

Mininet

Prof. Ankit Gangwal

Centre for Security, Theory, and Algorithmic Research (CSTAR),
IIIT Hyderabad, India.

Agenda

- Comparison of Various Network Testing Platform
- Introduction to Mininet
- Mininet - Comparison
- Installing Mininet
 - ① Mininet VM Installation
 - ② Native Installation from Source
 - ③ Installation from Package
- Preparing Mininet
- Running Mininet
- Mininet Commands
- Mininet Other Utilities
- References
- Warm-up Exercises

Platforms for Network Testing

Platforms	Advantages	Disadvantages
Hardware Testbed	Fast. Accurate.	Expensive. Hard to reconfigure. Hard to change.
Simulators	Inexpensive. Flexible. Easy to download.	May not be "Believable". May be slow.
Emulators	Inexpensive. Flexible. Easy to download. Reasonably accurate.	Slower than hardware. Experiments may not fit. Possibility of inaccuracy from multiplexing.

Introduction to Mininet

- Mininet is an **Open Source Project**.
- Mininet depends on the **Linux Kernel**.
- Mininet is a **Network Emulator**.
- Mininet is used to develop, share & experiment with **OpenFlow & SDN**.
- Mininet experiments are **Python Scripts**.

Introduction to Mininet

- It makes a **single system** look like a **complete network** that includes end-hosts, switches, routers & links on a single Linux kernel.
- Mininet host behaves as a real machine, where you can **SSH** into it & run arbitrary programs.
- Mininet uses **process-based virtualisation** to run many (upto **4096**) hosts and switches on a single OS kernel.
- You can create **custom** topologies.

Introduction to Mininet

- Programs can send packets through virtual Ethernet interface, with given link speed & delay.
- Packets are processed by virtual Ethernet switch, router, or middle-box, with given amount of queueing.
- You can **customize** packet forwarding as Mininet's switches are programmable using the OpenFlow protocol.
- For custom routing or switching behaviour separate OpenFlow controller must be developed for required features.
- Mininet doesn't do NAT out of the box. This means that your virtual hosts will be **isolated** from your LAN by default.

Comparing Mininet

Mininet combines many of the best features of emulators, hardware testbeds, and simulators.

Compared to full system virtualization based approaches, Mininet

- **Boots faster:** seconds instead of minutes.
- **Scales larger:** hundreds of hosts and switches.
- **Provides more bandwidth:** typically 2Gbps total bandwidth on modest hardware.
- **Installs easily:** Apart from pre-built VM image, Mininet Ubuntu Package is also available.

Comparing Mininet

Compared to hardware testbeds, Mininet is

- **inexpensive.**
- **quickly reconfigurable and restart-able.**

Compared to simulators, Mininet

- easily **connects to real networks.**
- offers **interactive performance** - you can type at it.

Limitation

- Mininet cannot (currently) run **non-Linux-compatible** OpenFlow switches or applications; this has not been a major issue in practice.

Installing Mininet

- Go to <http://mininet.org/download/>
- Option 1: Mininet VM Installation (easy, recommended)
- Option 2: Native Installation from Source
- Option 3: Installation from Packages

Option 1: Mininet VM Installation

- ① Download the **Mininet VM image** @ <https://github.com/mininet/mininet/wiki/Mininet-VM-Images>
 - Go under **Mininet 2.2.0 on Ubuntu 14.04** & select:
Ubuntu 14.04 - 32 bit (recommended for Windows users using VirtualBox or Hyper-V)
- ② Download and install **VirtualBox** @ <https://www.virtualbox.org/wiki/Downloads>
 - Go under **VirtualBox platform packages** & select appropriate version for your host OS.

