**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 | **191220154** | 姓名 | **张涵之** |
| Email | **1683762615@qq.com** | 开始/完成日期 | **2021/5/15-2021/5/16** |

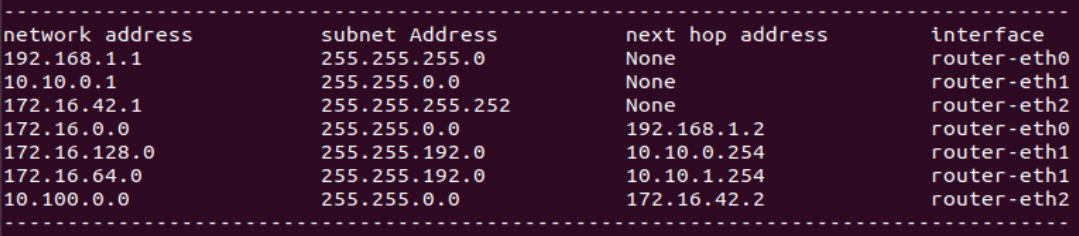
1. **实验名称：Lab 5: Respond to ICMP**
2. **实验目的：**

Respond to ICMP messages like echo requests ("pings").

Generate ICMP error messages when necessary.

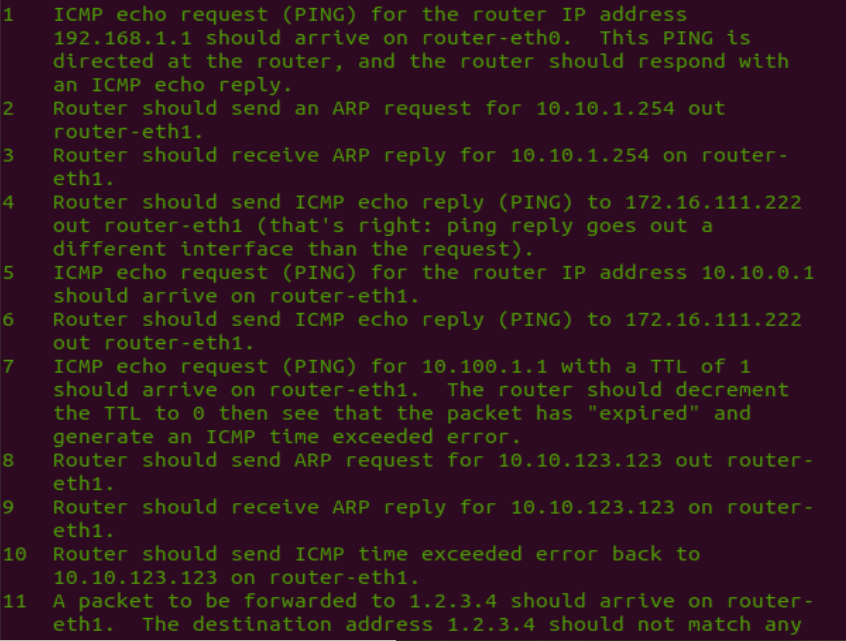
1. **实验内容**
   1. Task 2: Responding to ICMP echo requests
   2. Task 3: Generating ICMP error messages
2. **实验结果**
   1. Task 2 & Task 3 in test scenario:

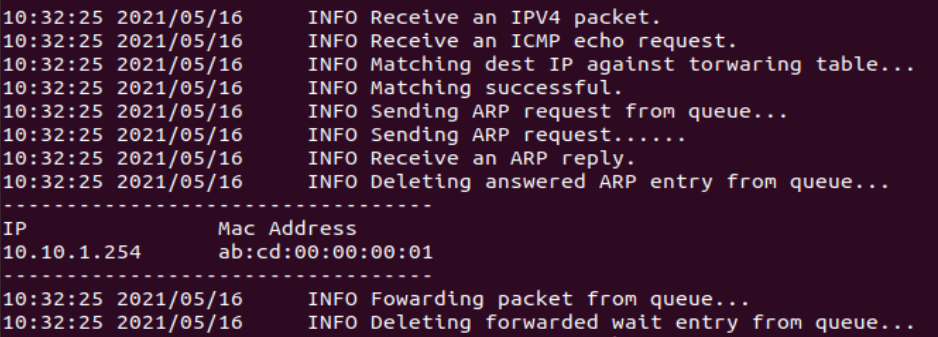
Forwarding table is shown as below:

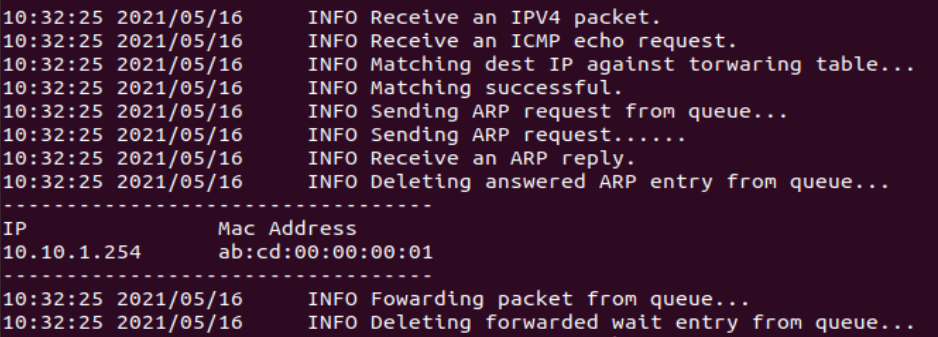


The first (set of) testcase involves replying to a ping request:

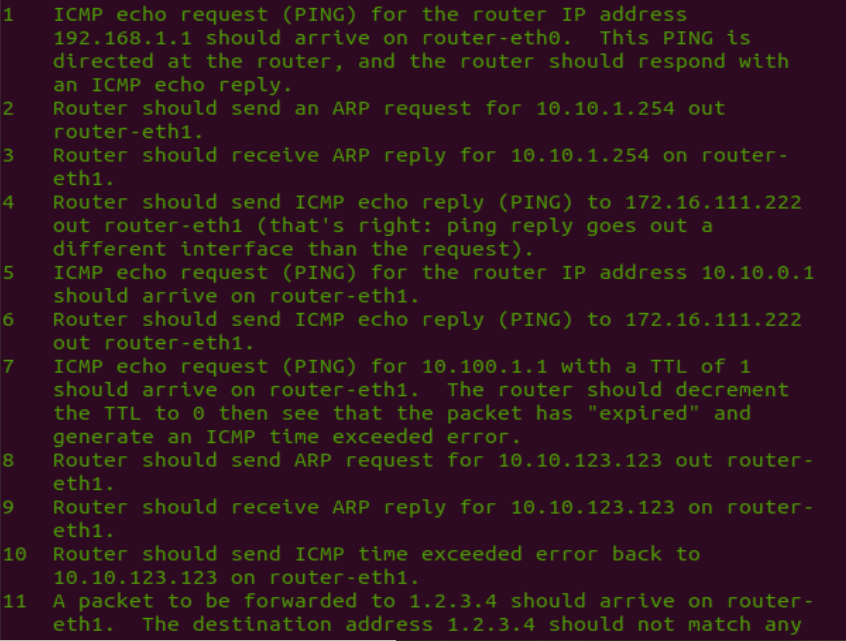
An ARP request is needed before the reply is sent.

