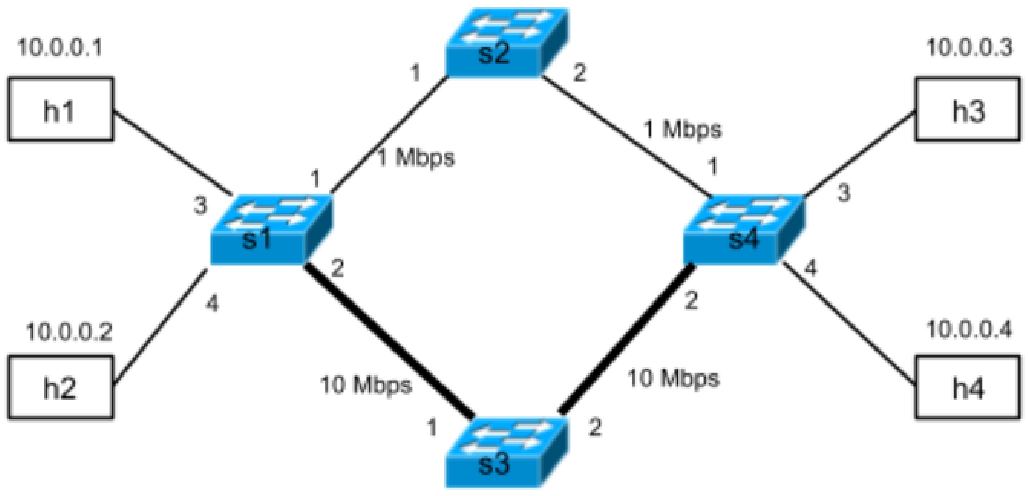
PRACTICAL-8

AIM:

Develop the given topology and Understand Network Virtualization and FlowVisor - Slice OpenFlow network using the "Pox" OpenFlow controller (Self-study)

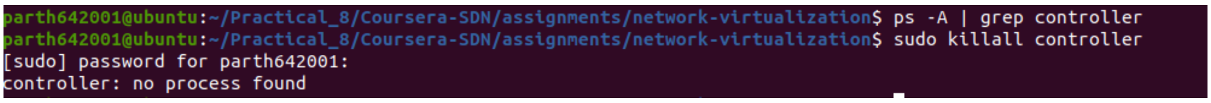


THEORY:

Flowvisor:

- A mechanism for allowing multiple controllers to operate on distinct portions of network flowspace. Flowvisor is implemented in Java, which would require you to install a completely new tool and operating environment. In lieu of doing so, we have created a simpler assignment that illustrates some of the concepts of slicing that Flowvisor uses.
- we will slice a wide-area network (WAN). The WAN shown in the figure above connects two sites. For simplicity, we'll have each site represented by a single OpenFlow switch, s1 and s4, respectively. The sites, s1 and s4, have two paths between them:
- a low-bandwidth path via switch s2
- a high-bandwidth path via switch s3

- Switch s1 has two hosts attached: h1 and h2, and s4 has two hosts attached: h3 and h4.
- Remember to always ensure that there is no other controller running in the background. Check if any controller is running in the background.



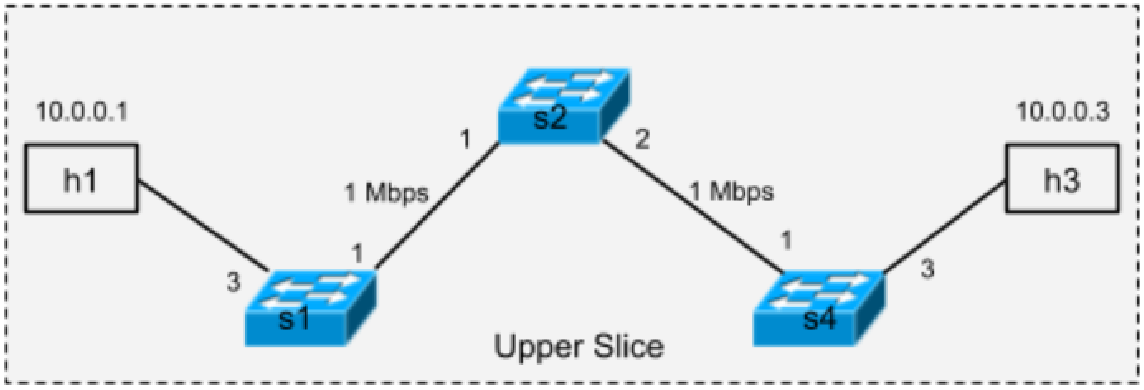
```
parth642001@ubuntu:~/Practical_8/Coursera-SDN/assignments/network-virtualization$ ps -A | grep controller
parth642001@ubuntu:~/Practical_8/Coursera-SDN/assignments/network-virtualization$ sudo killall controller
[sudo] password for parth642001:
controller: no process found
```

