Lab of Computer Network: Hub & Switch **Fall 2024**

Report 4 — September 26

Lecturer: Wu Qinghua *Completed by: Zhang Jiawei*

第一部分 Hub

4.1 实验内容

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(分两种场景在 three_nodes_bw.py 进行 iperf 测量,h1 同时向 h2 和 h3 测量与 h2 和 h3 同时向 h1 测量)
4. 自己动手构建环形拓扑,验证该拓扑下节点广播会产生数据包环路

4.2 实验过程

4.2.1 实现节点广播的 broadcast_packet 函数

```
// 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
    // fprintf(stdout, "TODO: broadcast packet.\n");
    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);
    }
}
```

在这个函数里,我们遍历所有的接口,如果遍历到的接口不是发送所收到网络包的接口,就调用 `iface_send_packet` 函数广播发送数据包。

4.2.2 验证广播网络能够正常运行

拓扑图如下:

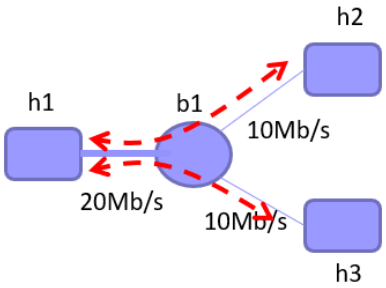

图 **4.1.** 广播网络拓扑图

h1, h2, h3 分别是三个节点，它们如果能够相互 ping 通，说明广播网络能够正常运行。这里 h1 的 IP 地址是 10.0.0.1，h2 的 IP 地址是 10.0.0.2，h3 的 IP 地址是 10.0.0.3。

