Lab 7 (Week 7)
Building Network Topology with Mininet

CAN201

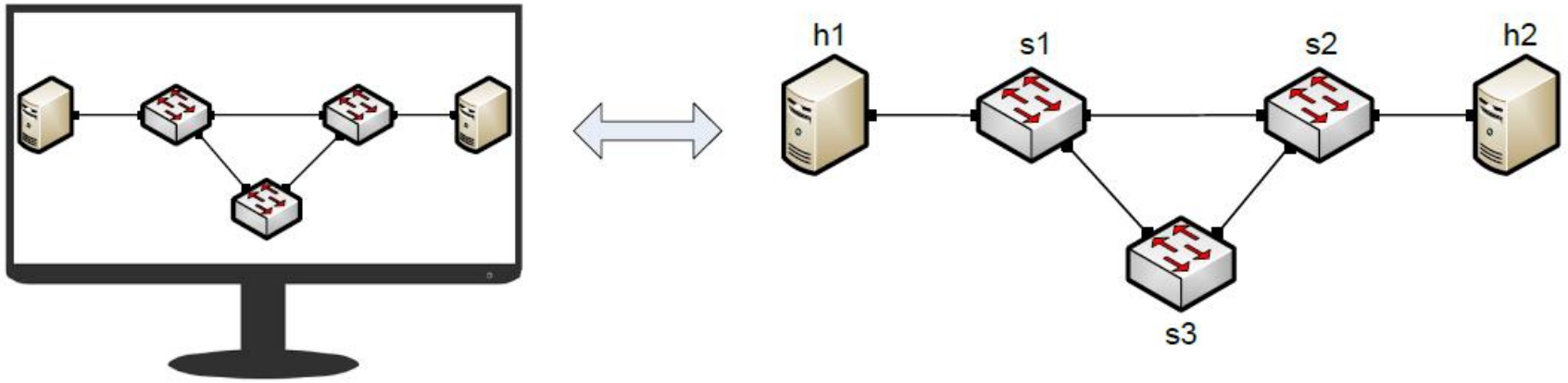
Dr. Gordon Boateng and Dr. Fei Cheng

Outline

- Introduction to Mininet
- Mininet CLI commands
- Build Custom Network Topology in Mininet
- Demo

Introduction to Mininet

- Mininet: a virtual testbed used for testing network tools and protocols.

