路由器转发实验报告

李昊宸

2017K8009929044

（一）路由器转发机制实现

一、实验内容

1. 运行给定网络拓扑(router\_topo.py), 在h1上进行ping实验

Ping 10.0.1.1 (r1)，能够ping通

Ping 10.0.2.22 (h2)，能够ping通

Ping 10.0.3.33 (h3)，能够ping通

Ping 10.0.3.11，返回ICMP Destination Host Unreachable

Ping 10.0.4.1，返回ICMP Destination Net Unreachable

2.

1)构造一个包含多个路由器节点组成的网络

手动配置每个路由器节点的路由表

有两个终端节点，通过路由器节点相连，两节点之间的跳数不少于3跳，手动配置其默认路由表

2)连通性测试

终端节点ping每个路由器节点的入端口IP地址，能够ping通

3)路径测试

在一个终端节点上traceroute另一节点，能够正确输出路径上每个节点的IP信息

二、实验流程

1. 搭建实验环境

include：相关头文件

scripts：禁止协议栈的数据包处理

main.c： 路由器的代码实现，编译后在路由器结点上运行

arp.c: 发送ARP请求和应答

arpcache.c：ARP缓存相关操作

icmp.c：发送ICMP数据包

ip\_base.c：前缀查找和发送IP数据包

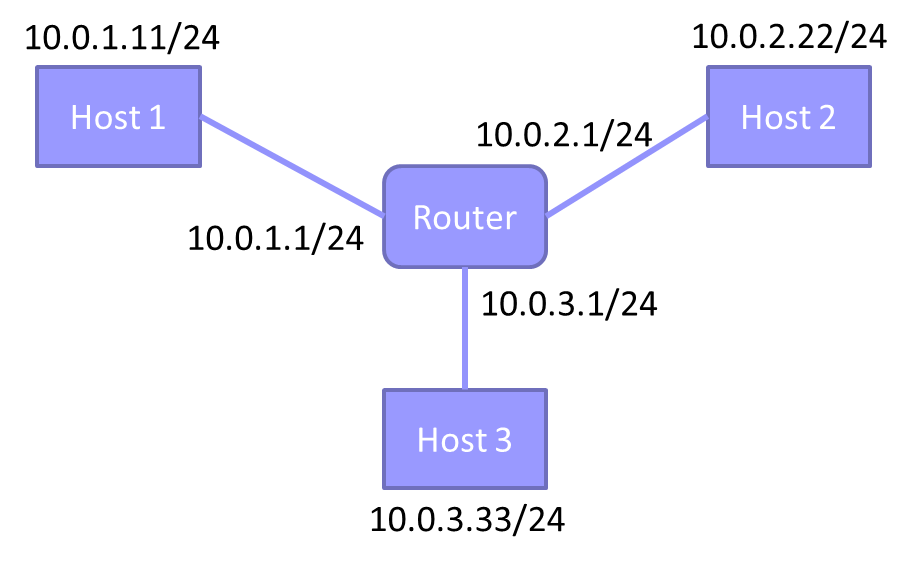
packet.c：发送数据包函数

rtable.c：路由表相关

rtable\_internal.c：从协议栈中读取路由条目

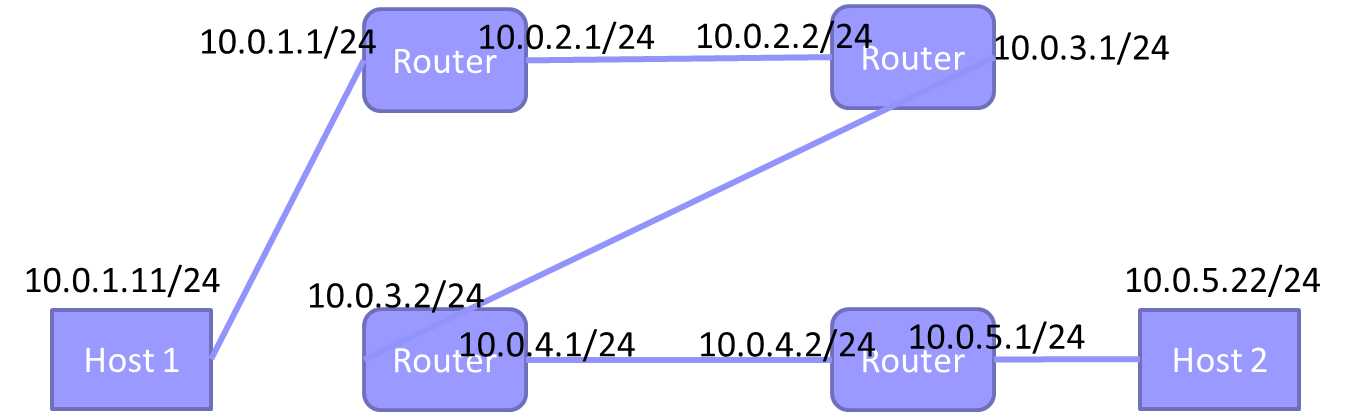
ip.c：处理IP数据包，包括转发

router\_topo.py：实现如下图的四节点拓扑



图一 四节点网络拓扑

my\_topo.py：实现如下图的六节点拓扑



图二 六节点网络拓扑

路由器需要维护两个表：路由表和ARP缓存

路由表：routing table，或称路由择域信息库（RIB, Routing Information Base），是一个存储在路由器或者联网计算机中的电子表格（文件）或类数据库。路由表存储着指向特定网络地址的路径（在有些情况下，还记录有路径的路由度量值）。路由表中含有网络周边的拓扑信息。

