

南京大学计算机网络实验报告

任课教师:田臣

实验四 Forwarding Packets

计算机科学与技术系

181860055 刘国涛

邮箱: 181860055@smail.nju.edu.cn

2020年4月21日

实验目的

- 学习router对ARP的响应机制
- 实现带缓存表的router类

实验内容

TASK 2 IP Forwarding Table Lookup

任务概述 建立转发表，识别接收到的数据包，并将目标地址与转发表匹配，得到应该将要转发出的接口

任务实现

需要实现一个 `forwarding_table` 类，并定义如下接口

```
1  class forwarding_table():
2      table = [] # 用于存储表项信息
3      def __init__(self, interfaces):
4          #初始化，从forwarding_table.txt和interfaces中读取
5          #每一个表项包含ip, prefix, next hop和interface
6          #表项存入table中
7      def get(self, destaddr):
8          #get方法
9          #根据destaddr在表中按照最长前缀匹配一个表项
10     def print(self):
11         #打印每个表项，用于调试
12
```

首先要实现表项类 `forwarding_item`

```
1  class forwarding_item():
2      def __init__(self, ip, prefix, next_hop, itf):
3          ipaddr = IPv4Address(ip)
4          prefixaddr = IPv4Address(prefix)
5          ipnum = int(ipaddr)&int(prefixaddr)
6          ipnet = str(IPv4Address(ipnum))
7          self.prefixnet = IPv4Network(ipnet+'/'+prefix)
8          if next_hop is not None:
9              self.nexthop = IPv4Address(next_hop)
```

```

10         else:
11             self.nexthop = None
12             self.interface = itf
13         def match(self, destaddr):
14             return destaddr in self.prefixnet
15         def prefixlen(self):
16             return self.prefixnet.prefixlen
17         def __str__(self):
18             return "ip:{}    nexthop:{}    interface:
{}".format(self.prefixnet.with_netmask, self.nexthop, self.i
nterface)

```

然后以表项类实现 forwarding_table

```

1     def get(self, destaddr):
2         max_prefixlen = -1
3         sel = None
4         for item in self.table: # 最长前缀匹配
5             if item.match(destaddr) and
item.prefixlen() > max_prefixlen:
6                 max_prefixlen = item.prefixlen()
7                 sel = item
8         return sel

```

TASK 3 Forwarding the Packet and ARP

任务概述 实现转发数据包

任务实现

要实现数据包的转发，需要考虑以下问题：

1. 数据包从哪个接口发出
2. 下一跳的MAC地址如何获得

首先对于第一个问题，可以通过在Task 2中实现的 forwarding_table 获得对应的接口，即

```

1  ipv4 = pky.get_header(IPv4)
2  if ipv4 is not None:
3      res = self.forwarding_table.get(ipv4.dst)
4      log_info("forwarding interface is
{}").format(res.interface))

```

对于第二个问题，则有：

1. 在缓存的ARP表中查找IP和MAC的映射
2. 发送ARP请求询问下一跳IP的MAC

第一个在 lab 3 已经实现，第二个的实现需要建立一个arp等待队列 `ARP_queue` 类，实现如下：

```

1  class ARP_queue():
2      q = []
3      def __init__(self):
4          pass
5
6      def add(self, arp, packet, ip, dp, sp):
7
8          self.q.append({'arp':arp, 'dest_port':dp, 'source_port':sp,
9
10             'packet':packet, 'ip':ip, 'time':time.time(), 'times':0})
11
12      def reply(self, arp, itf, net):
13          ip = arp.senderprotoaddr
14          mac = arp.senderhwaddr
15          for i in range(len(self.q)-1, -1, -1):
16              if ip == self.q[i]['ip']:
17                  # creat eth header
18                  self.q[i]['packet'][0].dst = mac
19                  self.q[i]['packet'][0].src =
20                  net.interface_by_name(itf).ethaddr
21                  # send
22                  net.send_packet(itf, self.q[i]['packet'])
23                  # pop
24                  self.q.pop(i)
25
26      def resend(self, net):

```

```

24         now = time.time()
25         for i in range(len(self.q)-1,-1,-1):
26             if self.q[i]['times'] >= 4:
27                 self.q.pop(i)
28                 continue
29             if now - self.q[i]['time'] > 1: # Timeout ,
resend arp pkt
30                 net.send_packet(self.q[i]
['dest_port'],self.q[i]['arp'])
31                 self.q[i]['times']+=1
32                 log_info(self.q[i]['times'])
33
34     def print(self):
35         for i in self.q:
36             print(i)

```

主要实现的是超时重发，和响应

实验结果

TASK 3

Deploying:

步骤:

1、正常的转发流程

```
server1 ping -c2 client
```

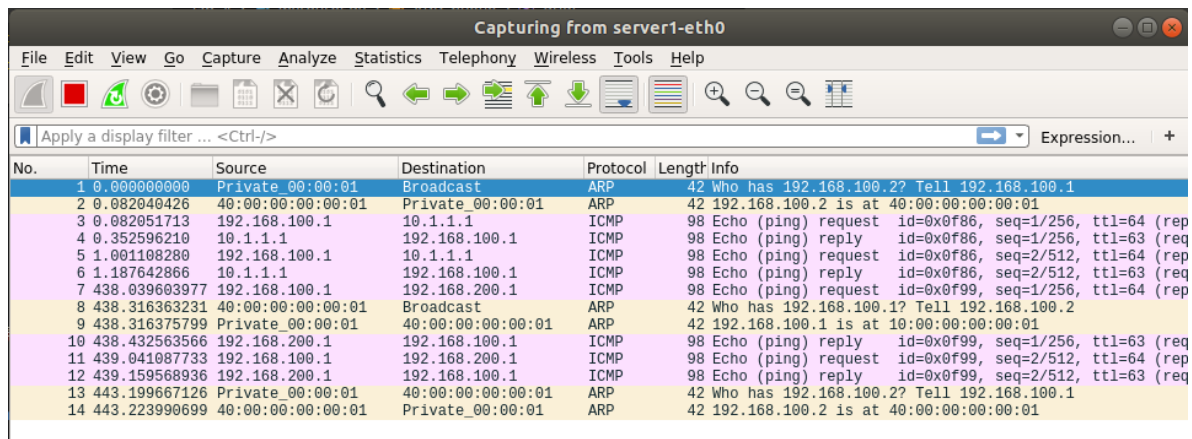
在router-eth2上wireshark的抓包结果如下

The image shows a Wireshark packet capture window titled "Capturing from router-eth2". The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help), a toolbar with various icons, and a display filter bar set to "Apply a display filter ... <Ctrl-/>". The packet list pane shows 8 captured packets. The first two are ARP requests from 40:00:00:00:00:03 to 40:00:00:00:00:03. The next four are ICMP Echo (ping) requests and replies between 192.168.100.1 and 10.1.1.1. The last packet is an ARP request from 40:00:00:00:00:03 to 30:00:00:00:00:01.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.1? Tell 10.1.1.2
2	0.000025011	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	10.1.1.1 is at 30:00:00:00:00:01
3	0.103645641	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x0f86, seq=1/256, ttl=63 (req)
4	0.103679173	10.1.1.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f86, seq=1/256, ttl=64 (rep)
5	0.938872690	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x0f86, seq=2/512, ttl=63 (req)
6	0.938906601	10.1.1.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f86, seq=2/512, ttl=64 (rep)
7	5.293473797	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
8	5.318571807	40:00:00:00:00:03	30:00:00:00:00:01	ARP	42	10.1.1.2 is at 40:00:00:00:00:03

```
server1 ping -c2 server2
```

在server1上wireshark的抓包结果如下，而router-eth2的结果依然如上
图



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
2	0.082040426	40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01
3	0.082051713	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x0f86, seq=1/256, ttl=64 (req)
4	0.352596210	10.1.1.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f86, seq=1/256, ttl=63 (rep)
5	1.001108280	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x0f86, seq=2/512, ttl=64 (req)
6	1.187642866	10.1.1.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f86, seq=2/512, ttl=63 (rep)
7	438.039603977	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x0f99, seq=1/256, ttl=64 (req)
8	438.316363231	40:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
9	438.316375799	Private_00:00:01	40:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01
10	438.432563566	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f99, seq=1/256, ttl=63 (rep)
11	439.041087733	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x0f99, seq=2/512, ttl=64 (req)
12	439.159568936	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f99, seq=2/512, ttl=63 (rep)
13	443.199667126	Private_00:00:01	40:00:00:00:00:01	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
14	443.223990699	40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01

结果分析：

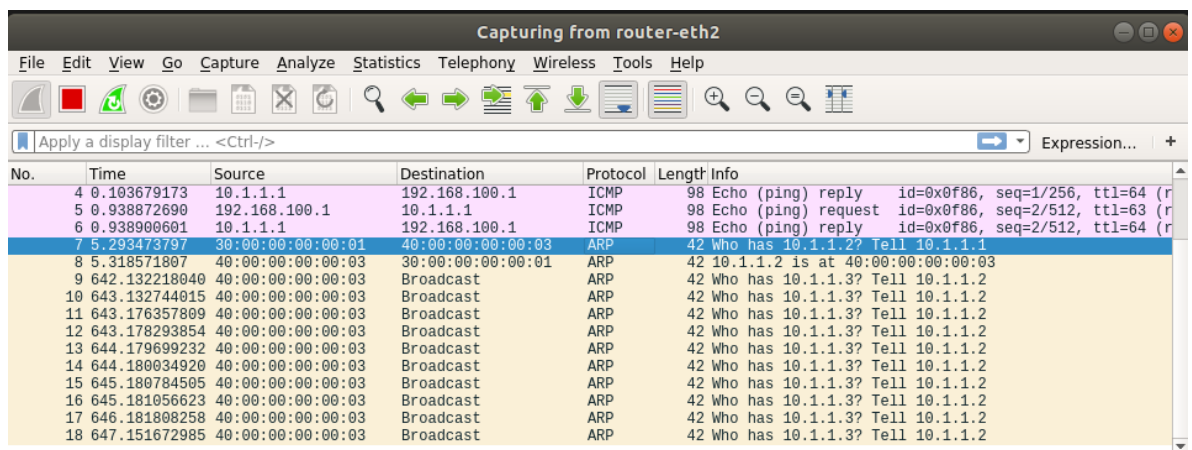
可以看到router完成了数据包的转发，其转发流程为：

1. server1发送ARP请求和client应答时，router学习到了双方的IP->MAC规则
2. router接收到第一个server1发来的ICMP数据包后，在转发表中找到需要发出的接口 **eth2**
3. router通过ARP缓存表获得了client的IP对应的MAC
4. 根据以上信息，修改原数据包的以太网包头，然后从 **eth2** 发出

