Lab9 实验报告

高鸣驹 2019K8009907015

1. 实验任务
2. 实验路由器，可以处理ARP请求和应答，进行ARP缓存管理，IP地址查找和IP数据包的转发，以及ICMP数据包的发送。
3. 手动构造一个包含多个路由器节点的网络，并且进行连通性测试和路径测试。
4. 实验设计
5. ARP请求和应答

我们在处理ARP请求和应答包的时候，主要是组包，我们以请求包为例进行分析：

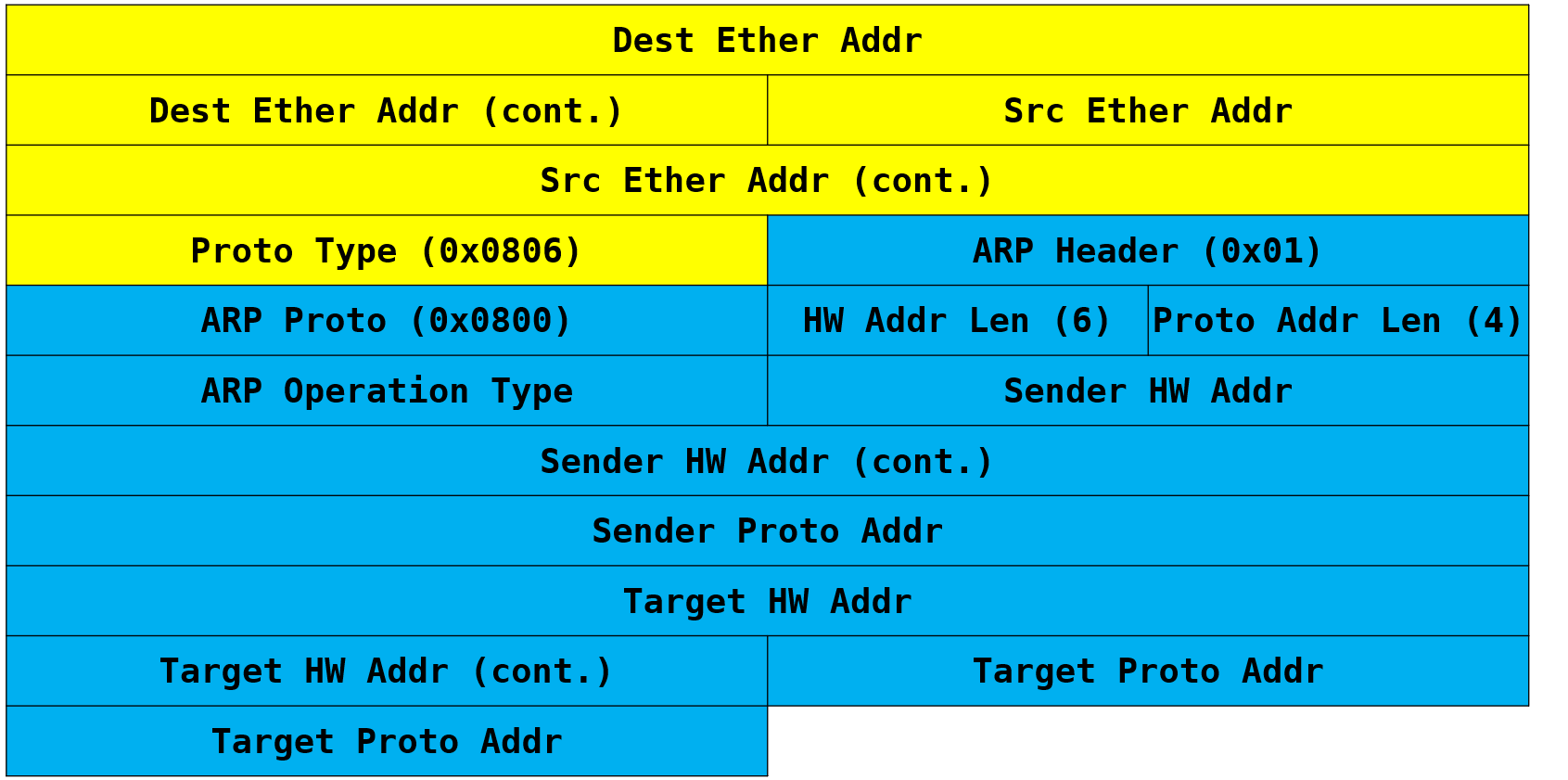


图1：ARP包格式

我们的Src Ether addr可以从当前端口的配置信息得到，目标的MAC地址在不知道的时候填上全1，协议号按照上表填写0x8086即可：

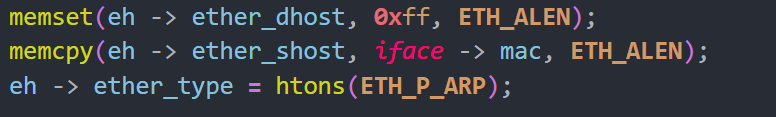


图2：ARP包的Ether Header填写

接下来我们填写ARP Header的内容，这里的Sender Proto Addr和Target Proto Addr为发送方和接收方的IP地址，Target HW Addr不清楚的情况下设为全0：

