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.

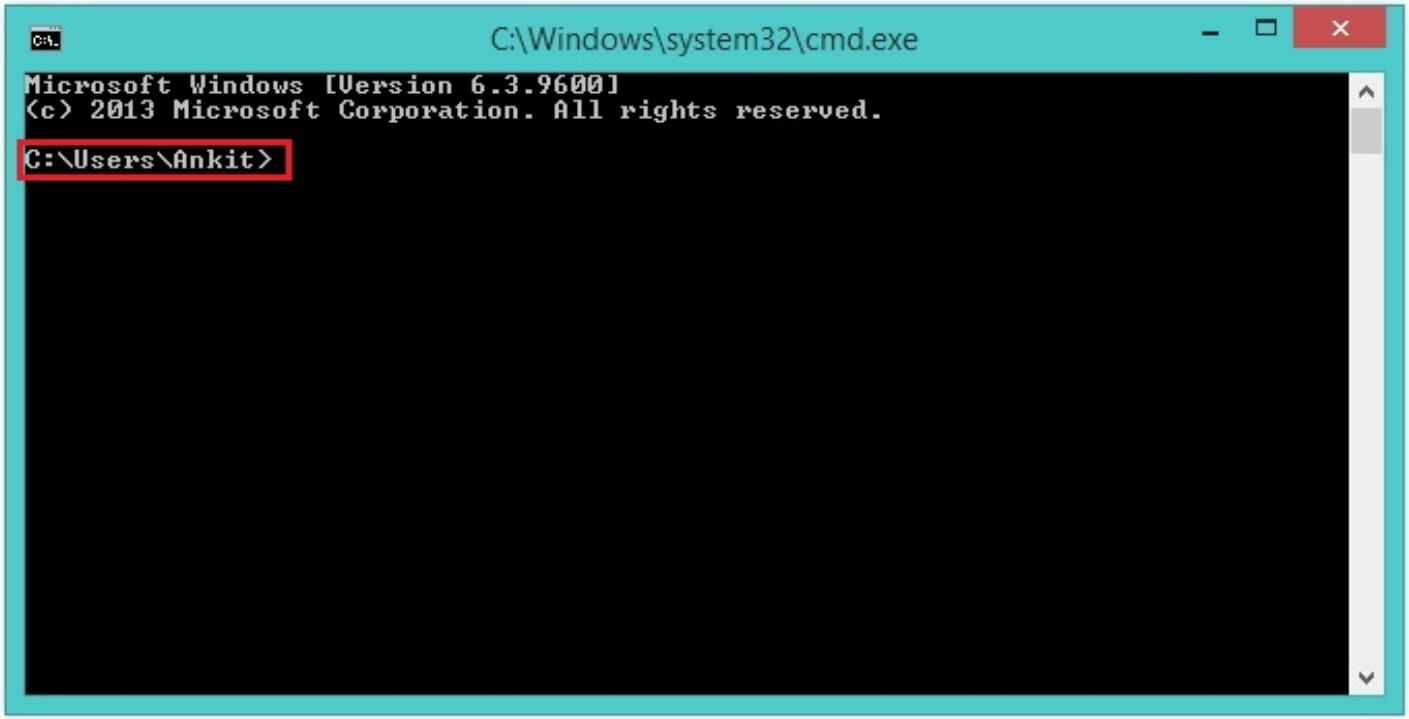
Option 1: Mininet VM Installation

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 Leapord	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

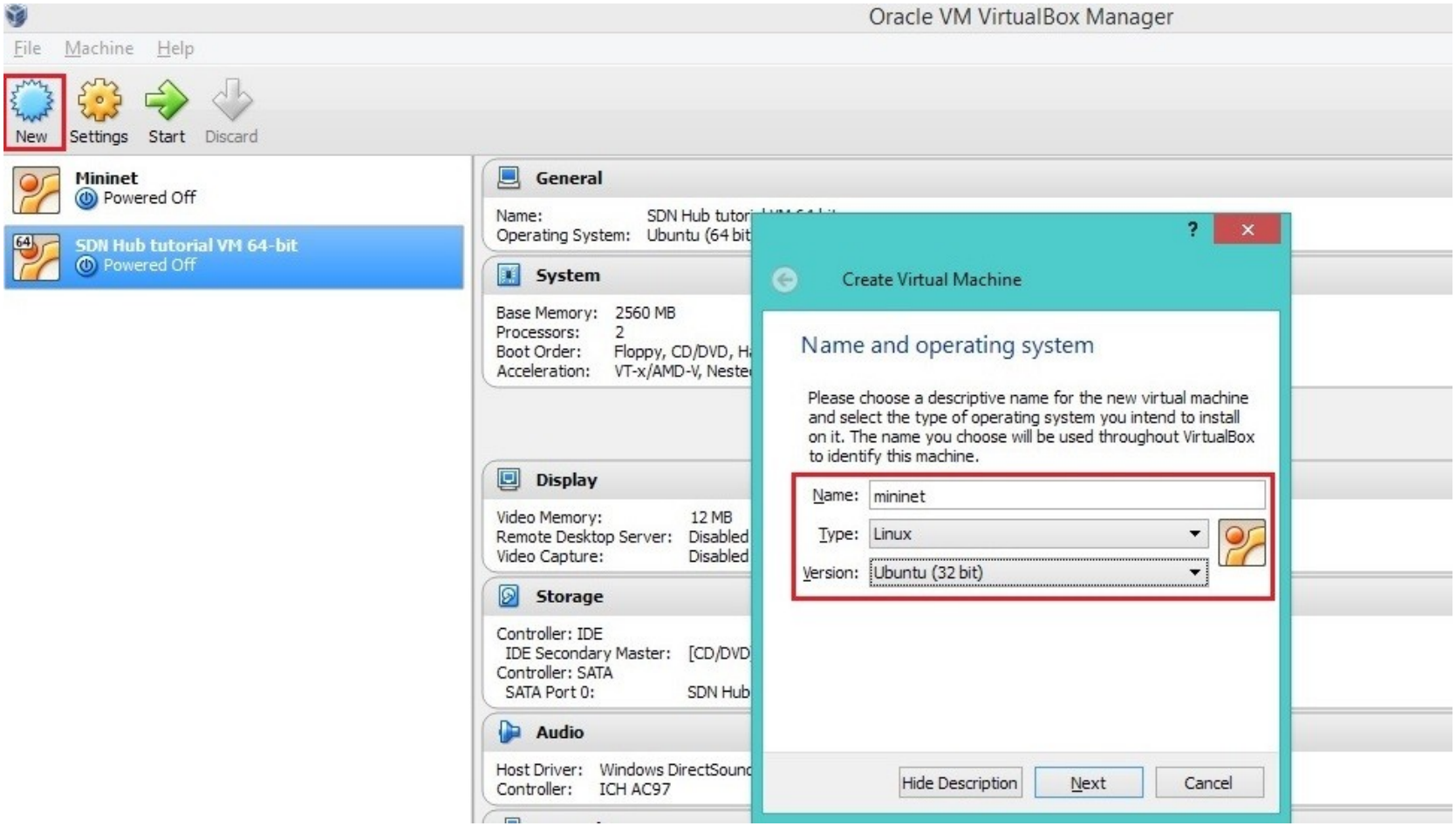
Option 1: Mininet VM Installation

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".



