广播网络实验

孙佳钰 2015K8009929051

2018 年 9 月 27 日

1 实验内容

- 实现节点广播：实现 main.c 中的 *broadcast_packet* 函数，将收到的数据包从其余所有端口转发出去，使网络中任意一个节点能 ping 通其余节点。
- 验证广播网络的效率：利用 *iperf* 工具测量上面实现的广播网络的带宽。
- 验证环形拓扑下节点广播会产生数据包环路：先定义一个网络拓扑，其中有三个 hub 节点和两个主机节点，三个 hub 节点两两互连，两个主机节点 h1 连到 b1，h2 连到 b2。在 h1 中 ping h2，通过 wireshark 进行抓包，观察数据包被不断转发的现象。

2 实验流程

1. 网络拓扑已实现好，所有端口都存储在 $instance->iface\_list$ 中，因此只需通过宏定义中的 $list\_for\_each\_entry(pos, head, member)$ 找出所有其它端口，然后通过 $iface\_send\_packet$ 函数将数据包发送出去即可。实现的 *broadcast_packet* 函数如下

```
void broadcast_packet(iface_info_t *iface, const char *packet, int len)
{
    iface_info_t *iface_t = NULL;
    list_for_each_entry(iface_t, &instance->iface_list, list)
        if (iface_t->fd != iface->fd)
            iface_send_packet(iface_t, packet, len);
}
```

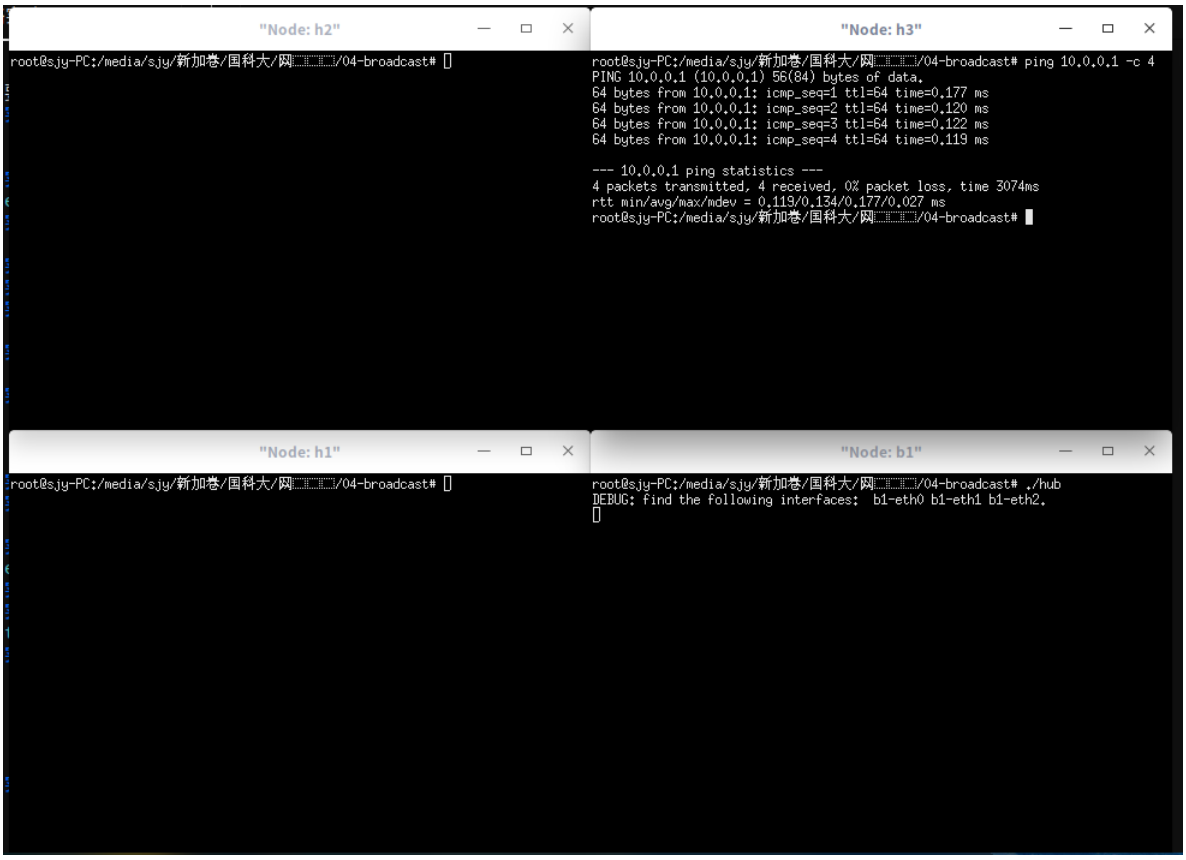
2.创建环形网络的拓扑脚本部分如下

```
class BroadcastTopo(Topo):
    def build(self):
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
```