ARP高速缓存：ARPcache，由最近的ARP项组成的内在中的一个临时表每个主机或者路由器都有一个ARP高速缓存表。它用来存放最近Internet地址到硬件地址之间的映射记录。高速缓存表中每一项的生存时间都是有限的，起始时间从被创建时开始计算的。

路由器路由流程：

1.路由器某端口收到一个包后，解析该包以太网首部的协议类型，如果是IP，就跳转到2；如果是ARP，就跳转到14。

2.收到的包是IP，调用对IP包的解析过程，转到3。

3.首先解析IP首部的目的IP地址，如果等于本端口IP，转到4；否则，转到5。

4.目的IP等于本端口IP，说明该包要与本端口进行交互。解析该包的ICMP首部类型。如果类型是ping，就调用ICMP响应函数对源主机进行响应；否则将该包丢弃。

5.目的IP不等于本端口IP，说明要转发该包，转到6。

6.在路由表中使用最长前缀匹配查找目的IP。如果找到对应表项，转到7；否则调用ICMP响应函数回复源主机网络不可达。

7.找到对应表项，先将IP首部的ttl减1。如果减1后等于0，说明生存时间耗尽，调用ICMP响应函数回复源主机生存期耗尽；否则转到8。

8.重新计算校验和并写入IP首部，转到9。

9.查看表项中记录的下一跳网关IP。如果为全0，说明该表项记录的端口与目的IP在同一个网段内，转10；否则说明该路由器任何一个端口都不与目的IP在同一网段，该包需要被转发到下一跳网关，转到12。

10.修改以太网首部中源mac地址为端口mac地址。在ARP缓存中查找目的IP对应的表项，如果找到，就将表项记录的映射mac地址填写到以太网首部中目的mac地址，然后从该端口将包发送出去；否则，转11。

11.该目的IP到mac地址的映射未被保存在ARP缓存中，但该IP与端口处于同一个网段，就将该包挂起，并从该端口发送ARP请求。每隔1s，如果没有收到ARP应答，就重发1次。如果超过5次仍没有收到应答，就将该包丢弃，并调用ICMP响应函数，从收到该包的端口回应源主机，目的主机不可达。如果收到ARP应答，就准备好所有因等待该ARP应答而挂起的包，将应答中的映射mac地址填写到以太网首部中目的mac地址，然后从该端口将包发送出去。