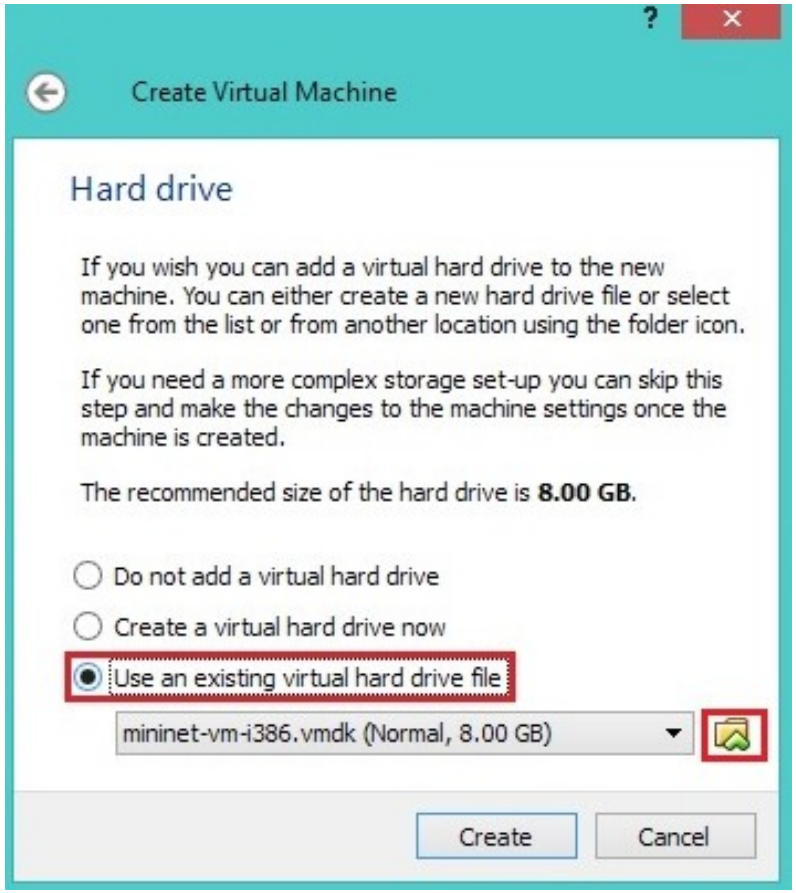
Option 1: Mininet VM Installation

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



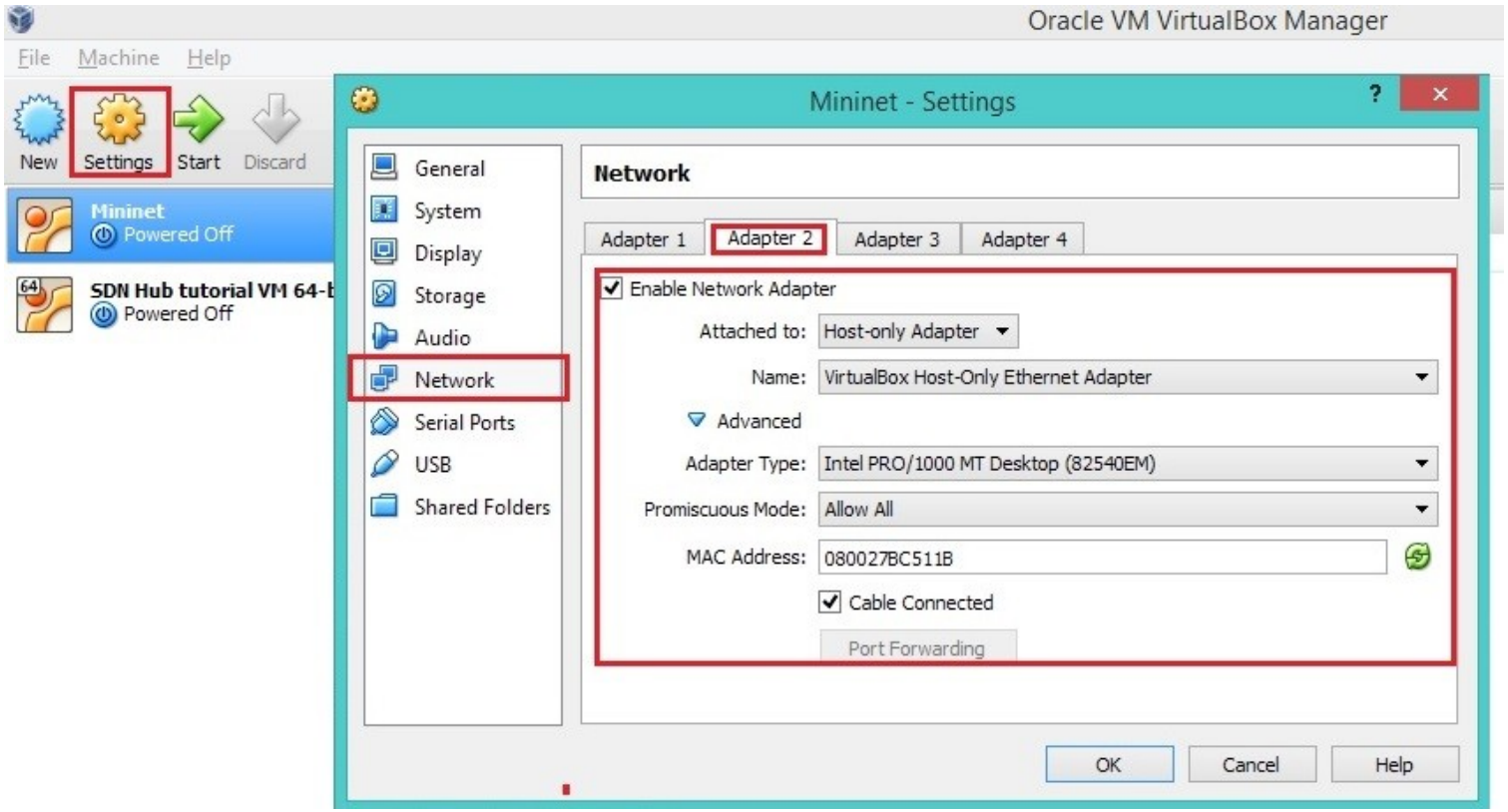
Option 1: Mininet VM Installation

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



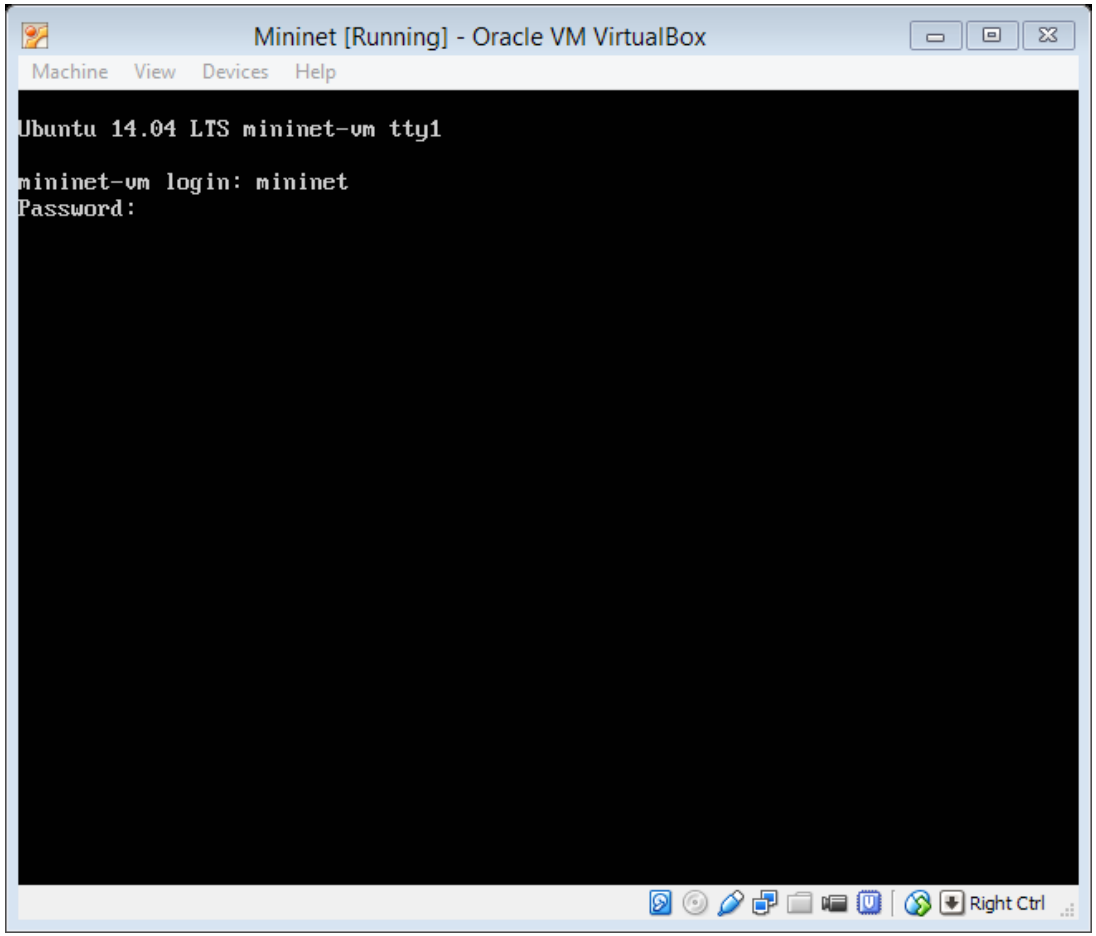
Option 1: Mininet VM Installation

