

# 交换机转发实验

2020年10月17日

蔡润泽

本实验 [Github](#) [地址](#)

## 实验内容

### 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

```
- iface_info_t *lookup_port(u8 mac[ETH_ALEN]);  
- void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface);  
- int sweep_aged_mac_port_entry();  
- void broadcast_packet(iface_info_t *iface, const char *packet, int len);  
- void handle_packet(iface_info_t *iface, char *packet, int len);
```

### 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

## 设计思路

```
iface_info_t *lookup_port(u8 mac[ETH_ALEN]);
```

该函数的作用是：在转发表中查找对应 `mac` 地址和 `iface` 映射的表项。若找到对应的表项，则返回查询 `mac` 地址对应的 `iface`。

由于交换机转发过程中，会存在另一个线程进行超时表项的清理工作，因此查找操作需要加上锁来确保原子性。

该函数具体实现代码如下：

```
iface_info_t *lookup_port(u8 mac[ETH_ALEN])  
{  
    pthread_mutex_lock(&mac_port_map.lock);  
    u8 hash_value = hash8((char *)mac, ETH_ALEN);  
    mac_port_entry_t * mac_port_entry_pos = NULL;  
    list_for_each_entry(mac_port_entry_pos, &mac_port_map.hash_table[hash_value], list) {  
        if (mac_cmp(mac_port_entry_pos->mac, mac, ETH_ALEN) == 0) {  
            mac_port_entry_pos->visited = time(NULL);  
            pthread_mutex_unlock(&mac_port_map.lock);  
            return mac_port_entry_pos->iface;  
        }  
    }  
    pthread_mutex_unlock(&mac_port_map.lock);  
    return NULL;  
}
```

注：其中`mac_cmp`为自编函数，作用是比较两个`mac`地址是否相同。

```
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface);
```

该函数的作用是：当转发表中没有 源mac 地址和对应 iface 的映射表项时，将 源mac 地址与该 iface 插入到转发表当中。

同样的，由于交换机转发过程中，会存在另一个线程进行超时表项的清理工作，因此插入操作同样需要加上锁来确保原子性。

该函数具体实现代码如下：

```
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
{
    pthread_mutex_lock(&mac_port_map.lock);
    mac_port_entry_t * new_mac_port_entry = safe_malloc(sizeof(mac_port_entry_t));
    bzero(new_mac_port_entry, sizeof(mac_port_entry_t));
    mac_cpy(new_mac_port_entry->mac, mac, ETH_ALEN);
    new_mac_port_entry->iface = iface;
    new_mac_port_entry->visited = time(NULL);
    u8 hash_value = hash8((char *)mac, ETH_ALEN);
    list_add_tail(&new_mac_port_entry->list, &mac_port_map.hash_table[hash_value]);
    pthread_mutex_unlock(&mac_port_map.lock);
}
```

注：其中mac\_cpy为自编函数，作用是将源mac地址的内容复制到目的mac地址。

## **int sweep\_aged\_mac\_port\_entry();**

该函数的作用是：当转发表中的表项超过30s没有被查询，则删除冗旧的表项。

同样的，由于多线程，因此清理操作需要加上锁来确保原子性。

该函数具体实现代码如下：

```
int sweep_aged_mac_port_entry()
{
    pthread_mutex_lock(&mac_port_map.lock);
    mac_port_entry_t *entry = NULL;
    mac_port_entry_t *q = NULL;
    time_t now = time(NULL);
    int rm_entry_num = 0;
    for (int i = 0; i < HASH_8BITS; i++) {
        list_for_each_entry_safe(entry, q, &mac_port_map.hash_table[i], list) {
            if ((int)(now - entry->visited) > MAC_PORT_TIMEOUT) {
                list_delete_entry(&entry->list);
                free(entry);
                rm_entry_num ++;
            }
        }
    }
    pthread_mutex_unlock(&mac_port_map.lock);
    return rm_entry_num;
}
```

## **void broadcast\_packet(iface\_info\_t \*iface, const char \*packet, int len);**

该函数的功能为广播收到的包，代码复用上次实验的广播代码即可。

## **void handle\_packet(iface\_info\_t \*iface, char \*packet, int len);**

该函数的功能为处理收到的包。

处理包的逻辑为先调用 lookup\_port 函数，检查目的mac与端口的映射有无在映射表中。若存在，则根据这个表项进行发包，若没有则广播。另外需要检查源mac地址与转发端口的映射是否存在表中，若没有，则调用 insert\_mac\_port 函数插入表中。