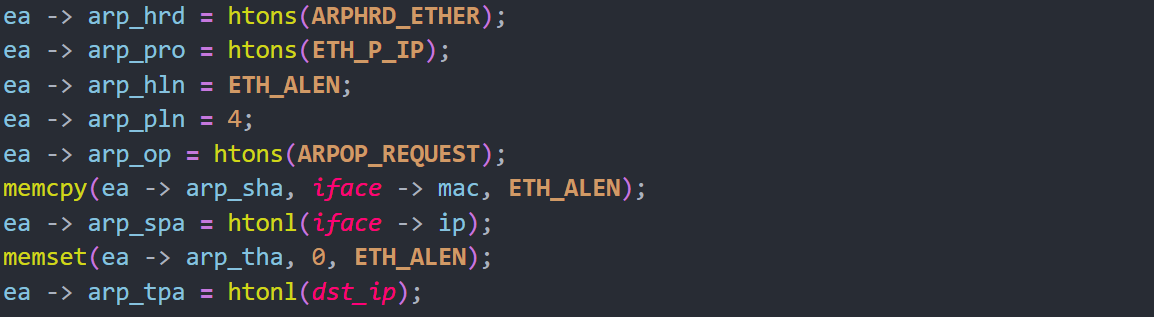


图3：ARP包的ARP Header部分的填写

Reply包的格式和Request包的格式基本一样， 但是在Reply包中，这个包的目标的MAC地址我们是知道的，所以对应字段要填写上对应的MAC地址，而不能是全0或者全1。

1. ARPCache的管理

每个表项里存有IP地址和MAC地址的映射，同时还有这个表项是否有效的valid位，与这个表项加入的时间added。

1. 查找IP地址对应的MAC地址

这个比较直观，我们遍历所有表项，如果有表项的IP和我们要查询的IP相等并且表项有效，我们就找到了对应的MAC地址：

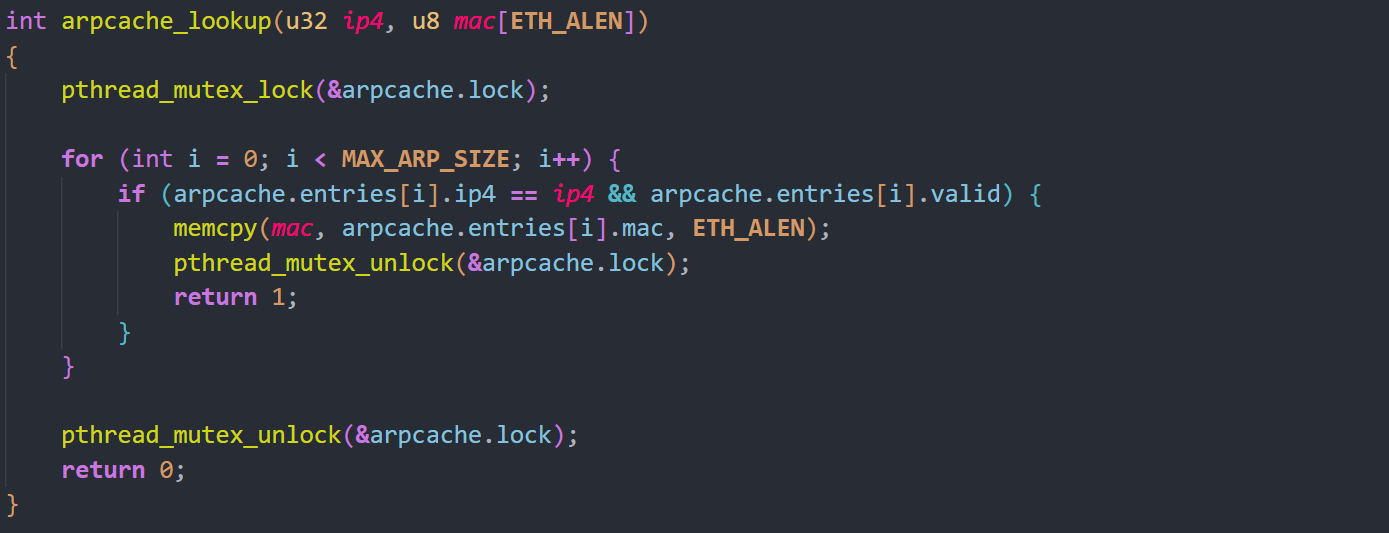


图4：查询IP地址对应的MAC地址

1. 缓存查不到相应条目而等待ARP应答的数据包

我们的数据包查不到IP地址对应的MAC地址时，要挂到ARPCache的一个等待队列里，直到我们收到这个IP地址对应的MAC地址，再释放这个包。具体等待队列的结构如下：

