广播网络实验报告

张翔雨 2018K8009929035

一、实验题目：广播网络实验

二、实验内容

- 实现节点广播的broadcast_packet函数
- 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
- 验证广播网络的效率
    - 在three_nodes_bw.py进行iperf测量
    - 两种场景：
        - H1: iperf client; H2, H3: servers （h1同时向h2和h3测量）
        - H1: iperf server; H2, H3: clients （ h2和h3 同时向h1测量）
- 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

三、实验过程

1.实现节点广播

（1）实现broadcast_packet函数

节点广播的实现逻辑为：对于节点的所有网络端口进行遍历，只要不是发送该数据包的端口，便向其发送数据包。代码实现如下，这里用到了宏定义中的list_for_each_entry函数，作用是遍历链表，其中第三个参数`list`代表结构体的成员名称，而`instance -> iface_list`保存了所有网络端口的信息。因此遍历时，只需要检查端口号是否相同，不同则发送数据包。

```
void broadcast_packet(iface_info_t *iface, const char *packet, int len)
{
    // TODO: broadcast packet
    fprintf(stdout, "TODO: broadcast packet.\n");
    iface_info_t *entry;
    list_for_each_entry(entry,&instance -> iface_list,list)
    {
        if(entry->index != iface->index)
            iface_send_packet(entry,packet,len);
    }
}
```

（2）验证结果

在可执行文件所在目录下执行 `three_nodes_bw.py` 脚本启动mininet，在mininet中启动h1,h2,h3,b1四个终端，在b1中执行 `./hub` 启动hub，在其他三个节点运行ping验证是否可以相互ping通。

2.测量广播网络传输效率

在可执行文件所在目录下执行 `three_nodes_bw.py` 脚本启动mininet，在mininet中启动h1,h2,h3,b1四个终端，在b1中执行 `./hub` 启动hub，分别将h1作为服务器和客户端运行iperf测量网络实际带宽。

3.实现数据包在环路中不断广播

实现环形网络拓扑，包括三个Hub节点，b1, b2, b3，两两互联，两个主机节点，h1连接到b1，h2连接到b2。需要修改脚本文件实现，命名为 `circle_network.py`,需要修改BroadcastTopo类和主函数，带宽统一设置为20Mbits/s

```
class BroadcastTopo(Topo):
    def build(self):
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        b2 = self.addHost('b2')
        b1 = self.addHost('b1')
        b3 = self.addHost('b3')
        self.addLink(h1, b1, bw=20)
        self.addLink(b3, b1, bw=20)
        self.addLink(b2, b1, bw=20)
        self.addLink(h2, b2, bw=20)
        self.addLink(b2, b3, bw=20)

if __name__ == '__main__':
```

