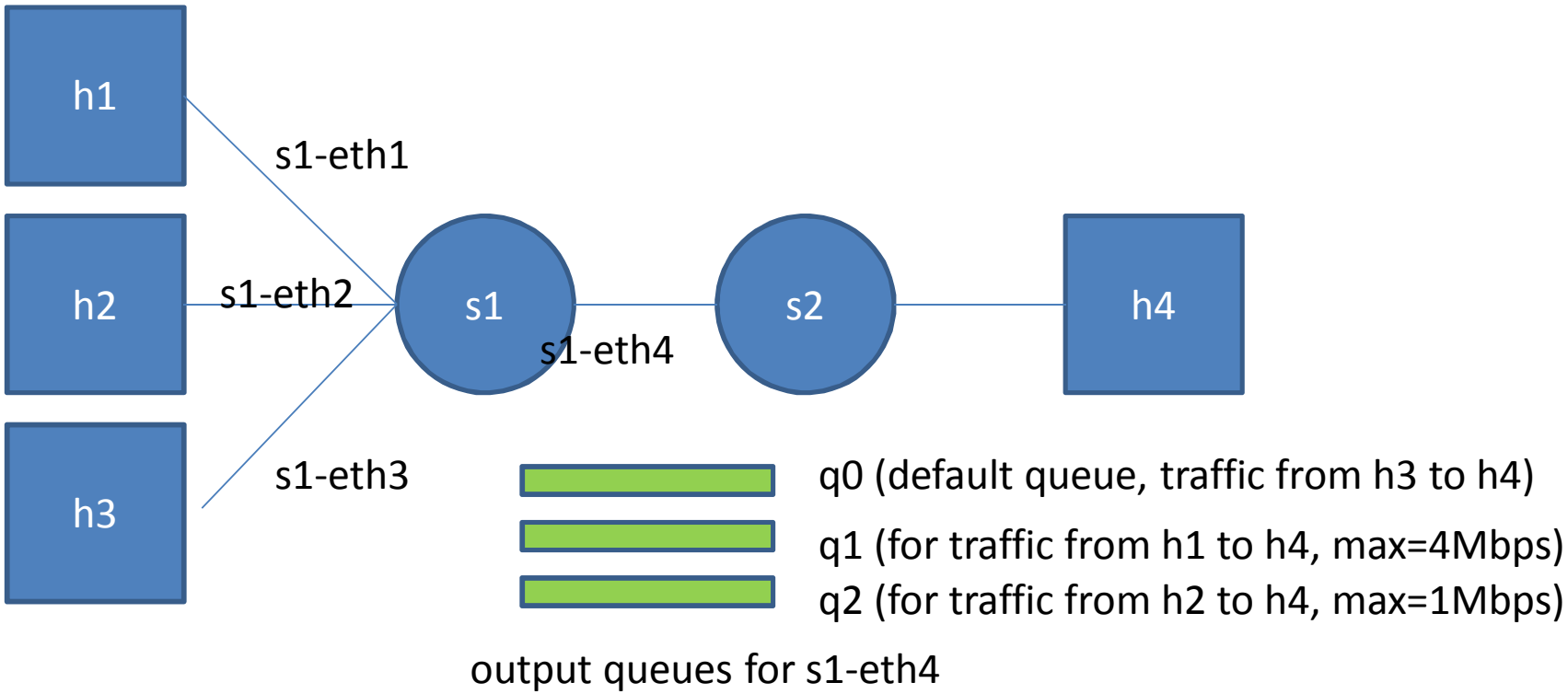
Lab 5: set traffic to different output queues (QoS issue)



Dr. Chih-Heng Ke
smallko@gmail.com

lab5.py

```
from mininet.topo import Topo

class MyTopo( Topo ):
   "Simple topology example."

   def __init__( self ):
      "Create custom topo."

      # Initialize topology
      Topo.__init__( self )

      # Add hosts and switches
      h1 = self.addHost( 'h1' )
      h2 = self.addHost( 'h2' )
      h3 = self.addHost( 'h3' )
      h4 = self.addHost( 'h4' )
      s1 = self.addSwitch( 's1' )
      s2 = self.addSwitch( 's2' )

      # Add links
      self.addLink( h1, s1 )
      self.addLink( h2, s1 )
      self.addLink( h3, s1 )
      self.addLink( s1, s2 )
      self.addLink( s2, h4 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

lab5_controller.py

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr
log = core.getLogger()

s1_dpid=0
s2_dpid=0

def _handle_ConnectionUp (event):
  global s1_dpid, s2_dpid
  print "ConnectionUp: ",
dpidToStr(event.connection.dpid)

 #remember the connection dpid for switch
  for m in event.connection.features.ports:
    if m.name == "s1-eth1":
      s1_dpid = event.connection.dpid
      print "s1_dpid=", s1_dpid
    elif m.name == "s2-eth1":
      s2_dpid = event.connection.dpid
      print "s2_dpid=", s2_dpid
```

```
def _handle_PacketIn (event):
  global s1_dpid, s2_dpid
  # print "PacketIn: ", dpidToStr(event.connection.dpid)

  if event.connection.dpid==s1_dpid:
    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0806
    msg.actions.append(of.ofp_action_output(port = of.OFPP_ALL))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =100
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0800
    msg.match.nw_src = "10.0.0.1"
    msg.match.nw_dst = "10.0.0.4"
    msg.actions.append(of.ofp_action_enqueue(port = 4, queue_id=1))
    event.connection.send(msg)
```

```
msg = of.ofp_flow_mod()
msg.priority =100
msg.idle_timeout = 0
msg.hard_timeout = 0
msg.match.dl_type = 0x0800
msg.match.nw_src = "10.0.0.2"
msg.match.nw_dst = "10.0.0.4"
msg.actions.append(of.ofp_action_enqueue(port = 4, queue_id=2))
event.connection.send(msg)

msg = of.ofp_flow_mod()
msg.priority =10
msg.idle_timeout = 0
msg.hard_timeout = 0
msg.match.dl_type = 0x0800
msg.match.nw_dst = "10.0.0.1"
msg.actions.append(of.ofp_action_output(port = 1))
event.connection.send(msg)
```