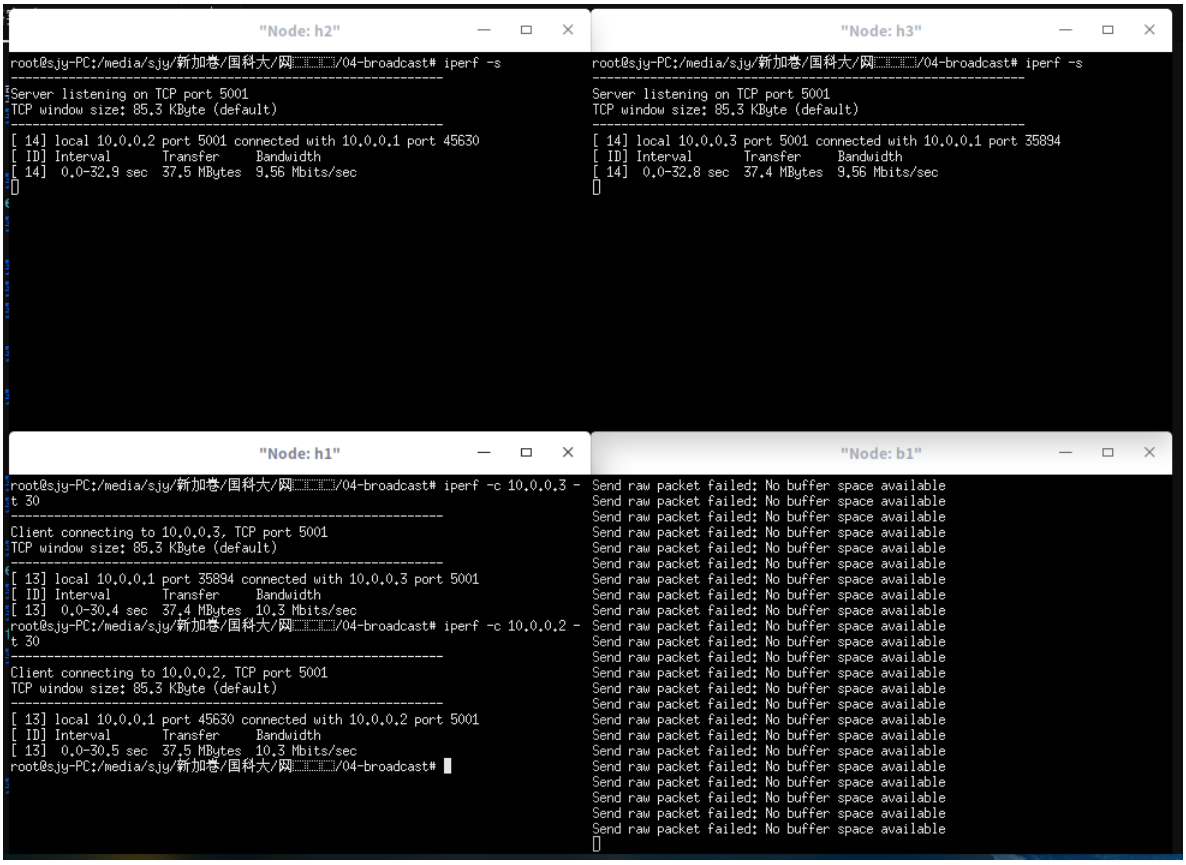
```
        b1 = self.addHost('b1')
        b2 = self.addHost('b2')
        b3 = self.addHost('b3')

        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(b1, b3, bw=10)

if __name__ == '__main__':
    topo = BroadcastTopo()
    net = Mininet(topo = topo, link = TCLink, controller = None)

    h1, h2, b1, b2, b3 = net.get('h1', 'h2', 'b1', 'b2', 'b3')
    h1.cmd('ifconfig h1-eth0 10.0.0.1/8')
    h2.cmd('ifconfig h2-eth0 10.0.0.2/8')
    clearIP(b1)
    clearIP(b2)
    clearIP(b3)

    h1.cmd('./disable_offloading.sh')
    h2.cmd('./disable_offloading.sh')

    net.start()
    CLI(net)
    net.stop()
```

