Hands-on Lab Description

CS-NET-00007 – Mininet Lab

Category:

```
CS-NET: Computer Networking
```

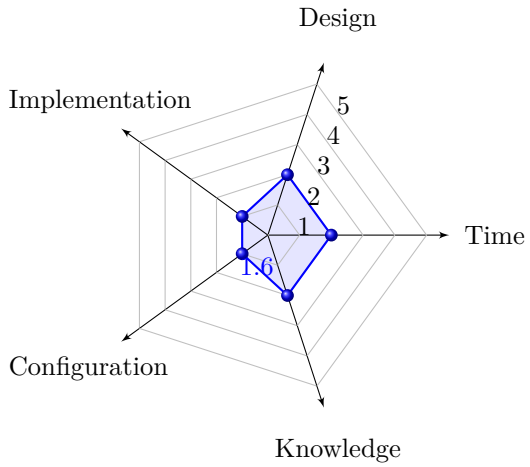
Objectives:

```
1 Learn some basic Mininet Topologies Setup
2 Learn multiple Mininet commands to get familiar with the Mininet network emulator.
```

Estimated Lab Duration:

```
1 Expert: 20 minutes
2 Novice: 35 minutes
```

Difficulty Diagram:



Difficulty Table.

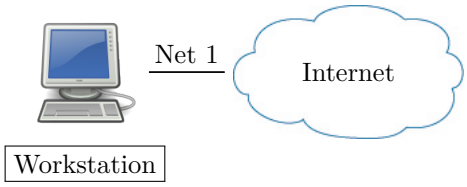
Measurements	Values (0-5)
Time	2
Design	2
Implementation	1
Configuration	1
Knowledge	2
Score (Average)	1.6

Required OS:

```
Linux: Ubuntu 18.04 LTS
```

Lab Running Environment:

```
VirtualBox https://www.virtualbox.org/ (Reference Labs: CS-SYS-00101)
```




```
1  Workstations: Linux (Ubuntu 18.04 LTS)
2  Internet: Accessible
```

Lab Preparations:

```
1  Know how to use Linux OS and Linux commands (Reference Lab: CS-SYS-00001)
2  Basic knowledge about computer networking (Reference Lab: CS-NET-00001, CS-NET-00002)
3  Open Virtual Switch (OVS) (Reference Lab: CS-NET-00006)
```

Task 1.0 Initial setup, testing, and basic mininet commands

In this task, you need to access the SDN VM and check the installation of *mininet* by creating an default minimal topology contains one OpenFlow kernel switch with two hosts (h1 & h2) and OpenFlow reference controller.

First, you need to check if *mininet* (mn) is installed in the Linux VM

```
$    mn --version   % check if mininet is installed or not. If not, please execute
       the next command
$    sudo apt install mininet % install {mininet from the distribution
```

To test *mininet* setup, you can use the following command, and it will create a temporary *mininet* topo and automatically ping all the nodes in that topo, then it will exit and clean temp topo. Note that the system will show *mininet* > header when entering the networking environment. If you entered *containernet*, *containernet* > will be presented, which is the same as you are in a *mininet* topology, it may vary depends on the installation of the *mininet* or *containernet*.

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

It will create two hosts and a switch and issue *ping* between hosts. If it shows successful of pings, then *mininet* is installed properly.

You will start to use some basic *mininet* commands to familiarize yourself with the *mininet* emulator.

1. Create a single *mininet* topology with a single switch and three hosts:

   ```
   $       sudo mn --topo=single,3
   ```

2. Exit and Clean from current *mininet* topology:

   ```
        mininet> exit
   ```

   It is recommended to clean it up after you have done using *mininet* topology or *mininet* crashes for some other reasons.

   ```
   $       sudo mn -c
   ```

Task 2.0 Create Various mininet Topologies

In this task, you will follow the similar steps as **Task1** to create some different *mininet* topologies based on *mininet* commands as similar as last part.

1. You can create a reversed *mininet* topology with a single switch and three hosts, where the topology setup is illustrated in Figure CS-NET-00007.2.

   ```
        sudo mn --topo=single,3
   ```

2. You can create a linear *mininet* topology with four switches and a single host per switch, where the topology setup is illustrated in Figure CS-NET-00007.3

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

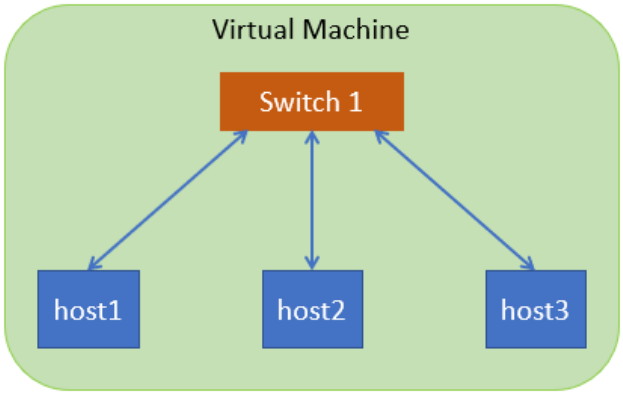


Figure CS-NET-00007.2
Mininet Topology with A Single Switch and Three Hosts.

