Lab5 实验报告

1. 实验任务

1. 实现对数据结构mac\_port\_map的所有操作，以及数据包的转发和广播操作。

2. 使用iperf和给定的拓扑进行实验，对比交换机转发与集线器广播的性能。

1. 实验设计
2. 转发表结构

我们的转发表结构如下图所示：



图1：MAC转发表的图形化结构

在程序里这样表示：

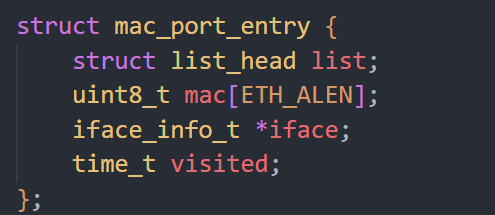


图2：转发表表项的C语言表示

我们用链表把每个表项串联起来，所以有一个list域。然后mac数组存放mac地址，iface存放端口信息，即从哪个端口来的；visited是我们上次访问这个端口的时间，用来更新老化。

但是我们如果把所有的表项全部只用一个链表连起来，查找会花费很长的时间，因此我们采用hash表，映射mac地址。Hash冲突通过链表解决；具体如下图：

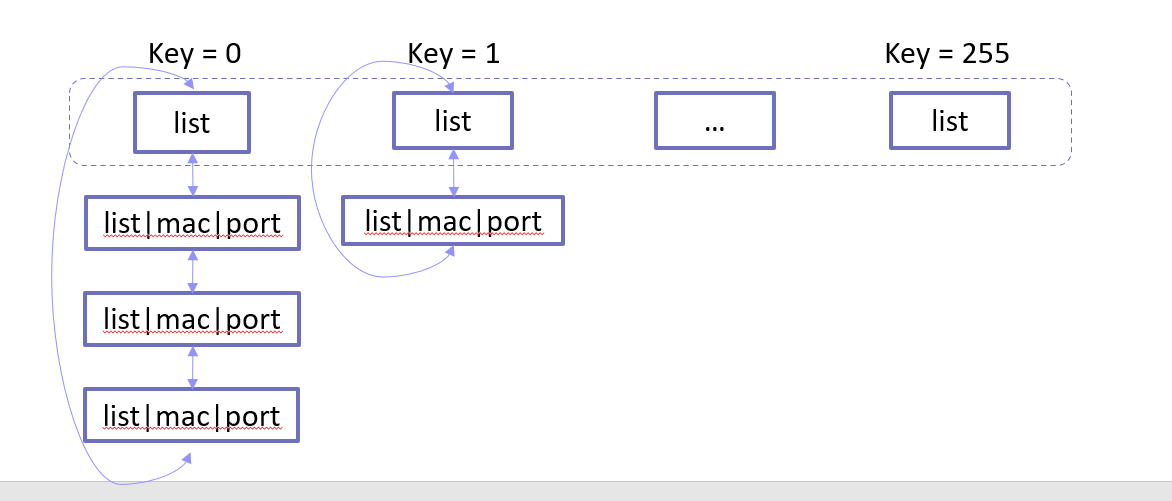


图3：利用hash表存放表项

1. 查找端口

如果我们需要查找某一个端口，我们首先利用hash函数映射这个端口的mac地址到0-255，再遍历这个读取这个hash表项，如果有hash冲突，则遍历这个表项上挂的链表。

