

第四周实验报告

广播网络实验

2015K8009922021

李一苇

一、实验内容

实现一个能转发数据包的 Hub，使得连接到同一交换机的主机能通信。

并在此基础上进行三个子实验：

- 验证广播网络正常运行
- 验证广播网络效率
- 验证环形拓扑的广播网路会产生数据包环路

二、实验流程

实现 hub 里缺失的 broadcast_packet 函数：

只需要实现遍历链表，判等，发送三个操作。

便利链表用 list.h 里的宏定义函数实现，遍历的接口链表是 instance->iface_list;

判等指判断两个端口的 index 属性相等

发送可利用写好的 iface_send_packet 用底层的 Socket 发送

```
iface_info_t *entry;
    struct list_head *list = &instance->iface_list;
    list_for_each_entry(entry, list, list) {
        if (entry->index != iface->index) {
            iface_send_packet(entry, packet, len);
        }
    }
```

三、实验结果和分析

- 1) 用 ping 验证广播网络的正常运行

```

"Node: h1"
root@lygPC:/network-lab/04-broadcast# 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=0.142 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.121 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.122 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.140 ms

--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/ndev = 0.121/0.131/0.142/0.012 ms
root@lygPC:/network-lab/04-broadcast#

"Node: h2"
root@lygPC:/network-lab/04-broadcast# 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=0.246 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.115 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.144 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.151 ms

--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3070ms
rtt min/avg/max/ndev = 0.115/0.164/0.246/0.049 ms
root@lygPC:/network-lab/04-broadcast#

"Node: h3"
root@lygPC:/network-lab/04-broadcast# 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=0.132 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=0.123 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.132 ms

--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3065ms
rtt min/avg/max/ndev = 0.123/0.127/0.132/0.012 ms
root@lygPC:/network-lab/04-broadcast#

"Node: b1"
root@lygPC:/network-lab/04-broadcast# ./hub
DEBUG: find the following interfaces: bi-eth0 bi-eth1 bi-eth2 .

```

截图展示了三台计算机 h1、h2、h3 连接一个交换机 b1 之后依次 ping 的结果。结果表明主机是单向连通的。又由于 ping 包含数据包的回传操作，因此主机之间是双向导通，验证了广播网络正常运行。

2) 用 iperf 验证广播网络的效率

```

"Node: b1"
root@lygPC:/network-lab/04-broadcast# ./hub-reference
DEBUG: find the following interfaces: bi-eth0 bi-eth1 bi-eth2.

"Node: h1"
root@lygPC:/network-lab/04-broadcast# iperf -s
Server listening on TCP port 5001
TCP window size: 65.3 KByte (default)
[ 14] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 55182
[ 15] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 55184

"Node: h2"
root@lygPC:/network-lab/04-broadcast# iperf -c 10.0.0.1 -t 10
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.2 port 55182 connected with 10.0.0.1 port 5001
[Croot@lygPC:/network-lab/04-broadcast# iperf -c 10.0.0.1 -t 1
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.2 port 55184 connected with 10.0.0.1 port 5001

```

很奇怪，即使用老师的 hub-reference，执行 iperf 也不能成功，表现为类似死循环的效果。

Disable_offloading.sh 是有可执行权限的。

Update: 在安装老师给出的 linux 内核和 header 之后，重新实验成功。

结果 1) h1 作服务器，h2 和 h3 同时连接 h1 的效果：

```
"Node: h3"
root@lywPC:~/network-lab/04-broadcast# 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 49854 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.2 sec  33.0 MBytes  9.17 Mbits/sec
root@lywPC:~/network-lab/04-broadcast#

"Node: h1"
root@lywPC:~/network-lab/04-broadcast# 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 10.0.0.3 port 49854
[ 15] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 39454
[ ID] Interval      Transfer    Bandwidth
[ 14] 0.0-30.4 sec  33.0 MBytes  9.11 Mbits/sec
[ 15] 0.0-30.7 sec  33.2 MBytes  9.09 Mbits/sec

"Node: h2"
root@lywPC:~/network-lab/04-broadcast# 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 10.0.0.2 port 39454 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.3 sec  33.2 MBytes  9.22 Mbits/sec
root@lywPC:~/network-lab/04-broadcast#
```

