交换机转发实验

学号：2022K8009929011
姓名：王泽黎

实验任务一（hub）

一、任务一：实验目的

了解广播网络的原理，实现节点广播的 broadcast_packet 函数。验证广播网络能够正常运行，并通过 iperf 测试广播网络的效率，掌握其运行特点。最后构建环形拓扑网络，验证该拓扑下节点广播会产生数据包环路。

二、任务一：实验流程

1. 实现节点广播的 broadcast_packet 函数
2. 验证广播网络能够正常运行
    - 从一个端节点 ping 另一个端节点
3. 验证广播网络的效率
    - 在 three_nodes_bw.py 进行 iperf 测量
    - 两种场景：h1 同时向 h2 和 h3 测量；h2 和 h3 同时向 h1 测量
4. 构建环形拓扑网络，验证该拓扑下节点广播会产生数据包环路

三、任务一：实验结果与分析

（一）实现节点广播

1. 广播节点设计思路
    - 广播节点的逻辑较为简单，即每次收到网络包消息时，遍历与之相邻的每个网络端口，只要不是发送该网络包的端口，就将网络包广播到这个端口。代码如下：

```c
#include "base.h"
#include <stdio.h>

extern ustack_t *instance;

// the function for sending packet, defined in device_internal.c
extern void iface_send_packet(iface_info_t *iface, const char *packet, int len);

// the memory of ``packet'' will be free'd in handle_packet().
void broadcast_packet(iface_info_t *iface, const char *packet, int len)
{
    // TODO: broadcast packet
    iface_info_t *iface_entry;

    list_for_each_entry(iface_entry, &instance -> iface_list, list)
    {
        if (iface_entry -> fd != iface -> fd)
        {
            iface_send_packet(iface_entry, packet, len);
        }
    }
}

```

2. 验证广播网络能够正常运行
   - 三个节点各自向其它两个节点发送消息，验证其两两相互连通。
     - h1 节点的验证结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# ping 10.
0.0.2 -c 4
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=0.945 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.125 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3037ms
rtt min/avg/max/mdev = 0.113/0.324/0.945/0.358 ms
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# ping 10.
0.0.3 -c 4
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=0.232 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.112 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.169 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.119 ms

--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.112/0.158/0.232/0.048 ms
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub#
```

     - h2 节点的验证结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub#ping 10.0.0.1 -c 4
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.161 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.123 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.101 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.092 ms

--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3071ms
rtt min/avg/max/mdev = 0.092/0.119/0.161/0.026 ms
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# ping 10.0.0.3 -c 4
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=1.11 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.129 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.155 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.084 ms

--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3087ms
rtt min/avg/max/mdev = 0.084/0.369/1.109/0.427 ms
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub#
```

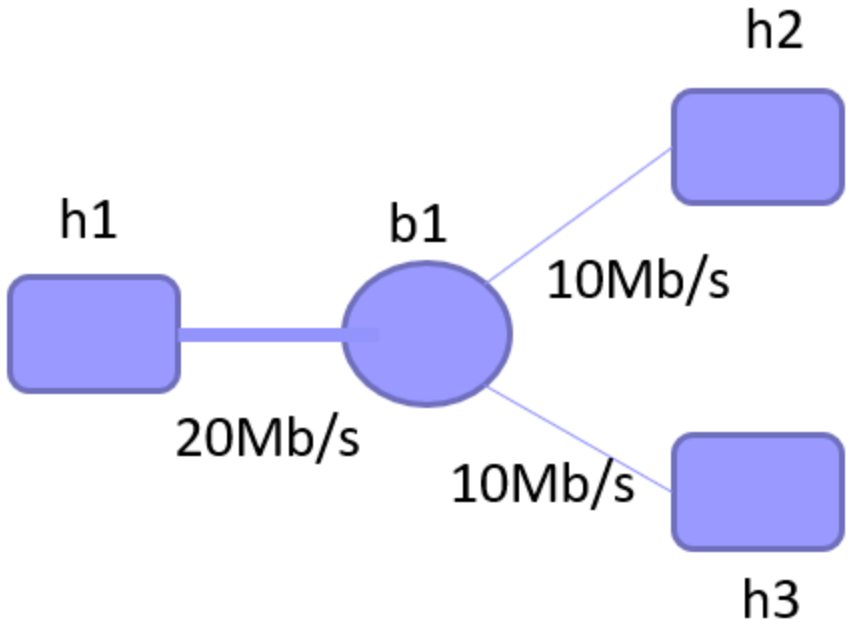
  - h3 节点的验证结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# ping 10.0.0.1 -c 4
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.117 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.124 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.152 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.131 ms

