

## 一、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

## 二、实验流程

- 1、实现对数据结构 `mac_port_map` 的所有操作

- (1) 修改 `iface_info_t *lookup_port(u8 mac[ETH_ALEN])` (`mac.c`):

每收到一个数据包，根据目的 MAC 地址查询相应转发条目，如果查询到对应条目，则根据相应转发端口转发数据包；否则，广播该数据包。由于需要访问的转发表 `mac_port_map` 属于临界资源，即可能存在另一个线程访问转发表进行其他操作，因此需要加上锁来确保操作的原子性。

```
iface_info_t *lookup_port(u8 mac[ETH_ALEN])  
{  
    // TODO: implement the lookup process here  
    fprintf(stdout, "TODO: implement the lookup process here.\n");  
    iface_info_t *iface = NULL;  
    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) {  
            int cmp = memcmp((void*)entry->mac, (void*)mac, sizeof(u8) * ETH_ALEN);  
            if(cmp==0) return entry->iface;  
        }  
    }  
    return iface;  
}
```

- (2) 修改 `void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)` (`mac.c`):

每收到一个数据包，如果其源 MAC 地址-入端口映射关系在转发表中，更新访问时间；否则，将该地址与入端口的映射关系写入转发表。该操作同样需要保证原子性。

```
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)  
{  
    // TODO: implement the insertion process here
```

```

fprintf(stdout, "TODO: implement the insertion process here.\n");
mac_port_entry_t *entry = malloc(sizeof(mac_port_entry_t));
bzero(entry, sizeof(mac_port_entry_t));
time_t now = time(NULL);
entry->visited = now;
memcpy(entry->mac, mac, sizeof(u8) * ETH_ALEN);
entry->iface = iface;
list_add_tail(&entry->list, &mac_port_map.hash_table[0]);
}

```

(3) 修改 `int sweep_aged_mac_port_entry()` (`mac.c`):

每秒钟运行一次老化操作，删除超过 30 秒未访问的转发条目。该操作同样需要保证原子性。

```

int sweep_aged_mac_port_entry()
{
    // TODO: implement the sweeping process here
    fprintf(stdout, "TODO: implement the sweeping process here.\n");
    int n=0;
    mac_port_entry_t *entry, *q;
    time_t now = time(NULL);
    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){
                n = entry->iface->index;
                list_delete_entry(&entry->list);
                free(entry);
                return n;
            }
        }
    }
    return n;
}

```

(4) 修改 `void handle_packet(iface_info_t *iface, char *packet, int len)` (`main.c`):

收到数据包后，交换机根据转发表中对应的转发端口转出数据包，若没有在转发表中查询到对应端口，则直接广播该数据包。

```

void handle_packet(iface_info_t *iface, char *packet, int len)
{
    // TODO: implement the packet forwarding process here
    fprintf(stdout, "TODO: implement the packet forwarding process here.\n");
    // 得到头部信息
    struct ether_header *eh = (struct ether_header *)packet;

```

```

// fdb 中寻找目的地址mac
iface_info_t *tx_iface = lookup_port(eh->ether_dhost);
if (tx_iface) {
    iface_send_packet(tx_iface, packet, len);
}
else {
    broadcast_packet(iface, packet, len);
}
// 存入源mac+port
if (!lookup_port(eh->ether_shost)) {
    insert_mac_port(eh->ether_shost, iface);
}
}

```

(5) 修改 void broadcast\_packet(iface\_info\_t \*iface, const char \*packet, int len) (broadcast.c):

广播功能沿用实验 1-1 实现的 void broadcast\_packet(iface\_info\_t \*iface, const char \*packet, int len) 函数。

```

void broadcast_packet(iface_info_t *iface, const char *packet, int len)
{
    // TODO: broadcast packet
    fprintf(stdout, "TODO: broadcast packet.\n");
    iface_info_t *iface_entry = NULL;
    list_for_each_entry(iface_entry, &instance->iface_list, list) {
        if (iface_entry->fd != iface->fd) {
            iface_send_packet(iface_entry, packet, len);
        }
    }
}

```

(6) 执行命令 make, 生成可执行程序 switch:

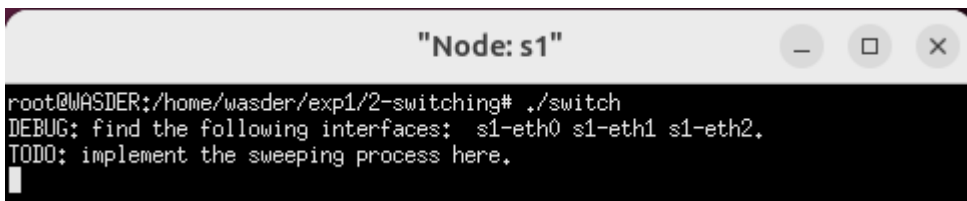
```
wasder@WASDER:~/exp1/2-switching$ make
```

(7) 运行拓扑文件 three\_nodes\_bw.py, 启动 Mininet 网络:

```
wasder@WASDER:~/exp1/2-switching$ sudo python3 three_nodes_bw.py
```