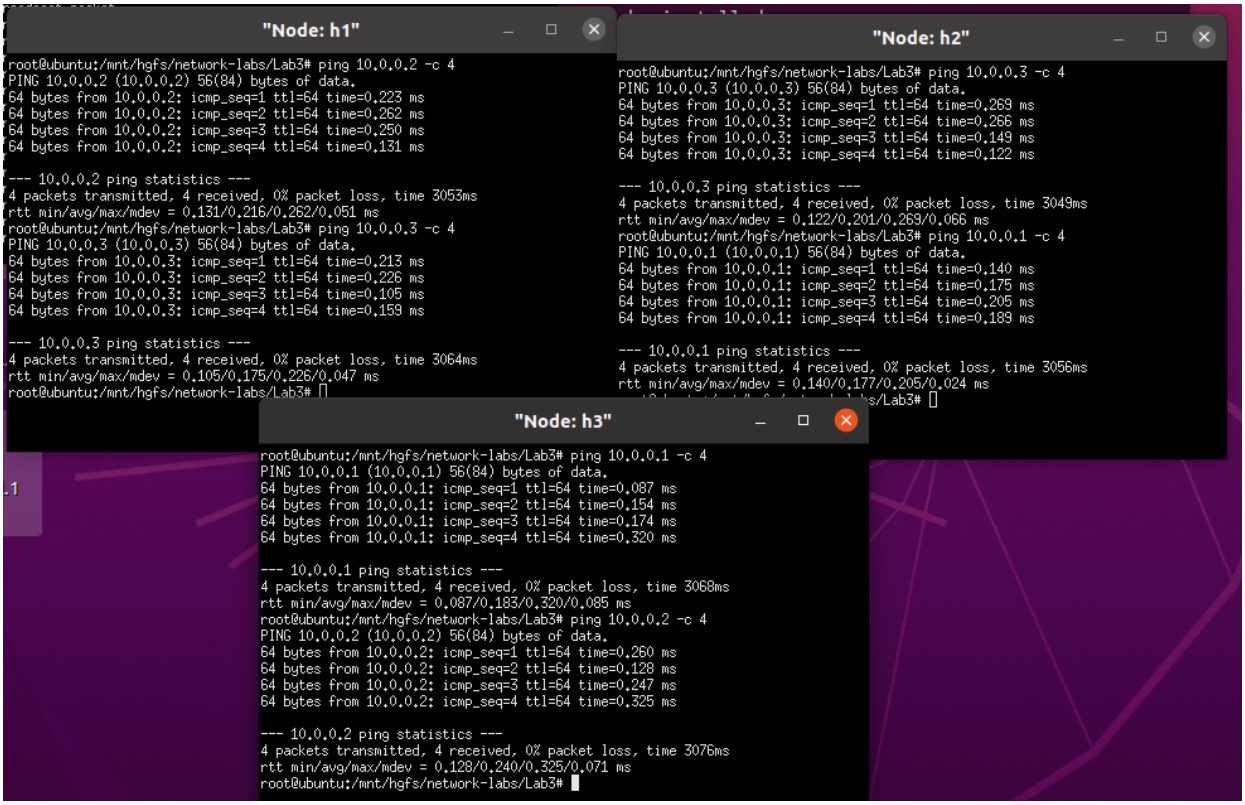
```
    check_scripts()

    topo = BroadcastTopo()
    net = Mininet(topo = topo, link = TCLink, controller = None)

    h1, h2, b2, b1, b3 = net.get('h1', 'h2', 'b2', 'b1', '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)

    for h in [ h1, h2, b2, b1, b3 ]:
        h.cmd('./scripts/disable_offloading.sh')
        h.cmd('./scripts/disable_ipv6.sh')

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

在可执行文件所在目录下执行 `circle_network.py` 脚本启动mininet，在mininet中启动h1,h2,b2,b1,b3五个终端，在b1,b2,b3中执行 `./hub` 启动hub，用h1向h2发包，在h2中启动wireshark查看收到的包。