--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3077ms
rtt min/avg/max/mdev = 0.117/0.131/0.152/0.013 ms
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# ping 10.0.0.2 -c 4
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=0.273 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.130 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.154 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.194 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3061ms
rtt min/avg/max/mdev = 0.130/0.187/0.273/0.054 ms
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub#
```

- 综上，三个节点两两连通，广播网络能够正常运行

（二）验证广播网络的效率

1. 网络的拓扑结构如下所示：



2. h1 向 h2 和 h3 同时传输
   - h1 向 h2 传输的结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# 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)
------------------------------------------------------------
[  1] local 10.0.0.1 port 38450 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.2695 sec  8.88 MBytes  2.46 Mbits/sec
```

   - h1 向 h3 传输的结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# iperf -c
 10.0.0.3 -t 30
------------------------------------------------------------
Client connecting to 10.0.0.3, TCP port 5001
TCP window size:  110 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 49200 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.9259 sec  27.0 MBytes  7.32 Mbits/sec
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub#
```

   - 从上图可以看出，h1 到 h2 实际传输速率和 h1 到 h3 实际传输速率都小于h2/h3 到b1的带宽（10.0Mb/s）。同时我们也能注意到两者速率加起来差不多正好达到 b1 到 h2/h3 的带宽（10.0Mb/s）。

     这是由于 h1 发给 h2的数据在 b1，会同时广播给 h2 和 h3，这样给 h2 的数据也会占据 h3 的传输带宽。同理，h1 发给 h3 的数据也会占据b1到h2的传输带宽。于是 b1 到 h2 和 b1 到 h3 两条传输通路都会传输h1发送给h2和h3的全部数据。因此 h1 到 h2 的传输速率与 h1 到 h3 的传输速率都会小于 b1 到 h2/h3 的带宽 10Mb/s。

     理论上来说，b1 到 h2 和 b1 到 h3 两条通路的效率应都为 50%左右。但实际中 h1 到h2 的速率与 hl 到h3 的速率有些差异，笔者认为这是受到了先后启动的影响，测试进程先启动的一方 TCP 窗口更大，速率会略大一些。但无论如何两者速率之和上限只有 10Mb/s，传输速率远没有达到带宽，可以看出广播网络效率低下。

3. h2 和 h3 向 h1 同时传输

- h2 向 h1 传输的结果如下：

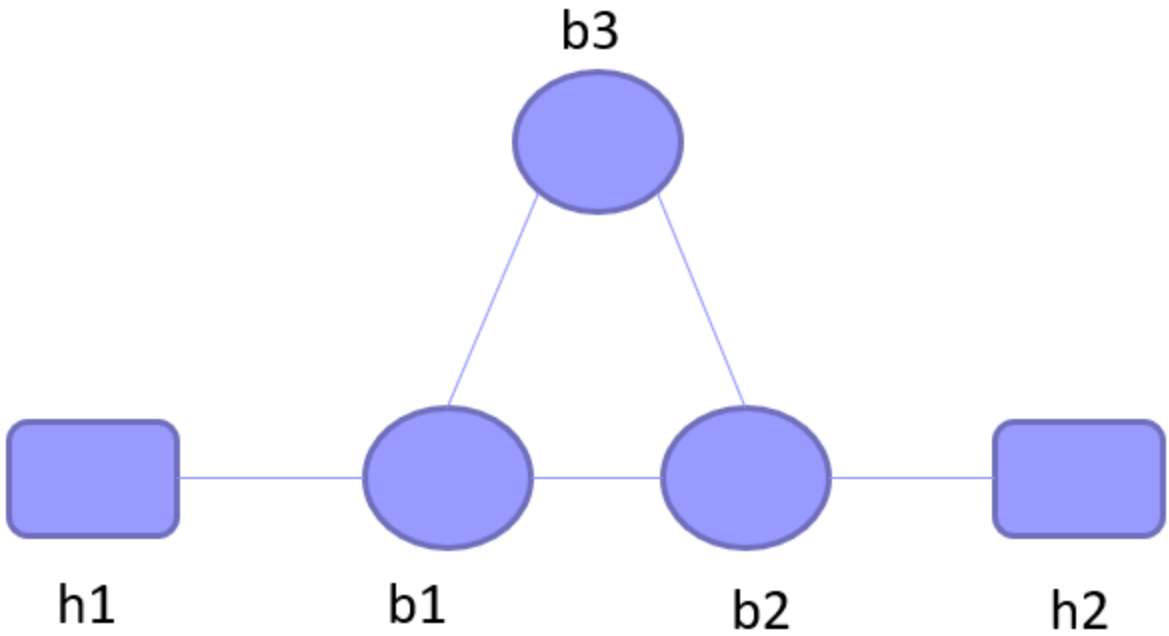
```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# iperf -c
 10.0.0.1 -t 30
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.2 port 33198 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.2328 sec  33.4 MBytes  8.96 Mbits/sec
```

- h3 向 h1 传输的结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/hub# 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 36076 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.9954 sec  33.4 MBytes  9.03 Mbits/sec
```

