

广播网络实验

中国科学院大学
袁欣怡 2018K8009929021
2021.4.6

实验内容

1. 实现结点广播 `broadcast_packet` 函数。
2. 使用 `three_nodes_bw.py` 验证三个节点相互能够ping通。
3. 用 `iperf` 测量广播网络的效率。
4. 构建环形拓扑结构，复现数据包环路现象。

实验流程

1. 搭建实验环境

本实验中涉及到的文件主要有：

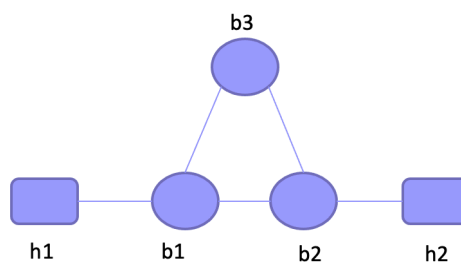
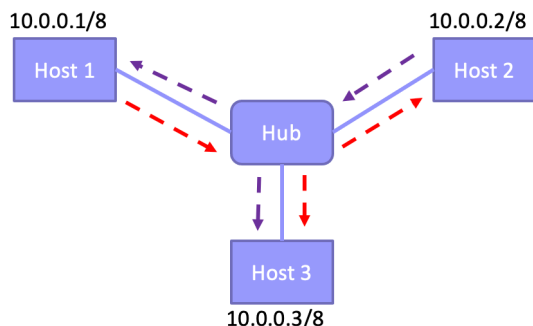
`main.c`：编译后生成 `hub`。调用了 `broadcast_packet()` 函数，这个函数在文件 `broadcast.c` 中实现。

`boradcast.c`：主要实现 `boradcast_packet()` 的功能。

`Makefile`：处理 `make clean` 和 `make all` 指令。

`three_nodes_bw.py`：三结点网络拓扑结构。如下图中左图所示。

`loop_topo.py`：自己实现的环形网络拓扑结构。如下图中右图所示。



2. 实验代码设计

`broadcast_packet` 函数:

```
1 void broadcast_packet(iface_info_t *iface, const char *packet, int len)
2 {
3     //lab04 TODO: broadcast packet
4     fprintf(stdout, "TODO: broadcast packet here.\n");
5     // 在屏幕进行打印, 告知包已经到达Hub, 可以进行广播
6
7     iface_info_t * iface_entry = NULL; // 初始化一个新变量
8
9     list_for_each_entry(iface_entry, &instance->iface_list, list) {
10     // 调用list_for_each_entry, 对整个链表进行遍历
11         if (iface_entry -> fd != iface -> fd) { // 当前主机不是发送消息的主机
12             iface_send_packet(iface_entry, packet, len); // 调用
13             // iface_send_packet, 发包给这个主机
14         }
15     }
```

`loop_topo.py` (节选) :

```
1 # 构建网络拓扑结构
2 class BroadcastTopo(Topo):
3     def build(self):
4         h1 = self.addHost('h1')
5         h2 = self.addHost('h2')
6
7         b1 = self.addHost('b1')
8         b2 = self.addHost('b2')
9         b3 = self.addHost('b3')
10
11         self.addLink(h1, b1)
12         self.addLink(h2, b2)
13         self.addLink(b1, b2)
14         self.addLink(b2, b3)
15         self.addLink(b1, b3)
16
17
18 # 设置IP地址
19 if __name__ == '__main__':
20     check_scripts()
21
22     topo = BroadcastTopo()
23     net = Mininet(topo = topo, link = TCLink, controller = None)
24
25     h1, h2, b1, b2, b3 = net.get('h1', 'h2', 'b1', 'b2', 'b3')
26     h1.cmd('ifconfig h1-eth0 10.0.0.1/8')
27     h2.cmd('ifconfig h2-eth0 10.0.0.2/8')
```

```

28
29 clearIP(b1)
30 clearIP(b2)
31 clearIP(b3)
32
33 for h in [ h1, h2, b1, b2, b3 ]:
34     h.cmd('./scripts/disable_offloading.sh') # 禁用offloading
35     h.cmd('./scripts/disable_ipv6.sh') # 禁用IPv6
36
37 net.start()
38 CLI(net)
39 net.stop()

