

# 01-switching

---

=====

My name: 王聪颖

My Student ID: 211220180

This lab took me about 3 hours to do.

## Implementation Explanation:

自己实现简单的mac地址哈希函数

```
uint8_t hash_val(u8 mac[ETH_ALEN])
{
    uint8_t val = 0;
    for (int i = 0; i < ETH_ALEN; ++i)
        val ^= mac[i];
    return val;
}
```

查找mac->iface映射，成功查找则更新访问时间，注意使用互斥锁访问临界区资源

```
// lookup the mac address in mac_port table
iface_info_t *lookup_port(u8 mac[ETH_ALEN])
{
    uint8_t index = hash_val(mac);
    mac_port_entry_t *entry = NULL;
    pthread_mutex_lock(&mac_port_map.lock);
    list_for_each_entry(entry, &mac_port_map.hash_table[index], list)
    {
        int b = memcmp(mac, entry->mac, ETH_ALEN);
        if (b == 0)
        {
            entry->visited = time(NULL);
            pthread_mutex_unlock(&mac_port_map.lock);
            return entry->iface;
        }
    }
    pthread_mutex_unlock(&mac_port_map.lock);
    return NULL;
}
```

插入mac->iface映射，使用前面实现的lookup\_port，仅当查找失败时，插入新映射，注意互斥锁

```
// insert the mac -> iface mapping into mac_port table
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
{
    iface_info_t *forward_iface = lookup_port(mac);
    // already existed
    if (forward_iface != NULL)
        return;
    // not exist, insert
    pthread_mutex_lock(&mac_port_map.lock);
    mac_port_entry_t *new_entry = malloc(sizeof(mac_port_entry_t));
    new_entry->iface = iface;
    memcpy(new_entry->mac, mac, ETH_ALEN);
    new_entry->visited = time(NULL);
    list_add_tail(&new_entry->list,
&mac_port_map.hash_table[hash_val(mac)]);
    pthread_mutex_unlock(&mac_port_map.lock);
}
```

定期删除过期表项，注意互斥锁

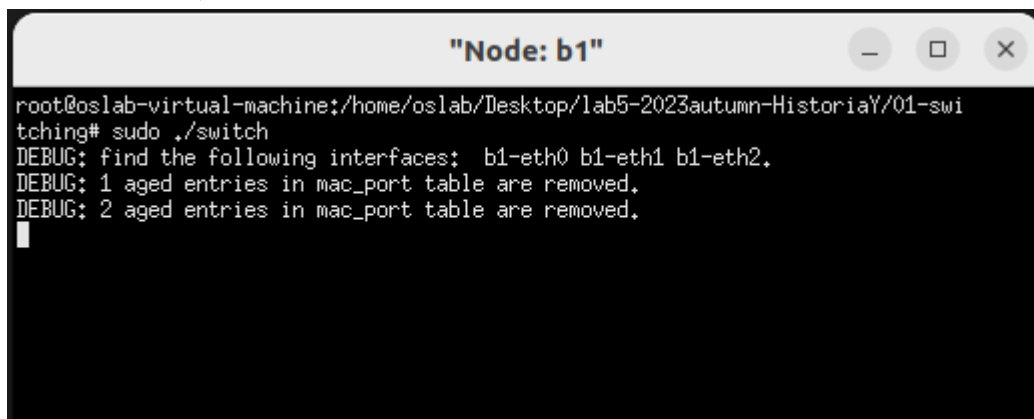
```
// sweeping mac_port table, remove the entry which has not been visited in
the
// last 30 seconds.
int sweep_aged_mac_port_entry()
{
    int count = 0;
    time_t now = time(NULL);
    pthread_mutex_lock(&mac_port_map.lock);
    mac_port_entry_t *entry, *q;
    for (int i = 0; i < HASH_8BITS; i++)
    {
        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);
                ++count;
            }
        }
    }
    pthread_mutex_unlock(&mac_port_map.lock);
    return count;
}
```

收到packet时处理逻辑，查找成功->转发，查找失败->广播，最后插入源mac->iface的映射关系

```
// handle packet
// 1. if the dest mac address is found in mac_port table, forward it;
otherwise,
// broadcast it.
// 2. put the src mac -> iface mapping into mac hash table.
void handle_packet(iface_info_t *iface, char *packet, int len)
{
    struct ether_header *eh = (struct ether_header *)packet;
    iface_info_t *forward_iface = lookup_port(eh->ether_dhost);
    if (forward_iface != NULL)
        iface_send_packet(forward_iface, packet, len);
    else
        broadcast_packet(iface, packet, len);
    insert_mac_port(eh->ether_shost, iface);
}
```

## Screenshots:

使用xterm命令，在b1上执行switch



```
root@oslab-virtual-machine:/home/oslab/Desktop/lab5-2023autumn-HistoriaY/01-switching# sudo ./switch
DEBUG: find the following interfaces: b1-eth0 b1-eth1 b1-eth2.
DEBUG: 1 aged entries in mac_port table are removed.
DEBUG: 2 aged entries in mac_port table are removed.
```

## 使用ping命令，测试每对Host之间连通性

```
oslab@oslab-virtual-machine: ~/Desktop/lab5-2023autumn-HistoriaY/01-switching$ sudo ./three_nodes_bw.py
mininet> xterm b1
mininet> h1 ping h2 -c 2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.203 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.152 ms

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.152/0.177/0.203/0.025 ms
mininet> h1 ping h3 -c 2
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.204 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.275 ms

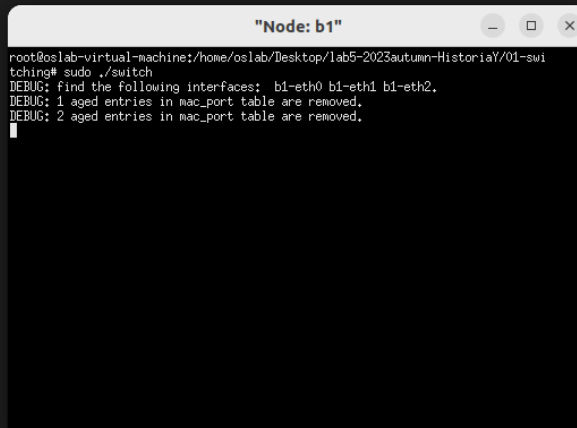
--- 10.0.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.204/0.239/0.275/0.035 ms
mininet> h2 ping h1 -c 2
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.138 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.154 ms

--- 10.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.138/0.146/0.154/0.008 ms
mininet> h2 ping h3 -c 2
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.188 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.196 ms

--- 10.0.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.188/0.192/0.196/0.004 ms
mininet> h3 ping h1 -c 2
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.162 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.160 ms

--- 10.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1028ms
rtt min/avg/max/mdev = 0.160/0.161/0.162/0.001 ms
mininet> h3 ping h2 -c 2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.138 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.129 ms

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1027ms
rtt min/avg/max/mdev = 0.129/0.133/0.138/0.004 ms
mininet>
```



## Remaining Bugs:

Host间全部连通，无遗留bug