计算机网络实验 Lab4 实验报告

计算机学院 计算机科学与技术系 221220095 曾凡俊

Email：221220095@smail.nju.edu.cn

一、实验名称：Forwarding Packets

二、实验目的

接收和转发到达链路并发往其他主机的数据包。转发过程的一部分是在转发表中执行地址查找（“最长前缀匹配”查 找）。 对没有已知以太网 MAC 地址的 IP 地址发出 ARP 请求

三、实验内容

Task1：IP Forwarding Table Lookup

实现:

```
def initialize_forward_table(self):
    for interface in self.interfaces:
        self.forward_table.append([IPv4Address(int2ipv4(int(IPv4Address(interface.ipaddr)) & int(IPv4Address(interface.netma
    with open('forwarding_table.txt','r') as txt:
        lines=txt.readlines()
    for line in lines:
        line=line.strip().split(' ')
        self.forward_table.append([IPv4Address(line[0]),IPv4Address(line[1]),IPv4Address(line[2]),line[3]])
    self.forward_table.sort(key=lambda x: IPv4Network(str(x[0])+'/'+str(x[1])).prefixlen, reverse=True)
```

构建转发表

通过调用 net.interfaces()和读入名为 .forwarding_table.txt 的转发表文件构建

最后按照子网掩码排序,保证按照顺序匹配转发表项时就是最长前缀匹配

```
def matches(self,destaddr):
    for item in self.forward_table:
        prefix=IPv4Address(item[0])
        mask=IPv4Address(item[1])
        if((int(mask)&int(destaddr))==int(prefix)):return item
    return None
```

通过实验手册提供的方法构建转发表项匹配

Task2：Forwarding the Packet and ARP

实现:

```python
def arp_handler(self):
    while (not self.end):
        try:reply_pkt =self.replyQueue.get(block=False)
        except queue.Empty:
            self.arp_requester()
            continue
        self.lock.acquire()
        arp_header=reply_pkt.get_header(Arp)
        src_ip=arp_header.senderprotoaddr
        src_mac=arp_header.senderhwaddr
        try:del self.ArpWaitingList[src_ip]
        except KeyError:pass
        for i in range(self.requestQueue.qsize()):
            packet=self.requestQueue.get()
            if(packet[3]==src_ip):
                packet[6][Ethernet].dst=src_mac
                packet[6][Ethernet].src=self.mac_list[self.port_list.index(packet[4])]
                packet[6][IPv4].ttl-=1
                self.send_queue.put([packet[4],packet[6]])
            else:
                self.requestQueue.put(packet)
        self.lock.release()
        self.arp_requester()
```

函数 arp_handler()用于处理接收的 arp 回应包,若 arp 包的发出地址与下一跳相同,代表下一跳收到了 arp 的回应,则将对应的 IP 包发出

```python
def arp_requester(self):
    toDelete=[]
    if(self.ArpWaitingList=={}):return
    self.lock.acquire()
    for key,value in self.ArpWaitingList.items():
        if time.time()-value[0]>=1.5:
            if value[1]>=5:
                for i in range(self.requestQueue.qsize()):
                    tmp=self.requestQueue.get(block=False)
                    if(tmp[3]!=key):
                        self.requestQueue.put(tmp)
                toDelete.append(key)
            else:
                self.ArpWaitingList[key][0]=time.time()
                self.ArpWaitingList[key][1]+=1
                forward_info=self.matches(key)
                arp_request_packet = Ethernet(src=self.mac_list[self.port_list.index(forward_info[3])],dst='ff:ff:ff:ff:ff
                self.send_queue.put([forward_info[3], arp_request_packet])
    for item in toDelete:
        del self.ArpWaitingList[item]
    self.lock.release()
```

函数 arp_resquester 用于发出 arp 请求,从字典中读取待请求 arp 的包,并遵循 1s 一次,最多 5 次的规则

```python
def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
    timestamp, ifaceName, packet = recv
    # TODO: your logic here
    arp=packet.get_header(Arp)
    icmp=packet.get_header(ICMP)
    eth=packet.get_header(Ethernet)
    udp=packet.get_header(UDP)

    if eth.dst not in self.mac_list and eth.dst!='ff:ff:ff:ff:ff:ff':return
    if eth.dst!='ff:ff:ff:ff:ff:ff' and self.port_list[self.mac_list.index(eth.dst)]!=ifaceName:return
    if packet[Ethernet].ethertype == EtherType.VLAN:return
```

函数 handle_packet 用于处理收到的所有包,这里先过滤一些无用的包

```python
if arp:
    if arp.targetprotoaddr in self.ip_list:
        self.lock.acquire()
        if arp.operation==ArpOperation.Request:
            self.arp_table[arp.senderprotoaddr] = [arp.senderhwaddr,timestamp]
            index=self.ip_list.index(arp.targetprotoaddr)
            reply_pkt=create_ip_arp_reply(self.mac_list[index],arp.senderhwaddr,arp.targetprotoaddr,arp.senderprotoaddr)
            self.send_queue.put([ifaceName,reply_pkt])
        elif arp.operation==ArpOperation.Reply:
           if (eth.dst != 'ff:ff:ff:ff:ff:ff') and (eth.src!='ff:ff:ff:ff:ff:ff'):
                forward_info = self.matches(arp.targetprotoaddr)
                if forward_info != None and forward_info[3]==ifaceName:
                    self.arp_table[arp.senderprotoaddr] = [arp.senderhwaddr, time.time()]
                self.replyQueue.put(packet)
        else :
            return
        self.lock.release()
```