Option 1: Mininet VM Installation

- ③ Download X server & Terminal for your host OS @ <https://github.com/mininet/openflow-tutorial/wiki/Installing-Required-Software>

OS Type	OS Version	Virtualization Software	X Server	Terminal
Windows	7+	VirtualBox	Xming	PuTTY
Windows	XP	VirtualBox	Xming	PuTTY
Mac	OS X 10.7 - 10.9 Lion/Mountain Lion/Mavericks	VirtualBox	download & install XQuartz	Terminal.app
Mac	OS X 10.5-10.6 Leopard/Snow Leopard	VirtualBox	X11 (install from OS X main system DVD, preferred), or download XQuartz	Terminal.app
Linux	Ubuntu 10.04+	VirtualBox	X server already installed	gnome terminal + SSH

Option 1: Mininet VM Installation

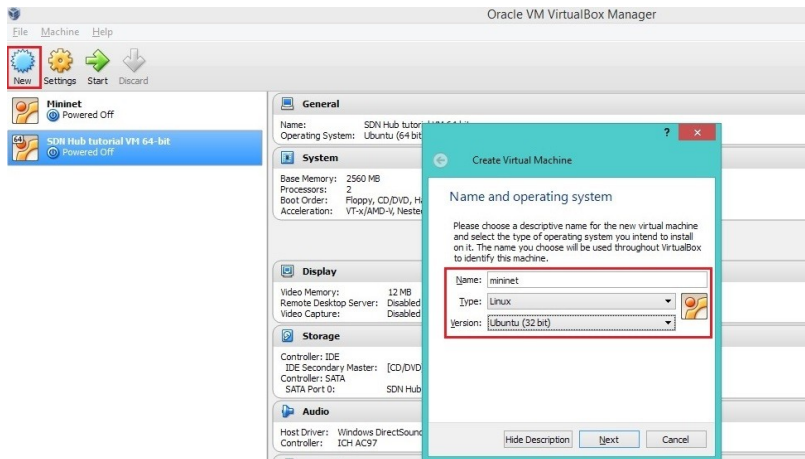
- ④ Extract the Mininet image you downloaded in step1.
- ⑤ Install the X server & launch it.
- ⑥ In case of windows copy the putty.exe to the folder where the command prompt starts up. In this case “c:\users\ankit”.



A screenshot of a Windows Command Prompt window. The title bar reads "C:\Windows\system32\cmd.exe". The window content shows the following text:
Microsoft Windows [Version 6.3.9600]
<C> 2013 Microsoft Corporation. All rights reserved.
C:\Users\Ankit>

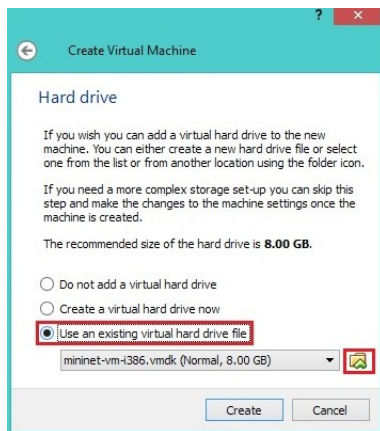
Option 1: Mininet VM Installation

- Launch VirtualBox; select New & Name your VM; Type ->Linux; Version ->Ubuntu (32 bit).



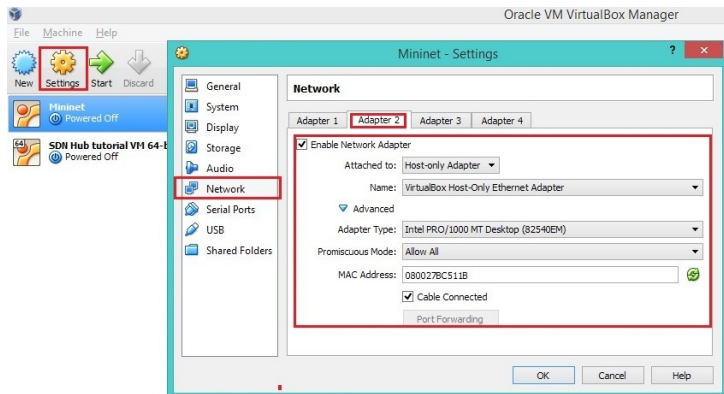
Option 1: Mininet VM Installation

- 8 Allocate RAM 512 MB is Good.
- 9 Next “Select Use an existing virtual hard drive file” & locate .vmdk file. Then click “Create”.