10. Click on Settings ->Network ->Adapter 2 ->check "Enable Network Adapter" ->Attached to "Host-only Adapter" ->Advanced ->Promiscuou 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

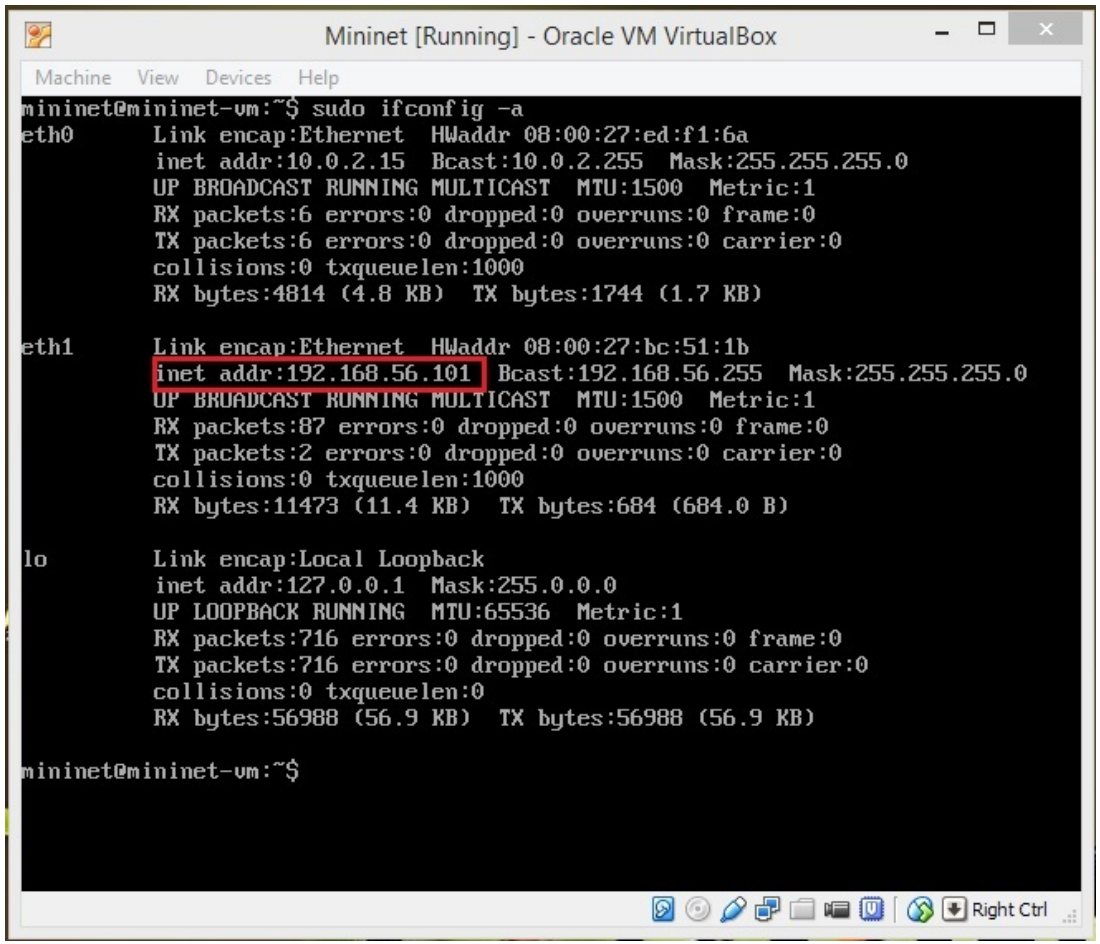
Preparing Mininet

- **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

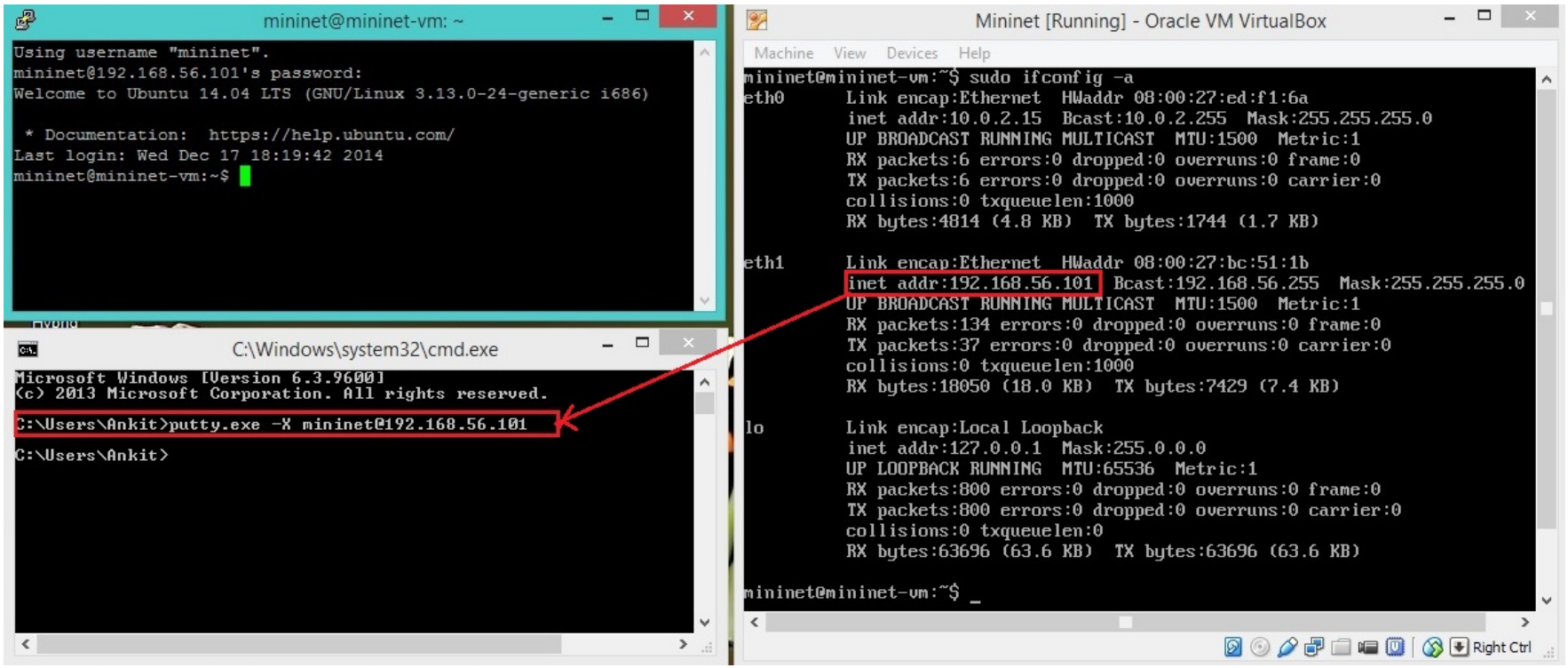
3. Repeat step 1 & note "inet addr" for eth1.