12.修改以太网首部中源mac地址为端口mac地址。在ARP缓存中查找下一跳网关IP对应的表项，如果找到，就将表项记录的映射mac地址填写到以太网首部中目的mac地址，然后从该端口将包发送出去；否则，转13。

13. 下一跳网关IP到mac地址的映射未被保存在ARP缓存中，但该IP与端口处于同一个网段，就将该包挂起，并从该端口发送ARP请求。每隔1s，如果没有收到ARP应答，就重发1次。如果超过5次仍没有收到应答，就将该包丢弃，并调用ICMP响应函数，从收到该包的端口回应源主机，目的主机不可达。如果收到ARP应答，就准备好所有因等待该ARP应答而挂起的包，将应答中的映射mac地址填写到以太网首部中目的mac地址，然后从该端口将包发送出去。

14.收到的包是ARP，解析ARP首部。如果目的IP不是端口IP，说明源主机要交互的对象不是该端口，将该包丢弃；否则，转到15。

15.解析ARP首部，如果代码是ARP请求，就从该端口发送ARP应答；如果是ARP应答，就将应答中IP->mac的映射关系添加到ARP缓存中。

主机请求交互流程：

1.源主机在发起通信之前，将自己的IP与目的主机的IP进行比较，如果两者位于同一网段（用子网掩码计算后具有相同的网络号），那么源主机直接向目的主机发送ARP请求，在接收到目的主机的ARP应答后获取对方MAC地址，然后用对方的MAC地址作为目标MAC地址进行报文发送，位于同一网段的主机互访时属于这种情况，这是互联的交换机做二层转发。

2、当源主机判断目的主机与自己位于不同网段时，它会通过网关来提交报文，即发送ARP请求来获取网关IP地址对应的MAC。通常来讲，每个主机也有自己的路由表，查询路由表一般情况下都会选择默认网关。在得到网关的ARP应答后，用网关MAC作为报文目的MAC进行报文发送，但是，报文的源IP是源主机IP，目的IP依然是目的主机IP。