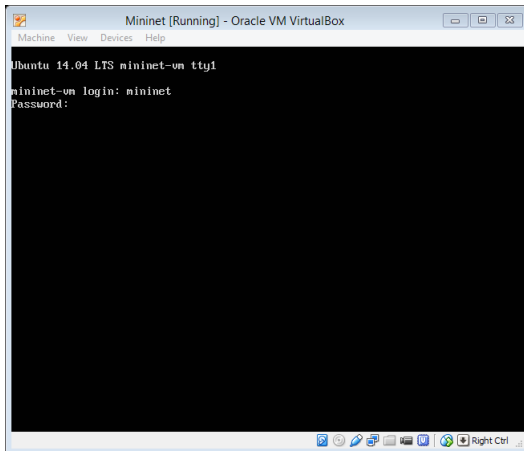
Option 1: Mininet VM Installation

- 10 Click on Settings -> Network -> Adapter 2 -> check "Enable Network Adapter" -> Attached to "Host-only Adapter" -> Advanced -> Promiscuous Mode -> "Allow All".



Option 1: Mininet VM Installation

- 11 Double click on VM name or click on Start to boot.
- 12 Username & Password is “mininet”.



Option 2: Native Installation from Source

1. To install natively from source, first get the source code:

```
$ git clone git://github.com/mininet/mininet
```

****Note that the above command will check out the latest Mininet. If you want to run any other version, you may checkout that version explicitly:**

- **\$ cd mininet**
- **\$ git tag** /*lists available versions*/
- **\$ git checkout -b 2.2.1 2.2.1** /*or other version required*/
- **\$ cd ..**

2. Once you have the source tree, install Mininet with:

```
$ mininet/util/install.sh [options]
```

Option 2: Native Installation from Source

****Typical *install.sh* options include:**

- To install everything (using your home directory):
install.sh -a
- To install everything (using another directory):
install.sh -s mydir -a
- To install Mininet + user switch + OVS (using your home dir):
install.sh -nfv
- To install Mininet + user switch + OVS (using other dir):
install.sh -s mydir -nfv
- For other options:
install.sh -h

3. After the installation, test the basic Mininet functionality:

\$ sudo mn --test pingall

Option 3: Installation from Packages

1. Remove any traces of earlier versions of Mininet and Open vSwitch from `/usr/local/`:

```
sudo rm -rf /usr/local/bin/mn /usr/local/bin/mnexec \  
sudo rm -rf /usr/local/lib/python*/**mininet* \  
sudo rm -rf /usr/local/bin/ovs-* /usr/local/sbin/ovs-*
```

2. Confirm which OS version you are running, run the command:

```
lsb_release -a
```

Option 3: Installation from Packages

3. Install the base Mininet package:

Mininet 2.1.0 on Ubuntu 13.10:

```
sudo apt-get install mininet
```

Mininet 2.0.0 on Ubuntu 13.04:

```
sudo apt-get install mininet
```

Mininet 2.0.0 on Ubuntu 12.10:

```
sudo apt-get install mininet/quantal-backports
```

Mininet 2.0.0 on Ubuntu 12.04:

```
sudo apt-get install mininet/precise-backports
```

4. After this deactivate openvswitch-controller if it is running:

```
sudo service openvswitch-controller stop
```

```
sudo update-rc.d openvswitch-controller disable
```

Option 3: Installation from Packages

5. Test Mininet :

```
sudo mn --test pingall
```

6. If Mininet complains that Open vSwitch isn't working:

```
sudo dpkg-reconfigure openvswitch-datapath-dkms  
sudo service openflow-switch restart
```

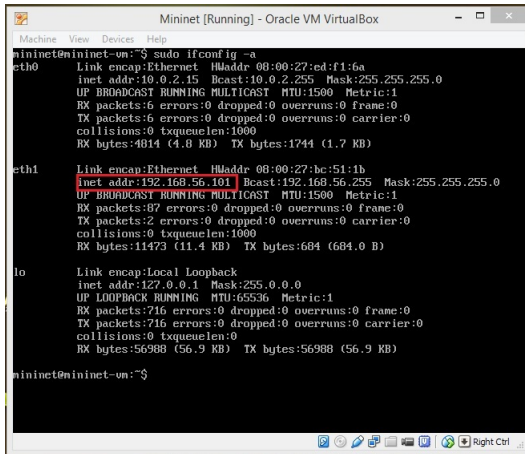
7. To install additional software:

```
git clone git://github.com/mininet/mininet  
mininet/util/install.sh -fw
```

