**网络实验报告 exp3**

**2020K8009929032 刘耕印**

1. *广播网络实验*
2. 实验内容

在一个以广播网络节点hub为中心的星形拓扑结构中，在给定框架的基础上，为hub添加广播功能，使得hub能够把收到的数据包从除了收取端口以外的所有端口广播转发。实现完成后，以ping命令测试各个主机之间的连通性。测试无误后，以iperf命令考核各条链路的利用效率。然后，构建具有循环pub结构的拓扑，抓包验证该拓扑下会产生数据包环路。最后，对实验现象加以思考。

1. 实验过程
2. 实现hub的广播功能

为实现hub的广播功能，只需填充broadcast函数，遍历接口链表发送信息即可。其中利用到了链表的遍历宏。实现比较简单，代码截图如下：

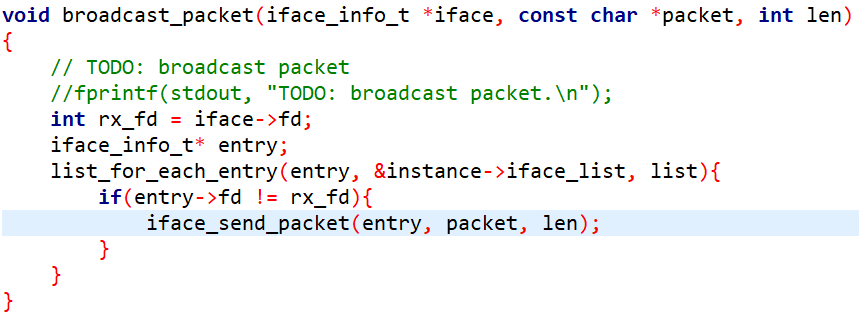


Figure 1 广播代码实现

简单地发送信息即可。接下来，运行脚本测试连通性：（ifconfig指出主机h1, h2, h3的ip分别是10.0.0.1, 10.0.0.2, 10.0.0.3）

