路由器实验开发指南

1.开发环境搭建

1.1 安装 python3.7

首先要先安装依赖包：
yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gcc make

找到 python3.7.4 的安装包：
wget https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tgz
接着解压：
tar -zxvf Python-3.7.4.tgz

configure 和 make&&make install:
cd Python-3.7.0
./configure
make&&make install

执行完 make&&make install 之后，可能会出现这种报错：
“ModuleNotFound：No module named '_ctypes'”
这里需要执行：
yum install libffi-devel –y
执行完继续 make&&make install

这样，基本上 python3.7.4 我们就安装完成了，默认情况下，python3.7.4 安装在 /usr/local/bin/，这里为了使默认 python 变成 python3，需要加一条软链接,并把之前的 python 命令改成 python.bak：
mv /usr/bin/python /usr/bin/python.bak
ln -s /usr/local/bin/python3 /usr/bin/python

1.2 安装 pip3

首先安装 setuptools：
wget --no-check-certificate
https://pypi.python.org/packages/source/s/setuptools/setuptools-19.6.tar.gz

tar -zxvf setuptools-19.6.tar.gz

cd setuptools-19.6.tar.gz
python setup.py build
python setup.py install

然后安装 pip3:
wget --no-check-certificate
https://pypi.python.org/packages/source/p/pip/pip-8.0.2.tar.gz

tar -zxvf pip-8.0.2.tar.gz
cd pip-8.0.2
python setup.py build
python setup.py install

