交换机转发实验报告

陈彦帆 2018K8009918002

1、实验内容

（1）实现对数据结构 mac_port_map 的所有操作，包括查询，插入和老化，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。

（3）完成思考题。

2、实验流程

（1）实现下列函数。

```
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）执行脚本:

```
make
sudo python three_nodes_bw.py
mininet> xterm h1 h1 h2 h3 s1
s1# ./switch
h2# iperf -s
h3# iperf -s
h1# iperf -c 10.0.0.2 -t 30 & iperf-c 10.0.0.3 -t 30
mininet> quit
```

3、功能实现

（1）转发表数据结构

采用拉链式哈希表。数据结构的实现已经给出。哈希函数默认采用的是 mac 地址各字节的异或值，这并不是一个高效的哈希函数，在表项较多时冲突较大。改进可考虑采用 djb2, one-at-a-time 或 Murmur3 哈希函数。

（2）插入

首先计算哈希值确定哈希表下标，遍历哈希表项对应的链表，若找到相同 mac 地址的项，则更新它的 iface 和 visit 时间；否则，在链表的最后插入一个新的项。

```
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
{
    u8 idx = hash8(mac,ETH_ALEN);
    time_t now = time(NULL);
    mac_port_entry_t *entry = NULL;
    pthread_mutex_lock(&mac_port_map.lock);
    list_for_each_entry(entry,&mac_port_map.hash_table[idx],list) {
        if(memcmp(entry->mac,mac,ETH_ALEN)==0){
            entry->iface = iface;
            entry->visited = now;
            goto DONE;
        }
    }
    entry = malloc(sizeof(mac_port_map_t));
    entry->iface = iface;
    memcpy(entry->mac,mac,ETH_ALEN);
    entry->visited = now;
    list_add_tail(&entry->list,&mac_port_map.hash_table[idx]);
    fprintf(stdout, "Inserted at map[%d]. " ETHER_STRING " -> %s\n",idx, ETHER_FMT(
entry->mac), \
            entry->iface->name);
DONE:
    pthread_mutex_unlock(&mac_port_map.lock);
}
```

（3）查找

与插入函数类似。区别在于查找失败直接返回 NULL。

（4）老化

创建一个子线程，每秒扫描整个哈希表，若发现有表项的存在时间超过 30s，将其删除。

该线程与主线程存在竞争，需要用锁结构保护。本实验给整个哈希表加上一把互斥锁，在表项较多时，老化线程可能会影响查找的效率。为了改进，可考虑采取类似 RCU 锁的结构。

```
int sweep_aged_mac_port_entry()
{
    int count = 0;
    mac_port_entry_t *entry = NULL, *q;
    time_t now = time(NULL);

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

4、结果与讨论

（1）广播网络效率测试(iperf)

结点拓扑结构如图 1。Host2(h2)和 Host3(h3)为 Server，Host1(h1)为 Client，同时向 h2 和 h3 发送 iperf 测试请求，测试结果如图 2。switch 的打印结果如图 3。

