**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 | **2021/5/1-2021/5/2** |

1. **实验名称：Lab 4: Forwarding Packets**
2. **实验目的：**

Continue to improve the router to implement the function of packets forwarding.

1. **实验内容**
   1. Task 2: IP Forwarding Table Lookup

Build Forwarding Table

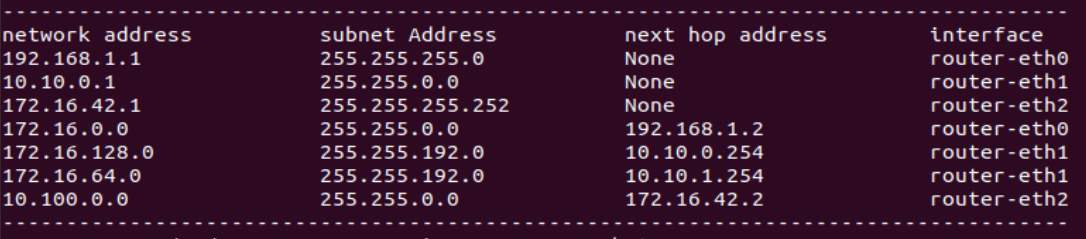
Match Destination IP Addresses against Forwarding Table

* 1. Task 3: Forwarding the Packet and ARP

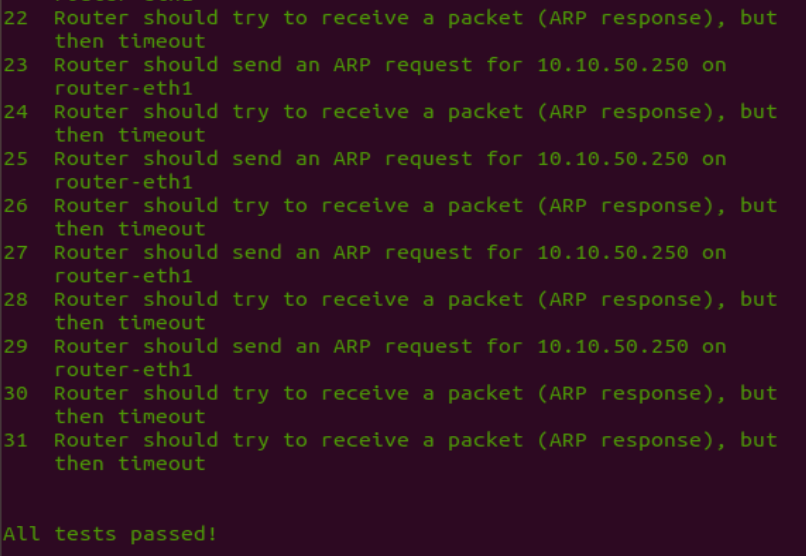
Send ARP Request and Forward Packet

1. **实验结果**
   1. Task 2 & Task 3 in test scenario:

