Lab 1: Installing and Using Mininet

Part 1: Installing Virtualbox

The first thing required for mininet is a Virtual Machine (VM) manager to run our mininet VM. VirtualBox is free and open source and can be downloaded here. You are free to use kvm, vmware, or xen if you are familiar with those hypervisors - however, the TAs will not support them, so you are on your own.

If you are having trouble installing VirtualBox, make sure to consult the VirtualBox manual, Piazza, or the TAs. It is recommended that you **start the lab early** in case you encounter any problems setting up either VirtualBox or Mininet. If you do not have a computer, please contact the TAs and we can work something out.

Common Problems Installing VirtualBox:

- Did you download the correct version? (32-bit vs 64-bit)
- Is your computer really old? It might not be able to be virtualized -- talk to the TA.
- Make sure in your BIOS that virtualization is enabled.

Part 2: Installing Mininet

Once VirtualBox is installed, we can install the mininet VM. You can find the mininet VM on github: https://github.com/mininet/mininet/wiki/Mininet-VM-Images. For Windows users, mininet suggests that you download the 32-bit version (764 MB). If your Operating System supports 64-bit, you can download the 64-bit version (974 MB). Both versions are cached locally on the webpage. If accessing from on campus, it's recommended that you use the cached version, as it will download faster.

Using a GUI:

- Once the download is complete and the .ovf and .vmdk have been extracted, you can import the VM into VirtualBox. To do this, navigate to File > Import Appliance... Select the extracted files that you previously downloaded. The defaults are acceptable.
- When the VM has been successfully imported, start the machine. The username is **mininet**, and the password is **mininet**.
- We can now set up a GUI environment. Run the following commands:
    a. sudo apt-get update
    b. sudo apt-get install xinit lxde virtualbox-guest-dkms
- When the downloads are complete, shut down the VM.
- Right click on the VM, select Settings > Network > Adapter 2. Enable the adapter, and select Host-Only Adapter.
    a. Note: If you run into trouble on this step (VirtualBox says Invalid Configuration, and in the Name: dropdown box you only see "None"), you must create a Host-Only Network. To do this, select File > Preferences > Network > Host Only Networks from the VirtualBox main window. Press the Green + icon to the right of the window. When this is completed, you should be able to successfully add the Host-Only Adapter to the VM.

- Notes for using a GUI:
    a. To access the UI, type **startx** at the terminal.
    b. To type commands, you will use XTerm. This is available from the Main Menu > System Tools > XTerm, or the keyboard shortcut Ctrl + Alt + T.

Using X11 Forwarding:

- Follow Lincoln's Guide. The YouTube movies referenced are very helpful, and available for both Windows and Mac.
- Ensure that you set up Host-Only Adapter as mentioned in the final step of the GUI walkthrough above.

Part 3: Using Mininet

A walkthrough can be found on the mininet page. First and foremost, background and information on why we are using Mininet over the physical lab in BE301A.

What is Mininet? Mininet is a "network in a box", developed at Stanford in 2010. It is software designed to allow large scale networks to be emulated in software on a laptop. Its rise has also been dictated by the use of OpenFlow (which we will the subject of Lab 4 and the Final Project). If you are interested in reading, here is the original Mininet paper.

Why Mininet? Our physical lab, as you have seen is deteriorating from the wear and tear of 11 years (donated by Cisco in 2004). Mininet is probably one of the simplest forms of emulating networks, it is *free*, it is open source, and it is widely used by the academic community, as well as being used by universities for teaching computer networks. It allows for more interesting topologies than what can be achieved in the physical lab. Most importantly is the growing size of CMPE150, five years ago the course was around 40 students, today it is peaking at 100. The physical lab only has 10 workstations, however certain workstations are lacking parts (which either have not been or will be replaced) leaving even fewer usable stations per lab. So in light of the growing student base we are looking towards a solution which scales with the number of students. Almost all students are guaranteed to have a personal workstation, so instead of buying hundreds of thousands of dollars worth of equipment that will only be utilized by 300 students (max) per quarter, we want to utilize your existing hardware (as a side note: if you don't have a personal computer please talk to the professor or TA about Amazon's EC2 and using the BE109 lab).