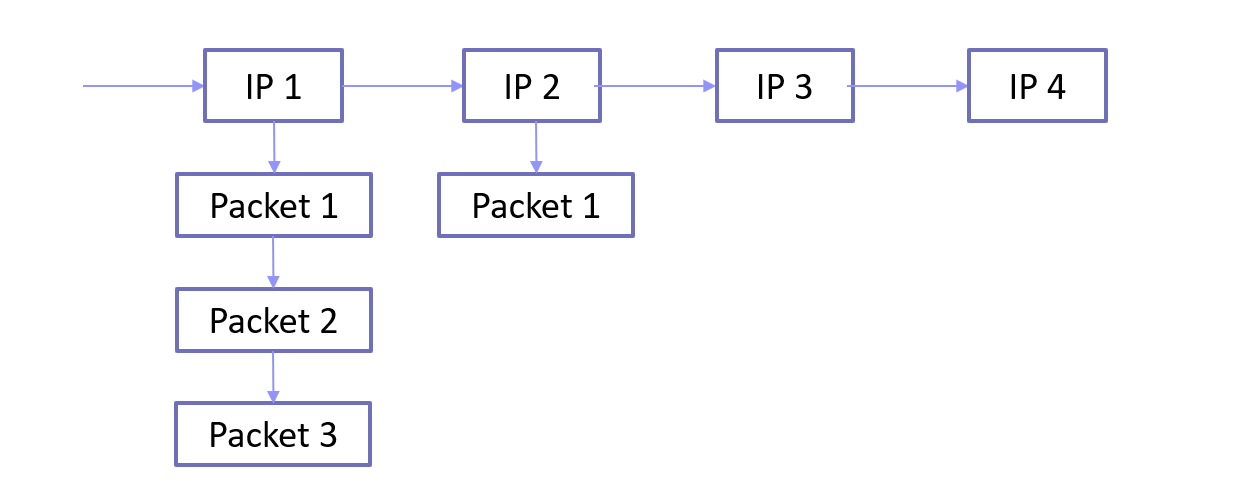


图5：等待队列的结构

首先我们先寻找我们这个要挂载的这个包寻找的IP是否已经存在，如果存在，我们直接挂在这个IP的链表上：

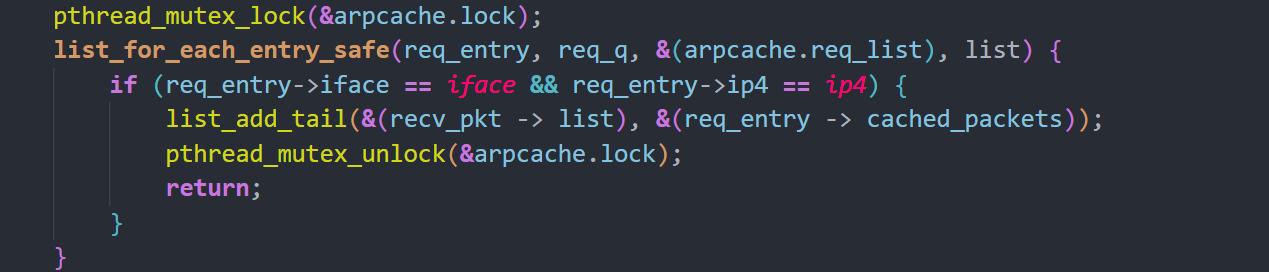


图6：直接挂到现有IP的等待队列里

如果没有，我们首先要创建对应的一个这个IP对应的arp\_seq链表项，再把数据包挂到这个链表项的等待队列上。挂载完后，我们发出arp请求查询对应的mac地址：

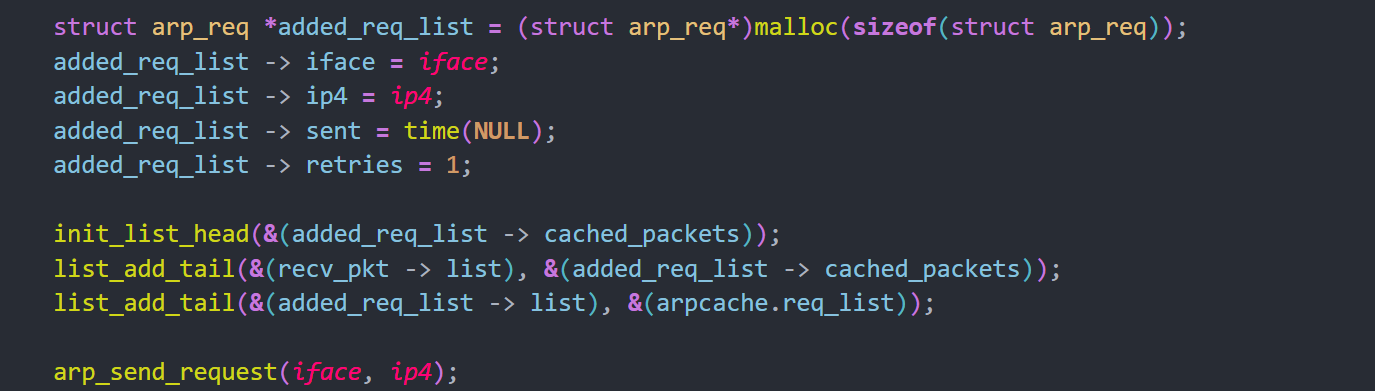


图7：创建新的arp\_seq然后挂载数据包

1. 插入IP -> MAC地址映射

我们插入映射时，需要找一个地方插入，如果有空闲表项（valid=0），我们插入这个空闲表项，如果没有我们随机替换一个。所以我们首先需要遍历整个表寻找空闲表项，找不到再随机替换：