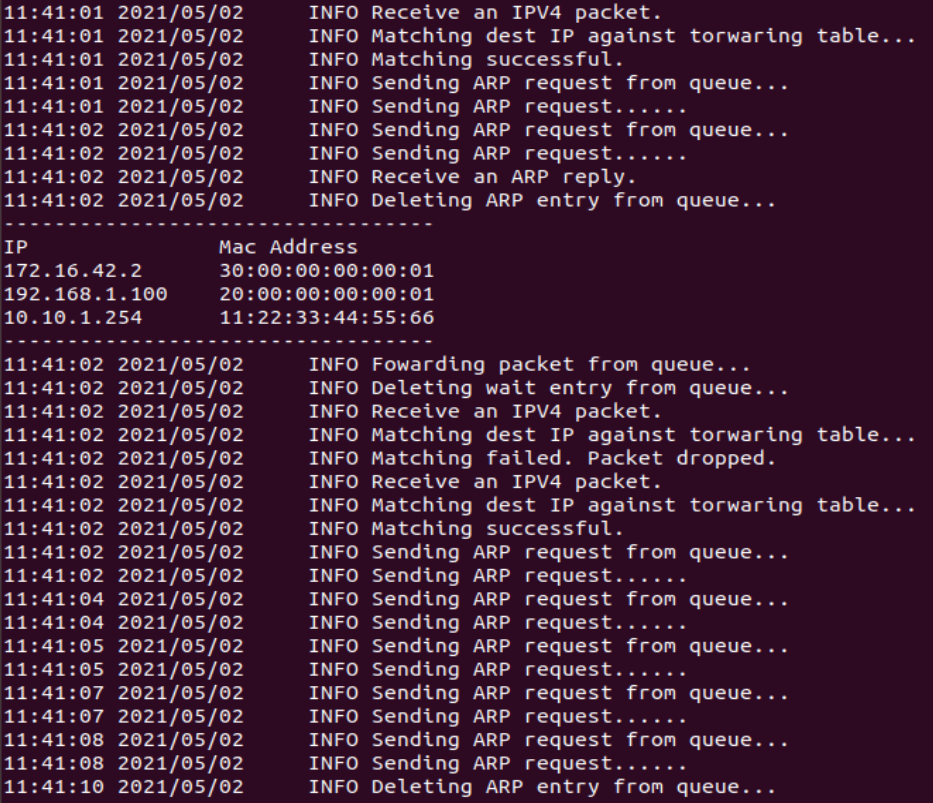
Building forwarding table:



Passing test cases:

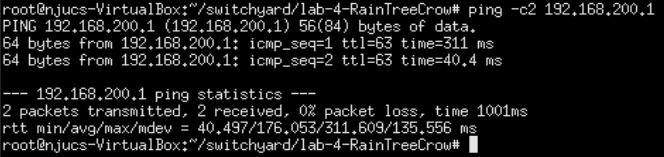


Some log info on handling packets and queue entries:

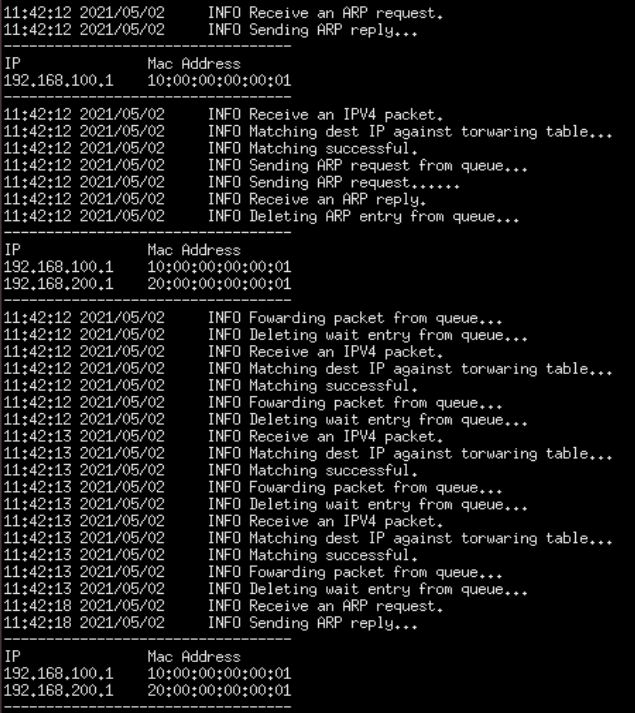


* 1. Task 2 & Task 3 in Mininet:

Ping server2 from server1:

