交换机转发实验

实验内容

- 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);
- 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

实验步骤及结果

`iface_info_t *lookup_port(u8 mac[ETH_ALEN])`实现

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

`void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)`实现

```
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
{
    // TODO: implement the insertion process here
    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]);//未发现哈希函数 全部放进第一个
    fprintf(stdout, "TODO: implement the insertion process here.\n");
}
```

`int sweep_aged_mac_port_entry()` 实现

```
int sweep_aged_mac_port_entry()
{
    // TODO: implement the sweeping process here
    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;
            }
        }
    }
    fprintf(stdout, "TODO: implement the sweeping process here.\n");
    return 0;
}
```

`void broadcast_packet(iface_info_t *iface, const char *packet, int len)` 实现

同实验1方法

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

```
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");
    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 *tx_iface = lookup_port(eh->ether_dhost);
    if (tx_iface) {
        iface_send_packet(tx_iface, packet, len);
    }
    else {
        broadcast_packet(iface, packet, len);
    }
    if (!lookup_port(eh->ether_shost)) {
        insert_mac_port(eh->ether_shost, iface);
    }
    free(packet);
}
```

运行步骤

make编译

```
nowcoder@nowcoder:~/ucas_network$ cd 2-switching/
nowcoder@nowcoder:~/ucas_network/2-switching$ ls
broadcast.c        hub-reference      include  Makefile          switch-reference.32
device_internal.c  hub-reference.32   mac.c    scripts           switch-reference.arm
example            hub-reference.arm  main.c   switch-reference  three_nodes_bw.py
nowcoder@nowcoder:~/ucas_network/2-switching$ make
gcc -c -g -Wall -Iinclude broadcast.c -o broadcast.o
gcc -c -g -Wall -Iinclude device_internal.c -o device_internal.o
gcc -c -g -Wall -Iinclude mac.c -o mac.o
gcc -c -g -Wall -Iinclude main.c -o main.o
gcc  broadcast.o device_internal.o mac.o main.o -o switch -lpthread
nowcoder@nowcoder:~/ucas_network/2-switching$ ls
broadcast.c        example            include  main.o    switch-reference
broadcast.o        hub-reference      mac.c    Makefile  switch-reference.32
device_internal.c  hub-reference.32   mac.o    scripts   switch-reference.arm
device_internal.o  hub-reference.arm  main.c   switch    three_nodes_bw.py
```

执行py代码

```
nowcoder@nowcoder:~/ucas_network/2-switching$ sudo python2 three_nodes_bw.py
mininet> net
h1 h1-eth0:s1-eth0
h2 h2-eth0:s1-eth1
h3 h3-eth0:s1-eth2
s1 s1-eth0:h1-eth0 s1-eth1:h2-eth0 s1-eth2:h3-eth0
```

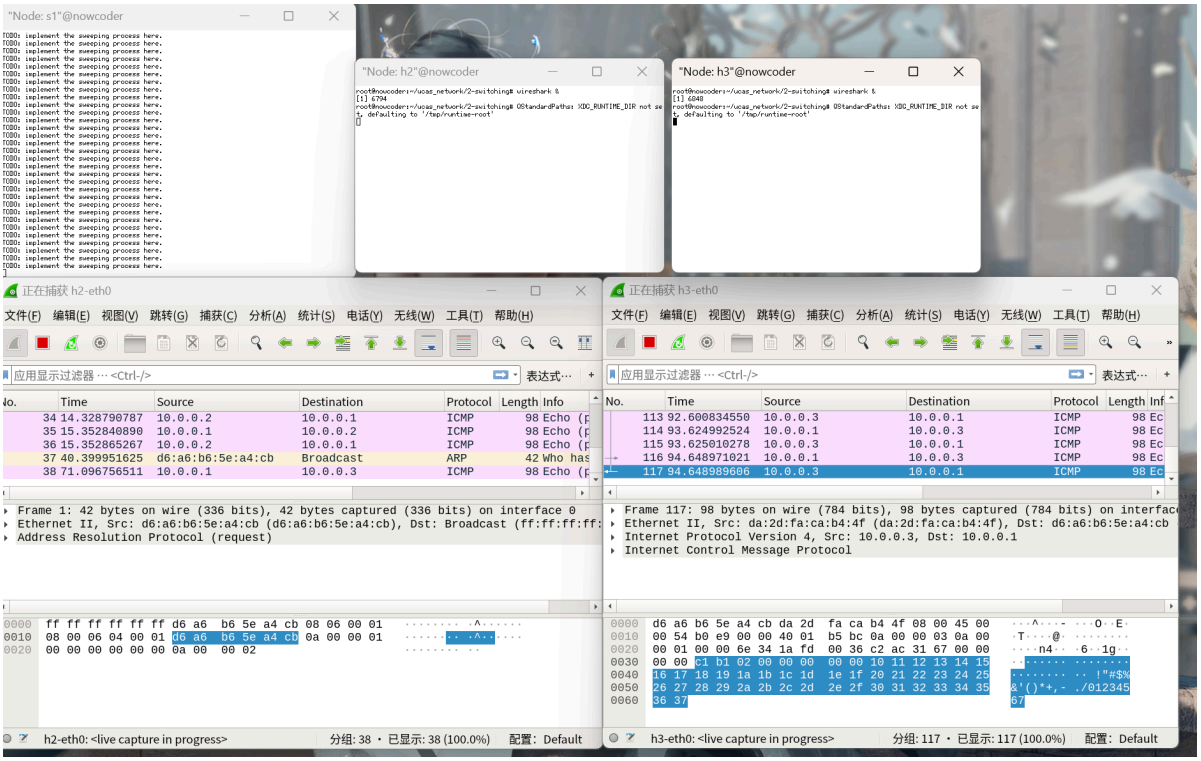
打开h1 h2 h3 s1节点终端，并在s1中执行交换器`./switch`

```
mininet> xterm s1
mininet> xterm h2 h3 h1
```