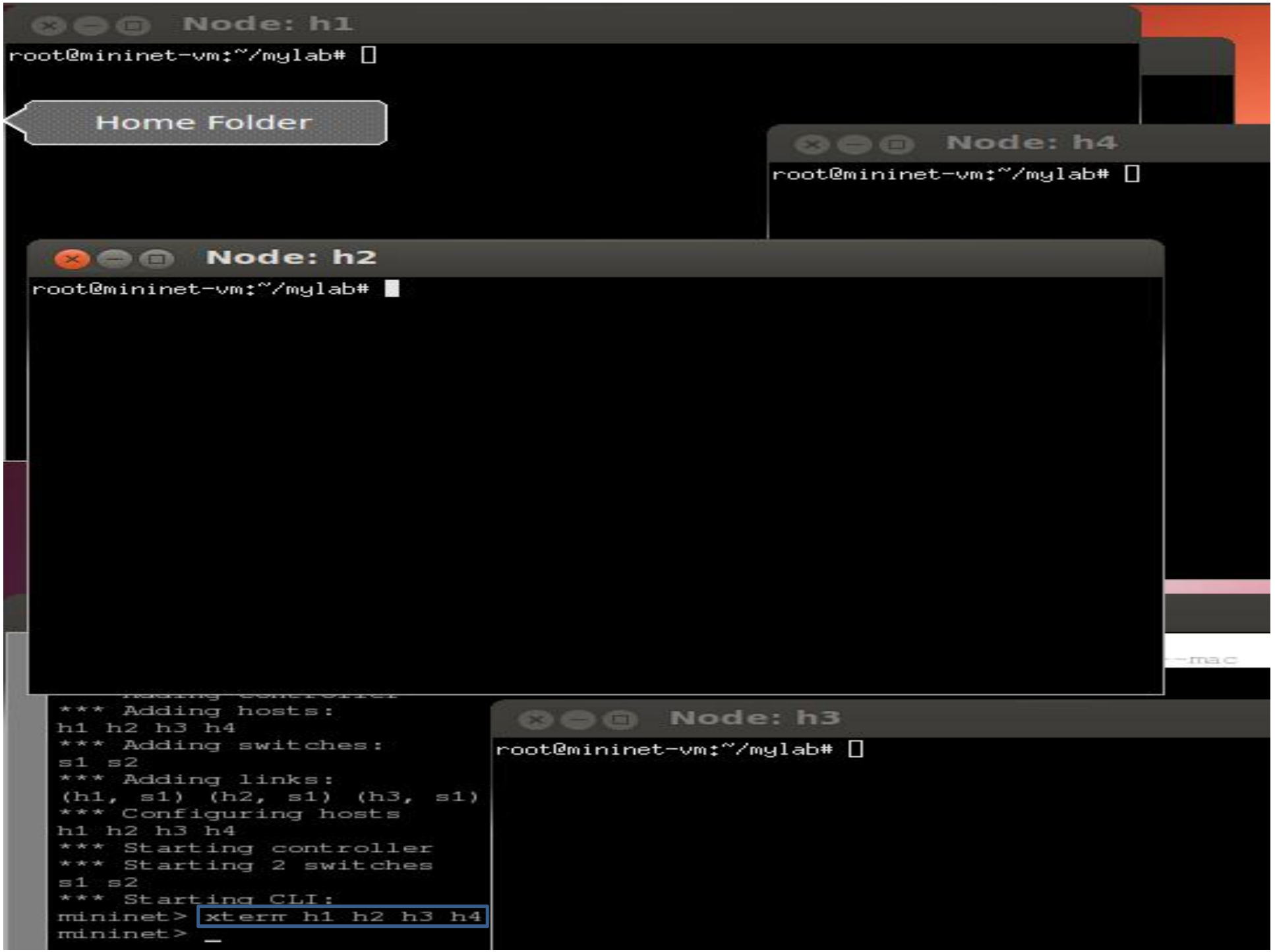
```
msg = of.ofp_flow_mod()
msg.priority =10
msg.idle_timeout = 0
msg.hard_timeout = 0
msg.match.dl_type = 0x0800
msg.match.nw_dst = "10.0.0.2"
msg.actions.append(of.ofp_action_output(port = 2))
event.connection.send(msg)

msg = of.ofp_flow_mod()
msg.priority =10
msg.idle_timeout = 0
msg.hard_timeout = 0
msg.match.dl_type = 0x0800
msg.match.nw_dst = "10.0.0.3"
msg.actions.append(of.ofp_action_output(port = 3))
event.connection.send(msg)
```

```
  msg = of.ofp_flow_mod()
  msg.priority =10
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.dl_type = 0x0800
  msg.match.nw_dst = "10.0.0.4"
  msg.actions.append(of.ofp_action_output(port = 4))
  event.connection.send(msg)

elif event.connection.dpid==s2_dpid:
  msg = of.ofp_flow_mod()
  msg.priority =1
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.in_port =1
  msg.actions.append(of.ofp_action_output(port = 2))
  event.connection.send(msg)
```

