Mininet Tutorial

James Won-Ki Hong, Jian Li, Seyeon Jeong

Dept. of Computer Science & Engineering

POSTECH

http://dpnm.postech.ac.kr/~jwkhong

jwkhong@postech.ac.kr

Outline

- **Introduction**
  - Motivation
- **Installation**
- **Mininet Tutorial**
  - Command line interface usage
  - Application programming interface usage
- **Demo**
- **Experiences on Mininet**

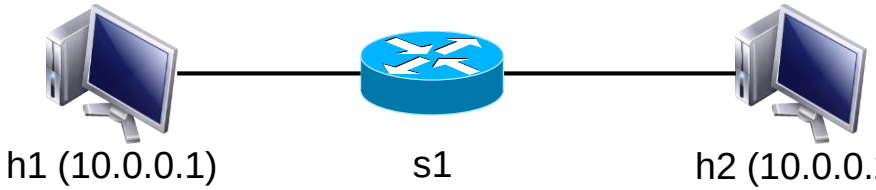
Types of Network Testbeds

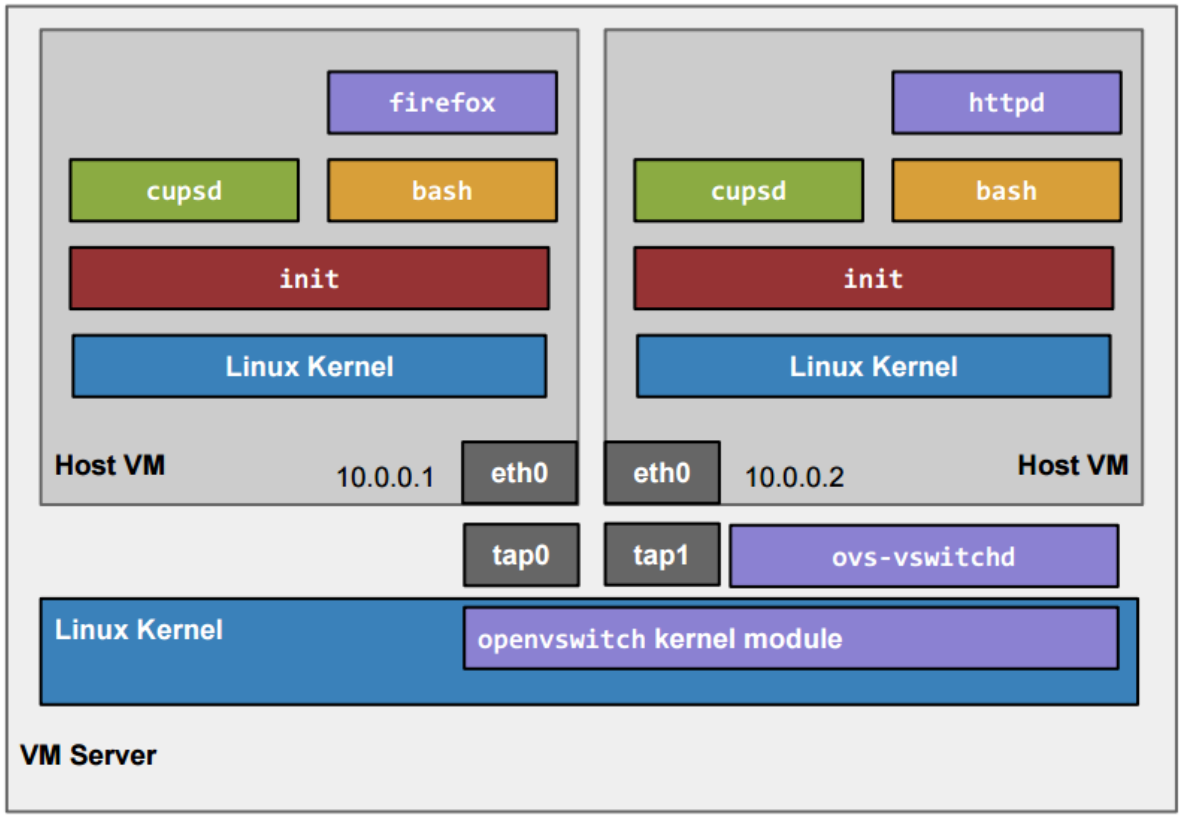
❖ Platforms for Network/Systems Teaching