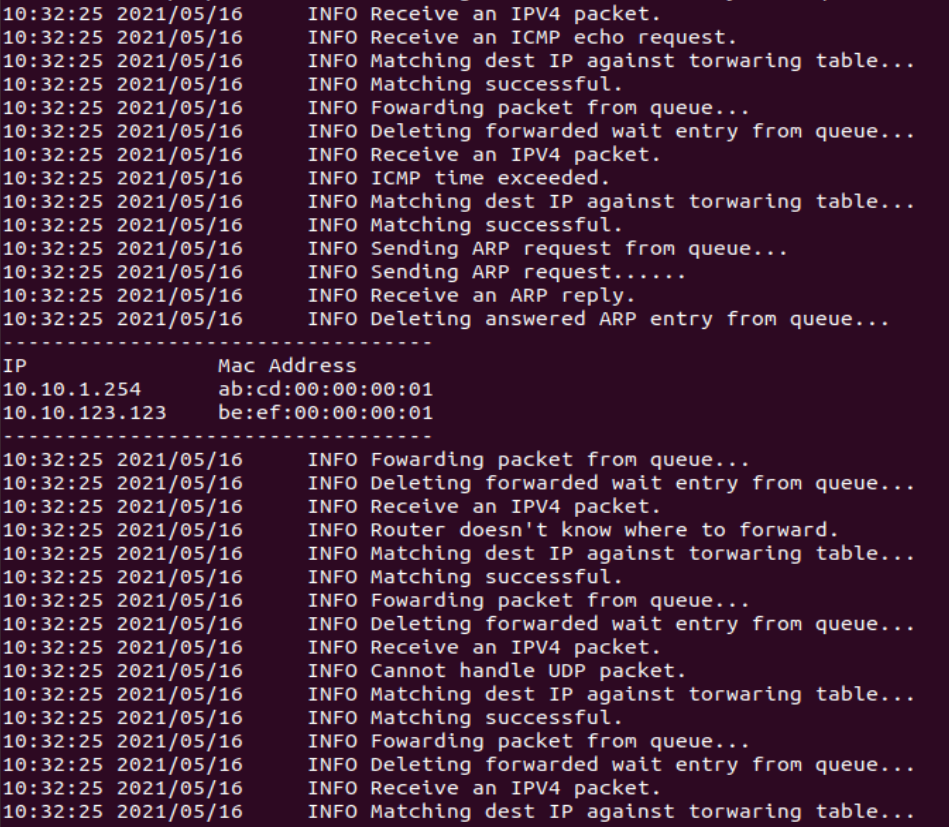






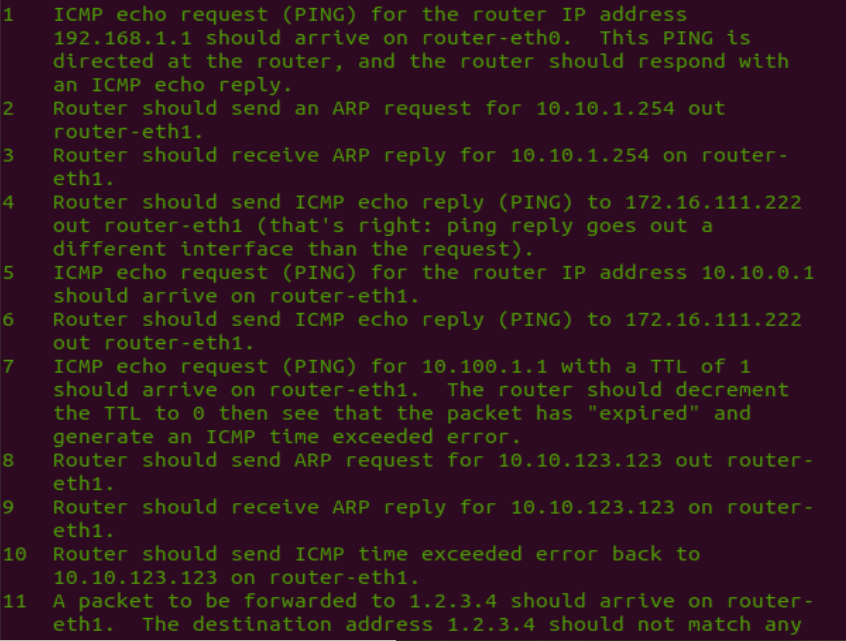
The second testcase involves replying to a ping request directly:

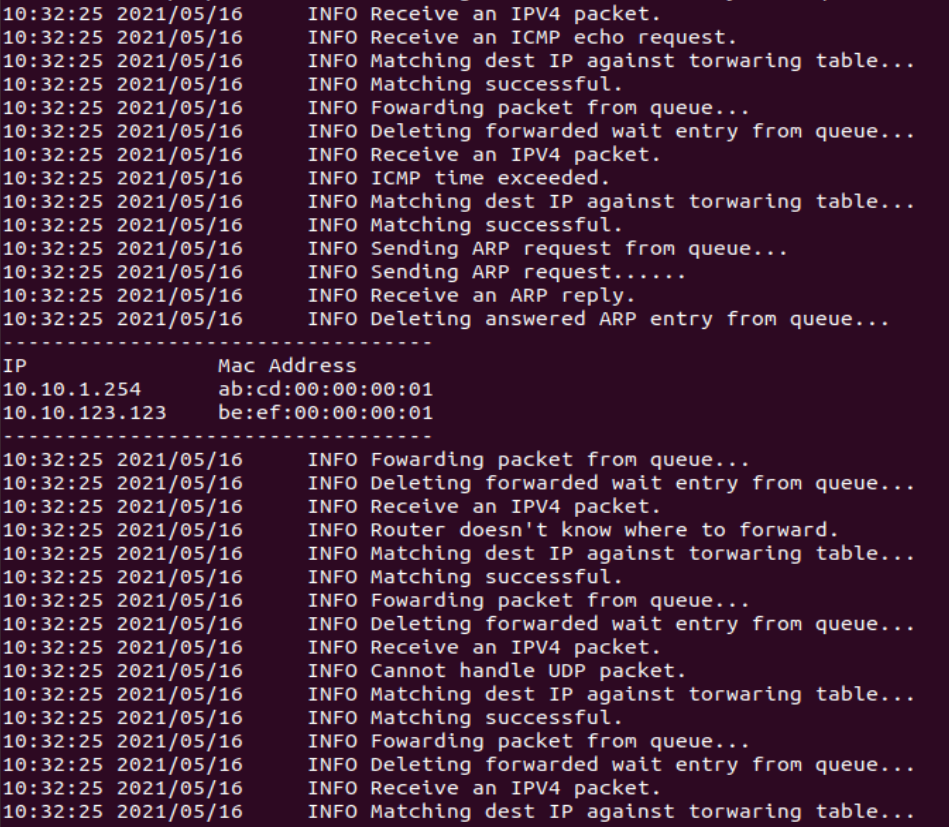


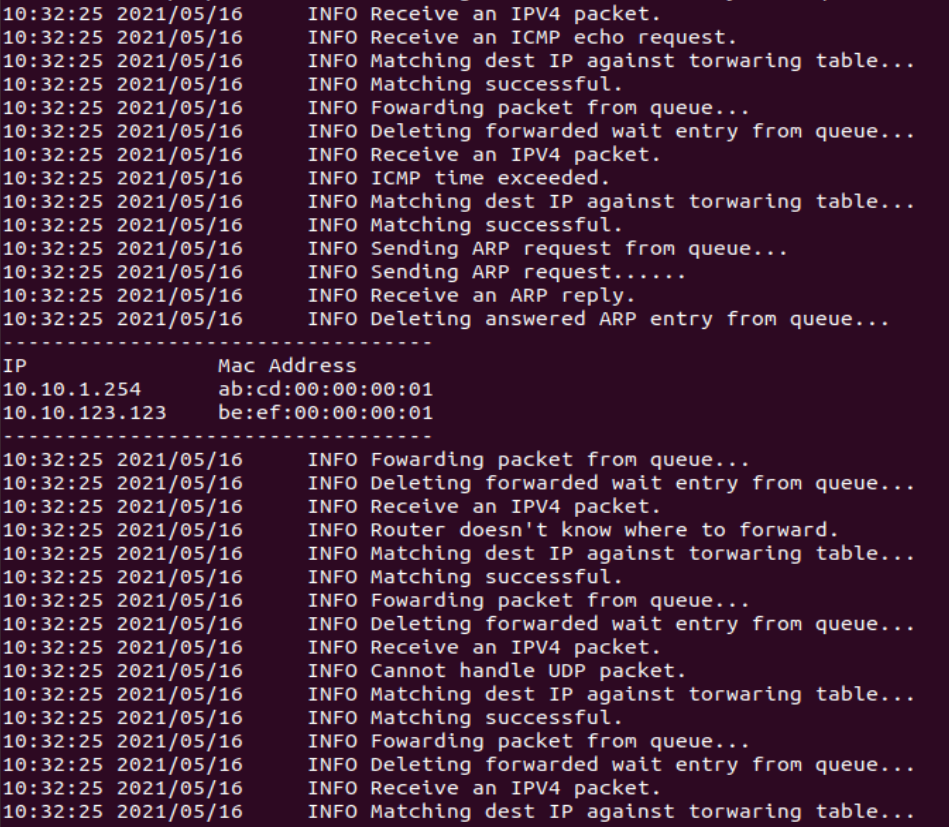


The third testcase handles a TTL exceeded ping request:

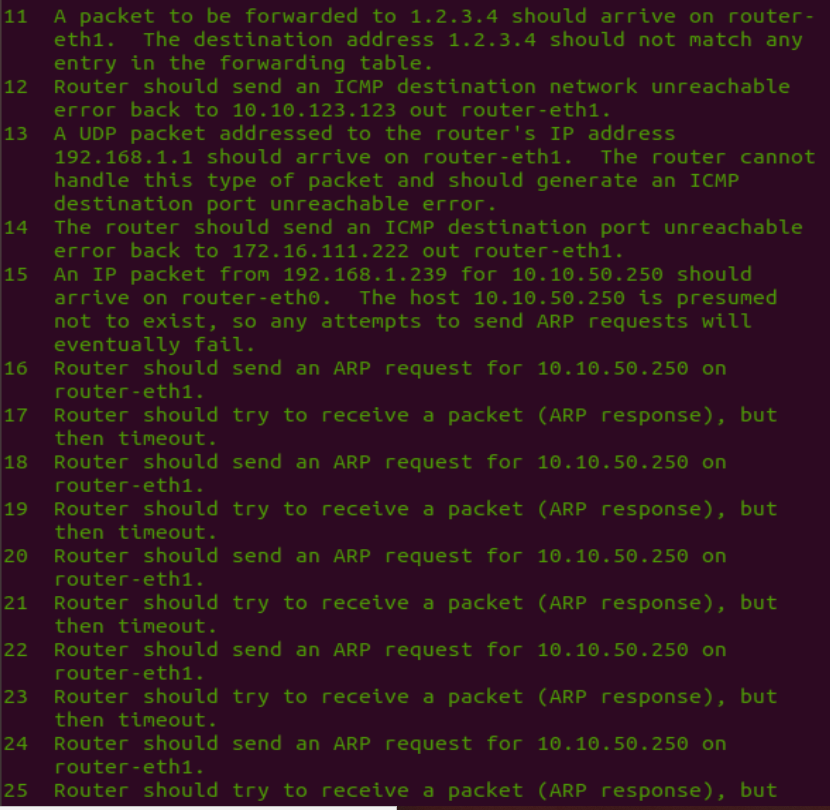
An ARP request is needed before the error message is sent.