- **Only VM based users need to follow preparation steps !!!**
- **With VM booted & logged in:**
 - 1 List all the network interfaces installed on the system:
sudo ifconfig -a
 - 2 Assign an address (e.g. for **eth1** interface) run:
sudo dhclient eth1

Preparing Mininet

- ③ Repeat step 1 & note “inet addr” for eth1.



```
Mininet [Running] - Oracle VM VirtualBox
Machine View Devices Help
mininet@mininet-vm:~$ sudo ifconfig -a
eth0      Link encap:Ethernet  HWaddr 08:00:27:ed:f1:6a
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4014 (4.0 KB)  TX bytes:1744 (1.7 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:bc:51:1b
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:87 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11473 (11.4 KB)  TX bytes:684 (684.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:716 errors:0 dropped:0 overruns:0 frame:0
          TX packets:716 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:56988 (56.9 KB)  TX bytes:56988 (56.9 KB)

mininet@mininet-vm:~$
```

Preparing Mininet

- ④ Windows users : From Cmd run putty.exe with eth1's inet addr. Password : "mininet".

putty.exe -X mininet@192.168.56.101

```
mininet@mininet-vm: ~  
Using username "mininet".  
mininet@192.168.56.101's password:  
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)  
  
* Documentation: https://help.ubuntu.com/  
Last login: Wed Dec 17 18:19:42 2014  
mininet@mininet-vm:~$  
  
C:\Windows\system32\cmd.exe  
Microsoft Windows [Version 6.3.9600]  
(c) 2013 Microsoft Corporation. All rights reserved.  
C:\Users\Ankit>putty.exe -X mininet@192.168.56.101  
C:\Users\Ankit>  
  
Mininet [Running] - Oracle VM VirtualBox  
Machine View Devices Help  
mininet@mininet-vm:~$ sudo ifconfig -a  
eth0 Link encap:Ethernet HWaddr 08:00:27:cd:f1:6a  
inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:6 errors:0 dropped:0 overruns:0 frame:0  
TX packets:6 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:4814 (4.8 KB) TX bytes:1744 (1.7 KB)  
  
eth1 Link encap:Ethernet HWaddr 08:00:27:bc:51:1b  
inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:134 errors:0 dropped:0 overruns:0 frame:0  
TX packets:37 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:18050 (18.0 KB) TX bytes:7429 (7.4 KB)  
  
lo Link encap:Local Loopback  
inet addr:127.0.0.1 Mask:255.0.0.0  
UP LOOPBACK RUNNING MTU:65536 Metric:1  
RX packets:800 errors:0 dropped:0 overruns:0 frame:0  
TX packets:800 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:0  
RX bytes:63696 (63.6 KB) TX bytes:63696 (63.6 KB)  
  
mininet@mininet-vm:~$ _
```

**** For other OS replace putty.exe with "Compatible Terminal".**

Running Mininet

sudo mn

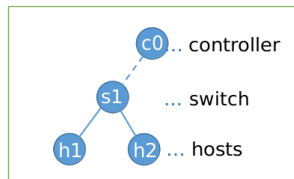
```
mininet@mininet-vm: ~  
Using username "mininet".  
mininet@192.168.56.101's password:  
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)  
  
 * Documentation:  https://help.ubuntu.com/  
Last login: Wed Dec 17 18:39:24 2014 from 192.168.56.1  
mininet@mininet-vm:~$ sudo mn  
*** Creating 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> █
```

Running Mininet

sudo mn runs a network with the default minimal topology:

- 1 Controller
- 1 Switch
- 1 Hosts
- 2 links (h1,s1 & h2,s1)