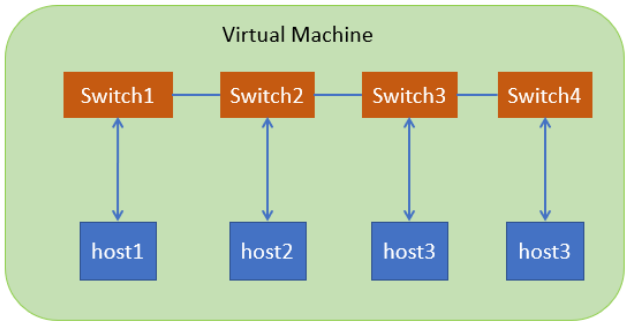


Figure CS-NET-00007.3
Mininet Topology with Four Switches and A Single Host per Switch.

3. You can create a tree *mininet* topology with a depth of 2 and fanout of 3, where the topology setup is illustrated in Figure CS-NET-00007.4

```
sudo mn --topo tree,depth=2,fanout=3
```

4. (Optional) You can customize your own *mininet* topology by using *mininet* visualization creator

```
$       cd mininet/mininet/examples # go to directory of mininet and jump to
          examples folder
$       python miniedit.py # open topology creator
```

Task 3.0 Advanced mininet Usage

1. Display *mininet* help for *mininet* startup options (Review options such as *–controller=remote* and *–topo=tree*)

```
$       sudo mn -h
```

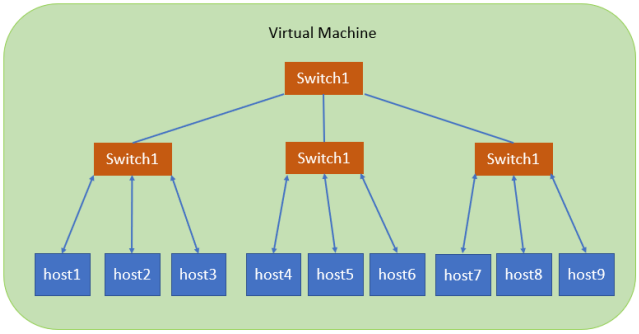


Figure CS-NET-00007.4
Mininet Topology with A Depth of Two and Fanout of Three Hosts.

2. Start an example mininet topology consisting of a single switch and 2 hosts using the *sudo mn* command.

```
$       sudo mn
```

3. Learning the functionalities of some useful *mininet* build-in commands.

   Create a linear *mininet* topology with four switches and a single host per switch

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

   Display *mininet* Nodes. The topology consist of nine hosts, one controller and 4 switches

```
    mininet> nodes
```

   Perform a bandwidth measurement with an iPerf2 remote station

```
    mininet> iperf
```

   Display detailed mininet information. The output displays IP addresses of devices in the mininet topology and the process numbers used (your output may be different)
   Note: Try command *dump*

   Display link information. The output shows that host h1 is connected to switch s1 and h2 to s2, h3 to s3, and h4 to s4 respectively.

```
    mininet> net
```

   Display h1 IP address information

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

   Next, please show the IP addresses information of h2, h3, h4 individually

   The ping command sends packets of data to a specific IP address on a network, and then lets you know how long it took to transmit that data and get a response.

```
    mininet> h1 ping h2 # h1 can ping h2, stop the continuous ping using the key
        sequence Ctrl-C
    mininet> h1 ping -c 10 h2 #send 10 pings from h1 to h2
```

   Display the IP address information for s1

```
mininet> s1 ifconfig -a
```

Next, please show the IP address information for the rest of switches in this *mininet* topology

Display ARP information on h1 and s1

```
mininet> h1 arp
mininet> s1 arp
```

Display the routes table information for specific hosts and switches

```
mininet> h1 route
mininet> s1 route
...
```

Exit the current *mininet* topology and clean up

```
mininet> exit
mininet> sudo mn -c
```

Next, please recreate *mininet* topology mentioned in Task 2 and try all the commands introduced in this task and see what will happen

4. Use the link in *mininet*

   mininet allows you to set link parameters, and these can be set automatically from the command line. In this case, we set the Bandwidth (bw) and Traffic Control (tc) to 10 and set delay time to 10 milliseconds.