- 从上图可以看出，实验中 h2 到 h1 和 h3 到 h1 的实际传输速率都接近 10Mb/s，与 h2/h3 到 b1 的带宽一致。
  这是因为，h2 发给 h1 的数据在 b1 处，会同时广播给 h1 和 h3；而 h3 发给 h1 的数据也会在 b1 处，同时广播给 h1 和 h2。但是这时数据并不是竞争关系，而是处于链路的两个不同方向。h2 给 h1 的数据从 b1 传到 h3，而 h3 给 h1 的数据从 h3 传到 b1，虽然他们都使用了 b1-h3 链路，但却是不同方向，互不影响，所以能达到链路最大带宽。而 b1 到 h1 链路带宽为 20Mb/s，刚好可以接收2个同时满带宽的 10Mb/s数据。因此在，路上的每一个链路带宽都被完全利用了，广播网络的效率达到最高。

4. 数据包在环路中不断广播
   - 对 three_nodes_bw.py 文件进行更改，将网络改为由2个 host 节点、3个 hub 节点构成的环状网络。



   - 其中，除了增加 b2 和 b3 节点的声明及相关定义外，最重要的是重新构建节点间的互联关系，以实现实验要求的环形拓扑。如下图所示，需要建立共计5条连接：b1 和 b2 和 b3 互相的连接、b1 和 h1 的连接、b2 和 h2 的连接。

```python
class BroadcastTopo(Topo):
    def build(self):
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        h3 = self.addHost('h3')
        b1 = self.addHost('b1')
        b2 = self.addHost('b2')
        b3 = self.addHost('b3')

        self.addLink(h1, b1, bw=10)
        self.addLink(h2, b2, bw=10)
        self.addLink(b1, b2, bw=20)
        self.addLink(b2, b3, bw=20)
        self.addLink(b3, b1, bw=20)
```

