证静态路由**

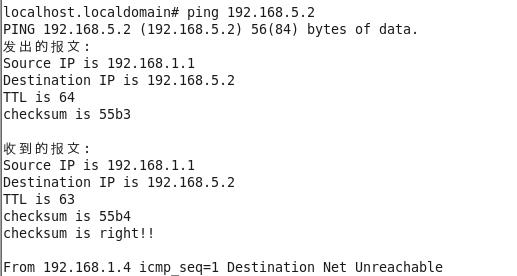
在学生机上配置一条静态路由，测试服务器向学生机发送ping报文，测试学生机的转发是否正常。如图4.2正常转发的现象:

图4.2 转发正常的现象

如图4.3不能正常转发的现象:

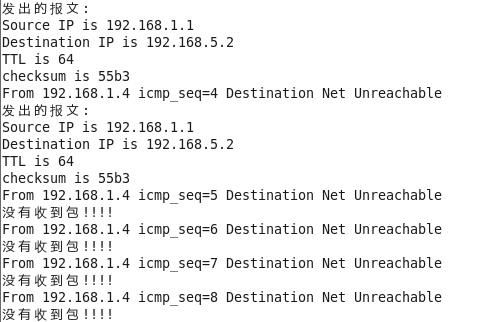


图4.3不能正常转发现象

**步骤4:验证RIP路由**

在学生机和测试服务器建立RIP连接，在测试服务器上配置一条静态路由，重分发此静态路由，学生机会通过RIP学到相应路由，在测试服务器向学生机发送ping报文，测试学生机的转发是否正常。

如图4.4学生机学到的重分发路由:

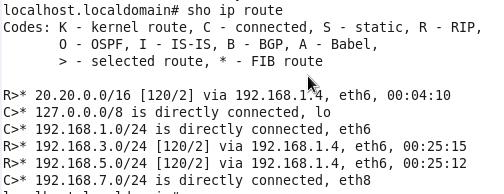


图4.4 RIP重分发路由

转发是否正常借鉴上一个步骤

**2.2实验二：路由管理开发**

**（1）实验目的**

通过实验熟悉路由优先级管理。

**（2）实验原理**

路由管理模块负责统一管理RIP、OSPF、BGP等模块学习到的路由信息，将这些模块学习到的路由信息存储在自己的路由表里，根据路由优先级的高低，选择相同路由信息中优先级最高的插入到转发表中；在向转发表插入表项时，查看是否有相同表项，当有路由表项相同时，原有的表项删除，选择优先级高的表项插入。

**（3）实验环境**

如图4.5所示，学生机通过网络与测试服务器连接；利用学生机上的网卡1和测试服务器相连接，测试服务器上运行路由协议RIP。



图4.5路由管理验证环境

**（3）实验操作**

**步骤1:开发管理模块**

* 在quagga向转发引擎传输路由数据时将所需信息从原有结构体里填充到发送数据里,实现函数位置:quagga代码rt\_netlink.c文件netlink\_route\_multipath函数中；
* 将接收到的路由信息插入转发表，当有相同转发表项时，进行优先级的比较，若原有的表项优先级高，不插入当前表项；若当前表项优先级高，则删除原来的表项，重新插入当前表项；
* 在匹配选中转发表中的表项时，添加打印信息，将路由优先级打印出来。

**步骤2:验证路由优先级管理**

在学生机和测试服务器上建立RIP连接，然后在学生机上配置一条静态路由，这条路由和学生机通过RIP学到的路由一样，测试服务器向学生机发送ping报文，要求将打印的路由优先级变量信息展示。

**2.3实验三：转发性能测试**

**（1）实验环境**

如图4.6所示，学生机通过网络与测试仪连接；利用学生机上的网卡1发送数据包，网口2接收数据包，来测试转发的最大速率。



图4.6转发性能验证环境

**（2）实验操作**

**步骤1:搭建测试环境**

* 学生机和测试服务器上都运行RIP,与测试服务器相连接。
* 学生机上运行转发引擎。
* 学生机上运行客户端程序,连接转发引擎。

**步骤2:向转发引擎发送多条路由信息，进行验证**

* 在测试服务器上随机配置一条静态路由，如图所示，触发读取路由表项，测试服务器会循环读取存储路由表项的文件，下发多条路由表项，学生机通过RIP学到这些路由并下发到转发引擎。

如图4.7测试服务器读取路由表项并下发:

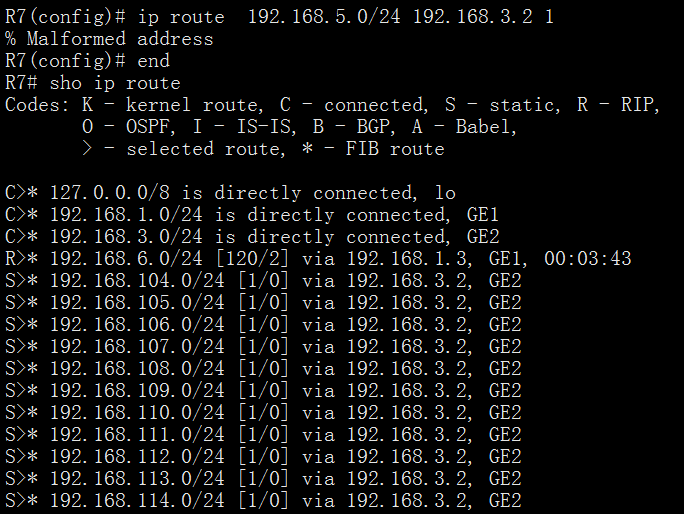


图4.7 下发路由表项

如图4.8学生机通过RIP学到测试服务器发送的多条路由表项：

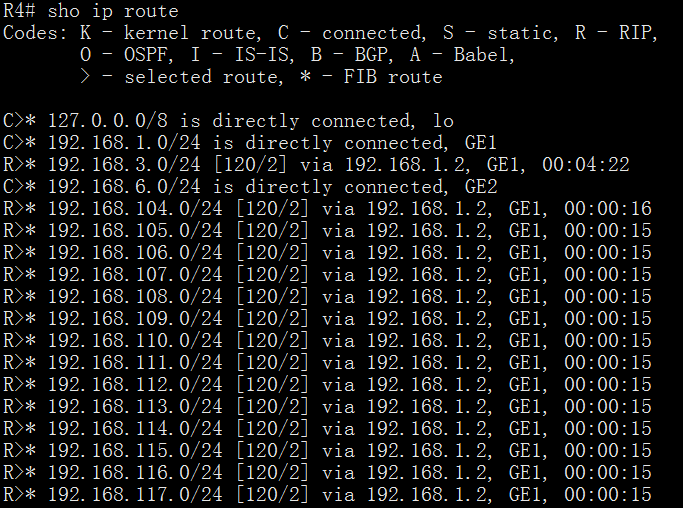


图4.8 学到的路由表项

* 文件中存储的路由表项下一跳都是192.168.3.2，所以要求学生机的网卡2的ip为192.168.3.0/24网段。
* 文件route10中有10条路由，随机一条路由通过测试软件来验证转发速率。
* 文件route100中有100条路由,随机一条路由通过测试软件来验证转发速率。

**步骤3:验证转发速率**

* 通过测试软件设置一个转发速率，查看发送报文和接收报文的个数大小。
* 如果发送报文的个数大小等于接收报文的个数大小，说明可以达到这个转发速率，继续设置更大的转发速率，以此来验证最大的转发速率。

运行测试转发性能程序(sendether.c)的结果:

