Section 3: Introduction to Mininet

—

Simulate a simple network using Mininet

Administrivia

- Assignment 3 due Tuesday Feb 2
- Project 2 due Wednesday Feb 16
- Midterm is coming next Friday (Feb 4)!

Before we start...

1. Download VirtualBox: https://www.virtualbox.org/wiki/Downloads

2. Download Vagrant: https://www.vagrantup.com/downloads

(We expect that you have already downloaded and installed VirtualBox and Vagrant!)

Software Defined Network (SDN)

Wikipedia:

(SDN) technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring making it more like cloud computing than traditional network management.

Controller

Switch

Control
Plane

OpenFlow
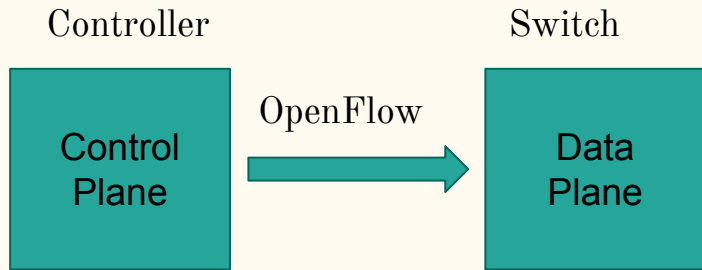
Data
Plane

Make Decisions:
1. How to forward data.
2. Order to send data.

Perform Actions:
1. Forward
2. Route

What is Mininet & Pox?

- **Mininet**: A software that creates a virtual network. We can set up switches, connect components, and ping, etc.

- **Pox**: A Python-based SDN controller platform geared towards research and education. You will be using it to set up rules on the Mininet switches in parts 2-4.

- **Helpful Links**: See the last slide.

Part 1: Build A Simple Network
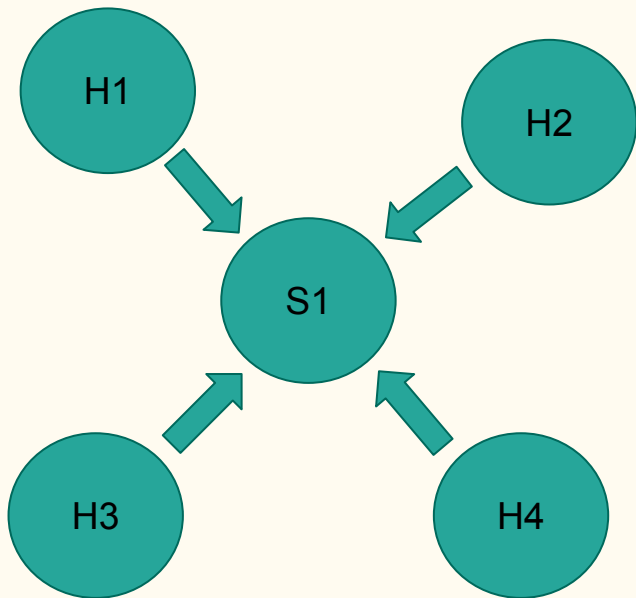
Goal of part 1:
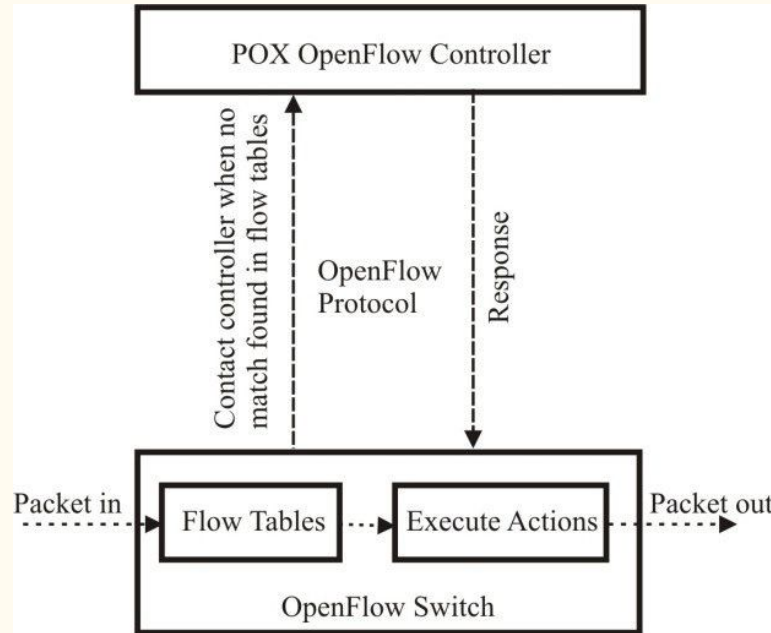Modify file part1.py to create the following topology.

Run part 1:
$ sudo python 461_mininet/topos/part1.py
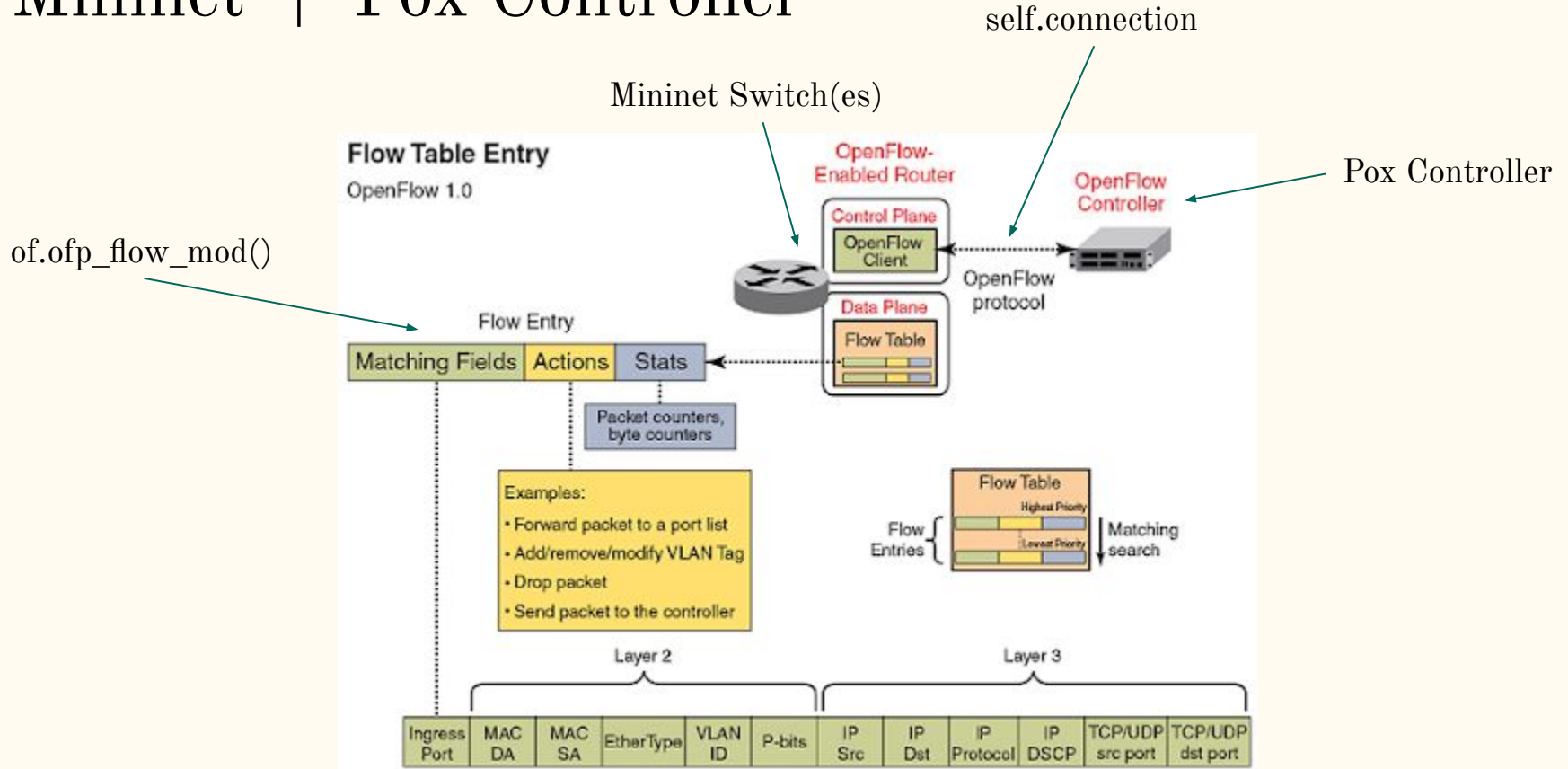
Some useful Mininet commands:

1. dump (dump info about all nodes)
2. pingall (Literally, ping all the connections)
3. ping (ping the connection h1 and h2)
4. iperf (Test the bandwidth between two hosts)

Mininet + Pox Controller

In parts 2-4 you using a Pox Controller to manage Mininet switches.