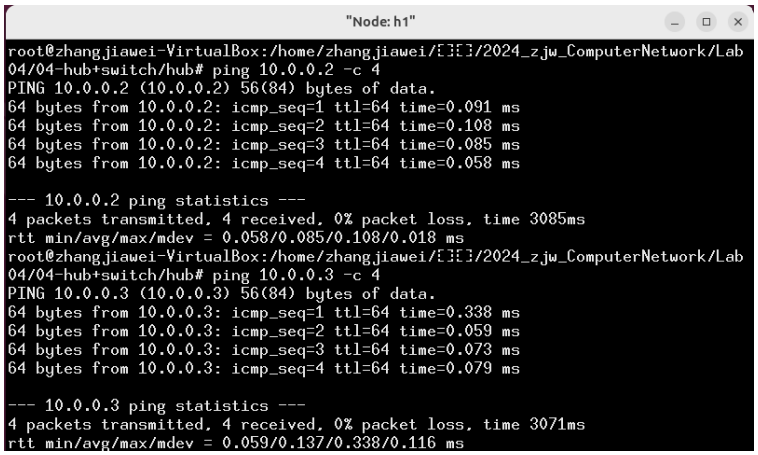

图 **4.2.** h1 ping h2, h3

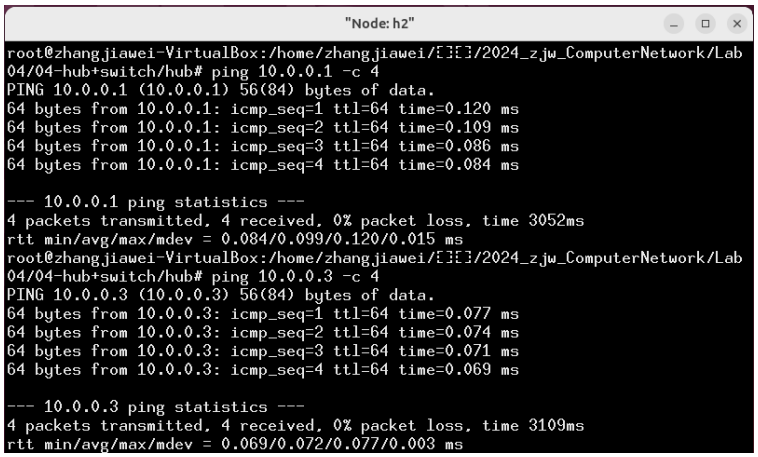

图 **4.3.** h2 ping h1, h3

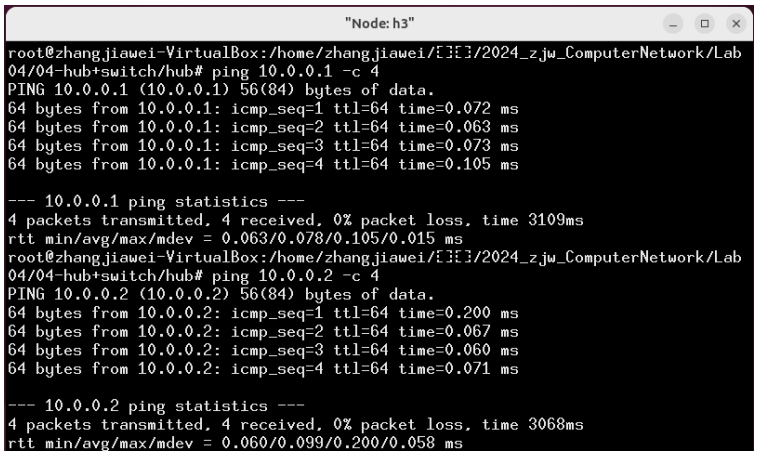

图 **4.4.** h3 ping h1, h2

可以看出，h1, h2, h3 三个节点确实能够相互 ping 通，这表明广播网络能够正常运行。

4.2.3 验证广播网络的效率

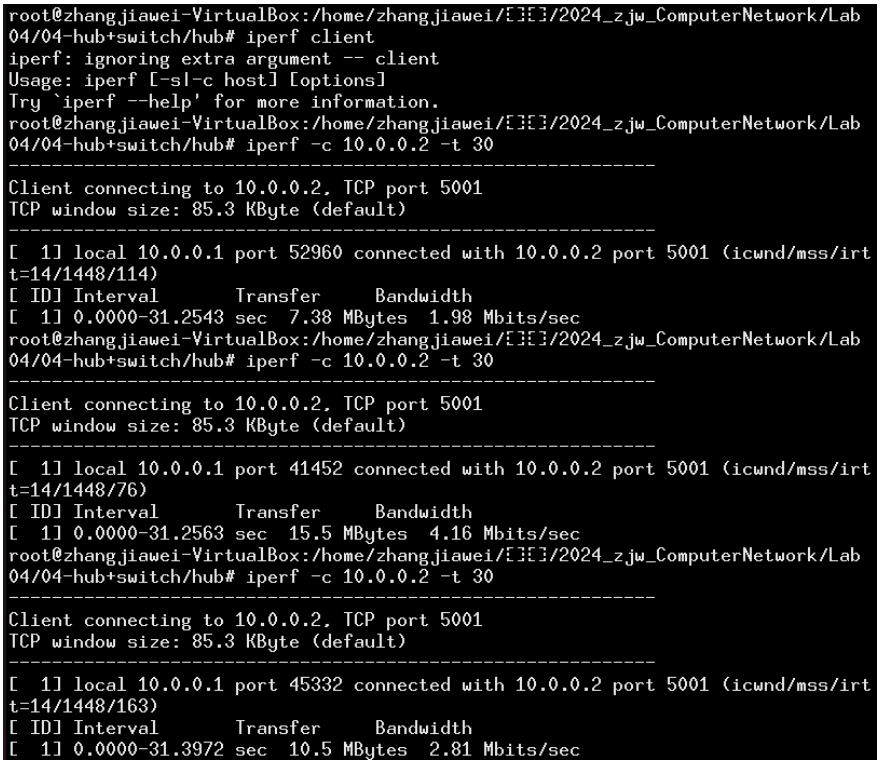

图 **4.5.** h1 向 h2 测量

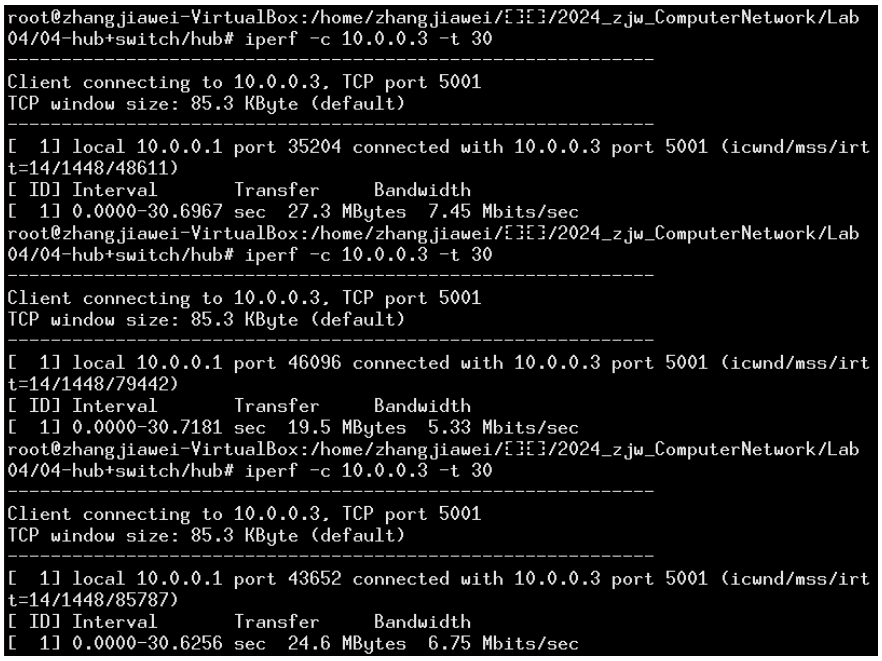

图 **4.6.** h1 向 h3 测量

可以看出，h1 向 h2 测量和 h1 向 h3 测量的实际速率差异较大，而且都远未达到理论带宽值。这是因为 h1 发出的数据都会广播至 h2 和 h3，实际来看，h1 发送给 h2 的数据会占据 b1 到 h3 的带宽，而 h1 发送给 h3 的数据也会占据 b1 到 h2 的带宽，所以实际速率必定达不到理论带宽值。而对于实际速率的差异，我的解释是，因为我们在 xterm 终端中并不能保证两个 iperf 进程同时启动，所以会造成 TCP 窗口大小的差异，从而导致实际速率的差异。可以预测出的是，在接下来的 h2 向 h1 测量和 h3 向 h1 测量中，因为路径不是竞争关系，所以速率应该会更接近理论带宽值。事实果然如此：

