南京大学本科生实验报告

- 课程名称：**计算机网络** 任课教师：田臣/李文中
  助教：

学院	计算机科学与技术系	专业（方向）	计算机科学与技术
学号	201220102	姓名	武雅琛
Email	201220102@smail.nju.edu.cn	开始/完成日期	2022.4.19

1. 实验名称

Forwarding Packet

2. 实验目的

- **进一步熟悉Switchyard框架。**

  通过自行学习Switchyard相关**API的实现和接口**进一步熟悉lab提供的Switchyard框架。
- **了解IPv4网络ARP请求分组的结构和功能**

  根据实验手册了解ARP请求分组的结构和功能，并且在实验中适当的时机构造正确的ARP请求分组从端口发送出去。
- **了解路由器静态转发表的构造和转发功能**

  了解路由器基于目的IP地址前缀匹配的转发表的构造和查找，在实验中实现对IPv4包头解析并查询转发表实现转发数据报的功能。

3. 实验内容

task 1:Preparation

task 2:IP Forwarding Table Lookup

- 这一步要求路由器实现路由器最基础和结构和功能：**建立、维护并查找**路由转发表。
- **建立转发表**

  根据需求，转发表需要每一个记录中包括四个表项：**IPv4地址前缀，子网掩码，下一跳IP地址和转发端口名称。**

  在**逻辑**上转发表结构如下：

address prefix	netmask	next hop	intf
192.168.1.0	255.255.255.0	192.168.2.1	eth-0
192.168.1.128	255.255.255.128	192.168.3.0	eth-1
...	...	...	...

在实现中，选用**列表**的数据结构来组织转发表，同时转发表作为一个**单独的类**定义了和它匹配的一组操作，向上提供操作接口，有利于实现的抽象和封装。

另外，为方便数据处理的一致性，转发表的内容**统一转化为字符串**类型。

```
class forwarding_table(object):
```

- **维护转发表**

  本实验中使用静态前缀转发表，在路由器运行的过程中转发表不发生更新，所以只需要完成一个任务：**初始化转发表表项。**

  转发表表项有两个来源：路由器**端口提供的子网信息**和同目录下的`forwarding_table.txt`**文件**提供的**文本内容。**

  - **端口关联的子网信息**：

    可以通过interface类得到端口的ip地址和子网掩码，对于从端口生成的表项，直接将nexthop设置为`0.0.0.0`，也就是**从端口直接抛出**就可以，**默认目的主机**恰好在端口的**下一跳。**

    比较复杂的是**没有直接给出端口的IP地址前缀**，需要将ip地址和子网掩码运算后得到地址前缀填入转发表。

    ```
    IPv4addr = intf.ipaddr
    mask = intf.netmask
    #compute the IPv4address prefix
    prefix = IPv4Address(int(IPv4addr) & int(mask))
    ```

  - **同目录下的文本文件**：

    只需要通过python提供的文件操作的API以及字符串分割的库函数即可生成表项。并不复杂，不赘述。

- **查找转发表：**

  在列表中遍历表项，通过IP地址和子网掩码构造IPv4网络得到`prefixnet`，判断查询的IP地址是否包含在其中即可。

```
#myrouter.py
prefixnet =IPv4Network(format(entry[0]+ '/' + entry[1]),strict=False)
    if destaddr in prefixnet:
```