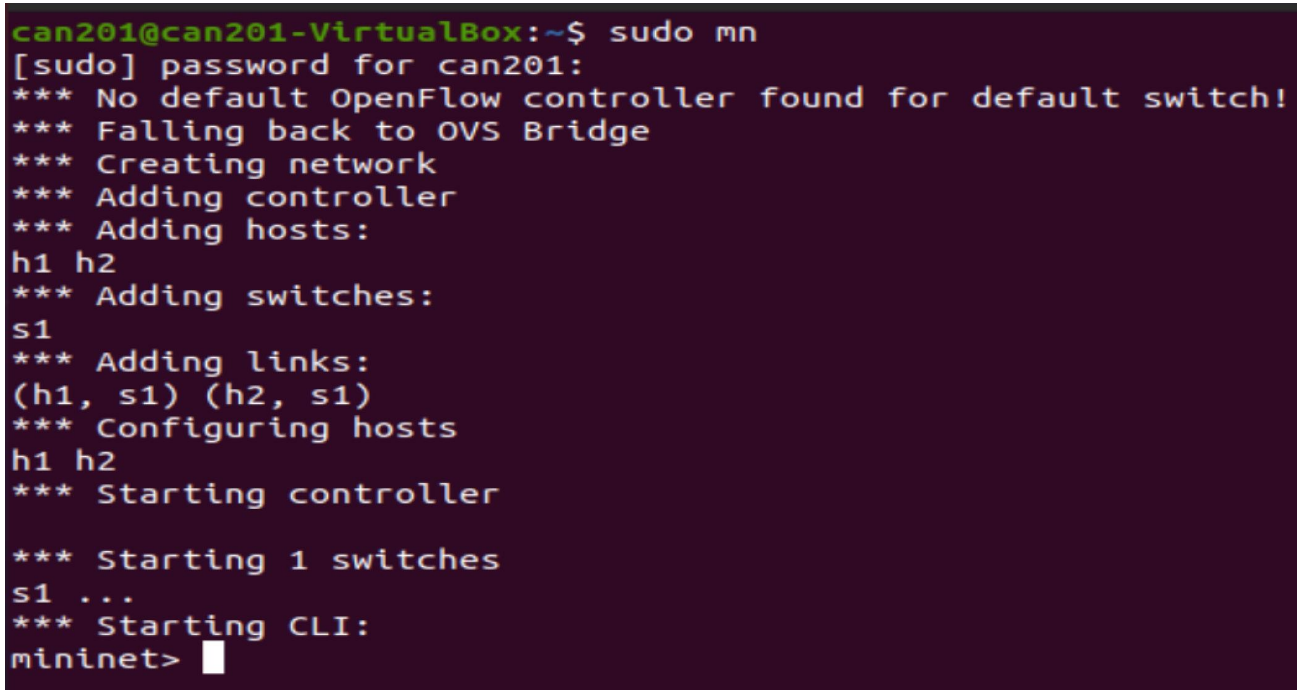


Introduction to Mininet

- Mininet offers the following features:
  - Fast prototyping for new networking protocols.
  - Simplified testing for complex topologies without the need of buying expensive hardware.
  - Realistic execution as it runs real code on the Unix and Linux kernels.
  - Open-source environment backed by a large community contributing extensive documentation.

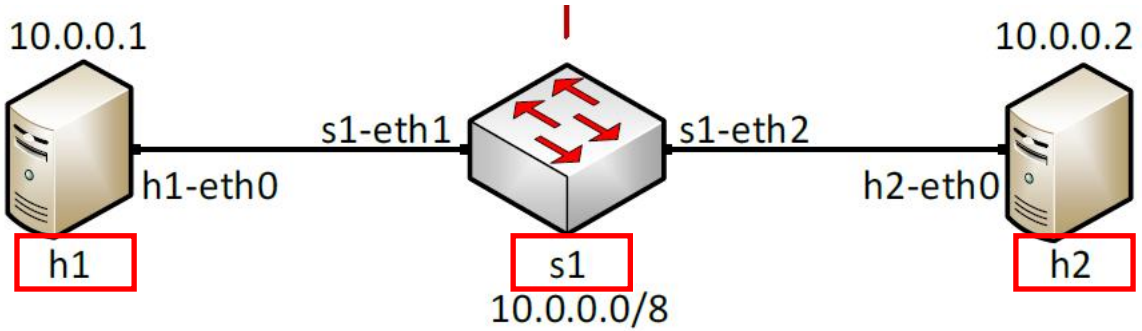
Introduction to Mininet

- Install Mininet in Ubuntu (don't need to do for CAN201-Default.ova):
  - $ sudo apt-get install mininet
- Test if it is installed successfully
  - $ sudo mn



Introduction to Mininet

- The default minimal network topology




```
can201@can201-VirtualBox:~$ sudo mn
[sudo] password for can201:
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Mininet CLI commands

- Command line interface:

  mininet>

- Command “help” shows the list of Mininet CLI commands and examples on their usage:

  Mininet> help

```
s1 ...
*** Starting CLI:
mininet> help

Documented commands (type help <topic>):
========================================
EOF     gterm   iperfudp   nodes         pingpair      py       switch
dpctl   help    link       noecho        pingpairfull  quit     time
dump    intfs   links      pingall       ports         sh       x
exit    iperf   net        pingallfull   px            source   xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

mininet>
```

Mininet CLI commands

- To display the available nodes, type the following command

  mininet> nodes

```
mininet> nodes
available nodes are:
h1 h2 s1
```

- To display the links to understand the topology, type the command

  mininet> net

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
```

Mininet CLI commands

- Mininet allows you to execute commands on a specific device. To issue a command for a specific node, you must specify the device first, followed by the command.

  mininet> h1 ifconfig

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::94f4:a9ff:febe:93cc  prefixlen 64  scopeid 0x20<link>
        ether 96:f4:a9:be:93:cc  txqueuelen 1000  (Ethernet)
        RX packets 47  bytes 4820 (4.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15  bytes 1146 (1.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Mininet CLI commands