2.启动脚本

1)四节点拓扑ROUTER测试

make all

sudo python router\_topo.py

mininet> xterm h1 h2 h3 r1

r1# ./router

h1# ping 10.0.1.1 -c 6

h1# ping 10.0.2.22 -c 6

h1# ping 10.0.3.33 -c 6

h1# ping 10.0.3.11 -c 6

h1# ping 10.0.4.1 -c 6

mininet> quit

2)六节点拓扑ROUTER测试

make all

sudo python my\_topo.py

mininet> xterm h1 h2 r1 r2 r3 r4

r1# ./router

r2# ./router

r3# ./router

r4# ./router

h1# ping 10.0.1.1 -c 6

h1# ping 10.0.2.2 -c 6

h1# ping 10.0.3.2 -c 6

h1# ping 10.0.4.2 -c 6

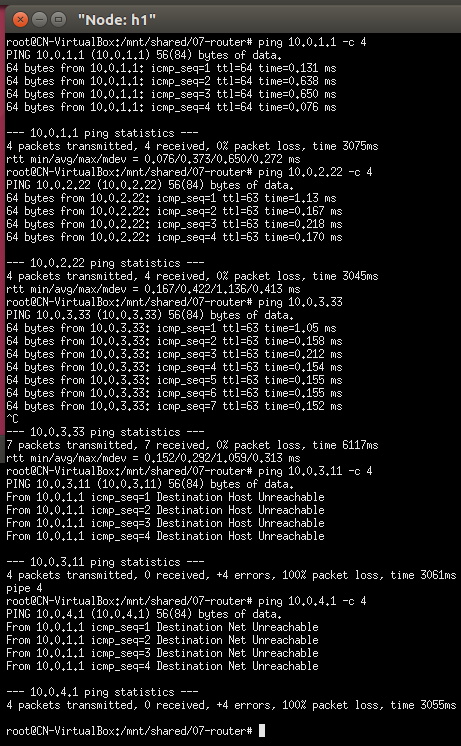
h1# ping 10.0.5.22 -c 6

h1# tracerouter 10.0.5.22 -m 6

mininet> quit

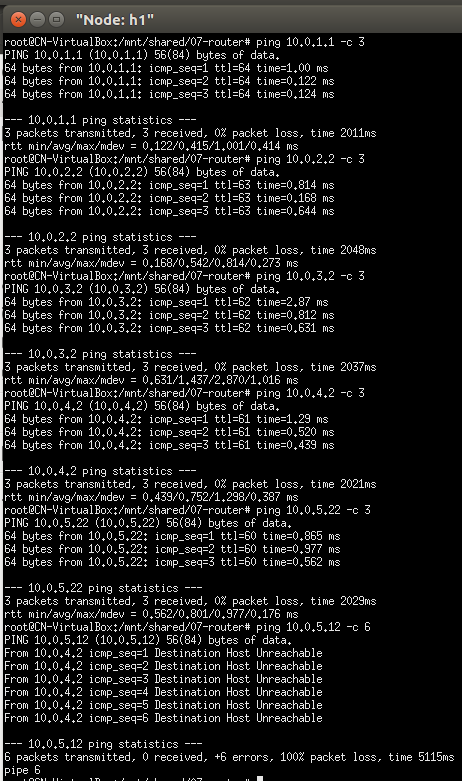
三、实验结果及分析

1. 实验结果



图三 四节点拓扑路由结果

可以看到，h1节点ping网关以及其他两个节点都能ping通，如果ping不存在的节点会报Host Unreachable，ping不存在的网段会报Net Unreachable。



图四 六节点拓扑路由结果

可以看到，h1节点ping网关以及其他三个路由器以及h2节点都能ping通，如果ping不存在的节点会报Host Unreachable，traceroute h2会返回正确的路由途径。

（二）实验代码详解

本次实验代码过长，不再赘述代码的内容。

一、arp.c:发送ARP请求和应答

（1）void handle\_arp\_packet(iface\_info\_t \*iface, char \*packet, int len)：处理一个收到的ARP包

首先从收到的packet中提取出属于arp报文的部分。

如果arp报文中target protocol address经过ntohl转换后与本地端口中保存的IP不相等的话，说明该arp报文并不是发给自己的，丢弃。

否则，进行如下的操作：

检查操作码，经过ntohs转换后如果为Request报文，说明是请求获得自己mac地址的服务，就调用arp\_send\_reply将自己的信息回复给源主机。如果为Reply报文，说明是自己的arp请求报文被回应，需要将收到packet中的sender protocol address和sender hareware address调用arpcache\_insert保存到ARP缓存中。

（2）void arp\_send\_reply(iface\_info\_t \*iface, struct

ether\_arp \*req\_hdr)：发送ARP应答

首先分配一块内存packet，大小为一个以太网报头加一个ARP报文的大小。

对该packet做预处理，装填好：ether\_shost为端口的mac地址，ether\_type为ARP，arp\_header为htons（0x0001），arp protocol为htons（0x0800），hardware address length为6，protocol address length为4，arp\_sender hardware address为端口的mac地址，arp\_sender protocol address为端口的IP。

然后装填：ether\_dhost为收到arp报文中的arp\_sender hardware address，arp\_op为htons（ARPOP\_REPLY），target hardware address为收到arp报文中的arp\_sender hardware address，target protocol address为收到arp报文中的arp\_sender protocol address。

最后将packet从该端口发送出去。

（3）void arp\_send\_request(iface\_info\_t \*iface, u32 dst\_ip)：发送ARP请求

首先分配一块内存packet，大小为一个以太网报头加一个ARP报文的大小。

对该packet做预处理，装填好：ether\_shost为端口的mac地址，ether\_type为ARP，arp\_header为htons（0x0001），arp protocol为htons（0x0800），hardware address length为6，protocol address length为4，arp\_sender hardware address为端口的mac地址，arp\_sender protocol address为端口的IP。