(8) 打开交换机 s1 的终端窗口, 启动可执行程序 switch:

```
mininet> xterm s1
```



## 2、实现数据包的转发和广播操作

- (1) 启动 wireshark 分别监听主机 h2 和 h3
 

```
mininet> h2 wireshark &
mininet> h3 wireshark &
```
- (2) 用主机 h1 分别 ping 主机 h2 和 h3
- (3) 打开主机 h2 和 h3 的终端窗口，启动 wireshark，等待捕获主机 h1 发来的数据包：
 

```
mininet> xterm h2 h3
h2# wireshark
h3# wireshark
```
- (4) 打开主机 h1 的终端窗口，向主机 h2 和 h3 分别发送一个数据包：
 

```
mininet> xterm h1
h1# ping -c 1 10.0.0.2
h1# ping -c 1 10.0.0.3
```

正在捕获 h2-eth0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x2
2	0.000053932	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2
3	5.221230989	6e:70:0e:39:48:f2	62:18:14:de:69:41	ARP	42	Who has 10.0.0.1? Tell 10.0
4	5.221632273	62:18:14:de:69:41	6e:70:0e:39:48:f2	ARP	42	Who has 10.0.0.2? Tell 10.0
5	5.221645398	6e:70:0e:39:48:f2	62:18:14:de:69:41	ARP	42	10.0.0.2 is at 6e:70:0e:39:4
6	5.221809399	62:18:14:de:69:41	6e:70:0e:39:48:f2	ARP	42	10.0.0.1 is at 62:18:14:de:6

正在捕获 h3-eth0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=0x2
2	0.000020258	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2
3	5.365419715	2a:c7:6f:db:66:58	62:18:14:de:69:41	ARP	42	Who has 10.0.0.1? Tell 10.0
4	5.365794153	62:18:14:de:69:41	2a:c7:6f:db:66:58	ARP	42	Who has 10.0.0.3? Tell 10.0
5	5.365814291	2a:c7:6f:db:66:58	62:18:14:de:69:41	ARP	42	10.0.0.3 is at 2a:c7:6f:db:6
6	5.365885346	62:18:14:de:69:41	2a:c7:6f:db:66:58	ARP	42	10.0.0.1 is at 62:18:14:de:6

在主机 h2 的 wireshark 中只捕获到了主机 h2 节点和主机 h1 节点发送的数据，在主机 h3 的 wireshark 中只捕获到了主机 h3 节点和主机 h1 节点发送的数据，表明交换机转发成功。

### 3、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

- (1) H1: iperf client; H2, H3: iperf servers (h1 同时向 h2 和 h3 测量)

```
h2# iperf -s
h3# iperf -s
```

```
"Node: h1"
root@WASDER:/home/wasder/exp1/2-switching# iperf -c 10.0.0.2 -t 30 & iperf -c 10.0.0.3 -t 30
[1] 4506
-----
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 1] local 10.0.0.1 port 42918 connected with 10.0.0.2 port 5001 (icwnd/mss/irt
t=14/1448/754)
[ 1] local 10.0.0.1 port 37234 connected with 10.0.0.3 port 5001 (icwnd/mss/irt
t=14/1448/217)
[ ID] Interval      Transfer      Bandwidth
[ 1] 0.0000-31.3203 sec 35.6 MBytes 9.54 Mbits/sec
[ ID] Interval      Transfer      Bandwidth
[ 1] 0.0000-31.3252 sec 34.5 MBytes 9.24 Mbits/sec
[1]+  已完成      iperf -c 10.0.0.2 -t 30
root@WASDER:/home/wasder/exp1/2-switching#
```

h1 节点向 h2 节点和 h3 节点的发送带宽分别为 **9.54Mbps** 和 **9.24Mbps**，而在拓扑文件中，h1->b1 的带宽为 20Mbps，b1->h2 的带宽为 10Mbps，b1->h3 的带宽为 10Mbps，因此 h1 同时向 h2 和 h3 测量得到的交换机转发网络效率约为 **93.9%**，而由实验 1-1 测得的集线器广播网络的效率约为 **47.5%**，提升了近一倍（97.7%），符合理论实际。因此，**交换机转发网络比集线器广播网络性能更优**。

- (2) H1: iperf servers; H2, H3: iperf client (h2 和 h3 同时向 h1 测量)
- h1# iperf -s

```
"Node: h2"
root@WASDER:/home/wasder/exp1/2-switching# 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)
-----
[ 1] local 10.0.0.2 port 54948 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/255)
[ ID] Interval      Transfer      Bandwidth
[ 1] 0.0000-31.7878 sec 36.3 MBytes 9.57 Mbits/sec
root@WASDER:/home/wasder/exp1/2-switching#
```

```
"Node: h3"
root@WASDER:/home/wasder/exp1/2-switching# 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)
-----
[ 1] local 10.0.0.3 port 39832 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/2368)
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-31.3221 sec 35.6 MBytes 9.54 Mbits/sec
root@WASDER:/home/wasder/exp1/2-switching# a
```

h1 节点对 h2 节点和 h3 节点的接收带宽分别为 9.57Mbps 和 9.54Mbps，而在拓扑文件中，h1->b1 的带宽为 20Mbps，b1->h2 的带宽为 10Mbps，b1->h3 的带宽为 10Mbps，因此 h2 和 h3 同时向 h1 测量得到的交换机转发网络效率约为 95.55%，而由实验 1-1 测得的集线器广播网络的效率约为 90.7%，略有提升（5.3%），符合理论实际。因此，交换机转发网络比集线器广播网络性能更优。