特别需要注意的是，我们要求路由器可以在转发表中匹配和目的IP地址**匹配的前缀最长的表项**，需要特殊处理。

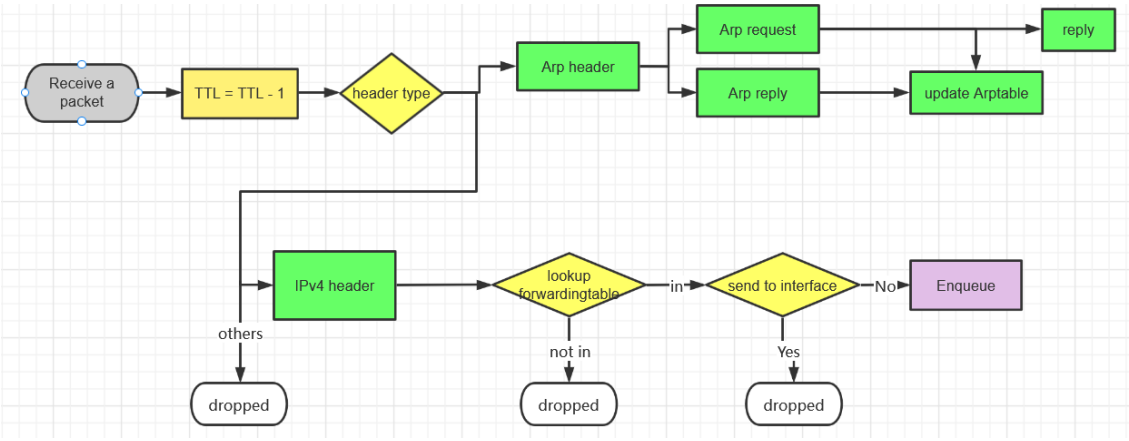
我采用了**记录**当前匹配的表项的**前缀长度**的形式，如果发生了更长的匹配，那么匹配结果更新。

```
 if prefixnet.prefixlen > prefixlen:
        #update for the longer prefixlen
      prefixlen = prefixnet.prefixlen
      nexthop = entry[2]
      forward_intf = entry[3]
```

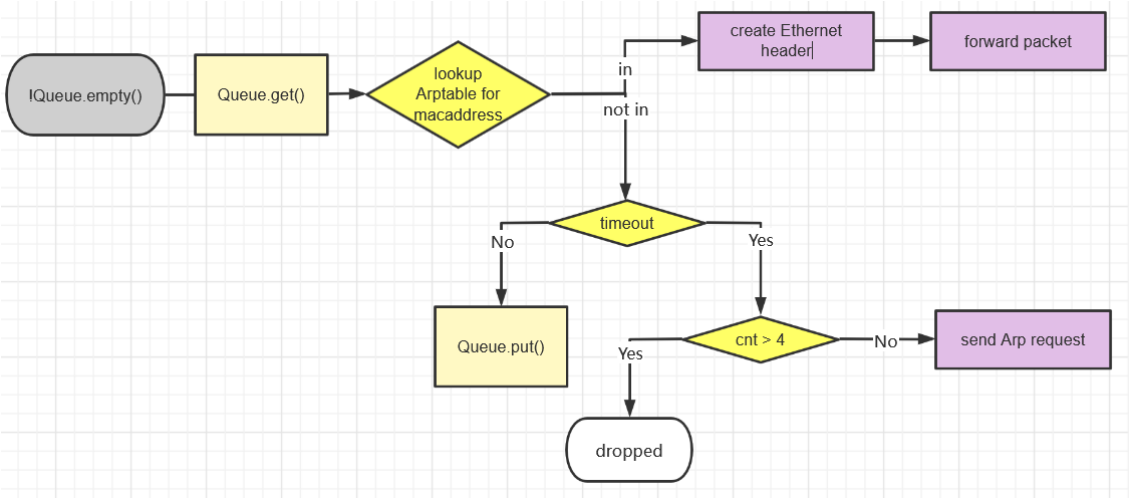
task 3:Forwarding the Packet and ARP

- 在建立好ARP高速缓存表和路由转发表后，我们接下来只需要在适当的时候**发送ARP请求分组**和并且**转发数据包**就可以实现路由器的基础功能转发了。
- 因为我们**不能忍受**路由器发送ARP请求分组后**阻塞**其它任务等待回复，所以我选择了**双线程**实现任务。