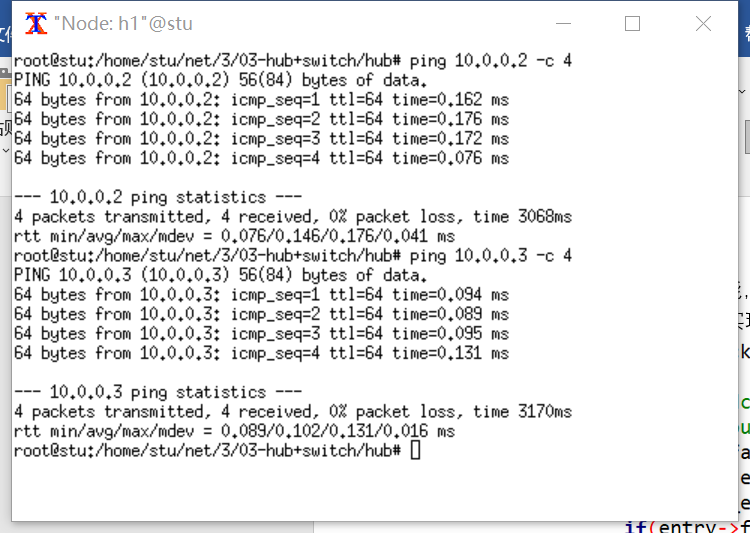


Figure 2 h1连通h2与h3

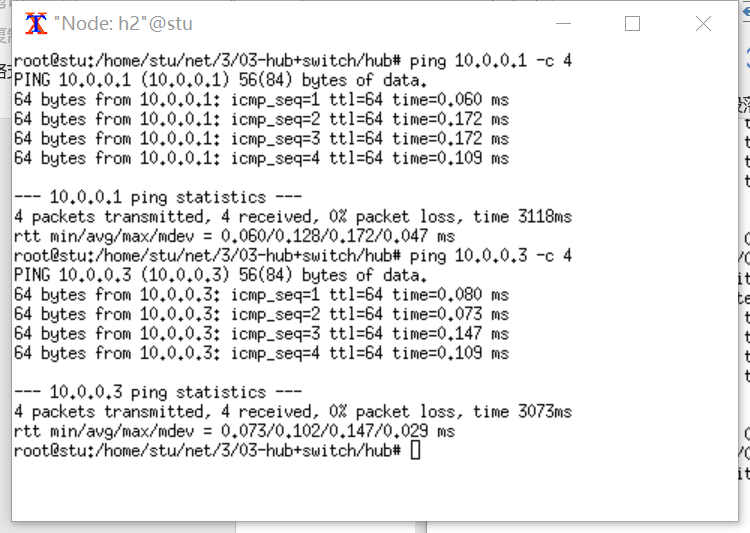


Figure 3 h2连通h1与h3

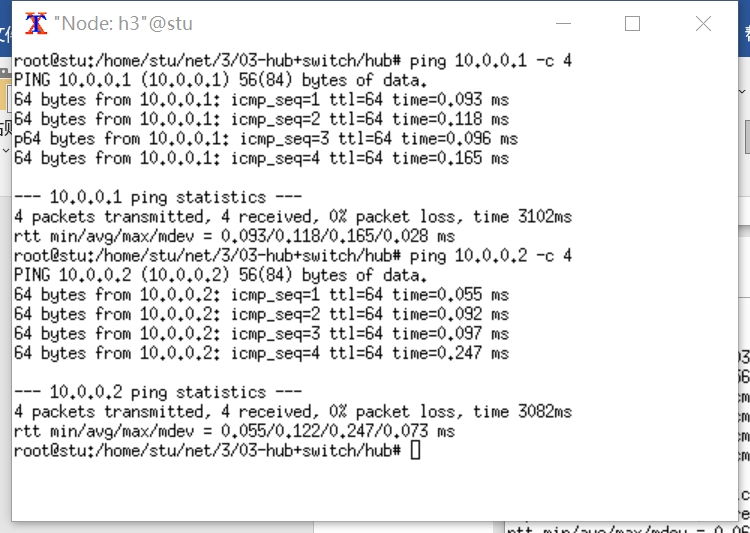


Figure 4 h3连通h1与h2

确认形成了全连通，功能正常。

1. 考察线路利用率

根据脚本，各线路的带宽限制为：

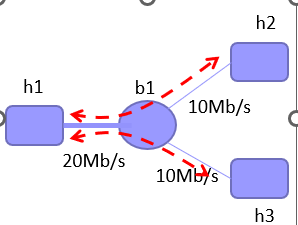


Figure 5 带宽限制

使用iperf命令，检测线路上的实际带宽。首先令h2与h3为服务器，h1为客户机，执行iperf：

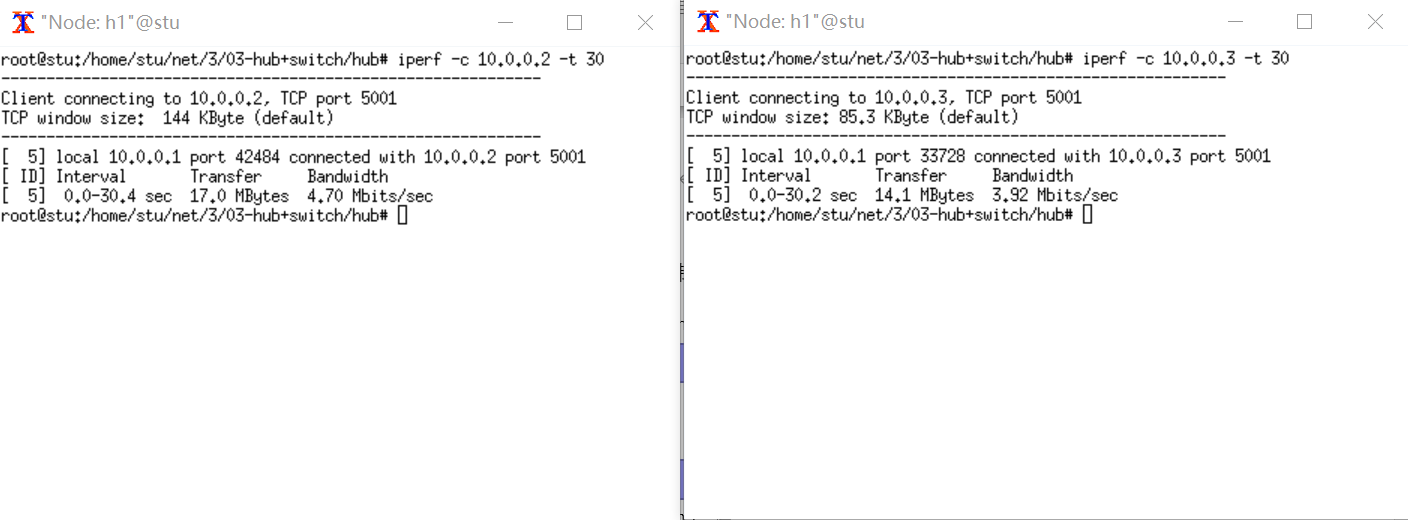


Figure 6 iperf执行效果（h1为主机）

可以看到，当h1同时向h2和h3发送信息时，带宽分别只有4.7M/s和3.9M/s，利用率比较低。接下来令h1为服务器：

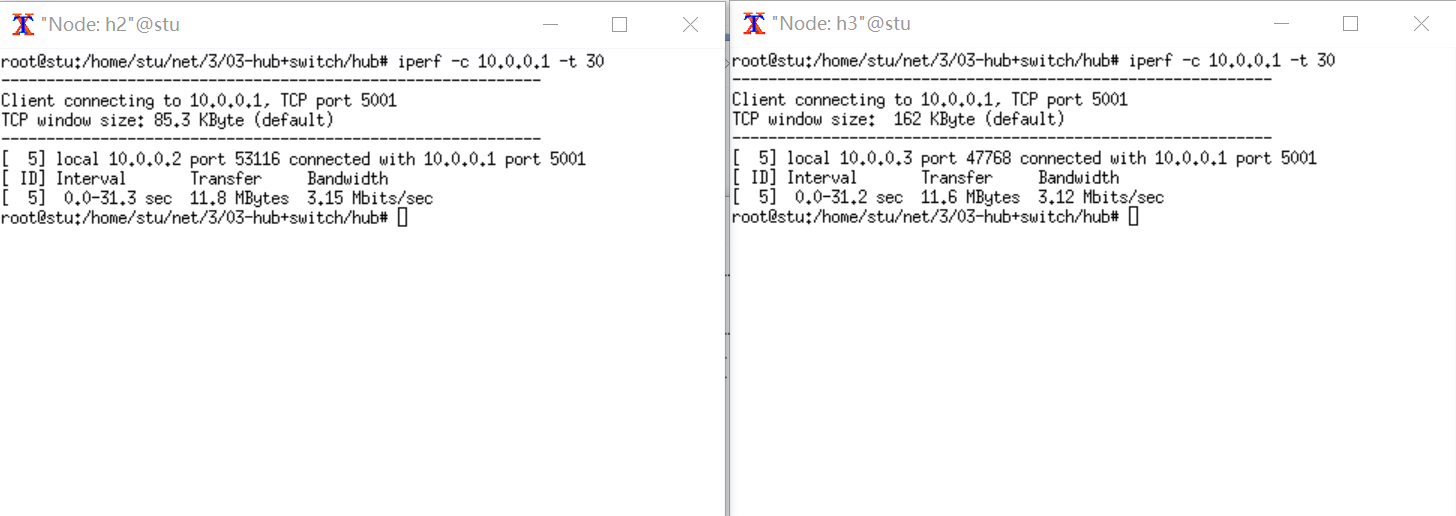