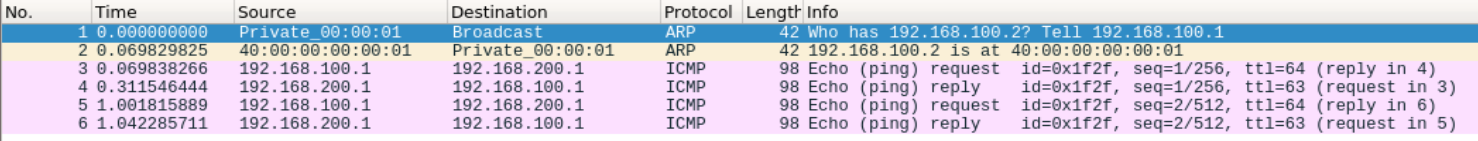


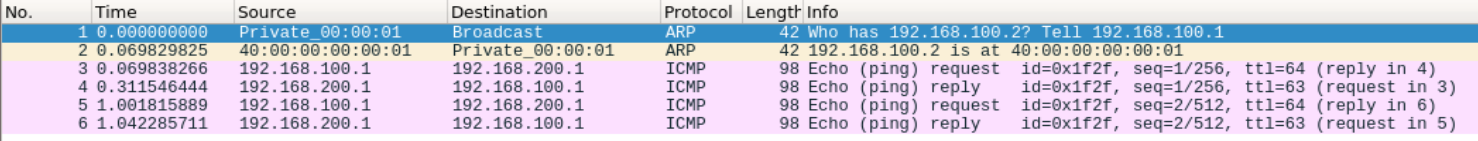
Router log info:



From the log info I can infer that the interface received four IPV4 packets, two ARP requests and one ARP reply, all of which are dealt with accordingly.

Captured file on server1-eth0:

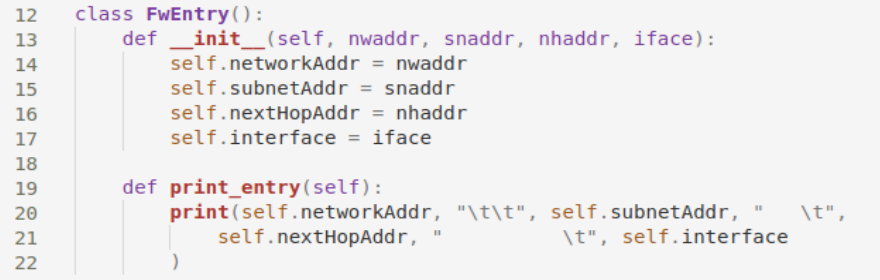




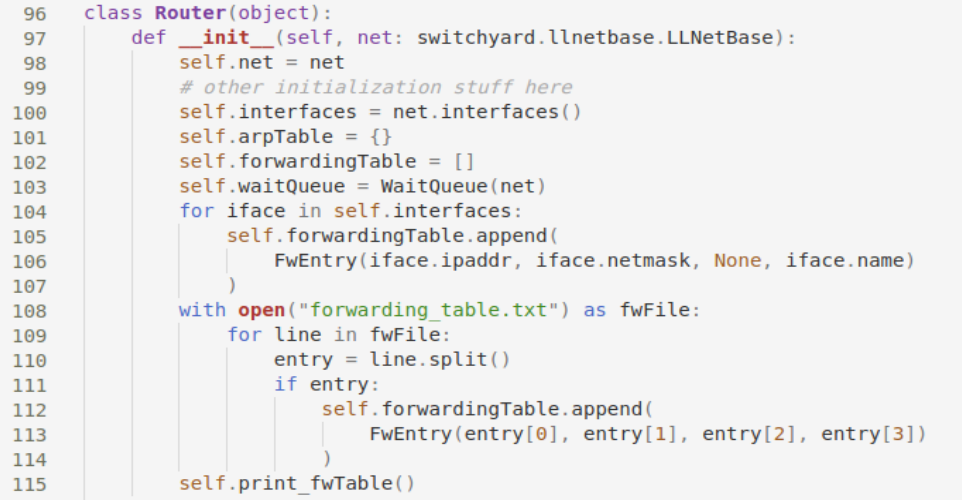
1. **核心代码**
   1. Task 2: IP Forwarding Table Lookup

Building Forwarding Table:

The forwarding table entry class is defined as below:



The router’s forwarding table is initialized together with the router itself. The entries are added from the router’s interfaces and the txt file.