然后装填：ether\_dhost为0xFFFFFFFFFFFF，arp\_op为htons（ARPOP\_REQUEST），target hardware address为0，target protocol address为dst\_ip。

最后将packet从该端口发送出去。

二、arpcache.c:ARP缓存相关操作

（1）int arpcache\_lookup(u32 ip4, u8 mac[ETH\_ALEN])：查询ARP缓存

首先获取ARP缓存的互斥锁。

然后遍历查找ARP缓存的所有表项，如果某一表项的valid值为1，且表项中的IP地址与要查找的IP地址相同，就将该表项中缓存的mac地址保存到mac[ETH\_ALEN]中，释放互斥锁，返回1。

如果没找到，就释放互斥锁，返回0。

（2）void arpcache\_append\_packet(iface\_info\_t \*iface, u32 ip4, char \*packet, int len)：查询ARP缓存失败，将要发送的包挂起，发送ARP请求

首先分配一个cached\_pkt类型的cached\_packet节点，缓存packet和len。

获取互斥锁。

随后调用list\_for\_each\_entry宏查找链表arpcache.req\_list，链表中的节点类型为arp\_req。

如果找到链表中某个节点的等待ARP回复IP地址与ip4相同，且转发数据包的端口等于iface，那么就将cached\_packet调用list\_add\_tail挂到该arp\_req节点中的cached\_packets末端。

如果没找到，就新建这样一个arp\_req节点，设置其发送ARP请求的端口为iface，等待ARP回复的IP地址为ip4，发送ARP请求时间为当前时间，重试次数为0，初始化cached\_packets链表头，调用list\_add\_tail将cached\_packet挂载到cached\_packets，调用list\_add\_tail将该arp\_req节点挂到arpcache.req\_list上，然后调用arp\_send\_request从iface端口发送ARP请求ip4。

最后释放互斥锁。

（3）void arpcache\_insert(u32 ip4, u8 mac[ETH\_ALEN])：收到ARP应答后将IP->MAC映射写到缓存中，并将等待该映射的数据包发送出去

首先获取互斥锁。

随后对ARP缓存表项进行遍历，查找是否有valid项为0的表项（即超时）。如果查找到1个，就将该表项记录的IP替换为ip4，MAC地址替换为mac，添加时间修改为当前时间，valid修改为1，停止遍历。

如果没有找到，就要随机挑选一个表项删除：随机生成一个0到31之间的整数index，修改ARP缓存表中index号表项：IP替换为ip4，MAC地址替换为mac，添加时间修改为当前时间，valid修改为1。

之后调用list\_for\_each\_entry\_safe宏对arpcache.req\_list进行遍历：

查找链表中是否有arp\_req节点的IP与ip4相等。如果有，就继续调用list\_for\_each\_entry\_safe遍历其下链表cached\_packets：

将链表下每一个cached\_pkt节点中packet项的首部ether\_dhost填充为mac，将该包从arp\_req节点中记录的端口iface发送出去，然后释放该cached\_pkt节点。

处理完cached\_packets链表后，调用宏list\_delete\_entry将该arp\_req节点从arpcache.req\_list链表中删除，释放节点空间。

最后释放互斥锁。

（4）void \*arpcache\_sweep(void \*arg)：如果ARP缓存表中某表项存在15s以上，就标记为失效；对于等待ARP应答而挂起的待发包，如果ARP请求发送超过1s并且还没收到应答，就重发请求。如果发送5次请求后仍未收到应答，就向这些包的源主机发送ICMP DEST\_HOST\_UNREACHABLE，并丢弃这些包。