四、实验结果

1.节点广播网络连通性验证

分别在h1，h2，h3三个节点中ping其他两个节点，结果如下，证明节点广播网络连通性正常。

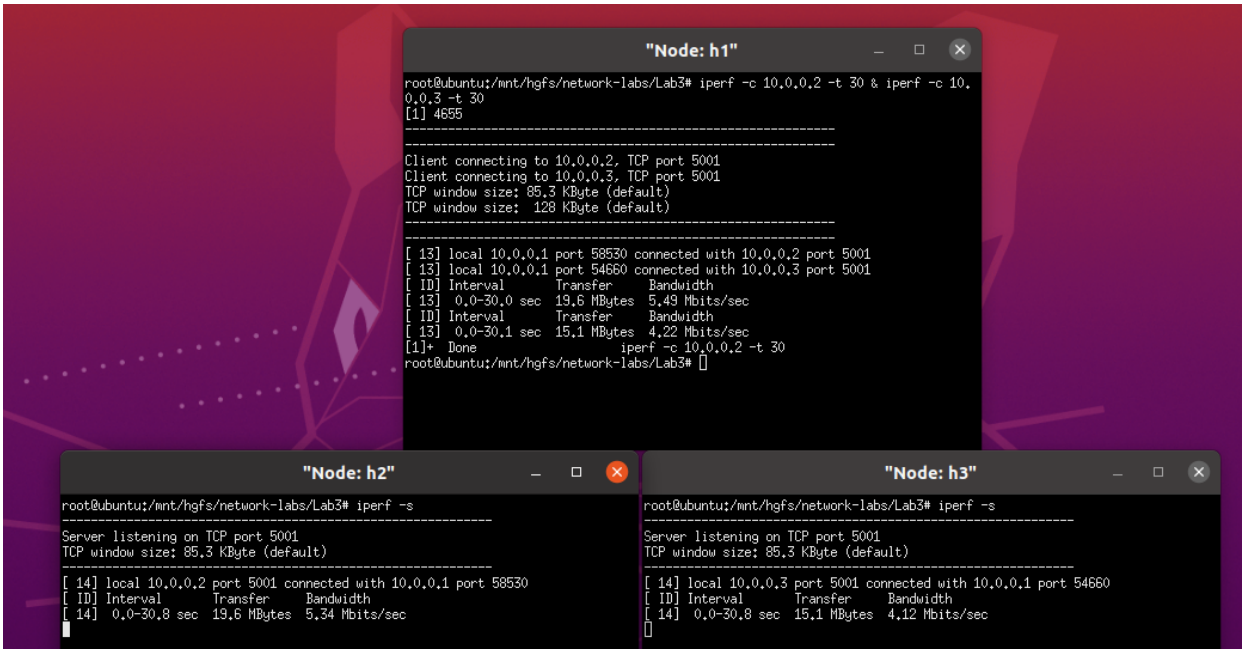


2.广播网络传输效率测量

(1)h1为客户端，h2，h3为服务器

	h1→h2	h1→h3		h2	h3
发送带宽(Mbits/s)	5.49	4.22	接受带宽(Mbits/s)	5.34	4.12
实际带宽(Mbits/s)	20	20	实际带宽(Mbits/s)	10	10
利用率	47.925%				

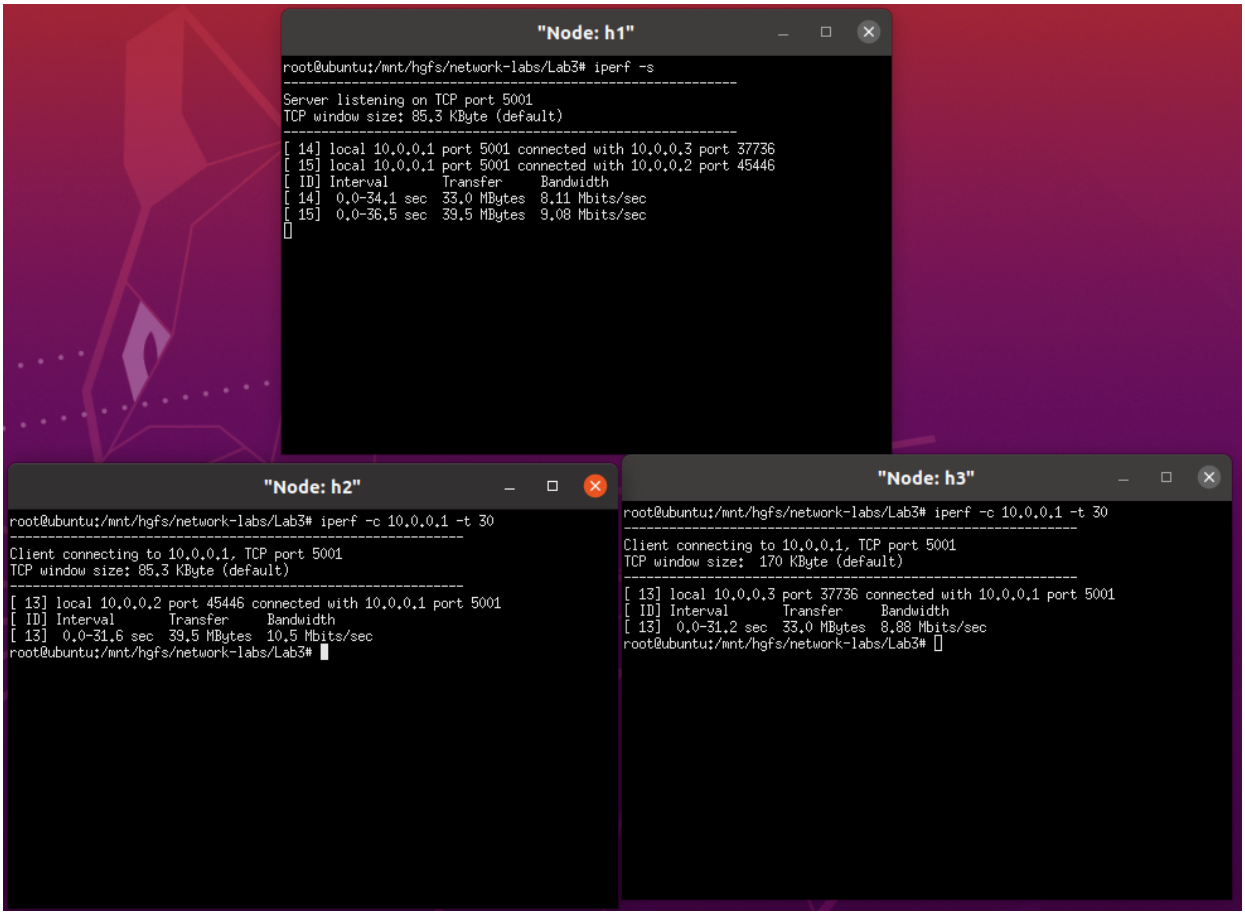
测试截图



(2)h2，h3为客户端，h1为服务器

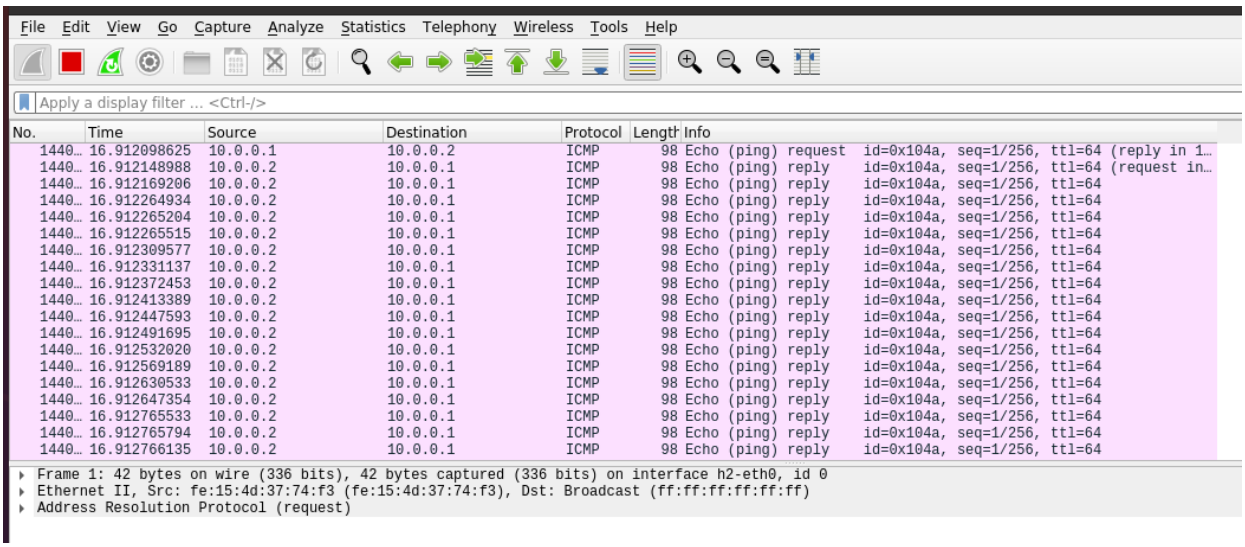
	h2→h1	h3→h1		h2	h3
接受带宽(Mbits/s)	9.08	8.11	发送带宽(Mbits/s)	10.5	8.88
实际带宽(Mbits/s)	20	20	实际带宽(Mbits/s)	10	10
利用率	91.425%				

测试截图



3.环形网络拓扑实现

启动脚本，启动3个hub，在h2中启动wireshark，用h1向h2发一个包，抓包结果如下：



