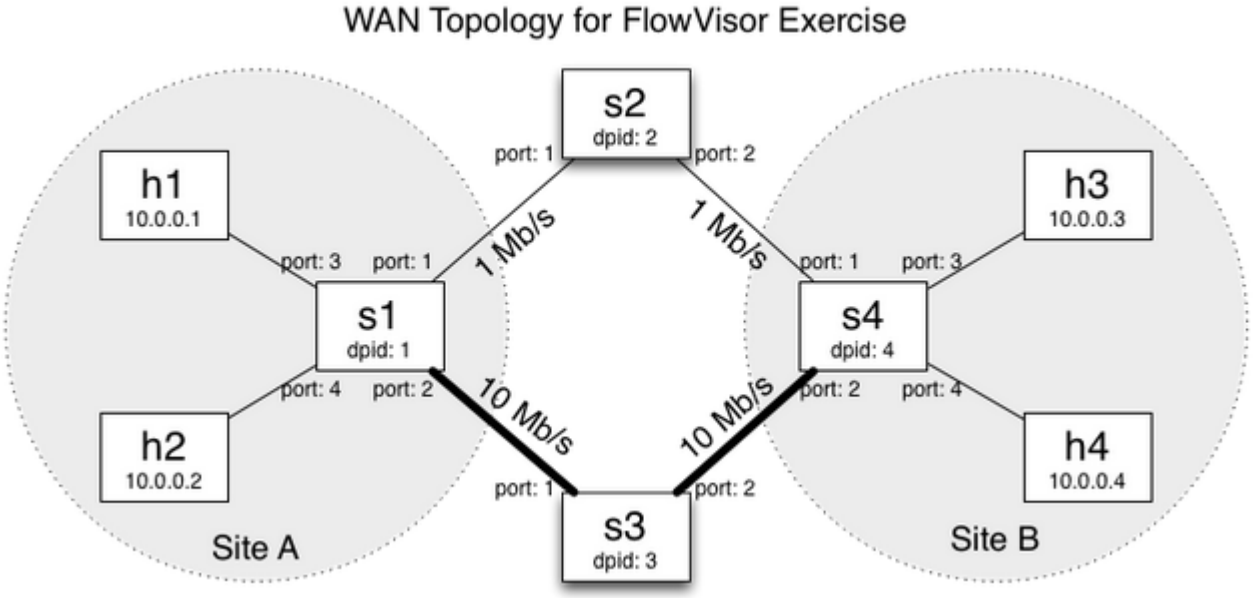
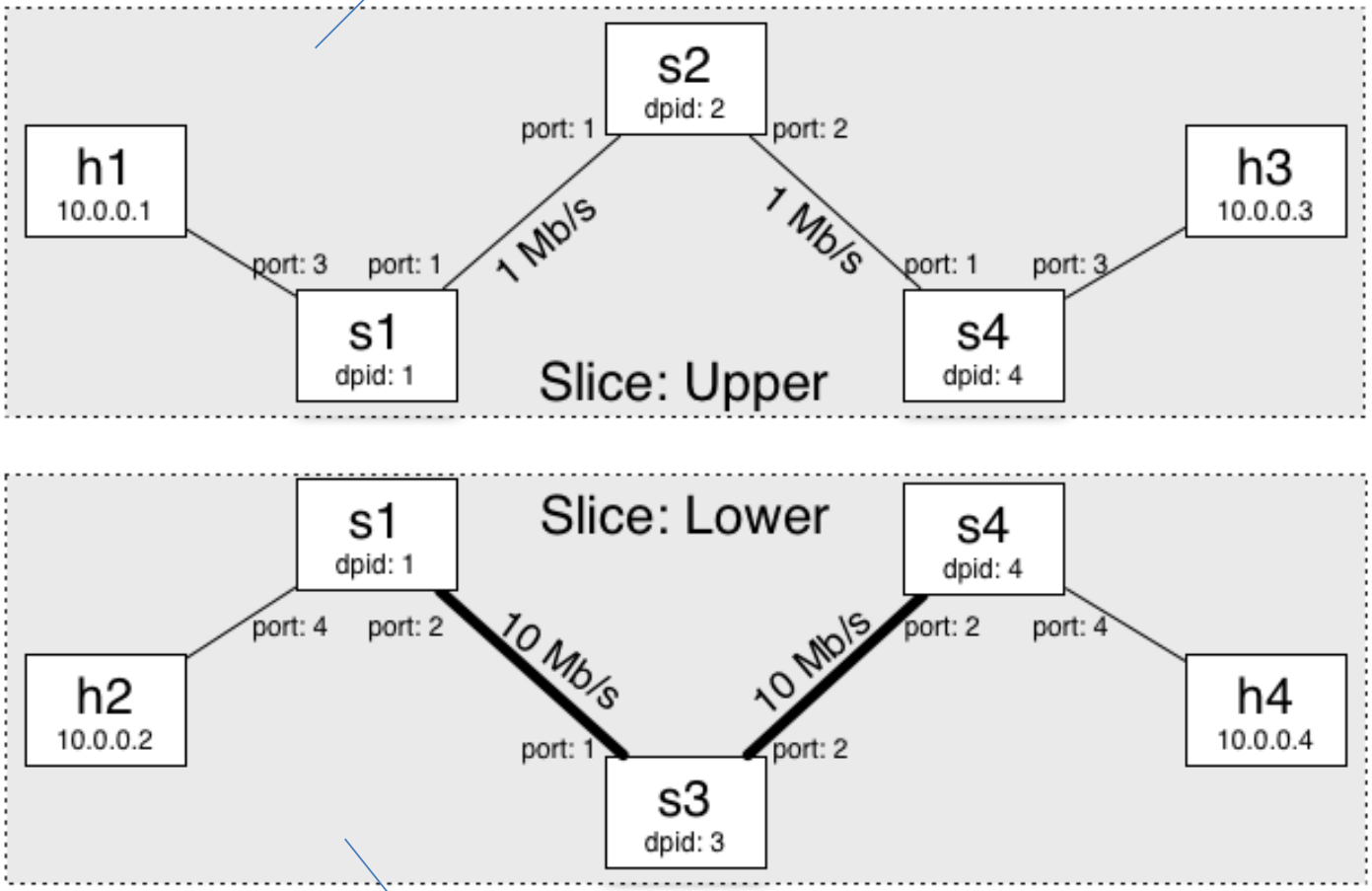
Lab 6 (FlowVisor)

