

计算机网络 lab_5

181840135 梁俊凱

Task1:

完成实验所需的相关准备工作。

Task2:

产生 ICMP echo reply, 我实现的逻辑如下:

1>首先检查包的内容，若有 ICMP 头部且目的 IP 属于路由器的某一端口，执行 2>

2>设置 reply 包的 ICMP 头，目的 ip 为原包的源 ip，源 mac 和 ip 地址为路由器相应端口的地址，根据原包的信息写入 reply 包的 sequence number，identifier 和 data field 三个部分。

3>调用转发函数，在 forward_table 进行匹配，得到最长匹配后在 cache_table 进行匹配，若得到目的 mac 地址，则发包；若未得到，则将包放入队列 arp_queue 中，进行 arp 请求后再发包。

```

if ipv4.dst in router_ip:

    if ipv4.protocol==IPProtocol.ICMP:
        if icmp.icmptype==ICMPType.EchoRequest:
            print('recieve pkt ICMP request\n')
            pkt=pkt_ping(eth.dst,eth.src,ipv4.dst,ipv4.src,reply=True,\
                           payload=icmp.icmpdata.data,sequence=icmp.icmpdata.sequence)
            #pos=forward_match(ipv4.src)
            print('\nicmp pkt\n')
            pkt_path(pkt,self.net)
            continue

```

包的设置和转发分别封装在函数 `pkt_ping` 和 `pkt_path` 中

Task3:

实现 ICMP error message

共有四个 error message, 我将根据他们共性和区别来介绍我的实现。

共性：

1>在包的生成中均需要设置 ICMPType, ICMPCode 和 origdgramlen 字段，仅相应的类型不同

2>payload 要设置为 IP 头部的 28 字节

我的实现代码部分如下

```
elif Type=='ttl':
    icmp=ICMP()
    icmp.icmptype=ICMPType.TimeExceeded
    icmp.icmpcode=ICMPCodeTimeExceeded
    icmp.icmpdata.origdgramlen=len
elif Type=='unreachable':
    icmp=ICMP()
    icmp.icmptype=ICMPType.DestinationUnreachable
    icmp.icmpcode=ICMPCodeDestinationUnreachable
    icmp.icmpdata.origdgramlen=len
elif Type=='port_unreach':
    icmp=ICMP()
    icmp.icmptype=ICMPType.DestinationUnreachable
    icmp.icmpcode=ICMPCodeDestinationUnreachable.PortUnreachable
    icmp.icmpdata.origdgramlen=len
elif Type=='arp_fail':
    icmp=ICMP()
    icmp.icmptype=ICMPType.DestinationUnreachable
    icmp.icmpcode=ICMPCodeDestinationUnreachable.HostUnreachable
    icmp.icmpdata.origdgramlen=len
icmp.icmpdata.data=payload
```

可以看到，我根据错误类型的不同，来设置不同的 type，payload 作为该函数的输入在最后统一设置

区别：

四个错误的主要区别在于其产生的条件和时间不同，实现如下

1>在 forward_table 中找不到匹配，产生 ICMP destination network unreachable 错误，实现在 pkt_path 中，此函数为每个包的转发函数，一旦在 forward_table 中找不到表项，就产生一条 ICMP error 信息，并根据 ip-mac 表来决定是进入队列还是立即转发

```

pos=forward_match(ipv4.dst)

if pos==None:
    print('\npkt unreachable\n')
    origin_pkt=pkt
    del origin_pkt[origin_pkt.get_header_index(Ethernet)]
    pkt2=pkt_ping(eth.dst,eth.src,ipv4.dst,ipv4.src,Type='unreachable',payload=origin_pkt.to_bytes()[28],len=len(pkt))
    pkt_path(pkt2,net,error=True,depth=depth-1)