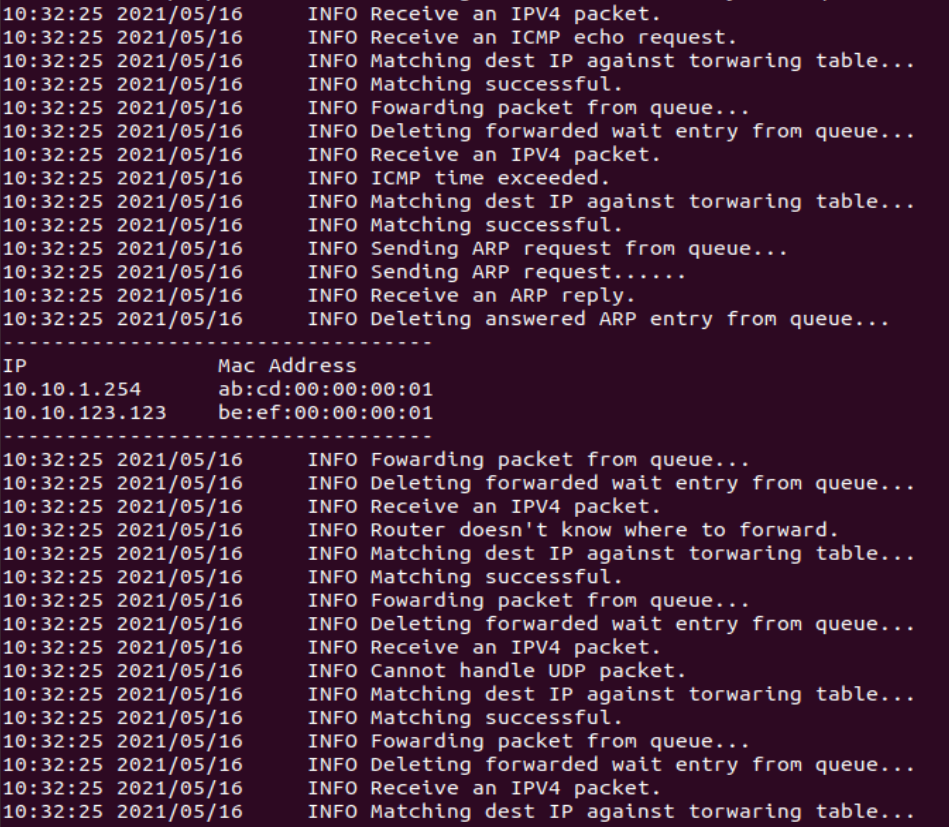




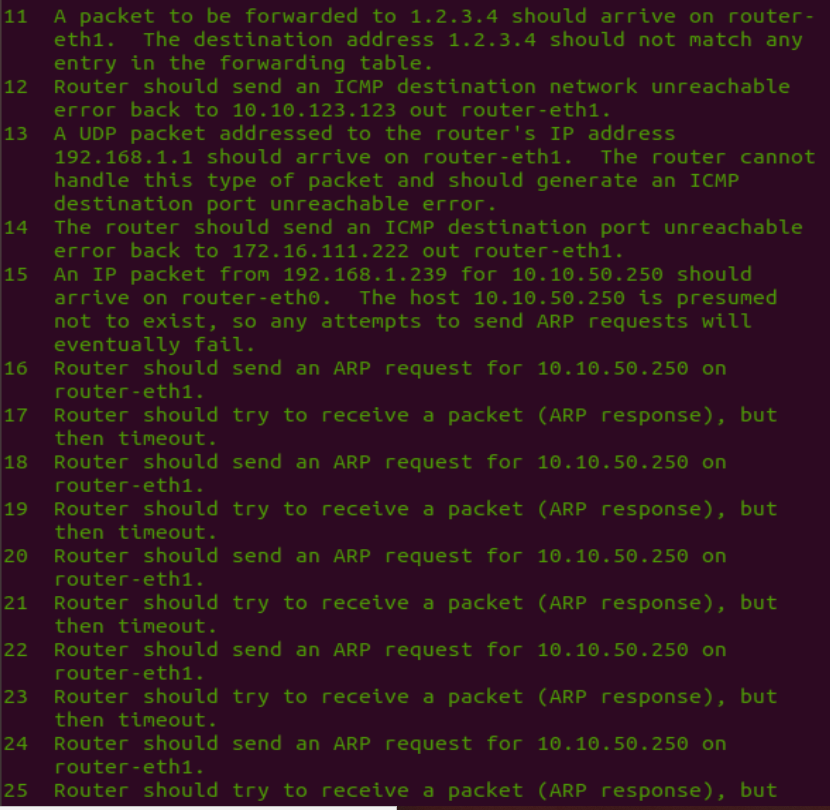


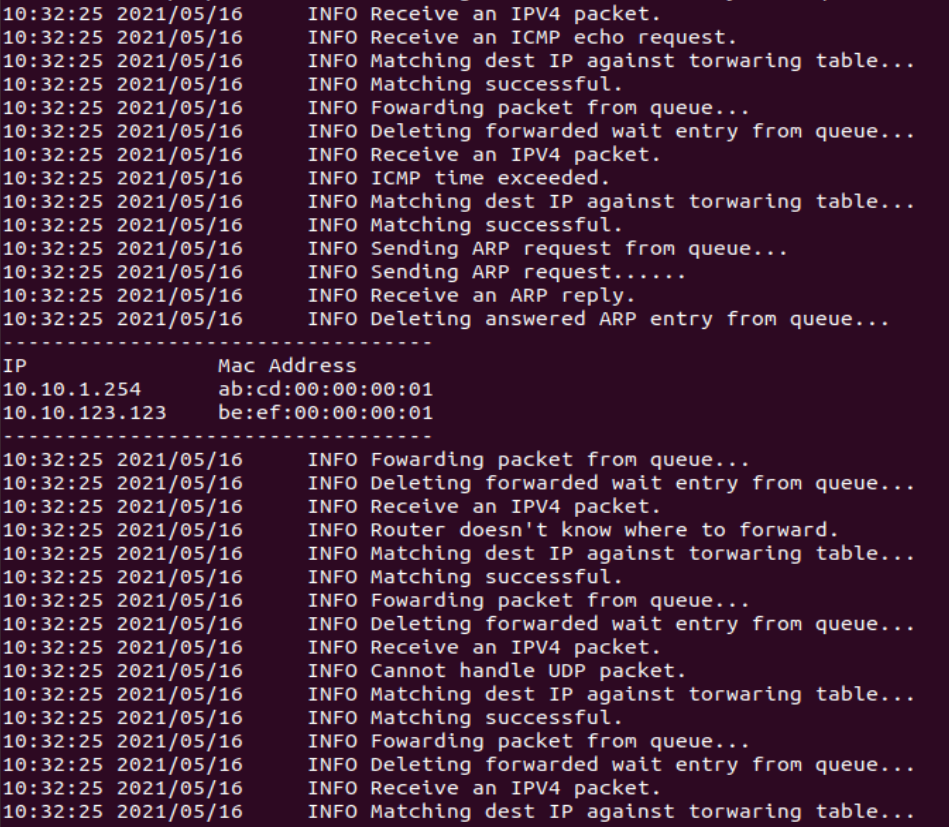
The fourth testcase handles a request with unreachable destination:





The fifth testcase fails to handle a UDP packet:



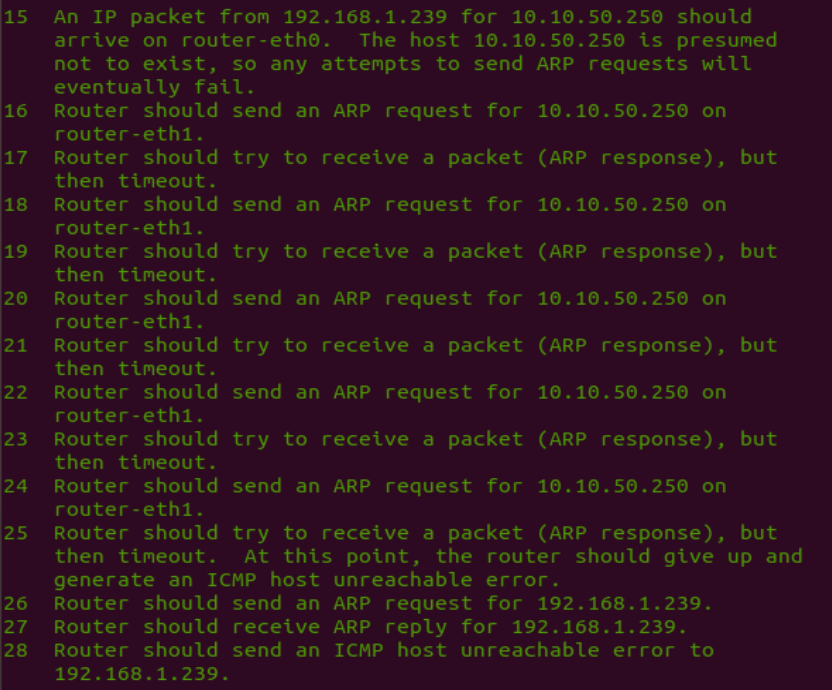


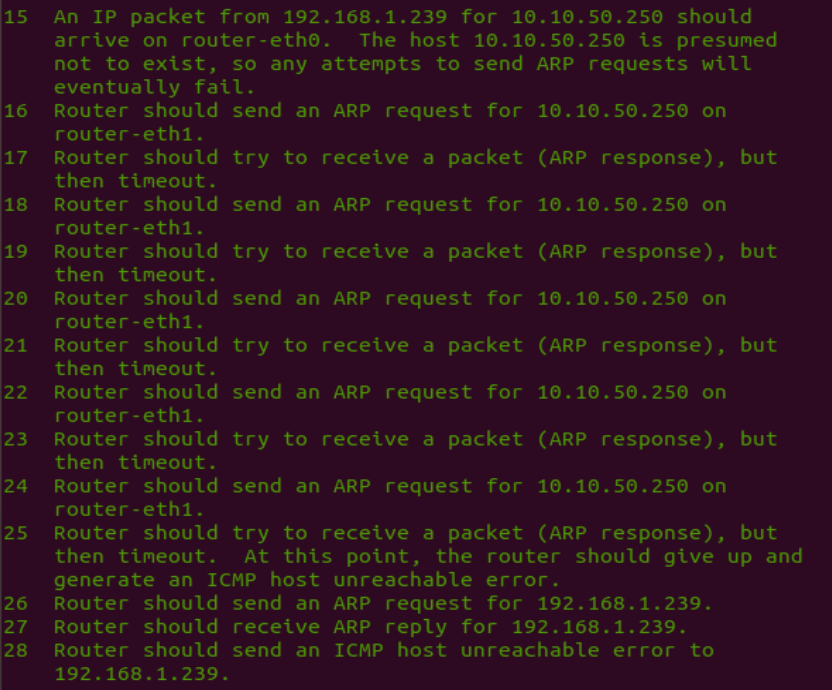
The sixth testcase deals with a destination host that does not exist:

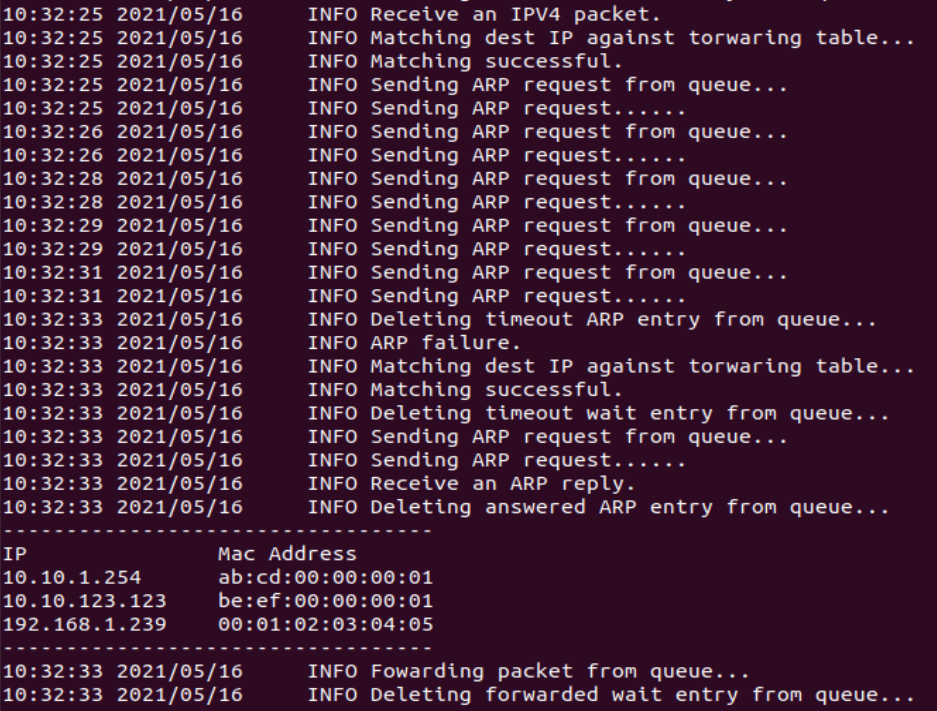
Five ARP requests are sent but no replies are received.