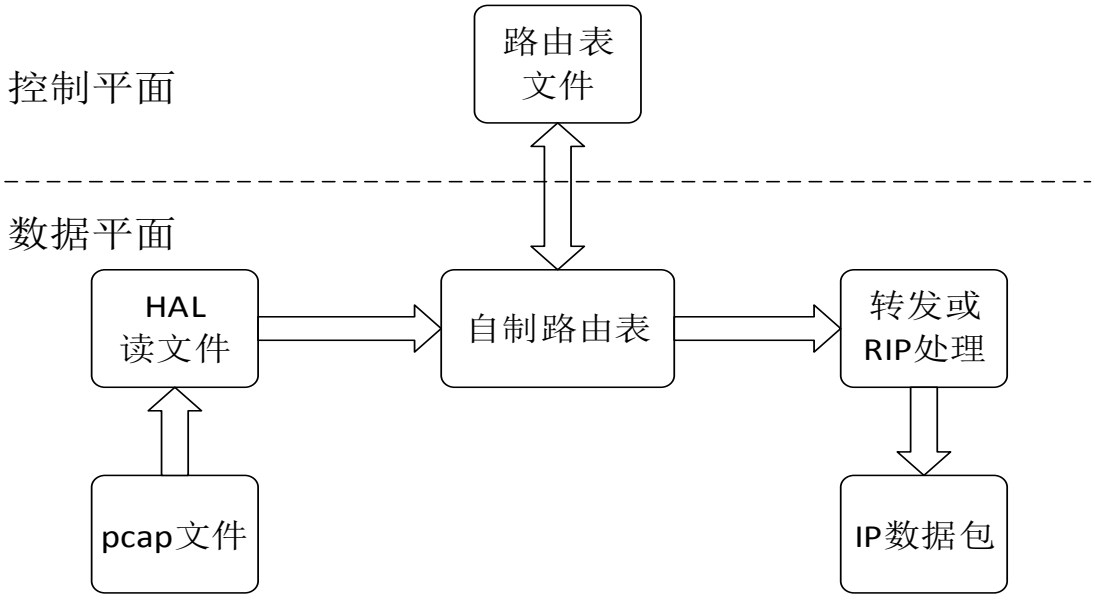
2.开发总体说明

2.1 开发功能：　转发引擎

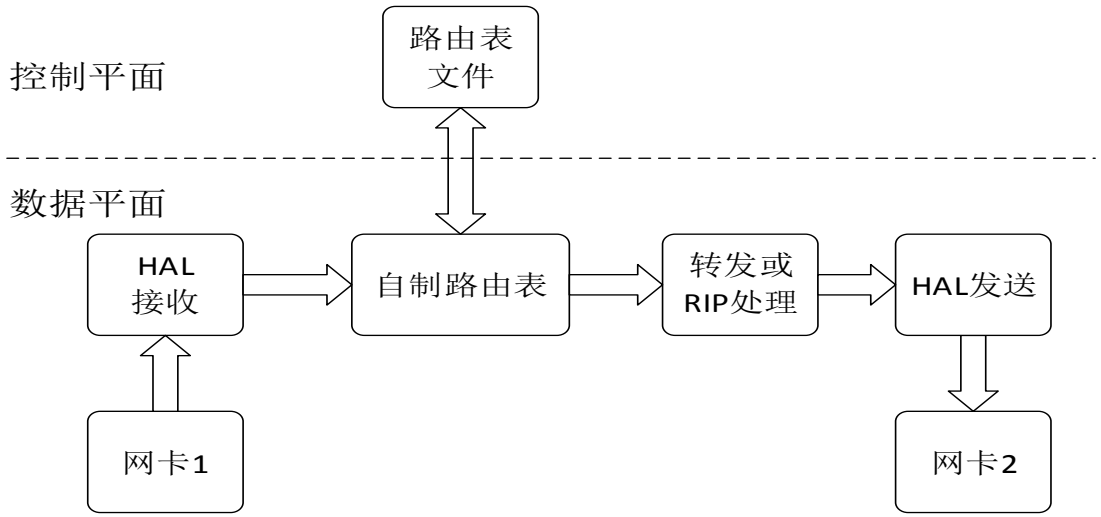
RIP 路由协议

2.2 开发框架：

基于 HAL 读写文件形式



基于 HAL 真实网络形式



2.3 框架使用：　　见 5.框架使用说明

3.转发引擎的开发

3.1 转发流程图

如图 3.1 所示，基于 HAL 读写文件形式的转发流程，红圈标注部分需要依次完成，具体操作见实验操作 3.2.1。

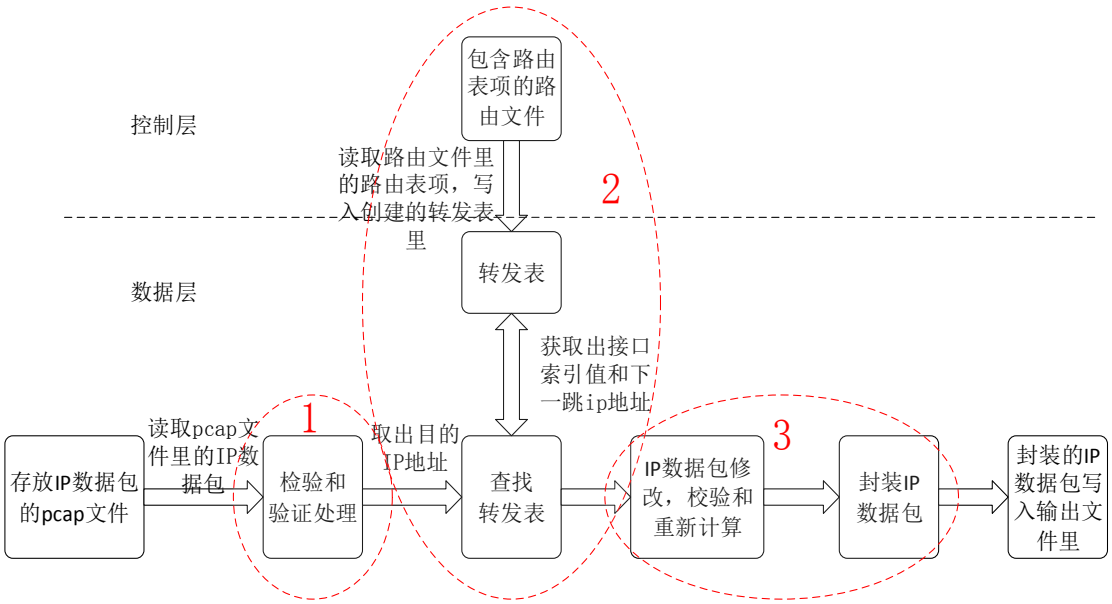


图 3.1 基于 HAL 读写文件的转发流程图

如图 3.2 所示，基于 HAL 真实网络形式的转发流程，红圈标注部分需要依次完成，具体操作见实验操作 3.2.2。

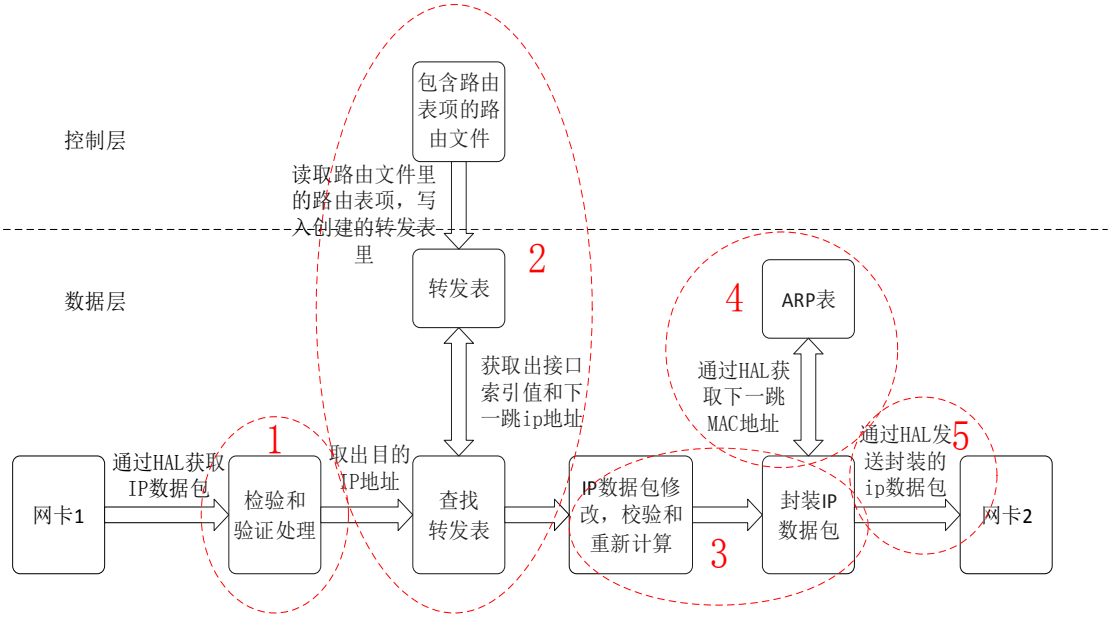


图 3.2 基于 HAL 真实网络的转发流程图

3.2 实验操作

3.2.1 基于 HAL 读写文件形式的转发

步骤一. 校验和验证：

需要完成部分：

填写 validateIPChecksum 函数，对 IP 数据包的 IP 首部依次进行校验和验证，正确的返回 ture,错误返回 false；

说明：

程序框架的 HAL_ReceiveIPPacket 函数会依次读取 pcap 文件里的 IP 数据包，框架会根据 validateIPChecksum 函数返回的 true 或者 false 往输出文件写验证结果。

测试说明：

输入：包含多条任意报文的 pcap 文件

输出：Yes/No

步骤二. 查表验证：

需要完成部分：

<1>.获取 IP 数据包的目的 ip 地址；

<2>.创建转发表：读取路由表项文件，将路由表项写入自己创建的转发表，转发表的数据结构选用 trie 树、链表、哈希等其中之一；

<3>.根据目的 ip 地址查找转发表，获取正确的下一跳信息；

<4>.将找到的下一跳信息调用 printf 打印出来（打印顺序依次是目的 ip 网段、掩码长度、出接口索引值、下一跳 ip 地址）；

注意：

路由表项文件存储形式如下（依次是目的 ip 网段、掩码长度、出接口索引值、下一跳 ip 地址）：

0x4a8c0,24,3,0x203a8c0

0x6a8c0,24,4,0x303a8c0

说明：

程序框架会将打印的信息写入输出文件里。

测试说明：

输入：包含多条 IP 报文的 pcap 文件、包含路由表项的路由表文件

输出：匹配到的路由信息

步骤三. 重新封装报文阶段验证：

需要完成部分：

<1>.将 IP 数据包的 IP 首部 TTL 值减一，重新计算校验和；

<2>.重新封装 IP 数据包；

<3>.调用 printf 将重新封装的 IP 数据包打印出来；

说明：

程序框架会将打印的信息写入输出文件里。

测试说明：

输入：包含多条 IP 报文的 pcap 文件、包含路由表项的路由表文件

输出：转发处理完后重新封装的 IP 数据包

3.2.2 基于 HAL 真实网络形式的转发

需要完成部分：

<1>. 填写 validateIPChecksum 函数，对 IP 数据包的 IP 首部依次进行校验和验证，正确的返回 ture,错误的返回 false，主函数根据 validateIPChecksum 函数返回作出相应处理，返回 true 则继续,返回 false 则丢弃数据包；

<2>.

