**路由转发实验**

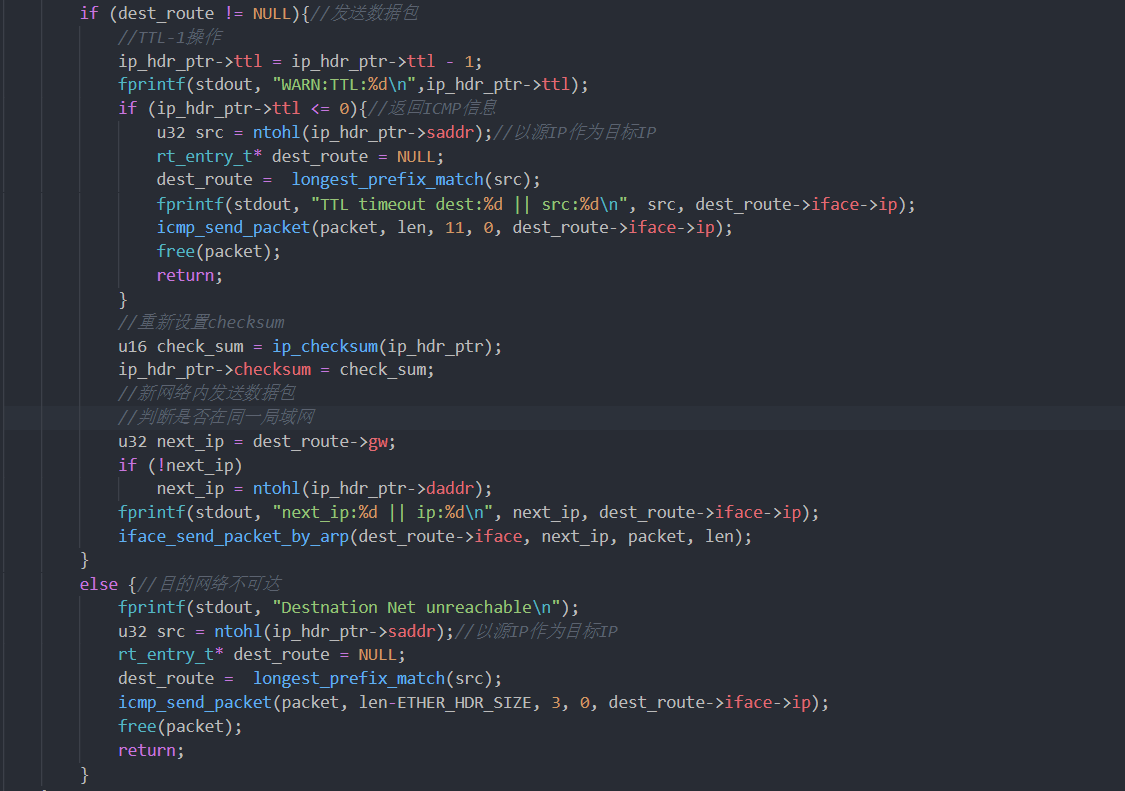
秦宏

2016K8009929009

1. 实验目的
2. 熟悉并使用IP数据包进行数据包传递转发。
3. 理解ICMP协议的作用，熟悉并使用ICMP数据包实现错误返回。
4. 理解ARP协议机制，实现arpcache的动态缓存。
5. 学会使用ping操作和traceroute操作原理。
6. 实验内容
7. 实验首先处理收到IP数据包的情况，判断是否是ping本节点的数据包。
8. 若是，则返回ICMP数据包，不是，则进行路由表查找，
9. 未查找到，返回ICMP数据包，查找到，从ARP协议中查找mac地址。
10. 查找到mac地址，转发数据包，未查到mac地址，返回ICMP数据包。
11. 不同情况下返回的ICMP数据包type不同。
12. 然后处理ARP请求与应答机制，处理ARP数据包时判断是请求操作还是应答操作，若是请求操作，则作出应答，并将IP与mac映射加入缓存中。若是应答操作，则将IP与mac映射加入缓存，并将req\_list中等待该映射的数据包转发出去。
13. 考虑老化线程，需要清理ARP中长时间未使用的IP与mac映射，并且将长时间未应答的ARP请求重复发送或者清除等待的数据包，并对应回复ICMP数据包。
14. 实验流程
15. 首先处理接收到IP数据包的情况，代码如下