Figure 7 iperf执行效果（h1为服务器）

带宽均为3.1M/s左右。

1. 构建环路情景

修改脚本，使hub成环：

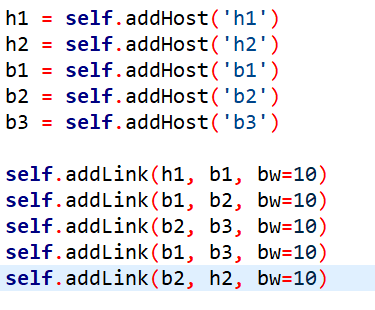


Figure 8 带有环路的拓扑

之后从h1向h2发送数据，使用wireshark监听：

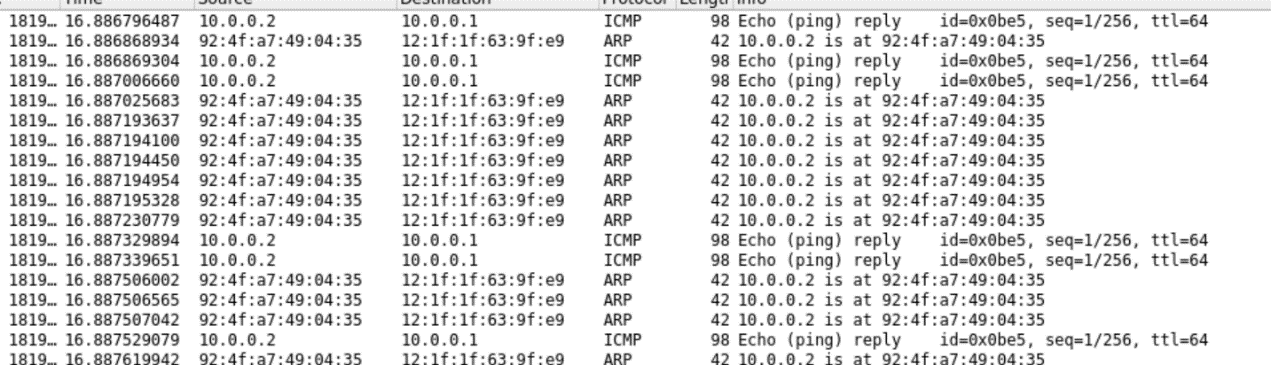


Figure 9 抓包记录

上图显示，有几个包被反复监听到，说明存在循环广播问题。

oj通过：

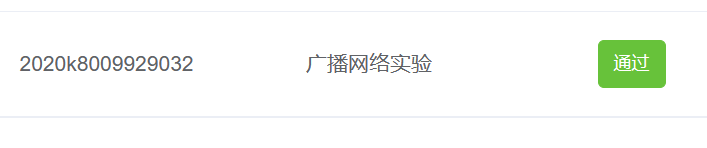


Figure 10 通过

1. *交换机实验*
2. 实验内容

为解决全广播的资源浪费问题，应使用交换机代替广播节点，只向接收方的方向转发数据。交换机通过学习构建mac地址-网口映射，在收到数据时，将mac地址和接收网口构成表项填入表中。每个表项拥有老化机制，30s未访问，即由清扫线程删除此表项，以维护映射表。本次实验实现对于映射表的查找和维护操作，并利用这些操作建立起交换机。

1. 实验过程