How does Mininet Work? Mininet works simply by creating a virtual network on your computer/laptop. It accomplishes this task by creating host namespaces (h1, h2, etc) and connecting them through virtual interfaces. So when we run ping between the linux namespaces h1 and h2, the ping will run from h1's namespace through a virtual interface pair created for h1 and h2, before it reaches h2. If h1 and h2 are connected through a switch as done with the example python code, the ping will transit multiple virtual interface pairs. The switches that we will be using are running OpenVSwitch (OVS). Mininet will connect additional virtual interfaces between each virtual port on the switch with each connected host. The host name space allows each host to see the same file system, but operates as its own process that will run separately then each of the other host processes. The OVS version running on the Ubuntu image supports OpenFlow.

Understanding some mininet commands:

1. *sudo mn*: will start mininet

```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

2. You can use *mn -h* or type *help* after you have run mininet net.

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      x
exit   iperf  net       pingallfull  px            source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2
```

3. We can see that when we launched mininet, it created a mini-network from the output. When we use the *net* command we can see *h1* indicated host 1, it has one network interface *eth0* which is connected to the switch on interface *eth1*. This is shown on the output line: *h1 h1-eth0:s1-eth1*. There is also host 2 (*h2*), switch 1 (*s1*), and controller 0 (*c0*).

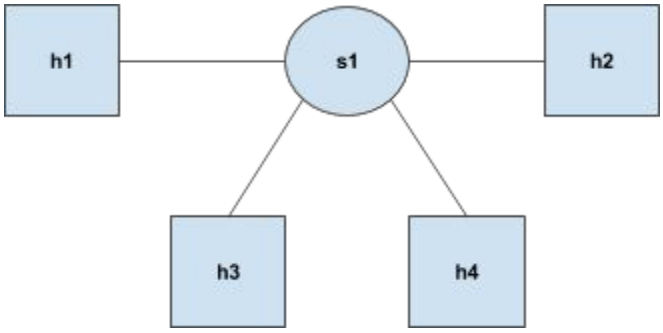
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1049>
<Host h2: h2-eth0:10.0.0.2 pid=1052>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1057>
<Controller c0: 127.0.0.1:6633 pid=1042>
```

   a. From dump we can also see what IP address have been assigned to h1 and h2. We also are given the pids of each process. Processes in mininet are used for hosts, switches, and controllers. Mininet is composed of processes using Interprocess Communication (IPC) to

emulate a network environment. You can review the prelab's resources and reading to understand more.

The Lab [100 pts]:

1. In Mininet change the default configuration to have 4 hosts connected to a switch. [*hint*,*hint*,*hint*]



2. [30 pts] Save a screenshot of *dump* and *pingall* output.
3. [10 pts] Run the *iperf* command as well, and screenshot the output, how fast is the connect?
4. Either through X11 Forwarding or the GUI:
    a. Run wireshark, and using the display filter, filter for "of". Note: When you run wireshark you should do so as "sudo wireshark". When you choose an interface to capture on, you should select "any".
    b. [20 pts] Run ping from a host to any other host using *hX ping -c 5 hY*. How many *of_packet_in* messages show up?
    c. [20 pts] What is the source and destination IP addresses for these entries? Find another packet that matches the "of" filter with the OpenFlow typefield set to *OFPT_PACKET_OUT*. What is the source and destination IP address for this entry?
    d. Replace the display filter for "of" to "icmp && not of"
    e. [20 pts] Run *pingall* again, how many entries are generated in wireshark?
        i. What types 2 types of icmp entries show up?

Submit your screenshots inline (in the document) with your answers. You must submit as a PDF file. No other files will be accepted.