注意，我们对转发表进行任何操作时，都要上锁，以防出现冲突：

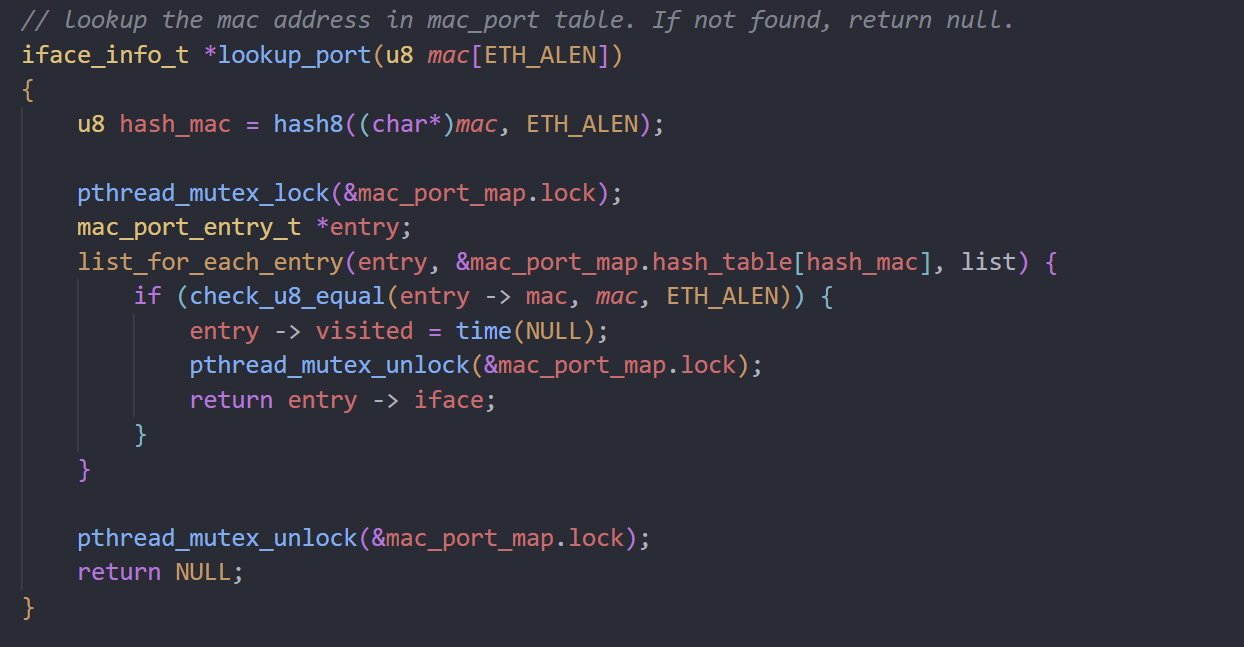


图3：端口的查找操作

同时注意，在查找到这个端口时候之后，我们需要更新这个端口的老化时间。

1. 插入表项

我们如果要在转发表之中插入表项，我们首先需要给这个表项分配内存空间。然后在这片内存空间插入这个表项的信息（mac地址，iface，访问时间等）。最后将这个表项插入哈希表表项链表的末尾。具体逻辑如下图所示：



图4：表项插入操作

1. 扫除过期表项

如果我们一个表项30s没有访问，我们就要把这个表项删除；我们遍历整个hash表，如果发现某个表项的访问时间和现在差的时间大于30s，就删掉这个表项，具体逻辑如下：