①获取 IP 数据包的目的 ip 地址；

②创建转发表，读取路由表项文件，将路由表项写入自己创建的转发表，转发表的数据结构选用 trie 树、链表、哈希等其中之一；

③根据目的 ip 地址查找转发表，获取正确的下一跳信息；

<3>.

①将 IP 数据包的 IP 首部 TTL 值减一，重新计算校验和

②重新封装 IP 数据包；

<4>.根据查找到的下一跳信息调用 HAL_ArpGetMacAddress 函数获取下一跳的 MAC 地址；

<5>.调用 HAL_SendIPPacket 函数将重新封装的 IP 数据包发送出去。

4.RIP 路由协议的开发

4.1 RIP 简介

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。

路由设备启动 RIP 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，同时对端路由也会发送 RIP 请求报文。路由设备在接收到 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。

4.2 RIP 报文格式

RIP 报文格式如下：报文头部是 4 个字节，包括了“命令”字段 1 个字节、“版本”字段 1 个字节、未使用 2 个字节。路由表项是 20 个字节，其中地址族标识占 2 个字节，路由标记占 2 个字节，IP 地址、子网掩码、下一跳、度量值各自占 4 个字节。一个完整的 RIP 报文可以最多 25 个路由表项，报文的总长度不能超过 udp 报文长度。详见表 4.1。

命令 command(1)	版本 version(1)	未使用设置 0（2）
地址族标识 AF(2)		路由标记 Tag(2)
ip 地址(4)		
子网掩码(4)		
下一跳(4)		

度量值 metric(4)