首先，在给定的框架之下，捋清交换机主体的构建过程：首先调用init函数对映射表和互斥锁进行初始化，这里建立的映射表拥有256个链表供散列加快查找速度。同时启动一个新线程来进行清扫。之后，每当交换机收到数据包时，首先考虑对其进行转发。使用lookup函数在本地的映射表中查找目标mac地址对应的网口，若找到则向该网口直接转发，否则进行广播。在转发完毕后，使用insert函数进行映射学习。在接收时，我们有源mac地址与入口，首先查找表中有无此映射。若有，则更新其老化时间；若有但映射关系不正确，更新映射关系并更新老化时间；若没有，则在对应链表中加入该映射关系。如此循环。

接下来，简单讲讲关键部分的实现细节：

最关键的两个操作是查找和插入操作。为了防止多次查找，我将对映射表的更新动作合并在了查找中。当需要对表项进行更新时，对查找函数传入更新参数和更新后网口：

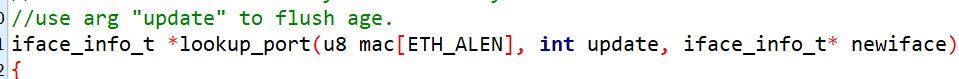


Figure 11 添加参数指示更新

不需更新直接传入空参数即可。

查找的具体动作则是首先由哈希函数确定目标在哪个链表中，之后用给定的宏遍历该链表进行条件判断即可。这里我使用的哈希函数是最简单的根据mac的最后一字节进行散列。代码如下：

