Introduction to mINInet

Introduction

Mininet is a network emulator for creating virtual hosts, switches and controllers on a Linux OS, which support OpenFlow for SDN.

Part I: <http://mininet.org/walkthrough/>

Part II: <http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/>

part I – Starting mininet

1. Create an instance titled “INFR4599U Student” from the HRL cloud and connect to it via the console.
2. Start Mininet by typing: sudo mn. This command creates a default topology consisting of two hosts h1 and h2 connected through switch s1. It also starts controller c0. Then, you will get the Mininet prompt.

|  |
| --- |
| 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> |

At this point you can run mininet’s internal commands. Type “help” and review some of the available commands. Try “nodes”, “net” and “dump”.

1. You can execute a limited set of Linux functions on the hosts. For example, check the IP configuration of host h1 by typing the following command:

|  |
| --- |
| mininet> h1 ifconfig -a |

What interfaces do you see when you execute the above command?

1. you should be able to successfully ping all hosts as below:

|  |
| --- |
| mininet> pingall  \*\*\* Ping: testing ping reachability  h1 -> h2  h2 -> h1  \*\*\* Results: 0% dropped (2/2 received) |

1. Now let’s capture some traffic on mininet. First, start terminal windows on h1 and h2 using the following commands:

|  |
| --- |
| mininet> xterm h1  mininet> xterm h2 |

These commands should open two terminal windows, one for h1 and one for h2. Check the IP addresses for h1 and h2.

In the h1 terminal, run wireshark:

|  |
| --- |
| root@mininet-vm:~# wireshark & |

Start capturing traffic on h1-eth0 interface.

In the h2 terminal, start tcpdump on interface h2-eth0:

|  |
| --- |
| root@mininet-vm:~# tcpdump -I h2-eth0 |

Now ping h2 from h1 (you can do it from the mininet interface, or directly from the h1 terminal using h2’s IP address) and observe the traffic on Wireshark and tcpdump.

|  |
| --- |
| mininet> h1 ping -c 10 h2 |

Mininet hosts can run any command or application that is available to the underlying Linux system (or VM) and its file system. You can also enter any bash command, including job control (&, jobs, kill, etc..)

Next, try starting a simple HTTP server on h1, making a request from h2, then shutting down the web server:

|  |
| --- |
| mininet> h1 python -m SimpleHTTPServer 80 &  mininet> h2 wget -O - h1 *(careful of the spaces here!)*  ...  mininet> h1 kill %python |

Observe the HTTP traffic on Wireshark.

1. Mininet can be used with various built-in topologies, as well as custom topologies. In this lab we experiment with Mininet’s built-in topologies. You will learn about creating custom topologies in the next lab.

*Exit out of the existing Mininet environment you have by typing in ctrl-z at the prompt, and running sudo mn -c*

You can call Mininet with specific built-in topologies. The following command creates a network with one switch and three hosts, and runs pings between pairs of hosts:

|  |
| --- |
| $ sudo mn --test pingall --topo single,3 |

You can also create a linear network with 4 switches (connected in a line) with the following command:

|  |
| --- |
| $ sudo mn --topo linear,4 |

Type ‘net’ in Mininet shell to examine the nodes and links.

Try the following commands and examine the topology to see many switches and hosts are created in each case and how are they connected.

|  |
| --- |
| $ sudo mn --topo tree,2 |
| $ sudo mn --topo tree,3 |

1. Running Mininet with an external controller

The SDN controller is responsible for making control and management plane decisions, such as routing, flow table management, collecting statistics etc. Mininet has an internal controller that can be used with its OpenFlow switches. However, it is also possible to use external controllers with Mininet. The following include some examples of SDN controllers:

* Nox
  + First generation OpenFlow controller; stable and widely uses
  + Comes in two flavors:
    - NOX-Classic: C++/Python (no longer supported)
    - Nox – C++ only
* Pox
  + NOX in Python (as a result, not the best performing Controller)
  + Widely used and maintained
  + Code is easy to read and write for this type of Controller
  + Commonly used in research, experimentation and learning SDN concept
* Ryu
  + Implemented in Python (as a result, not the best performing Controller)
  + Works with OpenStack
* Floodlight
  + Java-based controller
  + Integrates with REST API and OpenStack
  + Production-level performance

In this lab we work primarily with Pox. A version of Pox has been included in the Mininet VM.