表 4.1 RIP 报文格式

command:定义了报文类型(response 或者 request)；

version:RIP 报文版本，默认情况下为 2；

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；

IP 地址:目标地址 IP。

子网掩码:用于区分子网；

Next Hop:下一跳地址；

metric:路由跳数，请求完整的 RIP 报文时该值为 16；

4.3 流程图

如图 4.1 所示，基于 HAL 读写文件形式的 RIP 协议，红圈标注部分需要依次完成，具体操作见实验操作 4.4.1。

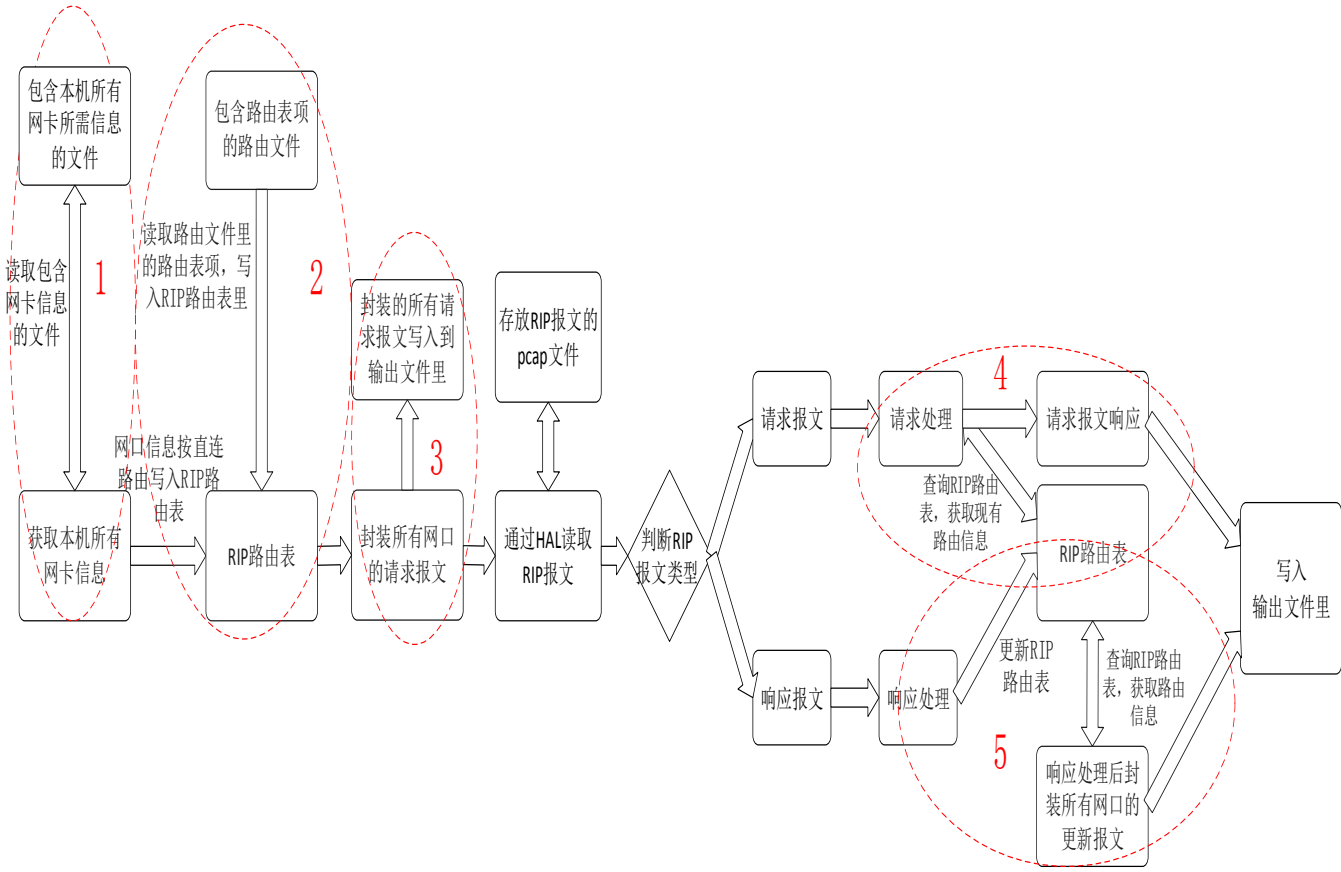


图 4.1 基于 HAL 读写文件的 RIP 协议

如图 4.2 所示，基于 HAL 真实网络形式的 RIP 协议，红圈标注部分需要依次完成，具体操作见实验操作 4.4.2。

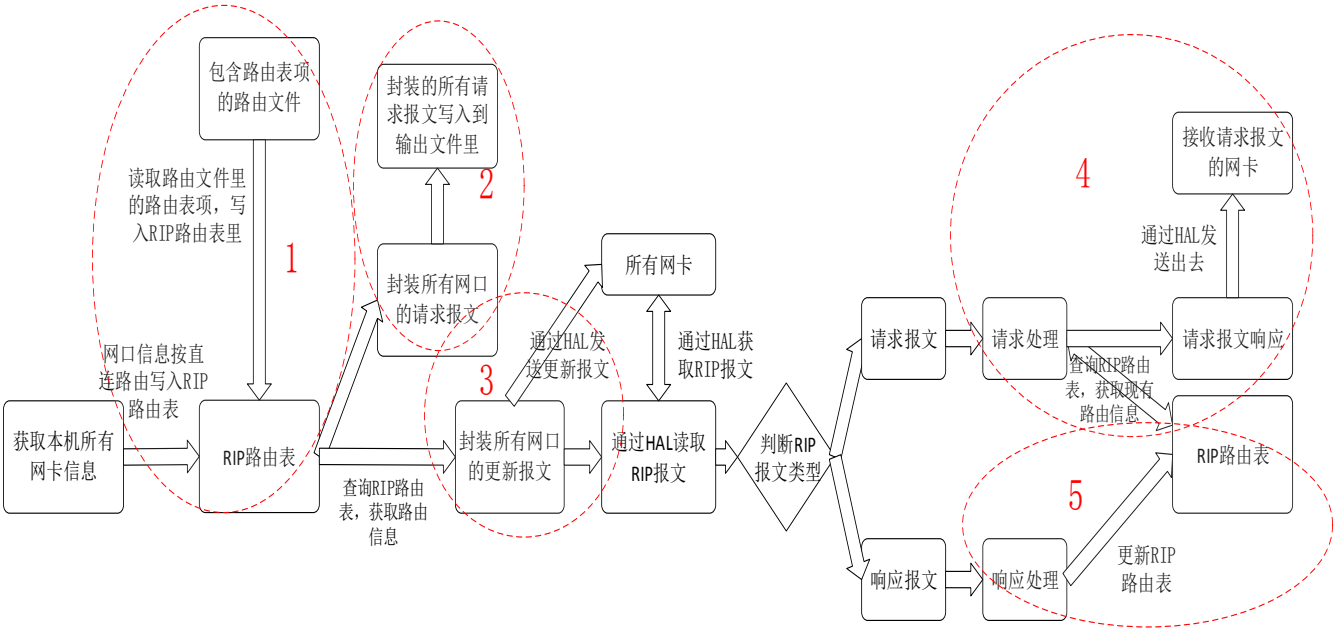


图 4.2 基于 HAL 真实网络的 RIP 协议

4.4 实验操作

4.4.1 基于 HAL 读写文件形式的 RIP 协议（需要完成部分）

<1>.获取本机网卡信息：

读取包含本机网卡信息的文件，将信息对于存储起来，方便后续使用。

说明：

包含本机网卡信息的文件格式：　eth0，192.168.1.2

eth1，192.168.3.1

<2>.创建 RIP 路由表：

①读取路由表项文件，将路由表项写入自己创建的 RIP 路由表；

② 将本机网卡信息按直连路由写入到 RIP 路由表。

<3>.请求报文封装：

① 封装 IP 头：源 ip 为本机网口 ip，目的 ip 为组播 ip 地址 224.0.0.9；

② 封装 UDP 头：源、目的端口都为 520；

③封装 RIP 请求，command 字段为 1，version 字段等于 2，其他字段等于 0，度量值字段等于 16；

④将封装的所有请求报文 printf 打印出来。

说明：

程序框架会将打印的信息写入输出文件里。

<4>.请求报文处理：

①检验是否为请求报文，判断度量值是否为 16，地址族标识是否为 0；

②获取请求报文的源 ip 地址；

③遍历 RIP 路由表，封装所有和源 ip 地址不在同一网段的路由表项到 RIP 报文里，填充 RIP 报文，command 为字段 2；

④封装成 UDP 的 IP 数据包，源 ip 为本机接收请求报文网口的 ip 地址，目的 ip 为请求报文的源 ip，然后 printf 打印出来。

说明：

程序框架会将打印的信息写入输出文件里。

<5>.响应报文处理（每一次处理完后重新遍历封装更新报文 printf 打印是为了验证处理是否正确）：

①按照 RIP 表项的数目对 RIP 报文进行依次处理；

②如果度量值字段加 1 后大于 16，表示删除这条路由，遍历 rip 路由表，删除和 RIP 报文对应的路由，同时将失效（删除的）路由封装成失效报文，转发出去（接收失效路由的网口不转发），这里只需封装失效报文 printf 打印；

③对比 rip 路由表，没有相同的路由则直接把这条 RIP 报文里的路由插入 rip 路由表；

④对比 rip 路由表，有相同的路由，如果 RIP 报文里路由的度量值加 1 后小于或等于 rip 路由表里的度量值，更新 rip 路由的原来度量值为 RIP 报文里路由的度量值加 1；若大于的则忽略。

⑤每一次处理完一条 RIP 响应报文后，重新遍历 RIP 路由表，将自己的所有路由封装到更新报文里，从所有网口转发出去，这里只需要将封装的更新报文 printf 打印。

说明：

程序框架会将打印的信息写入输出文件里。

4.4.2 基于 HAL 真实网络形式的 RIP 协议

<1>.创建 RIP 路由表：

①读取路由表项文件，将路由表项写入自己创建的 RIP 路由表；

② 将本机网卡信息按直连路由写入到 RIP 路由表。

<2>.请求报文封装及发送：

① 封装 IP 头：源 ip 为现有本机网口 ip，目的 ip 为组播 ip 地址 224.0.0.9；

② 封装 UDP 头：源、目的端口都为 520；

③封装 RIP 请求，command 字段为 1，version 字段等于 2，其他字段等于 0，度量值字段等于 16；

④将封装的所有网口请求报文调用 HAL_SendIPPacket 函数从各自网口发送出去。

<3>.更新报文封装及发送（更新报文与接受 RIP 报文放在同一循环里）：

① 封装 IP 头：源 ip 为现有本机网口 ip，目的 ip 为组播 ip 地址 224.0.0.9；

② 封装 UDP 头：源、目的端口都为 520；

③ 封装更新 RIP 路由条目，遍历 rip 路由表，将除了与源 ip 在同一网段和下一跳为源 ip 地址的路由都封装到路由条目里；

④调用 HAL_SendIPPacket 函数从所有网口转发出去。

<4>.请求报文处理：

①检验是否为请求报文，判断度量值是否为 16，地址族标识是否为 0；

②获取请求报文的源 ip 地址；

③遍历 RIP 路由表，封装所有和源 ip 地址不在同一网段的路由表项到 RIP 报文里，填充 Rip 报文，command 为字段 2；

④封装成 UDP 的 IP 数据包，源 ip 为本机接收请求报文网口的 ip 地址，目的 ip 为请求报文的源 ip，调用 HAL_SendIPPacket 函数从接收请求报文的网口发送出去。