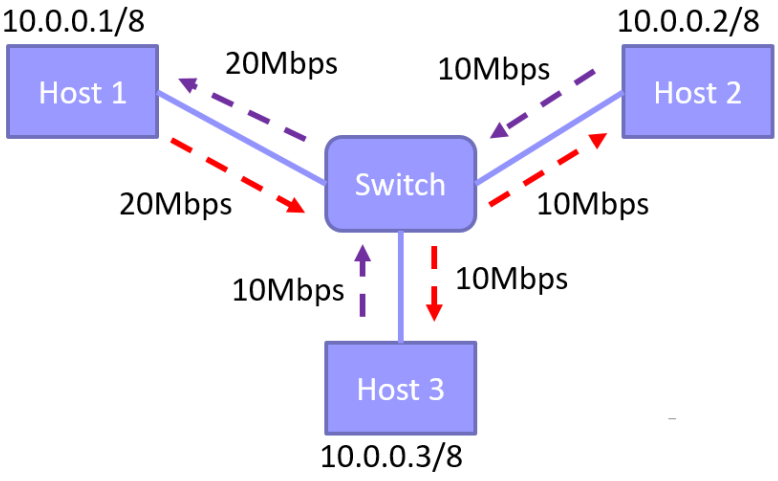

图 1

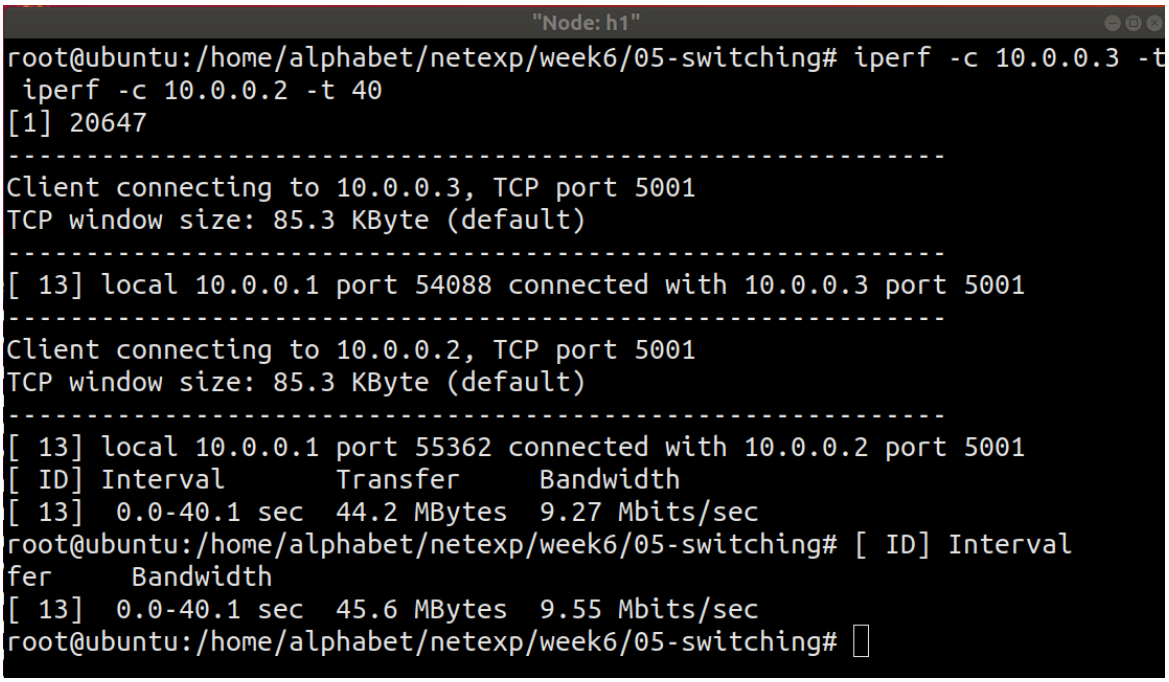

图 2

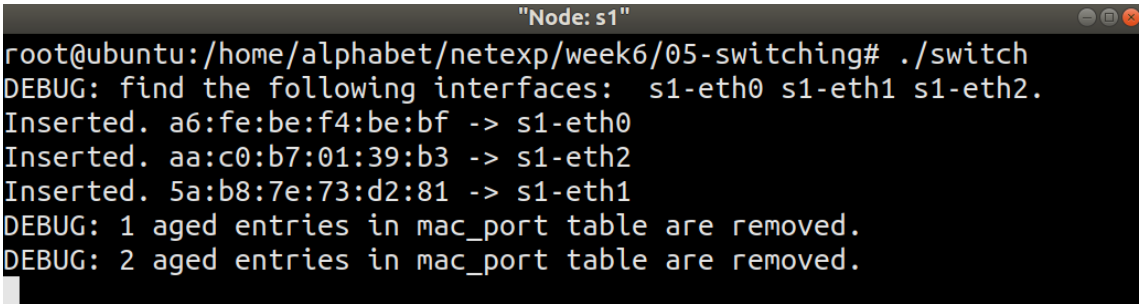

图 3

从图 2 中得到，h1->h2 的实测带宽为 9.27Mbps，h1->h3 的实测带宽为 9.55Mbps。带宽利用率接近 100%。上个实验中，这个结果为 2.06Mbps 和 7.10Mbps。h1->h2 与 h1->h3 的带宽之和提升了约 105%。这是因为在学习完毕后，switch 只向目的端口转发数据包而不再向所有端口转发数据包，避免了网络拓扑中的冗余包占用带宽（详见网络广播实验报告）。

图 3 显示了 switch 的学习结果和老化线程运行结果。学习和老化运行正常。

5、思考题

（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？提示：1、查询目的 MAC 地址时是否有必要更新；2、如果更新的话，当一个主机从交换机的一个网口切换到了另一个网口，会有什么问题？

目的地址不需要更新，老化操作可以把过期的目的地址删除。如果更新目的地址，当目的MAC 地址变更时，原来地址的老化时间可能一直被更新，从而无法被老化删除。

（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？

全 0xFF 地址只会作为目的地址出现而不会作为源地址出现，将会被广播，不会更新交换机转发表。

（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。

不可行。为了避免数据包环路，交换机网络拓扑结构为生成树，两个节点之间只能有一条路径。当节点数量足够多时，网络中的一台交换机需要处理大量结点的请求，转发表需要为所有地址进行学习，交换机的计算和存储能力有限，达不到要求。而且，生成树算法难以收敛。

另外，用交换机连接全世界的终端会带来安全性问题。