Platform	Advantages	Disadvantages
Hardware testbed	▪ Fast ▪ Accurate: “ground truth”	▪ Expensive ▪ Hard to reconfigure ▪ Hard to change ▪ Hard to download
Simulator	▪ Inexpensive, flexible ▪ Detailed ▪ Easy to download ▪ Virtual time	▪ May require app changes ▪ Might not run OS code ▪ May not be “believable” ▪ May be slow/non-interactive
Emulator	▪ Inexpensive, flexible ▪ Real code ▪ Reasonably accurate ▪ Easy to download ▪ Fast/interactive usage	▪ Slower than hardware ▪ Possible inaccuracy from multiplexing

Hardware based Network Testbed (1/2)

❖ Very Simple Network using Full System Virtualization

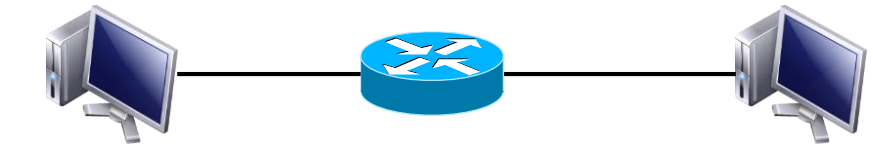




Hardware based Network Testbed (2/2)

❖ Problems

- Too much work even for creating such a simple network topology
- Not programmable



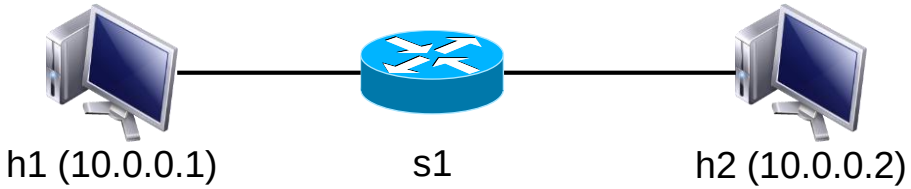

```
sudo bash
# Create host namespaces
ip netns add h1
ip netns add h2
# Create switch
ovs-vsctl add-br s1
# Create links
ip link add h1-eth0 type veth peer name s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2
ip link show
# Move host ports into namespaces
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
ip netns exec h1 ip link show
ip netns exec h2 ip link show
```

```
# Connect switch ports to OVS
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
ovs-vsctl show
# Set up OpenFlow controller
ovs-vsctl set-controller s1 tcp:127.0.0.1
ovs-controller ptcp: &
ovs-vsctl show
# Configure network
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h1 ifconfig lo up
ip netns exec h2 ifconfig h2-eth0 10.2
ip netns exec h1 ifconfig lo up
ifconfig s1-eth1 up
ifconfig s1-eth2 up
# Test network
ip netns exec h1 ping -c1 10.2
```

Emulator based Network Testbed

What We Want is

- A simple command-line tool / API which can ease the work
- The solution should allow us to easily create topologies for varying size, up to hundreds and thousand of nodes



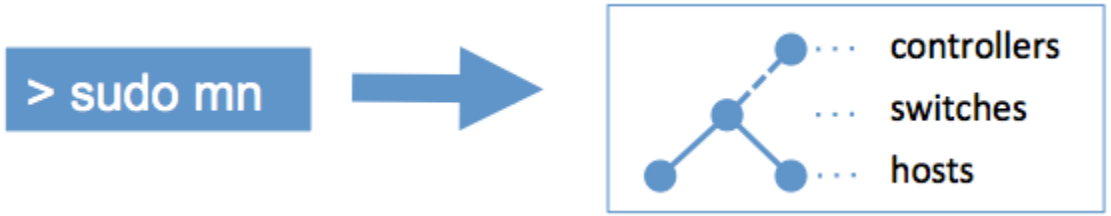

```
h1 = net.addHost( 'h1' )
h2 = net.addHost( 'h2' )
s1 = net.addSwitch( 's1' )
c0 = net.addController( 'c0' )
net.addLink( h1, s1 )
net.addLink( h2, s1 )
net.start()
CLI( net )
```