  主线程任务流程图：（主要负责**分组接收，初步判断后放入队列**，出于队列元素的一致性，我将回复ARP请求的部分亦在主线程完成）



  子线程任务流程图：（在实验中子线程负责数据包的发出并且执行在`wait_queue_operation`函数上）



4. 实验结果

task 2:IP Forwarding Table Lookup

- 通过**输出日志**的形式可以看到最终的路由转发表的各项的内容和数据类型：

```
20:19:13 2022/04/20     INFO ipaddr:192.168.1.0/<class 'str'>,netmask:255.255.255.0/<class 'str'>,next hop:0.0.0.0/<class 'str'>,intfname:router-eth0/<class 'str'>)
20:19:13 2022/04/20     INFO ipaddr:10.10.0.0/<class 'str'>,netmask:255.255.0.0/<class 'str'>,next hop:0.0.0.0/<class 'str'>,intfname:router-eth1/<class 'str'>)
20:19:13 2022/04/20     INFO ipaddr:172.16.42.0/<class 'str'>,netmask:255.255.255.252/<class 'str'>,next hop:0.0.0.0/<class 'str'>,intfname:router-eth2/<class 'str'>)
20:19:13 2022/04/20     INFO ipaddr:172.16.0.0/<class 'str'>,netmask:255.255.0.0/<class 'str'>,next hop:192.168.1.2/<class 'str'>,intfname:router-eth0/<class 'str'>)
20:19:13 2022/04/20     INFO ipaddr:172.16.128.0/<class 'str'>,netmask:255.255.192.0/<class 'str'>,next hop:10.10.0.254/<class 'str'>,intfname:router-eth1/<class 'str'>)
20:19:13 2022/04/20     INFO ipaddr:172.16.64.0/<class 'str'>,netmask:255.255.192.0/<class 'str'>,next hop:10.10.1.254/<class 'str'>,intfname:router-eth1/<class 'str'>)
20:19:13 2022/04/20     INFO ipaddr:10.100.0.0/<class 'str'>,netmask:255.255.0.0/<class 'str'>,next hop:172.16.42.2/<class 'str'>,intfname:router-eth2/<class 'str'>)
```

task 3:Forwarding the Packet and ARP

Testing

```
    from a recent ARP request should be cached)
11  IP packet to be forwarded to 192.168.1.100 should arrive on
    router-eth2
12  IP packet should be forwarded to 192.168.1.100 out router-
    eth0 (again, no ARP request should be necessary since the
    information from a recent ARP request should be cached)
13  An IP packet from 10.100.1.55 to 172.16.64.35 should arrive
    on router-eth1
14  Router should send an ARP request for 10.10.1.254 on router-
    eth1
15  Application should try to receive a packet, but then timeout
16  Router should send another an ARP request for 10.10.1.254 on
    router-eth1 because of a slow response
17  Router should receive an ARP response for 10.10.1.254 on
    router-eth1
18  IP packet destined to 172.16.64.35 should be forwarded on
    router-eth1
19  An IP packet from 192.168.1.239 for 10.200.1.1 should arrive
    on router-eth0.  No forwarding table entry should match.
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!
```

Deploying

- 创建默认的Mininet拓扑网络。
- 在Mininet中输入以下命令。

```
mininet> server1 ping -c2 192.168.200.1 #server1 ping -c2 (server2)
```

在router-eth1端口打开wireshark捕获到分组：（和server2在一个子网的分组）

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	40:00:00:00:00:02	Broadcast	ARP	42	Who has 192.168.200.1? Tell 192.168.200.2
2	0.000024828	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	192.168.200.1 is at 20:00:00:00:00:01
3	0.934081375	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x0be8, seq=2/512, ttl=63 (reply in 4)
4	0.934121052	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0be8, seq=2/512, ttl=64 (request in 3)
5	2.041618626	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x0be8, seq=1/256, ttl=63 (reply in 6)
6	2.041643767	192.168.200.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x0be8, seq=1/256, ttl=64 (request in 5)
7	5.972537649	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	Who has 192.168.200.2? Tell 192.168.200.1
8	6.046633268	40:00:00:00:00:02	20:00:00:00:00:01	ARP	42	192.168.200.2 is at 40:00:00:00:00:02

- 按照时间顺序，第一组ARP分组是路由器从eth1发出**ARP请求**目的IP地址的mac地址，并且收到了回复。
- 接下来是两组ICMP分组，说明路由器是实现了完整的**转发功能**，并且在经过时**跳数减一。**
- 最后应该时**触发了server2的ARP确认机制**，server2向eth-1发送ARP请求分组，并且受到了正确的回复。