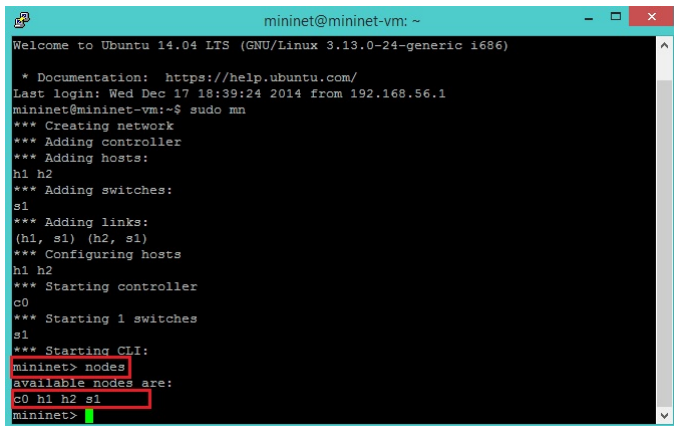
sudo mn



Mininet Command - **nodes**

Lists available **nodes**.

mininet>nodes

A terminal window titled 'mininet@mininet-vm: ~' with standard Ubuntu window controls. The terminal shows the output of the 'sudo mn' command, which sets up a Mininet environment. It lists the steps: creating the network, adding a controller (c0), adding hosts (h1, h2), adding switches (s1), adding links, and configuring the hosts. At the bottom, the 'mininet> nodes' command is entered and executed, resulting in the output 'available nodes are: c0 h1 h2 s1'.

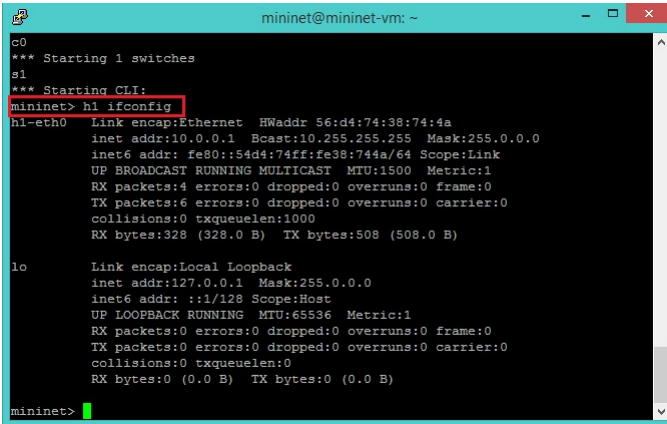
```
mininet@mininet-vm: ~
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Wed Dec 17 18:39:24 2014 from 192.168.56.1
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> nodes
available nodes are:
c0 h1 h2 s1
mininet>
```

Mininet Command - ifconfig

Running a command on a node: prepend the command with the name of the node. E.g. to check the NI config. of a virtual host :

mininet>h1 ifconfig



```
mininet@mininet-vm: ~
c0
*** Starting 1 switches
s1
*** Starting CLI:
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr 56:d4:74:38:74:4a
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::54d4:74ff:fe38:744a/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:328 (328.0 B)  TX bytes:508 (508.0 B)

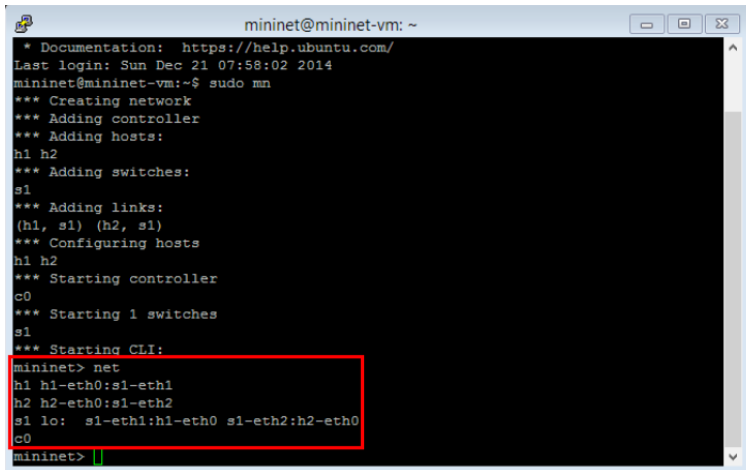
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet>
```

Mininet Command - **net**

Lists **NI-to-NI** connection information.