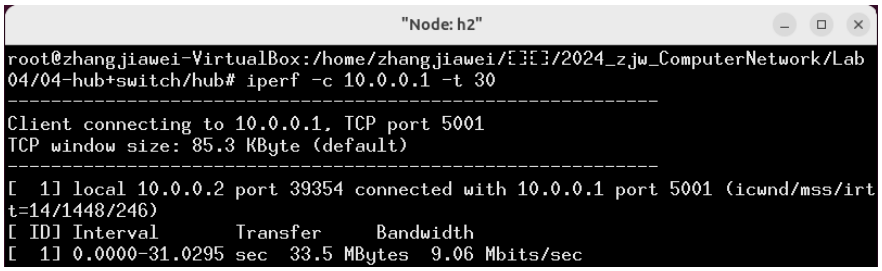

图 **4.7.** h2 向 h1 测量

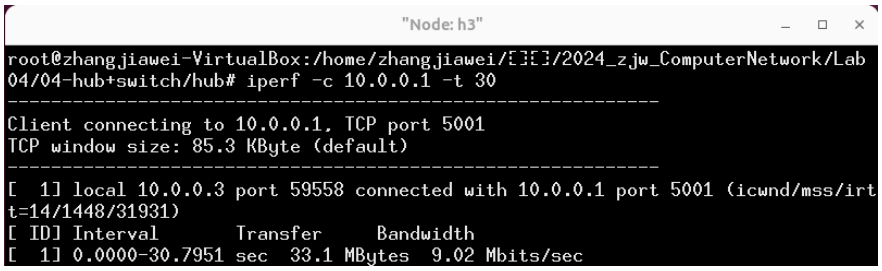

图 **4.8.** h3 向 h1 测量

4.2.4 构建环形拓扑,验证数据包环路

拓扑图如下：

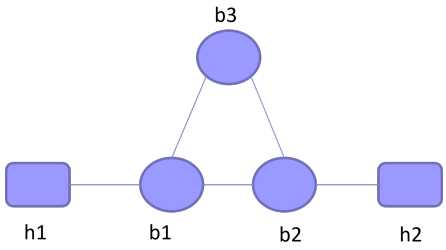

图 **4.9.** 环形拓扑图

此时需要更改 three_nodes_bw.py 文件,重新配置拓扑：

```
class BroadcastTopo(Topo):
def build(self):
    h1 = self.addHost('h1')
    h2 = self.addHost('h2')
    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=10)
    self.addLink(b2, b3, bw=10)
    self.addLink(b3, b1, bw=10)
```

使用 h1 向 h2 广播 ping 消息,经 wireshark 抓包,可以看到数据包在环路中不断传递：

No.	Time	Source	Destination	Protocol	Length	Info
4070…	26.599318531	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.599396930	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.599475200	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.599553726	2a:d2:14:e9:44:7d	72:75:7e:fe:90:74	ARP	42	10.0.0.1 is at 2a:d2:14:e9:44:7d
4070…	26.599587019	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.599665702	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.599744241	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.599822022	2a:d2:14:e9:44:7d	72:75:7e:fe:90:74	ARP	42	10.0.0.1 is at 2a:d2:14:e9:44:7d
4070…	26.599856163	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.599934259	2a:d2:14:e9:44:7d	72:75:7e:fe:90:74	ARP	42	10.0.0.1 is at 2a:d2:14:e9:44:7d
4070…	26.599968347	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600046493	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600124563	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600203348	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600277938	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600356105	2a:d2:14:e9:44:7d	72:75:7e:fe:90:74	ARP	42	10.0.0.1 is at 2a:d2:14:e9:44:7d
4070…	26.600388530	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600478161	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600553090	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600627009	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600708083	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600785838	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600862155	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.600939297	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.601018064	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.601096825	2a:d2:14:e9:44:7d	72:75:7e:fe:90:74	ARP	42	10.0.0.1 is at 2a:d2:14:e9:44:7d
4070…	26.601132956	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x3cf8, seq=1/256, ttl=64
4070…	26.601431655	2a:d2:14:e9:44:7d	72:75:7e:fe:90:74	ARP	42	10.0.0.1 is at 2a:d2:14:e9:44:7d

图 **4.10.** 数据包环路

可以看到，h1 一直收到来自 h2 的相同消息，这说明数据包在环路中不断被广播，根本在于拓扑网络出现了环路，使得数据包不断被循环转发。

4.3 实验总结

本次实验主要是关于广播网络的实验，我实现了节点广播的函数，验证了广播网络的正常运行和效率，以及验证了环形拓扑下数据包环路的现象。实验过程中，我对广播网络的工作原理有了更深入的了解，广播网络会占据更多的网络资源，效率低下。我也对环形拓扑下数据包环路的现象有了直观的认识，这是广播网络的一个关键弱点，一旦出现环路，数据包会不断被循环转发，直到网络崩溃。于是，为了避免这种情况以及提高效率，新的转发机制亟待设计，这就是马上就要进行实验的交换机网络。