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

   We can use link to manually disable the connection between a switch and a host. To disable both halves of a virtual ethernet pair

```
mininet> link s1 h1 down
```

   To bring the link back up:

```
mininet> link s1 h2 up
```

5. Namespaces and xterms By default, only the hosts are put in a separate namespace; the window for each switch is unnecessary (that is, equivalent to a regular terminal), but can be a convenient place to run and leave up switch debug commands, such as flow counter dumps.

   Start *mininet* with *-x* (xterm option)

```
$       sudo mn -x
```

   After a second, the xterms will pop up, with automatically set window names. Now in the xterm terminal labeled ”host h1”, try to ping another host in the current *mininet* topology

```
host1>  ping 10.0.0.2
```

6. Other switch types can be utilized in the *mininet* topology by adding *–switch SWITCH_NAME*. The user-space switch *–switch user* can be a great starting point for implementing new functionality, especially where software performance is not critical. Also, the user-space switch has much lower TCP bandwidth

compare with normal kernel switch because the packets have additional kernel-to-user-space transitions. In this case, we implement a preinstalled switch called Open vSwitch(OVS) with the *mininet* topology along with the iperf to measure the TCP bandwidth.

```
$       sudo mn --switch ovsk --test iperf
```

Next, try to run the iperf on the *mininet* topology with user-space switch and OpenFlow kernel model to compare the bandwidth between each of them.

7. You can also invoke python interpreter within your *mininet* topology by starting with *py* in *mininet* command line. It can help us to extend *mininet*, and discover some insights about the it's working. Note: the switches, nodes and controllers are represented by an unique associated node object.

   Print *hello world* to check your environment

   ```
   mininet> py 'hello ' + 'world'
   ```

   Print the accessible local variables

   ```
   mininet> py locals()
   ```

   Using dir() to check the property of a node

   ```
   mininet> py dir()
   ```

   Read the on-line documentation for methods available on a node by using the help() function

   ```
   mininet> py help(h1)
   ```

   Note: press *q* to exit the scenario

   Check the IP address of a host

   ```
   mininet> py h1.IP()
   ```

Next, Try to implement the above commands in your own *mininet* topology.

Lab Assessments (100 points)

Lab assessment for accomplishing Task 1, Task 2, and Task3 depends on the following facts:

1. (25 points) Setting up *mininet* and Running *mininet* topology
   - can correctly create the topology with a single switch and four hosts

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

     ```
             (h1, s1) (h2, s1) (h3, s1) (h4, s1)
     ```

   - can correctly create the linear topology with five switch and one host for every switch

     ```
     $         sudo mn --topo=linear,5
     ```

```
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4,
    s3) (s5, s4)
```

2. (40 points) For each topology above, they should be able to

   - show the correct number of nodes within the current topology
     Single switch and four hosts:

     ```
     mininet> nodes
     available nodes are:
     c0 h1 h2 h3 h4 s1
     ```

     Linear topology of five switch and one host for each switch:

     ```
     mininet> nodes
     available nodes are:
     c0 h1 h2 h3 h4 h5 s1 s2 s3 s4 s5
     ```

   - perform bandwidth measurement

     ```
     mininet> iperf
     *** Iperf: testing TCP bandwidth between h1 and h4
     .*** Results: ['46.3 Gbits/sec', '46.3 Gbits/sec']
     ```

     ```
     mininet> iperf
     *** Iperf: testing TCP bandwidth between h1 and h5
     *** Results: ['37.0 Gbits/sec', '37.1 Gbits/sec']
     ```

   - display the correct link information among hosts and switches

     ```
     mininet> net
     h1 h1-eth0:s1-eth1
     h2 h2-eth0:s1-eth2
     h3 h3-eth0:s1-eth3
     h4 h4-eth0:s1-eth4
     s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
         s1-eth4:h4-eth0
     c0
     ```

     ```
     mininet> net
     h1 h1-eth0:s1-eth1
     h2 h2-eth0:s2-eth1
     h3 h3-eth0:s3-eth1
     h4 h4-eth0:s4-eth1
     h5 h5-eth0:s5-eth1
     s1 lo: s1-eth1:h1-eth0 s1-eth2:s2-eth2
     s2 lo: s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
     s3 lo: s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
     s4 lo: s4-eth1:h4-eth0 s4-eth2:s3-eth3 s4-eth3:s5-eth2
     s5 lo: s5-eth1:h5-eth0 s5-eth2:s4-eth3
     c0
     ```

3. (35 points) Create another tree topology of depth 2 and fanout 8,

   - the *host*1 can correctly ping the *host*64

```
mininet> h1 ping -c 2 h64
PING 10.0.0.64 (10.0.0.64) 56(84) bytes of data.
64 bytes from 10.0.0.64: icmp_seq=1 ttl=64 time=50.5 ms
64 bytes from 10.0.0.64: icmp_seq=2 ttl=64 time=1.45 ms
```

The challenging task evaluation is optional and the instructor can decide on how to evaluate students' performance.