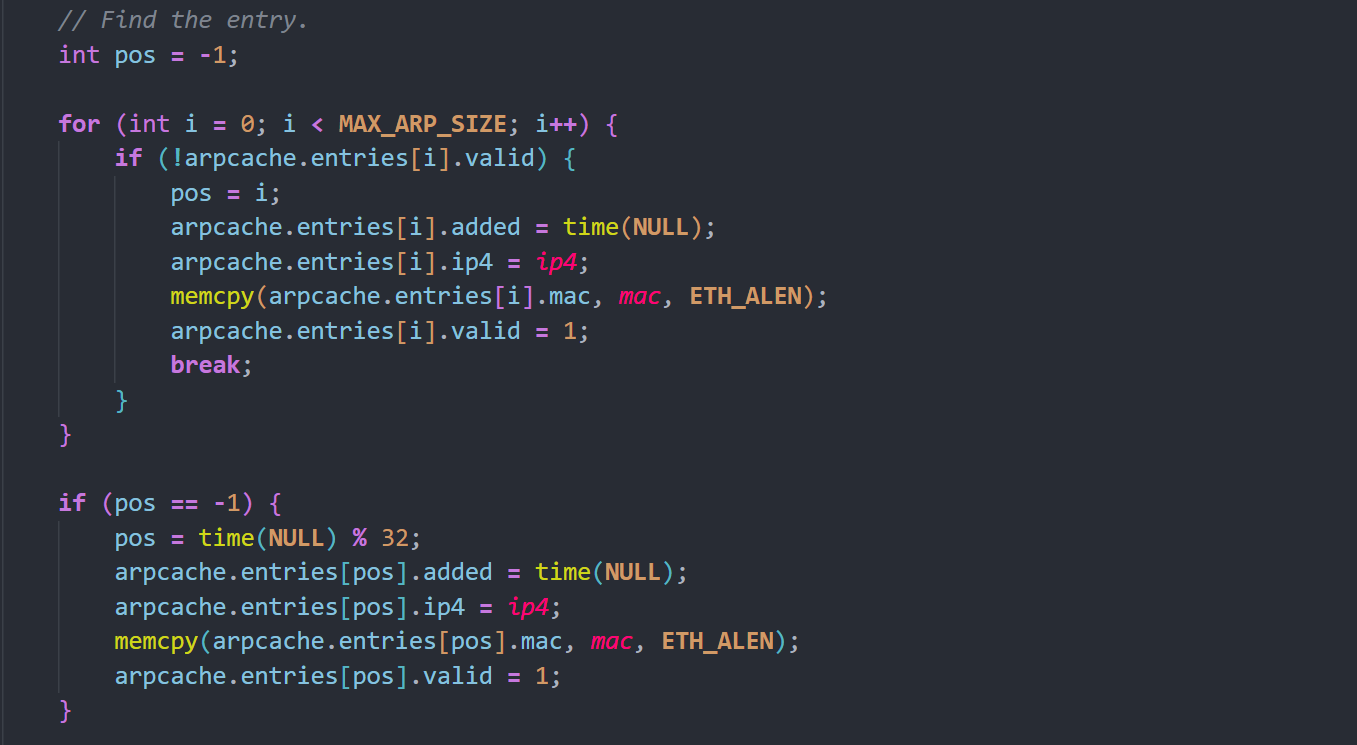


图8：找到插入位置并插入

之后，我们需要释放所有的pending packets，因为我们已经查到了他们所需要的MAC地址：

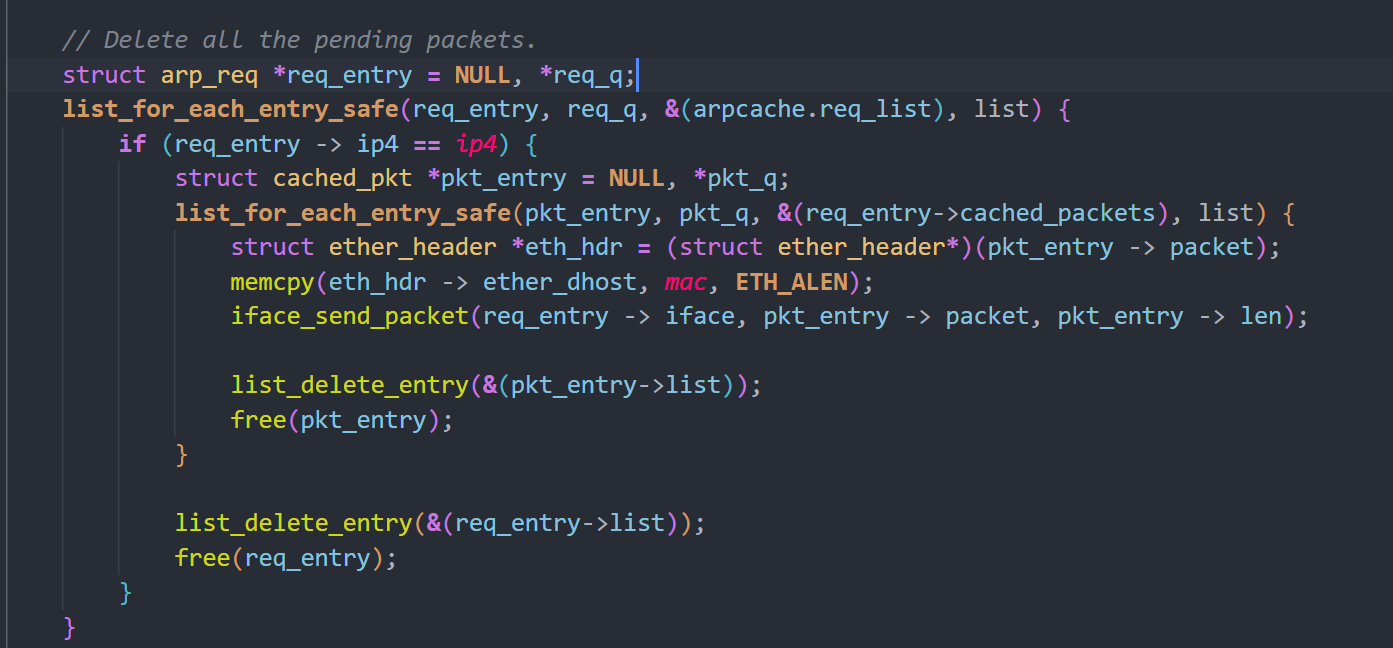


图9：释放所有的pending packets

1. 清理表项和包缓存

这个主要有两部分，第一部分是清楚存在时间过长的表项，第二部分是清除请求次数过多的pending packets。

首先我们记录下现在的时间，和表项加入的时间对比，如果表项的存在时间超过了最大时间，则删除表项，即置valid = 0：

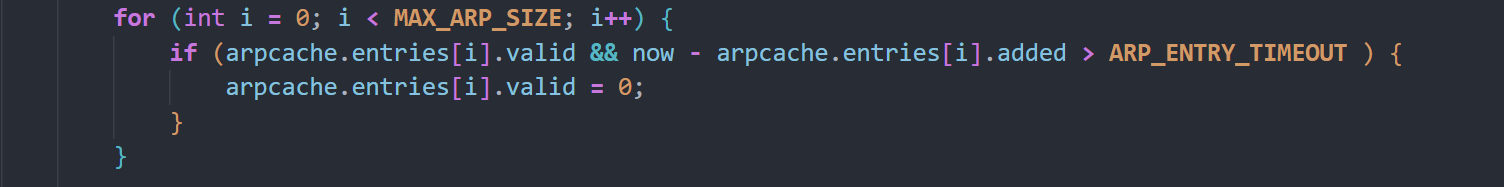


图10：清除过期表项

接下来我们遍历所有的arp\_seq，看他们发arp请求包的次数，如果没超过最大限制则继续发请求包，如果超过了，我们清除这个arp\_seq下面挂载的所有pending packets，然后回复icmp数据包，回复目的地不可达：



