Lab5 实验报告

一. 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。

二. 实验设计

1. 转发表结构

我们的转发表结构如下图所示：

目的地址	转发端口	老化时间
Host 1 MAC Addr	Port 1	30 sec
Host 2 MAC Addr	Port 2	30 sec
Host 3 MAC Addr	Port 3	30 sec

图 1：MAC 转发表的图形化结构

在程序里这样表示：

```
struct mac_port_entry {
    struct list_head list;
    uint8_t mac[ETH_ALEN];
    iface_info_t *iface;
    time_t visited;
};
```

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：

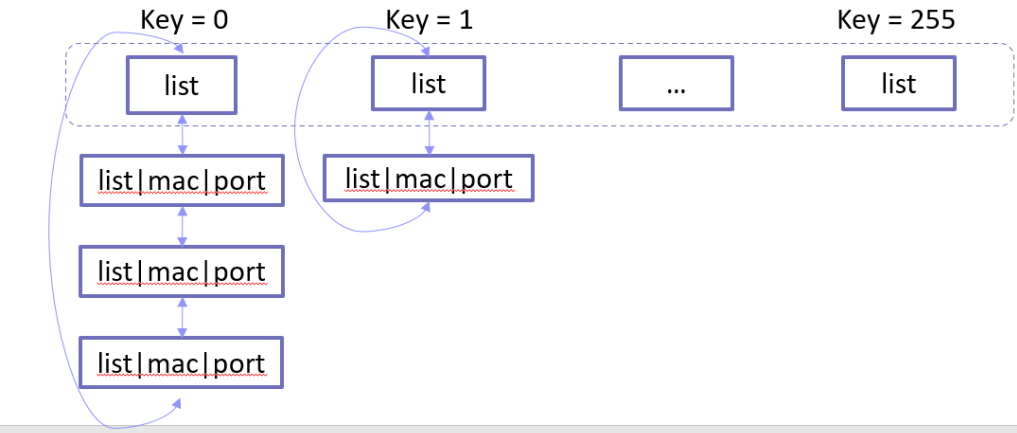

图 3：利用 hash 表存放表项

2. 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：

```c
// lookup the mac address in mac_port table. If not found, return null.
iface_info_t *lookup_port(u8 mac[ETH_ALEN])
{
    u8 hash_mac = hash8((char*)mac, ETH_ALEN);

    pthread_mutex_lock(&mac_port_map.lock);
    mac_port_entry_t *entry;
    list_for_each_entry(entry, &mac_port_map.hash_table[hash_mac], list) {
        if (check_u8_equal(entry -> mac, mac, ETH_ALEN)) {
            entry -> visited = time(NULL);
            pthread_mutex_unlock(&mac_port_map.lock);
            return entry -> iface;
        }
    }

    pthread_mutex_unlock(&mac_port_map.lock);
    return NULL;
}
```

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。

3. 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：

```c
// insert the mac -> iface mapping into mac_port table
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 = (mac_port_entry_t*)safe_malloc(sizeof(mac_port_entry_t));
    bzero(new_mac_port_entry, sizeof(mac_port_entry_t));

    u8_cpy(new_mac_port_entry -> mac, mac, ETH_ALEN);
    new_mac_port_entry -> iface = iface;
    new_mac_port_entry -> visited = time(NULL);

    u8 hash_mac = hash8((char*)mac, ETH_ALEN);
    list_add_tail(&new_mac_port_entry->list, &mac_port_map.hash_table[hash_mac]);

    pthread_mutex_unlock(&mac_port_map.lock);
}
```

图 4：表项插入操作

4. 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：

```c
// sweeping mac_port table, remove the entry which has not been visited in the
// last 30 seconds.
int sweep_aged_mac_port_entry()
{
    pthread_mutex_lock(&mac_port_map.lock);
    mac_port_entry_t *mac_entry, *q;
    time_t now = time(NULL);
    int count = 0;

    for (int i = 0; i < HASH_8BITS; i++) {
        list_for_each_entry_safe(mac_entry, q, &mac_port_map.hash_table[i], list) {
            if ((int)(now - mac_entry -> visited) > MAC_PORT_TIMEOUT) {
                list_delete_entry(&mac_entry -> list);
                free(mac_entry);
                count ++;
            }
        }
    }
    pthread_mutex_unlock(&mac_port_map.lock);

    return count;
}
```

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：

```c
// sweeping mac_port table periodically, by calling sweep_aged_mac_port_entry
void *sweeping_mac_port_thread(void *nil)
{
    while (1) {
        sleep(1);
        int n = sweep_aged_mac_port_entry();

        if (n > 0)
            log(DEBUG, "%d aged entries in mac_port table are removed.", n);
    }

    return NULL;
}
```

图 6：间歇性调用扫除函数

5. 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：

```c
void handle_packet(iface_info_t *iface, char *packet, int len)
{

    struct ether_header *eh = (struct ether_header *)packet;
    // log(DEBUG, "the dst mac address is " ETHER_STRING ".\n", ETHER_FMT(eh->ether_dhost));
    iface_info_t *iface_entry;
    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);
    }

    free(packet);
}
```