- Flowvisor creates rich slices of network resources and delegates control of each slice to each slice to a different controller.



- a low bandwidth path via switch s2
- a high bandwidth path via switch s3

Controlled by flowvisor_lab1_upper



Controlled by flowvisor_lab1_lower

```
#!/usr/bin/python

from mininet.topo import Topo

class FVTopo(Topo):

    def __init__(self):
        # Initialize topology
        Topo.__init__(self)

        # Create template host, switch, and link
        hconfig = {'inNamespace':True}
        http_link_config = {'bw': 1}
        video_link_config = {'bw': 10}
        host_link_config = {}

        # Create switch nodes
        for i in range(4):
            sconfig = {'dpid': "%016x" % (i+1)}
            self.addSwitch('s%d' % (i+1), **sconfig)

        # Create host nodes
        for i in range(4):
            self.addHost('h%d' % (i+1), **hconfig)

        # Add switch links
        self.addLink('s1', 's2', **http_link_config)
        self.addLink('s2', 's4', **http_link_config)
        self.addLink('s1', 's3', **video_link_config)
        self.addLink('s3', 's4', **video_link_config)

        # Add host links
        self.addLink('h1', 's1', **host_link_config)
        self.addLink('h2', 's1', **host_link_config)
        self.addLink('h3', 's4', **host_link_config)
        self.addLink('h4', 's4', **host_link_config)

topos = { 'fvtopo': ( lambda: FVTopo() ) }
```

flowvisor_topo.py

flowvisor_lab1_upper.py

Put this file under ~/pox/ext

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr
from pox.openflow.of_json import *
from pox.lib.recoco import Timer

log = core.getLogger()

s1_dpid=0
s2_dpid=0
s3_dpid=0
s4_dpid=0

def _timer_func():
  for connection in core.openflow._connections.values():
    connection.send(of.ofp_stats_request(body=of.ofp_port_stats_request()))
  print "Sent %i port stats request(s)" % (len(core.openflow._connections))

def _handle_portstats_received(event):
  #stats=flow_stats_to_list(event.stats)
  #print "PortStatsReceived from %s: %s" % (dpidToStr(event.connection.dpid), stats)
  for f in event.stats:
    if int(f.port_no)<65534:
      print "PortNo:", f.port_no, " dpid:", event.connection.dpid
```