Preparing Mininet

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

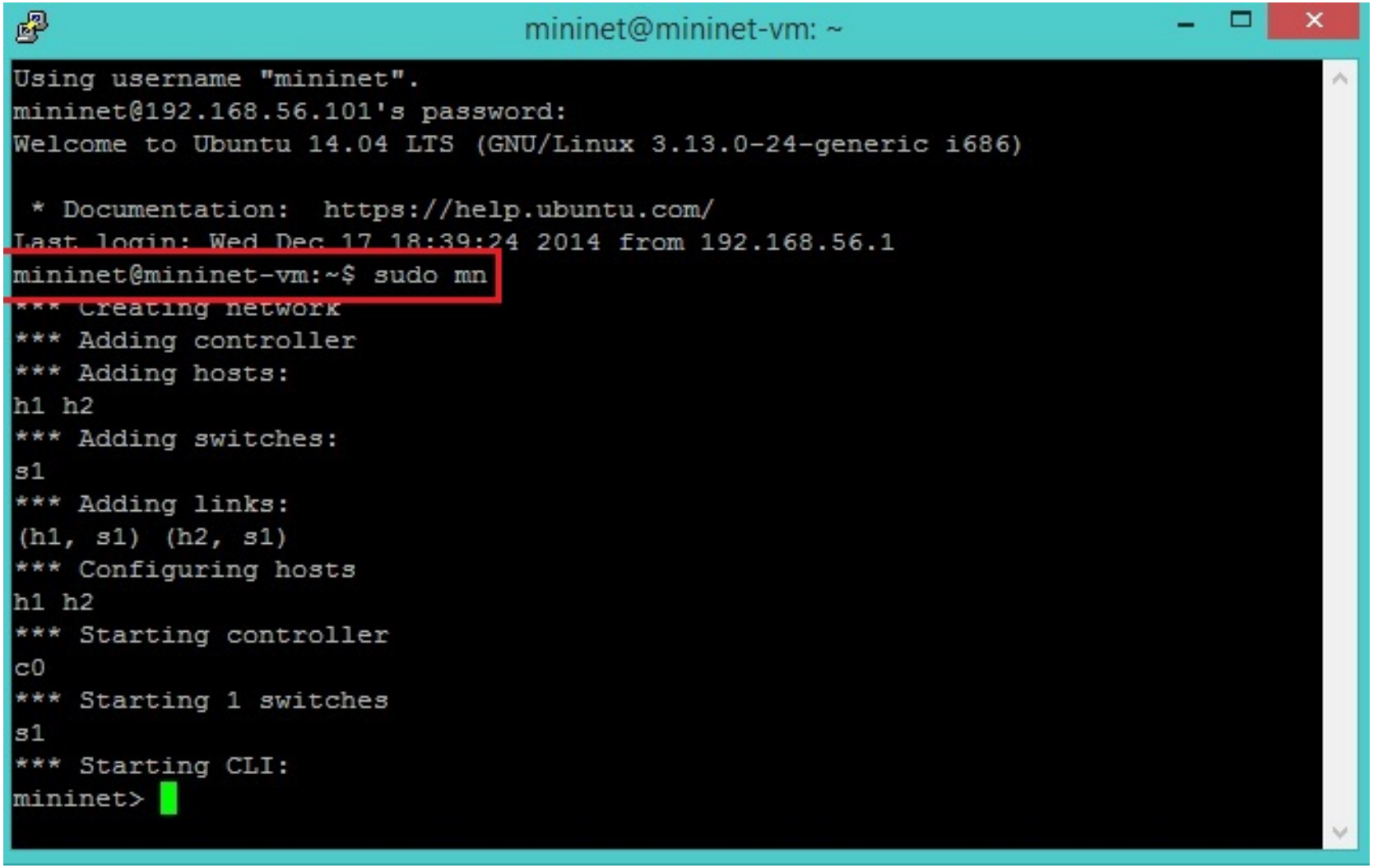
putty.exe -X mininet@192.168.56.101



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

Running Mininet

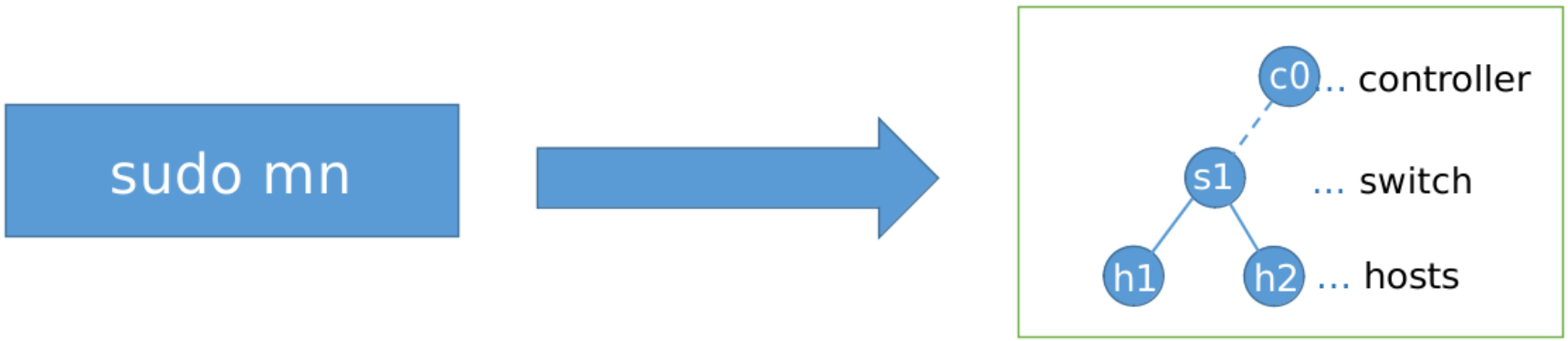
sudo mn



Running Mininet

sudo mn runs a network with the default minimal topology:

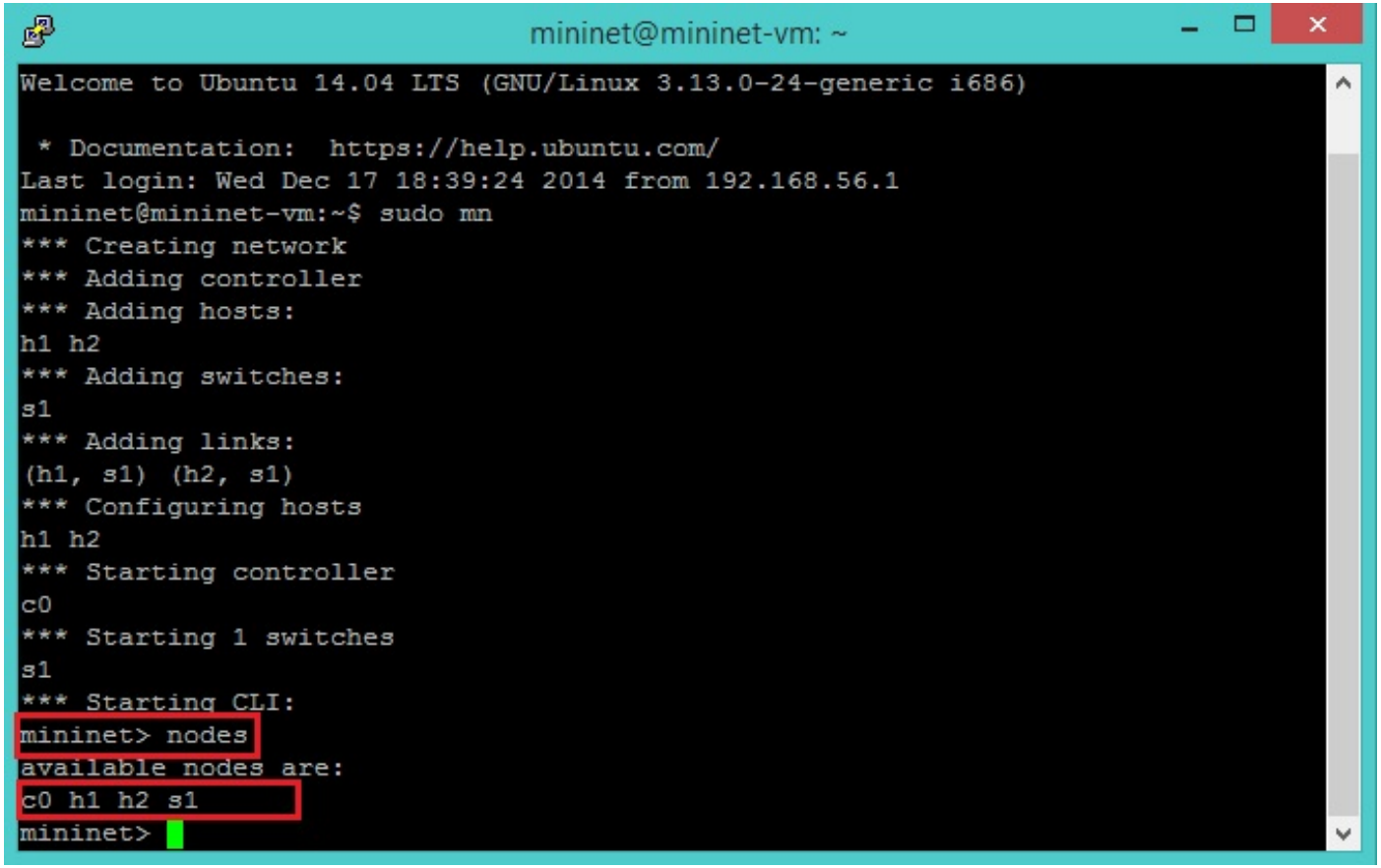
- 1 Controller
- 1 Switch
- 2 Hosts
- 2 links (h1,s1 & h2,s1)



Mininet Command - **nodes**

Lists available **nodes**.

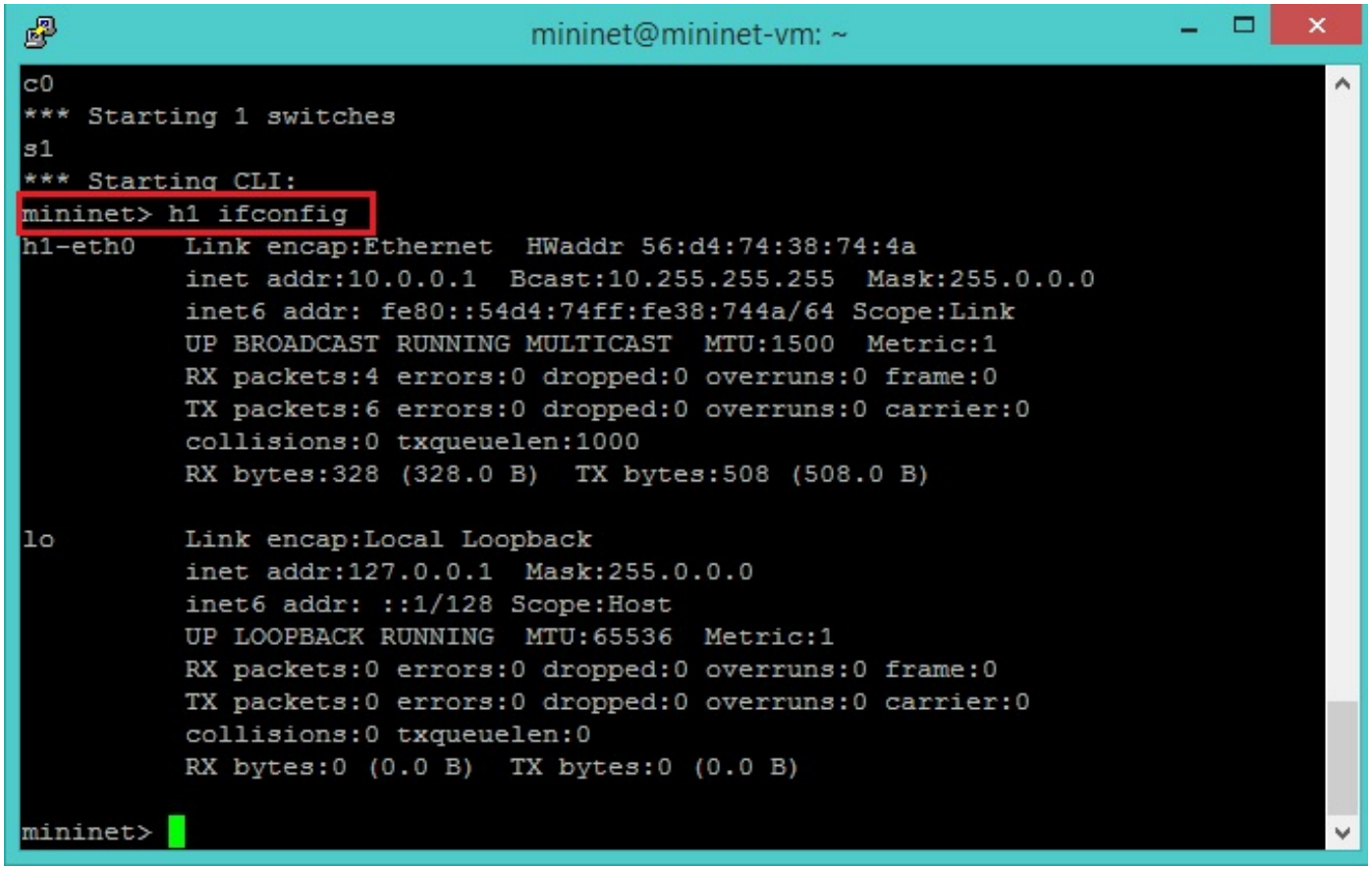
mininet>nodes



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 :