图5：扫除过期表项的处理逻辑

我们在初始化的时候会调用一个线程，间歇性的调用扫除函数：

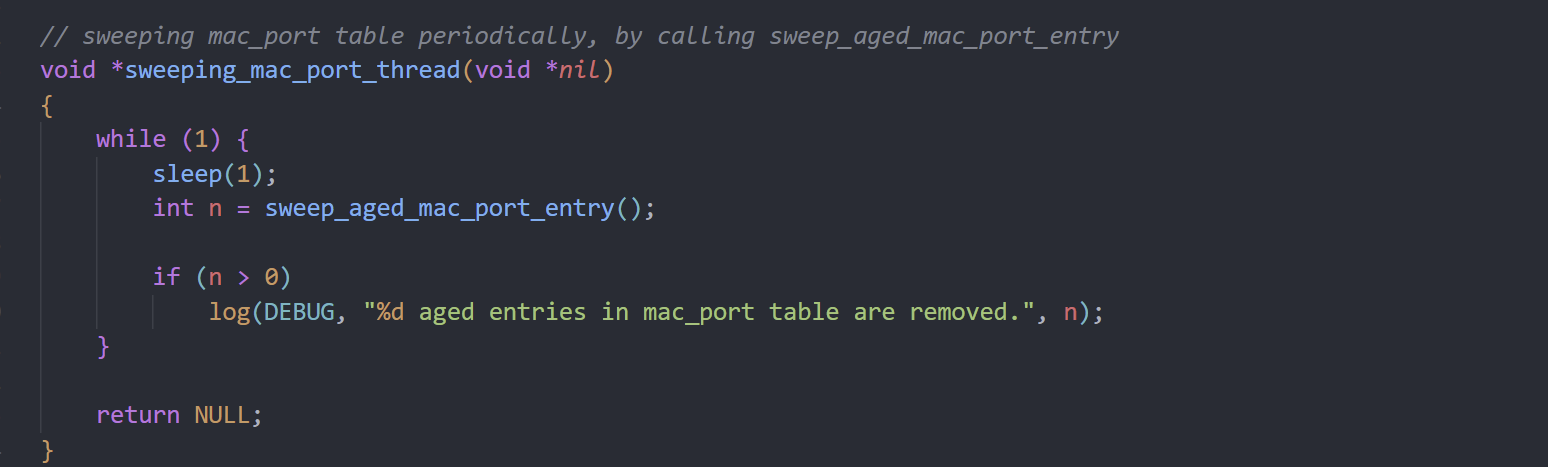


图6：间歇性调用扫除函数

1. 处理逻辑

我们的处理逻辑：如果我们收到一个数据包，先查询看发送的mac地址有没有对应的表项，如果有的话直接发到对应的端口，如果没的话则广播这个包。这些工作做完后，如果我们没在表中查到源端口对应的表项，我们将其插入，具体代码逻辑如下图：