Mininet + Pox Controller

self.connection

Mininet Switch(es)

Pox Controller

of.ofp_flow_mod()

Flow Table Entry
OpenFlow 1.0

OpenFlow-Enabled Router

Control Plane
OpenFlow Client

Data Plane
Flow Table

OpenFlow Controller

OpenFlow protocol

Flow Entry

Matching Fields	Actions	Stats

Packet counters, byte counters

Examples:
• Forward packet to a port list
• Add/remove/modify VLAN Tag
• Drop packet
• Send packet to the controller

Flow Table
Highest Priority
Lowest Priority

Flow Entries

Matching search

Layer 2   Layer 3

Ingress Port	MAC DA	MAC SA	EtherType	VLAN ID	P-bits	IP Src	IP Dst	IP Protocol	IP DSCP	TCP/UDP src port	TCP/UDP dst port

Part 2: Firewall

- Install Firewall rules to accept/drop packets
- Launch controller with `$ sudo ~/pox/pox.py misc.part2controller`
- Run network topo with `$ sudo python ~/461_mininet/topos/part2.py`
- Make sure that controller and topo are running at the same time!
- Look for constant definitions in ~/pox/pox/lib/packet

src ip	dst ip	protocol	action
any ipv4	any ipv4	icmp	accept
any	any	arp	accept
any ipv4	any ipv4	-	drop

Part 3: L2 Switch

- Install forwarding rules to forward packets to its destination
  - No flooding for the central switch!
  - Other switches can continue to flood packets
- Launch controller with `$ sudo ~/pox/pox.py misc.part3controller`
- Run network topo with `$ sudo python ~/461_mininet/topos/part3.py`

```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/      |
                                 |
                    [hnotrust1@172.16.10.100/24]
```

Part 4: Learning Router

- Router has to "learn" which switch is at which port
  - No hardcoding!
- By looking at ARP messages!
- Launch controller with $ sudo ~/pox/pox.py misc.part4controller
- Run network topo with $ sudo python ~/461_mininet/topos/part4.py

```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                  [hnotrust1@172.16.10.100/24]
```

Mininet: Download & Install

1. Download VirtualBox: https://www.virtualbox.org/wiki/Downloads

1. Download Vagrant: https://www.vagrantup.com/downloads

1. Clone/download: https://github.com/kaichengyan/mininet-vagrant

1. Follow the following commands to complete mininet setup:

```
$ cd mininet-vagrant          # go to the mininet-vagrant dir
$ vagrant up                  # setup the vm (takes a few minutes)
$ vagrant ssh                 # ssh into the vm
$ sudo mn --test pingall      # test mininet has been correctly installed
```

Some Errors and Solutions

- "Error creating interface pair... File exists" or "Please shut down the controller which is running on port..."
  - Run `sudo mn -c`
- VBoxManage.exe: error... VM_CREATE_FAILED)
  - Downgrade to VirtualBox 6.1.26
- VBoxManage.exe: error... interface IConsole
  - Disable Windows Subsystem for Linux and Hyper-V
- Mininet not a python module
  - Install pip3 (sudo apt-get install python3-pip) and then run `sudo pip3 install mininet`

(Optional) Visual Studio Code setup

1. Run `vagrant ssh-config` from your `mininet-vagrant` directory
2. Add the output from running above command to your `~/.ssh/config` file. You can rename the host as you like (change the first line to `Host mininet`)
3. Install the "Remote Development" extension in VS Code
4. Click on the "><" button in the very bottom left of the VS Code window
5. In the pop-up menu, select Remote-SSH: Connect to Host
6. Here, you should be able to connect to the `mininet` host.
7. Optionally, for auto completion, install the Python and Pylance extensions on the remote VM, and configure python.autoComplete.extraPaths to include "/home/vagrant/mininet" "/home/vagrant/pox" and "/home/vagrant/openflow"
8. Start developing!

Walk Through Setup and Part 1

Watch how the TA creates the network and try it yourself!

Start early!

It is not hard, but you will probably spend a lot time looking for and reading documentation...

Resources About Mininet & Pox

Mininet:

➤ https://github.com/mininet/mininet/wiki/Documentation

➤ https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#creating

Pox Wiki and API docs:

➤ https://noxrepo.github.io/pox-doc/html/#id97

Pox OpenFlow Tutorials:

➤ https://github.com/mininet/openflow-tutorial/wiki/Create-a-Learning-Switch#Controller_Choice_POX_Python

➤ https://haryachyy.wordpress.com/2014/06/14/learning-pox-openflow-controller-proactive-approach/