Mininet: First steps

9/05/2017

Configure Mininet VM

- Download VirtualBox from:

  https://www.virtualbox.org/

- Download and install the mininet VM from:

  http://mininet.org/download/

Configure VM: Linux Users

- Change network settings by enabling «bridge»
- Start the mininet VM
- From Host terminal(Ubuntu) launch:
  - `ssh –Y mininet@<address_of_VM>`
- Password is mininet

MORE INFO at http://mininet.org

Configure VM: all users

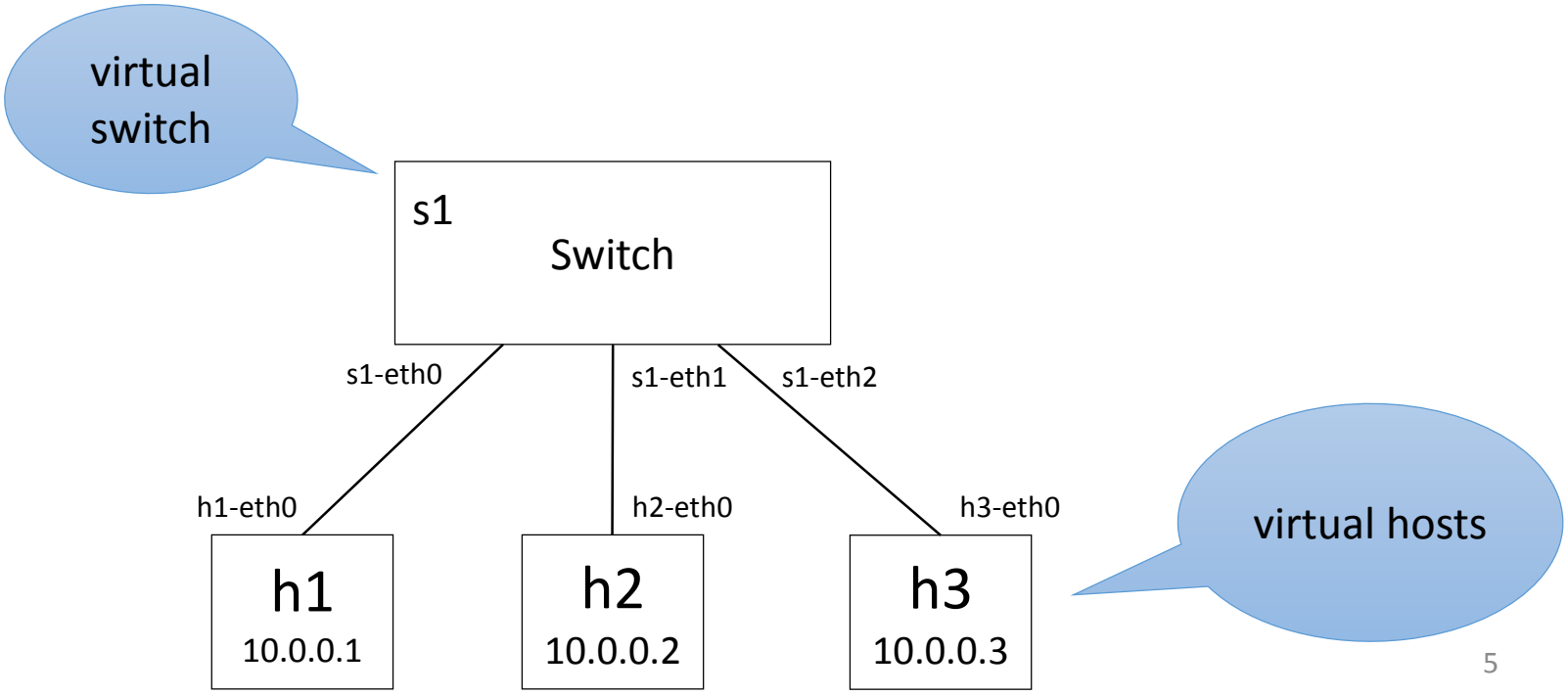
Run the following commands inside the VM to configure the GUI

- `sudo apt-get update`
- `sudo apt-get install xinit lxde`
- `startx`
- `sudo apt-get install virtualbox-guest-dkms`

