

交换机转发实验报告

姓名：钟赞

学号：2016K8009915009

实验内容

- 实现一个具有转发表学习功能的交换机，使得连接到同一交换机的主机可以相互通信。
- 实现对数据结构`mac_port_map`的所有操作(包括查询操作`lookup_port`、插入操作`insert_mac_port`和老化操作`sweep_aged_mac_port_entry`)，以及数据包的转发`handle_packet`和广播`broadcast_packet`操作。
- 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。

实验步骤

1. 补全`mac.c`中对`mac_port_map`的三种操作

• 查找 `lookup_port`

- 利用`hash8`函数可得出`mac`的`hash`值，可在`hash`链表中进行快速查找，查找成功后需要更新访问时间`visited`。代码如下：

```
iface_info_t *lookup_port(u8 mac[ETH_ALEN])
{
    // TODO: implement the lookup process here
    fprintf(stdout, "implement the lookup process here.\n");

    mac_port_entry_t * entry = NULL;
    iface_info_t * result = NULL;

    pthread_mutex_lock(&mac_port_map.lock);
    list_for_each_entry(entry, &mac_port_map.hash_table[hash8((char *)mac, sizeof(u8) * ETH_ALEN)], list)
    {
        if (!memcmp(entry->mac, mac, ETH_ALEN * sizeof(u8)))
        {
            // if found, get its iface info and the time
            entry->visited = time(NULL);
            result = entry->iface;
            break;
        }
    }
    pthread_mutex_unlock(&mac_port_map.lock);

    return result;
}
```

• 插入 `insert_mac_port`

- 先分配一个新的`entry`结点，初始化后用`list_add_tail`函数将其插入到链表中。代码如下：

```
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
{
    // TODO: implement the insertion process here
    fprintf(stdout, "implement the insertion process here.\n");

    // initialize a new entry
    mac_port_entry_t * entry = (mac_port_entry_t *)malloc(sizeof(mac_port_entry_t));
    entry->iface = iface;
```

```

entry->visited = time(NULL);
memcpy(entry->mac, mac, sizeof(u8) * ETH_ALEN);
u8 hash_val = hash8((char *)mac, sizeof(u8) * ETH_ALEN);

pthread_mutex_lock(&mac_port_map.lock);
// add entry into list
list_add_tail(&entry->list, &mac_port_map.hash_table[hash_val]);
pthread_mutex_unlock(&mac_port_map.lock);
}

```

• 老化 `sweep_aged_mac_port_entry`

- 遍历hash表，用`list_delete_entry`函数删除老化的结点。代码如下：

```

int sweep_aged_mac_port_entry()
{
    // TODO: implement the sweeping process here
    fprintf(stdout, "implement the sweeping process here.\n");

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

    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)
            {
                // delete the entry
                list_delete_entry(&entry->list);
                free(entry);
                ++ count;
            }
        }
    }
    pthread_mutex_unlock(&mac_port_map.lock);

    return count;
}

```

2. 实现交换机处理数据包的函数`handle_packet`

代码如下：

```

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));

    // TODO: implement the packet forwarding process here
    fprintf(stdout, "TODO: implement the packet forwarding process here.\n");
    iface_info_t * result = lookup_port(eh->ether_dhost);

    if (result)
    {
        iface_send_packet(result, packet, len);
    }
    else
    {
        broadcast_packet(iface, packet, len);
    }

    if (!lookup_port(eh->ether_shost))
    {
        insert_mac_port(eh->ether_shost, iface);
    }
}

```

3. 实现数据包的广播函数

该函数在广播实验中已实现，代码如下：

```
void broadcast_packet(iface_info_t *iface, char *packet, int len)
{
    // TODO: implement the broadcast process here
    fprintf(stdout, "TODO: implement the broadcast process here.\n");

    iface_info_t *ifaces = NULL;
    list_for_each_entry(ifaces, &instance->iface_list, list)
    {
        if(ifaces != iface)
        {
            iface_send_packet(ifaces, packet, len);
        }
    }
}
```

实验结果及分析

- 各个节点相互ping通

结果如下：

The screenshot displays four terminal windows, each representing a different node in a network experiment. The windows are titled "Node: s1", "Node: h1", "Node: h2", and "Node: h3".