具体代码实现如下：

```
void handle_packet(iface_info_t *iface, char *packet, int len)
{
    struct ether_header *eh = (struct ether_header *)packet;
    iface_info_t * iface_entry = NULL;
    if ((iface_entry = lookup_port(eh->ether_dhost)) != NULL) {
        iface_send_packet(iface_entry, packet, len);
    } else {
        broadcast_packet(iface, packet, len);
    }

    if (lookup_port(eh->ether_shost) == NULL) {
        insert_mac_port(eh->ether_shost, iface);
    }
}
```

## 结果验证

利用网络拓扑进行 iperf 测试，本设计的测试结果如下：

The image displays four terminal windows showing the execution of iperf tests on a network topology. The windows are titled "Node: s1", "Node: h2", "Node: h1", and "Node: h3".

- Node: s1**: Shows the execution of `./switch` and `DEBUG: find the following interfaces: s1-eth0 s1-eth1 s1-eth2. DEBUG: 3 aged entries in mac_port table are removed.`
- Node: h2**: Shows the execution of `iperf -s` as a server listening on TCP port 5001. It receives a connection from 10.0.0.2 port 5001 and reports a bandwidth of 9.56 Mbits/sec.
- Node: h1**: Shows the execution of `iperf -c 10.0.0.2 -t 30` as a client connecting to 10.0.0.2 port 5001. It reports a bandwidth of 9.57 Mbits/sec.
- Node: h3**: Shows the execution of `iperf -s` as a server listening on TCP port 5001. It receives a connection from 10.0.0.3 port 5001 and reports a bandwidth of 9.56 Mbits/sec.

上图中 h1 节点同时接收 h2 节点和 h3 节点，可以看出 h2 节点和 h3 节点的发送带宽分别为 9.56Mbps 和 9.56Mbps，利用率为 95.6%。

switch-reference 的结果如下：



上图中 h1 节点同时接收 h2 节点和 h3 节点，可以看出 h2 节点和 h3 节点的发送带宽分别为 4.07Mbps 和 4.36Mbps，平均利用率为 42.15%。

在本实验中，switch 的带宽利用率比 hub 高出 127%。因此 switch 利用转发表的方式明显比 hub 的直接广播模式效率要高。

## 思考题

### 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。

假如所有的主机、交换机刚刚通电，内部没有缓存任何转发表。A若想和B通信，而A此时并不知道B的MAC地址，于是A通过ARP广播来试图获取B的MAC地址。

交换机在收到这个ARP广播包后，首先学习到了A原来是和某个端口（假设为1号口）相连。然后在缓存中查找B的MAC地址，但是没有找到，于是交换机将这个包从所有端口（连接A的1号口除外）发出去，其他交换机收到后也会继续广播出去。当其他主机接收这个广播包之后，不会接受他；而B在收到这个广播包后，发现是找自己的，于是它发出类似的回应内容，来告知A自己的身份。<sup>1</sup>

在这个过程中帮助所有参与的交换机学习建立起了映射转发表。

### 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。足够多的交换机虽然可以链接全世界的终端，但是这样缺少层次化的网络结构难以维护，例如会出现转发表空间迅速膨胀，如果转发表空间有限，而网络出现大量使用时，可能会导致转发表中的映射项不断地被更新替换出来，此时交换机网络和hub网络可能会效率一样的低。

全通过交换机连接会遇到其他问题。例如，我们在计算机网络理论课上学到，当网络连接中遇到环形拓扑时，需要通过生成树算法来破坏环形结构，以免出现在环形拓扑结构不停转发的情况。而假如用足够多个交换机可以连接起全世界所有的终端，生成树算法的消耗会非常大。

另外倘若全部用交换机进行连接而不采取其他协议，还会遇到数据安全的问题，并且可能会形成各样的网络攻击。

因此网络还是需要层次化的结构。

## 参考资料

[1]网络二层与三层数据包转发过程 <https://blog.csdn.net/finderskill/article/details/87860520>