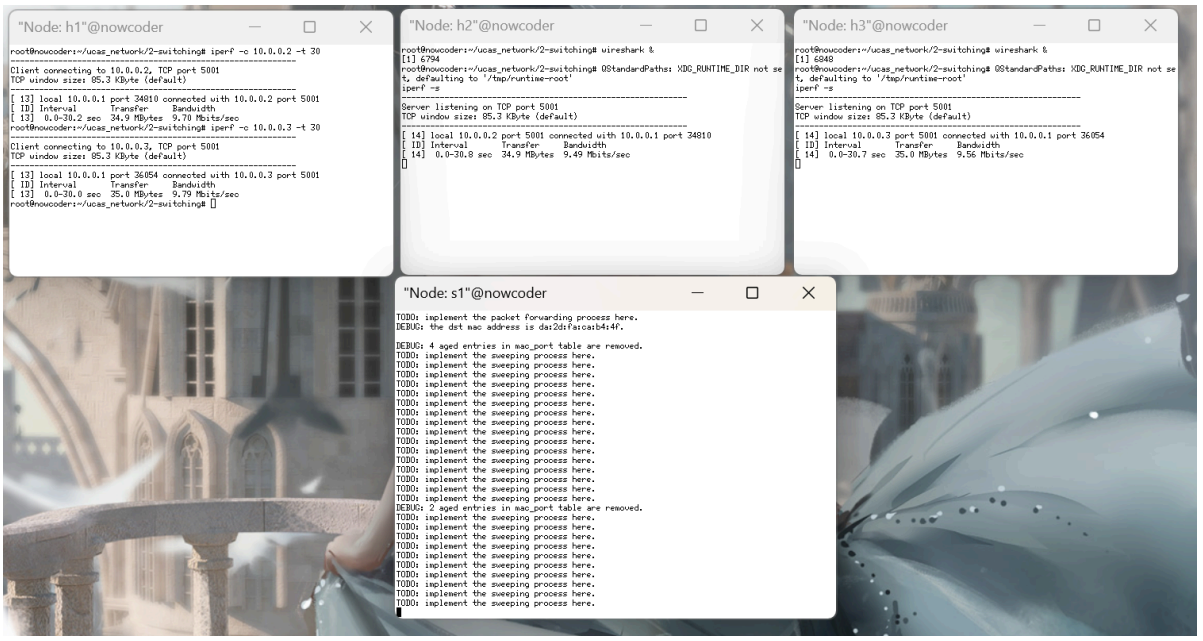
此时用h1分别ping h2 h3 同时在抓包工具中监视

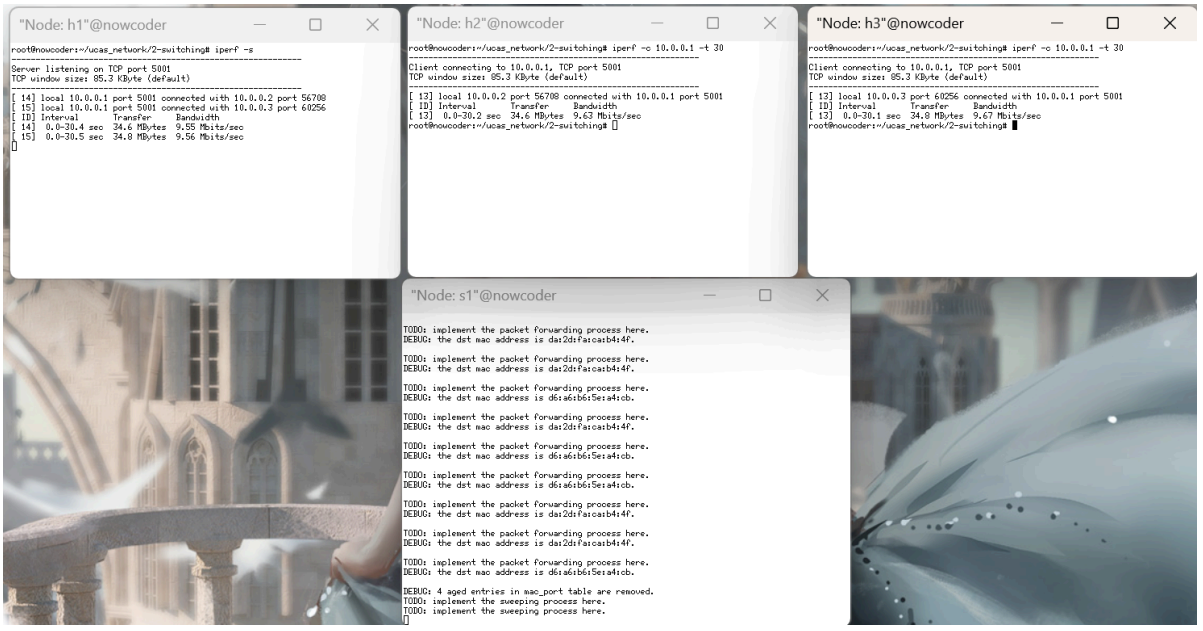
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.107 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.203 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.157 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.177 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.233 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.174 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.257 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.167 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.181 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.156 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.257 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.175 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.172 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.172 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.211 ms
^C
--- 10.0.0.2 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15352ms
```

```
mininet> h1 ping h3
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=0.111 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.174 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.165 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.137 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.112 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.127 