```

递归调用了 pkt_path 函数，并设置了最大深度，若该包的源地址也不可达（即 Error message 无法送至源地址）简单丢弃该包。

2>t1 为 0，在收到 IP 包之后并进行过一次地址匹配之后（个人感觉是否在地址匹配之后对网络传输影响不大），若 t1 为 0，则产生相应的 ICMP error 返回给发送端

```

ipv4.ttl-=1
if ipv4.ttl==0:
    #print('debug:icmpdata.data:{}, to_bytes:{}'.format icmp.icmpdata.data,pkt.to_bytes()[28]))
    origin_pkt=pkt
    del origin_pkt[origin_pkt.get_header_index(Ethernet)]
    pkt=pkt_ping(eth.dst,eth.src,ipv4.dst,ipv4.src,Type='ttl',\
                payload=origin_pkt.to_bytes()[28],sequence=icmp.icmpdata.sequence,len=len(pkt))
    pkt_path(pkt,self.net,error=True)
    continue

```

3>arp fail 主要发生在每次 while 循环对 arp_queue 的检查上，若有的表项已经重传了超过五次，则丢弃之，并给该表项的每个包发送一条错误提示。

```

for key in list(arp_queue.keys()):
    if time.time()-arp_queue[key][1]>1 and arp_queue[key][2]>=5:
        log_info('arp_table some item transmission exceed 5 times\n')
        port = self.net.interface_by_name(arp_queue[key][3])
        for pkt in arp_queue[key][0]:
            eth=pkt.get_header(Ethernet)
            ipv4=pkt.get_header(IPv4)
            origin_pkt=pkt
            del origin_pkt[origin_pkt.get_header_index(Ethernet)]
            pkt2=pkt_ping(port.ethaddr,eth.src,port.ipaddr,ipv4.src,Type='arp_fail',payload=origin_pkt.to_bytes()[28],len=len(pkt))
            pkt_path(pkt2,self.net,error=True)
        del arp_queue[key]