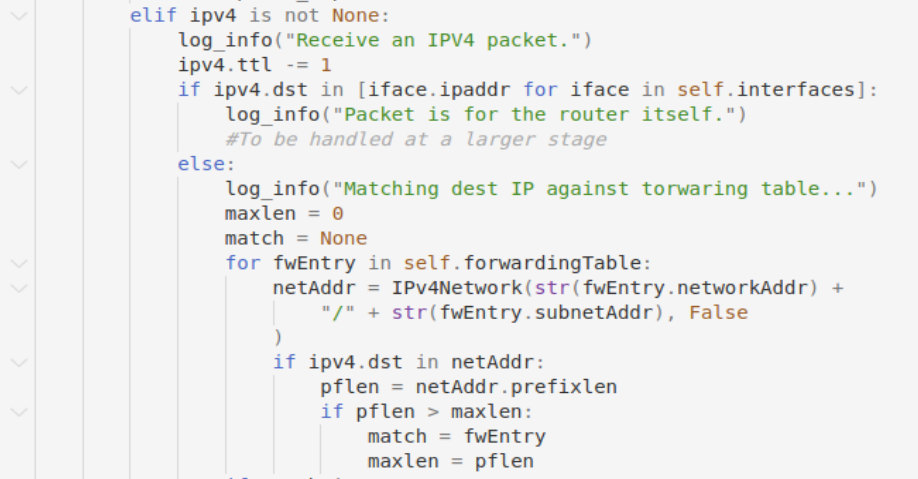


Match Destination IP Addresses against Forwarding Table:

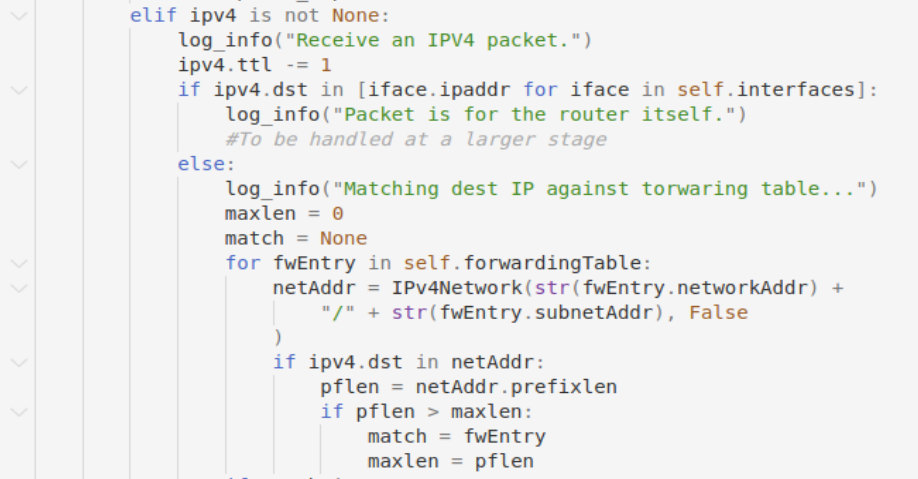
# this part of logic is defined in the handle\_packet function:



# we skip the part that deals with ARP packets and move on to IPV4:



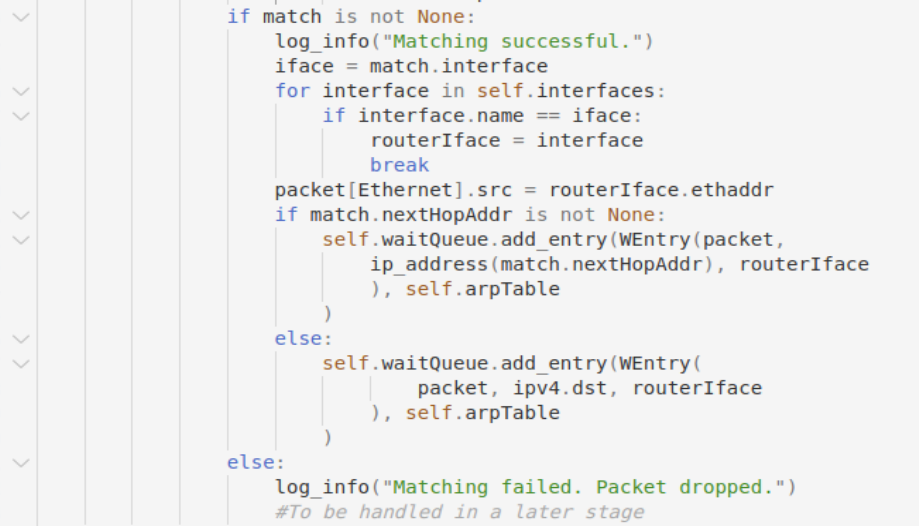
# Decrement the TTL field in the IP header by 1

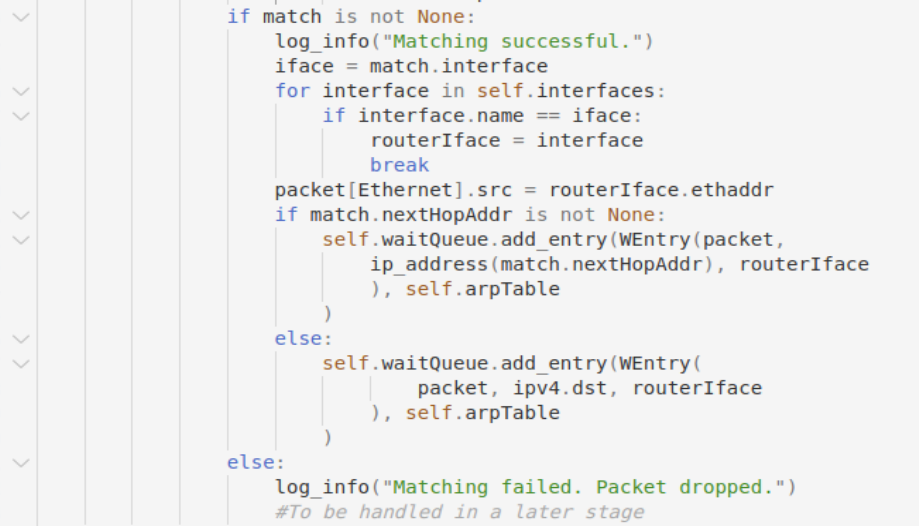


# If packet is for the router itself (i.e., destination address is an address of one of the router's interfaces), drop/ignore it. (to be handled at a later stage.)

# While matching destination IP address against the forwarding table, longest prefix match is used, the entry with the max prefix length is picked out

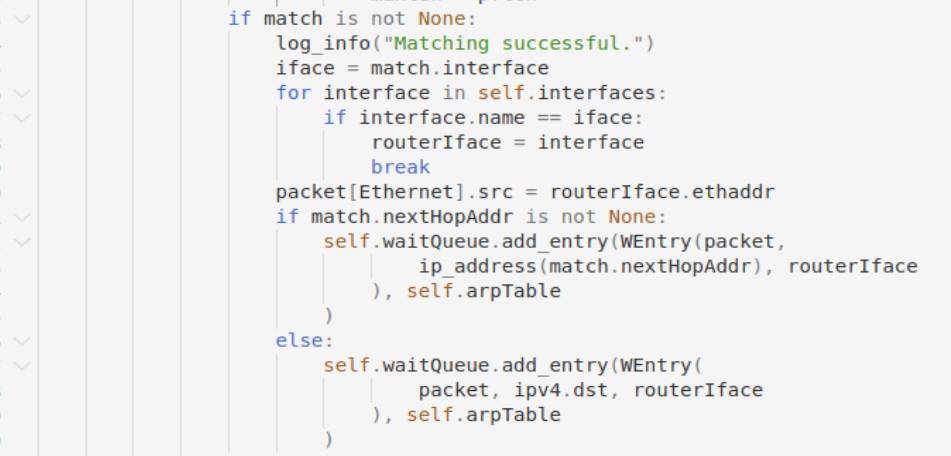
# If there is no match in the table, drop. (to be handled in a later stage.)