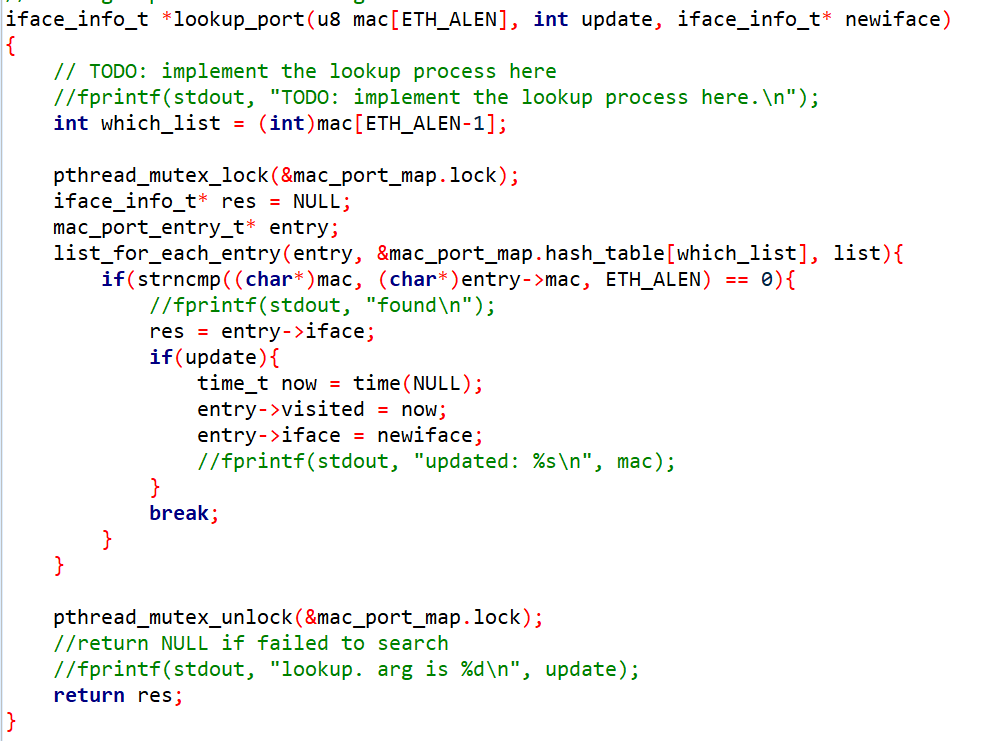


Figure 12 查找操作代码

其中应当以互斥锁来保护对映射表的访问。当查找失败时，将返回空指针。

插入的具体动作是先进行一次带有更新的查找，如果未找到，则对相应链表添加新表项。代码如下：

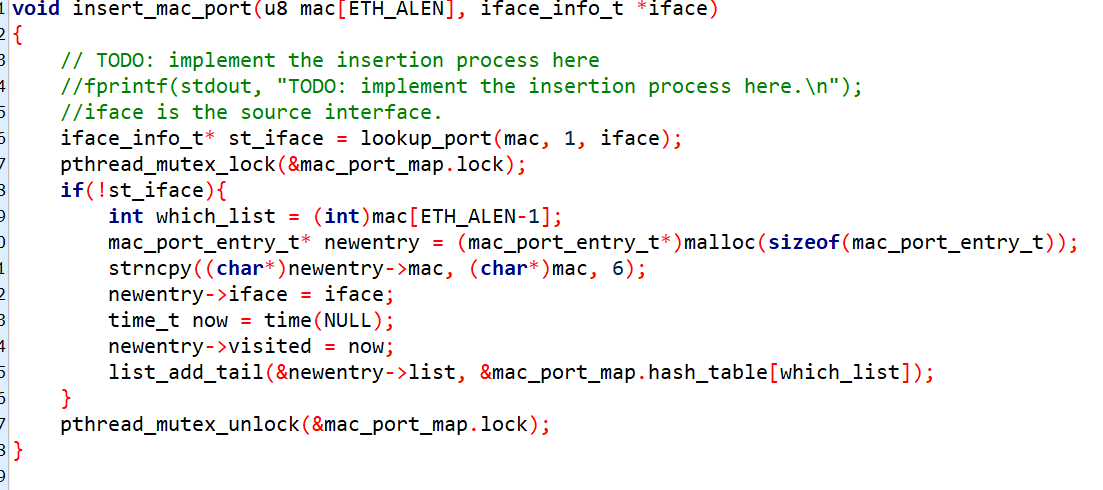


Figure 13 插入操作代码

同样以互斥锁保护，在添加表项时要使用与查找相同的哈希函数。这里还要提一下老化时间的概念，实际上是在每个表项被加入或更新时，将其老化时间成员设为当前时间，清扫线程根据清扫时时间与记录时间的差值判断是否清扫。这一点在接下来的清扫线程中有所体现。