<5>.响应报文处理：

①按照 RIP 表项的数目对 RIP 报文进行依次处理；

②如果度量值字段加 1 后大于 16，表示删除这条路由，遍历 rip 路由表，删除和 RIP 报文对应的路由，同时将失效（删除的）路由封装成失效报文，调用 HAL_SendIPPacket 函数从所有网口发送出去（接收失效报文的网口不发送）；

③对比 rip 路由表，没有相同的路由则直接把这条 RIP 报文里的路由插入 rip 路由表；

④对比 rip 路由表，有相同的路由，如果 RIP 报文里路由的度量值加 1 后小于或等于 rip 路由表里的度量值，更新 rip 路由的原来度量值为 RIP 报文里路由的度量值加 1；若大于的则忽略。

5.HAL 框架使用说明及自测

5.1 HAL 框架使用

5.1.1 基于 HAL 读写文件形式：

进入 Homework 目录下，是需要测各项功能的文件，eg：checksum，进入文件里，make 编译，执行./grade.py 即可自测；

5.1.2 基于 HAL 真实网络形式：

进入测试功能文件里后，需要修改 Makefile 里 hal.o 生成的依赖，将 stdio 改成 linux，然后 make 编译，执行./grade.py 即可自测。

5.2 自测指南

5.2.1 基于 HAL 读写文件形式

说明：

data 文件里的 input1 为输入文件，output1 为标准文件，程序输出后会生成输出文件 user1；

各个验证出错的话，程序框架会有错误提示，会告知文件第几个数据包处理错误，错误在哪，正确是什么，根据提示修改错误。

Grade 后面标注是否通过，正确是 Grade: 1/1，错误是 Grade: 0/1。

<1>.校验和验证自测结果

如图 5.1 所示，正确结果

```
[root@localhost 1checksum]# ./grade.py
Removing all output files
Running './checksum < data/checksum_input1.pcap > data/checksum_user1.out'
Grade: 1/1
[root@localhost 1checksum]#
```

图 5.1 校验和验证自测正确结果

如图 5.2 所示，有错误结果

```
[root@localhost 1checksum]# ./grade.py
Removing all output files
Running './checksum < data/checksum_input1.pcap > data/checksum_user1.out'
Wrong Answer:
Answer is wrong for packet #3: Layer IP:
        Version: 4
        Header length: 20 bytes
        Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
        Total Length: 32
        Identification: 0x0000 (0)
        Flags: 0x02 (Don't Fragment)
        0... .... = Reserved bit: Not set
        .1.. .... = Don't fragment: Set
        ..0. .... = More fragments: Not set
        Fragment offset: 0
        Time to live: 64
        Protocol: UDP (17)
        Header checksum: 0x1ec4 [validation disabled]
        Good: False
        Bad: False
        Source: 183.173.113.183 (183.173.113.183)
        Destination: 1.2.3.4 (1.2.3.4)

Grade: 0/1
[root@localhost 1checksum]#
```

图 5.2 校验和验证自测有错误结果

<2>.查表验证

```
[root@localhost PcapRouting]# ./grade.py
Removing all output files
Running './Routing < data/Routing_input1.pcap > data/Routing_user1.out'
-------------------------Wrong Answer:-------------------------
Answer is wrong for Route Number # 1
IpPrefix is wrong

Answer is wrong for Route Number # 2
PrefixLen is wrong

Answer is wrong for Route Number # 2
Route format is wrong

Answer is wrong for Route Number # 3
Ifindex is wrong

Answer is wrong for Route Number # 4
Nexthop is wrong

-----------------------------end---------------------------------

Grade: 0/1
[root@localhost PcapRouting]#
```

图 5.3 查表验证自测有错误结果

<3>.封装 IP 数据包自测

```
[root@localhost Pcapforward]# ./grade.py
Removing all output files
Running './Routing < data/Routing_input1.pcap > data/Routing_user1.out'

--------------------Wrong Answer--------------------:
Answer is wrong for packet # 2
IP header is wrong
IP header wrong member is TTL
Your TTL is 3f          The true TTL is 36

Answer is wrong for packet # 4
IP header is wrong
IP header wrong member is Header Length
Your Header Length is 5           The true Header Length is 4

Answer is wrong for packet # 7
IP header is wrong
IP header wrong member is Checksum
Your Checksum is 0e63          The true Checksum is 0e73

Answer is wrong for packet # 9
IP header is wrong
IP header wrong member is Source IP
Your Source IP is b7ad71b7          The true Source IP is b7ad78b7

-----------------one packet end-----------------------

Running './Routing < data/Routing_input2.pcap > data/Routing_user2.out'
Running './Routing < data/Routing_input3.pcap > data/Routing_user3.out'
Grade: 2/3
[root@localhost Pcapforward]#
```

图 5.4 封装 IP 数据包出错结果

<4>. RIP 路由协议自测

如图 5.5、5.6 所示，RIP 路由协议出错结果

```
[root@localhost PcapRip]# ./grade.py
Removing all output files
Rip './Rip < data/Rip_input1.pcap > data/Rip_user1.out'

---------------------------------Wrong Answer---------------------------------:
----Answer is wrong for packet--- # 1
Routing Info is wrong
Routing Info is number # 1
Routing Info wrong member is Nexthop
Your Nexthop is 00000000          The true Nexthop is 60000000

----Answer is wrong for packet--- # 2
Routing Info is wrong
Routing Info is number # 1
Routing Info wrong member is IpPrefix
Your IpPrefix is 00000000         The true IpPrefix is 07000000

----Answer is wrong for packet--- # 3
Routing Info is wrong
Routing Info is number # 1
Routing Info wrong member is IpPrefix
Your IpPrefix is c0a80300         The true IpPrefix is c0180300

Routing Info is wrong
Routing Info is number # 1
Routing Info wrong member is Netmask
Your Netmask is ffffff00          The true Netmask is ffff7f00

Routing Info is wrong
Routing Info is number # 1
Routing Info wrong member is Nexthop
Your Nexthop is 00000000          The true Nexthop is 00000003

----Answer is wrong for packet--- # 4
```