图 7：总的处理逻辑

6. 性能比较：

当 h2 做 server，h1，h3 做 client 时候：

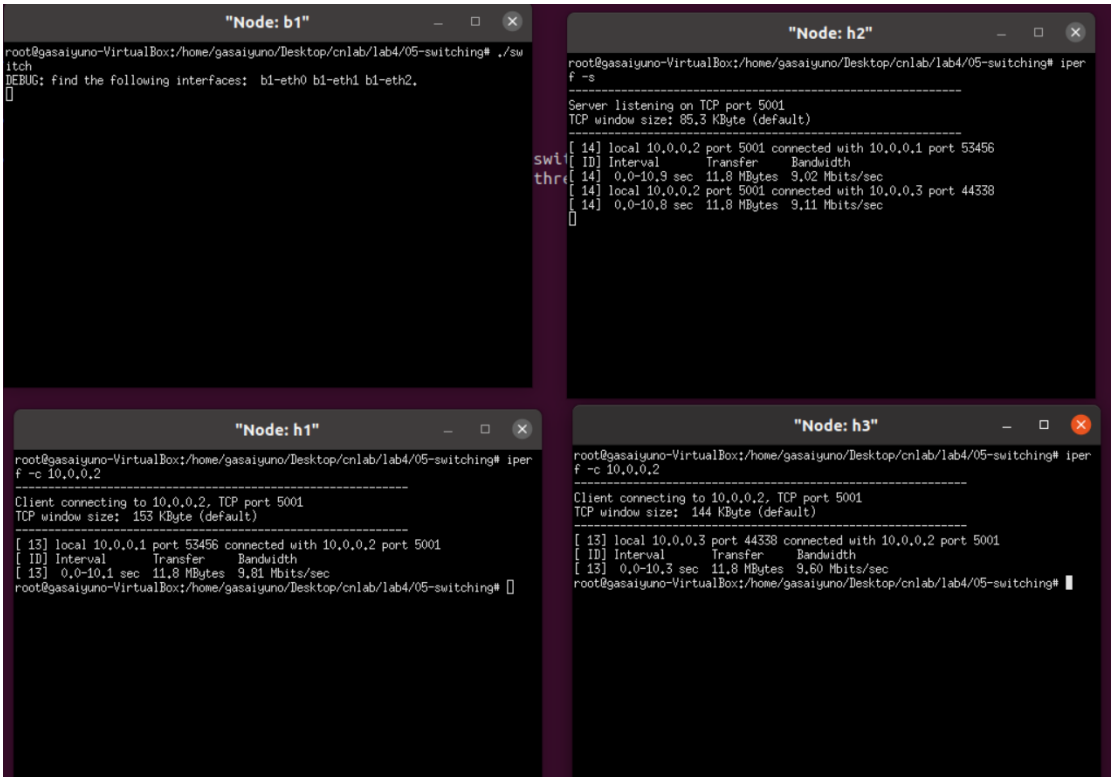

图 8：交换网络性能测试

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：

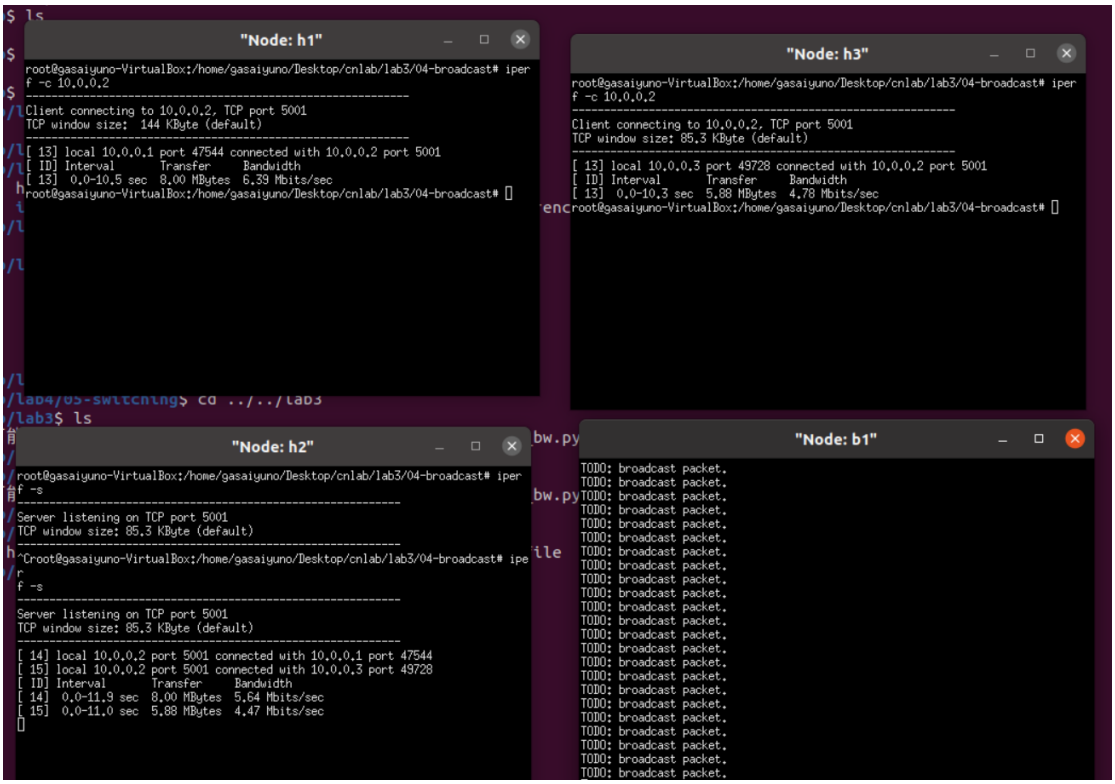

图 9：广播网络性能测试

我们的效率一个为 63.9%，一个为 47.8%，明显低于交换网络。

三. 实验思考

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？**

   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。

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？**

   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。

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。**

   不行，这样转发表会膨胀的很厉害，网络的效率会很低。