mininet>net

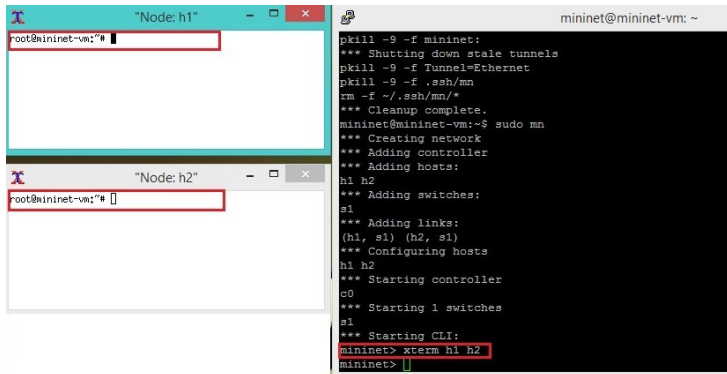


```
mininet@mininet-vm: ~  
* Documentation: https://help.ubuntu.com/  
Last login: Sun Dec 21 07:58:02 2014  
mininet@mininet-vm:~$ sudo mn  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Starting controller  
c0  
*** Starting 1 switches  
s1  
*** Starting CLI:  
mininet> net  
h1 h1-eth0:s1-eth1  
h2 h2-eth0:s1-eth2  
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0  
c0  
mininet> |
```

Mininet Command - xterm

Spawning an **xterm** for one or more virtual hosts.

mininet>xterm h1 h2



The screenshot displays the Mininet environment. On the left, two xterm windows are open, titled "Node: h1" and "Node: h2", both showing a root prompt at mininet-vm. On the right, a terminal window shows the command sequence to start Mininet:

```
mininet@mininet-vm: ~  
kill -9 -f mininet:  
*** Shutting down stale tunnels  
kill -9 -f Tunnel=Ethernet  
kill -9 -f .ssh/mn  
rm -f ~/.ssh/mn/*  
*** Cleanup complete.  
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> xterm h1 h2  
mininet>
```

Mininet Commands - **help**, **exit**, **-c**

Lists available commands

```
mininet>help
```

Exiting Mininet

```
mininet>exit
```

Clearing any residual state or processes

```
$sudo mn -c
```

Mininet Commands - ping & iperf

Ping from one host to another

```
mininet>h1 ping h2
```

Ping reachability from every host to every other host

```
mininet>pingall
```

Testing **bandwidth**

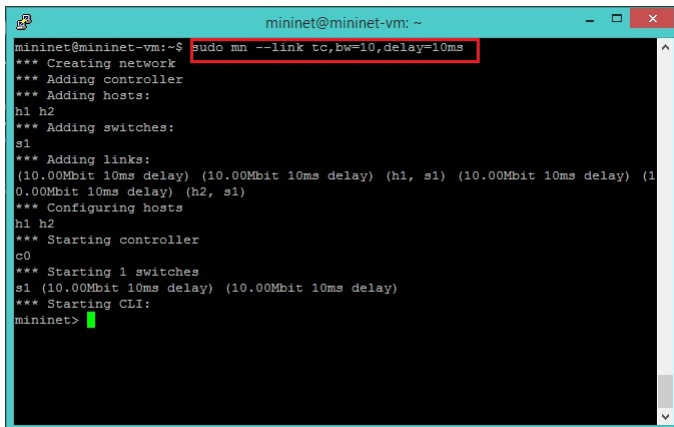
```
TCP : mininet>iperf
```

```
UDP : mininet>iperfudp
```


Mininet Command - [link](#)

Making links with custom Bandwidth & Delay

\$ sudo mn --link=tc,bw=10,delay=10

A terminal window titled 'mininet@mininet-vm: ~' with standard window controls. The command 'sudo mn --link tc,bw=10,delay=10ms' is entered and highlighted with a red box. The output shows the creation of a network with two hosts (h1, h2), one controller (c0), and one switch (s1). Links are added between h1-s1 and h2-s1 with 10.00Mbit bandwidth and 10ms delay. The CLI prompt 'mininet>' is shown at the bottom with a green cursor.

```
mininet@mininet-vm:~$ sudo mn --link tc,bw=10,delay=10ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay) (1
0.00Mbit 10ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 (10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet>
```