图 5.5 RIP 路由协议自测出错结果

```
Routing Info wrong member is Nexthop
Your Nexthop is 00000000          The true Nexthop is 00000003

----Answer is wrong for packet--- # 4
Routing Info is wrong
Routing Info is number # 1
Routing Info wrong member is Tag
Your Tag is 0000         The true Tag is 0040

Routing Info is wrong
Routing Info is number # 2
Routing Info wrong member is IpPrefix
Your IpPrefix is c0a80500         The true IpPrefix is c0180500

Routing Info is wrong
Routing Info is number # 2
Routing Info wrong member is Netmask
Your Netmask is ffffff00          The true Netmask is fff6ff00

Routing Info is wrong
Routing Info is number # 3
Routing Info wrong member is Nexthop
Your Nexthop is c0a80101          The true Nexthop is c0a70101

Routing Info is wrong
Routing Info is number # 4
Routing Info wrong member is Metric
Your Metric is 00000002           The true Metric is 00000004

------------------------------------one packet end-----------------------------

Rip './Rip < data/Rip_input2.pcap > data/Rip_user2.out'
Rip './Rip < data/Rip_input3.pcap > data/Rip_user3.out'
Grade: 2/3
[root@localhost PcapRip]#
```

图 5.6 RIP 路由协议自测出错结果

说明：

data 文件里的 input1 为输入文件，output1 为标准文件，程序输出后会生成输出文件 user1；

RIP 报文封装出错的话，程序框架会有错误提示，会告知文件里的第几个 RIP 报文封装错误，错误在哪，正确是什么，根据提示修改错误。

6.测试说明

如图 6.1 所示，学生提交程序代码到测试服务器，测试服务程序将报文写入到文件里，以便完成测试验证。

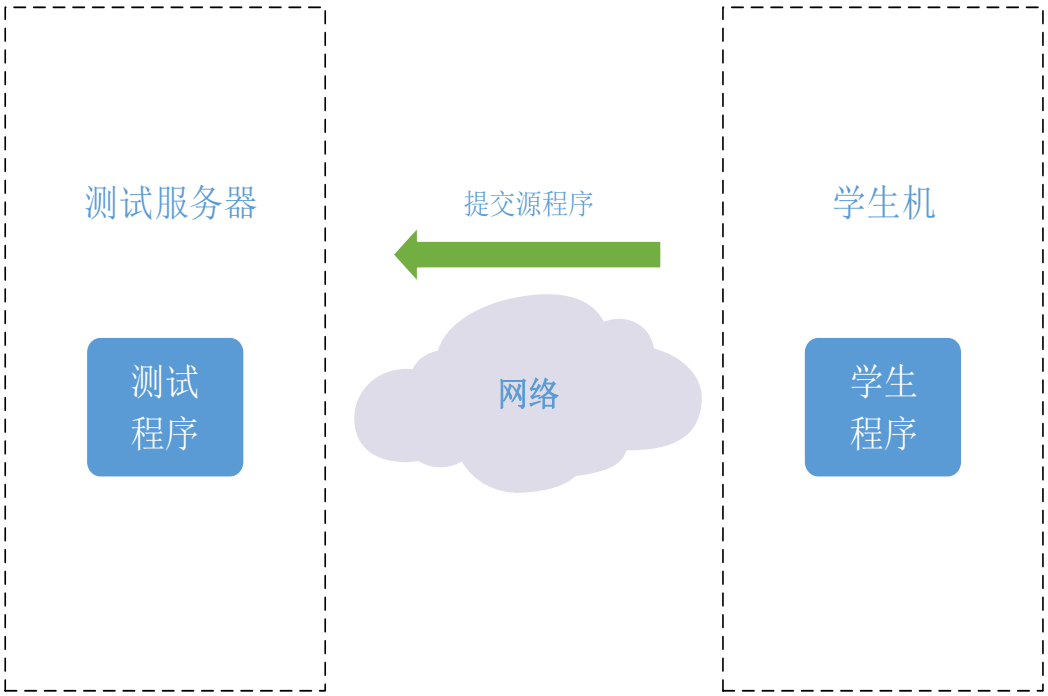


图 6.1

如图 6.2 所示，测试服务器会针对学生转发引擎的各个阶段（Checksum、Routing、Datagram）进行验证，学生程序需要对各个阶段做相应处理，最后将处理后的结果 printf 写入相应文件，测试服务程序读取文件中的报文，与标准的处理结果进行比对，对学生代码各个阶段进行验证并给出结果。

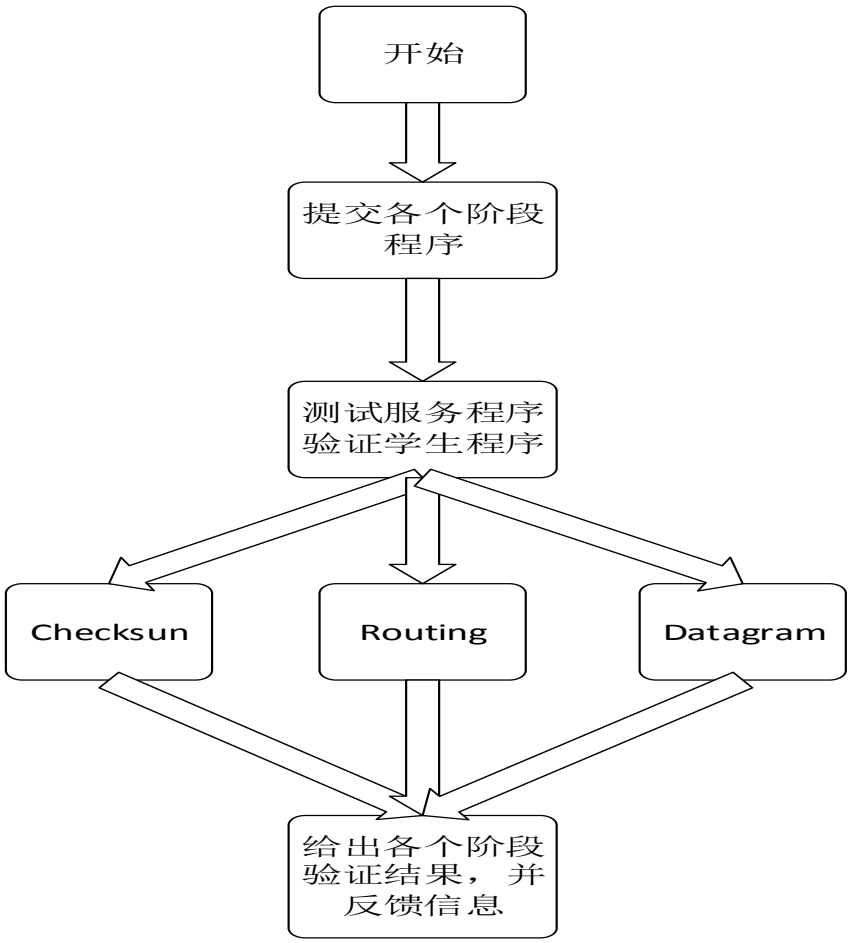


图 6.2 转发引擎

如图 6.3 所示，测试服务器会针对学生 RIP 协议进行测试，分别验证请求报文封装、请求报文处理、响应报文处理、更新报文，学生处理后将封装的 RIP 报文 printf 写入相应文件，测试服务程序读取文件中的报文，与标准的处理结果进行比对，对学生代码各个阶段进行验证并给出结果。

