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**	专业（方向）	
学号	**215220008**	姓名	**房艾婕**
Email	**3526549727@qq.com**	开始/完成日期	

1. 实验名称

实验 4：Forwarding Packets

2. 实验目的

Forwarding Packet

3. 实验内容

Task 1: Preparation

Task 2: IP Forwarding Table Lookup

Task 3: Forwarding the Packet and ARP

4. 实验结果&核心代码

Task 2:

```
def Found_Matching(self,dest):
    max_macth=0
    curr_match=0
    nxt_hop_intf=[]#[nexthop,out_intf]
    for x in self.forwardingtable:
        networkaddr=IPv4Network('{}/{}'.format(x[0],x[1]))
        matches=(int(networkaddr.netmask)&int(dest))==int(networkaddr.network_address)
        if matches:
            curr_match=networkaddr.prefixlen
            if curr_match>max_macth:
                max_macth=curr_match
                nxt_hop_intf.clear()
                nxt_hop_intf.append(x[2])
                nxt_hop_intf.append(x[3])

    return nxt_hop_intf
```

-在 Found_Matching 函数里把 forwardingtable 中所有条目的地址与 ipdest 进行比较，同时判断是否为最长匹配

Task 3:

```
def check_queue(self):
    print("")
    print("CHECK_QUEUE")
    max_count_reached=[]
    for k in self.queue.keys():
        if time.time() - self.queue[k][1] >= 1.0:
            if self.queue[k][2] == 0:
                max_count_reached.append(k)
                continue
            senderhwaddr=self.net.interface_by_name(self.queue[k][3]).ethaddr
            senderprotoaddr=self.net.interface_by_name(self.queue[k][3]).ipaddr
            arp_request=create_ip_arp_request(senderhwaddr,senderprotoaddr,k)
            print("SEND ARP_REQ")
            print(type(k))
            self.net.send_packet(self.queue[k][3],arp_request)
            self.queue[k][1]=time.time()
            self.queue[k][2]-=1

    for k in max_count_reached:
        ##got something to do in lab 5
        del self.queue[k]
    print("END CHECK_QUEUE")
```

-每次新的循环开始，都会进入 check_queue 函数，检查是否已经收到 ARP 回复

-如果该包下一跳地址上次的 ARP 请求与现下时间差超过 1 秒，则不做任何操作

-如果该包下一跳地址的请求次数未达上限（已规定上限为 5 次），重新发 ARP 请求

-否则，丢弃该包

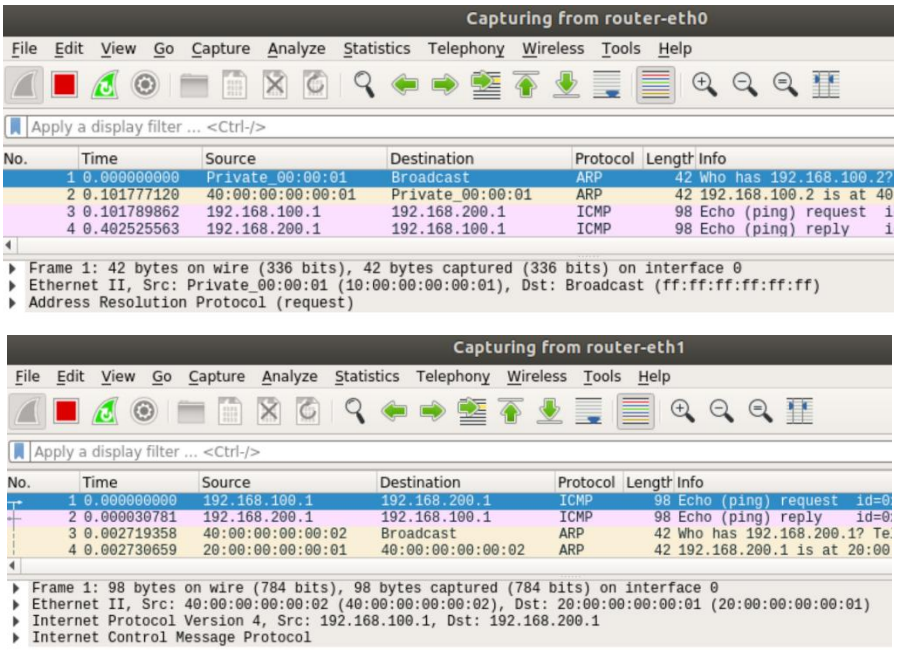
```
def handle_packet(self,recv:switchyard.llnetbase.ReceivedPacket):
    timestamp,ifaceName,packet=recv
    #debugger()

    #check the pkt dst MAC Address is in the intf of this router or not
    check=False
    for i in self.net.interfaces():
        if i.ethaddr == packet[Ethernet].dst:
            check=True
            break
    #if yes or MAC Address is broadcast,do it
    #else don't do anythinf
    if check or packet[Ethernet].dst=="ff:ff:ff:ff:ff:ff":
        pass
    else:
        return
    if packet[Ethernet].ethertype == 33024:
        return
```

-路由器接收包后，会判断其包含的是 ARP 头部还是 IPv4 头部，并进入对应分支进行处理

结果：

```
1207Bonus: V2FybWVkIHVw
1208Bonus: V2h1dCBkJyB5YSBob3BlIHQnIGZpbmQgaGVyZT8=


Failed:
    Bonus: SGFsZndheQ==
        Expected event: Timeout after 1.2s on a call to recv_packet


Pending (couldn't test because of prior failure):
1   Bonus: Tm90aGluJyBmb3IgeWEgdCcgZmluZCBoZXJlIQ==
2   Bonus: Q29uZ3JhdHMh


************************************************************
Your code didn't crash, but a test failed.
************************************************************
```

```
This is the Switchyard equivalent of the blue screen of death.
As far as I can tell, here's what happened:

    Expected event:
        Bonus: SGFsZndheQ==

    Failure observed:
        send_packet was called instead of Timeout after 1.2s on a
        call to recv_packet

You can rerun with the -v flag to include full dumps of packets that
may have caused errors. (By default, only relevant packet context may
be shown, not the full contents.)
```



-执行 server1 ping -c2 server2

-由于一开始 server1 不知道 server2 的 MAC 地址，所以发送 ARP request 得出 server2 的 MAC 地址

-得到回复后，就能陆续发送 ping 了