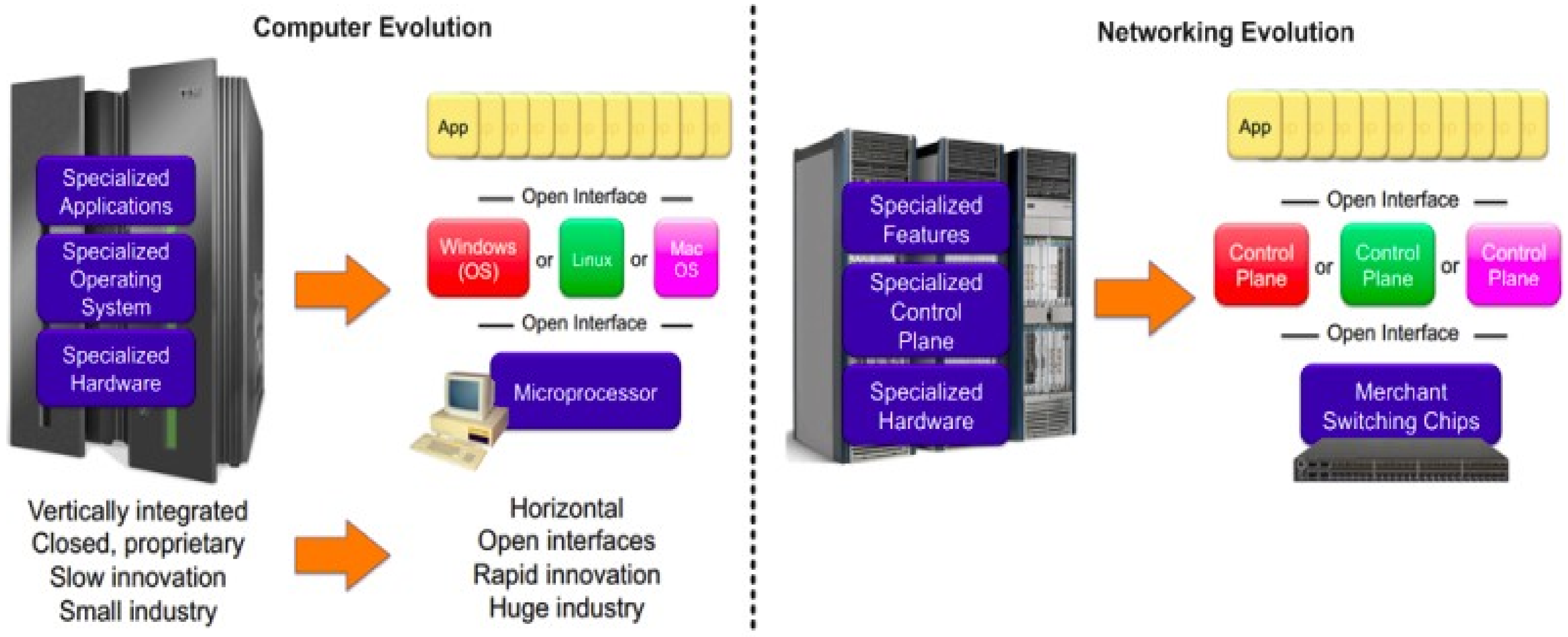
CS 154: Introduction to Mininet

(Last Revised August 2014)

Department of Information Systems and Computer Science
Ateneo de Manila University