- Other Mininet CLI commands
  - To turn on a terminal of a node

    Mininet> xterm h1
  - To exit from the command line interface

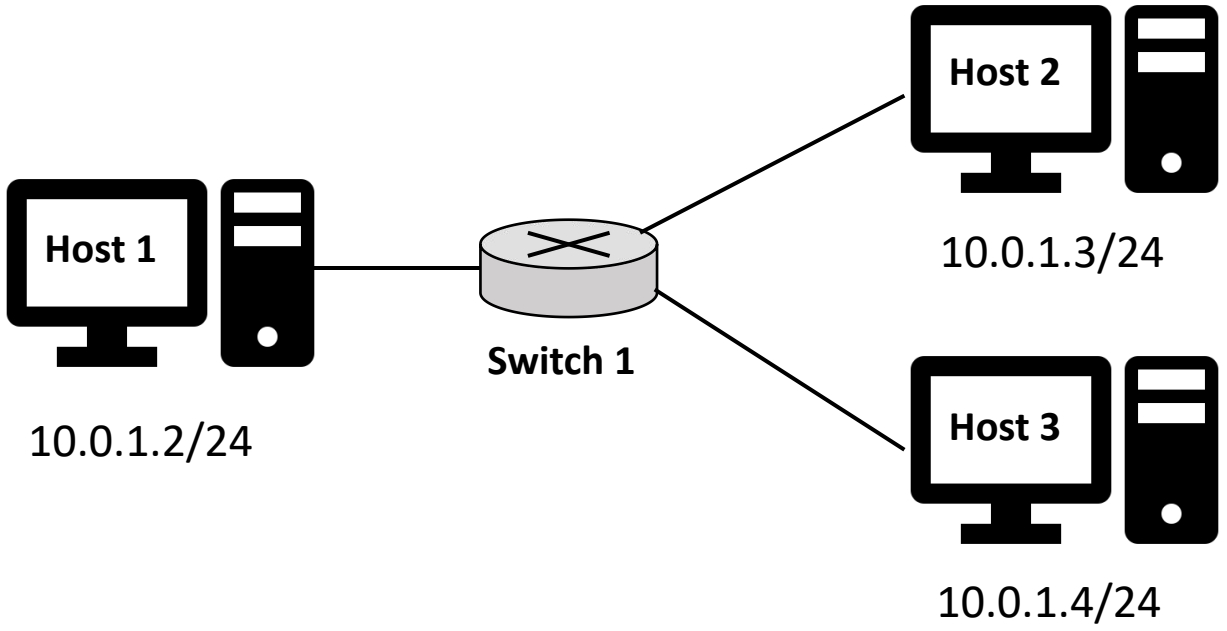
    Mininet> exit
- Other Host shell commands
  - To clean up the Mininet network configuration

    $sudo mn -c
  - To start the network with all terminals open

    $sudo mn -x

Build Custom Network Topology in Mininet

- Mininet supports custom network topologies.
- You can create a flexible topology (in Python) with just a few lines of code).

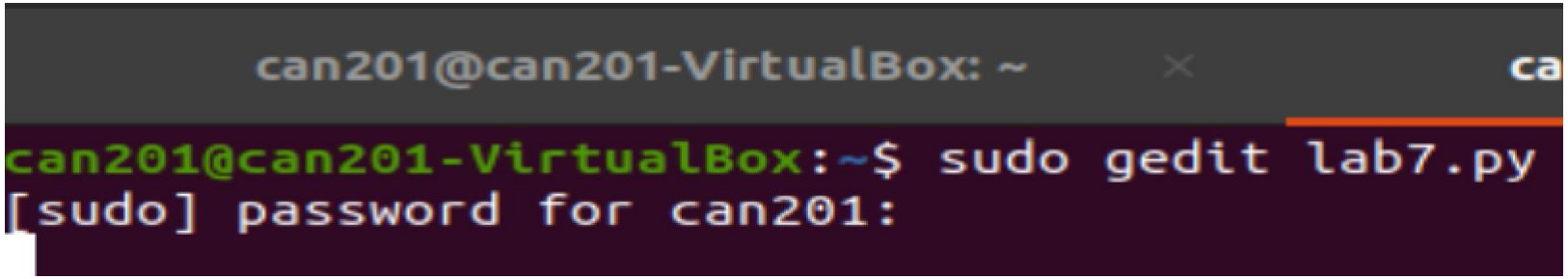


Build Custom Network Topology in Mininet

- Prepare the network topology python file on Linux

  $ sudo gedit lab7.py

  $ sudo vim lab7.py



Build Custom Network Topology in Mininet

- Import Mininet Python library

```
#/usr/bin/python
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.node import Host
from mininet.node import OVSKernelSwitch
from mininet.log import setLogLevel, info
```

```
1 #/usr/bin/python3
2 from mininet.net import Mininet
3 from mininet.cli import CLI
4 from mininet.node import Host
5 from mininet.node import OVSKernelSwitch
6 from mininet.log import setLogLevel, info
```