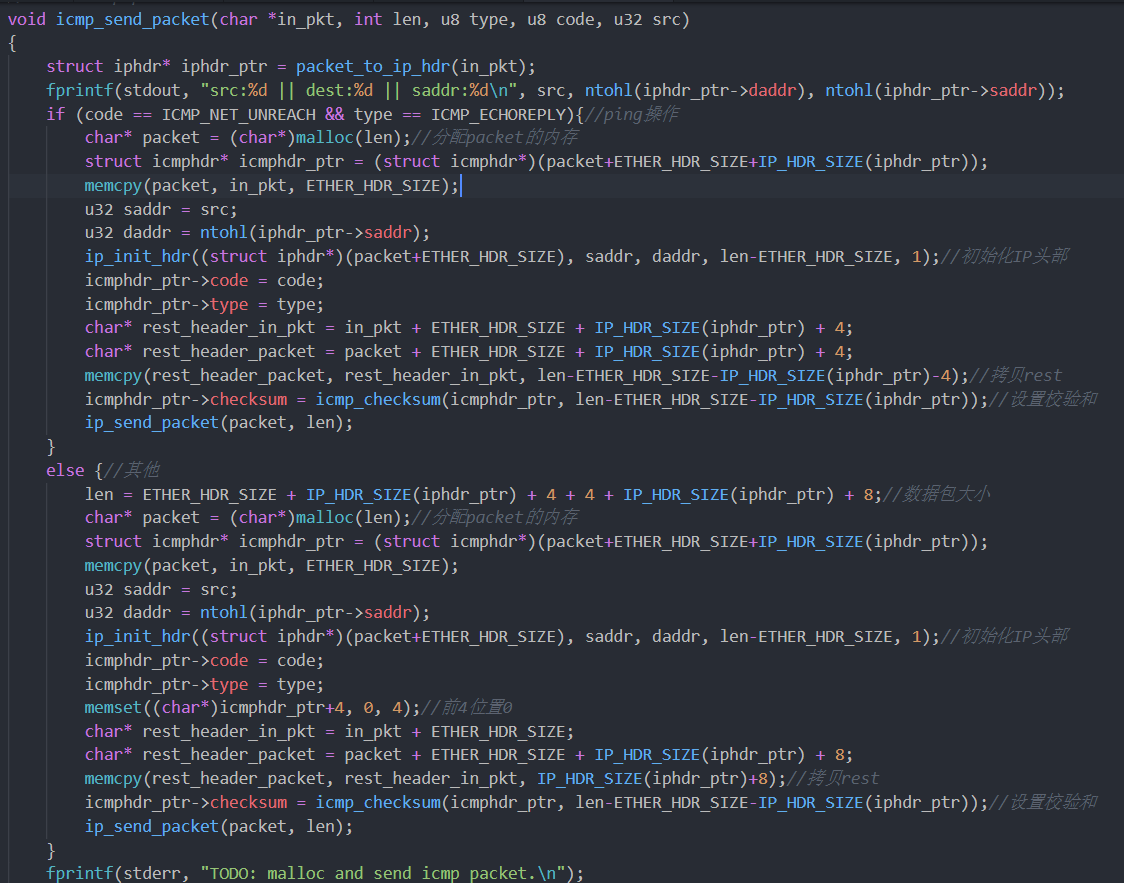


主要包括提取IP数据包中的目的地址，然后判断是否为ping本节点的操作，若是，则直接将源地址作为目的地址，使用icmp\_send\_apcket函数返回ICMP数据包，否则进入路由表查找逻辑。



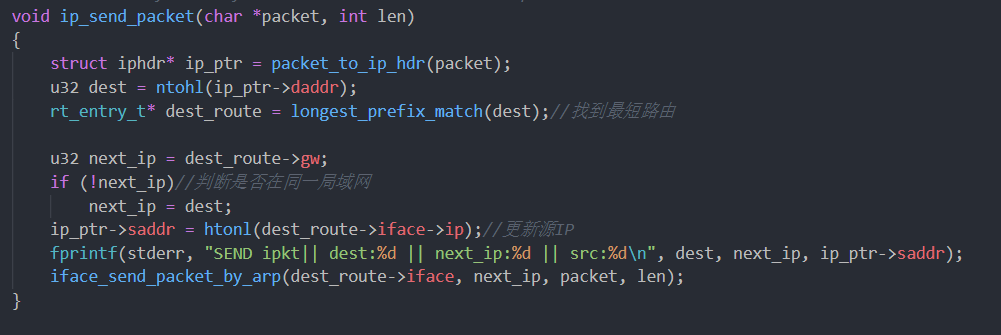
首先查找路由表，若查找到，则判断TTL的值，然后确定是否返回ICMP数据包或者转发IP数据包。如果没有找到映射，则返回ICMP数据包。

