计网实验报告

实验名称：**Forwarding Packets**

实验步骤：

TASK TWO：

采用字典构建的转发表，关键词为 IP 地址与"/"和网络掩码

```
    def init_forward(self):
        for intf in self.my_inft:
            intf_ip=intf.ipaddr
            intf_mask=intf.netmask
            NextHop = '0.0.0.0'
            intf_port=intf.name
            prefix=IPv4Address(int(intf_ip)&int(intf_mask))
            net_addr=IPv4Network(str(prefix)+'/'+str(intf_mask))
            self.forward_table[net_addr]=[IPv4Address(NextHop),intf_port]
            file= open("forwarding_table.txt","r")
            for line in file:
                temp = line.split()
                if temp:
                    self.forward_table[IPv4Network(temp[0]+'/'+temp[1])]=[temp[2],temp[3]]
```

```
                if packet[1].dst == interface.ipaddr:#目的地址为路由器接口，忽略
                    flag = False
                    break
            if flag:
#-------------------------------------------------------------------------------------------------------------------#
                maxlen = 0
                for context in self.forward_table.keys():
                    if packet[1].dst in context:
                        if context.prefixlen > maxlen:                    #在转发表中进行最大匹配
                            maxlen = context.prefixlen
                            text = self.forward_table[context]
#-------------------------------------------------------------------------------------------------------------------#
                if maxlen !=0:
                    packet[1].ttl-=1
                    if text[0]!=IPv4Address('0.0.0.0'):#其为路由器上可到达的地址
                        nexthop = IPv4Address(text[0])
                    else:
                        nexthop = packet[1].dst#此时下一跳IP就期待为目的ip，直接填入packet[1].dst
                    name = text[1]
```

在匹配的过程中我们采用了最大长度匹配，初始化一个 maxlen 为 0，然后对装发表的关键词进行遍历，判断数据包的 dst 是否前缀匹配，并通过库函数得出长度 prefixlen，与 maxlen 比较，如果比 maxlen 大，则更新 maxlen，并且记录下该转发表项，最后我们就可以找到与其最大匹配的转发表项。

TASK THREE

```
ipv4=packet.get_header(IPv4)
if ipv4 and packet[1].ttl>0:
    flag=True
    for interface in interfaces:
        if packet[1].dst == interface.ipaddr:#目的地址为路由器接口，忽略
            flag = False
            break
    if flag:
--------------------------------------------------------------------------------------------------------------------------------#
        maxlen = 0
        for context in self.forward_table.keys():
            if packet[1].dst in context:
                if context.prefixlen > maxlen:                          #在转发表中进行最大匹配
                    maxlen = context.prefixlen
                    text = self.forward_table[context]
--------------------------------------------------------------------------------------------------------------------------------#
        if maxlen !=0:
            packet[1].ttl-=1
            if text[0]!=IPv4Address('0.0.0.0'):#其为路由器上可到达的地址
                nexthop = IPv4Address(text[0])
            else:
                nexthop = packet[1].dst#此时下一跳IP就期待为目的ip，直接填入packet[1].dst
            name = text[1]
            for interface in interfaces:
                if interface.name == name:
                    packet[0].src = interface.ethaddr
                    self.waitinglist.append([packet,interface,nexthop,0,0,False]) #分别表示数据包，对应接口，ip地址，请求次数，请求
                    break
```

首先判断其是否为需要转发的数据包，然后判断数据包的正确性，其目的地址如果是路由器的，直接丢弃，然后在转发表内进行最大长度匹配，找到对应的转发表项，然后转发的接口就是转发表中的接口，目的地址就是转发表中的下一跳地址，ps;如果下一跳地址为 0.0.0.0，则目的地址就直接设为转发数据包期待的目的地址，构造相应的数据包，并将原 MAC 地址转变为该接口的 MAC 地址，然后将其与其他一些特性加入等待转发序列

```
def clean_list(self):#根据时间更新缓存表对于缓存队列的表进行处理
    i = 0
    while i < len(self.waitinglist):
        flag = self.handle_list(i)
        if flag==1 :#已经成功发送，需要在队列中删除
            self.Todelete.append(i)
        i+=1
    n = len(self.Todelete)#对于处理完的和超过五次访问请求得要删除
    while n > 0:
        del self.waitinglist[self.Todelete[n-1]]
        del self.Todelete[n-1]
        n-=1
```