```

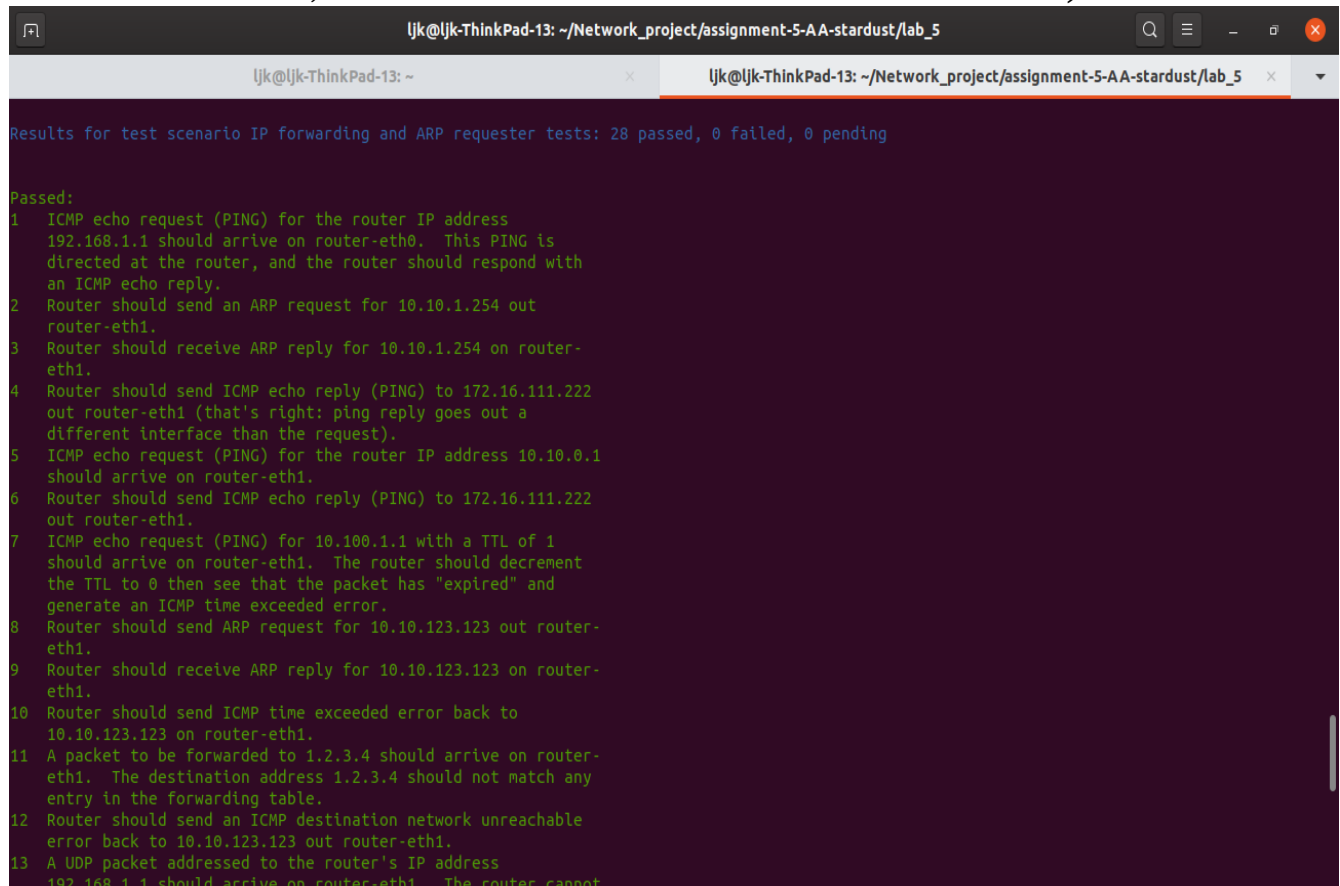
4>发送的路由器端口的包却不是 ICMP echo 包，就产生错误提示
使用了相应的 if 语句来判断条件

```
if ipv4.dst in router_ip:

    if ipv4.protocol==IPProtocol.ICMP:
        if icmp.icmptype==ICMPType.EchoRequest:
            print('recieve pkt ICMP request\n')
            pkt=pkt_ping(eth.dst,eth.src,ipv4.dst,ipv4.src,reply=True,\
                payload=icmp.icmpdata.data,sequence=icmp.icmpdata.sequence)
            #pos=forward_match(ipv4.src)
            print('\nicmp pkt\n')
            pkt_path(pkt,self.net)
            continue
        origin_pkt=pkt
        del origin_pkt[origin_pkt.get_header_index(Ethernet)]
        pkt2=pkt_ping(eth.dst,eth.src,ipv4.dst,ipv4.src,Type='port_unreach',payload=origin_pkt.to_bytes()[1:28],len=len(pkt))
        pkt_path(pkt2,self.net,True)
        continue
```