该程序通过sleep（1）控制，每1秒执行一次：

首先获取互斥锁，获取当前时间now。

遍历ARP缓存表，如果某个表项的valid为1，且当前时间与表项记录的入表时间之差大于15s，就将其valid修改为0。

随后调用宏list\_for\_each\_entry\_safe遍历arpcache.req\_list：

首先判断当前遍历到的arp\_req节点的发送次数是否超过5次：

如果超过5次，就要调用宏list\_for\_each\_entry\_safe对该节点下的cached\_packets链表进行遍历：

首先释放互斥锁。随后调用icmp\_send\_packet向各个包的源主机发送ICMP HOST不可达报文（type：3 code：1，ARP查询失败）。然后获取互斥锁，释放该节点。注意，在此处对锁的释放和获取是有意义的，因为在icmp\_send\_packet中控制流可能会通过调用到达arpcache\_append\_packet函数，从而发起对锁的另一次获取，导致死锁。

对cached\_packets的遍历结束后，从arpcache.req\_list中删除当前arp\_req节点，并释放节点空间。使用continue语句跳到外层遍历的下一个对象。

如果没超过5次，并且now减arp\_req节点记录的发送时间大于1s，就调用arp\_send\_request重新发送请求，将发送时间修改为now，发送次数加1。

对arpcache.req\_list的遍历结束后，释放互斥锁。

三、icmp.c:发送ICMP数据包

（1）void icmp\_send\_packet(const char \*in\_pkt, int len, u8 type, u8 code)：发送ICMP数据包

首先用in\_pkt\_etherhead指向in\_pkt的以太网首部，调用packet\_to\_ip将in\_pkt\_IPhead指向in\_pkt的IP首部。

随后计算要发送packet的空间：

如果是要回复ping本端口的数据包，那么大小为len；

如果是其他，大小为ETHER\_HDR\_SIZE + IP\_BASE\_HDR\_SIZE + ICMP\_HDR\_SIZE + IP\_HDR\_SIZE(in\_pkt\_IPhead) + 8：要发送的packet有自己的以太网头部，自己的标准IP首部（20字节），标准ICMP首部（type+code+Checksum+4，其中后4字节要设置为0），以及Rest of ICMP header（该部分填充in\_pkt的IP首部和随后的8字节）

按照计算出的空间大小分配packet。

设置packet的以太网头：ether\_dhost设置为in\_pkt\_etherhead的ether\_shost，ether\_shost设置为in\_pkt\_etherhead的ether\_dhost，ether\_type 设置为htons(ETH\_P\_IP)。

调用packet\_to\_ip找到packet的IP首部，然后设置IP首部：

调用longest\_prefix\_match在ARP缓存中查找经过ntohl（）转换的in\_ihdr的sender IP address，查找到表项后，调用ip\_init\_hdr对packet的IP首部进行设置：

设置version为4，ihl为5（对应首部大小20字节），tos为0，tot\_len为packet的大小减以太网首部的大小，id为rand（），frag\_off为htons(IP\_DF)，ttl为64（DEFAULT），protocol为1表示为ICMP，saddr为htonl（）处理后的ARP表项中的端口IP（表示该条ICMP由该端口发出），daddr为htonl（）处理后的[ntohl（）处理后的从in\_pkt\_IPhead提取出的sender IP address]，校验和为ip\_checksum(ip)。

之后设置ICMP首部：类型为type，代码为code。

随后查找到packet的Rest of ICMP header的起始处，对内容进行修改：

如果是响应ping的应答，就将in\_pkt相同位置起始处的剩余内容都copy过来。

否则，就先设置开头4个字节为0，剩余部分将in\_pkt的IP首部以及之后的8个字节（ICMP首部）copy过来。

最后设置ICMP首部的校验和为icmp\_checksum(packet\_IChead, pkt\_len - ETHER\_HDR\_SIZE - IP\_BASE\_HDR\_SIZE)。之所以这样计算，是因为当要回复的in\_pkt是PING时，PING发送的报文的IP Head大小为20；要回复的如果不是PING，那么IP Head会初始化为20。也就是说，无论应答谁，该报文的IP Head都是20（IP\_BASE\_HDR\_SIZE）。