After that the router sends an ICMP host unreachable error.

An ARP request is needed before the error message is sent.



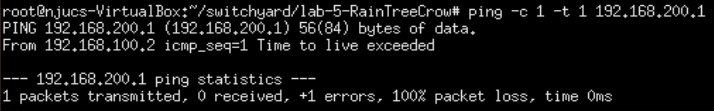




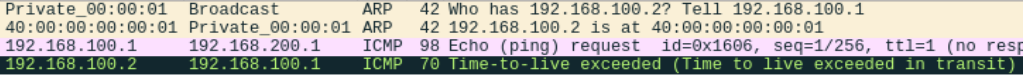
The logic of the testcases and log info match well.

* 1. Task 2 & Task 3 in Mininet:

Testing TTL exceeded error (using server1):

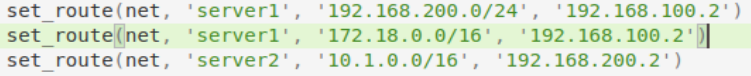


Capturing from server1-eth0:



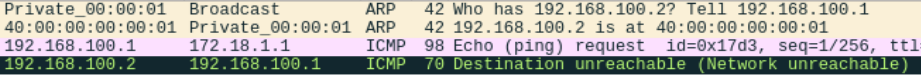
Testing network unreachable error (using server1):

Adding an additional route in start\_mininet.py imitating client:

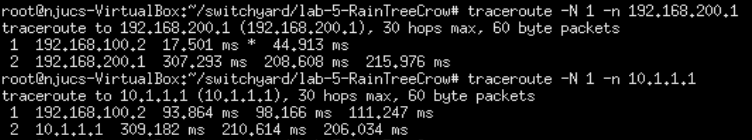




Capturing from server1-eth0



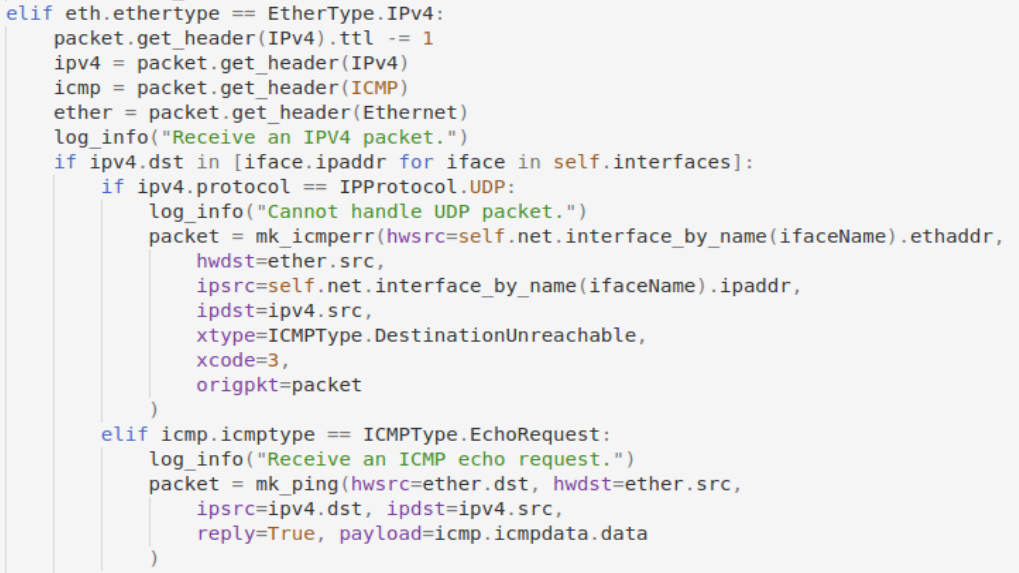
Testing trace route (using server1):



1. **核心代码**
   1. Task 2: Responding to ICMP echo requests

I ‘borrowed’ the functions mk\_ping and mk\_icmperr from the provided python test file router3\_testscenario\_template.py to generate packets.

#This part of code is added to the handle\_packet function. First check whether the IP destination address is the same as one of the addresses of the interfaces. If the packet is also an ICMP echo request, construct an ICMP echo reply.



#The reply is constructed using mk\_ping function. The destination address is set as the source address of the incoming echo request, and the source address is the router's interface address (which in this case is also the original request’s IP destination). Other information is in payload=icmp.icmpdata.data.

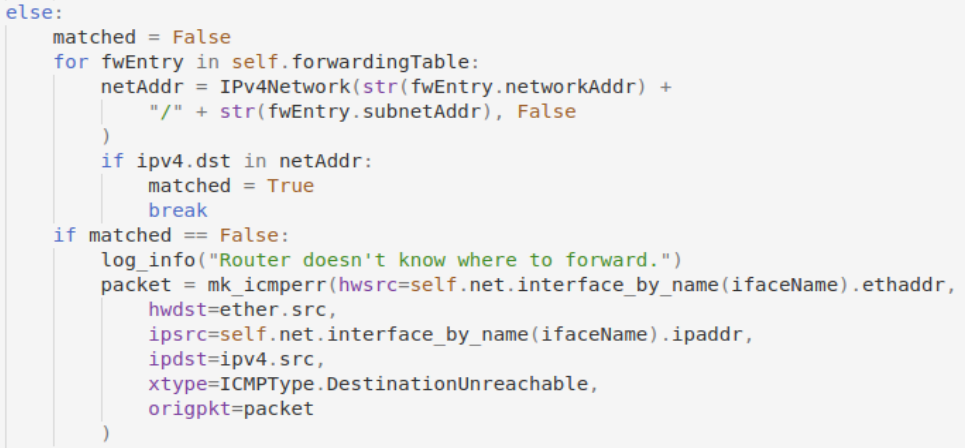
#I did not see any information regarding UCP packets until I came across the log info in the provided test scenario. It says the router cannot handle this type of packet and should generate an ICMP destination port unreachable error, so I wrote according to the description. Testcase oriented programming it is.

* 1. Task 3: Generating ICMP error messages

For situations in which ICMP error messages are generate are listed below:

1. Error: No matching entry found when attempting to match the destination address of an IP packet with entries in the forwarding table, (i.e., the router does not know where to forward the packet).