完整实验代码可见所附文件。

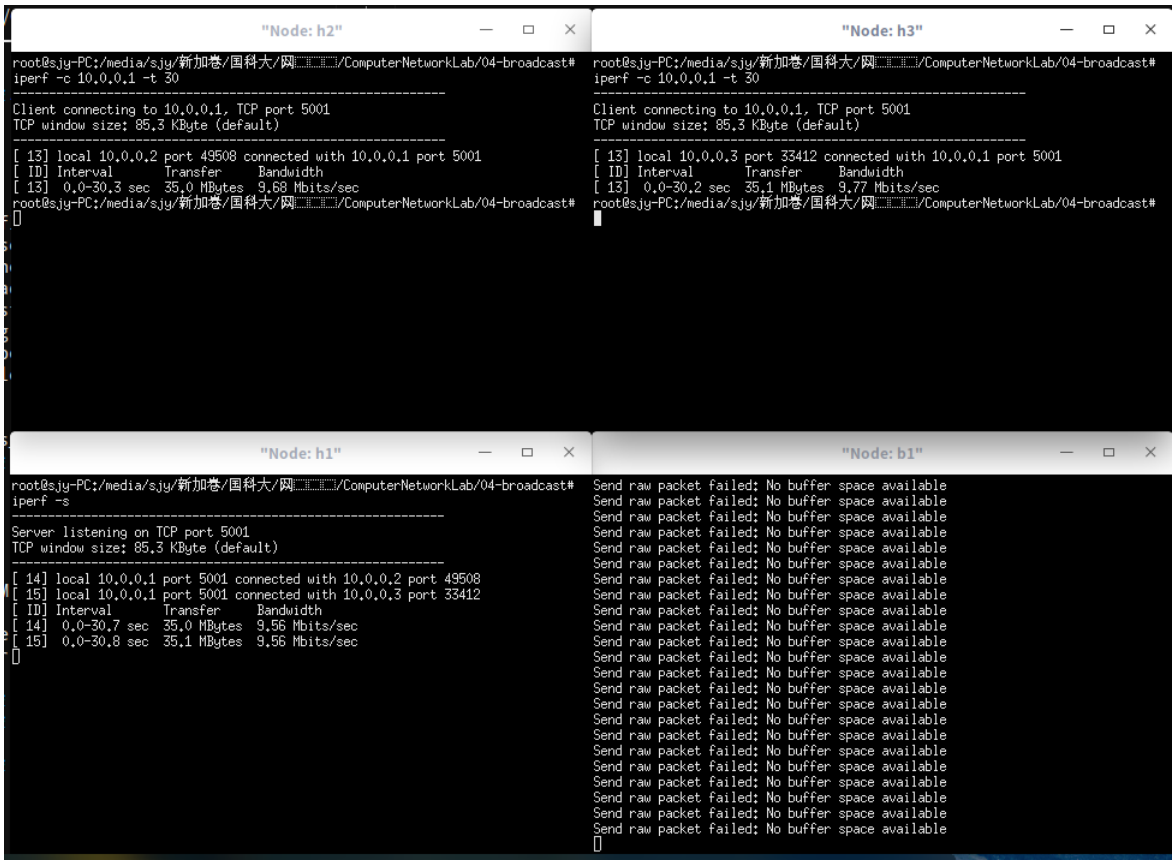
3 实验结果及分析



其中 b1 运行 hub。在 h3 中 ping h1，发现可以 ping 通。



h1 作为 client，h2 和 h3 作为 server。



h1 作为 server，h2 和 h3 作为 client。

No.	Time	Source	Destination	Protocol	Length	Info
77103	6.237265292	02:e6:9b:92:a1:2f	ae:11:d6:50:50:56	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
77104	7.261266684	02:e6:9b:92:a1:2f	ae:11:d6:50:50:56	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
14	0.000180047	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x6d0d, seq=1/256, ttl=64 (reply in 15)
15	0.000194011	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64 (request in 14)
16	0.000207924	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x6d0d, seq=1/256, ttl=64 (reply in 17)
17	0.000212515	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64 (request in 16)
30	0.000395265	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x6d0d, seq=1/256, ttl=64 (reply in 31)
31	0.000402281	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64 (request in 30)
32	0.000414748	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x6d0d, seq=1/256, ttl=64 (reply in 33)
33	0.000418488	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64 (request in 32)
39	0.000510843	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
40	0.000520802	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
43	0.000566598	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
44	0.000576699	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
54	0.000742390	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x6d0d, seq=1/256, ttl=64 (reply in 55)
55	0.000750201	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64 (request in 54)
56	0.000762208	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x6d0d, seq=1/256, ttl=64 (reply in 57)
57	0.000767043	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64 (request in 56)
62	0.000878911	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
63	0.000888851	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
67	0.000941231	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
68	0.001019122	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64
69	0.001097397	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x6d0d, seq=1/256, ttl=64

可以看出数据包在不断被转发。