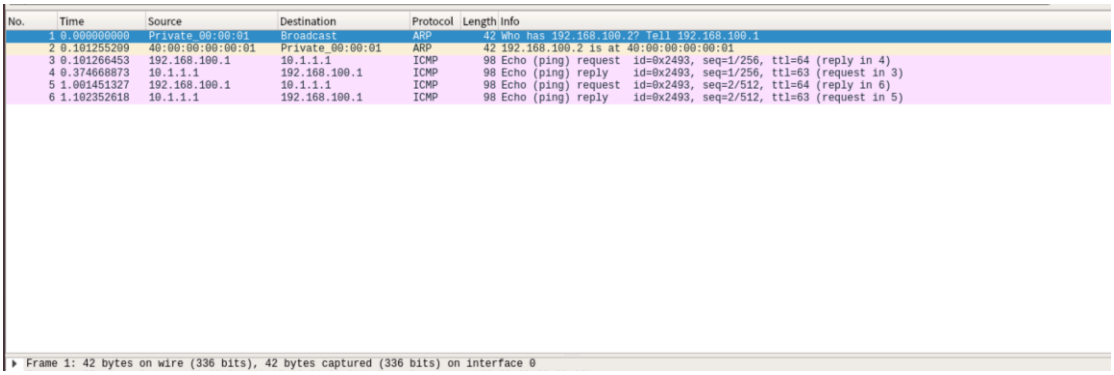
```
def handle_list(self,index):
    if self.waitinglist[index][5]==True :#超过五次,表明状态需要被丢弃
        return 1
    if self.waitinglist[index][2] in self.my_table.keys():#目的地址在arp缓存表内
        i = 0
        while i < len(self.sendlist):
            if self.sendlist[i]==self.waitinglist[index][2]:
                del self.sendlist[i]
            i+=1
        self.waitinglist[index][0][0].dst = self.my_table[self.waitinglist[index][2]][0]
        self.net.send_packet(self.waitinglist[index][1].name,self.waitinglist[index][0])
        return 1
    elif time.time() -self.waitinglist[index][4]>1:#距离上次请求超过一个时间单位

        if self.waitinglist[index][3]<5:
            if self.waitinglist[index][3]==0:
                tmp=0
                while tmp < len(self.sendlist):
                    if self.sendlist[tmp]==self.waitinglist[index][2]:#判断对应ip地址是否已经在处理一个数据包
                        return 0
                    tmp += 1
                self.sendlist.append(self.waitinglist[index][2])
            #packe = etherheader+arpheader
            packe=create_ip_arp_request(self.waitinglist[index][1].ethaddr, self.waitinglist[index][1].ipaddr,self.waitinglist[index][2])
            self.net.send_packet(self.waitinglist[index][1].name,packe)
            self.waitinglist[index][3]+=1
            self.waitinglist[index][4]=time.time()
            return 0  #重新请求并更新状态
        else:
            tmp1 =0
            while tmp1 < len(self.sendlist):
                if self.sendlist[tmp1]==self.waitinglist[index][2]:
                    del self.sendlist[tmp1]
                tmp1+=1
            tmp2 =index+1
            while tmp2 <len(self.waitinglist):
                if self.waitinglist[tmp2][2]==self.waitinglist[index][2]:
                    self.waitinglist[tmp2][5]=True#表明状态需要被丢弃
                tmp2+=1
            return 1
```

然后遍历等待序列中的转发数据包，判断目的地址是否在 ARP 缓存表中，如果在缓存表中，则就可以直接从其对应的接口发送，如果对应 ip 地址不在 ARP 缓存表中时，首先我们判断对于该 ip 地址的 ARP 请求是否超过 5 此，如果是这样则直接丢弃该数据包，否则的话，我们先判断距离上次发送时间是否超过 1s，如果没有不发 ARP 请求，如果有的话，则判断该 ip 地址是否已经被访问发送请求，是的话，则不做处理，没有的话将其加入，然后采用 lab3 中给出的函数构造请求包，向目标 ip 地址发送 ARP 请求。然后将成功转发的和需要丢弃的数据包直接丢弃。

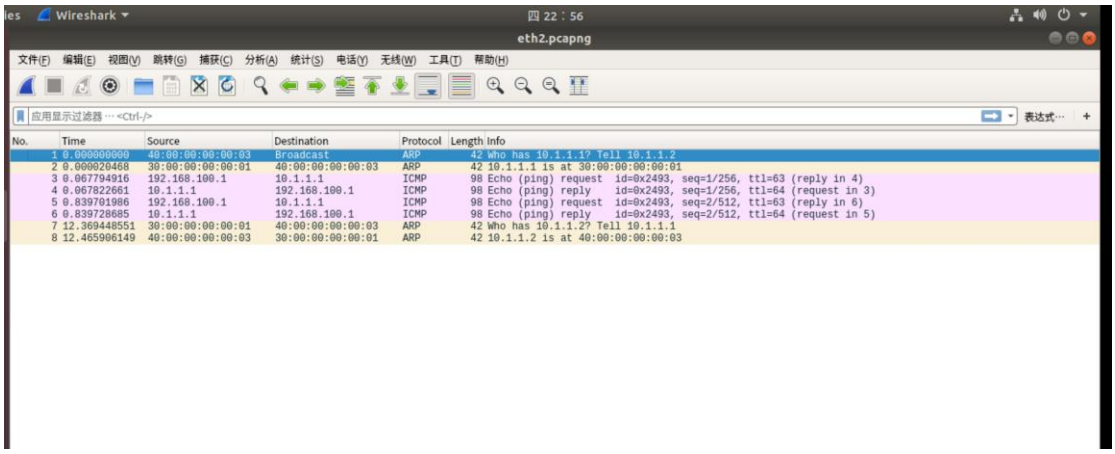
PART 2：

我采用的是从 server1 ping 到 client 来检测连接

Router-eth0



Router-eth2



首先我们可以看到 sever1 先发送 ARP 请求，与路由器建立连接，然后该 ping 的数据包最大 ip 匹配到接口 eth2，然后从该接口发送 ARP 请求查找目的 ip 地址，然后连接后从 server1 成功发送给了 client，然后 client 发送回复给