```
def _handle_ConnectionUp (event):
  global s1_dpid, s2_dpid, s3_dpid, s4_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
    elif m.name == "s3-eth1":
      s3_dpid = event.connection.dpid
      print "s3_dpid=", s3_dpid
    elif m.name == "s4-eth1":
      s4_dpid = event.connection.dpid
      print "s4_dpid=", s4_dpid
```

```
def _handle_PacketIn (event):
  global s1_dpid, s2_dpid, s3_dpid, s4_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.in_port =3
    msg.actions.append(of.ofp_action_output(port = 1))
    event.connection.send(msg)

    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 = 3))
    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)
```

```
elif event.connection.dpid==s4_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 = 3))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.in_port =3
    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)
  core.openflow.addListenerByName("PortStatsReceived",
_handle_portstats_received)

  Timer(5, _timer_func, recurring=True)
```

flowvisor_lab1_lower.py

Put this file under ~/pox/ext

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr
from pox.openflow.of_json import *
from pox.lib.recoco import Timer

log = core.getLogger()

s1_dpid=0
s2_dpid=0
s3_dpid=0
s4_dpid=0

def _timer_func():
  for connection in core.openflow._connections.values():
    connection.send(of.ofp_stats_request(body=of.ofp_port_stats_request()))
  print "Sent %i port stats request(s)" % (len(core.openflow._connections))

def _handle_portstats_received(event):
  #stats=flow_stats_to_list(event.stats)
  #print "PortStatsReceived from %s: %s" % (dpidToStr(event.connection.dpid), stats)
  for f in event.stats:
    if int(f.port_no)<65534:
      print "PortNo:", f.port_no, " dpid:", event.connection.dpid
```

```
def _handle_ConnectionUp (event):
  global s1_dpid, s2_dpid, s3_dpid, s4_dpid
  print "ConnectionUp: ",
dpidToStr(event.connection.dpid)

 #remember the connection dpid for switch
  for m in event.connection.features.ports:
    if m.name == "s1-eth2":
      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
    elif m.name == "s3-eth1":
      s3_dpid = event.connection.dpid
      print "s3_dpid=", s3_dpid
    elif m.name == "s4-eth2":
      s4_dpid = event.connection.dpid
      print "s4_dpid=", s4_dpid
```

```
def _handle_PacketIn (event):
  global s1_dpid, s2_dpid, s3_dpid, s4_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.in_port =4
    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 = 4))
    event.connection.send(msg)
```

```
elif event.connection.dpid==s3_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)
```

```
elif event.connection.dpid==s4_dpid:
    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 = 4))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.in_port =4
    msg.actions.append(of.ofp_action_output(port = 2))
    event.connection.send(msg)
```

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

  Timer(5, _timer_func, recurring=True)
```

Run Diamond Topology

```
root@mininet-vm:/home/mininet/mylab# sudo mn --custom flowvisor_topo.py --topo f
vtopo --link tc --controller remote --mac --arp
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s1) (h3, s4) (h4, s4) (1.00Mbit) (1.00Mbit) (s1, s2) (10.00Mbit) (
10.00Mbit) (s1, s3) (1.00Mbit) (1.00Mbit) (s2, s4) (10.00Mbit) (10.00Mbit) (s3,
s4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 4 switches
s1 (1.00Mbit) (10.00Mbit) s2 (1.00Mbit) (1.00Mbit) s3 (10.00Mbit) (10.00Mbit) s4
 (1.00Mbit) (10.00Mbit)
*** Starting CLI:
mininet>
```

Start FlowVisor