最后调用ip\_send\_packet将packet发送出去。

四、icmp.c:最长前缀查找和发送IP数据包

（1）rt\_entry\_t \*longest\_prefix\_match(u32 dst)：最长前缀查找

调用list\_for\_each\_entry遍历转发表rtable，如果(rt->dest & rt->mask) == (dst & rt->mask)，并且rt->mask比当前找到的最长前缀掩码更大，就更新最长前缀掩码的值和最长前缀掩码对应的表项。遍历结束后，返回最长前缀掩码对应的表项。

（2）void ip\_send\_packet(char \*packet, int len)：发送IP数据包（只用于自己发送ICMP报文，而不是转发IP数据包）

首先找到packet中IP首部中的dest IP address，用ntohl（）处理后记为dst\_ip。调用longest\_prefix\_match函数查找dst\_ip的表项。

返回的表项的下一跳网关地址gw若不为0，说明该路由器任何端口的IP都与目的IP不在同一网段，就调用iface\_send\_packet\_by\_arp(rt->iface, rt->gw, packet, len)，含义为从该表项记录的端口将ICMP报文发出，以太网报头的目的IP为rt->gw，源IP为发送端口的IP。

如果gw为0，说明该表项记录的下一跳网关为空，即目标IP与该表项记录的端口在同一个网段，就调用iface\_send\_packet\_by\_arp(rt->iface, dst\_ip, packet, len)，含义为从该表项记录的端口将ICMP报文发出，以太网报头的目的IP为dst\_ip，源IP为发送端口的IP。

五、ip.c:处理IP数据包，包括转发

（1）void handle\_ip\_packet(iface\_info\_t \*iface, char \*packet, int len)：如果收到的包目标是本路由器端口，并且ICMP首部type为8，就回应ping的ICMP报文，否则丢弃；如果目标不是本路由器，就转发。

首先从packet的IP首部中找到dest IP address，用ntohl（）处理后记为IP\_dest\_addr。将IP\_dest\_addr与当前端口IP比较：

如果相同，说明是发给本路由器端口的ICMP包。解析该包的ICMP type字段：

如果为8，说明为PING本端口的数据包，就调用icmp\_send\_packet(packet, len, ICMP\_ECHOREPLY, 0)，type：0 code：0应答。

如果不为8的话就将该包丢弃。

如果不相同，说明是需要转发的包。首先在路由表中最长前缀匹配查找路由表项：

如果没找到（路由表查找失败，没有该网段），就调用icmp\_send\_packet(packet, len, ICMP\_DEST\_UNREACH, ICMP\_NET\_UNREACH)，发送type：3 code：1的ICMP报文。

如果找到了，首先将IP首部的ttl减1。

如果ttl变为0（生存期耗尽），就调用icmp\_send\_packet(packet, len, ICMP\_TIME\_EXCEEDED, ICMP\_EXC\_TTL)，发送type：11 code：0的ICMP报文，返回。

否则就重新计算校验和，填写到checksum中。返回的表项的下一跳网关地址gw若不为0，说明该路由器任何端口的IP都与目的IP不在同一网段，就调用iface\_send\_packet\_by\_arp(rt->iface, rt->gw, packet, len)，含义为从该表项记录的端口将ICMP报文发出，以太网报头的目的mac地址为rt->gw对应的mac地址，源mac地址为发送端口的mac地址。

如果gw为0，说明该表项记录的下一跳网关为空，即目标IP与该表项记录的端口在同一个网段，就调用iface\_send\_packet\_by\_arp(rt->iface, IP\_dest\_addr, packet, len)，含义为从该表项记录的端口将ICMP报文发出，以太网报头的目的mac地址为IP\_dest\_addr对应的IP地址，源mac地址为发送端口的mac地址。