中國科學院大學

实验报告

广播网络实验

课程名称:	计算机网络
学院:	光电学院
专业:	机械制造及自动化
姓名:	刘源
学号:	201828013919034

2019 年 7 月 8 日

中國科學院大學 实验报告

专业：机械制造及自动化
姓名：刘源
学号：201828013919034
日期：2019 年 7 月 8 日

课程名称：计算机网络　　实验名称：广播网络实验　　指导老师：谢高岗

一、 实验要求

本实验要求学生在已有代码基础上，实现节点广播的 broadcast_packet 函数，验证广播网络能够正常运行，验证广播网络的效率，验证环形拓扑下会产生数据报回路。

二、 实验内容和步骤

1. 实现节点广播的 broadcast_packet 函数

2. 验证广播网络正常运行

(1) 运行网络拓扑 (three_nodes_bw.py)

(2) 在 b1 交换机节点上编译运行 hub 程序

(3) 从 h1(10.0.0.1) Ping h2(10.0.0.2) 和 h3(10.0.0.3)，能够 ping 通

(4) 从 h2(10.0.0.2) Ping h1(10.0.0.1) 和 h3(10.0.0.3)，能够 ping 通

(5) 从 h3(10.0.0.3) Ping h1(10.0.0.1) 和 h2(10.0.0.2)，能够 ping 通

3. 验证广播网络的效率

(1) 运行网络拓扑 (three_nodes_bw.py)

(2) 在 b1 上运行 hub 程序

(3) 在 h2 和 h3 上执行 iperf -s 命令

(4) 在 h1 上依次执行 iperf -c 10.0.0.2 -t 30 和 iperf -c 10.0.0.3 -t 30 命令，得到实验结果

(5) 在 h1 上执行 iperf -s 命令

(6) 在 h2 和 h3 上分别执行 iperf -c 10.0.0.1 -t 30 命令，得到实验结果

4. 验证环形拓扑下会产生回路

(1) 运行网络拓扑 (ring_topo.py)

(2) 在 b1、b2、b3 上运行 hub 程序

(3) 在 h2(10.0.0.2) 上运行 wireshark 软件并监听数据报

(4) 从 h1(10.0.0.1) Ping h2(10.0.0.2)，分析 wireshark 数据，得到实验结果

三、 主要仪器设备

计算机，Mininet 软件，Wireshark 软件

四、 实验过程

1. 安装 mininet 与 wireshark 软件

在 Ubuntu 下输入 sudo apt install mininet 与 sudo apt install build-essential xterm wireshark ethtool iperf traceroute iptables arptables 命令进行软件安装，安装完成后，运行 sudo mn 验证 mininet 是否正确安装，验证结果如图 1 所示。

```
root@ubuntu:~# sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

图 1: mininet 安装成功

2. 编写 broadcast_packet 函数

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

3. 依步骤完成实验内容

五、 实验结果与分析

1. 验证广播网络正常运行

(1) 运行 broadcast_packet.py 生成的网络如图 2 所示。

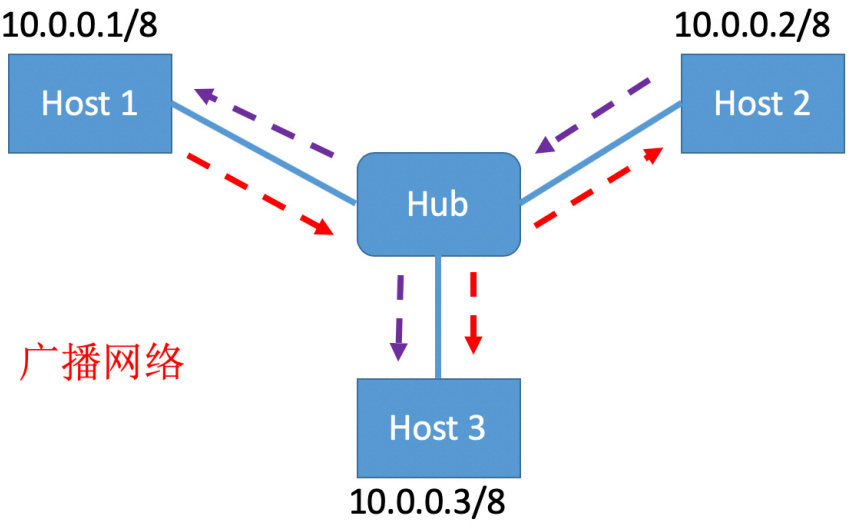


图 2: broadcast_packet.py 生成的网络

(2) 编译并在 b1 上运行 hub 程序，程序运行结果如图 3所示。

```
root@ubuntu:/home/091M4002HBP2# make all
gcc -c -g -Wall -Iinclude main.c -o main.o
gcc  main.o -o hub
root@ubuntu:/home/091M4002HBP2# ./hub
DEBUG: find the following interfaces:  b1-eth0 b1-eth1 b1-eth2.
```

图 3: hub 程序运行结果

(3) 从 h1(10.0.0.1) Ping h2(10.0.0.2) 和 h3(10.0.0.3)，结果如图 4所示，能够 Ping 通。