h1 和 h2 作服务器，h3 做客户端同时连接的效果

```
"Node: h1"
root@lywPC:~/network-lab/04-broadcast# 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 46008 connected with 10.0.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.2 sec  8.38 MBytes  2.32 Mbits/sec
root@lywPC:~/network-lab/04-broadcast#

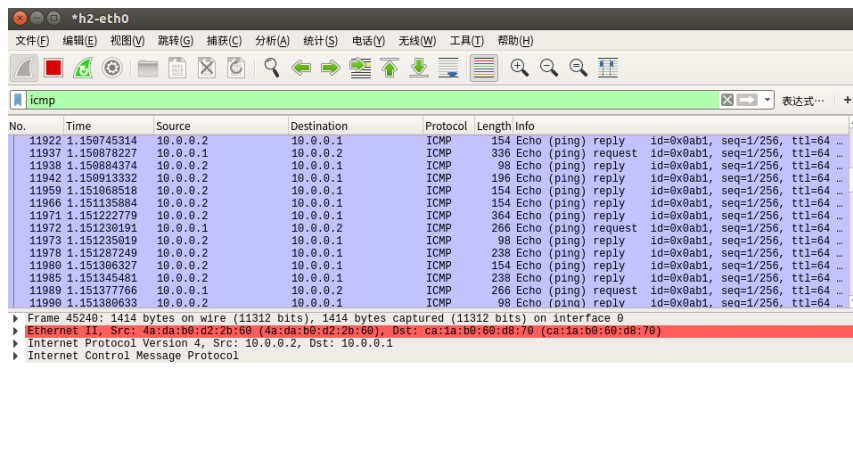
"Node: h1"
root@lywPC:~/network-lab/04-broadcast# iperf -c 10.0.0.3 -t 30
Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 13] local 10.0.0.1 port 48816 connected with 10.0.0.3 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 13] 0.0-30.2 sec  27.2 MBytes  7.58 Mbits/sec
root@lywPC:~/network-lab/04-broadcast#
```

分析：在前一种情况下，h2 和 h3 共同占用了 h1 到集线器的 20Mb/s 带宽，以及他们各自到集线器的 10Mb/s 带宽，因此发挥了最大效能，带宽评估接近 10Mb/s。在后一

种情况下，因为 h3 连接 h1 时，由于 b1 广播所有包，h2-b1 的带宽受影响，最终结果一个为 7Mb/s，一个为 2Mb/s，出现了竞争关系，而且带宽相加和近乎为单个 h2-b1 的带宽，猜想得到了证实。

3) 编写环形拓扑，验证环形广播出现数据包环路

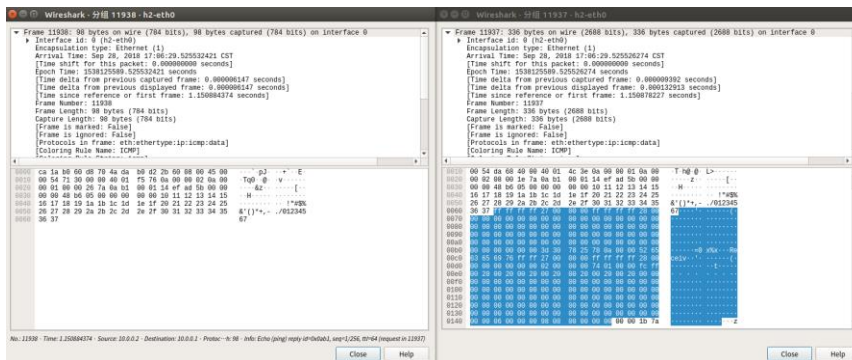
环形拓扑：三个 Hub 节点 b1, b2, b3 两两互连，两个主机节点 h1 连 b1, h2 连 b2 在 h1 中向 h2 发送一个数据包，在 wireshark 中抓包结果如下：