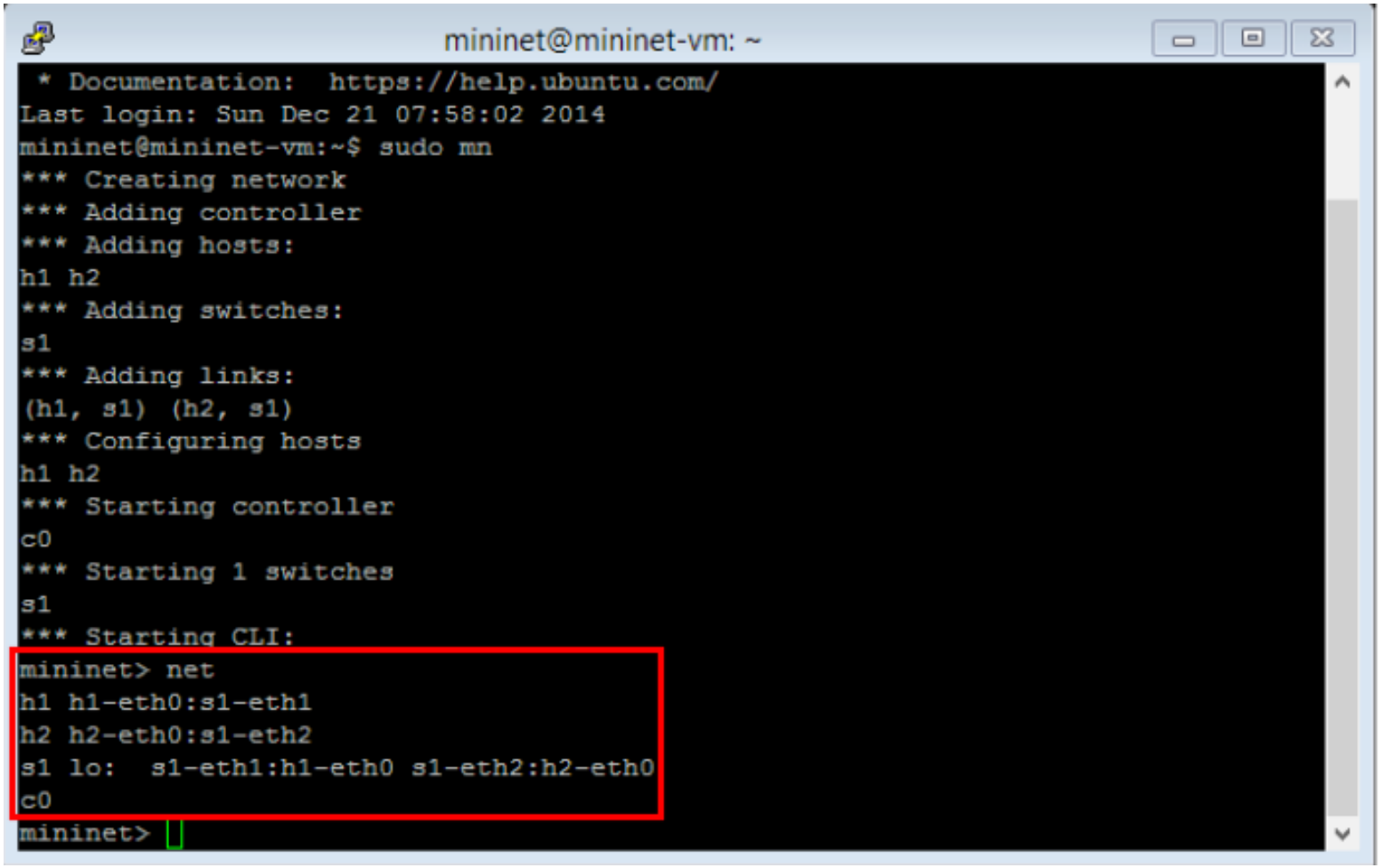
mininet>h1 ifconfig



Mininet Command - **net**

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

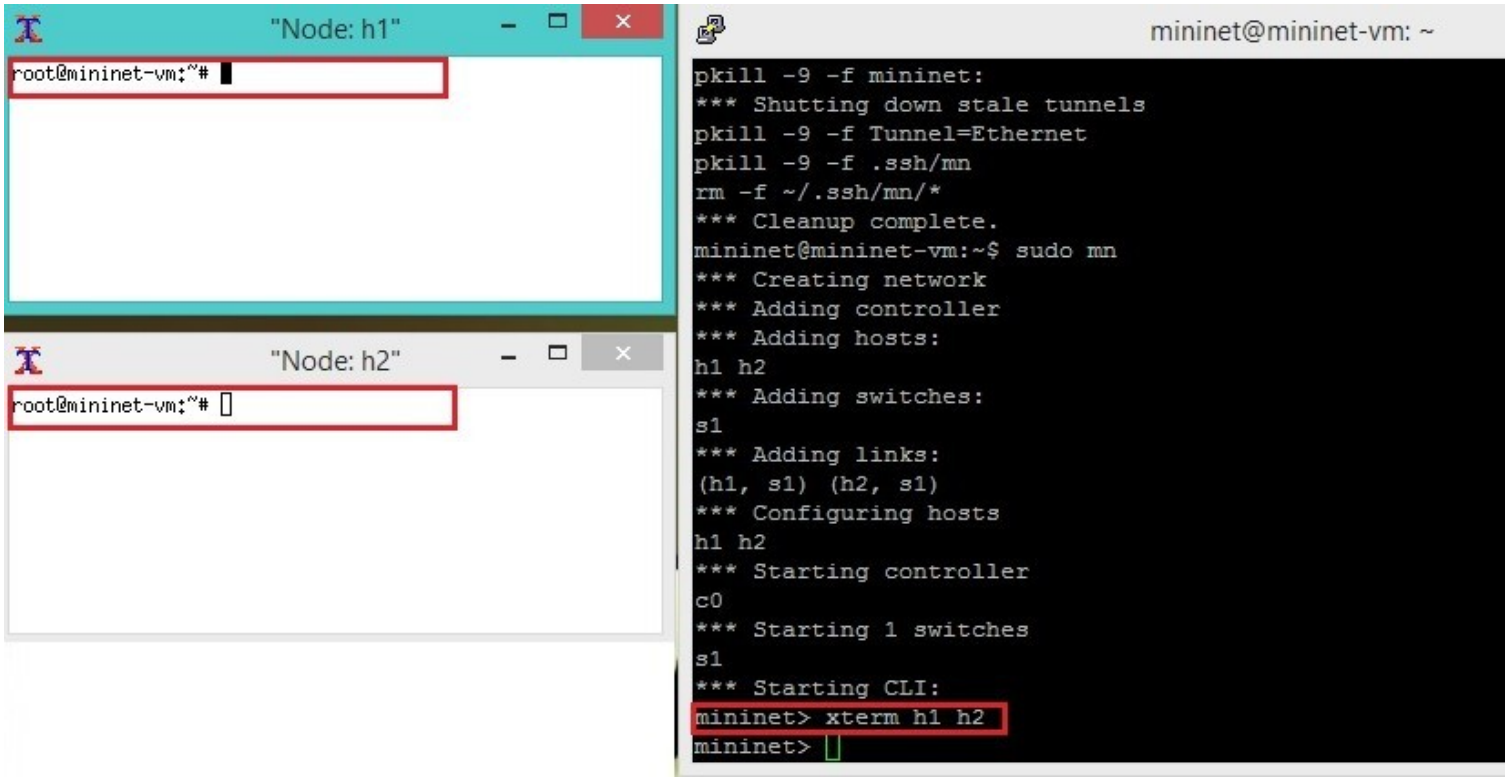
mininet>net



Mininet Command - **xterm**

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

mininet>xterm h1 h2



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**

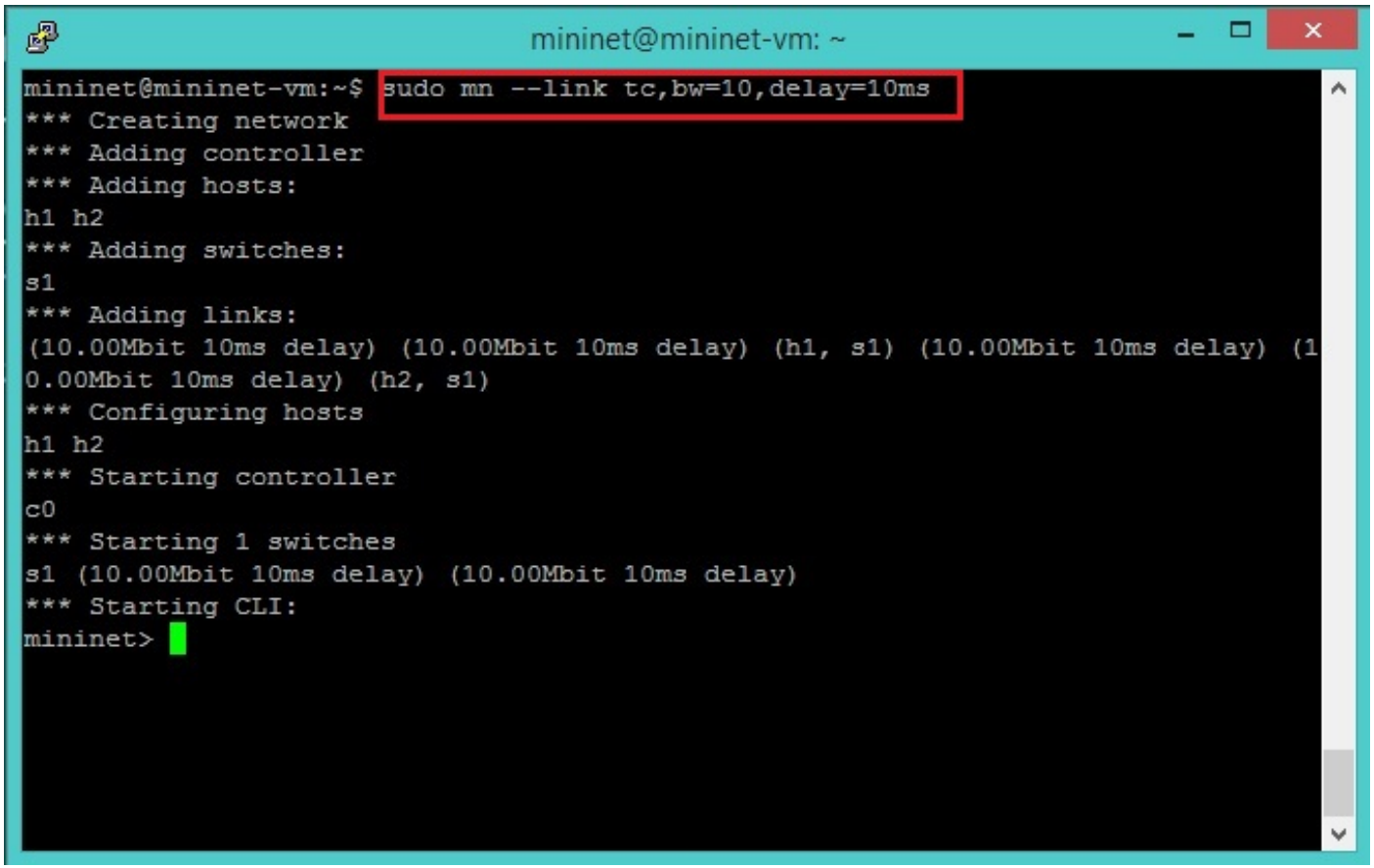
TCP : **mininet>iperf**