可以看到ping的请求包和回复包在不停地被转发，同时发现请求包的数量远小于回复包，环形网络实现成功。

五、结果分析

1.网络传输效率测试

经过测试发现由h1向h2，h3发包时的网络带宽约为h2，h3向h1发包时的一半，即为h2，h3到b1的理论带宽的一半。

经过分析猜测原因为，当h1向h2发包时需要经过b1节点，而b1节点会同时向h3广播，h1向h3发送时同理。因此在h1同时向h1,h2发包时相当于向这两个节点每个节点发了两个包，因此一个包的速度约为理论接受带宽的一半。

而h2，h3向h1发包时，因为h1到b1的带宽为20Mbits/s，是h2，h3的两倍，因此当h2，h3同时向h1发包时，正好占满h1带宽，因此每个包的实际带宽约和h2，h3带宽相同。

2.环形网络拓扑

当h1 向h2发包时，会先从h1 发向b1。而当b1 接受到包时，会同时向b2 和b3进行发送。发送向b2 的包会发送到h2，与此同时也会发送给b3。发送向b3 的包也会在环状结构中进行广播，数据包因此会在环状网络中不停传输。

六、实验总结

本次实验较上次实验来说难度降低了不少，需要实现的代码量也较少，但是通过这次实验我加深了对广播网络的理解。同时对于实验用到的脚本以及相关的链表数据结构的使用也清晰了不少。