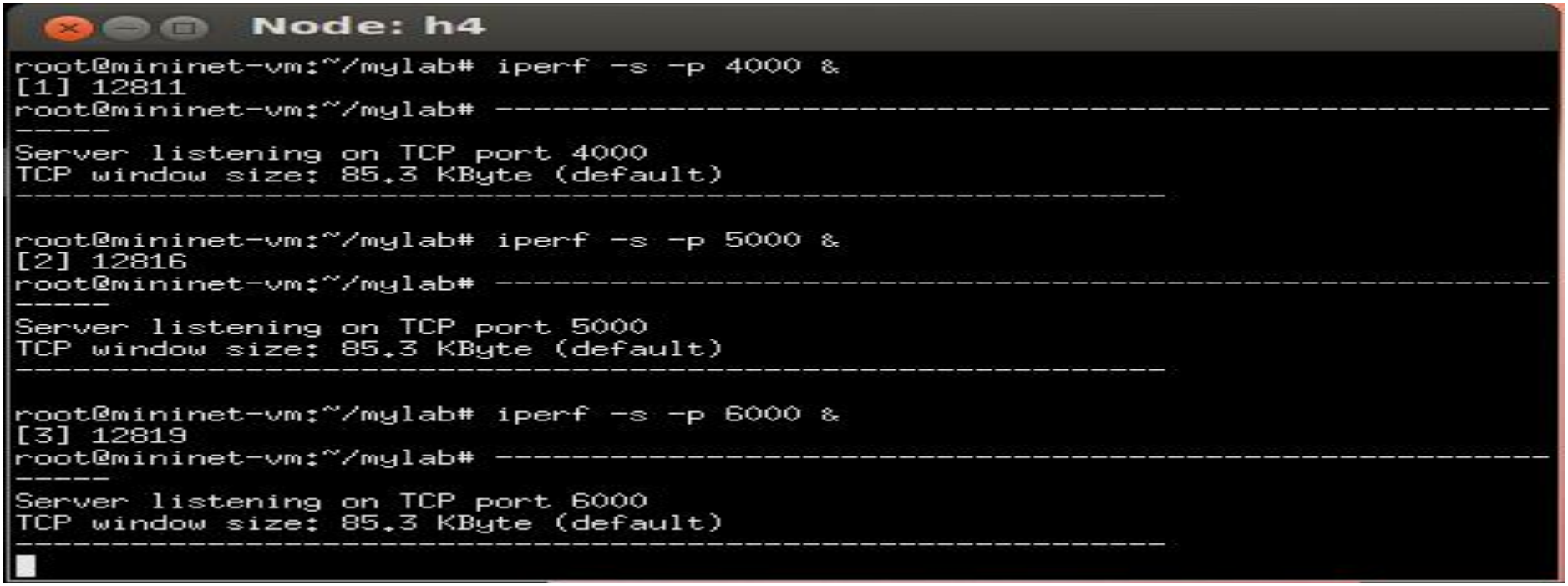
```
    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.in_port =2
    msg.actions.append(of.ofp_action_output(port = 1))
    event.connection.send(msg)

def launch ():
  core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
  core.openflow.addListenerByName("PacketIn", _handle_PacketIn)
```