1. 然后是ICMP数据包发送函数icmp\_send\_packet



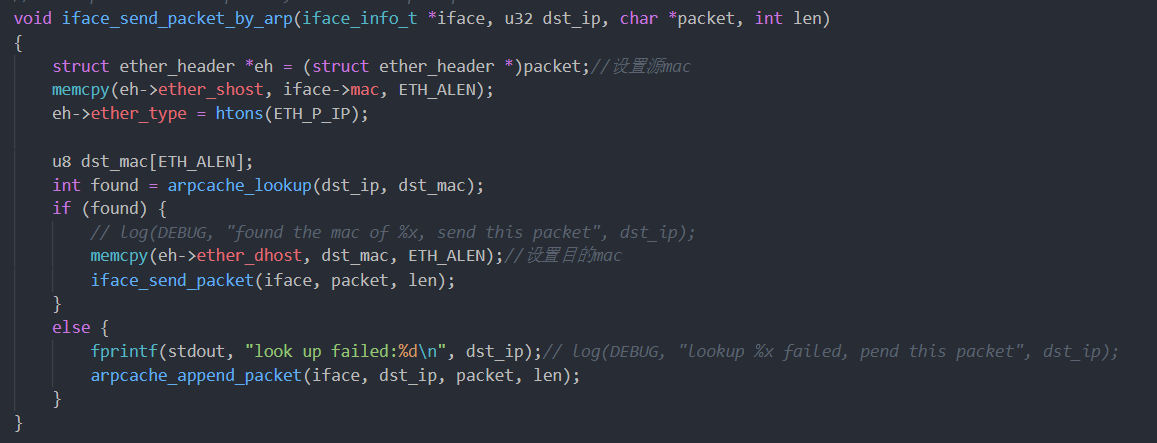
该函数主要分为处理ping本节点的操作和其他操作，因为其他操作的ICMP数据包数据段都一样。这里路由器生成ICMP数据包，因此采用malloc函数分配内存，分配ICMP包的目的地址和源地址，拷贝相应的数据信息，不断利用指针操作实现内存拷贝，最后将数据包packet传递给ip\_send\_packet函数。

1. 然后是ip\_send\_packet函数



通过longest\_prefix\_match函数找到数据包packet里面目的地址对应的最短路由，判断目的地址是否位于同一网段，并将端口、下一段IP、数据包传递给iface\_send\_packet\_by\_arp函数。

1. iface\_send\_packet\_by\_arp函数分析



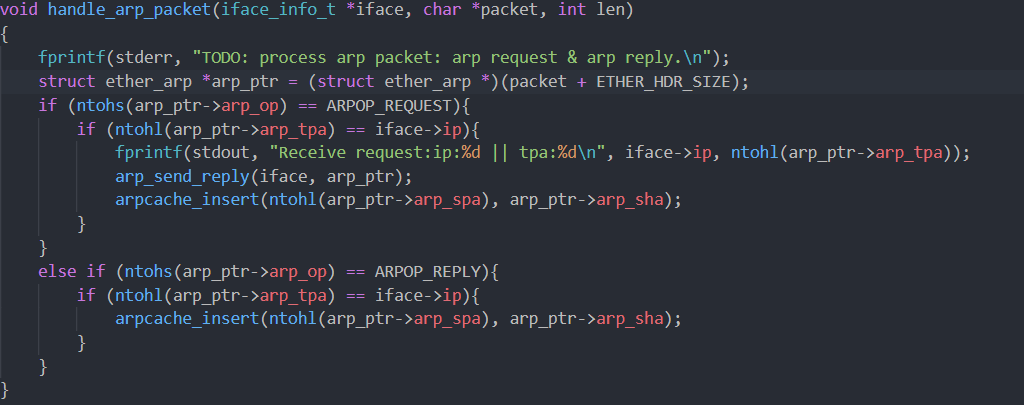
尽管该函数已经实现，但是其内包含了对mac地址的填充操作，因此不必在其他地方填充mac地址，只需找到匹配项即可，即调用了arpcache\_lookup函数，如果没有找到对应映射则交给arpcache\_append\_packet函数处理，否则直接通过iface\_send\_packet函数转发出去。