部分抓包结果如下：

```
58843 4.094268903   7e:e6:8e:f8:3b:b6   d6:65:15:ca:f2:9d   ARP    42 10.0.0.2 is at 7e:e6:8e:f8:3b:b6
58844 4.094269071   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58845 4.094269170   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58846 4.094379509   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58847 4.094379642   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58848 4.094494799   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58849 4.094494936   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58850 4.094604197   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58851 4.094604298   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58852 4.094719698   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58853 4.094719831   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58854 4.094838896   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58855 4.094839124   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58856 4.094950650   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58857 4.094950771   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58858 4.095070658   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58859 4.095070890   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58860 4.095180815   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58861 4.095180894   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58862 4.095289905   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58863 4.095290000   7e:e6:8e:f8:3b:b6   d6:65:15:ca:f2:9d   ARP    42 10.0.0.2 is at 7e:e6:8e:f8:3b:b6
58864 4.095399574   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58865 4.095399660   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58866 4.095508666   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58867 4.095508893   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58868 4.095617611   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58869 4.095617677   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58870 4.095726381   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58871 4.095726525   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
58872 4.095837274   10.0.0.2            10.0.0.1            ICMP   98 Echo (ping) reply    id=0x5b33, seq=4/1024, ttl=64
58873 4.095837476   d6:65:15:ca:f2:9d   7e:e6:8e:f8:3b:b6   ARP    42 10.0.0.1 is at d6:65:15:ca:f2:9d
```

- 从抓包结果可以看出，数据包在环路中不断广播，形成了数据包环路。
  造成这种数据包环路现象的原因是网络中 hub 节点构成了一个环。由于广播网络的工作模式，当网络包从 h2 达到 b2 后，b2 将数据包广播到 b3、b1。而下一时刻，b3 又将数据包广播到 b1，b1 又将数据包广播到 b3。之后它们又将数据包传回 b2，然后在 hub 环中重复上面的过程，这使得数据包的传输在网络中不断循环转发。

四、任务一：实验总结

通过本次实验，我对广播网络有了更多的了解。在本次实验中，我更加深入的明白了广播网络的工作方式，并直接体会到了广播网络的效率特点，知道了数据传输方向对其的影响，也明白了广播的方式效率不高。而在最后一个实验中，我深刻认识到广播网络有着致命的弱点。其要求拓扑结构不能有环路，否则会造成数据包在环路中不断被转发，占据资源，冬儿对网络产生极大破坏。

实验任务二（switch）

1.任务二：实验目的

了解交换机的转发原理和转发表的构建方式，理解交换机如何学习和维护转发表。实现转发表的数据结构，支持转发表的查询、插入、老化操作，完成一个能自动学习转发表的交换机。使用 iperf 和给定的拓扑进行测试，对比交换机转发与之前集线器广播的性能差异。

2.任务二：实验流程

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

3.任务二：实验结果与分析

（一）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

1. 交换机查询操作：

```c
// lookup the mac address in mac_port table
iface_info_t *lookup_port(u8 mac[ETH_ALEN])
{
    // TODO: implement the lookup process here
    // hash8 is a hash function provided by hash.h
    int idx = (int)hash8((char*)mac, ETH_ALEN);
    mac_port_entry_t *entry;
    // lock the table to prevent
    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)
        {
            pthread_mutex_unlock(&mac_port_map.lock);
            return entry->iface;
        }
    }
    // Unlock the table
    pthread_mutex_unlock(&mac_port_map.lock);

    return NULL;
}
```

2. 交换机插入操作：

```c
// insert the mac -> iface mapping into mac_port table
void insert_mac_port(u8 mac[ETH_ALEN], iface_info_t *iface)
{
    // TODO: implement the insertion process here
    int idx = (int)hash8((char*)mac, ETH_ALEN);
    mac_port_entry_t *entry;
    time_t now = time(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)
        {
            if(entry->iface!=iface)
                entry->iface = iface;
            entry->visited = now;
            pthread_mutex_unlock(&mac_port_map.lock);

            return ;
        }
    }

    mac_port_entry_t *new = malloc(sizeof(mac_port_entry_t));
    new->iface = iface;
    new->visited = now;
    for(int i=0;i<ETH_ALEN;i++)
        new->mac[i] = mac[i];

    list_add_head(&new->list, &(mac_port_map.hash_table[idx]));
    pthread_mutex_unlock(&(mac_port_map.lock));

    return ;
}