```

3. 启动脚本进行测试

(1) 广播网络功能测试

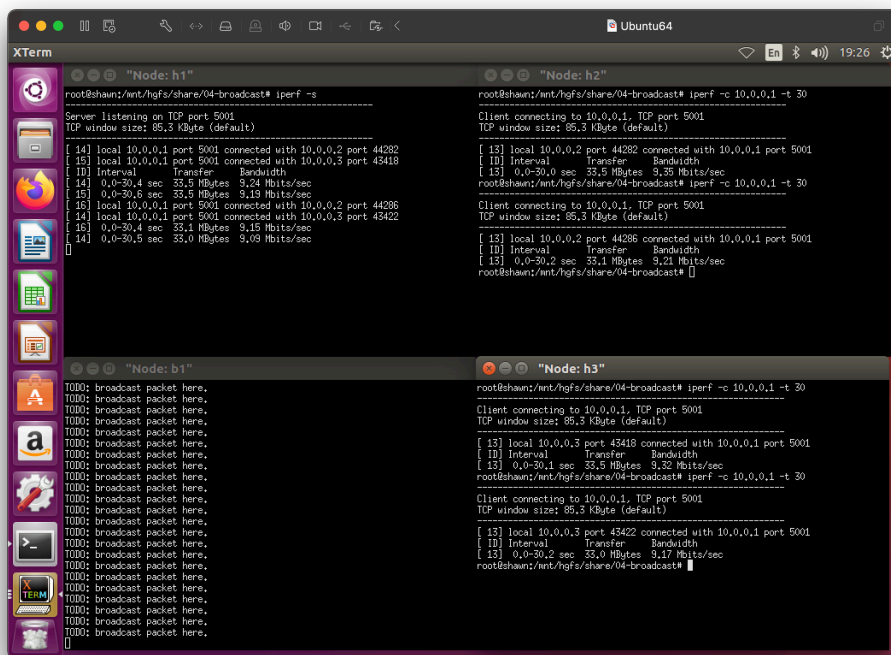
运行 `three_node_bw.py`，开启 `h1`、`h2`、`h3` 和 `b1` 四个结点。通过从 `h1` ping `h2` 和 `h3`，从 `h2` ping `h1` 和 `h3`，从 `h3` ping `h1` 和 `h2`，可以发现这些数据通路都是连通的，这说明我们实现的 `hub` 可以完成广播的功能。

The screenshot displays four terminal windows, each representing a different node in a network topology. The windows are titled "Node: h1", "Node: h2", "Node: h3", and "Node: b1". Each window shows the output of a series of ping commands. For example, in the "Node: h1" window, the user runs `ping 10.0.0.1 -c 4`, `ping 10.0.0.2 -c 4`, and `ping 10.0.0.3 -c 4`, all of which succeed with 0% packet loss. Similar results are shown for the other nodes, confirming that all nodes can reach each other via the network.

(2) 广播网络效率测试

运行 `three_node_bw.py`，开启 `h1`、`h2`、`h3` 和 `b1` 四个结点。

先用 `h1` 作为服务器，`h2` 和 `h3` 作为客户进行访问。



图中做了两组测试：

1. **h2** 先进行 **iperf** 测试，结束后再启动 **h3** 的测试。

测试结果：h2-h1：9.35 Mbps，h3-h1：9.32 Mbps。

2. **h2** 和 **h3** 同时测试。

测试结果：h2-h1：9.21 Mbps，h3-h1：9.17 Mbps。

可以看出来，是否并行测试对于带宽的影响很小，这说明了带宽的双向性，即：当 **h2** 与 **h3** 同时以 **10Mbps** 的速率向 **h1** 发送数据包时，**h2** 与 **b1**、**h3** 与 **b1** 之间的通路上都存在着双向的传输速率为 **10Mbps** 的数据流。理论分析结果与实际测量结果一致。

再用 **h2** 和 **h3** 作为服务器，**h1** 作为客户进行访问。此时需要打开两个 **h1** 的终端。

```
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.1 port 54750 connected with 10.0.0.2 port 5001
[ 10] Interval Transfer Bandwidth
[ 13] 0.0-30.2 sec 33.9 MBytes 9.42 Mbits/sec
root@shaun:/mnt/hgfs/share/04-broadcast# iperf -c 10.0.0.2 -t 30

Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.1 port 54754 connected with 10.0.0.2 port 5001
[ 10] Interval Transfer Bandwidth
[ 13] 0.0-30.6 sec 17.2 MBytes 4.72 Mbits/sec
root@shaun:/mnt/hgfs/share/04-broadcast# iperf -c 10.0.0.2 -t 30

Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.1 port 54758 connected with 10.0.0.2 port 5001
[ 10] Interval Transfer Bandwidth
[ 13] 0.0-30.3 sec 23.1 MBytes 6.40 Mbits/sec
root@shaun:/mnt/hgfs/share/04-broadcast#

Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.1 port 37536 connected with 10.0.0.3 port 5001
[ 10] Interval Transfer Bandwidth
[ 13] 0.0-30.1 sec 33.9 MBytes 9.45 Mbits/sec
root@shaun:/mnt/hgfs/share/04-broadcast# iperf -c 10.0.0.3 -t 30

Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.1 port 37540 connected with 10.0.0.3 port 5001
[ 10] Interval Transfer Bandwidth
[ 13] 0.0-30.2 sec 19.1 MBytes 5.32 Mbits/sec
root@shaun:/mnt/hgfs/share/04-broadcast# iperf -c 10.0.0.3 -t 30

Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 65.3 KByte (default)
[ 13] local 10.0.0.1 port 37544 connected with 10.0.0.3 port 5001
[ 10] Interval Transfer Bandwidth
[ 13] 0.0-30.1 sec 12.6 MBytes 3.52 Mbits/sec
root@shaun:/mnt/hgfs/share/04-broadcast#

Server listening on TCP port 5001
TCP window size: 65.3 KByte (default)
[ 14] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 54750
[ 10] Interval Transfer Bandwidth
[ 14] 0.0-30.3 sec 33.9 MBytes 9.37 Mbits/sec
[ 15] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 54754
[ 15] 0.0-31.1 sec 17.2 MBytes 4.65 Mbits/sec
[ 14] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 54758
[ 14] 0.0-30.7 sec 23.1 MBytes 6.33 Mbits/sec

Server listening on TCP port 5001
TCP window size: 65.3 KByte (default)
[ 14] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 37536
[ 10] Interval Transfer Bandwidth
[ 14] 0.0-30.3 sec 33.9 MBytes 9.37 Mbits/sec
[ 15] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 37540
[ 15] 0.0-30.2 sec 19.1 MBytes 5.31 Mbits/sec
[ 14] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 37544
[ 14] 0.0-30.1 sec 12.6 MBytes 3.51 Mbits/sec
```