using default controller: all traffic go into the same output queue for s1-eth4

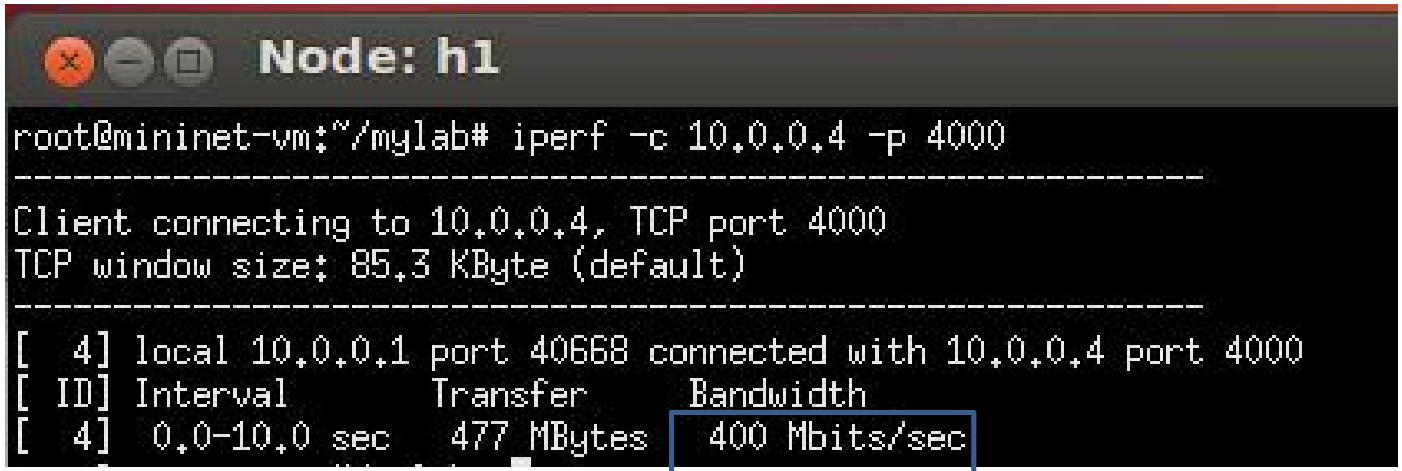
```
mininet@mininet-vm:~$ cd mylab/
mininet@mininet-vm:~/mylab$ sudo mn --custom lab5.py --topo mytopo --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 2 switches
s1 s2
*** Starting CLI:
mininet> _
```


Node: h1
root@mininet-vm:~/mylab#
Home Folder
Node: h4
root@mininet-vm:~/mylab#
Node: h2
root@mininet-vm:~/mylab#
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 2 switches
s1 s2
*** Starting CLI:
mininet> xterm h1 h2 h3 h4
mininet>
Node: h3
root@mininet-vm:~/mylab#

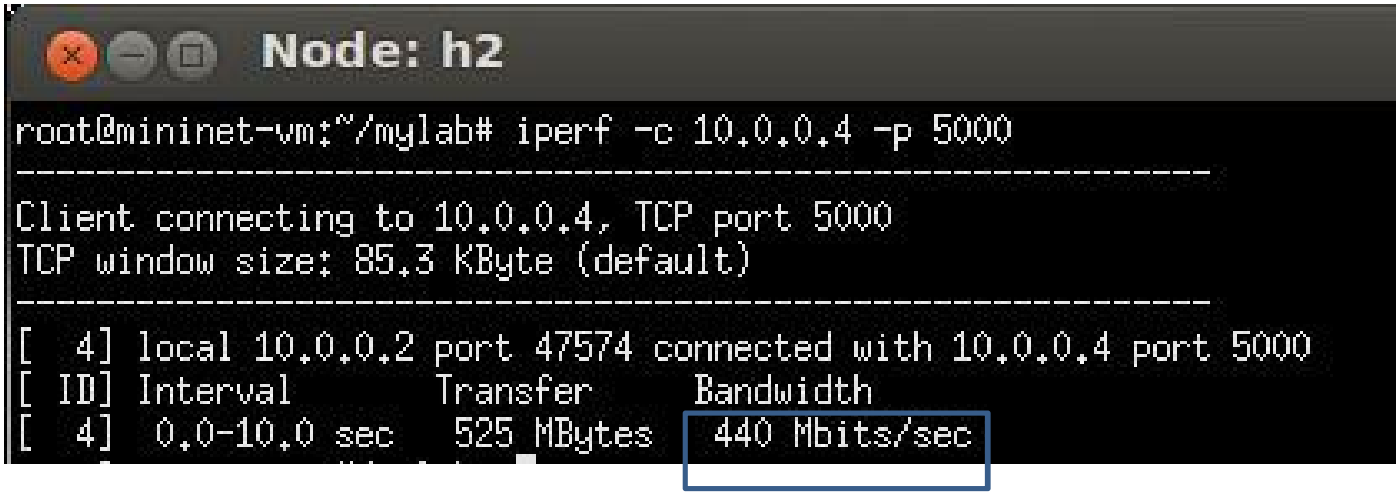
For h4, start Iperf servers at port 4000, 5000, 6000 respectively