UDP : **mininet>iperfudp**

Mininet Command - **link**

Making links with custom **Bandwidth** & **Delay**

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



Remote Controllers

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

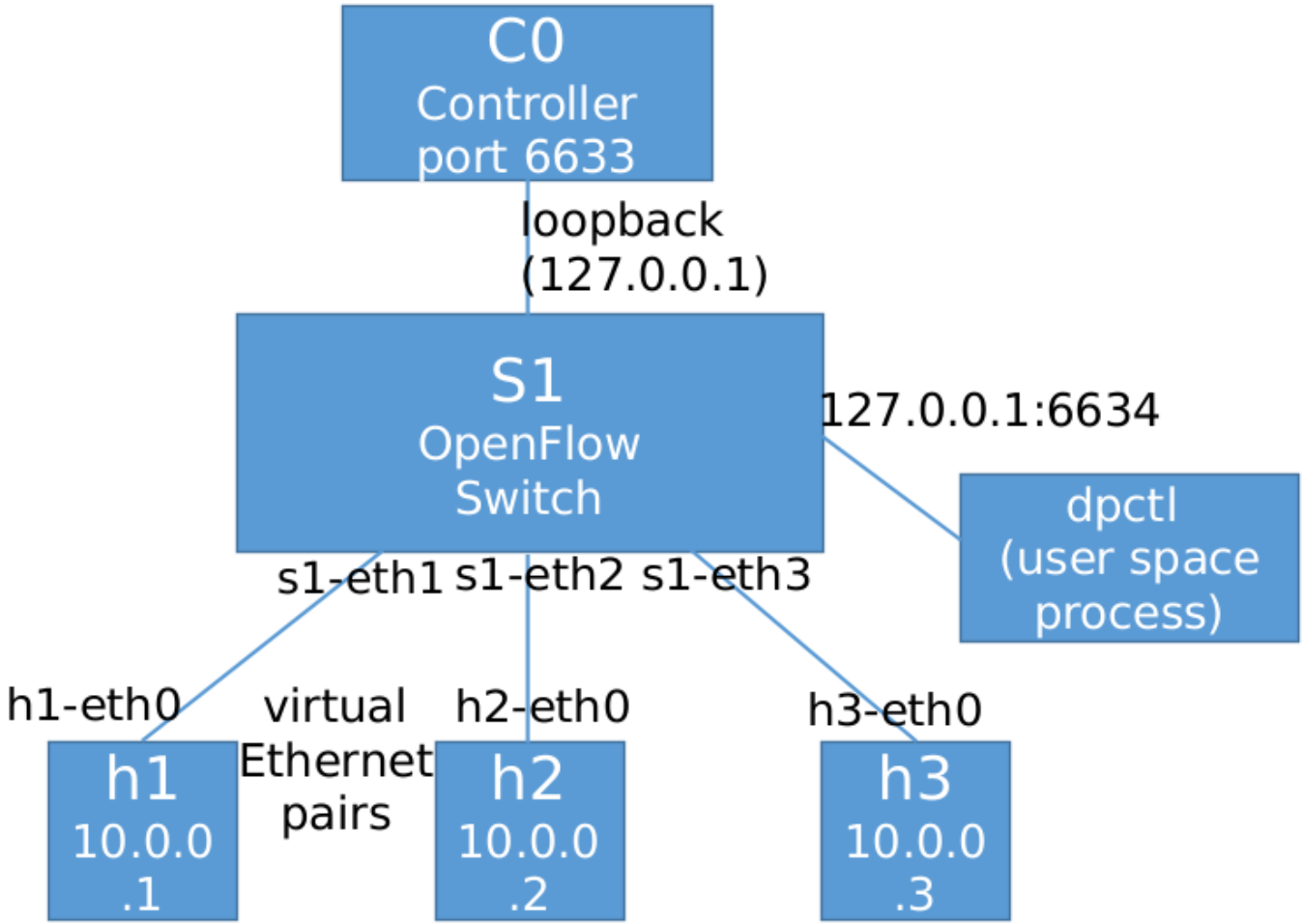
Mininet has

- 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

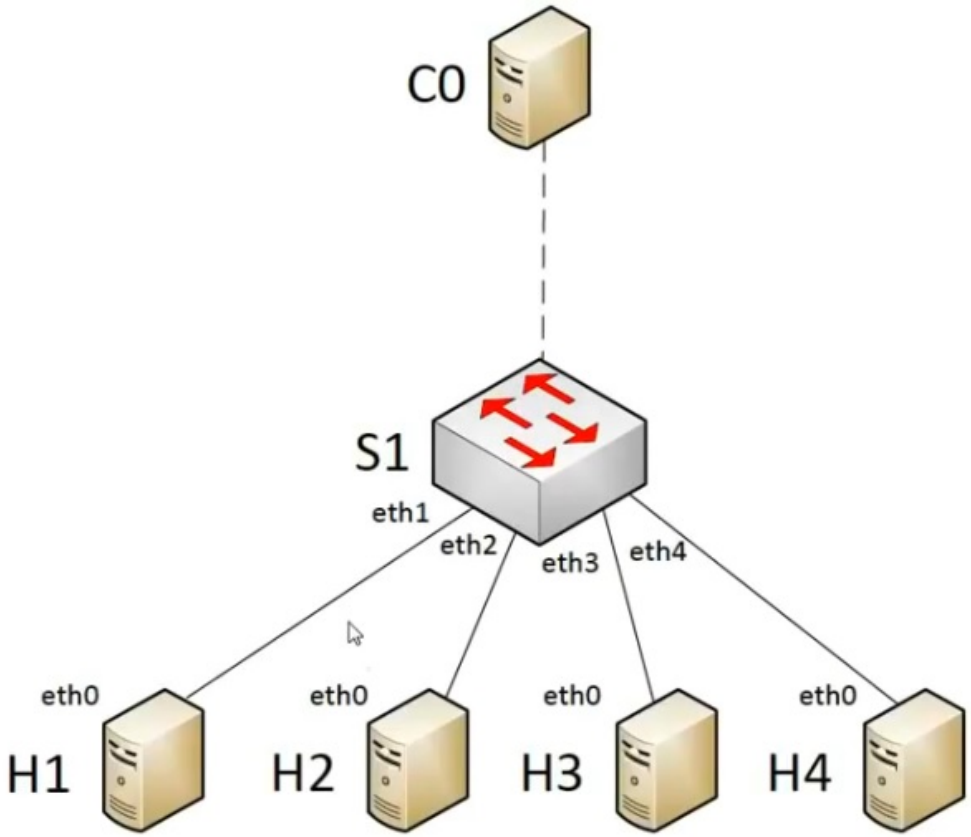
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote



Mininet Command - **topo**

4 Nodes in **Single** Topology

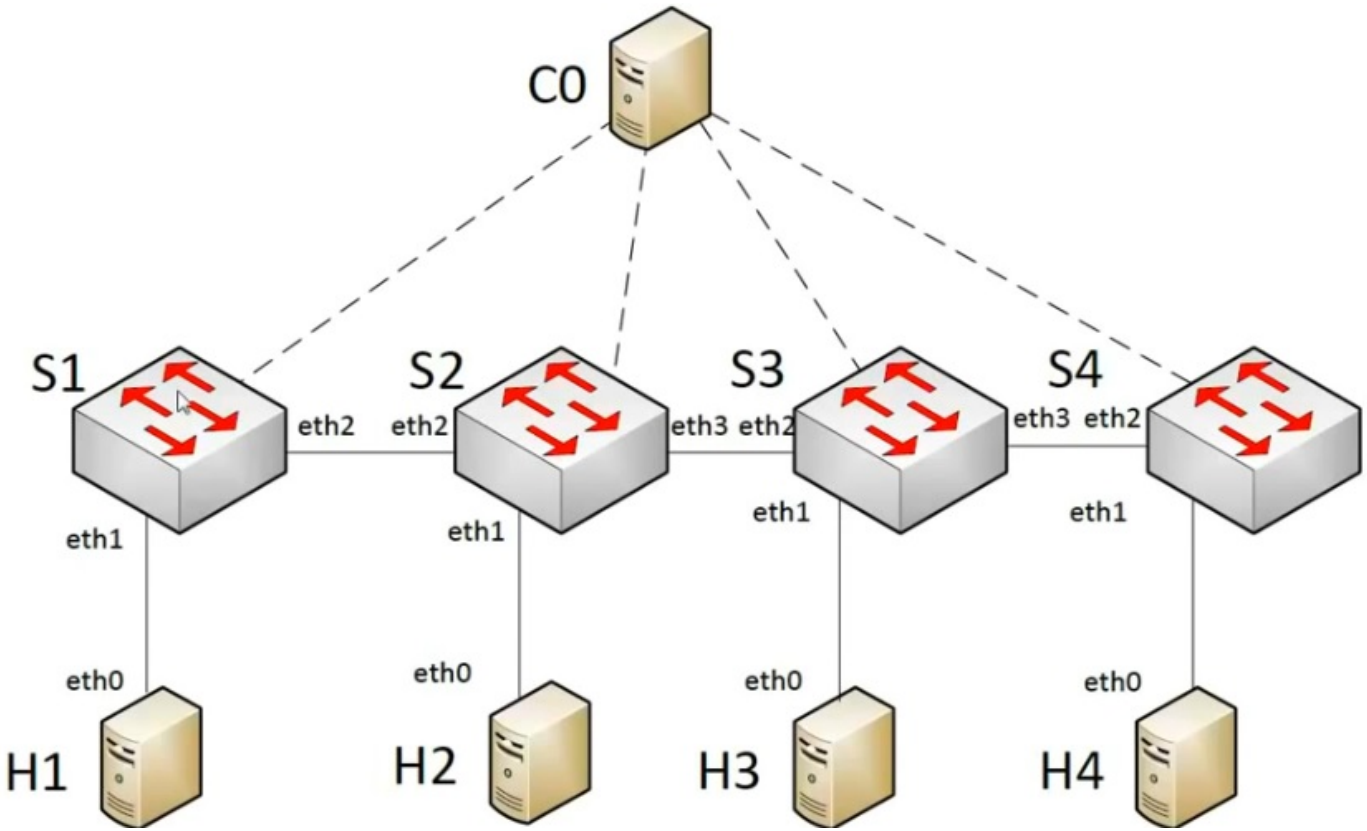
$ sudo mn --topo=single,4



Mininet Command - **topo**

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

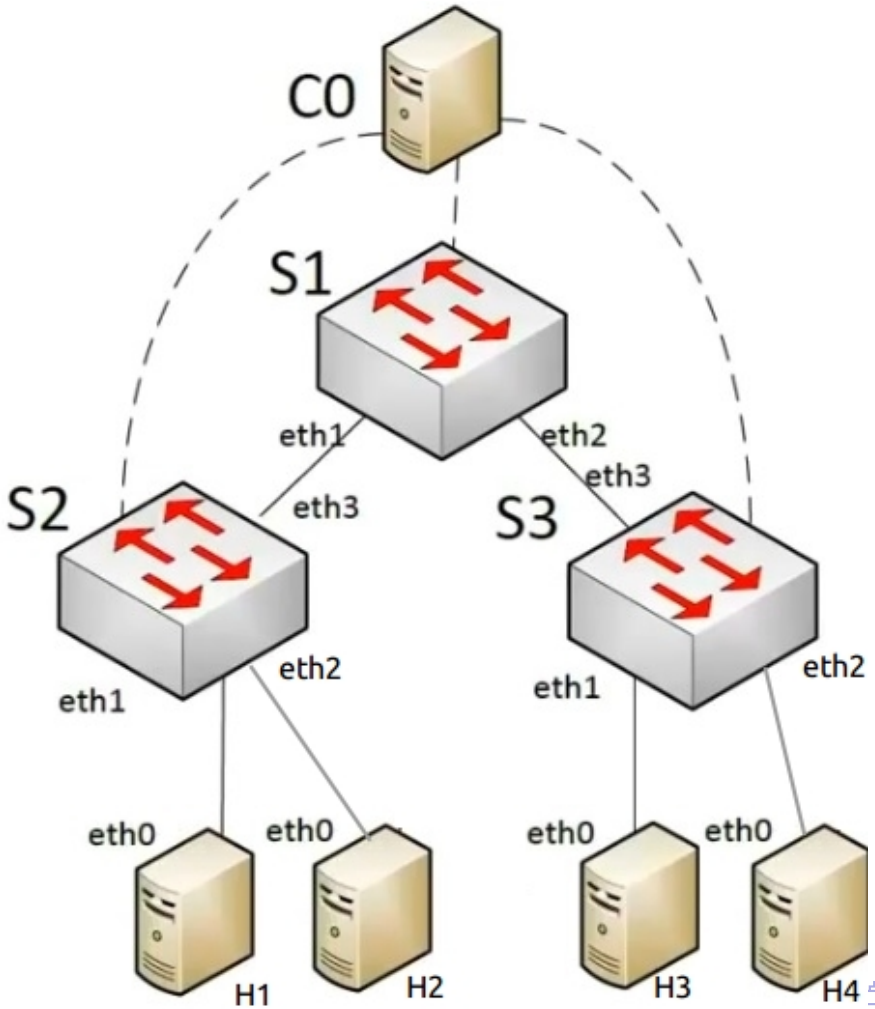
$ sudo mn --topo=linear,4



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
```

Mininet Other Utilities - **Wireshark**

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 Hotnets'10, 2010.

Thank You!!!

Check Warm-Up Exercise.

Exercise 1

Creating a network manually & running a simple performance test.

ex1.py

```
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()
```

Commands to use:

1. dumpNodeConnections(net.hosts)
2. net.pingAll()
3. host_1, host_3 = net.get('host_1', 'host_3')
4. net.iperf((host_1, host_3))

Exercise 2

Installing rules manually using *ovs-ofctl*.

ex2.py

```
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\"' )
```

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.