同样进行了多组测试：

1. **h1** 先访问 **h2** 进行测试，结束后再访问 **h3** 进行测试。

测试结果：**h1-h2**: 9.37 Mbps, **h1-h3**: 9.37 Mbps。

2. **h1** 同时访问 **h2** 和 **h3**。

测试结果：**h1-h2**: 4.65 Mbps, **h1-h3**: 5.31 Mbps。

3. (同上) **h1** 同时访问 **h2** 和 **h3**。

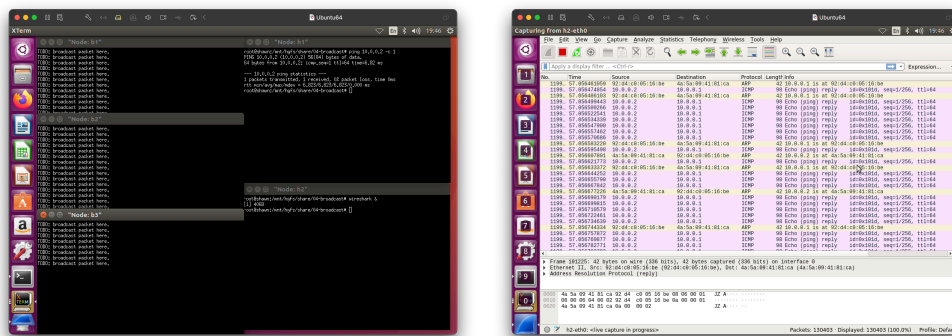
测试结果：**h1-h2**: 6.33 Mbps, **h1-h3**: 3.51 Mbps。

此时发现并发对带宽的影响较大，这说明广播转发会占用带宽，导致传输速度降低。单发时理想传输速度为 **10Mbps**，实验测出来的结果也很接近这个值。但是并发时，**h1** 向 **b1** 以 **20Mbps** 的速率发送数据包，其中一半目的主机是 **h2**，另一半目的主机是 **h3**。数据包到达 **b1** 后，开始向 **h2** 和 **h3** 转发。每一个数据包，都会被复制后发往 **h2** 和 **h3**，所以对于 **h2**，虽然其接受速率理论上最高为 **10Mbps**，但是其中约有一半是 **h1** 要发往 **h3** 的包，对于 **h2** 而言属于无效包，白白占用带宽。对于 **h3** 也是类似的情况。实际测试中，第一次 **h1-h2**: 4.65 Mbps, **h1-h3**: 5.31 Mbps，二者相加为 **9.96Mbps** < 10Mbps，第二次 **h1-h2**: 6.33 Mbps, **h1-h3**: 3.51 Mbps，二者相加为 **9.84Mbps** < 10Mbps，均满足理论分析。

(3) 环形拓扑中出现数据包环路

运行 **loop_topo.py**，开启 **h1**、**h2**、**b1**、**b2** 和 **b3** 五个结点。

让 **b1**、**b2** 和 **b3** 作为hub，在 **h2** 中打开 **wireshark**，然后用 **h1** 发送一个数据包。



当 **h1** 发送数据包后，**b1**、**b2** 和 **b3** 都在不停打印广播的信息，同时 **wireshark** 中也一直在抓包，说明数据包在环形拓扑中被不停地转发，对资源造成了极大的浪费。

实验总结

无。

参考资料

1. HUB（多端口转发器）：<https://baike.baidu.com/item/HUB/703984>