Lab 3: SDN Simulation

# Objectives

* Fully understand the operation of Openflow and observe the operations
* Master the simulation tool mininet

# Equipment Needs

* Computers
* Internet

# Experiments

## Basics

1. Install Ubuntu on your computer, you can install it on a VM using either VMWare player or VirtualBox:  
   VMWare: [https://my.vmware.com/web/vmware/free#desktop\_end\_user\_computing/vmware\_player/6\_0](https://my.vmware.com/web/vmware/free)  
   VirtualBox:   
   <https://www.virtualbox.org/wiki/Downloads>
2. Follow the instructions to set up Mininet on your Ubuntu VM:  
   <http://mininet.org/download/>
3. Perform basic simulations following these steps:

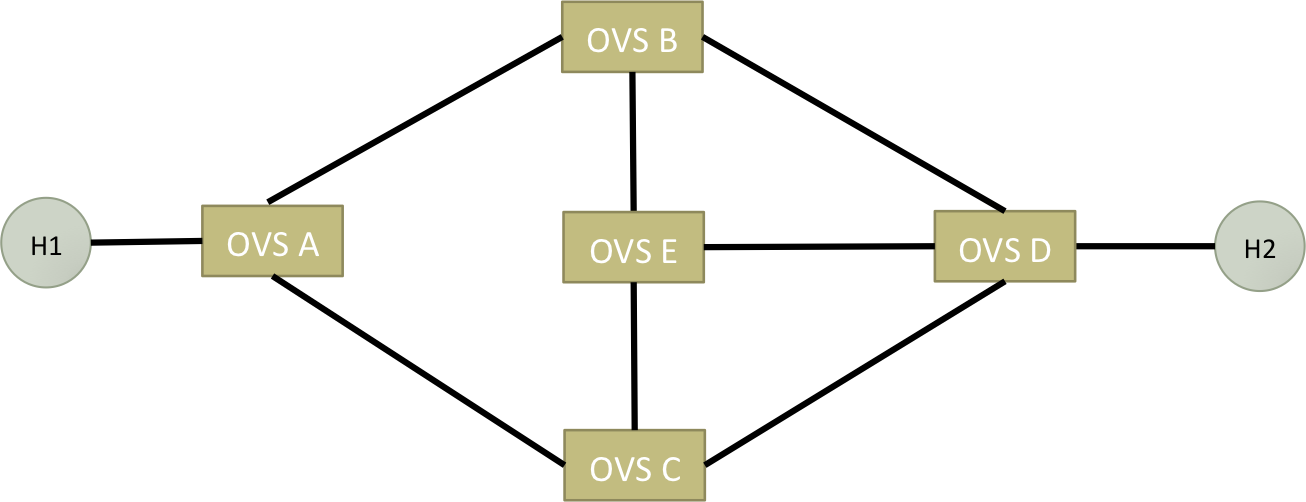
<http://mininet.org/walkthrough>

1. Practice writing simulation with python scripts:  
   <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>

## Openflow

1. Use Mininet to create the topology below, where A, B, C, D, and E are all Openflow switches.
2. Enforce the following policies so that,
   1. Traffic from H1 🡺 H2
      1. HTTP traffic with d\_port=80 follows path: A-C-D
      2. other traffic follows path: A-B-E-D
   2. Traffic from H2 🡺 H1
      1. HTTP traffic with s\_port=80, follow path: D-B-A
      2. other traffic, follow path: D-C-E-B-A
   3. verify your policies by,
      1. generating corresponding traffic
      2. capturing packets with Wireshark

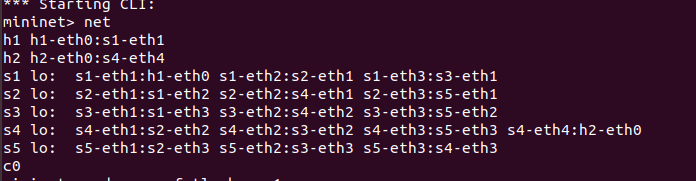
You can use OVS-OFCTL to manually install rules on switches (preferred method), or you can install a simple controller using RYU/POX/NOX/Beacon (Not recommended for lab 3 you’ll do it in lab 4).



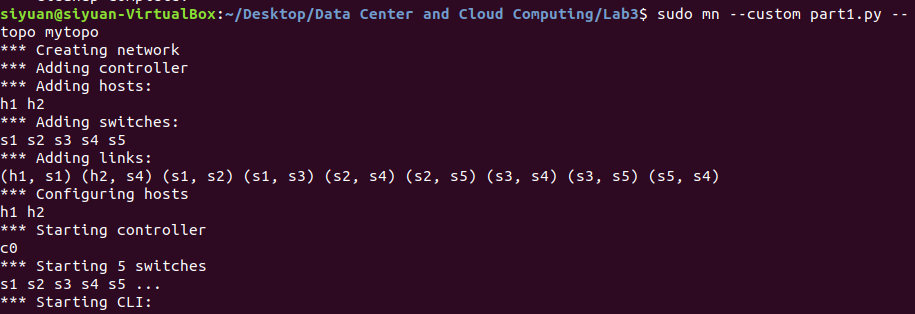
1. Using mininet, create a 2-stage Fat Tree network using N-port switches, where N is an input parameter to your python script running Mininet simulation. N should be an even number. (You don’t need to check the connectivity, just create the topology.)

# Reports

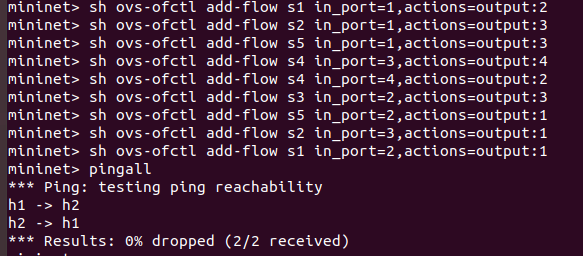
1. A screenshot of OpenFlow control messages you captured with WireShark.
2. Output of mininet “net” command for both topologies. (you can use any N for Fat-tree, ex: 4, 6, 8)



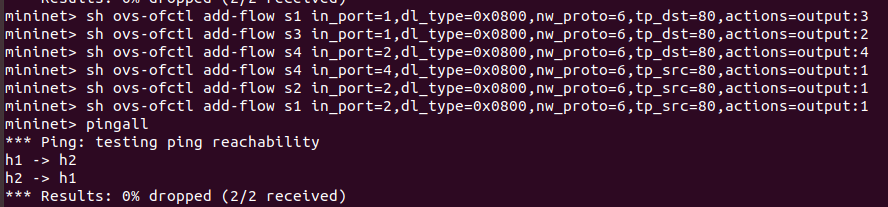
1. Mininet output while creating the networks



1. Briefly explain how you produce different traffic to verify whether the rules installed function correctly.
2. With the produced traffic, show the screenshots of Wireshark capture on different links (switch with interface) to verify the paths taken by different traffic are correct.
3. OVS-OFCTL commands used to install the rules on switches. (If you use a controller, upload your controller program)



* 1. Traffic from H1 🡺 H2
     1. other traffic follows path: A-B-E-D
  2. Traffic from H2 🡺 H1
     1. other traffic, follow path: D-C-E-B-A



* 1. Traffic from H1 🡺 H2
     1. HTTP traffic with d\_port=80 follows path: A-C-D
  2. Traffic from H2 🡺 H1
     1. HTTP traffic with s\_port=80, follow path: D-B-A

1. Also submit all your python files used in this lab (do NOT paste code in report).

**We have zero tolerance to forged or fabricated data!!** A single piece of forged/fabricated data would bring the total score down to zero.