Topology Generation using Mininet API

Introduction to Mininet

❖ Mininet

- A network emulator which creates realistic virtual network
- Runs real kernel, switch and application code on a single machine



- Provides both Command Line Interface (CLI) and Application Programming Interface (API)
  - CLI: interactive commanding
  - API: automation

- Abstraction
  - Host: emulated as an OS level process
  - Switch: emulated by using software-based switch
    - E.g., Open vSwitch, SoftSwitch

Mininet Installation (1/4)

❖ Mininet VM Installation

- The easiest and most fool-proof way of installing Mininet
- Procedures
  - Download the Mininet pre-installed VM image
  - Download and install one of the hypervisors (e.g., VirtualBox, Qemu, VMware Workstation, VMware Fusion, or KVM)
  - Import VM image into selected hypervisor

❖ Native Installation from Source (Ubuntu)

- Procedures
  - Download source from github

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

  - Full installation: Mininet + Open vSwtich + wireshark + etc.

```
$ mininet/util/install.sh -a
```

  - Minimum installation: + Mininet + Open vSwitch

```
$ mininet/util/install.sh -fnv
```

Mininet Installation (2/4)

❖ Native Installation from Package (Ubuntu)

- Recommended OS: Ubuntu 12.04 and later
- Procedures
  - Remove all previously installed Mininet and Open vSwitch

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

  - Install Mininet package according to your Ubuntu version (choose one of them!)

```
$ sudo apt-get install mininet                          ← Ubuntu 13.04
$ sudo apt-get install mininet/quantal-backports        ← Ubuntu 12.10
$ sudo apt-get install mininet/precise-backports        ← Ubuntu 12.04
```

  - Deactive OpenvSwitch controller if it is running

```
$ sudo service openvswitch-controller stop
$ sudo update-rc.d openvswitch-controller disable
```

  - You can also install additional software from mininet source

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

Mininet Installation (3/4)

❖ Mininet Installation (CentOS)

- Download mininet source from github repository

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

- Edit installation script which located in the path of mininet/util/install.sh

```
test -e /etc/centos-release && DIST="CentOS"
if [ "$DIST" = "CentOS" ]; then
    install='sudo yum -y install'
    remove='sudo yum -y erase'
    pkginst='sudo rpm -ivh'
    # Prereqs for this script
    if ! which lsb_release &> /dev/null; then
        $install redhat-lsb-core
    fi
fi
```

  - Add CentOS into following line

```
if ! echo $DIST | egrep 'Ubuntu|Debian|Fedora|CentOS|RedHatEnterpriseServer|
SUSE LINUX'; then
```

- Install mininet

```
# mininet/util/install.sh -nf
```

Mininet Installation (4/4)

❖ OpenvSwitch Installation (CentOS)

- Get the dependencies needed to build OVS

```
# yum -y install gcc make python-devel openssl-devel kernel-devel
graphviz \ kernel-debug-devel autoconf automake rpm-build redhat-rpm-
config \ libtool
```

- Build RPM binary from OVS source

```
# mkdir -p ~/rpmbuild/SOURCES/
# cd ~/rpmbuild/SOURCES/
# wget http://openvswitch.org/releases/openvswitch-2.5.0.tar.gz
# tar zxvf openvswitch-2.5.0.tar.gz
# cd openvswitch-2.5.0
# rpmbuild -bb --without check rhel/openvswitch.spec
# rpm -ivh --nodeps ~/rpmbuild/RPMS/x86_64/openvswitch*.rpm
```

- Launch OVS and check OVS version

```
# systemctl start openvswitch
# ovs-vsctl show
1d0e71af-d5ff-4c7b-8224-2804849824c3
    ovs_version: "2.5.0"
```