图11：清除请求次数过多的pending packets

1. ICMP数据包发送

icmp数据包的格式如下：

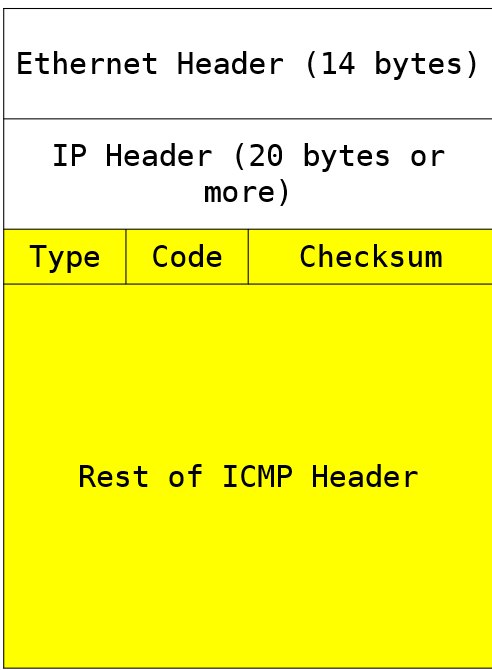


图12：icmp的数据包格式

我们首先确定ICMP数据包的长度，当收到ping包时，我们的长度和ping包是一样的，但是其它的，我们的rest of icmp header要拷贝收到的IP数据包的头部和随后的8字节：

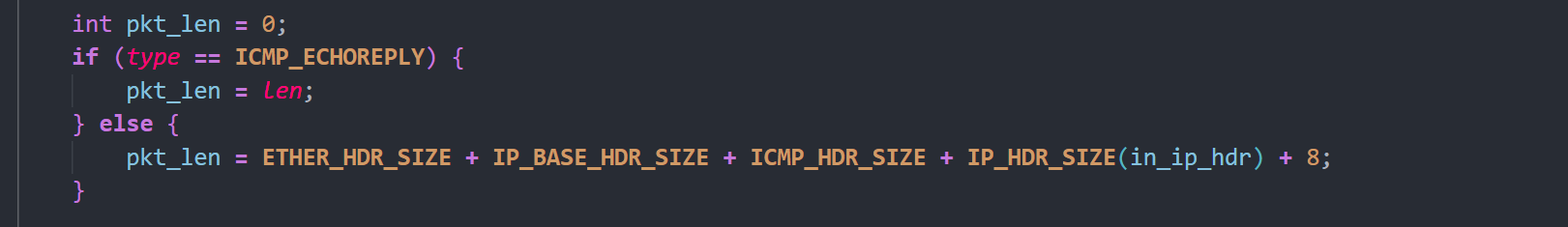


图13：icmp包长度的确定

确定长度后，我们开始填充各部分的内容，Ethernet Header我们只需要填充type就可以，其他字段在发送时会自动填上，IP字段需要我们查找本端口的ip地址，然后调用ip\_init\_hdr函数进行ip字段的填充，最后的icmp的type和code直接填即可：

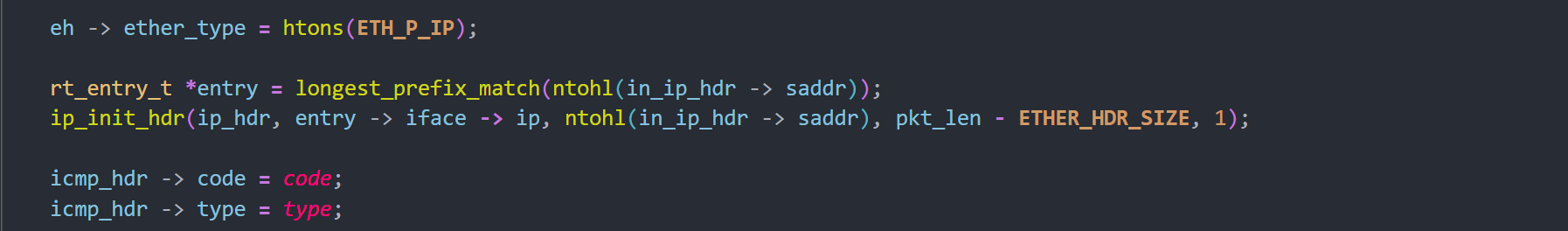


图14：icmp包部分字段的填充

接下来就是rest of icmp header的拷贝，具体的拷贝内容开始已经提过，逻辑如下:

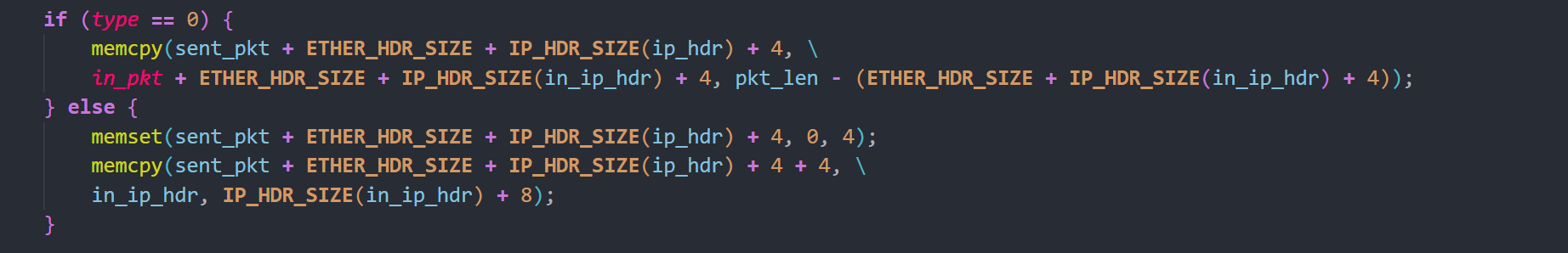


图15：rest of icmp header的填充

最后发送这个数据包。

1. IP地址查找与数据包的转发

我们首先完成转发ip数据包的操作，首先我们需要进行最长的前缀匹配，即遍历路由表，找到对应的路由器表项且mask位数最多，然后进行转发。



图26：最长前缀匹配

在得到对应的路由器表项后，我们看它的网关地址，如果为0，说明数据包已经到达了目的主机，这时我们下一跳地址设为目的地址，否则为查到的路由器表项的网关地址：



图27：路由器转发的逻辑

我们收到ip数据包，也要对数据包进行解析，这就需要handle\_ip\_packet函数。

首先我们先看目的地址和目前的iface的addr是不是一样，如果一样说明这是ICMP echo请求，我们返回相应的reply即可：

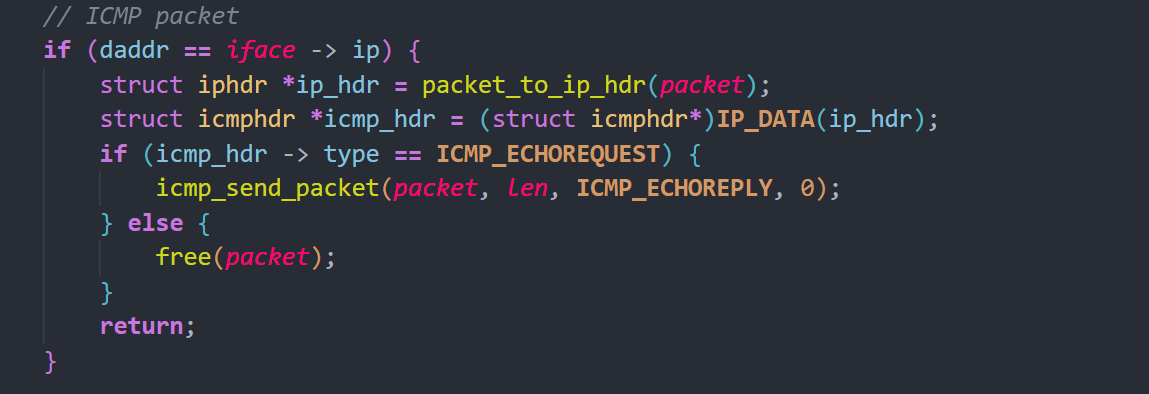


图28：ICMP ECHO数据包的处理

第二部分我们在路由表查找下一跳ip地址，如果查不到，说明我们无法到达，返回icmp unreachable异常：

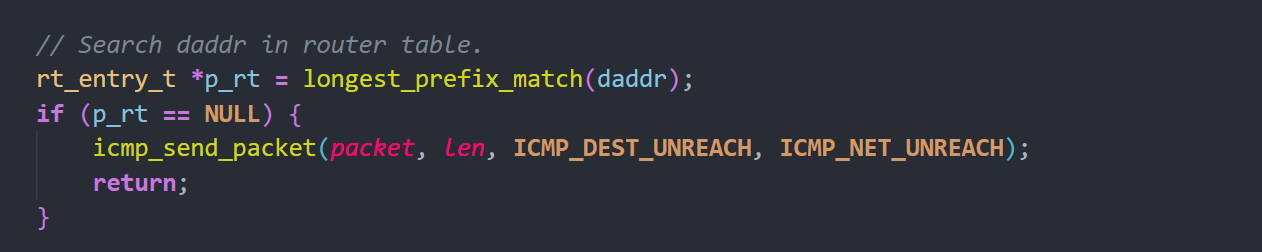