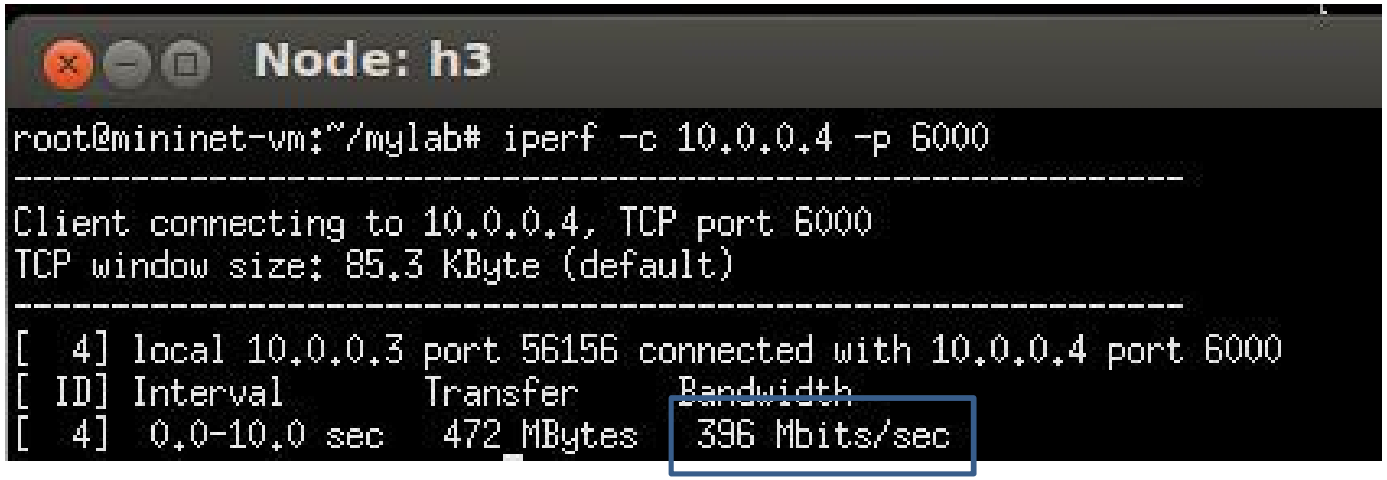
Test the throughput from h1 to h4 (no other background traffic)



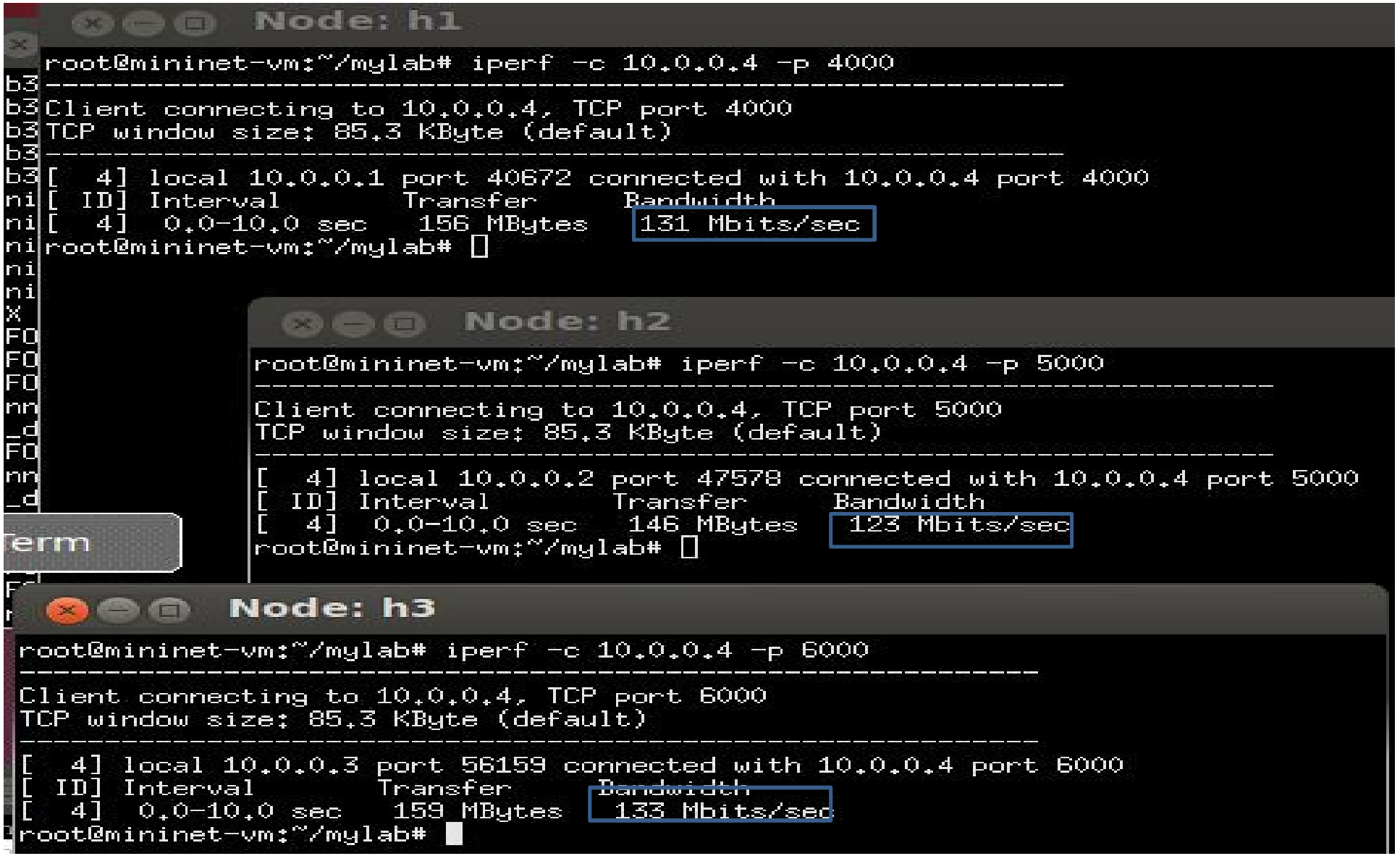
Test the throughput from h2 to h4 (no other background traffic)