下面是清扫线程的实现。清扫时遍历整个映射表，检查它们的老化时间是否超过上限。若超过时，将其从表中删除。这里使用的遍历操作要支持增删。清扫线程所循环调用的清扫函数如下：

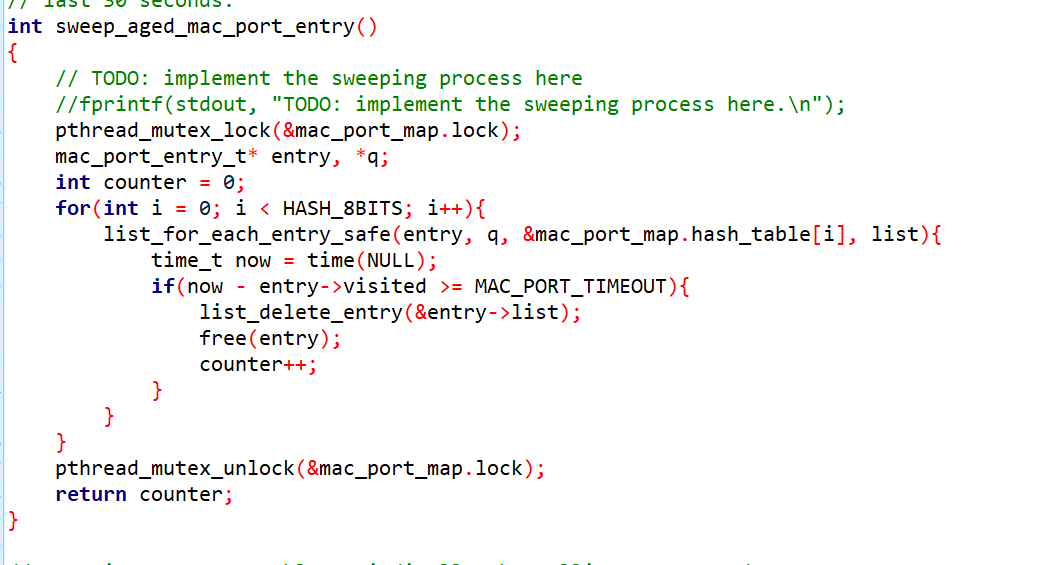


Figure 14 清扫时代码

可以看到判断是否超时是使用的当前时间减去记录时间。

最后，在handle函数中添加查找、发送和更新的操作即可。

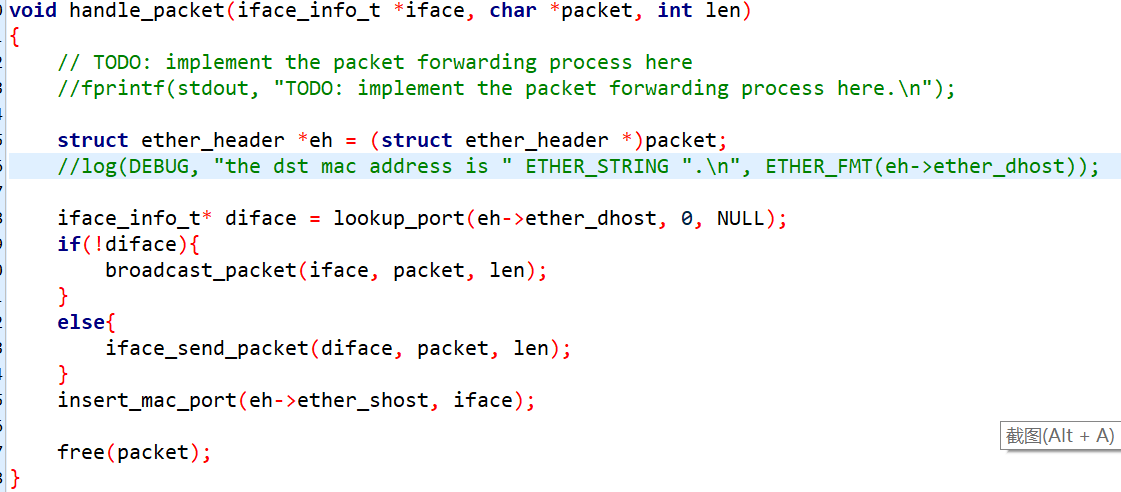


Figure 15 一般处理流程

其中以太头是在包的头部。做完以上操作并删除输出语句后，提交通过。

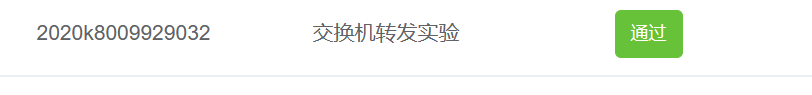


Figure 16 通过

1. 性能分析

在将中间节点改为交换机后，理论上带宽的利用率将较广播版本有所提升。下面使用iperf工具测试。拓扑和带宽限制仍与广播测试时相同，首先让h2与h3为服务器，h1同时对二者发起请求。效果如下：