Parallels of Evolution

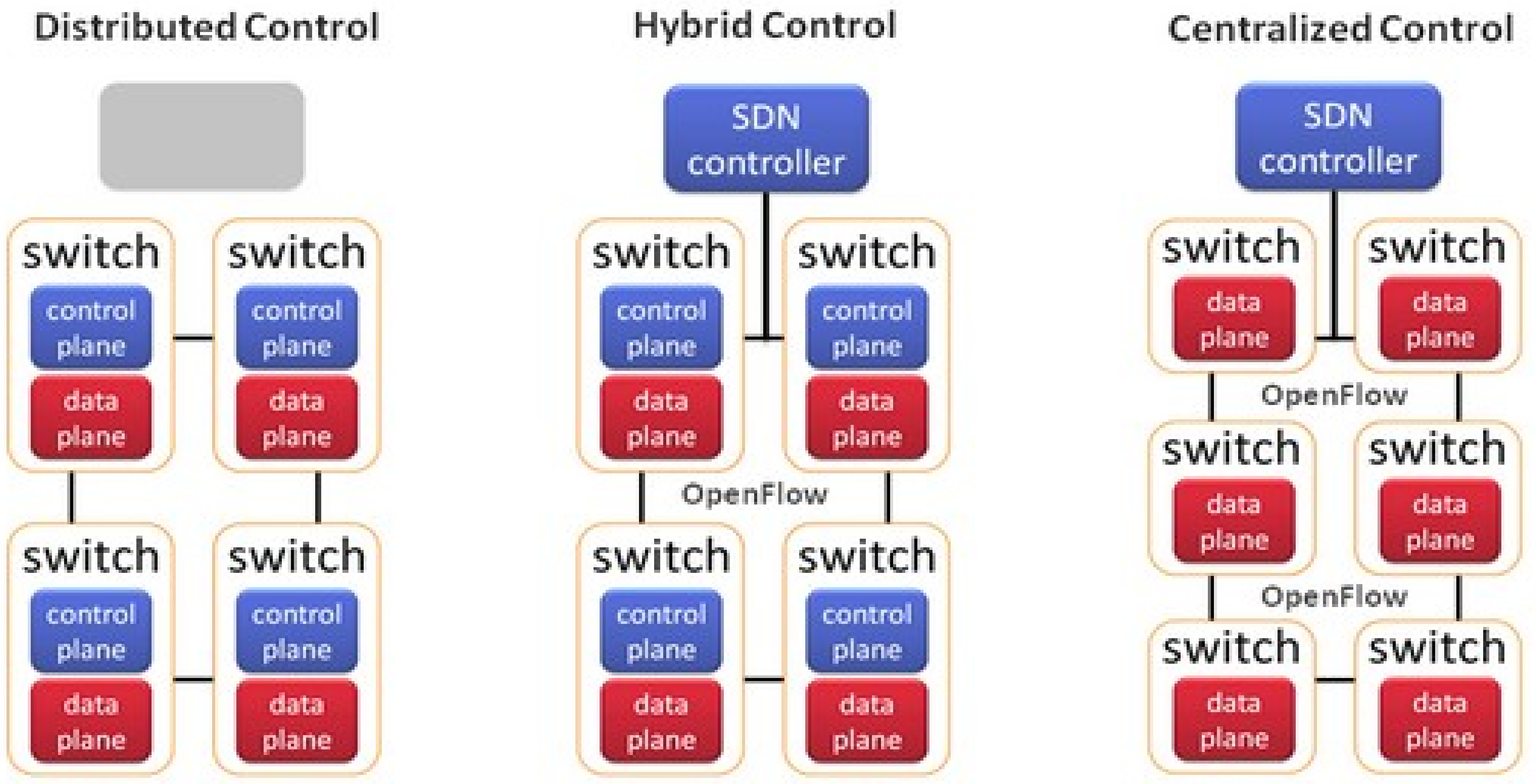


[1] Taken from Nick McKeown's presentation "How SDN will shape networking" - See more at: http://www.costiser.ro/2014/08/07/sdn-lesson-1-introduction-to-mininet/

Software Defined Networks

- Acknowledges the opportunities presented by the **emergence of SDNs**.
  - Ability to quickly create network test beds and try out new protocols and topologies.
  - Remove the restriction to doing networking research with micro-modeling or with real equipment.
  - Potentially create a new class of network applications.
- There is still much work to be done.

Software Defined Networks



What is POX?

- **POX** is a platform for the rapid development and prototyping of network control software using Python.
- Features:
  - “Python” OpenFlow interface.
  - Provides reusable sample components for path selection, topology, discovery, etc.
  - “Runs anywhere” – Can bundle with install-free PyPy runtime for easy deployment.
  - Specifically targets Linux, Mac OS, and Windows,
  - Supports the same GUI and visualization tools as NOX.
  - Performs well compared to NOX applications written in Python.

This is our **OpenFlow/SDN Controller** for this class.

What is Mininet?

- A tool for **rapid prototyping** of software-defined networks (SDN).
- Create a **realistic virtual network** with real working components but running on a single machine for ease of testing.
- Provide the ability to create hosts, switches and controllers via:
  - Command Line
  - Interactive User Interface
  - Python application
- More information: http://mininet.github.com/

Getting Started

- Download Mininet Virtual Machine Image
  - http://mininet.org/download/ (the latest one)
  - Username: mininet (for Mininet 2.1 VM)
  - Password: mininet
  - http://mininet.org/vm-setup-notes/ (more details)
- Ensure your workstation has the appropriate virtual machine software (ie. VirtualBox, KVM).
  - http://www.virtualbox.org/wiki/Downloads

Is it working?