1. 最长地址匹配的函数实现如下



遍历路由表即可，不再赘述。

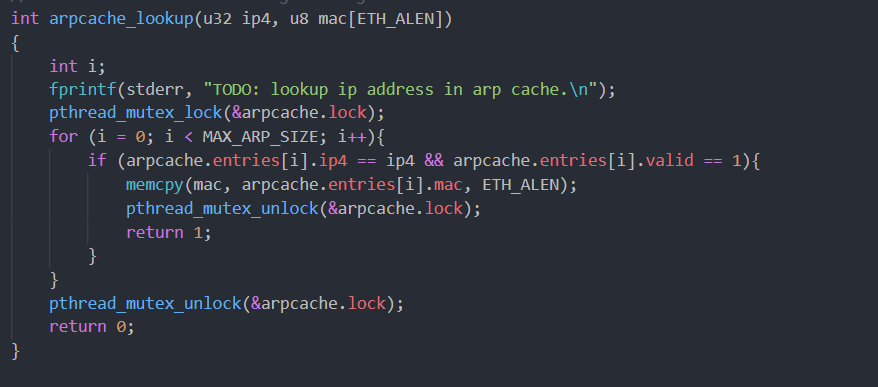
1. 然后是对ARP数据包的处理函数handle\_arp\_packet



如果数据包是请求包，则判断端口ip后作出应答，并将ip与mac映射插入arpcache中；

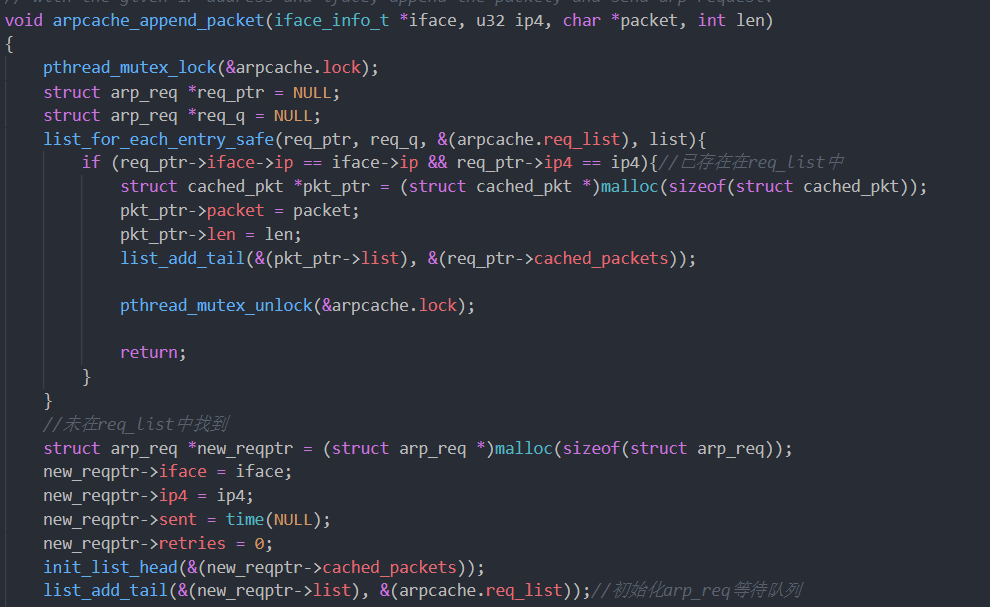
如果数据包是应答包，则将应答包中ip与mac的映射插入arpcache中。

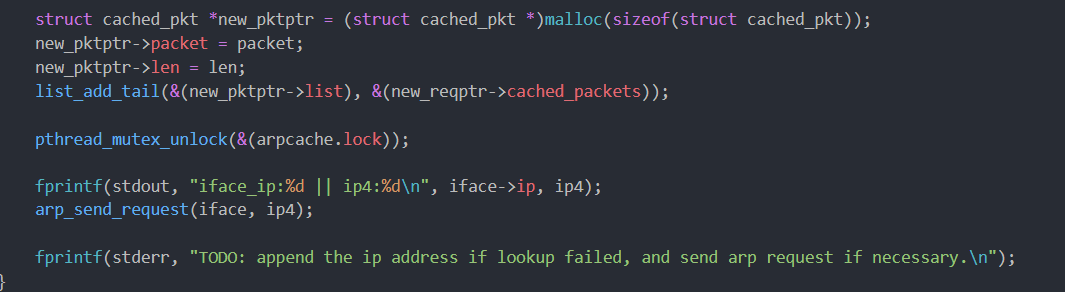