前三个 if 代表发送到 router 的包是 ICMP.EchoRequest
下半部分的代码是代表此次错误的条件，进行相应的处理

根据上述的思路，我通过了相应的 test 文件（共 28 个 test）结果展示如下：



```
ljk@ljk-ThinkPad-13: ~/Network_project/assignment-5-AA-stardust/lab_5

Results for test scenario IP forwarding and ARP requester tests: 28 passed, 0 failed, 0 pending

Passed:
1 ICMP echo request (PING) for the router IP address
  192.168.1.1 should arrive on router-eth0. This PING is
  directed at the router, and the router should respond with
  an ICMP echo reply.
2 Router should send an ARP request for 10.10.1.254 out
  router-eth1.
3 Router should receive ARP reply for 10.10.1.254 on router-
  eth1.
4 Router should send ICMP echo reply (PING) to 172.16.111.222
  out router-eth1 (that's right: ping reply goes out a
  different interface than the request).
5 ICMP echo request (PING) for the router IP address 10.10.0.1
  should arrive on router-eth1.
6 Router should send ICMP echo reply (PING) to 172.16.111.222
  out router-eth1.
7 ICMP echo request (PING) for 10.100.1.1 with a TTL of 1
  should arrive on router-eth1. The router should decrement
  the TTL to 0 then see that the packet has "expired" and
  generate an ICMP time exceeded error.
8 Router should send ARP request for 10.10.123.123 out router-
  eth1.
9 Router should receive ARP reply for 10.10.123.123 on router-
  eth1.
10 Router should send ICMP time exceeded error back to
  10.10.123.123 on router-eth1.
11 A packet to be forwarded to 1.2.3.4 should arrive on router-
  eth1. The destination address 1.2.3.4 should not match any
  entry in the forwarding table.
12 Router should send an ICMP destination network unreachable
  error back to 10.10.123.123 out router-eth1.
13 A UDP packet addressed to the router's IP address
  192.168.1.1 should arrive on router-eth1. The router cannot
```

```
ljk@ljk-ThinkPad-13: ~/Network_project/assignment-5-AA-stardust/lab_5
error back to 172.16.111.222 out router-eth1.
15 An IP packet from 192.168.1.239 for 10.10.50.250 should
    arrive on router-eth0. The host 10.10.50.250 is presumed
    not to exist, so any attempts to send ARP requests will
    eventually fail.
16 Router should send an ARP request for 10.10.50.250 on
    router-eth1.
17 Router should try to receive a packet (ARP response), but
    then timeout.
18 Router should send an ARP request for 10.10.50.250 on
    router-eth1.
19 Router should try to receive a packet (ARP response), but
    then timeout.
20 Router should send an ARP request for 10.10.50.250 on
    router-eth1.
21 Router should try to receive a packet (ARP response), but
    then timeout.
22 Router should send an ARP request for 10.10.50.250 on
    router-eth1.
23 Router should try to receive a packet (ARP response), but
    then timeout.
24 Router should send an ARP request for 10.10.50.250 on
    router-eth1.
25 Router should try to receive a packet (ARP response), but
    then timeout. At this point, the router should give up and
    generate an ICMP host unreachable error.
26 Router should send an ARP request for 192.168.1.239.
27 Router should receive ARP reply for 192.168.1.239.
28 Router should send an ICMP host unreachable error to
    192.168.1.239.

All tests passed!

(syenv) ljk@ljk-ThinkPad-13:~/Network_project/assignment-5-AA-stardust/lab_5$
```

部署到我的 mininet 上测试结果如下：

使用 server1 来进行讲义中的测试

1>ping -c1 192.168.100.2(router 的一个端口 ip), 结果如下：

```
root@ljk-ThinkPad-13:/home/ljk/Network_project/assignment-5-AA-stardust/lab_5#
ping -c1 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=100 time=212 ms

--- 192.168.100.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 212.221/212.221/212.221/0.000 ms
```

收到了 ICMP reply, 符合预期

wireshark 结果如下：可以

1	0.000000000	192.168.100.1	192.168.100.2	ICMP	98 Echo (ping) request id=0x5012, seq=1/256, ttl=64 ...
2	0.121824607	40:00:00:00:00:01	Broadcast	ARP	42 Who has 192.168.100.1? Tell 192.168.100.2
3	0.121849939	Private_00:00:01	40:00:00:00:00:01	ARP	42 192.168.100.1 is at 10:00:00:00:00:01
4	0.212179895	192.168.100.2	192.168.100.1	ICMP	98 Echo (ping) reply id=0x5012, seq=1/256, ttl=100...
5	5.111873615	Private_00:00:01	40:00:00:00:00:01	ARP	42 Who has 192.168.100.2? Tell 192.168.100.1
6	5.216801010	40:00:00:00:00:01	Private_00:00:01	ARP	42 192.168.100.2 is at 40:00:00:00:00:01

2>ping -c1 -t1 10.1.1.1, 测试 ttl error, 结果如下：

```

root@ljk-ThinkPad-13:/home/ljk/Network_project/assignment-5-AA-stardust/lab_5#
ping -c1 -t1 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
From 192.168.100.2 icmp_seq=1 Time to live exceeded

--- 10.1.1.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

```

收到了 ttl error message

wireshark 中结果如下：可见

7	280.931239898	192.168.100.1	10.1.1.1	ICMP	98 Echo (ping) request id=0x5135, seq=1/256, ttl=1 (...)
8	281.001875808	192.168.100.2	192.168.100.1	ICMP	70 Time-to-live exceeded (Time to live exceeded in tr...