- To do a quick check if it is working, type the following command:
  - sudo mn --test pingall --topo single,3
- This command will:
  - Create a single network with 3 hosts connected to a single switch in non-interactive mode.
  - Perform a ping from all hosts to all other hosts.
  - This command uses a default switch controller and should work.

Is it working?

```
openflow@mininet-vm:~$ sudo mn --test pingall --topo single,3
*** Adding controller
*** Creating network
*** Adding hosts:
h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(s1, h2) (s1, h3) (s1, h4)
*** Configuring hosts
h2 h3 h4
*** Starting controller
*** Starting 1 switches
s1
*** Ping: testing ping reachability
h2 -> h3 h4
h3 -> h2 h4
h4 -> h2 h3
*** Results: 0% dropped (0/6 lost)
*** Stopping 3 hosts
h2 h3 h4
*** Stopping 1 switches
s1...
*** Stopping 1 controllers
*** Done
completed in 2.695 seconds
```

Creating the three (3) hosts and one switch.

Defining links between nodes.

Starting default controller.

Doing a ping from all nodes to all nodes.

Basic Interactive Commands

- **help** – display a list of possible mininet commands.
- **nodes** – displays a list of nodes.
- **net** – displays network topology (in ASCII art).
- **dump** – displays interface setup per node and the PID of the process representing each node.
- **<node id> ifconfig** – similar to the Linux command for defining a network interface.
- **<node id> route** – similar to the Linux command for defining entries in the nodes' routing table.
- **iperf <node id> <node id>** - network performance test between two (2) nodes.
- **<node id> ping <node id>** - perform a network ping between two (2) nodes.
- **pingall** – perform a ping test between all hosts in the network.
- **xterm <node id>** - launch a remote xterm logged into that specified node.

Basic Command Line

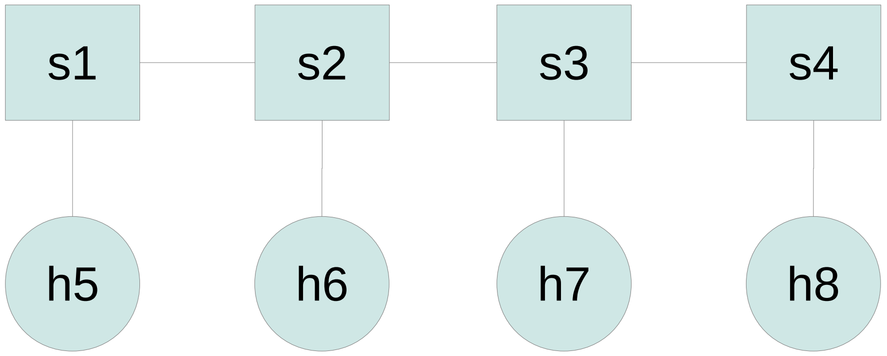
- **--topo** – defines a topology via command line upon mininet start-up.
- **--switch** – defines the switch to be used. By default the OVSK software switch is used.
- **--controller** – defines the controller to be used. If unspecified default controller is used with a default hub behavior.

Some Topologies to Try Out

- Minimal network with two (2) hosts connected to one (1) switch.
  - sudo mn –topo minimal
- Each host is connected to one switch and all switches are connected to each other.
  - In this example, there are 4 hosts and 4 switches.
  - sudo mn --topo linear,4
- Each host is connected to a single switch.
  - In this example, there are 3 hosts and 1 switch.
  - sudo mn --topo single,3
- Tree based topology with defined depth and fan-out.
  - sudo mn --topo tree,depth=2,fanout=2

--topo linear,4

(s1, s2) (s1, h5) (s2, s3) (s2, h6) (s3, s4) (s3, h7) (s4, h8)

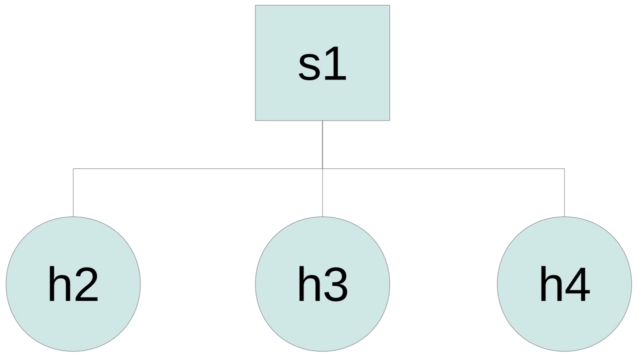


To run script: **sudo python <script name>**

```
from mininet.net
import Mininet

from mininet.topo
import LinearTopo

Linear4 =
LinearTopo(k=4)

net =
Mininet(topo=Linear4)

net.start()

net.pingAll()

net.stop()
```