图29：ICMP\_DEST\_UNREACH的处理

第三部分我们处理是否超时：

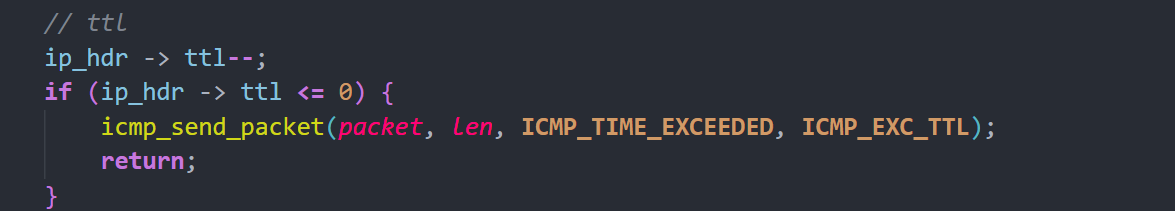


图30：超时的处理

最后一部分，我们处理ip数据包的转发：

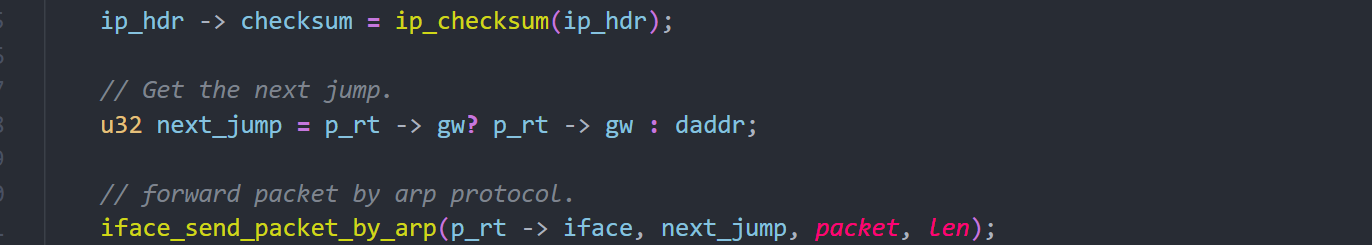


图31：ip数据包的转发

1. 实验过程
2. 路由器功能的测试

我们在h1上ping h2, h3：

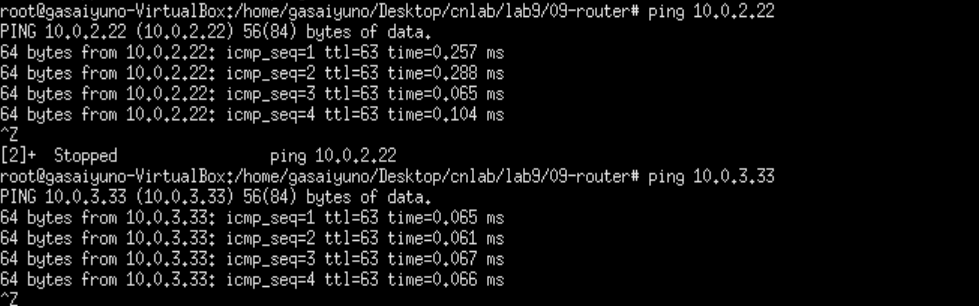


图32：在h1上ping h2, h3的结果

我们可以看出，可以ping通。

我们ping 10.0.3.11：

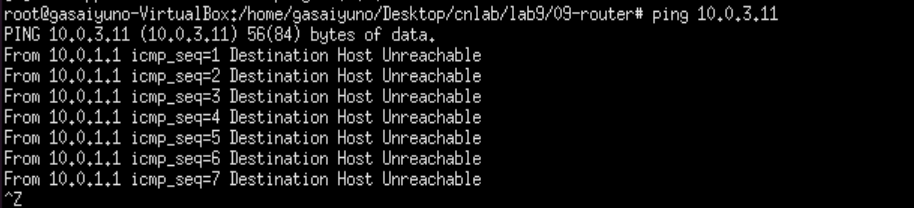


图33：在h1 Ping 10.0.3.11的结果

我们可以看出，这里符合预期结果：Destination Host Unreachables

我们接着ping 10.0.4.1：

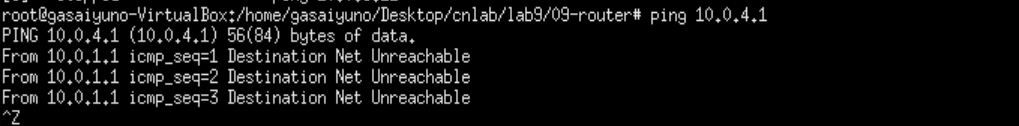


图34：在h1 ping 10.0.4.1

符合预期：Destination Net Unreachable.