3>ping -c1 10.0.123.234 该 ip 地址不在路由器的转发表里，故应产生 Net Unreachable 错误信息

```

root@ljk-ThinkPad-13:/home/ljk/Network_project/assignment-5-AA-stardust/lab_5#
ping -c1 10.1.123.234
PING 10.1.123.234 (10.1.123.234) 56(84) bytes of data.
From 192.168.100.2 icmp_seq=1 Destination Net Unreachable

--- 10.1.123.234 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

```

与预期一致

11	557.465685866	192.168.100.1	10.1.123.234	ICMP	98 Echo (ping) request id=0x531f, seq=1/256, ttl=64 ...
12	557.496443423	192.168.100.2	192.168.100.1	ICMP	70 Destination unreachable (Network unreachable)

wireshark 中产生了相应的错误信息

4>traceroute 10.1.1.1，结果如下：

```

root@ljk-ThinkPad-13:/home/ljk/Network_project/assignment-5-AA-stardust/lab_5#
traceroute 10.1.1.1
traceroute to 10.1.1.1 (10.1.1.1), 30 hops max, 60 byte packets
 1  192.168.100.2 (192.168.100.2)  15.949 ms  23.153 ms  29.118 ms
 2  10.1.1.1 (10.1.1.1)  116.882 ms  129.959 ms  133.089 ms

```

可见两个包后到达目的地，符合预期

通过 wireshark 发现 traceroute 发送的都是 UDP 包，但 ttl 是含在 IP 头部的，故我的代码可以兼容。

40	769.523916720	Private_00:00:01	40:00:00:00:00:01	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
41	769.618476711	40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01
42	816.255459817	192.168.100.1	10.1.1.1	UDP	74	34674 → 33434 Len=32
43	816.255561374	192.168.100.1	10.1.1.1	UDP	74	58394 → 33435 Len=32
44	816.255617174	192.168.100.1	10.1.1.1	UDP	74	40076 → 33436 Len=32
45	816.255661539	192.168.100.1	10.1.1.1	UDP	74	45921 → 33437 Len=32
46	816.255706557	192.168.100.1	10.1.1.1	UDP	74	45238 → 33438 Len=32
47	816.255749417	192.168.100.1	10.1.1.1	UDP	74	57414 → 33439 Len=32
48	816.255843949	192.168.100.1	10.1.1.1	UDP	74	41177 → 33440 Len=32
49	816.255893055	192.168.100.1	10.1.1.1	UDP	74	57466 → 33441 Len=32
50	816.255937770	192.168.100.1	10.1.1.1	UDP	74	39027 → 33442 Len=32
51	816.255980465	192.168.100.1	10.1.1.1	UDP	74	34975 → 33443 Len=32
52	816.256031635	192.168.100.1	10.1.1.1	UDP	74	48083 → 33444 Len=32
53	816.256076512	192.168.100.1	10.1.1.1	UDP	74	36867 → 33445 Len=32
54	816.256127698	192.168.100.1	10.1.1.1	UDP	74	53479 → 33446 Len=32
55	816.256171591	192.168.100.1	10.1.1.1	UDP	74	51965 → 33447 Len=32
56	816.256215256	192.168.100.1	10.1.1.1	UDP	74	40348 → 33448 Len=32

通过 wireshark 截取的部分结果如下，其中 UDP 包的 ttl 不同，每三个包为一组，然后 ttl 加 1，符合预期。

本次实验到此结束，感谢阅读！