- Node: s1**: Shows a series of "TODO: implement the packet forwarding process here." messages and "Send raw packet failed: Socket operation on non-socket" errors. It also displays "DEBUG: the dst mac address is 2e:86:73:9f:e5:92." and "DEBUG: the dst mac address is 42:1f:40:af:42:2a." messages.
- Node: h1**: Shows successful ping results for 10.0.0.2 and 10.0.0.3. For 10.0.0.2, it shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3071ms. For 10.0.0.3, it shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3067ms.
- Node: h2**: Shows successful ping results for 10.0.0.1 and 10.0.0.3. For 10.0.0.1, it shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3079ms. For 10.0.0.3, it shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3048ms.
- Node: h3**: Shows successful ping results for 10.0.0.1 and 10.0.0.2. For 10.0.0.1, it shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3048ms. For 10.0.0.2, it shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3068ms.

- H1: iperf client; H2, H3: iperf servers

下图为交换机实验的结果图：

```

"Node: s1"
TODO: implement the packet forwarding process here.
implement the lookup process here.
implement the lookup process here.
DEBUG: the dst mac address is 62:63:29:b3:43:39.

TODO: implement the packet forwarding process here.
implement the lookup process here.
implement the lookup process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
DEBUG: the dst mac address is 92:48:97:49:a0:6a.

TODO: implement the packet forwarding process here.

"Node: h1"
Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.1 port 58390 connected with 10.0.0.3 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 13] 0.0-30.1 sec  32.6 MBytes  9.08 Mbits/sec
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switchin
[ ID] Interval      Transfer      Bandwidth
[ 13] 0.0-30.2 sec  31.4 MBytes  8.72 Mbits/sec
^C
[1]+  Done                  iperf -c 10.0.0.2 -t 30
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switchin
iperf -s

Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

[ 14] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 49206
[ 15] local 10.0.0.1 port 5001 connected with 10.0.0.3 port 38114
[ ID] Interval      Transfer      Bandwidth
[ 14] 0.0-31.0 sec  33.0 MBytes  8.93 Mbits/sec
[ 15] 0.0-30.7 sec  33.1 MBytes  9.04 Mbits/sec

"Node: h2"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switch
iperf -s

Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

[ 14] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 52666
[ ID] Interval      Transfer      Bandwidth
[ 14] 0.0-30.2 sec  31.4 MBytes  8.70 Mbits/sec
^Croot@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-swit

iperf -c 10.0.0.1 -t 30

Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)

"Node: h3"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switch
iperf -s

Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

[ 14] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 58390
[ ID] Interval      Transfer      Bandwidth
[ 14] 0.0-30.1 sec  32.6 MBytes  9.08 Mbits/sec
^Croot@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-swit

iperf -c 10.0.0.1 -t 30

Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.3 port 38114 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 13] 0.0-30.1 sec  33.1 MBytes  9.25 Mbits/sec
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switch