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.140 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=0.159 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=0.149 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=0.256 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=0.158 ms
64 bytes from 10.0.0.3: icmp_seq=12 ttl=64 time=0.159 ms
64 bytes from 10.0.0.3: icmp_seq=13 ttl=64 time=0.126 ms
64 bytes from 10.0.0.3: icmp_seq=14 ttl=64 time=0.165 ms
64 bytes from 10.0.0.3: icmp_seq=15 ttl=64 time=0.159 ms
64 bytes from 10.0.0.3: icmp_seq=16 ttl=64 time=0.193 ms
64 bytes from 10.0.0.3: icmp_seq=17 ttl=64 time=0.218 ms
64 bytes from 10.0.0.3: icmp_seq=18 ttl=64 time=0.160 ms
64 bytes from 10.0.0.3: icmp_seq=19 ttl=64 time=0.180 ms
64 bytes from 10.0.0.3: icmp_seq=20 ttl=64 time=0.149 ms
64 bytes from 10.0.0.3: icmp_seq=21 ttl=64 time=0.160 ms
64 bytes from 10.0.0.3: icmp_seq=22 ttl=64 time=0.162 ms
64 bytes from 10.0.0.3: icmp_seq=23 ttl=64 time=0.132 ms
64 bytes from 10.0.0.3: icmp_seq=24 ttl=64 time=0.216 ms
64 bytes from 10.0.0.3: icmp_seq=25 ttl=64 time=0.230 ms
64 bytes from 10.0.0.3: icmp_seq=26 ttl=64 time=0.090 ms
64 bytes from 10.0.0.3: icmp_seq=27 ttl=64 time=0.146 ms
64 bytes from 10.0.0.3: icmp_seq=28 ttl=64 time=0.162 ms
64 bytes from 10.0.0.3: icmp_seq=29 ttl=64 time=0.261 ms
64 bytes from 10.0.0.3: icmp_seq=30 ttl=64 time=0.149 ms
64 bytes from 10.0.0.3: icmp_seq=31 ttl=64 time=0.161 ms
```



测量带宽性能





和广播对比，有明显效率提升