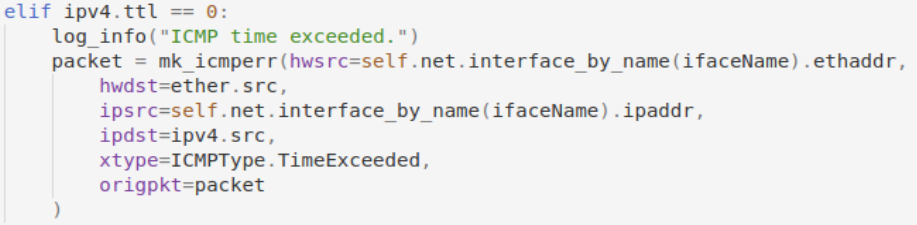
Solution: Send an ICMP destination network unreachable error back to the host referred to by the source address in the IP packet.



#The matching process is similar to Lab 4, only the packet is not directly put into wait queue if matching is successful. The part of code where the packets join the queue is moved to a separate function, send\_ipv4, which will be called after every type of error is inspected and dealt with.

1. Error: The IP packet's TTL becomes zero after decrementing.

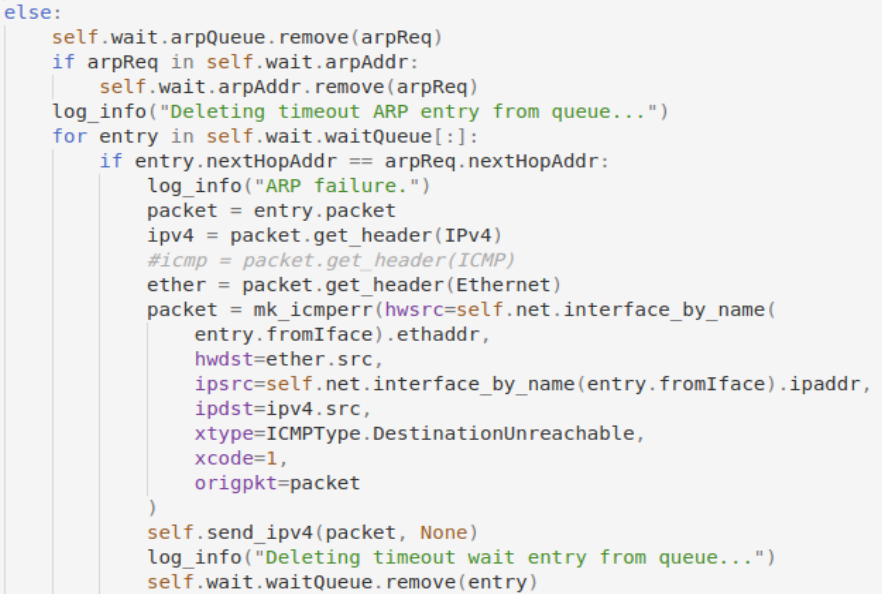
Solution: Sending an ICMP time exceeded error message back to the host referred to by the source address in the IP packet.

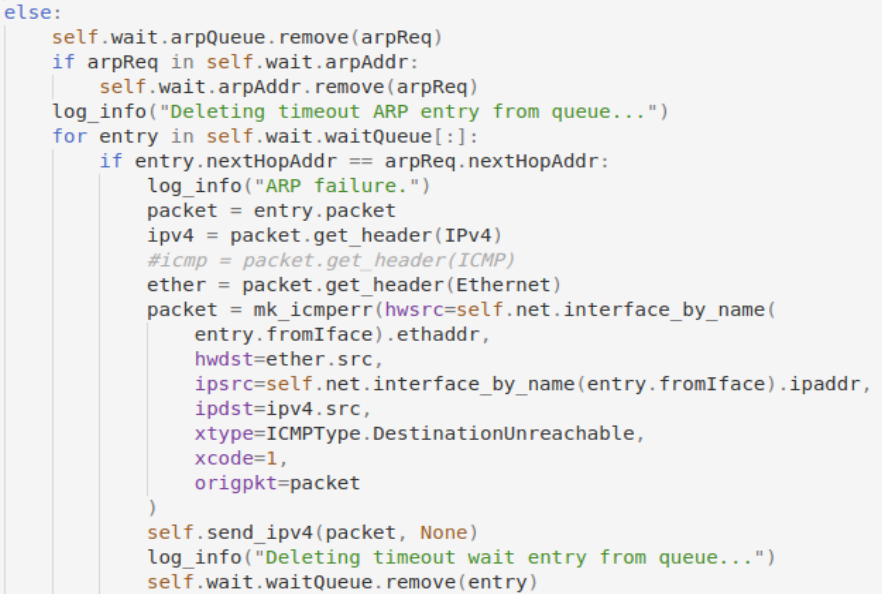


1. Error: ARP Failure. After 5 retransmissions of an ARP request, the router does not receive an ARP reply, which indicates there is no host that "owns" a particular IP address of the next hop or the destination host.

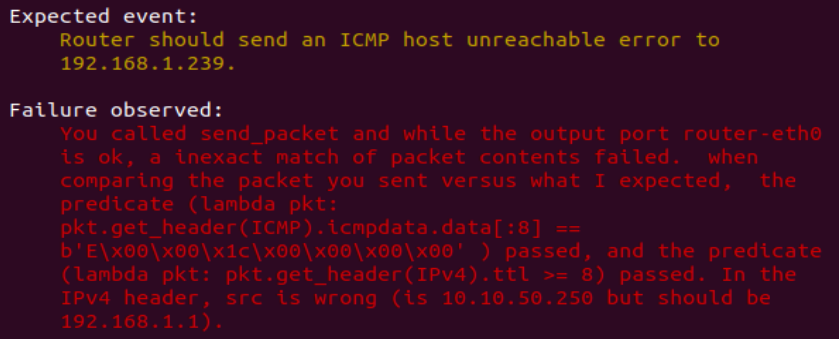
Solution: Sending an ICMP destination host unreachable back to the host referred to by the source address in the IP packet.

#This part of the logic (transmitting and retransmitting ARP requests, and dropping timeout ARP requests) used to be dealt with in class WaitQueue’s member function update\_queue, it is now moved to the class Router, and is modified to deal with timeout ARP requests better.





#The WEntry and add\_entry function in Lab 4 do not remember where the packets to be forwarded come from. So, when the ARP failure error should be sent back, it does not know the source address (to be honest, I did not know the source address either, until I saw the log info in the test scenario and assume it should be the address of the interface the original input port of the packet saved to the queue, thus, a variable fromIface is added to the WEntry to solve this issue, it seems to work okay.



1. Error: An incoming packet is destined to an IP addresses assigned to one of the router's interfaces, but the packet is not an ICMP echo request.