1. 分析arpcache\_lookup函数



Look\_up函数遍历缓存中存储的有效映射，并返回输入IP的mac地址。

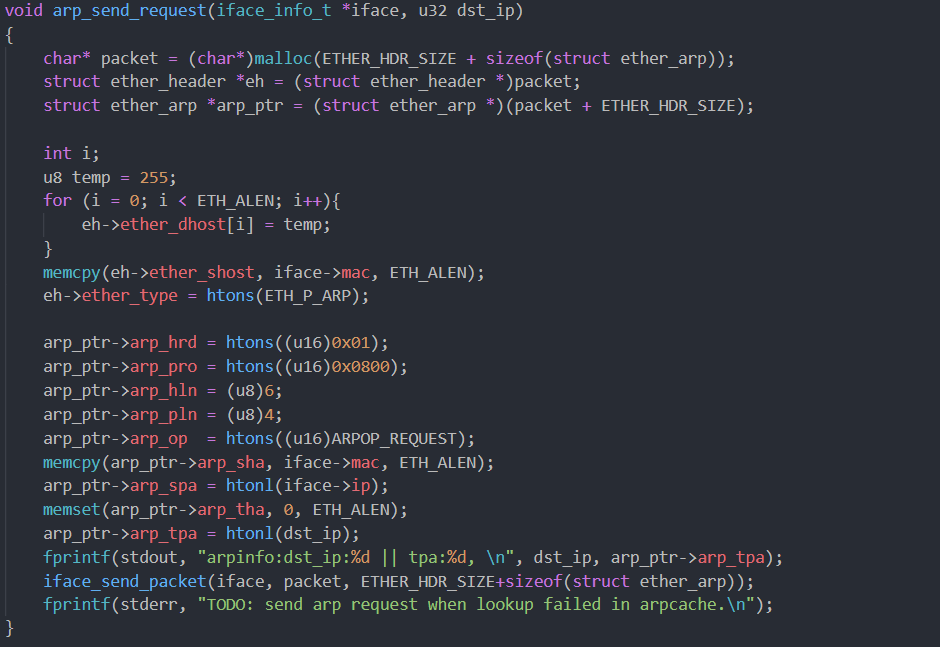
1. 分析arpcache\_append\_packet函数





Arpcache\_append\_packet函数对输入的数据包进行处理，如果该端口的ip与某个请求序列的ip一致，则只需加入到该队列的尾部即可，此时该ip已有arp请求发出。否则需要新建一个ip队列等待映射查找，并在该队列下添加该数据包。并发出arp请求。