在 h2 中执行 ping 10.0.0.1 -c 2

只在 h1 中对 ping 发送的 ICMP 协议包进行跟踪，发现两点：

- 1) 这个过程并不是无限持续的，直到 45240 后转发结束。此后三个 hub 都不再收到转发消息
- 2) 转发时数据包的长度呈现递增的趋势（如下图）



经过老师进行分析和网上查找资料，了解到一个网络层帧长度的最大值 MTU 为 1514 字节（一说对应的 IP 包为 1500 字节），主要是为了提高发送和接收效率而做出的折衷妥协（大包的丢包率更高）

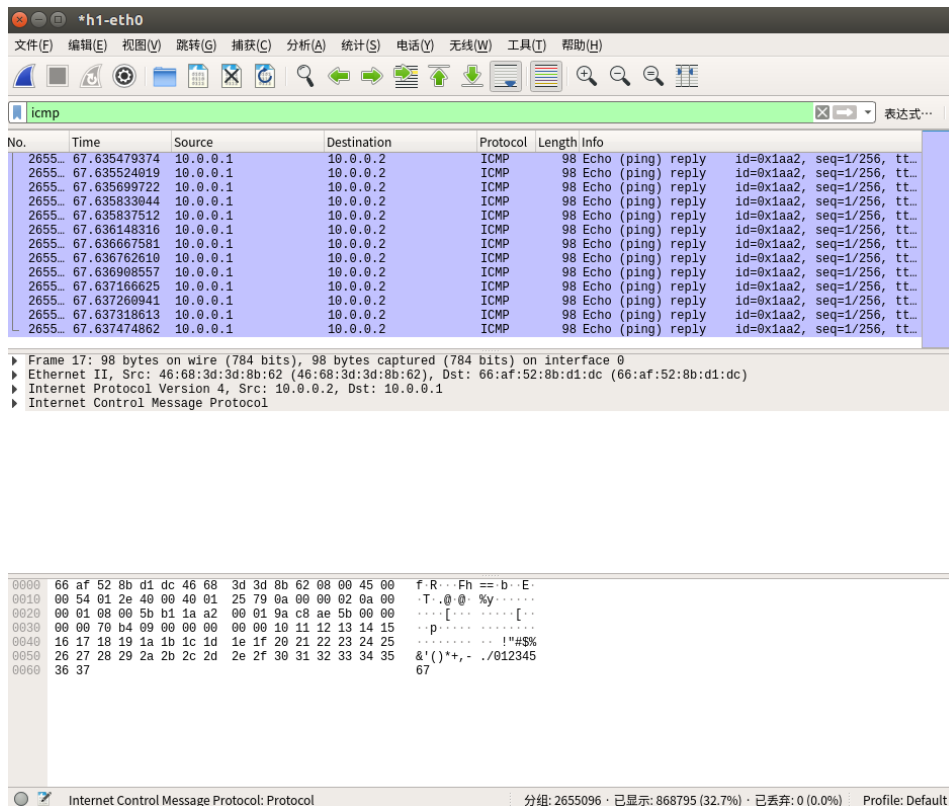
疑问：

- 对于数据包在转发过程中的增大，没有搜到合理的解释。

- 而且在发送大于 1500 字节的包的过程中，socket 并没有报错，也不知道原因。

期待老师和助教解答！

Update: 在采用老师的内核之后问题解决，数据包长度没有增加，所以数据包会一直在网络中转发，直到资源耗尽，用 wireshark 抓包截图如下（发包未终结）：



The screenshot shows a Wireshark capture on interface h1-eth0. The packet list displays a series of ICMP Echo (ping) replies from source 10.0.0.1 to destination 10.0.0.2. The packet details for the selected packet (No. 17) show the following structure:

Offset	Length	Protocol	Info
0000	66	af 52 8b d1 dc 68 3d 3d 8b 62 08 00 45 00	f.R...Fh ==:b...E
0010	00 54 01 2e 40 00 40 01 25 79 0a 00 00 02 0a 00	.T...@.%y.....	
0020	00 01 08 00 5b b1 1a a2 00 01 9a c8 ae 5b 00 00[...%y.....	
0030	00 00 70 b4 09 00 00 00 00 00 10 11 12 13 14 15	..p.....	
0040	16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25!""#\$%	
0050	26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35	&'()*+,-./012345	
0060	36 37	67	

The packet details pane shows the Internet Control Message Protocol (ICMP) structure, including the type (0), code (0), checksum, and sequence number (1/256).