1. 连通性和路径测试

我们的结构如下图所示：

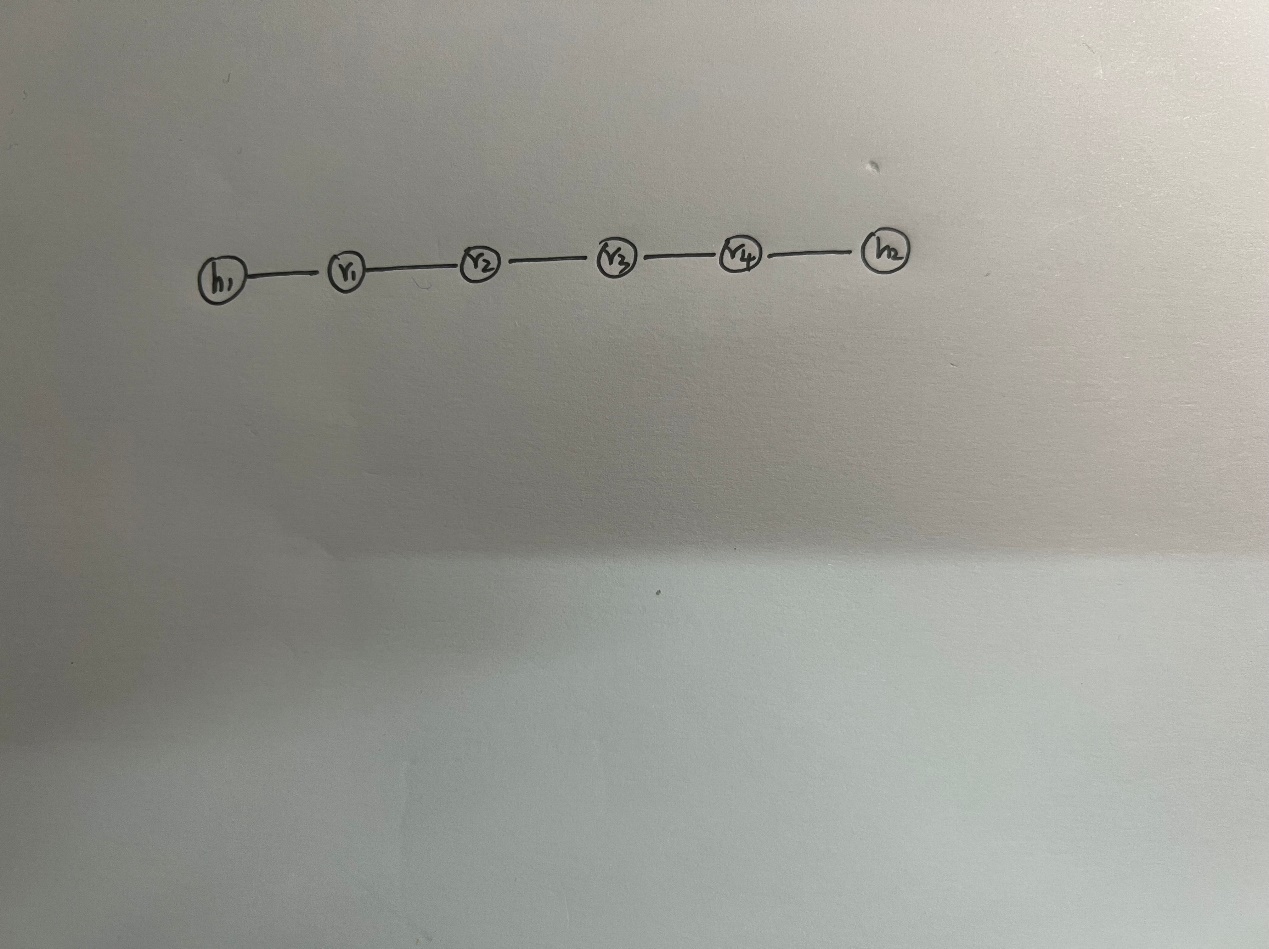


图35：拓扑结构

我们进行连通性测试：

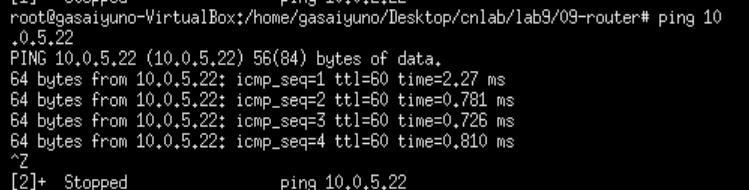


图36：h1 ping h2

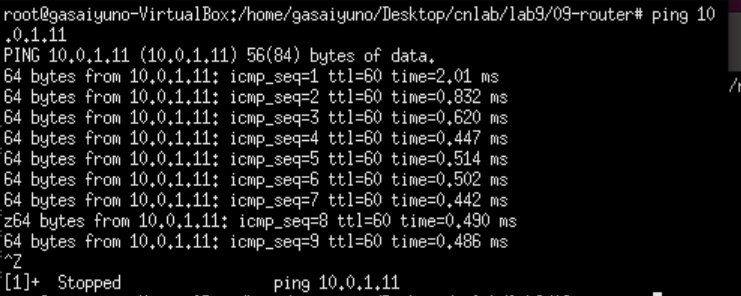


图37：h2 ping h1

我们可知，连通性测试通过。

我们的traceout测试：

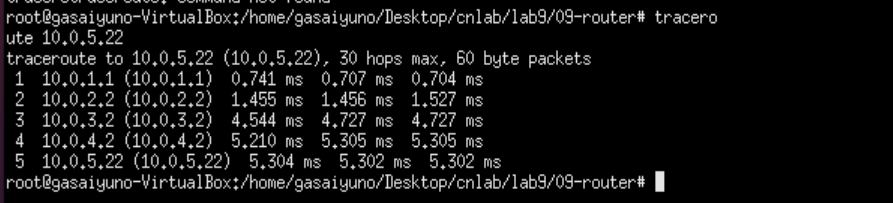


图38：h2 ping h1的traceroute测试

我们可以看出经过的路径是正确的，从r1-r2-r3-r4-h2:

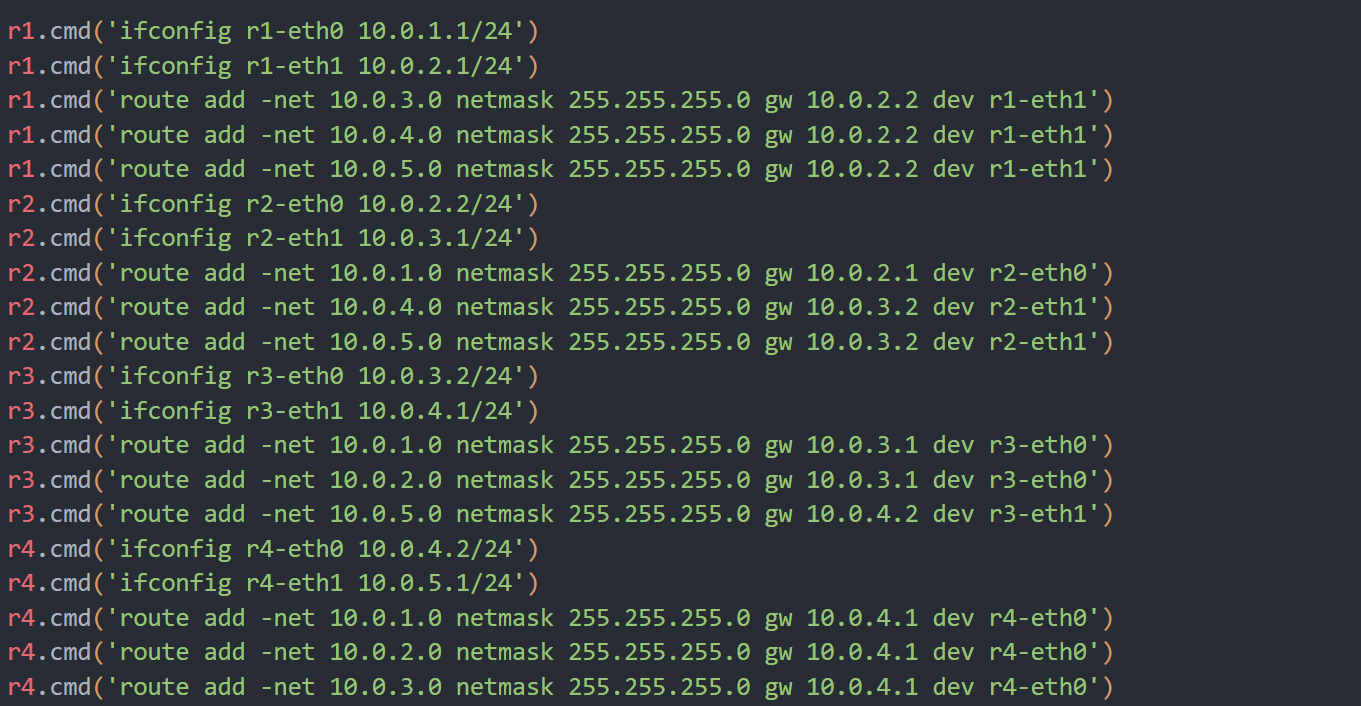


图39：r1-r4的地址