To Run Remote Controllers:

- Open separate terminal & navigate to controller:

```
$ cd pox
```

- To run native POX:

```
$ ./pox.py
```

- To run your module of POX (paste in /pox/ext):

```
$ ./pox.py mycontroller
```

- To run POX with DEBUG Level Info:

```
$ ./pox.py mycontroller log.level --DEBUG
```

- To run POX with CRITICAL Level Info:

```
$ ./pox.py mycontroller log.level --CRITICAL
```

Mininet Commands - **topo, mac, switch, controller**

```
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

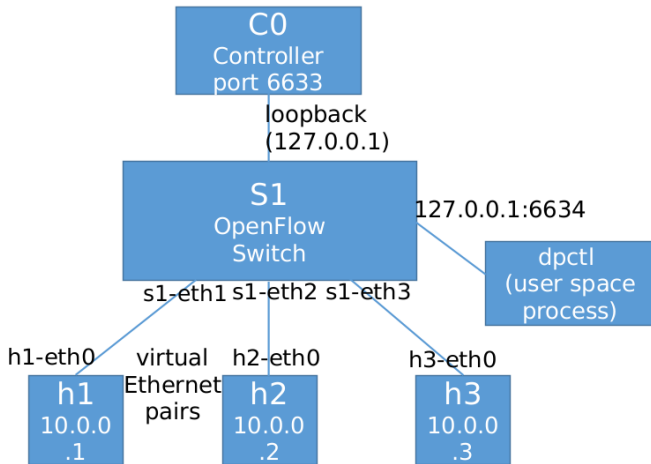
Mininet has

- Created 3 virtual hosts, each with a separate IP address.
- Created a “single” OpenFlow software (openvSwitch-based) switch in kernel with 3 ports.
- Connected each virtual host to the switch with a virtual Ethernet cable.
- Set the MAC address of each host equal to its IP.
- Configure the OpenFlow switch to connect to a remote controller (default is localhost).

Mininet Commands - topo, mac, switch, controller

Creating custom networks

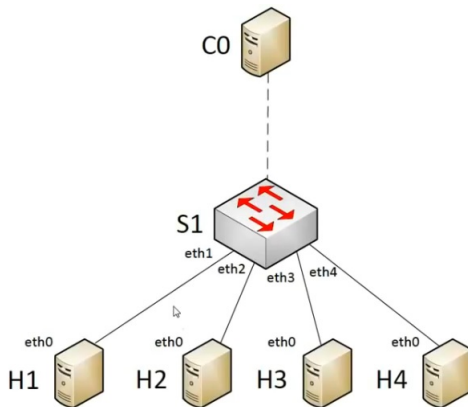
```
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
```



Mininet Command - **topo**

4 Nodes in **Single** Topology

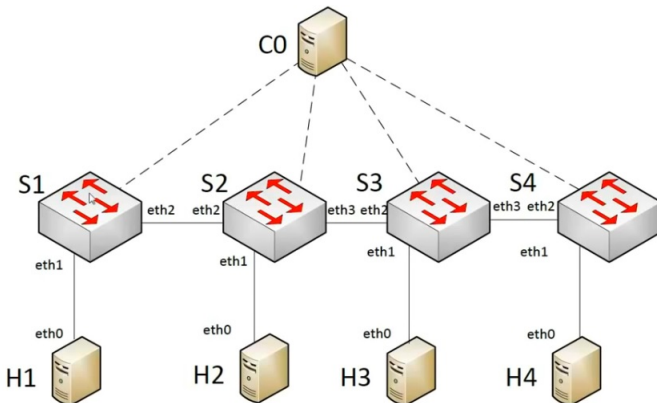
```
$ sudo mn --topo=single,4
```



Mininet Command - **topo**

4 Nodes, 4 Switches in **Linear** Topology

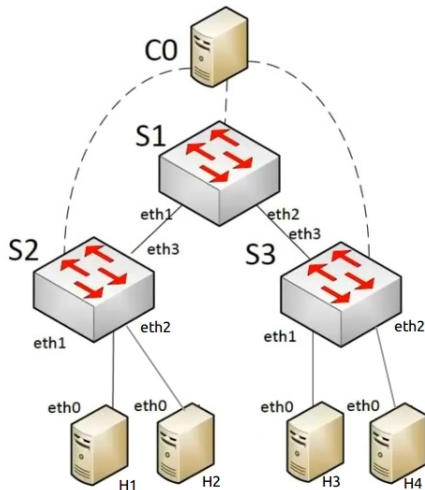
```
$ sudo mn --topo=linear,4
```



Mininet Command - **topo**

Tree Topology: 2 levels deep & fan-out of 2