Test the throughput from h3 to h4 (no other background traffic)

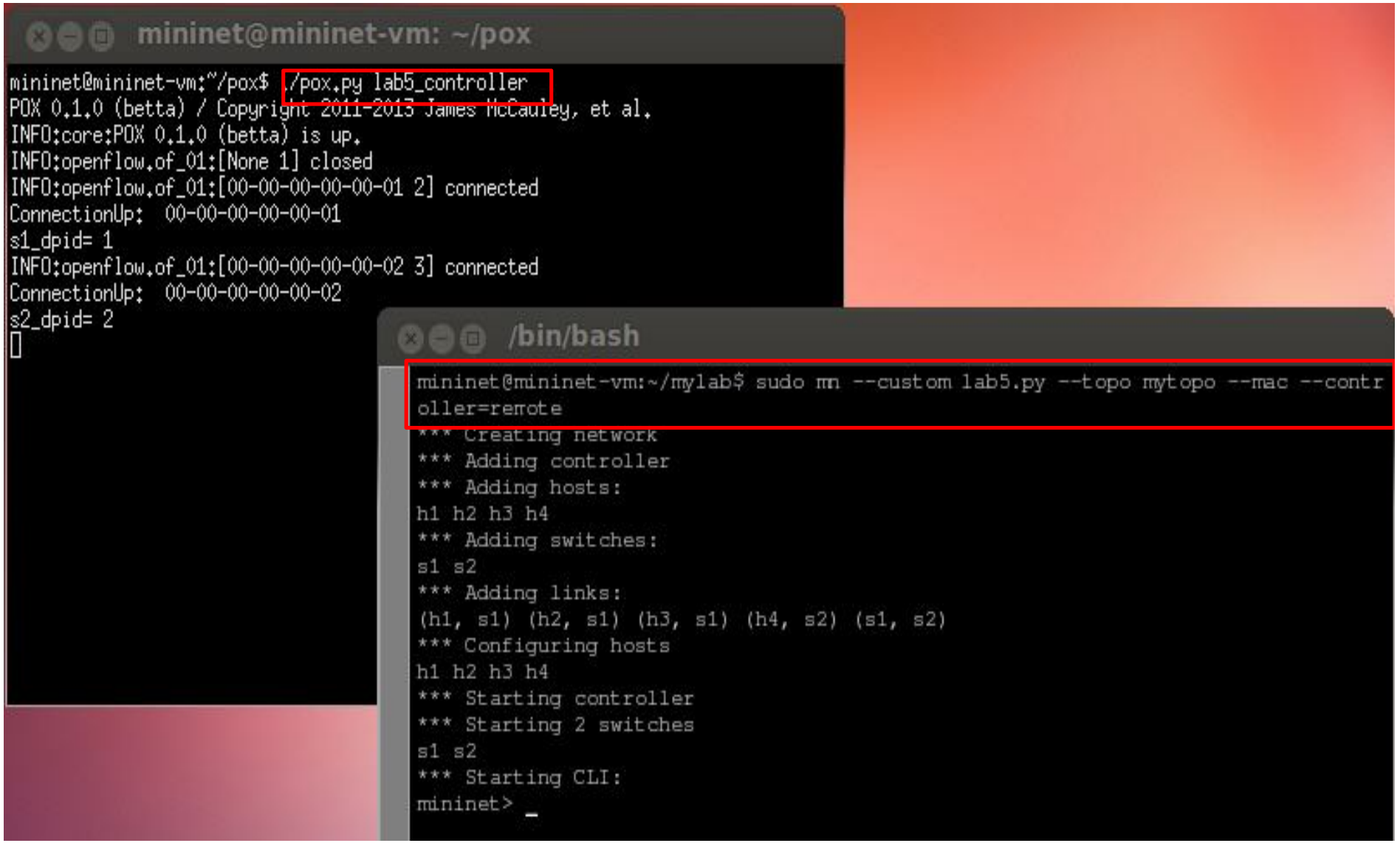


The throughputs measured are around 400 Mbps. These values are depending the emulation environment, such as CPU and working load.