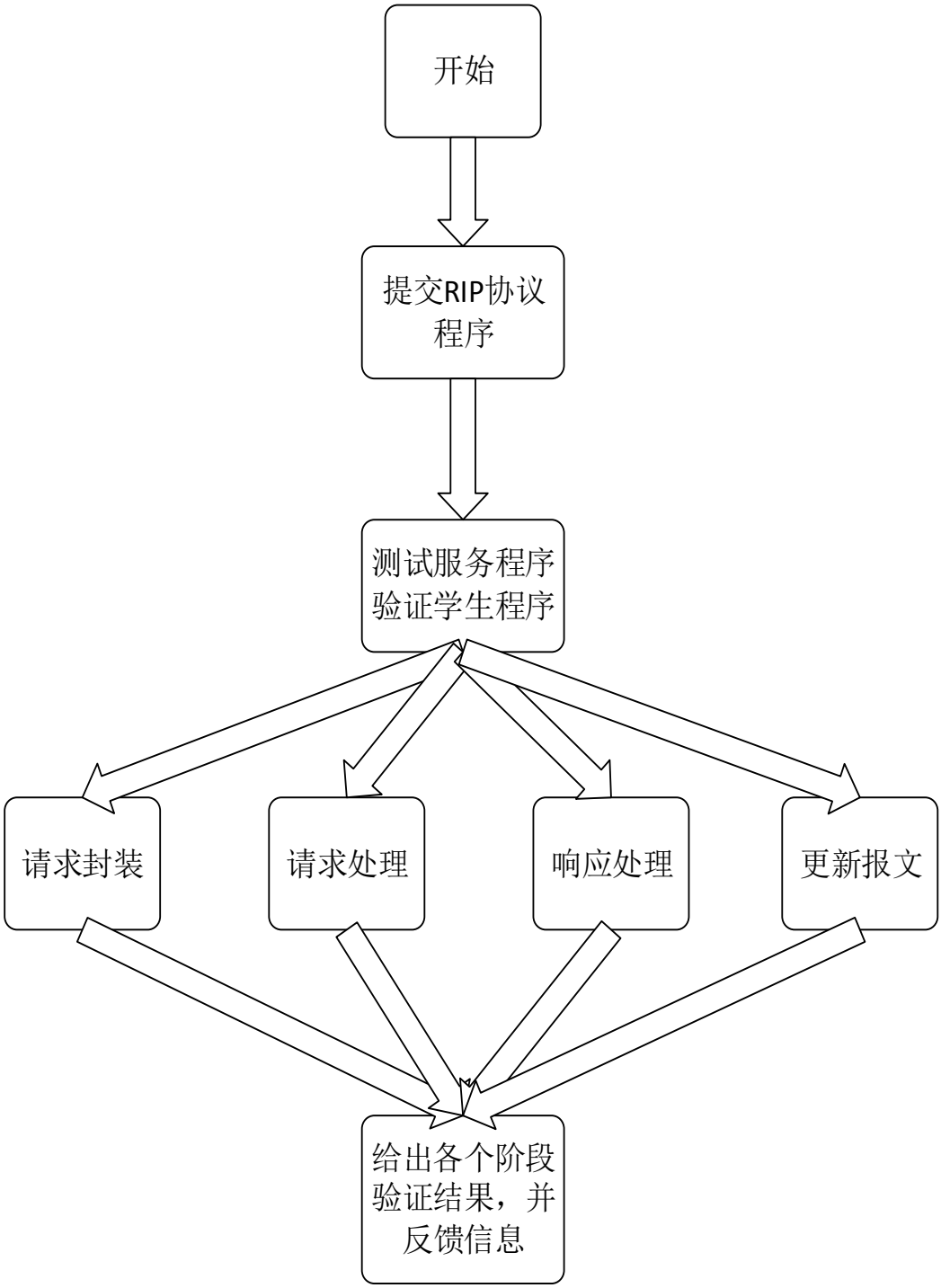

开始
提交RIP协议程序
测试服务程序验证学生程序
请求封装
请求处理
响应处理
更新报文
给出各个阶段验证结果，并反馈信息

图 6.3 RIP 协议

7.综合实验（转发引擎和 RIP 路由协议）

7.1 综合实验流程图

如图 7.1 所示，基于 HAL 读写文件形式的转发流程

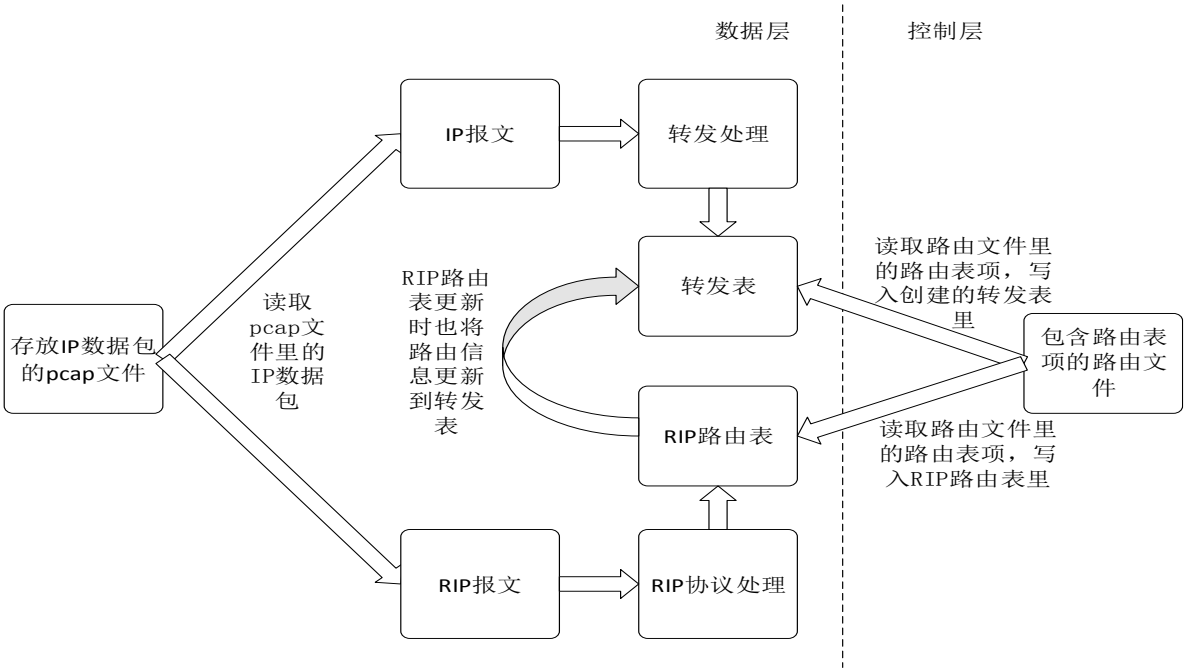


图 7.1 基于 HAL 读写文件的综合实验

如图 7.2 所示，基于 HAL 真实网络形式的转发流程

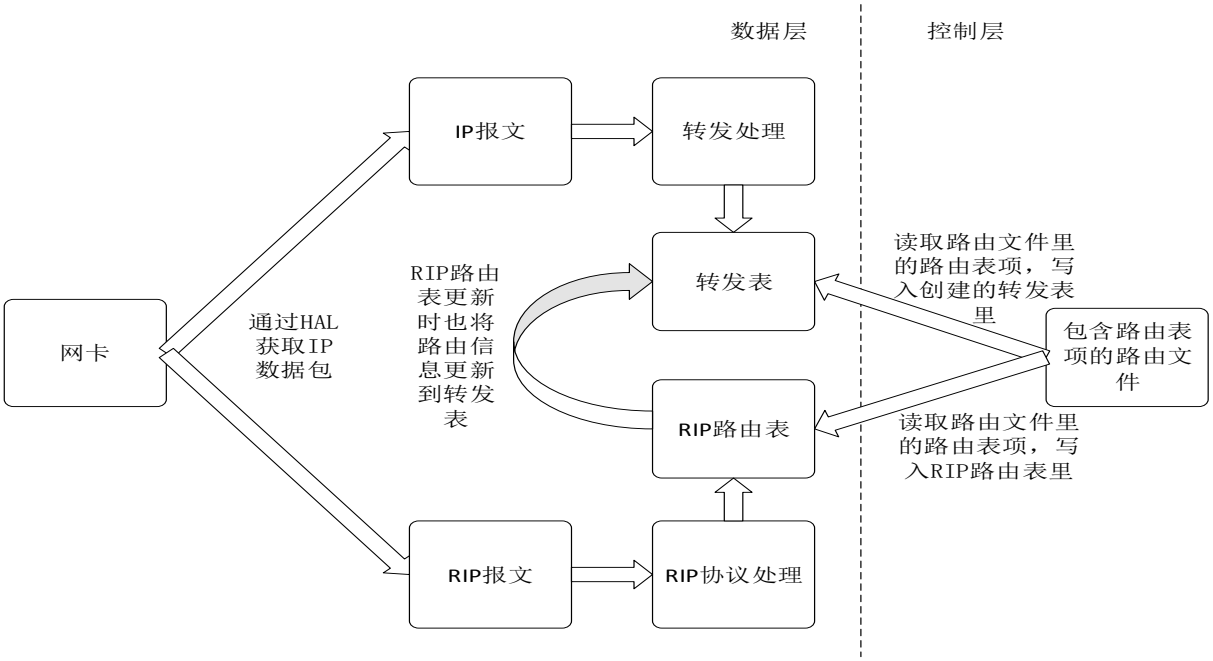


图 7.2 基于 HAL 真实网络的综合实验

7.2 实验操作

<1>.创建转发表和 RIP 路由表，读取路由文件里的路由表项，分别写入两个表里；

<2>.读取 IP 数据包，先判断校验和是否正确，正确则继续，错误则丢弃；

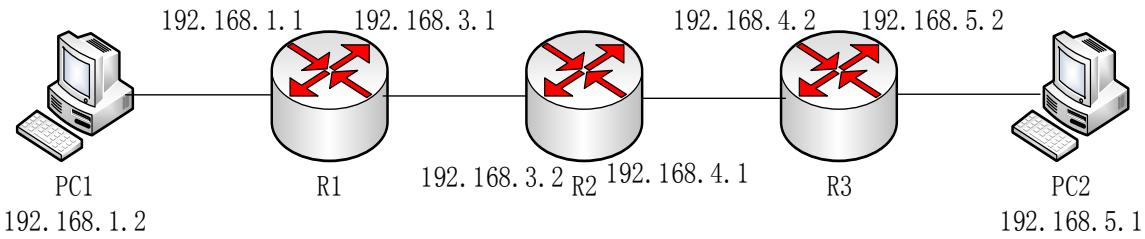
<3>.判断是转发报文还是 RIP 报文；

①如果是转发报文，则进行转发处理；

②如果是 RIP 报文，则进行 RIP 协议处理，如果是响应报文更新 RIP 路由表时，填加、删除、更新都要同步更新到转发表里；

7.3 组网

7.3.1 基于 HAL 真实网络的组网拓扑图



7.3.2 基于 HAL 真实网络实验操作

<1>.RIP 建联

按照网络拓扑图搭建环境，使 3 台路由之间能够正确学到 RIP 路由，通过抓取 RIP 报文来分析路由是否正确。

<2>.RIP、转发引擎的集成

按照网络拓扑图搭建环境，使 3 台路由能正常交互，将学到的 RIP 路由正常同步添加、删除、更新到转发表。

<3>.组网成功现象

3 台路由各自启动，等待相互学习过程；PC1 ping PC2，可以 ping 通，反过来 PC2 也可以 ping 通 PC1：

```
[root@localhost ~]# ping  192.168.5.1
PING 192.168.5.1 (192.168.5.1) 56(84) bytes of data.
64 bytes from 192.168.5.1: icmp_seq=1 ttl=61 time=11.3 ms
64 bytes from 192.168.5.1: icmp_seq=2 ttl=61 time=10.6 ms
64 bytes from 192.168.5.1: icmp_seq=3 ttl=61 time=9.84 ms
64 bytes from 192.168.5.1: icmp_seq=4 ttl=61 time=11.1 ms
64 bytes from 192.168.5.1: icmp_seq=5 ttl=61 time=11.6 ms
^C
--- 192.168.5.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4374ms
rtt min/avg/max/mdev = 9.843/10.941/11.641/0.645 ms


[root@localhost ~]# ping  192.168.1.2
PING 192.168.1.2 (192.168.5.1) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=61 time=10.5 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=61 time=10.2 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=61 time=11.1 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=61 time=9.73 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=61 time=10.7 ms
^C
--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4239ms
rtt min/avg/max/mdev = 9.73/10.446/11.1/0.672 ms
```