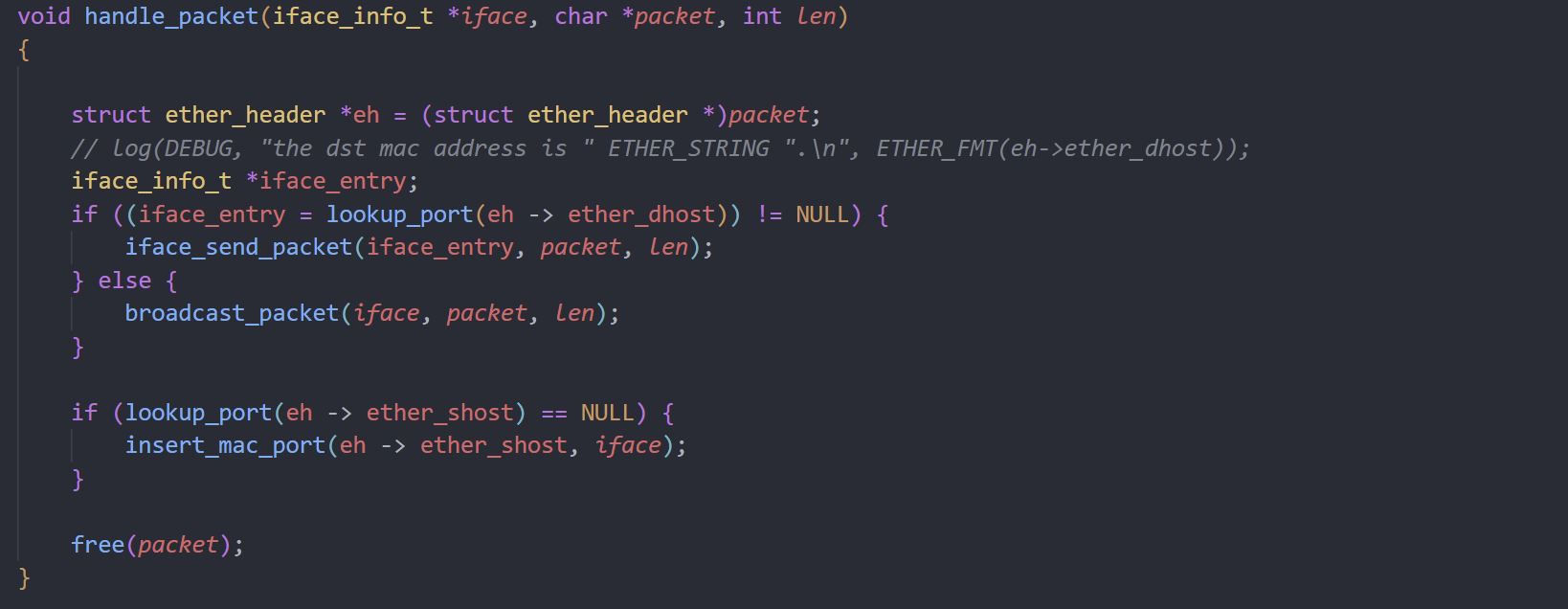


图7：总的处理逻辑

1. 性能比较：

当h2做server，h1, h3做client时候：

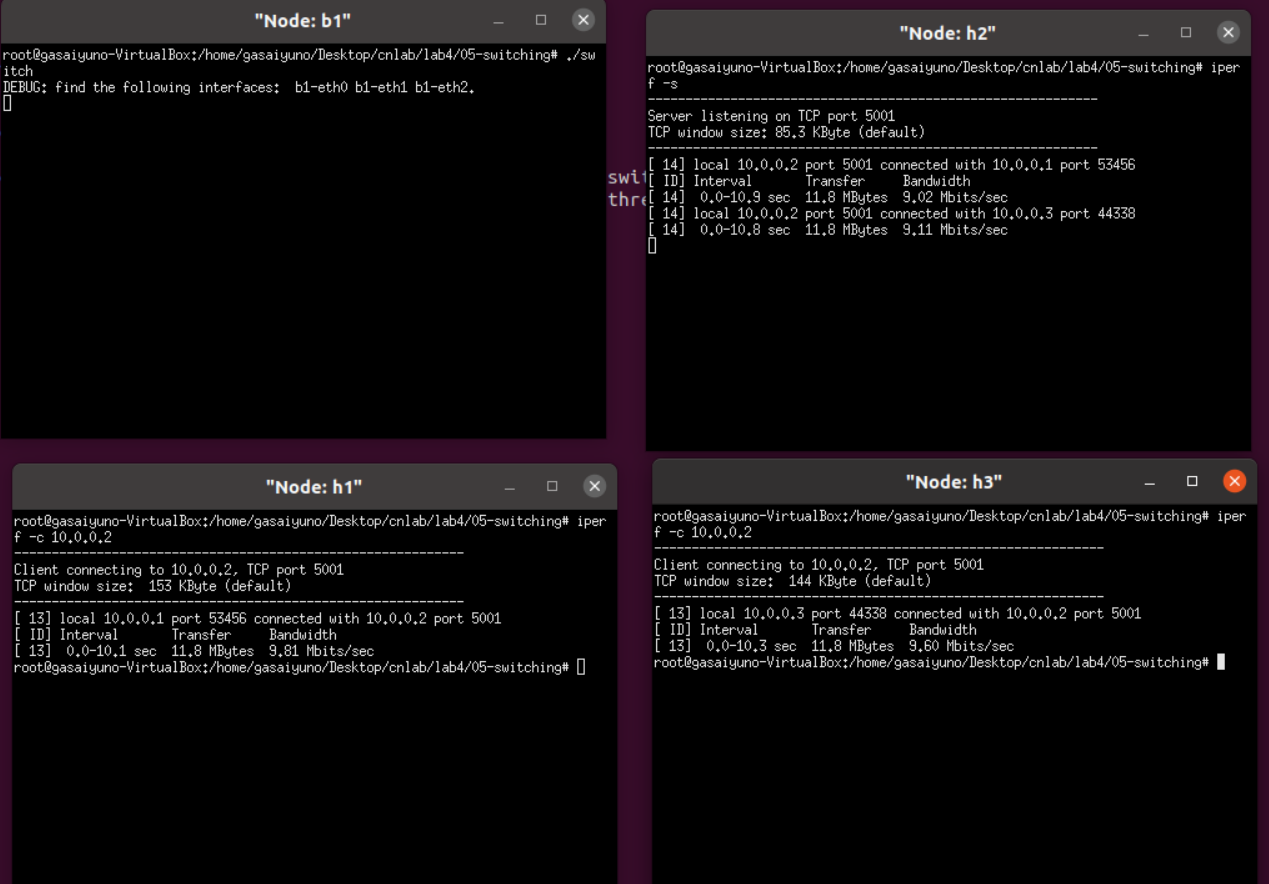


图8：交换网络性能测试

我们的速率为9.81Mbps，利用率约为98.1%。

同样的情况下，如果是hub：

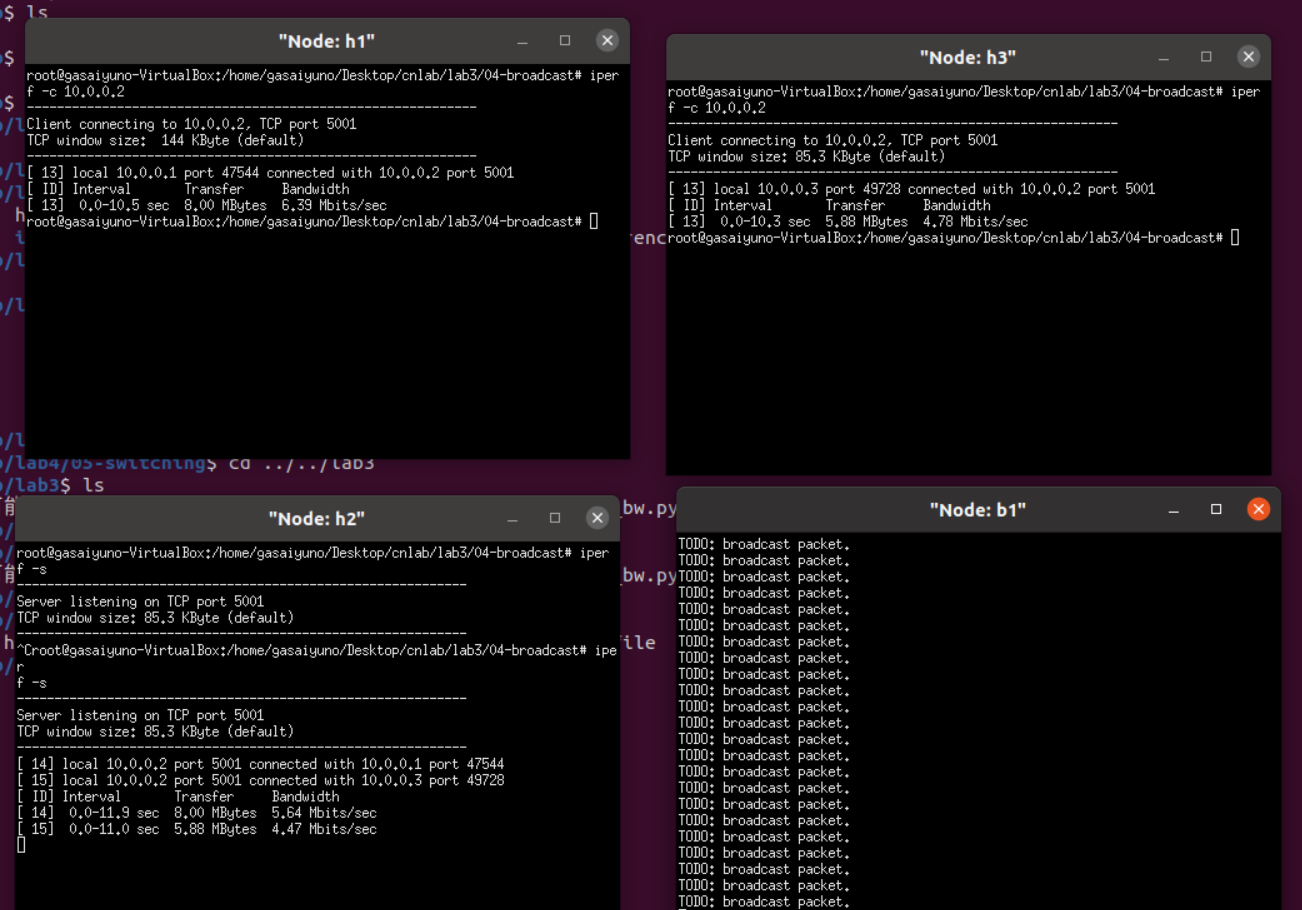


图9：广播网络性能测试

我们的效率一个为63.9%，一个为47.8%，明显低于交换网络。

1. 实验思考
2. 交换机在转发数据包时有两个查表操作：根据源MAC地址、根据目的MAC地址，为什么在查询源MAC地址时更新老化时间，而查询目的MAC地址时不更新呢？

这时候不能保证目的目的地址一定没错，只能保证来源正确。

1. 网络中存在广播包，即发往网内所有主机的数据包，其目的MAC地址设置为全0xFF ，例如ARP请求数据包。这种广播包对交换机转发表逻辑有什么影响？

交换机的转发表遇到目标为全0xFF的数据包，会进行广播，此时数据包会记录源数据MAC地址到交换机的映射表。这对于设备之间第一次通信十分重要。当转发表没有建立起来，我们只能通过广播通信，不断学习完善转发映射表。

在这个过程帮助所有参与的交换机学习建立起了映射转发表。

1. 理论上，足够多个交换机可以连接起全世界所有的终端。请问，使用这种方式连接亿万台主机是否技术可行？并说明理由。

不行，这样转发表会膨胀的很厉害，网络的效率会很低。