2、ARP等待队列的重发

```
server1 ping -c2 10.1.1.3
```

如图得到router eth2的抓包结果



No.	Time	Source	Destination	Protocol	Length	Info
4	0.103679173	10.1.1.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f86, seq=1/256, ttl=64 (r
5	0.938872690	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x0f86, seq=2/512, ttl=63 (r
6	0.938900601	10.1.1.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0f86, seq=2/512, ttl=64 (r
7	5.293473797	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
8	5.318571807	40:00:00:00:00:03	30:00:00:00:00:03	ARP	42	10.1.1.2 is at 40:00:00:00:00:03
9	642.132218040	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
10	643.132744015	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
11	643.176357809	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
12	643.178293854	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
13	644.179699232	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
14	644.180034920	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
15	645.180784505	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
16	645.181056623	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
17	646.181808258	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2
18	647.151672985	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.3? Tell 10.1.1.2

并分析router的日志

```

here is table:
ip:10.1.1.1 mac:30:00:00:00:00:01 time:438,0723934173584
ip:192.168.200.1 mac:20:00:00:00:00:01 time:6,19888305640625e-06
ip:192.168.100.1 mac:10:00:00:00:00:01 time:0,51174540919714555

22:31:29 2020/04/22 INFO ipv4 header:None
22:34:48 2020/04/22 INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->10.1.1.3 ICMP | ICMP EchoRequest 4016 1 (56 data bytes) on router-eth0
22:34:48 2020/04/22 INFO it is for me
22:34:48 2020/04/22 INFO arp header: None
22:34:49 2020/04/22 INFO ipv4 header:IPv4 192.168.100.1->10.1.1.3 ICMP
22:34:49 2020/04/22 INFO search result:ip:10.1.1.0/255,255,252 nexthop:None interface:router-eth2
22:34:48 2020/04/22 INFO target None
22:34:48 2020/04/22 INFO send arp packet for 10.1.1.3's ethaddr
22:34:49 2020/04/22 INFO 1
22:34:49 2020/04/22 INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->10.1.1.3 ICMP | ICMP EchoRequest 4016 2 (56 data bytes) on router-eth0
22:34:49 2020/04/22 INFO it is for me
22:34:49 2020/04/22 INFO arp header: None
22:34:49 2020/04/22 INFO ipv4 header:IPv4 192.168.100.1->10.1.1.3 ICMP
22:34:49 2020/04/22 INFO search result:ip:10.1.1.0/255,255,252 nexthop:None interface:router-eth2
22:34:49 2020/04/22 INFO target None
22:34:49 2020/04/22 INFO send arp packet for 10.1.1.3's ethaddr
22:34:49 2020/04/22 INFO 2
22:34:50 2020/04/22 INFO 1
22:34:50 2020/04/22 INFO 3
22:34:51 2020/04/22 INFO 2
22:34:51 2020/04/22 INFO 4
22:34:52 2020/04/22 INFO 3
22:34:53 2020/04/22 INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.100.2 on router-eth0
22:34:53 2020/04/22 INFO it is for me
22:34:53 2020/04/22 INFO arp header: Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.100.2
22:34:53 2020/04/22 INFO my interface: router-eth0 mac:40:00:00:00:00:01 ip:192.168.100.2/30
22:34:53 2020/04/22 INFO reply arp packet: Ethernet 40:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 40:00:00:00:00:01:192.168.100.2 10:00:00:00:00:01:192.168.100.1 by port router-eth0

here is table:
ip:10.1.1.1 mac:30:00:00:00:00:01 time:641,6327672481537
ip:192.168.200.1 mac:20:00:00:00:00:01 time:208,76039002967834
ip:192.168.100.1 mac:10:00:00:00:00:01 time:1,049041748046875e-05

22:34:53 2020/04/22 INFO ipv4 header:None
22:34:53 2020/04/22 INFO 4

```

可以看到，router在接收到ICMP数据包（两次）后，由于ARP缓存表中不存在10.1.1.3对应的MAC地址，所以router发送了ARP询问，并将其加入到ARP等待队列中

日志中箭头标出的数字代表了重发次数，可以看到，每个ARP包在四次重发后（一共发送了五次ARP）和数据包一起被丢弃，结束重发

总结与感想

本次实验理解了router在数据包转发中的具体实现，感受到了网络中数据传输的复杂性