```
root@mininet-vm:~# sudo -u flowvisor flowvisor
Starting FlowVisor
--- Setting logging level to NOTE
2014-01-21 03:59:01.518:INFO::Logging to StdErrLog::DEBUG=false via org.eclipse.
jetty.util.log.StdErrLog
2014-01-21 03:59:01.616:INFO::jetty-7.0.2.v20100331
2014-01-21 03:59:03.318:INFO::Started SslSelectChannelConnector@0.0.0.0:8081
```

Create Slices

```
root@mininet-vm:~# fvctl -f /dev/null add-slice upper tcp:localhost:10001 admin@
upperslice
Slice password:
Slice upper was successfully created
root@mininet-vm:~# fvctl -f /dev/null add-slice lower tcp:localhost:10002 admin@
lowerslice
Slice password:
Slice lower was successfully created
root@mininet-vm:~# fvctl -f /dev/null list-slices
Configured slices:
fvadmin         --> enabled
upper           --> enabled
lower           --> enabled
root@mininet-vm:~# _
```

Create Flowspaces for upper

```
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid1-port3 1 1 in_port=3 up
per=7
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid1-port1 1 1 in_port=1 up
per=7
FlowSpace dpid1-port1 was added with request id 1.
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid2 2 1 any upper=7
FlowSpace dpid2 was added with request id 2.
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid4-port1 4 1 in_port=1 up
per=7
FlowSpace dpid4-port1 was added with request id 3.
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid4-port3 4 1 in_port=3 up
per=7
FlowSpace dpid4-port3 was added with request id 4.
```

Create Flowspaces for lower

```
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid1-port2 1 1 in_port=2 lo
wer=7
FlowSpace dpid1-port2 was added with request id 5.
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid1-port4 1 1 in_port=4 lo
wer=7
FlowSpace dpid1-port4 was added with request id 6.
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid3 3 1 any lower=7
FlowSpace dpid3 was added with request id 7.
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid4-port2 4 1 in_port=2 lo
wer=7
FlowSpace dpid4-port2 was added with request id 8.
root@mininet-vm:~# fvctl -f /dev/null add-flowspace dpid4-port4 4 1 in_port=4 lo
wer=7
FlowSpace dpid4-port4 was added with request id 9.
```

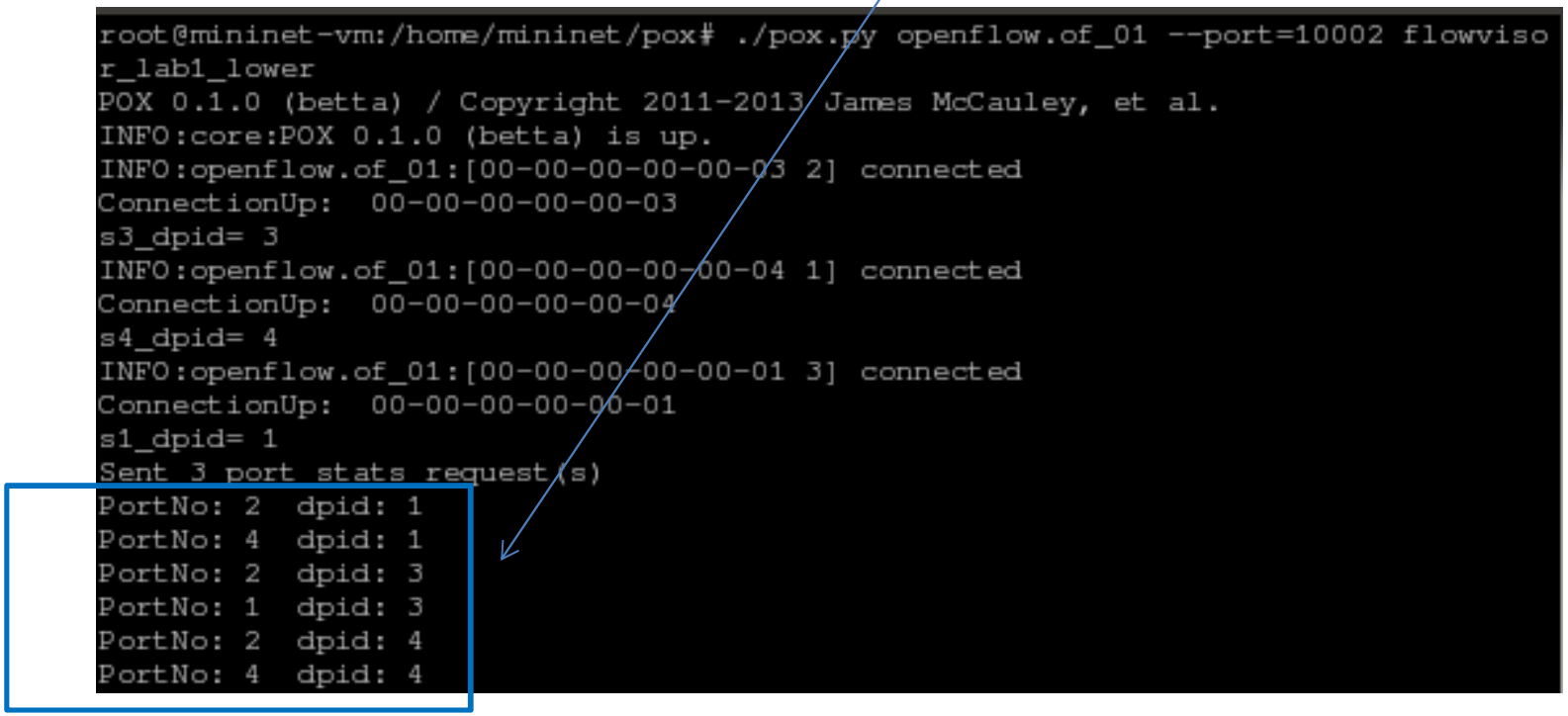
Start flowvisor_lab1_upper controller

Only s1, s2, and s4 can be seen