--topo single,3

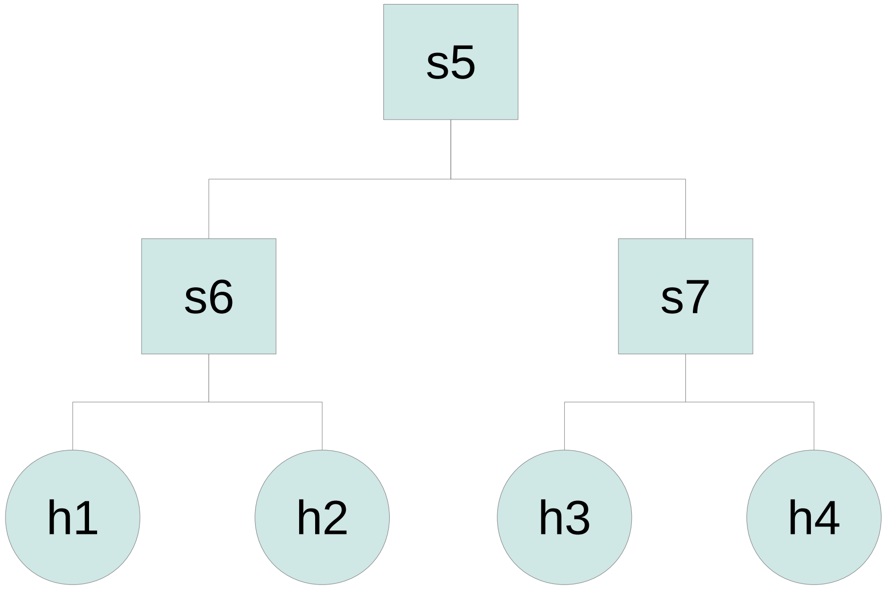
(s1, h2) (s1, h3) (s1, h4)



```
from mininet.net import
Mininet

from mininet.topo import
SingleSwitchTopo

Single3 =
SingleSwitchTopo(k=3)

net = Mininet(topo=Single3)

net.start()

net.pingAll()

net.stop()
```

--topo tree,depth=2,fanout=2

(h1, s6) (h2, s6) (h3, s7) (h4, s7) (s5, s6) (s5, s7)



```
from mininet.net import
Mininet

from mininet.topolib import
TreeTopo

Tree22 = TreeTopo(depth=2,
fanout=2)

net = Mininet(topo=Tree22)

net.start()

net.pingAll()

net.stop()
```

Unknown Topology

```
from mininet.net import Mininet
net = Mininet()

# Creating nodes in the network.
c0 = net.addController()
h0 = net.addHost('h0')
s0 = net.addSwitch('s0')
h1 = net.addHost('h1')

# Creating links between nodes in network
net.addLink(h0, s0)
net.addLink(h1, s0)

# Configuration of IP addresses in interfaces
h0.setIP('192.168.1.1', 24)
h1.setIP('192.168.1.2', 24)

net.start()
net.pingAll()
net.stop()
```

- Ability to create any topology desired.
- Please note the two (2) ways to add a link.
- By default, script drops the network once completed. You can put mininet.cli.CLI(net) before net.stop() to drop to interactive mode.
- What does this network look like?

Setting Performance Limits

```
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink

net = Mininet(host=CPULimitedHost, link=TCLink)

c0 = net.addController()
s0 = net.addSwitch('s0')
h0 = net.addHost('h0')
h1 = net.addHost('h1', cpu=0.5)
h2 = net.addHost('h2', cpu=0.5)

net.addLink(s0, h0, bw=10, delay='5ms',
max_queue_size=1000, loss=10, use_htb=True)
net.addLink(s0, h1)
net.addLink(s0, h2)

net.start()
net.pingAll()
net.stop()
```

- Notice the addHost() syntax allows you to specify:
  - CPU (**cpu**) in percentage allocation per virtual host
- Notice the addLink() syntax allows you to specify:
  - Bandwidth (**bw**) in Mbps
  - Delay (**delay**)
  - Maximum Queue Size (**max_queue_size**)
  - Loss (**loss**) in percentage

Testing Your Networks

```
# display network information

mininet.util.dumpNodeConnections(n
et.hosts)


# testing ping from all hosts to
all hosts

net.pingall()


# test ping from one host to
another host

net.ping(h0, h1)


# test packet performance from one
host to another host

net.iperf((h0,h1))
```

- Mechanisms to programmatically obtain information and test a Mininet network
  - **mininet.util.dumpNodeConnections()** is used to return all network information in a printable manner.
  - **mininet.net.Mininet.pingAll()** is a ping test from all nodes to all nodes while **mininet.net.Mininet.ping()** is from one host to anther.
  - **mininet.net.Mininet.iperf()** is a basic packet test.