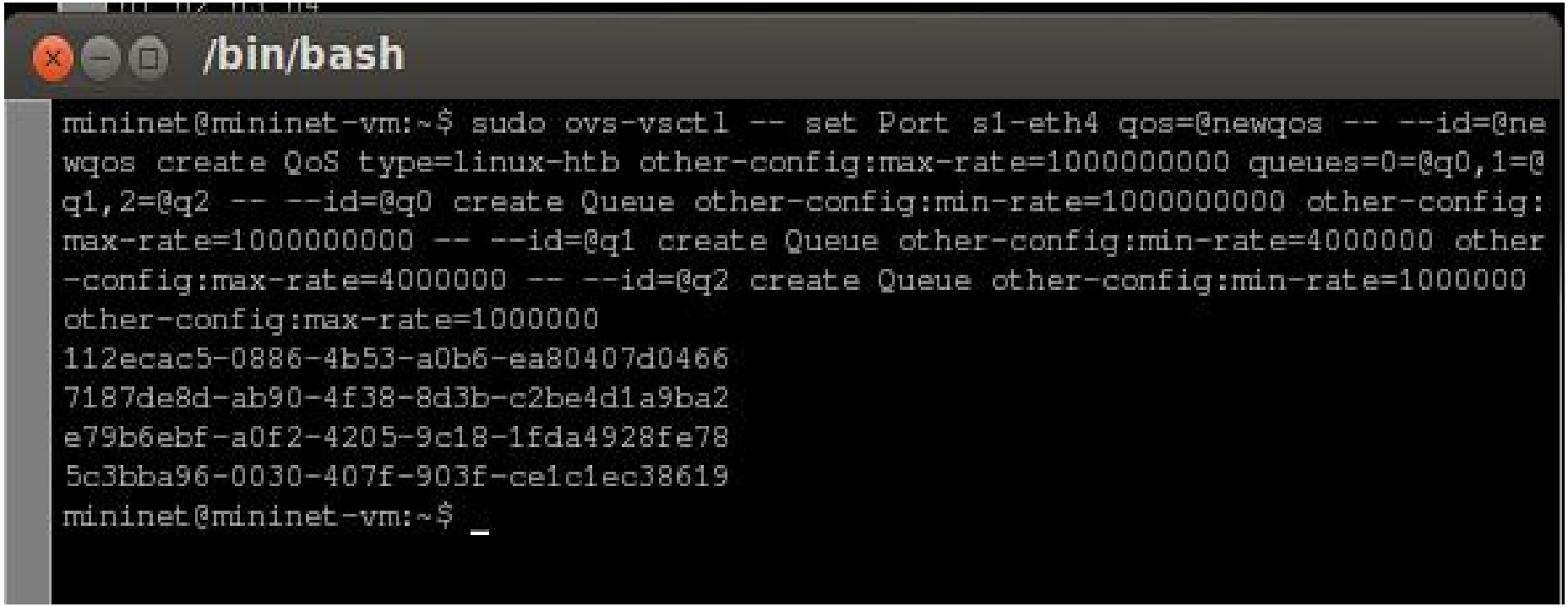


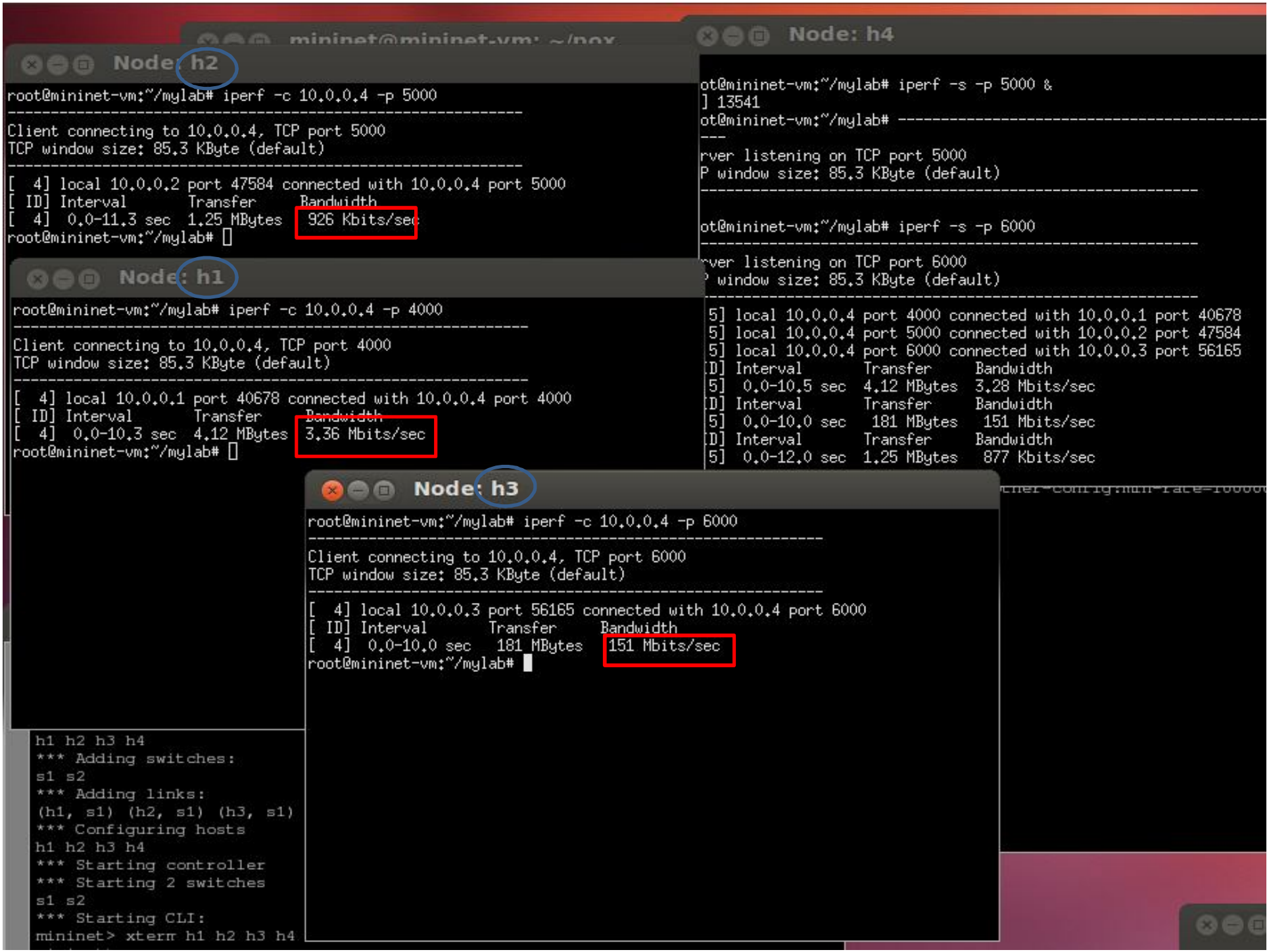
Start Iperf client at h1, h2, and h3 at almost the same time. We can see the measured throughput is similar. These three flows can equally divide the bandwidth from s1 to s2.

using lab5_controller: traffic from h1 goes to q1, traffic from h2 goes to q2, traffic from h3 goes to q0



Using ovs-vsctl to create three queues for s1-eth4, i.e. q0, q1, and q2 and to set the rate for each queue




Node: h2
root@mininet-vm:~/mylab# iperf -c 10.0.0.4 -p 5000
Client connecting to 10.0.0.4, TCP port 5000
TCP window size: 85.3 KByte (default)
[4] local 10.0.0.2 port 47584 connected with 10.0.0.4 port 5000
[ID] Interval Transfer Bandwidth
[4] 0.0-11.3 sec 1.25 MBytes 926 Kbits/sec
root@mininet-vm:~/mylab#
Node: h1
root@mininet-vm:~/mylab# iperf -c 10.0.0.4 -p 4000
Client connecting to 10.0.0.4, TCP port 4000
TCP window size: 85.3 KByte (default)
[4] local 10.0.0.1 port 40678 connected with 10.0.0.4 port 4000
[ID] Interval Transfer Bandwidth
[4] 0.0-10.3 sec 4.12 MBytes 3.36 Mbits/sec
root@mininet-vm:~/mylab#
Node: h4
ot@mininet-vm:~/mylab# iperf -s -p 5000 &
] 13541