```
root@mininet-vm:/home/mininet/pox# ./pox.py openflow.of_01 --port=10001 flowviso
r_lab1_upper
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.1.0 (betta) is up.
Sent 0 port stats request(s)
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
ConnectionUp:  00-00-00-00-00-01
s1_dpid= 1
Sent 1 port stats request(s)
PortNo: 3  dpid: 1
PortNo: 1  dpid: 1
INFO:openflow.of_01:[00-00-00-00-00-04 2] connected
ConnectionUp:  00-00-00-00-00-04
s4_dpid= 4
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
ConnectionUp:  00-00-00-00-00-02
s2_dpid= 2
Sent 3 port stats request(s)
PortNo: 3  dpid: 4
PortNo: 1  dpid: 4
PortNo: 3  dpid: 1
PortNo: 1  dpid: 1
PortNo: 2  dpid: 2
PortNo: 1  dpid: 2
```

Start flowvisor_lab1_lower controller

Only s1, s3, and s4 can be seen



Test Connectivity

h1 can ping h3

```
mininet> h1 ping -c5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_req=1 ttl=64 time=1.56 ms
64 bytes from 10.0.0.3: icmp_req=2 ttl=64 time=0.252 ms
64 bytes from 10.0.0.3: icmp_req=3 ttl=64 time=0.239 ms
64 bytes from 10.0.0.3: icmp_req=4 ttl=64 time=0.237 ms
64 bytes from 10.0.0.3: icmp_req=5 ttl=64 time=0.207 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.207/0.500/1.568/0.534 ms
mininet> h1 ping -c5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4009ms
```

H2 can ping h4

```
mininet> h2 ping -c5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_req=1 ttl=64 time=1.69 ms
64 bytes from 10.0.0.4: icmp_req=2 ttl=64 time=0.299 ms
64 bytes from 10.0.0.4: icmp_req=3 ttl=64 time=0.245 ms
64 bytes from 10.0.0.4: icmp_req=4 ttl=64 time=0.251 ms
64 bytes from 10.0.0.4: icmp_req=5 ttl=64 time=0.248 ms

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.245/0.548/1.697/0.574 ms
mininet> h2 ping -c5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4034ms
```

References

- Installation from Binary, https://github.com/OPENNETWORKINGLAB/flowvisor/wiki/Installation-from-Binary

- Flowvisor Exercise, https://github.com/onstutorial/onstutorial/wiki/Flowvisor-Exercise

- POX Wiki, https://openflow.stanford.edu/display/ONL/POX+Wiki