第二部分 Switch

4.1 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

4.2 实验过程

4.2.1 实现转发表的查询操作

```
// lookup the mac address in mac_port table
iface_info_t *lookup_port(u8 mac[ETH_ALEN])
{
    // TODO: implement the lookup process here
    // fprintf(stdout, "TODO: implement the lookup process here.\n");
    int hash = (int)hash8((char *)mac, ETH_ALEN);
    mac_port_entry_t *entry;
    pthread_mutex_lock(&mac_port_map.lock);

    list_for_each_entry(entry, &mac_port_map.hash_table[hash], list) {
        if (memcmp(entry->mac, mac, ETH_ALEN) == 0) {
            pthread_mutex_unlock(&mac_port_map.lock);
            return entry->iface;
        }
    }

    pthread_mutex_unlock(&mac_port_map.lock);
    return NULL;
}
```

在这个函数里，我们首先计算出 mac 地址的哈希值，然后遍历哈希表，如果找到了对应的 mac 地址，就返回对应的接口信息，否则返回 NULL。函数中需要线程互斥锁是因为多个线程可能同时访问转发表（比如之后的老化操作）。

4.2.2 实现转发表的插入操作

```
// 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
    // fprintf(stdout, "TODO: implement the insertion process here.\n");
    int hash = (int)hash8((char *)mac, ETH_ALEN);
    mac_port_entry_t *entry;
    pthread_mutex_lock(&mac_port_map.lock);

    list_for_each_entry(entry, &mac_port_map.hash_table[hash], list) {
        if (memcmp(entry->mac, mac, ETH_ALEN) == 0) {
            entry->iface = iface;
            entry->visited = time(NULL);
            pthread_mutex_unlock(&mac_port_map.lock);
            return;
        }
    }

    mac_port_entry_t *nentry = (mac_port_entry_t *)malloc(sizeof(mac_port_entry_t));
    nentry->iface = iface;
    nentry->visited = time(NULL);
    for (int i = 0; i < ETH_ALEN; i++)
        nentry->mac[i] = mac[i];
    list_add_head(&nentry->list, &mac_port_map.hash_table[hash]);

    pthread_mutex_unlock(&mac_port_map.lock);
    return;
}
```

仍然计算出 mac 地址的哈希值，然后遍历哈希表，如果找到了对应的 mac 地址，就更新对应的接口信息，否则新建一个新的表项插入到哈希表中。这里需要注意的是，每次插入新的表项时，需要更新访问时间。同理，函数中需要线程互斥锁，因为多个线程可能同时访问转发表（比如之后的老化操作）。

4.2.3 实现转发表的老化操作

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

```
    mac_port_entry_t *entry, *q;
    int n = 0;
    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;
}
```

老化操作是删除那些访问时间超过 30 秒的表项，即扫描转发表，清理那些当前时间与上次访问时间之差大于 30 秒的表项。这里需要注意的是，老化操作是线程安全的，因为在老化操作时，可能有其他线程在访问转发表（比如插入操作）。

4.2.4 实现数据包的广播操作

代码与实验一中的相同，这里不再赘述。

4.2.5 实现数据包的处理

```
// handle packet
// 1. if the dest mac address is found in mac_port table, forward it; otherwise,
// broadcast it.
// 2. put the src mac -> iface mapping into mac hash table.
// 3. release the memory of ``packet''

// Note: the log & fprintf here are only used for debug, which should be commented
// out for better performance.
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 *dst_iface = lookup_port(eh->ether_dhost);
    if (dst_iface)
        iface_send_packet(dst_iface, packet, len);
    else
        broadcast_packet(iface, packet, len);

    insert_mac_port(eh->ether_shost, iface);

    free(packet);
}
```

在这个函数里，我们首先解析数据包，然后查找目的 mac 地址对应的接口信息，如果找到了，就转发数据包，否则广播发送数据包。接着将源 mac 地址和接口信息插入到转发表中，最后释放数据包的内存。