若收到的包为 arp 包,则记录时间戳用于管理 arp 表,若为 arp 请求包则放入队列中,交给函数 handle_packet 处理,若为回应包则交给 arp_handler 处理

```python
elif icmp or udp:
    if 14 + packet[IPv4].total_length != packet.size():return
    if packet.get_header(IPv4).dst in self.ip_list:return
    if packet.get_header(Ethernet).dst in self.mac_list:
        forward_info = self.matches(packet.get_header(IPv4).dst)
        if forward_info != None:
            next_hop_ip = forward_info[2]
            if next_hop_ip == '0.0.0.0':next_hop_ip = packet.get_header(IPv4).dst
            if self.arp_table.get(next_hop_ip) == None:
                if next_hop_ip not in self.ArpWaitingList.keys():
                    self.lock.acquire()
                    self.ArpWaitingList[next_hop_ip] = [time.time()-10, 0]
                    self.lock.release()
                self.requestQueue.put([packet.get_header(IPv4).src, packet.get_header(IPv4).dst, packet.get_header(Ethernet).src, ne


                if next_hop_ip==IPv4Address('172.16.40.2') and self.ArpWaitingList[next_hop_ip][1]==5:
                    self.ArpWaitingList[next_hop_ip][0]-=10

            else:
                next_hop_mac = self.arp_table[next_hop_ip][0]
                packet[Ethernet].src = self.mac_list[self.port_list.index(forward_info[3])]
                packet[Ethernet].dst = next_hop_mac
                packet[IPv4].ttl -= 1
                self.send_queue.put([forward_info[3], packet])
```

若为 ip 包,则根据之前 arp 包的发送和处理情况决定是否转发

四、实验结果

testcase:

```
20  An IP packet from 192.168.1.239 for 10.10.50.250 should
    arrive on router-eth0.
21  Router should send an ARP request for 10.10.50.250 on
    router-eth1
22  Router should try to receive a packet (ARP response), but
    then timeout
23  Router should send an ARP request for 10.10.50.250 on
    router-eth1
24  Router should try to receive a packet (ARP response), but
    then timeout
25  Router should send an ARP request for 10.10.50.250 on
    router-eth1
26  Router should try to receive a packet (ARP response), but
    then timeout
27  Router should send an ARP request for 10.10.50.250 on
    router-eth1
28  Router should try to receive a packet (ARP response), but
    then timeout
29  Router should send an ARP request for 10.10.50.250 on
    router-eth1
30  Router should try to receive a packet (ARP response), but
    then timeout
31  Router should try to receive a packet (ARP response), but
    then timeout


All tests passed!
```

Advanced

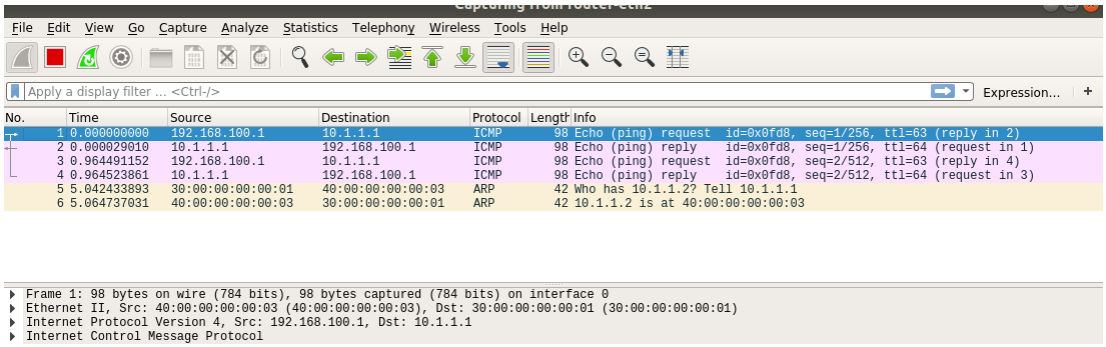
```
1187Ping request from 31.0.4.1 should arrive on eth4
1188Ping request from 31.0.4.1 should arrive on eth4
1189Ping request from 31.0.4.1 should arrive on eth4
1190Ping request from 31.0.4.1 should arrive on eth4
1191Router should not do anything
1192Ping request from 31.0.5.1 should arrive on eth5
1193Ping request from 31.0.5.1 should arrive on eth5
1194Ping request from 31.0.5.1 should arrive on eth5
1195Ping request from 31.0.5.1 should arrive on eth5
1196Ping request from 31.0.5.1 should arrive on eth5
1197Ping request from 31.0.5.1 should arrive on eth5
1198Router should not do anything
1199Ping request from 31.0.6.1 should arrive on eth6
1200Ping request from 31.0.6.1 should arrive on eth6
1201Ping request from 31.0.6.1 should arrive on eth6
1202Ping request from 31.0.6.1 should arrive on eth6
1203Ping request from 31.0.6.1 should arrive on eth6
1204Ping request from 31.0.6.1 should arrive on eth6
1205Router should not do anything
1206Bonus: V2FybWluZyB1cA==
1207Bonus: V2FybWVkIHVw
1208Bonus: V2h1dCBkJyB5YSBob3BlIHQnIGZpbmQgaGVyZT8=
1209Bonus: SGFsZndheQ==
1210Bonus: Tm90aGluJyBmb3IgeWEgdCcgZmluZCBoZXJlIQ==
1211Bonus: Q29uZ3JhdHMh


All tests passed!
```

部署情况:

输入命令:client ping -c2 192.168.100.1

监听 router-eth2 端口:



五、总结与感想

彻底理清了路由器转发的逻辑并且学会通过队列传递数据,实现多线程执行