1. 分析arp\_send\_request函数

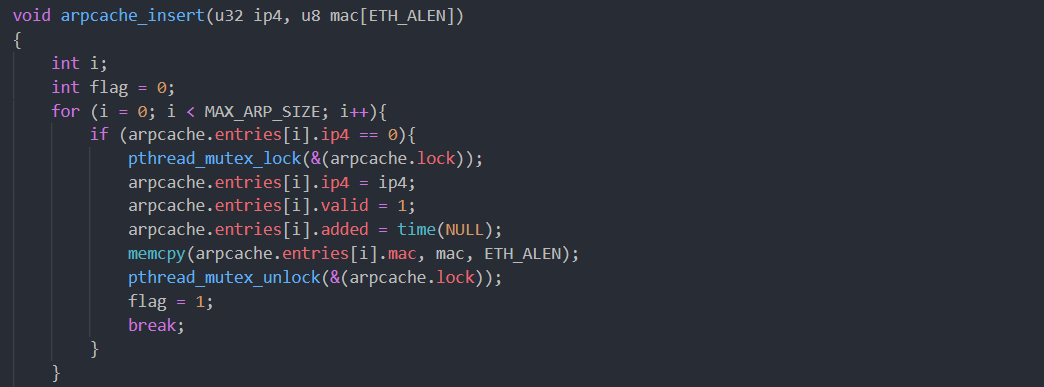


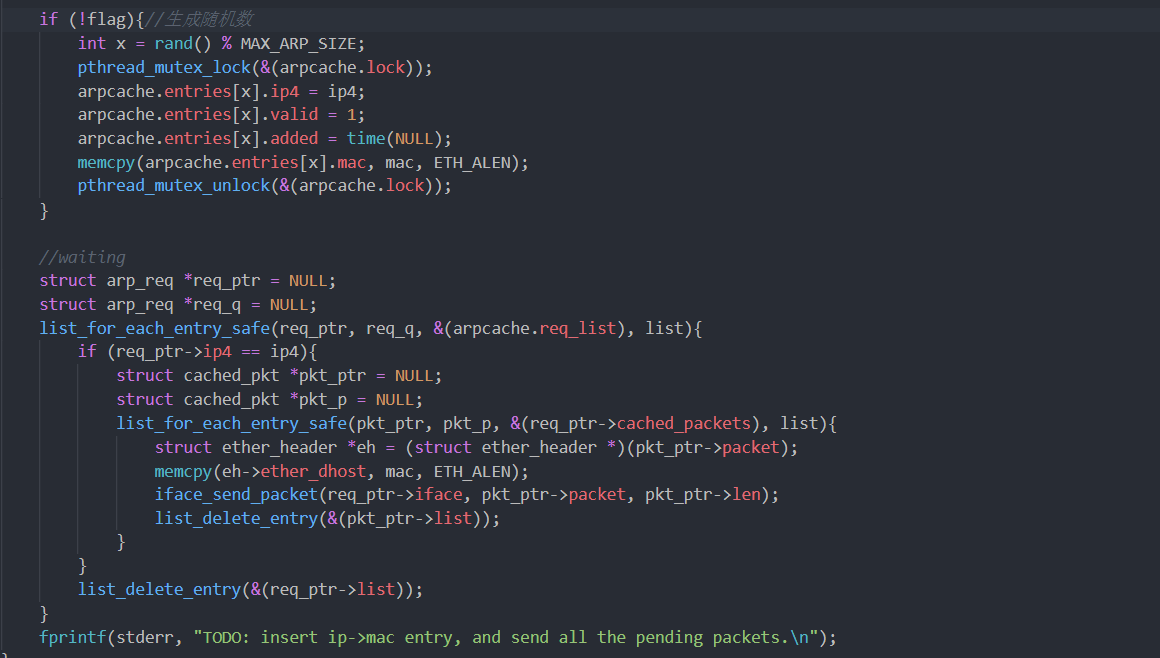
该函数的内容主要是生成一个新的arp数据包，其中大多数内容都是固定的，只需分配目的源的IP和mac地址，通过iface\_send\_packet函数发送。

