Lab2: Play with Mininet

1 问题1

1.1 实验步骤

1. 描述网络结构.

```
class TripleSwitchTopo( Topo ):
   def build(self):
         switchs = dict([(f"s{switch_id}",self.addSwitch(f"s{switch_id}")) for switch_id in irange(1,3)])
         hosts = dict([(f"h{host_id}",self.addHost(f"h{host_id}")) for host_id in irange(1,3)])

         self.addLink("h1", "s1", use_htb=True)
         self.addLink("s1", "s2", bw=10, use_htb=True)
         self.addLink("s2", "h2", use_htb=True)
         self.addLink("s1", "s3", bw=10, use_htb=True)
         self.addLink("s3", "h3", use_htb=True)
```

2. 构建网络时。将连接类型设置为`TCLink`，此设置可以使带宽限制生效。

```
topo = TripleSwitchTopo()
net = Mininet( topo=topo,
               host=Host, link=TCLink,
               autoStaticArp=True )
net.start()
```

3. 添加网络连接性与连接速度测试。

```
net.pingAll()
h1, h2, h3 = net.getNodeByName('h1', 'h2', 'h3')
info( "Testing bandwidth between h1 and h2\n" )
net.iperf(( h1, h2 ))
info( "Testing bandwidth between h1 and h3\n" )
net.iperf(( h1, h3 ))
info( "Testing bandwidth between h2 and h3\n" )
net.iperf(( h2, h3 ))
```

4. 运行代码，完整实验代码见`hw1.py`。

1.2 运行结果

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (10.00Mbit) (10.00Mbit) (s1, s2) (10.00Mbit) (10.00Mbit) (s1, s3) (s2, h2) (s3, h3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit)
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth2
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.56 Mbits/sec', '10.2 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['9.57 Mbits/sec', '9.81 Mbits/sec']
*** Iperf: testing TCP bandwidth between h2 and h3
*** Results: ['9.57 Mbits/sec', '10.0 Mbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 5 links
.....
*** Stopping 3 switches
s1 s2 s3
*** Stopping 3 hosts
h1 h2 h3
*** Done
```

1.3 结果分析

从图中可以看出，任意两个服务器之间的连接速度均大约为10Mbps，符合预期。

值得注意的是，`h1-h2`与`h2-h3`的测试结果中，客户端的测试结果大于10Mbps。从`mininet/net.py`第810行与`mininet/test/test_hifi.py`第150行的注释，我们可以大致猜出原因：iperf客户端作为主动发送的一端，并不会检查TCP的`ACK`回报，iperf显示的速度是根据其向运输层接口输出的数据量计算出的，由于TCP缓存的存在，测试结束时，应用层发送的数据量大于运输层实际发出的数据量，即部分数据在测试结束后才实际发出，因此客户端显示的测试数据会偏大（甚至高于链路限制）；而服务器端按照测试期间实际接收的数据量计数，因此不会出现此现象。

2 实验二

2.1 实验步骤

1. 在实验一的基础上，修改`TripleSwitchTopo`类，为`(s1,s2)`与`(s1,s3)`添加5%的丢包率。

```
self.addLink("s1", "s2", bw=10, loss=5, use_htb=True)
self.addLink("s1", "s3", bw=10, loss=5, use_htb=True)
```

2. 运行代码，完整实验代码见`hw2.py`。

2.2 运行结果

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (10.00Mbit 5.00000% loss) (10.00Mbit 5.00000% loss) (s1, s2) (10.
00Mbit 5.00000% loss) (10.00Mbit 5.00000% loss) (s1, s3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit 5.00000% loss) (10.00Mbit 5.00000% loss) (10.00Mbit 5.00000% loss) (1
0.00Mbit 5.00000% loss)
Dumping host connections
h1 h1-eth0:s1-eth3
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth2
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 X
*** Results: 16% dropped (5/6 received)
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['7.37 Mbits/sec', '7.51 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['6.45 Mbits/sec', '7.08 Mbits/sec']
*** Iperf: testing TCP bandwidth between h2 and h3
*** Results: ['1.09 Mbits/sec', '796 Kbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 5 links
.....
*** Stopping 3 switches
s1 s2 s3
*** Stopping 3 hosts
h1 h2 h3
*** Done
```