```
root@ubuntu:/home/091M4002HBP2# ping 10.0.0.2 -c 1
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=0.313 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.313/0.313/0.313/0.000 ms
root@ubuntu:/home/091M4002HBP2# ping 10.0.0.3 -c 1
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=0.163 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.163/0.163/0.163/0.000 ms
```

图 4: 从 h1 Ping h2 和 h3 结果

(4) 从 h2(10.0.0.2) Ping h1(10.0.0.1) 和 h3(10.0.0.3)，结果如图 5所示，能够 ping 通。

(5) 从 h3(10.0.0.3) Ping h1(10.0.0.1) 和 h2(10.0.0.2)，结果如图 6所示，能够 ping 通。

(6) 广播网络各节点能相互 ping 通，证明 broadcast_packet 函数能够完成实验所需功能，实验成功。

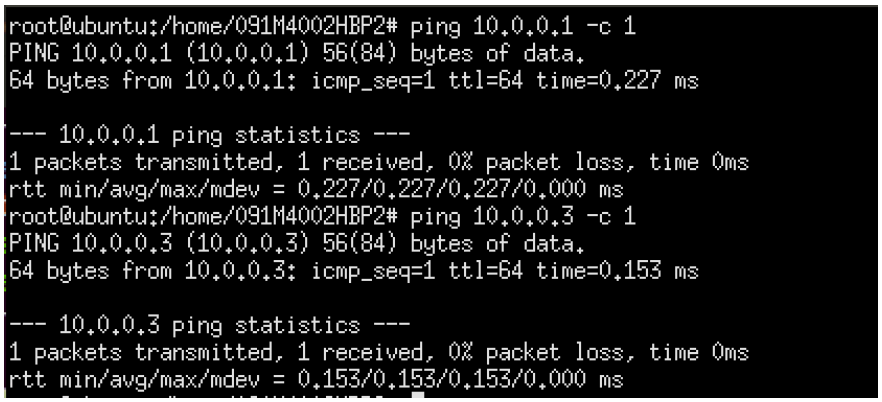


图 5: 从 h2 Ping h1 和 h3 结果

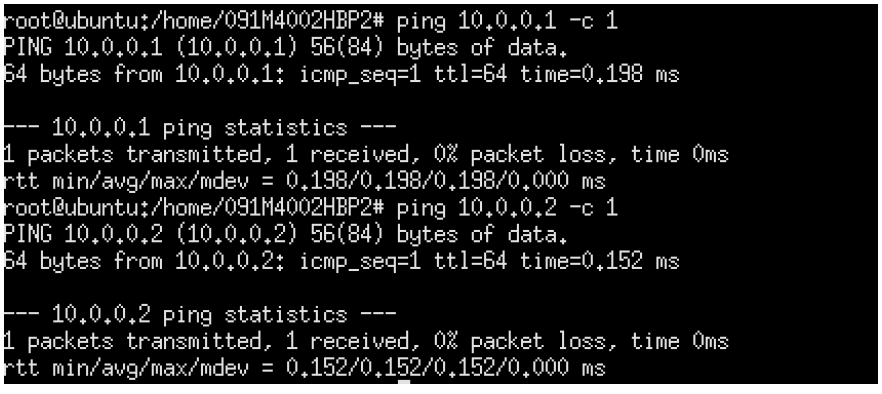


图 6: 从 h3 Ping h1 和 h2 结果

2. 验证广播网络的效率

(1) 运行 broadcast_packet.py

(2) 在 b1 节点上运行 hub 程序

(3) 依次在 h2 和 h3 上执行 iperf -s 命令，(4)执行后 h2 iperf 程序结果如图 7所示。

```
root@ubuntu:/home/091M4002HBP2# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 14] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 35392
[ ID] Interval       Transfer     Bandwidth
[ 14]  0.0-30.6 sec  34.1 MBytes  9.36 Mbits/sec
```

图 7: h2 和 h3 执行 iperf server 测试结果

(4) 在 h1 上依次执行 iperf -c 10.0.0.2 -t 30 和 iperf -c 10.0.0.3 -t 30 命令，结果如图 8所示。

```
root@ubuntu:/home/091M4002HBP2# 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 50706 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-30.1 sec  34.1 MBytes  9.52 Mbits/sec
root@ubuntu:/home/091M4002HBP2# 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 35392 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-30.1 sec  34.1 MBytes  9.50 Mbits/sec
```

图 8: h1 执行 iperf client 测试结果

(5) 在 h1 上执行 iperf -s 命令，(6)执行后 h1 iperf 程序结果如图 9所示。