* 1. Task 3: Forwarding the Packet and ARP

Here is how the router behaves after a successful match:

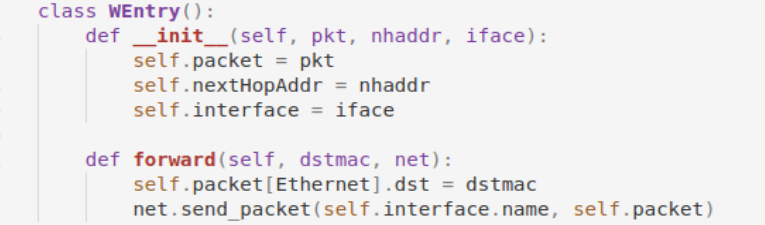


# Create a new Ethernet header for the IP packet to be forwarded. The next hop host is either the destination host if the destination address is directly reachable through one of the router interfaces (i.e., the subnet that the destination address belongs to is directly connected to a router interface), or IP address on a router through which the destination is reachable. In the initialization strategy, those entries we add from the router’s interfaces do not have a next hop address, they are the ones “directly reachable through one of the interfaces”, through this we can clearly and easily tell the two separate cases apart

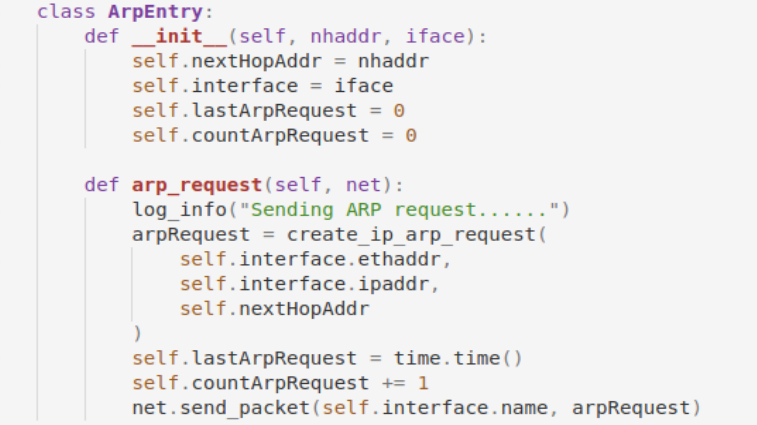
# The new Ethernet header should also include new source address, that is the ethernet address of the matched interface, the header is modified accordingly

At this point, we do not know yet whether an ARP request is needed before we send the packet, so we add an entry to the wait queue defined as below:

Here is how the wait entry is defined, when the router finds an entry’s next hop address in the ARP table, it informs the entry of the matching destination MAC address, and it forwards the Ethernet packet accordingly:



Another class defined is the ARP entry, for several different entries in the wait queue may “want” to send ARP requests with the same target address, the ARP entry has the variable lastArpRequest, which records the last time a request is send inquiring about the address, and countArpRequest that records how many ARP requests are sent (so far) inquiring about this address.



# Each time an ARP request is sent, the two variables are updated

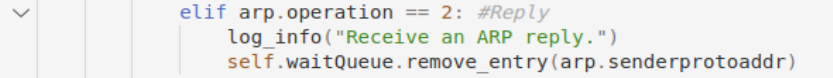
Here is the class that helps maintain the two entries above, waitQueue records the wait entries, arpQueue records the ARP entries, and the list arpAddr is used to remember which address are already in the ARP queue, so that we only need to traverse a list to determine whether the ARP entry is present.