Supplementary Tools Installation

❖ Mininet Command Line Interface Usage

- Interact with hosts and switches
  - Start a minimal topology

```
$ sudo mn
```

  - Start a minimal topology using a remote controller

```
$ sudo mn --controller=remote,ip=[IP_ADDDR],port=[listening port]
```

  - Start a custom topology

```
$ sudo mn --custom [topo_script_path] --topo=[topo_name]
```

  - Display nodes

```
mininet> nodes
```

  - Display links

```
mininet> net
```

  - Dump information about all nodes

```
mininet> dump
```

Mininet Tutorial (2/7)

❖ Mininet Command Line Interface Usage

- Interact with hosts and switches
  - Check the IP address of a certain node

```
mininet> h1 ifconfig -a
```

  - Print the process list from a host process

```
mininet> h1 ps -a
```

- Test connectivity between hosts
  - Verify the connectivity by pinging from host0 to host1

```
mininet> h1 ping -c 1 h2
```

  - Verify the connectivity between all hosts

```
mininet> pingall
```

Mininet Tutorial (3/7)

❖ Mininet Command Line Interface Usage

- Run a regression test
  - Traffic receive preparation

```
mininet> iperf -s -u -p [port_num] &
```

  - Traffic generation from client

```
mininet> iperf -c [IP] -u -t [duration] -b [bandwidth] -p [port_num] &
```

- Link variations

```
$ sudo mn -link tc,bw=[bandwidth],delay=[delay_in_millisecond]
```

- Python Interpreter
  - Print accessible local variables

```
$ py locals()
```

  - Execute a method through invoking mininet API

```
$ py [mininet_name_space].[method]
```

Mininet Tutorial (4/7)

❖ Mininet Application Programming Interface Usage

- Low-level API: nodes and links
  - mininet.node.Node
    - A virtual network node, which is a simply in a network namespace
  - mininet.link.Link
    - A basic link, which is represented as a pair of nodes

Class	Method	Description
Node	MAC/setMAC	Return/Assign MAC address of a node or specific interface
	IP/setIP	Return/Assign IP address of a node or specific interface
	cmd	Send a command, wait for output, and return it
	terminate	Send kill signal to Node and clean up after it
Link	Link	Create a link to another node, make two new interfaces

```
h1 = Host( 'h1' )
h2 = Host( 'h2' )
s1 = OVSSwitch( 's1', inNamespace=False )
c0 = Controller( 'c0', inNamespace=False )
Link( h1, s1 )
Link( h2, s1 )
h1.setIP( '10.1/8' )
h2.setIP( '10.2/8' )
```

```
c0.start()
s1.start( [ c0 ] )
print h1.cmd( 'ping -c1', h2.IP() )
s1.stop()
c0.stop()
```

Mininet Tutorial (5/7)

❖ Mininet Application Programming Interface Usage

- Middle-level API: network object
  - mininet.net.Mininet
    - Network emulation with hosts spawned in network namespaces

Class	Method	Description
Net	addHost	Add a host to network
	addSwitch	Add a switch to network
	addLink	Link two nodes into together
	addController	Add a controller to network
	getNodeByName	Return node(s) with given name(s)
	start	Start controller and switches
	stop	Stop the controller, switches and hosts
	ping	Ping between all specified hosts and return all data

```
net = Mininet()
h1 = net.addHost( 'h1' )
h2 = net.addHost( 'h2' )
s1 = net.addSwitch( 's1' )
c0 = net.addController( 'c0' )
net.addLink( h1, s1 )
net.addLink( h2, s1 )
```

```
net.start()
print h1.cmd( 'ping -c1', h2.IP() )
CLI( net )
net.stop()
```

Mininet Tutorial (6/7)

❖ Mininet Application Programming Interface Usage

- High-level API: topology templates
  - mininet.topo.Topo
    - Data center network representation for structured multi-trees