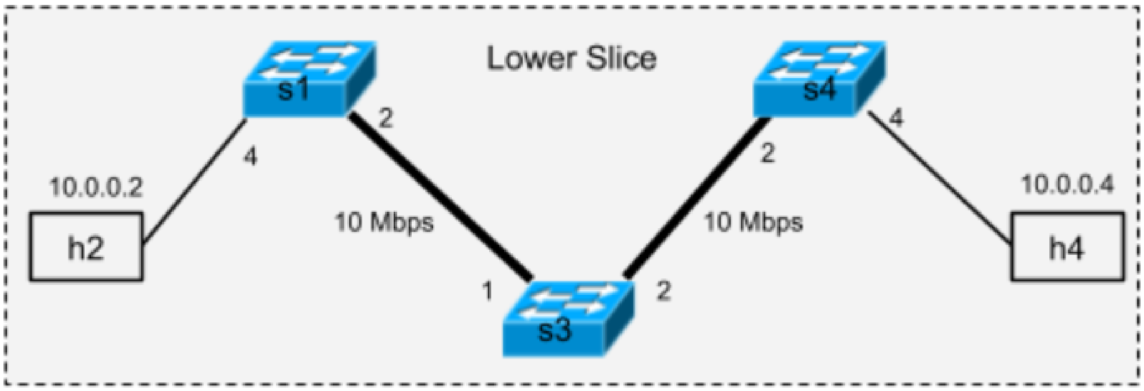
- Restart Mininet to make sure that everything is clean and using the faster kernel switch.
- Talking about the files:

- mininetSlice.py: Mininet script to create the topology used for this assignment.
- topologySlice.py: A skeleton class where you will implement simple topology-based slicing logic.
- videoSlice.py: A skeleton class where you will implement advanced flowspace slicing logic.
- submit.py: The submission script to submit your assignment's output to the Coursera servers.

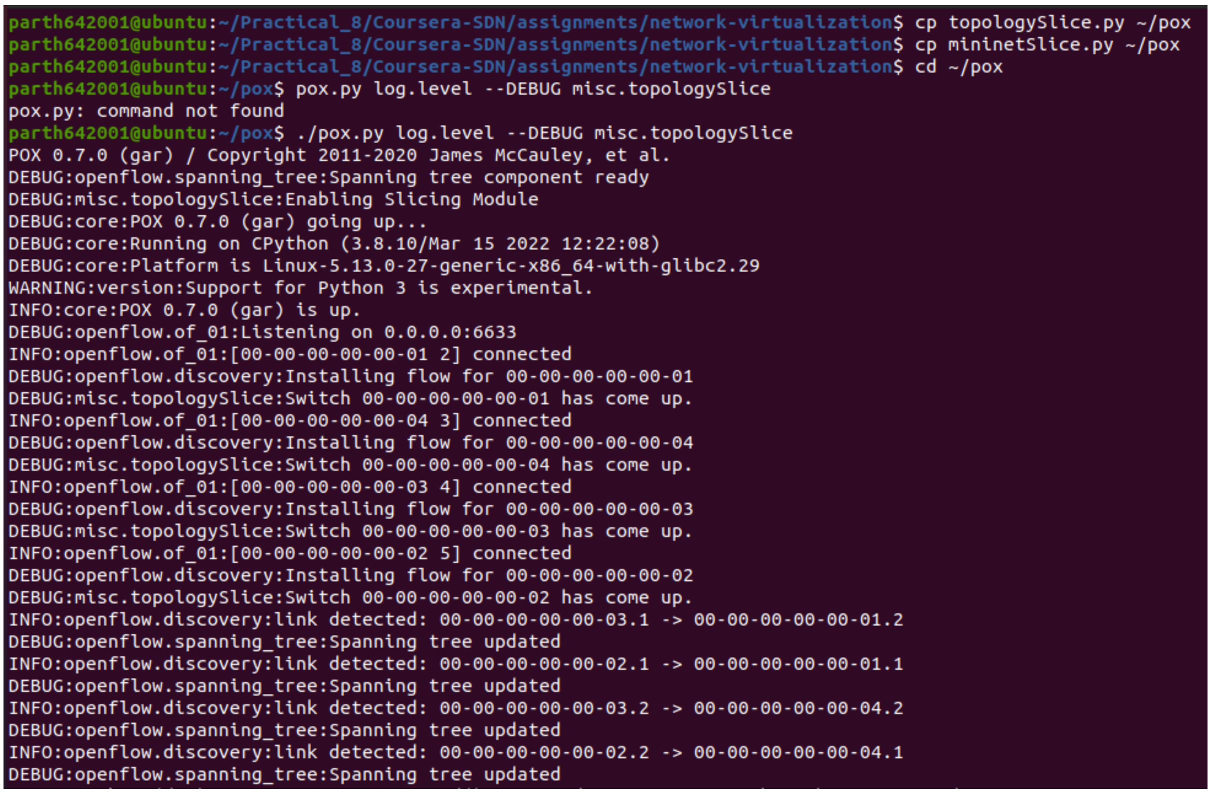
Part 1: Topology-based Slicing

- The goal of this part is to divide the network into two separate slices: upper and lower, as shown below. Users in different slices should not be able to communicate with each other. In practice, a provider may want to subdivide the network in this fashion to support multi-tenancy (and, in fact, the first part of this assignment provides similar functionality as ordinary VLANs).