ot@mininet-vm:~/mylab#
rver listening on TCP port 5000
P window size: 85.3 KByte (default)
ot@mininet-vm:~/mylab# iperf -s -p 6000
rver listening on TCP port 6000
window size: 85.3 KByte (default)
5] local 10.0.0.4 port 4000 connected with 10.0.0.1 port 40678
5] local 10.0.0.4 port 5000 connected with 10.0.0.2 port 47584
5] local 10.0.0.4 port 6000 connected with 10.0.0.3 port 56165
D] Interval Transfer Bandwidth
5] 0.0-10.5 sec 4.12 MBytes 3.28 Mbits/sec
D] Interval Transfer Bandwidth
5] 0.0-10.0 sec 181 MBytes 151 Mbits/sec
D] Interval Transfer Bandwidth
5] 0.0-12.0 sec 1.25 MBytes 877 Kbits/sec
Node: h3
root@mininet-vm:~/mylab# iperf -c 10.0.0.4 -p 6000
Client connecting to 10.0.0.4, TCP port 6000
TCP window size: 85.3 KByte (default)
[4] local 10.0.0.3 port 56165 connected with 10.0.0.4 port 6000
[ID] Interval Transfer Bandwidth
[4] 0.0-10.0 sec 181 MBytes 151 Mbits/sec
root@mininet-vm:~/mylab#
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 2 switches
s1 s2
*** Starting CLI:
mininet> xterm h1 h2 h3 h4

reference

- QoS on OpenFlow 1.0 with OVS 1.4.3 and POX inside Mininet (http://users.ecs.soton.ac.uk/drn/ofertie/openflow_qos_mininet.pdf)