5.核心代码

task 2:IP Forwarding Table Lookup

```
# myrouter.py
#转发表类的实现
class forwarding_table(object):
    #初始化对象
    def __init__(self, forwarding_table, net):
        self.forwarding_table = forwarding_table
        my_interface = net.interfaces()
        # fill forwardingtable with the interfaces of router
        for intf in my_interface:
            IPv4addr = intf.ipaddr
            mask = intf.netmask
            #compute the IPv4address prefix
            prefix = IPv4Address(int(IPv4addr) & int(mask))
            # print(f"{prefix}, {type(prefix)}")

            forwarding_table.append([format(prefix),
format(intf.netmask),'0.0.0.0',intf.name])

        # fill forwardingtable with the forwardingtable file
        file = open('forwarding_table.txt','r')
        text = file.readlines()
        for entry in text:
            cells = entry.split()
            ipaddr = cells[0]
            netmask = cells[1]
            nexthop = cells[2]
            name = cells[3]

            forwarding_table.append([ipaddr, netmask, nexthop, name])

        for entry in forwarding_table:
            log_info(f"ipaddr:{entry[0]}/{type(entry[0])},netmask:
{entry[1]}/{type(entry[1])},next hop:{entry[2]}/{type(entry[2])},intfname:
{entry[3]}/{type(entry[3])}})")#print the forwarding_table
    #查找转发表
    def lookup(self,ipaddr):
        prefixlen = 0
        nexthop = None
```

```
        forward_intf = None
        destaddr = IPv4Address(ipaddr)
        for entry in self.forwarding_table:#遍历转发表
            #根据前缀和子网掩码得到用于判断的子网
            prefixnet =IPv4Network(format(entry[0]+ '/' +
entry[1]),strict=False)
            if destaddr in prefixnet:
                log_info(f"{destaddr},{entry}")
                if prefixnet.prefixlen > prefixlen:
                    #update for the longer prefixlen
                    prefixlen = prefixnet.prefixlen
                    nexthop = entry[2]
                    forward_intf = entry[3]
        log_info(f"prefixlen:{prefixlen},nexthop:{nexthop}")
        return prefixlen, nexthop, forward_intf
```

task 3:Forwarding the Packet and ARP

主线程:

```
# myrouter.py
#主线程处理接收到的包
 if IPv4:
       IPv4.ttl = IPv4.ttl - 1
       prefixlen, nexthop, forward_intf =
self.forwarding_table.lookup(format(IPv4.dst))
       if prefixlen != 0:#exist entry in forwarding_table
              if IPv4.dst not in my_ip:# not send to interfaces of router
                    if (nexthop == '0.0.0.0'):# nexthop := 0 -> nextip =
dstipaddr
                         nextip = format(IPv4.dst)
                    else:
                         nextip = nexthop
                   # destmacaddr = self.arptable.lookup(nextip)
                    log_info("receive a new packet needed be forwarded")
                    #add to the queue to be send out from the router
                    self.wait_queue.put([packet, nextip, forward_intf, 0,
time.time()])

```

子线程:

```
#myrouter.py
#子线程主要负责发出arp请求分组和转发分组
    def wait_queue_operation(self, wait_queue, arptable):
        while True:
            if not wait_queue.empty():#队列不为空，说明存在数据包需要处理

                entry = wait_queue.get()
                #查找arp缓存表
                destmacaddr = self.arptable.lookup(entry[1])
                if destmacaddr:#目标mac地址存在，发出数据包
                    self.forward_packet(entry[0], entry[2], destmacaddr)
                else:
                    #log_info(f"{entry[4]}, {time.time()}")
                    if entry[3] == 0:#first request，直接发出Arp请求

```

```
                        self.send_Arp_request(entry[1], entry[2])
                        time.sleep(0.2)#发包以后阻塞0.2s，接收对应的reply
                        entry[3] = entry[3] + 1
                        entry[4] = time.time()
                        wait_queue.put(entry)
                    else:
                        if entry[4] + 1 < time.time():#timeout
                            if entry[3] < 5:#发送Arp请求次数未超限
                                self.send_Arp_request(entry[1], entry[2])
                                time.sleep(0.2)#发包以后阻塞0.2s，接收对应的
reply
                                entry[3] = entry[3] + 1#计数自增
                                entry[4] = time.time()#更新上一次发送Arp请求时间
                                wait_queue.put(entry) #加入队列继续等待
                        else:#time < 1.不做处理，继续加入队列等待
                             wait_queue.put(entry)
```

6.总结与感想

- 在本实验中熟悉了路由器在三层的功能，对前缀匹配有了更深的理解。
- 在本实验尝试了非常简单的多线程编程实践，提高了编程能力。~~对操作系统的学习起到了促进作用~~。