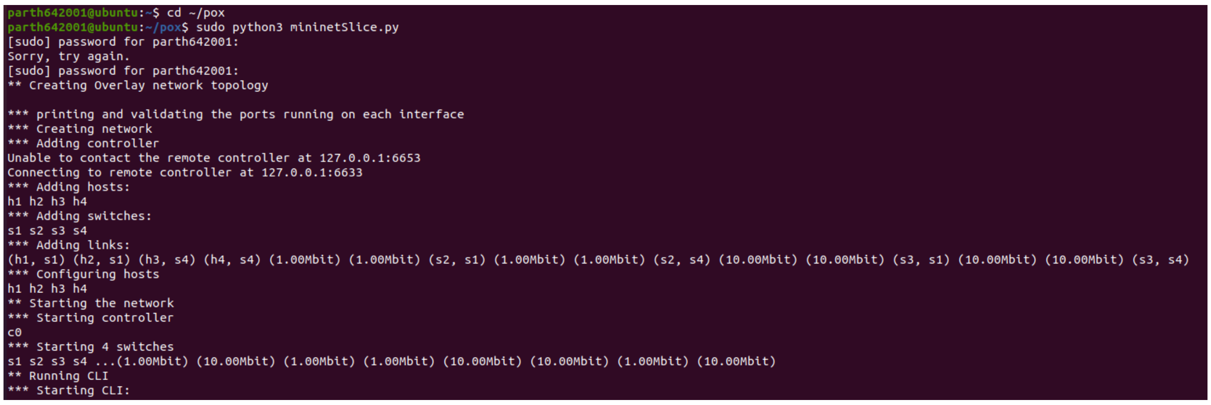


- The topologySlice.py file provides skeleton for this implementation. As in the previous assignment, we have implemented a launch() function, which registers a new component. The_handle_ConnectionUp() callback is invoked whenever a switch connects to the controller.
- We also launch the discovery and spanning tree modules, which compute a spanning tree on the topology, thus preventing any traffic that is flooded from looping (since the topology itself has a loop in it, computing a spanning tree is required)



```
parth642001@ubuntu:~/Practical_8/Coursera-SDN/assignments/network-virtualization$ cp topologySlice.py ~/pox
parth642001@ubuntu:~/Practical_8/Coursera-SDN/assignments/network-virtualization$ cp mininetSlice.py ~/pox
parth642001@ubuntu:~/Practical_8/Coursera-SDN/assignments/network-virtualization$ cd ~/pox
parth642001@ubuntu:~/pox$ pox.py log.level --DEBUG misc.topologySlice
pox.py: command not found
parth642001@ubuntu:~/pox$ ./pox.py log.level --DEBUG misc.topologySlice
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
DEBUG:openflow.spanning_tree:Spanning tree component ready
DEBUG:misc.topologySlice:Enabling Slicing Module
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.8.10/Mar 15 2022 12:22:08)
DEBUG:core:Platform is Linux-5.13.0-27-generic-x86_64-with-glibc2.29
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-01
DEBUG:misc.topologySlice:Switch 00-00-00-00-00-01 has come up.
INFO:openflow.of_01:[00-00-00-00-00-04 3] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-04
DEBUG:misc.topologySlice:Switch 00-00-00-00-00-04 has come up.
INFO:openflow.of_01:[00-00-00-00-00-03 4] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-03
DEBUG:misc.topologySlice:Switch 00-00-00-00-00-03 has come up.
INFO:openflow.of_01:[00-00-00-00-00-02 5] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-02
DEBUG:misc.topologySlice:Switch 00-00-00-00-00-02 has come up.
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.1 -> 00-00-00-00-00-01.2
DEBUG:openflow.spanning_tree:Spanning tree updated
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.1 -> 00-00-00-00-00-01.1
DEBUG:openflow.spanning_tree:Spanning tree updated
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.2 -> 00-00-00-00-00-04.2
DEBUG:openflow.spanning_tree:Spanning tree updated
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.2 -> 00-00-00-00-00-04.1
DEBUG:openflow.spanning_tree:Spanning tree updated
```

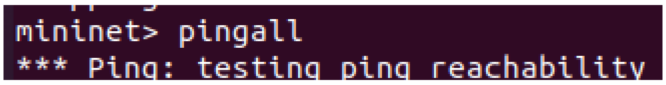
- In a separate terminal, launch your Mininet script.