```
$ sudo mn --topo=tree,2,2
```



Mininet Other Utilities - **dpctl**

- **dpctl** : enables visibility and control over a single switch's flow table. It is useful for debugging.

dpctl show : dumps out switch's port state and capabilities.

```
$ dpctl show tcp:127.0.0.1:6634
```

dpctl dump-flows : shows flows currently installed in SDN switch.

```
$ dpctl dump-flows tcp:127.0.0.1:6634
```

dpctl add-flow : installs the necessary flows. In your SSH terminal:

```
$ dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
```


Wireshark : general (non-OF-specific) graphical utility for viewing packets.

- **\$ sudo wireshark /dev/null**
- Click on Capture -> Interfaces.
- Check 'lo', the loopback interface.
- Click on the Start button.

References



<http://mininet.org/>



B. Lantz, B. Heller, and N. McKeown. **A Network in a Laptop: Rapid Prototyping for Software-Defined Networks**, Proc. of ACM Hot-nets'10, 2010.

Thank You!!!
Check Warm-Up Exercise.

Exercise 1

Creating a network manually & running a simple performance test.

```
def perfTest():  
    "Create network and run simple performance test"  
    topo = SingleSwitchTopo(n=4)  
    net = Mininet(topo=topo,host=CPULimitedHost, link=TCLink)  
    net.start()  
    print "Dumping host connections"  
    # To Do 1: Dump hosts connections here  
    print "Testing network connectivity"  
    # To Do 2: Ping all hosts via single command  
    print "Testing bandwidth between h1 and h3"  
    # To Do 3: Test bandwidth between host 1 & host 3  
    # HINT: First get the names of hosts from network then iperf them.  
    net.stop()
```

ex1.py

Commands to use:

- 1 dumpNodeConnections(net.hosts)
- 2 net.pingAll()
- 3 host_1, host_2 = net.get('host_1', 'host_2')
- 4 net.iperf((host_1, host_2))

Exercise 2

Installing rules manually using *ovs-ofctl*.

```
for intf in switch1.intfs.values():
    print intf
    print switch1.cmd( 'ovs-vsctl add-port dp1 %s' % intf )

# To Do 1:
# add-flow for switch 1 & 2 for both to & fro
# Use Syntax :
# switch_name.cmd('ovs-ofctl add-flow datapath_id \"in_port=port_X actions=output:port_Y\"' )
```

ex2.py

Info:

DatapathId(dpid) = Unique identifier for switches.

Exercise 3

Simple controller.

```
print "s0_dpid=", s0_dpid
#To Do 1: Do it for the rest of the switches as well!!! SEE TOPOLOGY FILE FOR DETAILS OF THE SWITCHES.

def _handle_PacketIn (event):
    global s0_dpid, s1_dpid, s2_dpid, s3_dpid
    print "PacketIn: ", dpidToStr(event.connection.dpid)

    if event.connection.dpid==s0_dpid:
        msg= of.ofp_flow_mod()
        msg.idle_timeout = 0
        msg.hard_timeout = 0
        # To Do 2: FLOOD THE PACKETS. USE SYNTAX:|
        # msg.actions.append(of.ofp_action_output(port = FLOOD_TO_ALL_CMD))
        event.connection.send(msg)

#To Do 3: Repeat for the rest of the switches as well!!!
```

ex3_ctrl.py

How to:

- 1 PASTE "ex3_ctrl.py" UNDER /pox/ext/
- 2 Use Command: *of.OFPP_ALL* & run as remote controller.
- 3 Open separate terminal & run "ex3_tp.py".
- 4 If everything is OK; hosts should ping each other.