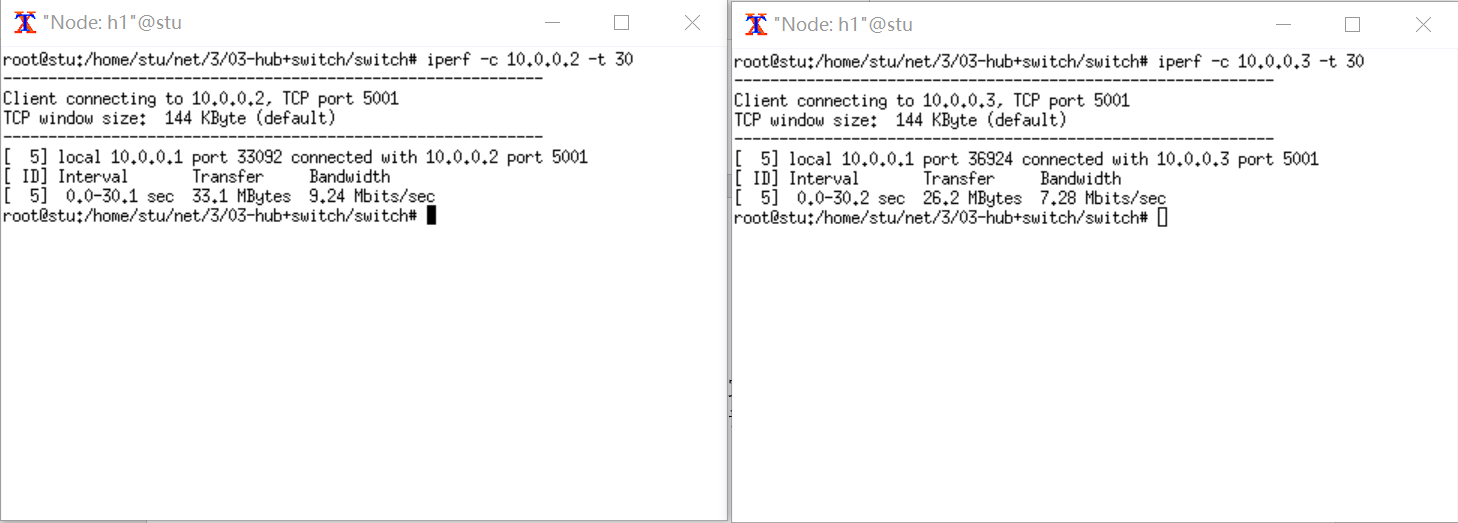


Figure 17 h1为客户机，h2与h3为服务器

带宽利用率分别为9.24/10和7.29/10，较广播机版本有了很大提升。

然后让h1为服务器，h2与h3发送请求：

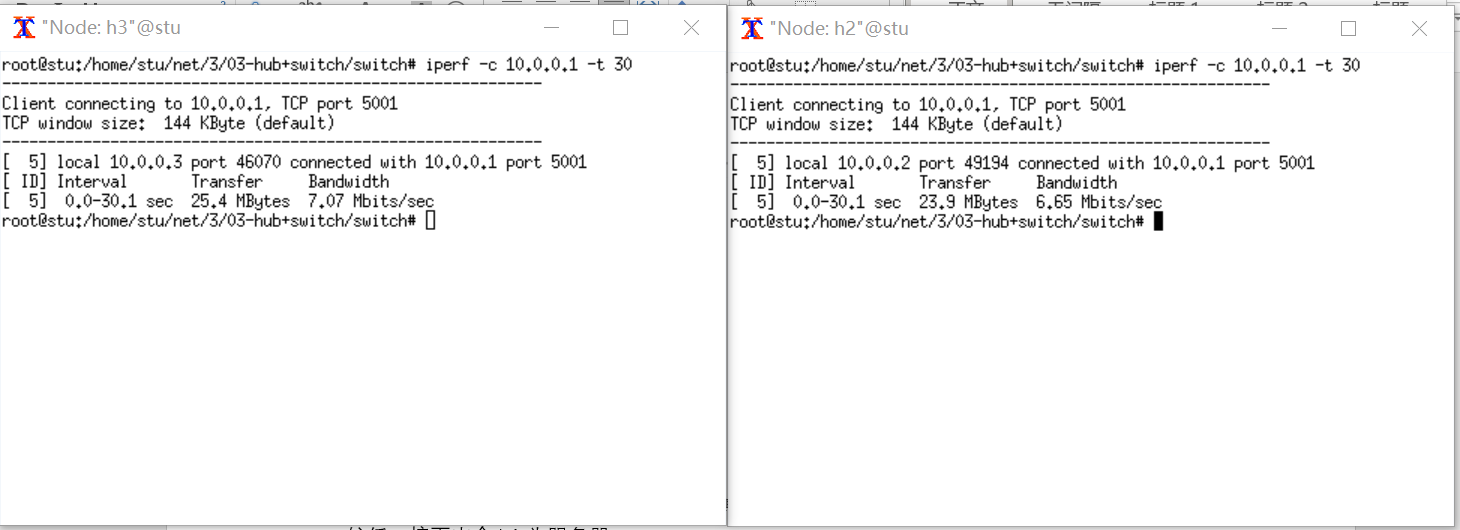


Figure 18 h1为服务器，h2与h3为客户机

利用率分别为7.07/10和6.65/10，也有所提升。这个提升幅度不是很理想，可能与虚拟机性能有关，oj上的测试是通过的。

1. *思考*

课后有三道思考题，现解答如下：

1. 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？

要回答这个问题，首先应该明确的是，更新老化时间的目的是什么。老化时间的唯一作用是作为清扫线程是否将本表项清扫掉的依据，更新老化时间将使得本表项更晚被删除。我们知道，如果一个表项长时间没有被访问，很可能是它所对应的主机已经脱机，此时继续保留这个表项造成了不必要的空间和时间开销。更进一步，如果主机在网络中进行了漫游，导致它所对应的网口发生改变，那么查表所得到的将是错误的网口，向此网口发送信息将导致丢失。考虑这样一种情景，某主机在改变网口后再也没有发送过消息，但它需要长时间接收消息。此时向它发送信息使用的是映射表中变化前的表项。如果查询目的MAC地址时也更新时间，那么这个错误的表项将永远不会被删除，信息总是在丢失。所以，查询目的MAC地址时若更新老化时间，不仅使得性能下降，还可能导致无效表项一直存在，发生错误。

2. 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？

若不加限制，广播包理论上将会被所有交换机都广播一遍。在此过程中，每个交换机都收到来自广播包源的信息，从而学习到源MAC地址与接收到广播包端口之间的联系。因此，广播包总的来看会使转发表条目增加。

3. 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。

不是很可行。最大的问题是，如果每个主机都对应一个转发表条目，在网络上流量比较大的时候，交换机本地将会存储过多的条目，无论时间上还是空间上开销都过大。还有，转发表的清扫可能过于频繁，因为可能的条目太多了。我们现在所使用的协议对一定区域内的若干主机进行了抽象，增加层次结构，以减少负担。