```
root@ubuntu:/home/091M4002HBP2# 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 34196
[ ID] Interval       Transfer     Bandwidth
[ 14]  0.0-31.1 sec  34.4 MBytes  9.27 Mbits/sec
[ 14] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 56524
[ 14]  0.0-30.8 sec  34.4 MBytes  9.35 Mbits/sec
```

图 9: h1 执行 iperf server 测试结果

(6) 在 h2 和 h3 上分别执行 iperf -c 10.0.0.1 -t 30 和 iperf -c 10.0.0.1 -t 30 命令，h2 的运行结果如图 10所示，h3 的运行结果如图 11所示。

```
root@ubuntu:/home/091M4002HBP2# 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 56524 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-30.2 sec  34.4 MBytes  9.54 Mbits/sec
```

图 10: h2 执行 iperf client 测试结果

```
root@ubuntu:/home/091M4002HBP2# 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 34196 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-30.3 sec  34.4 MBytes  9.52 Mbits/sec
```

图 11: h3 执行 iperf client 测试结果

(7) 以上实验结果表明，h2 与 hub 以及 h3 与 hub 之间的链路带宽和脚本中所设置的 10Mb/s 基本一致，实验成功。

3. 验证环形拓扑下会产生回路

(1) 运行网络拓扑 (ring_topo.py)，产生的网络结构如图 12所示。

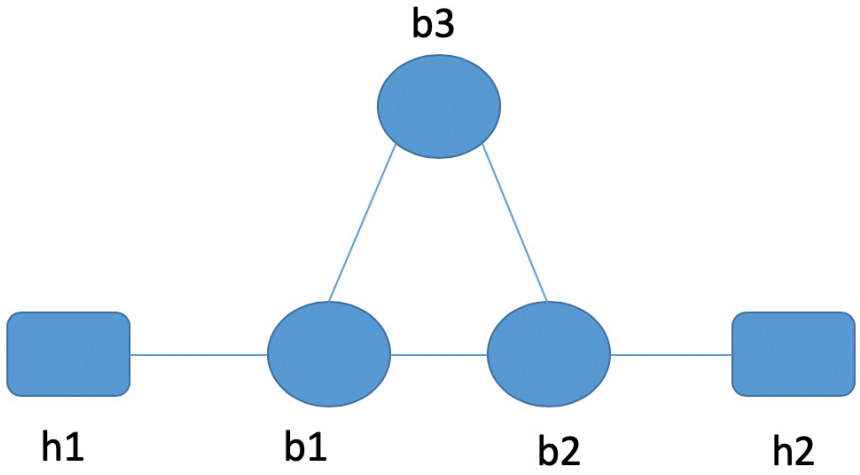
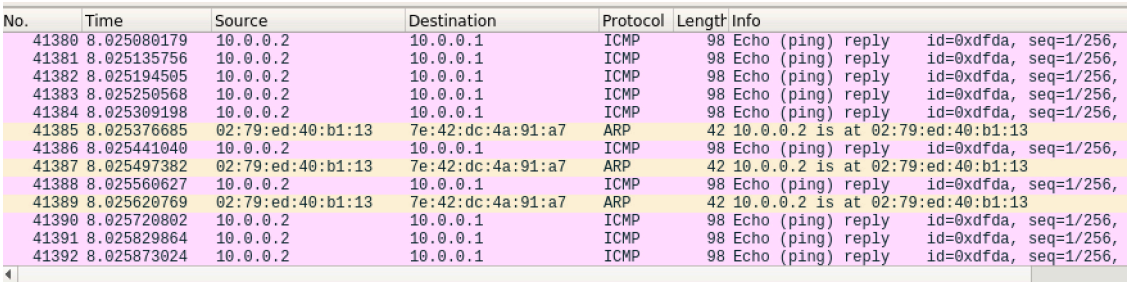


图 12: ring_topo.py 运行结果

(2) 在 b1、b2、b3 上运行 hub 程序。

(3) 在 h2(10.0.0.2) 上运行 wireshark 软件并监听数据报，在(4)执行后结果如图 13所示。

(4) 从 h1(10.0.0.1) Ping h2(10.0.0.2)，分析 wireshark 数据，由图 13可知，环路中有数据包在不断的转发，实验成功。



No.	Time	Source	Destination	Protocol	Length	Info
41380	8.025080179	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41381	8.025135756	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41382	8.025194505	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41383	8.025250568	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41384	8.025309198	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41385	8.025376685	02:79:ed:40:b1:13	7e:42:dc:4a:91:a7	ARP	42	10.0.0.2 is at 02:79:ed:40:b1:13
41386	8.025441040	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41387	8.025497382	02:79:ed:40:b1:13	7e:42:dc:4a:91:a7	ARP	42	10.0.0.2 is at 02:79:ed:40:b1:13
41388	8.025560627	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41389	8.025620769	02:79:ed:40:b1:13	7e:42:dc:4a:91:a7	ARP	42	10.0.0.2 is at 02:79:ed:40:b1:13
41390	8.025720802	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41391	8.025829864	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,
41392	8.025873024	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0xdfda, seq=1/256,

图 13: h2 上 wireshark 程序抓包的部分结果