1. 分析arp\_send\_reply函数

与arp\_send\_request函数内容基本一致，不再展示。

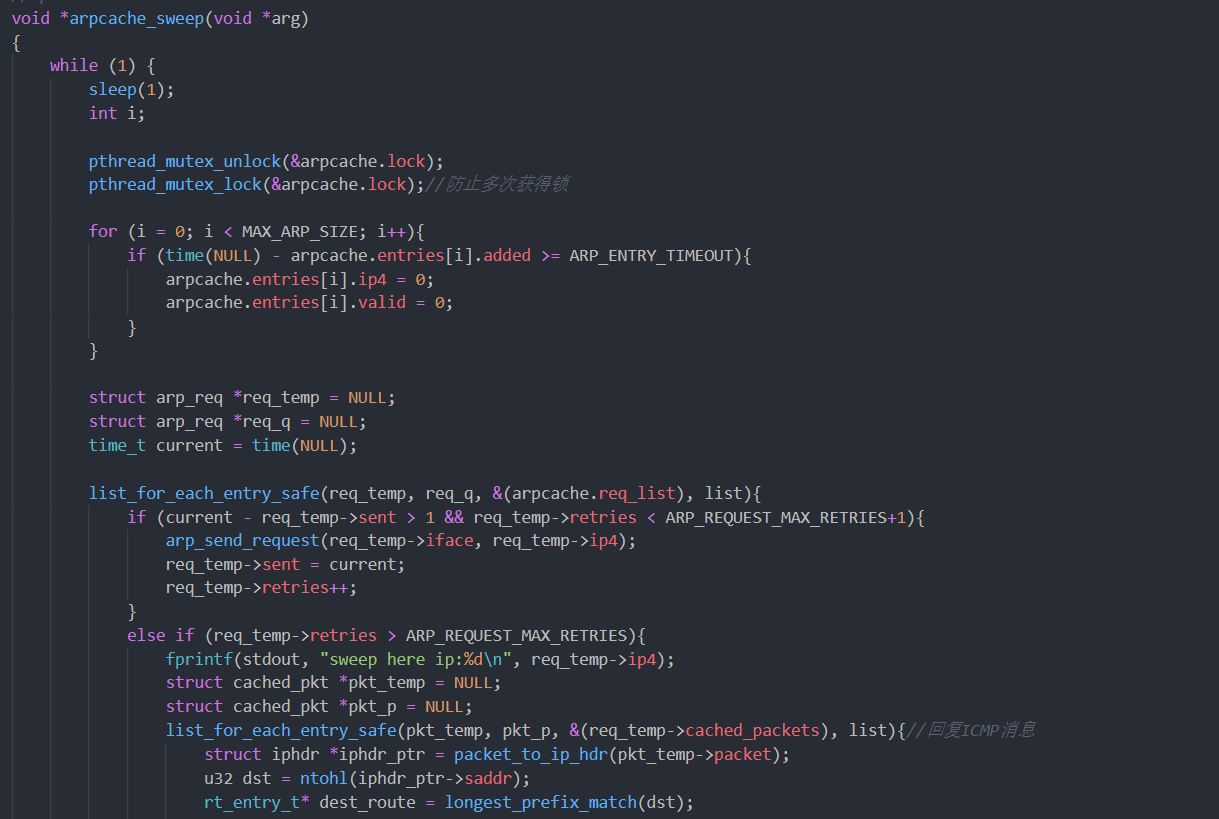
1. 分析arpcache\_insert函数





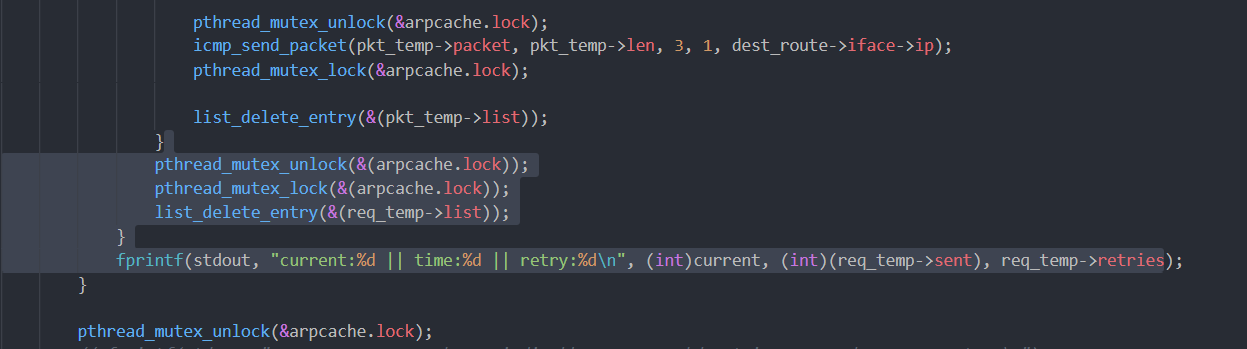
函数首先判断是否有空的IP与mac映射，若有则直接添加即可。若没有，使用随机数替换掉任意一个映射。当添加IP与mac映射时，将req\_list队列下请求该IP的所有数据包转发出去，并将这些数据包清除掉。

1. 分析老化函数arpcache\_sweep

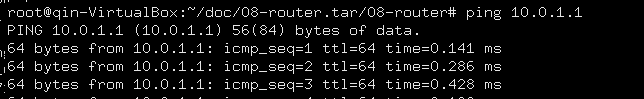


老化函数考虑两个方面，一个是IP与mac映射中长时间不使用的映射，只需要遍历数组判断间隔时间即可。另一个是考虑arp请求的发送，arp请求要求间隔时间超过1s就要重新发送，且发送次数不超过5次。上图中先是清理掉IP与mac映射中长时间未使用的部分。然后对req\_list进行遍历，如果arp请求发送间隔时间超过1s但不超过5次，则重新发送arp请求；如果超过5次，则将该请求下的所有数据包清除，并返回ICMP数据包。