Next, open two terminals on the VM from the Console.

In the first. window, start the pox controller using the following command. It creates a self-learning controller running on local machine port 6633.

|  |
| --- |
| $cd /home/ubuntu/pox  $./pox.py samples.pretty\_log forwarding.l2\_learning log.level --DEBUG |

In Mininet window, start Mininet with 3 switches and 3 hosts in a linear network and a remote controller.

|  |
| --- |
| $ sudo mn --topo linear,3 --mac -r--switch ovsk --controller remote |

What do you see in the controller window when Mininet starts?

In Mininet window, try “net” to check the topology. Then have all hosts ping each other. The results should be successful.

part II –minIEDIT GUI

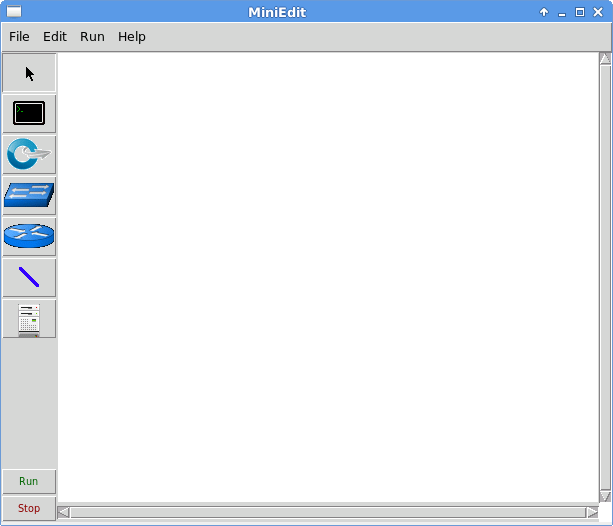
The Mininet network emulator includes MiniEdit, a simple GUI editor for Mininet.

1. The MiniEdit script is located in Mininet’s examples folder. To run MiniEdit, execute the following command on the console of your VM:

|  |
| --- |
| $ sudo python home/ubuntu/mininet/examples/miniedit.py |

Mininet needs to run with root privileges so we started MiniEdit using the sudo command.

MiniEdit has a simple user interface that presents a canvas with a row of tool icons on the left side of the window, and a menu bar along the top of the window.

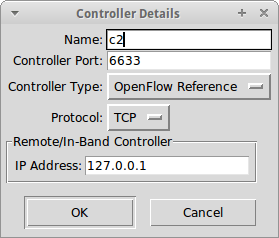


1. The toolbar on the left allows you to select objects, create host nodes, create OpenFlow switch nodes, create legacy switches, create legacy routers, and create network links. Use the tool bar to create the following network:

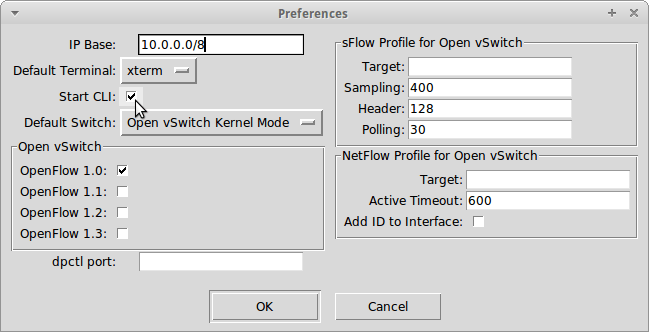


Note that the thick blue lines represent the data plane links between switches and hosts, while the dashed red lines represent the control plane links between controllers and switches. The control plane links may be in-band or out-of-band.

1. Right-click on each controller and select Properties from the menu that appears. The default port number for each controller is 6633. Change this so the port numbers used by controllers c0 and c1 are 6633, and 6634 respectively.



1. Click on Edit→Preferences to open MiniEdit preferences. By default, the MiniEdit console window does not give the user access to the Mininet command line interface. If you want to be able to use the Mininet CLI when a simulation is running, check the Start CLI box. You may also set the version of OpenFlow you will use.



1. To start the simulation scenario, click the Run button on the MiniEdit GUI. In the terminal window from which you started MiniEdit, you will see some messages showing the progress of the simulation startup and then the MiniEdit CLI prompt.



1. Challenge: repeat the network traffic capture scenario (Part I - #6) in this MiniEdit network environment through the following steps:

* Run Wireshark on h1
* Run tcpdump on h8
* ping h8 from h1