Class	Method	Description
Topo	Methods similar to net	E.g., addHost, addSwitch, addLink,
	addNode	Add node to graph
	addPort	Generate port mapping for new edge
	switches	Return all switches
	Hosts/nodes/switches/links	Return all hosts
	isSwitch	Return true if node is a switch, return false otherwise

```
class SingleSwitchTopo( Topo ):
    "Single Switch Topology"
    def build( self, count=1):
        hosts = [ self.addHost( 'h%d' % i )
            for i in range( 1, count + 1 ) ]
        s1 = self.addSwitch( 's1' )
        for h in hosts:
            self.addLink( h, s1 )
```

```
net = Mininet( topo=SingleSwitchTopo( 3 ) )
net.start()
CLI( net )
net.stop()
```

Mininet Tutorial (7/7)

❖ Mininet Application Programming Interface Usage

- Customized topology

```
# cat custom.py

LEN_DPID = 16
from mininet.topo import Topo
class MyTopo( Topo ):

    def name_dpid( self, index ):
        dpid = '%02d' % ( index )
        zeros = '0' * ( LEN_DPID - len( dpid ) )
        name = 's%02d' % ( index )
        return { 'name':name, 'dpid':zeros + dpid }

    def build( self, count=1):
        hosts = [ self.addHost( 'h%d' % i )
                  for i in range( 1, count + 1 ) ]
        s1 = self.addSwitch( **self.name_dpid(1) )
        for h in hosts:
            self.addLink( h, s1 )

topos = { 'mytopo': MyTopo }

# mn --custom custom.py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
```

More examples can be found here:
https://github.com/mininet/mininet/tree/master/examples

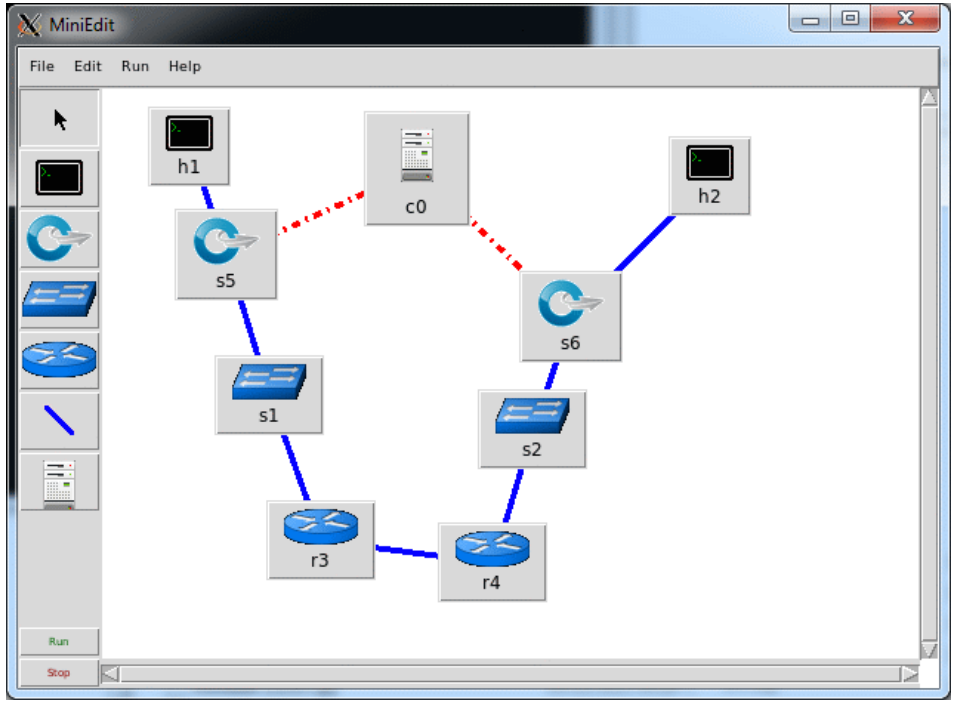
Mininet Applications

❖ MiniEdit

- A GUI application which eases the Mininet topology generation
- Either save the topology or export as a Mininet python script

❖ Visual Network Description (VND)

- A GUI tool which allows automate creation of Mininet and OpenFlow controller scripts



References

1. Mininet: http://mininet.org/
2. Floodlight: http://www.projectfloodlight.org/floodlight/