Solution: Send an ICMP destination port unreachable error message back to the source address in the IP packet.



\*One thing I didn’t quite understand about the functions mk\_ping is that in the manual, I am supposed to copy the echo request’s sequence number, identifier and data field. At first I used the function without modifying and it passed, but it did not handle sequence and identifier, just the data field, I supposed.

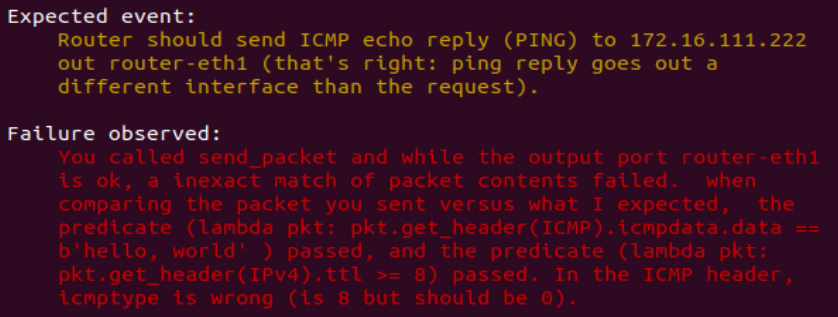
Then I tried two ways to deal with the problem and chose the second one.

\*If I pass in the entire icmpdata and copy it as a whole:

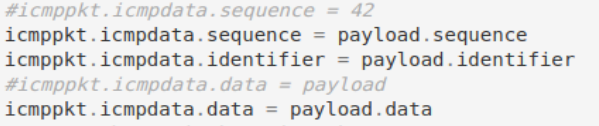




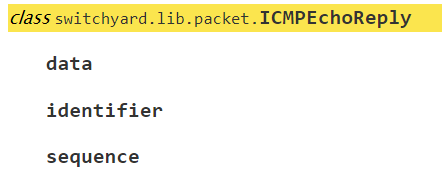
I cannot pass the testcases and will fail here:



\*If I copy the three parts respectively:

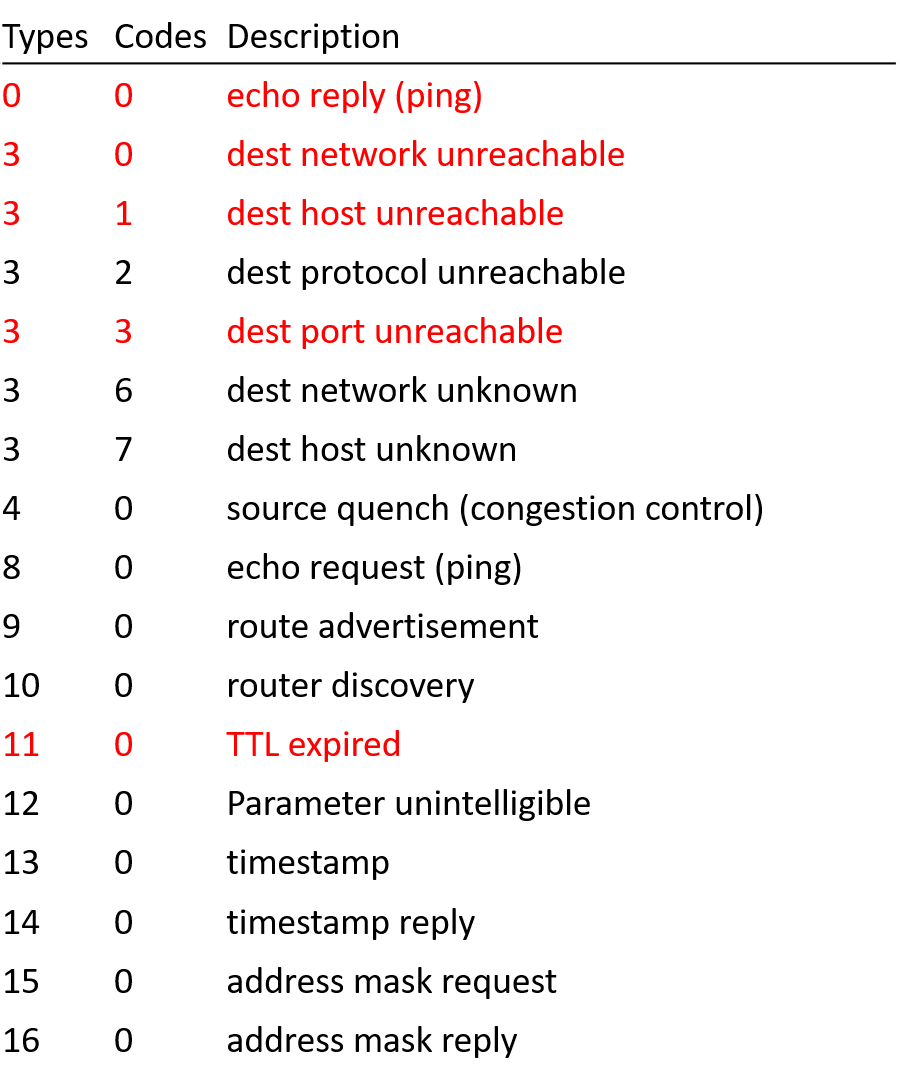
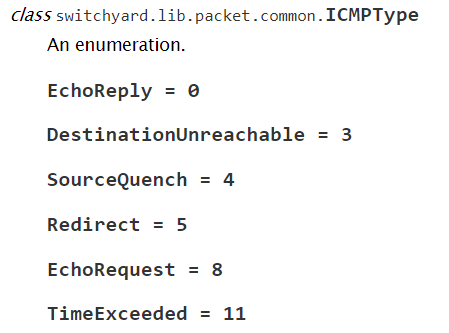


Then I can pass all testcases successfully. I thought the manual says icmpdata only has these three fields, but turns out to be a misunderstanding.



\*It says data, identifier and sequence, but well…

#In all the cases above, the ICMP type and code are set as below:



1. **总结与感想**
   1. I probably should not have designed so many classes in Lab 4, the entries and queues really took me a long time to modify. With so many functions calling each other, the structure became messy and confusing. I suppose the best way to solve this kind of problem is to read through Lab 3 to Lab 5 and design the structure as a whole, otherwise the classes should be designed carefully so they would easy to change. In a word, DO NOT use object-oriented programming UNLESS you have the confidence to arrange the classes neatly.
   2. Reading the manual is important, DO NOT take anything for granted.