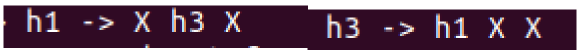
```
parth642001@ubuntu:~$ cd ~/pox
parth642001@ubuntu:~/pox$ sudo python3 mininetSlice.py
[sudo] password for parth642001:
Sorry, try again.
[sudo] password for parth642001:
** Creating Overlay network topology

*** printing and validating the ports running on each interface
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** 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) (s2, s1) (1.00Mbit) (1.00Mbit) (s2, s4) (10.00Mbit) (10.00Mbit) (s3, s1) (10.00Mbit) (10.00Mbit) (s3, s4)
*** Configuring hosts
h1 h2 h3 h4
** Starting the network
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...(1.00Mbit) (10.00Mbit) (1.00Mbit) (1.00Mbit) (10.00Mbit) (10.00Mbit) (1.00Mbit) (10.00Mbit)
** Running CLI
*** Starting CLI:
```

- Now, verify that the hosts in different slices are not able to communicate with each other.

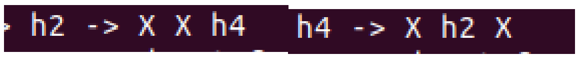


```
mininet> pingall
*** Ping: testing ping reachability
```



```
h1 -> X h3 X     h3 -> h1 X X
```

- H1 can communicate with h3 and vice versa as they are in same slice

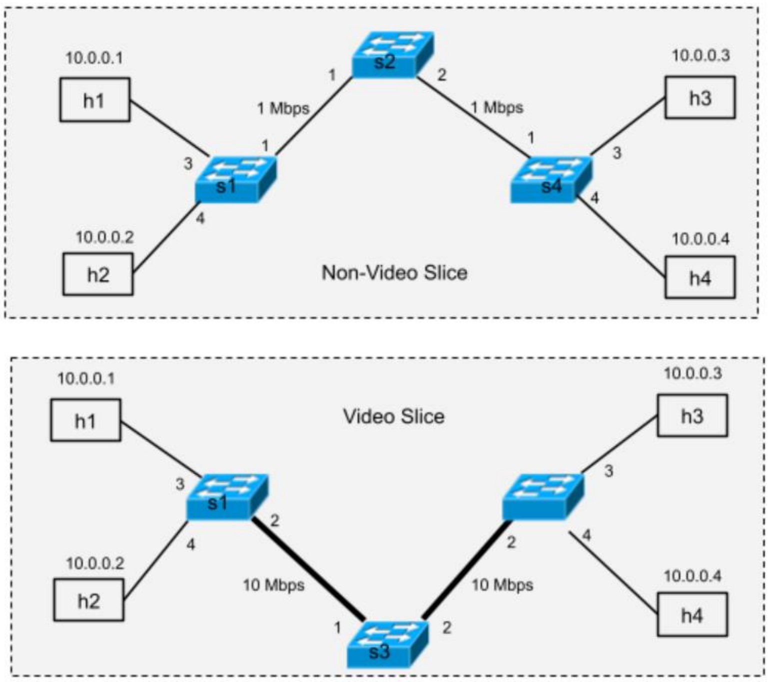


```
h2 -> X X h4   h4 -> X h2 X
```

- H2 and H4 can communicate

Part 2: Flowspace Slicing

- In the previous part of the assignment, you learned how to slice a network based on the network's physical topology. It is also possible to slice the network in more interesting ways, such as based on the application that is sending the traffic. SDN networks in principle can be sliced on any attributes of flowspace.
- Recall that the topology has two paths connecting two sites: a high-bandwidth path and a low-bandwidth path. Suppose that you want to prioritize video traffic in our network by sending all the video traffic over the high bandwidth path, and sending all the other traffic over the default low bandwidth path. File transfers won't be affected by the video traffic, and vice versa. In this part of the assignment, you'll use network slicing to implement this isolation.
- Let's assume for simplicity that all video traffic goes on TCP port 80. In this assignment you are required to write the logic to create two slices, "video" and "non-video", as shown below



- Launch the POX controller

```
~/pox/pox/misc$ pox.py log.level --DEBUG misc.videoSlice
```

- In a separate terminal, launch your Mininet script.

```
~/pox/pox/misc$ sudo python mininetSlice.py
```

- Wait until the application indicates that the OpenFlow switch has connected. Now verify that the all the hosts are able to communicate with each other. This is a simple sanity check to make sure that your logic is not affecting connectivity in general.
- You should see this output:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped 12/12 received
```

CONCLUSION:

- In this practical, we Understand Network Virtualization and FlowVisor – Slice in mininet.