```

由图可知，两者带宽均接近10Mb/s。因为H2和H3共同占用了H1到交换机的20Mb/s带宽，和它们各自到S1的10Mb/s带宽，发挥了最大效能。

此结果与广播实验一样。

• H1: iperf client; H2, H3: iperf servers

下图为交换机实验的结果图：

```

"Node: s1"
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.
implement the sweeping process here.

"Node: h1"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switchin
iperf -c 10.0.0.2 -t 30 & iperf -c 10.0.0.3 -t 30
[1] 15018

Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.1 port 52666 connected with 10.0.0.2 port 5001

Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.1 port 58390 connected with 10.0.0.3 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 13] 0.0-30.1 sec  32.6 MBytes  9.08 Mbits/sec
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switchin
[ ID] Interval      Transfer      Bandwidth
[ 13] 0.0-30.2 sec  31.4 MBytes  8.72 Mbits/sec

"Node: h2"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switchi
iperf -s

Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

[ 14] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 52666
[ ID] Interval      Transfer      Bandwidth
[ 14] 0.0-30.2 sec  31.4 MBytes  8.70 Mbits/sec

"Node: h3"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switchi
iperf -s

Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

[ 14] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 58390
[ ID] Interval      Transfer      Bandwidth
[ 14] 0.0-30.1 sec  32.6 MBytes  9.08 Mbits/sec

```

下图为广播实验的结果图：

```

"Node: h1"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/04-broadcast/04-broadcast-code#
iperf -c 10.0.0.2 -t 30

Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.1 port 47304 connected with 10.0.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.1 sec  33.9 MBytes  9.43 Mbits/sec
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/04-broadcast/04-broadcast-code#
iperf -c 10.0.0.3 -t 30

Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.1 port 36362 connected with 10.0.0.3 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.2 sec  33.8 MBytes  9.39 Mbits/sec
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/04-broadcast/04-broadcast-code#
iperf -c 10.0.0.3 -t 30 & iperf -c 10.0.0.2 -t 30
[1] 12652

Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.1 port 47308 connected with 10.0.0.2 port 5001

Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 13] local 10.0.0.1 port 36366 connected with 10.0.0.3 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.2 sec  25.2 MBytes  7.01 Mbits/sec
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.6 sec  9.12 MBytes  2.50 Mbits/sec
[1]+  Done                  iperf -c 10.0.0.3 -t 30
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/04-broadcast/04-broadcast-code#

"Node: b1"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/04-broadcast/04-broadcast-code#
perf -s

server listening on TCP port 5001
CP window size: 85.3 KByte (default)

14] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
14] 0.0-30.4 sec  33.9 MBytes  9.36 Mbits/sec
15] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
15] 0.0-30.7 sec  9.12 MBytes  2.49 Mbits/sec

"Node: h3"
root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/04-broadcast/04-broadcast-code#
perf -s

server listening on TCP port 5001
CP window size: 85.3 KByte (default)

14] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
14] 0.0-30.4 sec  33.8 MBytes  9.33 Mbits/sec
15] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
15] 0.0-30.4 sec  25.2 MBytes  6.97 Mbits/sec

```

对比得知，当H1同时向H2/H3发送数据包时，广播实验中两者带宽之和约为10MB/s，交换机实验中两者均为10MB/s左右，最大化利用了链路带宽。

这是因为在交换机实验中，H2/H3的mac地址均存储在转发表中，发送给H2的数据包到达交换机时，会直接发送到对应的目的地址，不需要向H3广播，因此不占用H3接收数据包链路S1->H3的带宽，链路S1->H2同理，所以这两个链路可以发挥最大带宽10MB/s的性能。相比广播，交换机提高了性能。

错误记录

实验中进行iperf测试时，在H2/H3中输入iperf -s，会报如下图错误：

```

root@zy-VB:/mnt/shared/Git/ComputerNetwork-Lab/05-switching/05-switching-code#
iperf -s
bind failed: Address already in use

```

查阅相关资料，并没有找到明确的解决方案，尝试了修改H1/H2/H3的IP地址、重新编译、kill进程等办法，最后通过注释掉three_nodes_bw.py中的部分代码：

```

s1.cmd('./switch-reference &')
h2.cmd('iperf -s &')
h3.cmd('iperf -s &')

```

xterm使用exit命令退出(之前使用逐个关掉s1, h1/2/3，然后quit命令的方式退出)，并且修改了一些bug后，成功进行iperf测试。但是仍不太清楚造成这个错误的具体原因是什么。

参考资料

1. [Address already in use .端口占用的错误](#)
2. [Linux下Socket编程的端口问题\(Bind error: Address already in use\)](#)