```

3. 交换机老化操作：

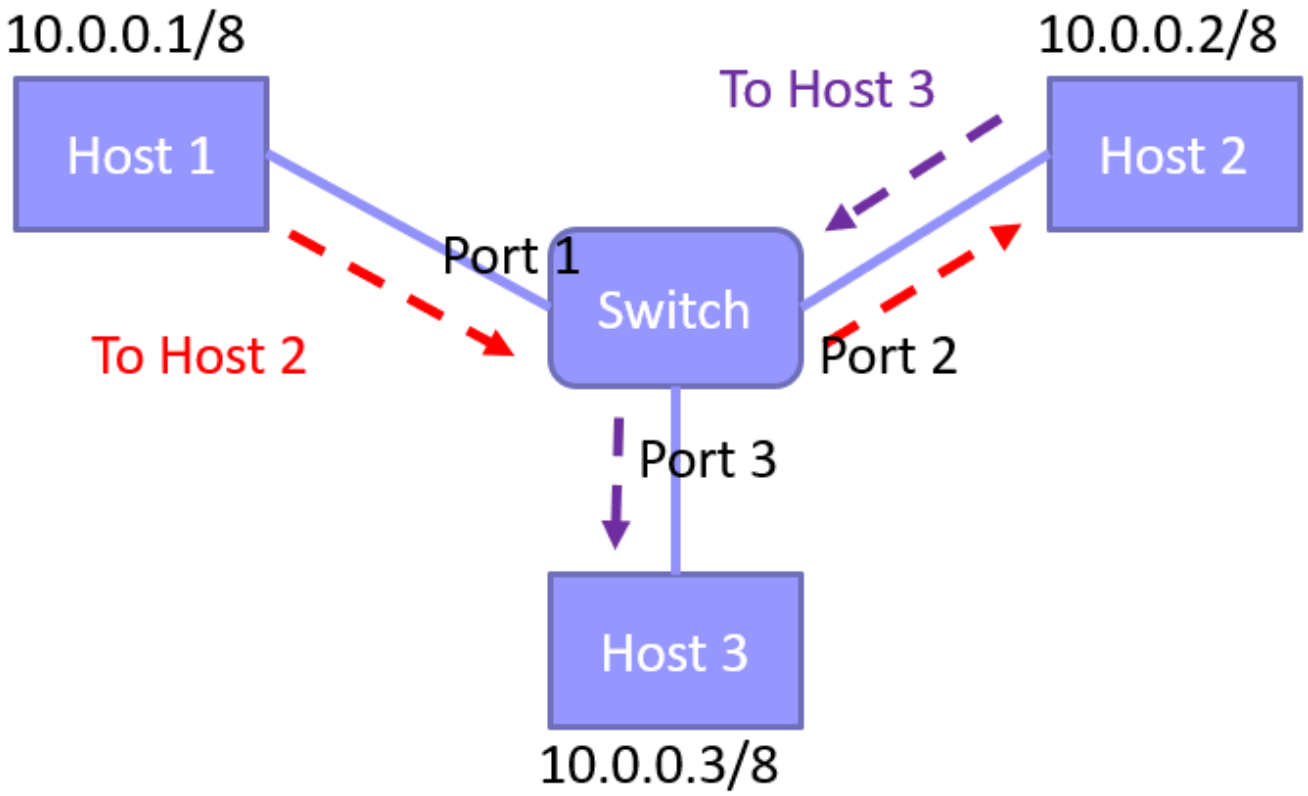
```c
// sweeping mac_port table, remove the entry which has not been visited in the
// last 30 seconds.
int sweep_aged_mac_port_entry()
{
    // TODO: implement the sweeping process here
    int n = 0;
    mac_port_entry_t *entry, *q;
    time_t now = time(NULL);

    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((int)(now - entry->visited) > MAC_PORT_TIMEOUT)
            {
                list_delete_entry(&entry->list);
                free(entry);
                n++;
            }
        }
    }
    pthread_mutex_unlock(&mac_port_map.lock);

    return n;
}
```