2.3 结果分析

从ping的结果可以看出，主机之间发生了数据包丢失。

iperf测试结果相比实验一中传输速度均有下降，我认为这是因为丢包使得TCP协议认为线路拥堵，提前触发流量控制算法降低发包速度。其中`h2-h3`的链路经过了两条不稳定的连接，其传输速度大幅度下降，其余主机之间的链路仅经过单条不稳定连接，受到的影响相对较小。

3 实验二

3.1 实验步骤

1. 在实验二的基础上添加`s2-s3`的连接。

```
self.addLink("s2", "s3", use_htb=True)
```

2. 运行程序，发现任意两台主机之间均无法通信。
3. 手动为交换机添加转发规则。

   规则设计如下：

   - 若交换机收到直接相连的主机的数据包，则向其他所有相连的交换机转发。

- 若交换机收到其他交换机的数据包，则向与自己直接相连的主机转发。

以上规则设计可以让本网络中任意两台主机间的数据包通过两次转发到达目的地。

此处使用`ovs-ofctl`命令设置转发规则，调用`mininet`的函数`quietRun`执行命令。代码如下：

```
quietRun("ovs-ofctl add-flow s1 in_port=s1-eth1,actions=output:all")
quietRun("ovs-ofctl add-flow s2 in_port=s2-eth2,actions=output:all")
quietRun("ovs-ofctl add-flow s3 in_port=s3-eth2,actions=output:all")
quietRun("ovs-ofctl add-flow s1 in_port=s1-eth2,actions=output:s1-eth1")
quietRun("ovs-ofctl add-flow s1 in_port=s1-eth3,actions=output:s1-eth1")
quietRun("ovs-ofctl add-flow s2 in_port=s2-eth1,actions=output:s2-eth2")
quietRun("ovs-ofctl add-flow s2 in_port=s2-eth3,actions=output:s2-eth2")
quietRun("ovs-ofctl add-flow s3 in_port=s3-eth1,actions=output:s3-eth2")
quietRun("ovs-ofctl add-flow s3 in_port=s3-eth3,actions=output:s3-eth2")
```

4. 重新运行程序，完整实验代码见`hw3.py`。

3.2 运行结果

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (10.00Mbit 5.00000% loss) (10.00Mbit 5.00000% loss
) (s1, s2) (10.00Mbit 5.00000% loss) (10.00Mbit 5.00000% loss) (s1, s3) (s2,
s3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit 5.00000% loss) (10.00Mbit 5.00000% loss) (10.00Mbit 5.
00000% loss) (10.00Mbit 5.00000% loss)
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth2
Dumping switch connections
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth1 s1-eth3:s3-eth1
s2 lo:  s2-eth1:s1-eth2 s2-eth2:h2-eth0 s2-eth3:s3-eth3
s3 lo:  s3-eth1:s1-eth3 s3-eth2:h3-eth0 s3-eth3:s2-eth3
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['6.96 Mbits/sec', '7.06 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['7.28 Mbits/sec', '8.16 Mbits/sec']
*** Iperf: testing TCP bandwidth between h2 and h3
*** Results: ['35.1 Gbits/sec', '35.2 Gbits/sec']
```

```
*** Starting CLI:
mininet> h1 ping h2 -c4
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=0.680 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.081 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.054 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3080ms
rtt min/avg/max/mdev = 0.043/0.214/0.680/0.269 ms
mininet>
```

3.3 结果分析

未添加路由规则时，主机之间无法通信，使用`h1` ping `h2`会丢失全部数据包，这是因为交换机没有正常工作。

从运行结果图2可以看出，在添加路由规则之后，任意两台主机之间均可以正常通信，通过`h1` ping `h2`可以得到响应，说明交换机正在按照路由规则转发数据包。

注意到图1的iperf测试中，`h2`与`h3`之间的连接速度远超过其他的主机间连接速度，这是因为创建连接`s2-s3`时，未限制其带宽，根据我们的路由规则，`h2`与`h3`之间的连接不会通过任意一条有带宽限制的连接，这也说明了我们的路由规则运行正常。