# The function add\_entry adds to the wait queue every time it is called, but the ARP queue is appended only if the target address is NOT already in the router’s ARP table or the wait queue’s ARP address list

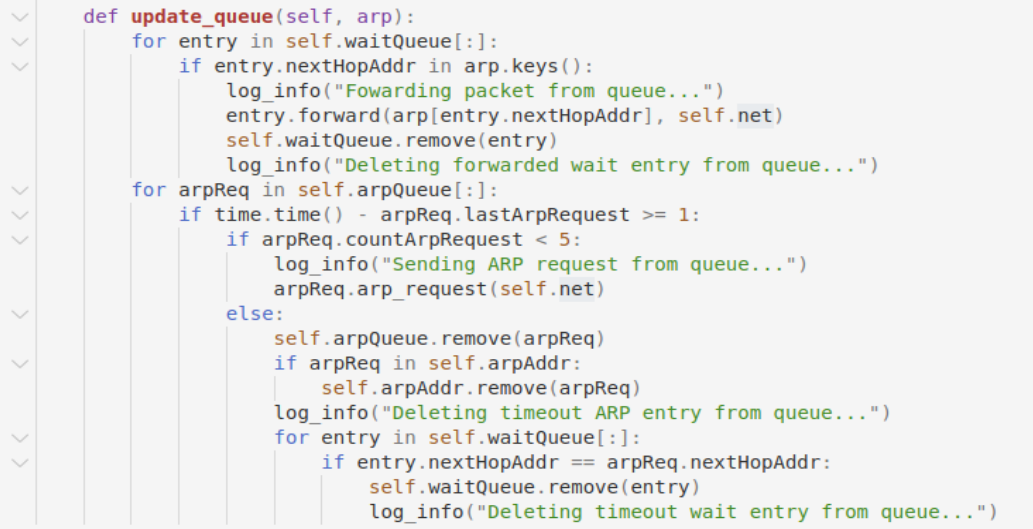


# The function remove\_entry deletes the corresponding entry in the ARP queue and removes the address from the ARP address list, the function is called every time the router receives an ARP reply:



Here is the update\_queue function, called in the while loop of the router’s start function before the try statement that captures packets:





# Each time the function is called, it goes through each entry in the wait queue finding whether the next hop address is in the ARP table, if there happen to be a new item (after receiving an ARP packet) that matches the address, the entry calls its forward function, and the entry is removed from the queue. The queue is designed using a list, so the entries in it are arranged according to the order in which they are added. Thus, the router buffers multiple packets sharing the same ARP request, and upon receiving the corresponding ARP reply these packets are forwarded in the order they arrived to the router

# For each ARP entry in the ARP queue, if the time elapsed since the last time a request is sent is more then one second, and the same request have been sent for less than five times, it calls arp\_request to resend. If the same request has been sent for more than five times, the ARP entry and each wait entry with the same target next hop address are removed from the queues they are in

1. **总结与感想**
   1. Next time I should read through the entire manual (including the FAQ) before I start coding. This time I did not take into account the case where several wait entries might share the same next hop and send redundant ARP requests, so I had to write another class to store and maintain ARP information. If I had taken it into consideration, I should have written a different data structure, perhaps I would put ARP requests into a wait queue and store the packets to be forwarded inside the ARP entry. The structure seems simpler and more efficient to me.
   2. At first, I did not realize the differences between the addresses I read from the txt file (strings) and the IP addresses in the net. Python is a language in which the data types can be very obscure and confusing, in a class we can use \_str\_ function, so even though an object can be printed, it does not necessarily mean it is a string, and even though the two objects appear identical when you print them out, it does not necessarily mean they are equal. I found a simple (stupid) way to debug, that is to print (x == y) or (Type(x)) all over the place. A better idea may be reading the definition of each class more patiently.
   3. Pdb is a good tool. Before this lab I did not even know how to use “where” to find out where my code had stop executing. Perhaps it was because in the first three labs I did not come across some bugs that are really confusing and hard to locate, but writing code blindly without and test methods is definitely not a good habit. I shall learn to use the tools myself instead of waiting for someone to ask questions in the QQ group and some TA come to their rescue.