Build Custom Network Topology in Mininet

- Define a function myTopo() to constitute all the nodes and links.
- The Mininet class returns an empty network object to which we add hosts, switches etc.

def myTopo():

net = Mininet(topo=None, autoSetMacs=True, build=False, ipBase='10.0.1.0/24')

```
1 #/usr/bin/python3
2 from mininet.net import Mininet
3 from mininet.cli import CLI
4 from mininet.node import Host
5 from mininet.node import OVSKernelSwitch
6 from mininet.log import setLogLevel, info
7
8 def myTopo():
9     net = Mininet( topo=None, autoSetMacs=True, build=False,
  ipBase='10.0.1.0/24')
```

Build Custom Network Topology in Mininet

- Add hosts and switch

  h1 = net.addHost('h1', cls=Host, defaultRoute=None)

  h2 = net.addHost('h2', cls=Host, defaultRoute=None)

  h3 = net.addHost('h3', cls=Host, defaultRoute=None)

  s1 = net.addSwitch('s1', cls=OVSKernelSwitch, failMode='standalone')

```
 8 def myTopo():
 9     net = Mininet( topo=None, autoSetMacs=True, build=False,
   ipBase='10.0.1.0/24')
10
11     h1 = net.addHost('h1', cls=Host, defaultRoute=None)
12     h2 = net.addHost('h2', cls=Host, defaultRoute=None)
13     h3 = net.addHost('h3', cls=Host, defaultRoute=None)
14
15     s1 = net.addSwitch('s1', cls=OVSKernelSwitch, failMode='standalone')
```

Build Custom Network Topology in Mininet

- Add links

  net.addLink(h1, s1)

  net.addLink(h2, s1)

  net.addLink(h3, s1)

```
 8 def myTopo():
 9     net = Mininet( topo=None, autoSetMacs=True, build=False,
   ipBase='10.0.1.0/24')
10
11     h1 = net.addHost('h1', cls=Host, defaultRoute=None)
12     h2 = net.addHost('h2', cls=Host, defaultRoute=None)
13     h3 = net.addHost('h3', cls=Host, defaultRoute=None)
14
15     s1 = net.addSwitch('s1', cls=OVSKernelSwitch, failMode='standalone')
16
17     #add links
18     net.addLink(h1, s1)
19     net.addLink(h2, s1)
20     net.addLink(h3, s1)
```

Build Custom Network Topology in Mininet

- Assign IP addresses to interfaces of hosts

  h1.setIP(intf="h1-eth0",ip='10.0.1.2/24')

  h2.setIP(intf="h2-eth0",ip='10.0.1.3/24')

  h3.setIP(intf="h3-eth0",ip='10.0.1.4/24')

```
17      #add links
18      net.addLink(h1, s1)
19      net.addLink(h2, s1)
20      net.addLink(h3, s1)
21
22      #assign IP address to interface
23      h1.setIP(intf='h1-eth0', ip='10.0.1.2/24')
24      h2.setIP(intf='h2-eth0', ip='10.0.1.3/24')
25      h3.setIP(intf='h3-eth0', ip='10.0.1.4/24')
```

Build Custom Network Topology in Mininet

- Network build and start

  net.build()

  net.start()

- CLI mode running

  CLI(net)

  net.stop()

```
27      #Network build an start
28      net.build()
29      net.start()
30
31      #CLI mode running
32      CLI(net)
33      net.stop()
```

Build Custom Network Topology in Mininet

- Add the main function to the python program

  if __name__ == '__main__':

    setLogLevel('info')

    myTopo()

```
27      #Network build an start
28      net.build()
29      net.start()
30
31      #CLI mode running
32      CLI(net)
33      net.stop()
34
35  if __name__ == '__main__':
36      setLogLevel('info')
37      myTopo()
```

Build Custom Network Topology in Mininet

- Save the python file and quit gedit
- Run the network topology python file

  $sudo python3 lab7.py

```
can201@can201-VirtualBox:~$ sudo python3 lab7.py
*** Configuring hosts
h1 h2 h3
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Demo

https://box.xjtlu.edu.cn/f/3df2c732618842f1bf13/

Exercise:

1) Modify the existing code by adding hosts h4, h5, and h6 to it. Run the code and explore the following CLI commands: help, nodes, net, dump, h5 config, pingall xterm h3, exit.

2) Add a second switch (s2) and connect h4-h6 to it. Link s1 and s2 together. Run pingall again. What do you observe?

NB:

After you complete these two tasks, show your codes to the T.A and get the attendance sign-in password.