4.2.6 测量交换机转发的性能与集线器广播的性能对比

拓扑图如下：

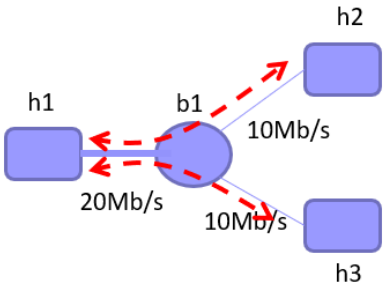

图 4.11. 交换机网络拓扑图

与实验一中的拓扑相同，这里直接开始测量交换机转发的性能。

```
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-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 45362 connected with 10.0.0.2 port 5001 (icwnd/mss/irt
t=14/1448/3677)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.6698 sec  34.4 MBytes  9.40 Mbits/sec
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-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 33130 connected with 10.0.0.2 port 5001 (icwnd/mss/irt
t=14/1448/1777)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.4188 sec  35.0 MBytes  9.34 Mbits/sec
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-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 58810 connected with 10.0.0.2 port 5001 (icwnd/mss/irt
t=14/1448/112)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-32.0691 sec  36.0 MBytes  9.42 Mbits/sec
```

图 **4.12.** h1 向 h2 测量

```
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-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 59052 connected with 10.0.0.3 port 5001 (icwnd/mss/irt
t=14/1448/119)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.8253 sec  35.0 MBytes  9.52 Mbits/sec
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-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 40816 connected with 1
t=14/1448/194)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.3387 sec  35.6 MBytes  9.54 Mbit
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□
04/04-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 41236 connected with 1
t=14/1448/95)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.4644 sec  35.3 MBytes  9.40 Mbit
```

图 **4.13.** h1 向 h3 测量

```
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-hub+switch/switch# 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 44908 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/13436)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.4415 sec  35.8 MBytes  9.54 Mbits/sec
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-hub+switch/switch# 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 35356 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/4115)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.8408 sec  36.3 MBytes  9.55 Mbits/sec
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-hub+switch/switch# iperf -c 10.0.0.1 -t 30
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 93.5 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.2 port 42720 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/58)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.5668 sec  35.9 MBytes  9.53 Mbits/sec
```

图 **4.14.** h2 向 h1 测量

```
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-hub+switch/switch# 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 37522 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/152)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.2045 sec  35.4 MBytes  9.51 Mbits/sec
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-hub+switch/switch# 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 43848 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/95)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.9559 sec  36.3 MBytes  9.52 Mbits/sec
root@zhangjiawei-VirtualBox:/home/zhangjiawei/□□□/2024_zjw_ComputerNetwork/Lab
04/04-hub+switch/switch# 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 55024 connected with 10.0.0.1 port 5001 (icwnd/mss/irt
t=14/1448/131)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-31.8827 sec  36.3 MBytes  9.54 Mbits/sec
```

图 **4.15.** h3 向 h1 测量

在 h1 向 h2 测量和 h1 向 h3 测量中，实验结果与实验一差距较大，速率已经接近理论带宽值。这是因为交换机网络中，数据包除了第一次转发表为空之外不会被广播，而是根据目的 mac 地址直接转发，所以不会出现

竞争关系，速率更接近理论带宽值，带宽得到了充分利用。

h2 向 h1 测量和 h3 向 h1 测量的结果也与预期一致，速率接近理论带宽值，这是因为路径不是竞争关系，所以速率应该会更接近理论带宽值。这个原因与实验一中的解释相同。

总体而言，交换机网络的性能要好于集线器广播网络，尤其是当广播网络因各种各样原因导致数据包竞争时，交换机网络的性能优势更加明显。即便首次传播时需要进行广播，但是交换机网络会学习到目的 mac 地址，之后的数据包都会直接转发，不会再广播，这样就避免了数据包竞争，提高了网络性能。

4.3 实验总结

本次实验主要是关于交换机网络的实验，我实现了转发表的查询、插入和老化操作，以及数据包的转发和广播操作，验证了交换机网络的性能优势。实验过程中，我对交换机网络的工作原理有了更深入的了解，交换机网络会学习到目的 mac 地址，之后的数据包都会直接转发，不会再广播，这样就避免了数据包竞争，提高了网络性能。而集线器广播网络则会占据更多的网络资源，效率低下。这也是交换机网络取代集线器网络的原因之一。