Running Commands in CLI

```
# to run a command in CLI
CLI>py net.pingAll()

# to run imports in CLI then run
CLI>px from mininet.util import dumpNodeConnections
CLI>py dumpNodeConnections(net.hosts)
```

- If you get a 'px' command not found error then edit the /usr/local/lib/python2.7/dist-packages/mininet-2.0.0-py2.7.egg/mininet/cli.py and add the do_px() function. Source code beside this note.

```
def do_px(self, line):
  try:
    exec(line, globals(), 
self.locals)
  except Exception, e:
    output(string(e)+'\n')
```

Complex Interactive Example

```
$ sudo mn -v output
mininet> py net.addHost('h3')
<Host h3:  pid=3405>
mininet> py net.addLink(s1,
net.get('h3'))
<mininet.link.Link object at
0x1737090>
mininet> py s1.attach('s1-eth3')
mininet> py
net.get('h3').cmd('ifconfig h3-eth0
10.0.0.3')
mininet> h1 ping -c1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes
of data.
64 bytes from 10.0.0.3: icmp_req=1
ttl=64 time=1.91 ms
```

- Interactive but using python programming primitives for Mininet 2.0 and up.

Other Things

- `net = Mininet(controller=RemoteController, switch=OVSKernelSwitch)`
  - Ability to customize controller or switch
  - `mininet.node.RemoteController(ip=?, port=?)` provides a mechanism to use an external OpenFlow controller like pox.
  - `mininet.node.UserSwitch` provides a mechanism to use an user-space switch instead of the default Kernel switch.
  - `mininet.node.OVSSwitch` provides a mechanism to use Open vSwitch that comes pre-installed with mininet.
- `mininet.cli.CLI(<Mininet object>)`
  - Allows the script to enter interactive mode.
  - Normally placed before `<Mininet object>.stop()`

Exercises

- Write python scripts for the following topologies:
  - --topo tree,depth=2,fanout=3
  - --topo single,5
- Create your own network topology.
  - 2 switches connected to each other with 3 hosts each
- Which of the following networks “performs” better?
  - --topo tree,depth=2,fanout=2
  - --topo single,4
  - --topo linear,4
- Is there a performance difference between the user space switch and the kernel switch?

FAQ

1. What should I do if strange errors appear in POX (stack traces, core dumps)? *Exit both mininet and POX then try again. No need to logout of the Linux terminal or shutdown Virtualbox. But you need to redo the experiment for that particular switch.*
2. What if I get a “controller already running” or similar error? Exit mininet (if not already exited) then kill the existing controller process. This can be done by issusing the “sudo killall controller” on the Linux command line. This means the controller loaded previously wasn't terminated properly.
3. What should I do if after importing the virtual machine (via import appliance) is just hangs with a blank screen? *Delete existing Mininet-VM folder located in your user directory's VirtualBox VMs folder and re-import the virtual machine. On the review screen before selecting “Import”, you need to replace the Guest OS Type to “Linux Ubuntu 64-bit”.*
4. What should I do if OVF does not load when installing application in Virtualbox (via import appliance)? *Simply create a new VM (Mininet-VM, Linux, Ubuntu 64-bit 1024MB RAM and select existing disk and point to the VMDK in the same folder as the OVF file). Note: Delete existing Mininet-VM folder located in your user directory's VirtualBox VMs folder.*
5. What should I do if I cannot access the Mininet VM via SSH or Putty? *You might have not configured the Mininet VM in Virtualbox to support Host-only Adapter (In Virtualbox, Settings -> Network -> Attach to "host only adapter").*
6. What happens if "pingall" does not work or I get a command not found error? *This means that you are not running the command on a mininet console. Please launch mininet (mn) with its appropriate command line arguments first.*
7. What do I do if I get a stuck in "Waiting for network connection"? *You wait and wait a bit more then you can login, login to the VM and run "sudo rm /etc/udev/rules.d/*-persistent-net.rules" then reboot.*