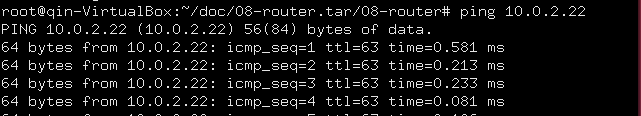
使用icmp\_send\_packet函数返回ICMP数据包。



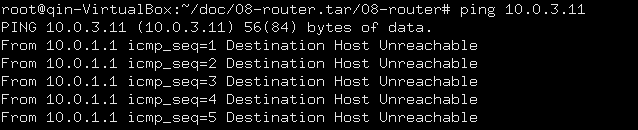
1. 实验结果



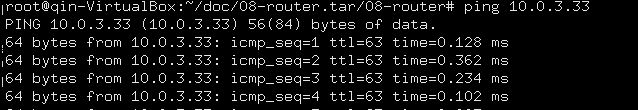
ping 10.0.1.1



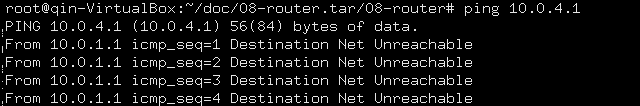
ping 10.0.2.22



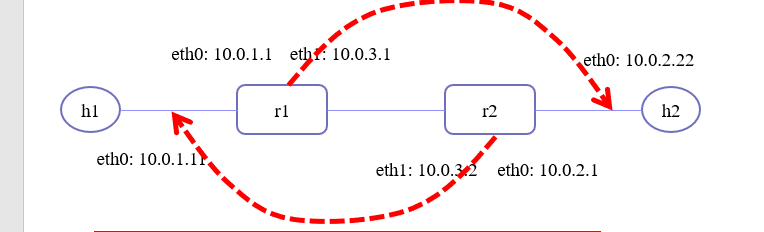
ping 10.0.3.11



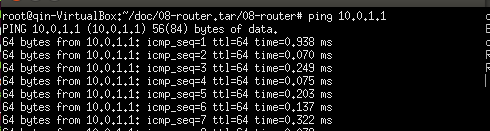
ping 10.0.3.33



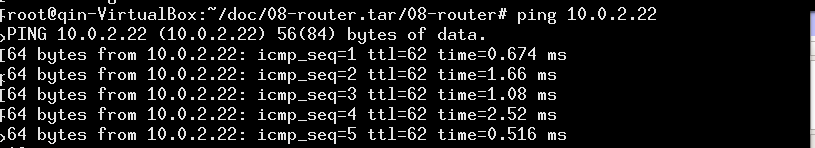
ping 10.0.4.1



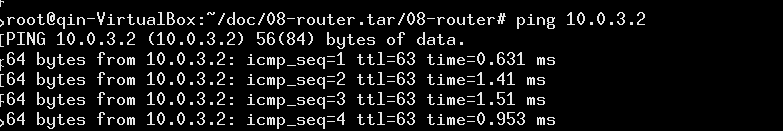
新建拓扑如图



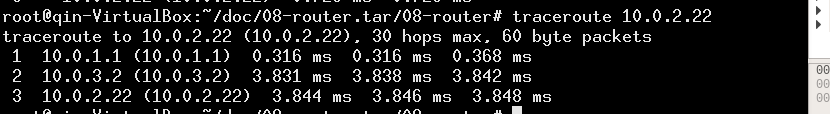
新拓扑ping 10.0.1.1



新拓扑ping 10.0.2.22



新拓扑ping 10.0.3.2



Traceroute实现

1. 实验分析

根据上图所示结果，拓扑均已连通，节点之间通信实现，符合实验目的要求。

1. 实验总结
2. 这次实验难度总体很大，时间太短，并且要处理好多函数，因此实验的处理要求很难。
3. 这次实验的函数总体较多，实验前需要准确分析数据包传递流程，才能分清楚函数的详细作用，包括各个函数之间的调用关系，都需要仔细分析。
4. 实验中直接处理的是网络字节序的packet，需要进行子节序转换，而生成ICMP包和ARP数据包时又需要从主机字节序转换为网络字节序，需要认真考虑。
5. 实验中lookup函数和老化函数部分涉及到了锁的获取与释放，在函数返回的地方一定要释放锁，否则引起死锁现象。
6. 这次实验代码细节较多，debug较为困难，单纯采用打印数据的方式debug较为困难，最后traceroute更需要wireshark来观察，时间紧迫，这次实验在看了很久没有头绪后参考了别人的思路，实现过程中避免了一定的bug出现。