4. 交换机广播操作（与任务一相同，不重复展示）
5. 交换机处理数据包操作：

```c
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 * dest_iface = lookup_port(eh->ether_dhost);
    if (dest_iface)
    {
        // log(DEBUG, "Send this packet to %s.", dest_iface->name);
        iface_send_packet(dest_iface, packet, len);
    } else
    {
        // log(DEBUG, "Broadcast this packet.");
        broadcast_packet(iface, packet, len);
    }

    insert_mac_port(eh->ether_shost, iface);

    free(packet);
}
```

（二）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. 网络的拓扑结构如下所示：



2. h1 向 h2 和 h3 同时传输
   - h1 向 h2 传输的结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/switch# 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)
------------------------------------------------------------
[  1] local 10.0.0.1 port 48024 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-32.3342 sec  36.0 MBytes  9.34 Mbits/sec
```

   - h1 向 h3 传输的结果如下：

```
root@zeri-virtual-machine:/home/zeri/2024Fall_CNLab/04-hub+switch/switch# 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)
------------------------------------------------------------
[  1] local 10.0.0.1 port 35470 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-33.5408 sec  36.5 MBytes  9.13 Mbits/sec
```

   - 从上图可以看出，实验中 h2 到 h1 和 h3 到 h1 的实际传输速率都接近 10Mb/s，与 h2/h3 到 b1 的带宽一致，可以说各自链路都充分利用了各自带宽。
     在交换机的情况下，除了第一次通讯时由于转发表为空需要广播数据包以外，由于服务端会发送响应包，所以，从第二个数据包开始，交换机就直接只向对应的端口进行转发，使得连接 h2 和 h3 的两条通路可以相对独立地进行数据收发。这样带宽得到了充分的利用，此时，只有交换机转的软件处理的时间（如交换表查询等操作）可能影响传输的效率。
3. h2 和 h3 向 h1 同时传输
   - h2和h3向h1传输的结果如下：

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 53438
[  2] local 10.0.0.1 port 5001 connected with 10.0.0.3 port 44690
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-32.2138 sec  35.1 MBytes  9.15 Mbits/sec
[  2] 0.0000-32.5354 sec  35.3 MBytes  9.09 Mbits/sec
```

   - 从上图可以看出，实验中 h2 到 h1 和 h3 到 h1 的实际传输速率都接近 10Mb/s，与 h2/h3 到 b1 的带宽一致，可以说各自链路都充分利用了各自带宽。而具体原因与上一个测量实验（h1 向h2和h3同时传输）一致，这里不进行赘述了，这种情况下广播网络也能充分利用各自的带宽，具体原因可见实验任务一。
4. 交换机转发与集线器广播性能对比
   - 对于集线器广播方式，从实验任务一中可以得知，h1向h2和h3发送时，总体速率远小于其对应的带宽。这是因为广播方式会将数据包向所有端口发送，占用其他链路的带宽。交换机在第一次转发时，也是广播方式。但之后学习到 MAC 地址和端口的对应关系之后，就只向目的端口发送数据包，只占用该链路带宽。因此总体上利用率可以拉满，实际传输速率接近满带宽。
     而对于 h2和 h3同时向h1发送数据时，集线器和交换机没有什么差异，都能完全利用带宽（具体分析可参见前文）。
     但总的来说，交换机的性能较好，它在学习完毕后可以定向发送数据包，没有无用数据挤占带宽。同时它不受传输方向的影响，上下行效率一致。而集线器效率受传输方向影响很大，上下

行不对等，坏的情况下效率很低。此外在节点更多后无用数据会更加挤占带宽，性能受到局限。

4.任务二：实验总结

通过本次实验，我对交换机及其工作原理有了更加深入的了解。首先，我学到了交换机的工作方式，即通过转发表来学习MAC地址与端口的对应关系，以此优化转发。其次，我知道了转发表的组织结构，明白了转发表的一些基本操作，并且本次实验中，我实现了转发表和交换机，并对比分析了集线器和交换机的性能差异，这让我对交换机的优势以及其巧妙的设计有了更准确的认识，这让我对交换机有了更深的理解。