MORE INFO at http://mininet.org

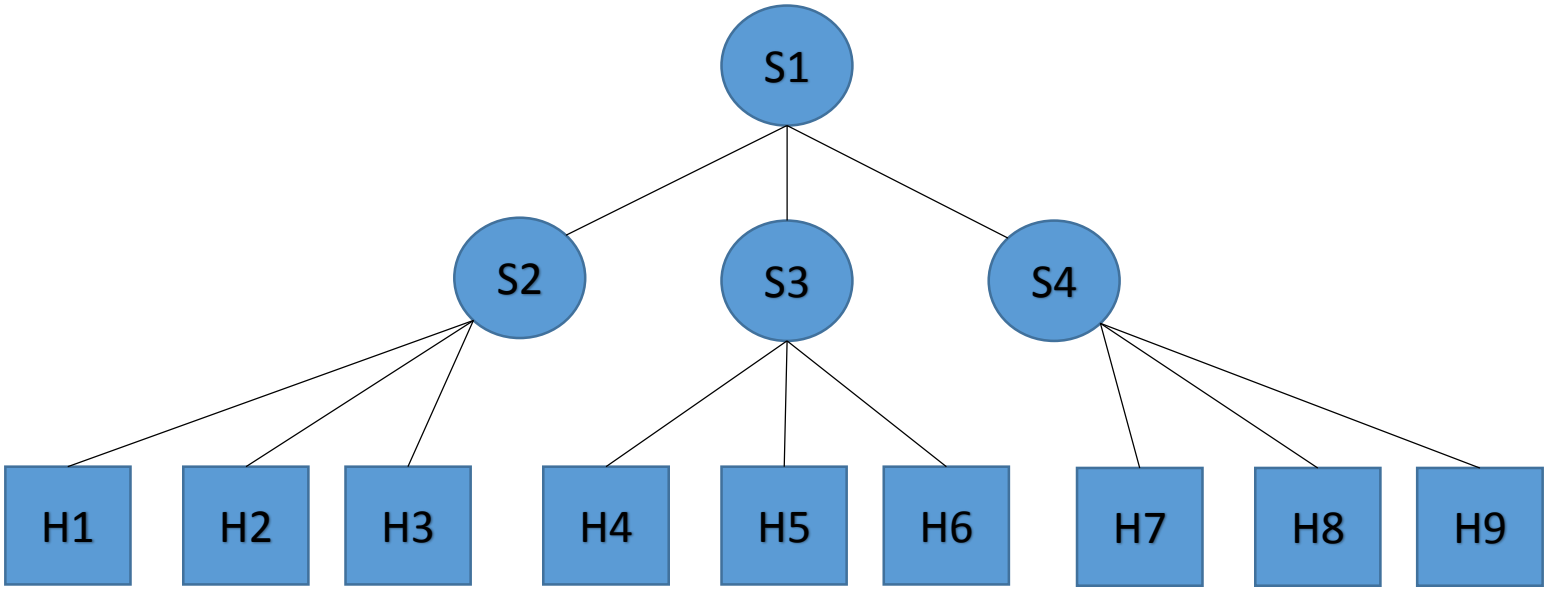
Setup 1: Mininet-based Single Switch

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



First sample commands

- `sudo mn --topo tree,depth=2,fanout=3 --test pingall`
- `sudo mn --topo tree,depth=2,fanout=3 --link tc,bw=5,delay=40ms`



First sample commands

- `sudo mn -h`
- `sudo mn --topo single,8 --test pingall`
- `sudo mn --topo single,8 --test iperf`
- `sudo mn --topo linear,8 --test pingall`
- `sudo mn -c`

Custom Topologies

```
from mininet.topo import Topo

class MyTopo( Topo ):

        def __init__( self ):

                # Initialize topology
                Topo.__init__( self )

                # Add hosts and switches
                leftHost = self.addHost( 'h1' )
                rightHost = self.addHost( 'h2' )
                leftSwitch = self.addSwitch( 's3' )
                rightSwitch = self.addSwitch( 's4' )

                # Add links
                self.addLink( leftHost, leftSwitch )
                self.addLink( leftSwitch, rightSwitch )
                self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Custom Topologies

```
switch = self.addSwitch('s1')

# Python's range(N) generates 0..N-1
for h in range(n):
        host = self.addHost('h%s' % (h + 1))
        self.addLink(host, switch)
```

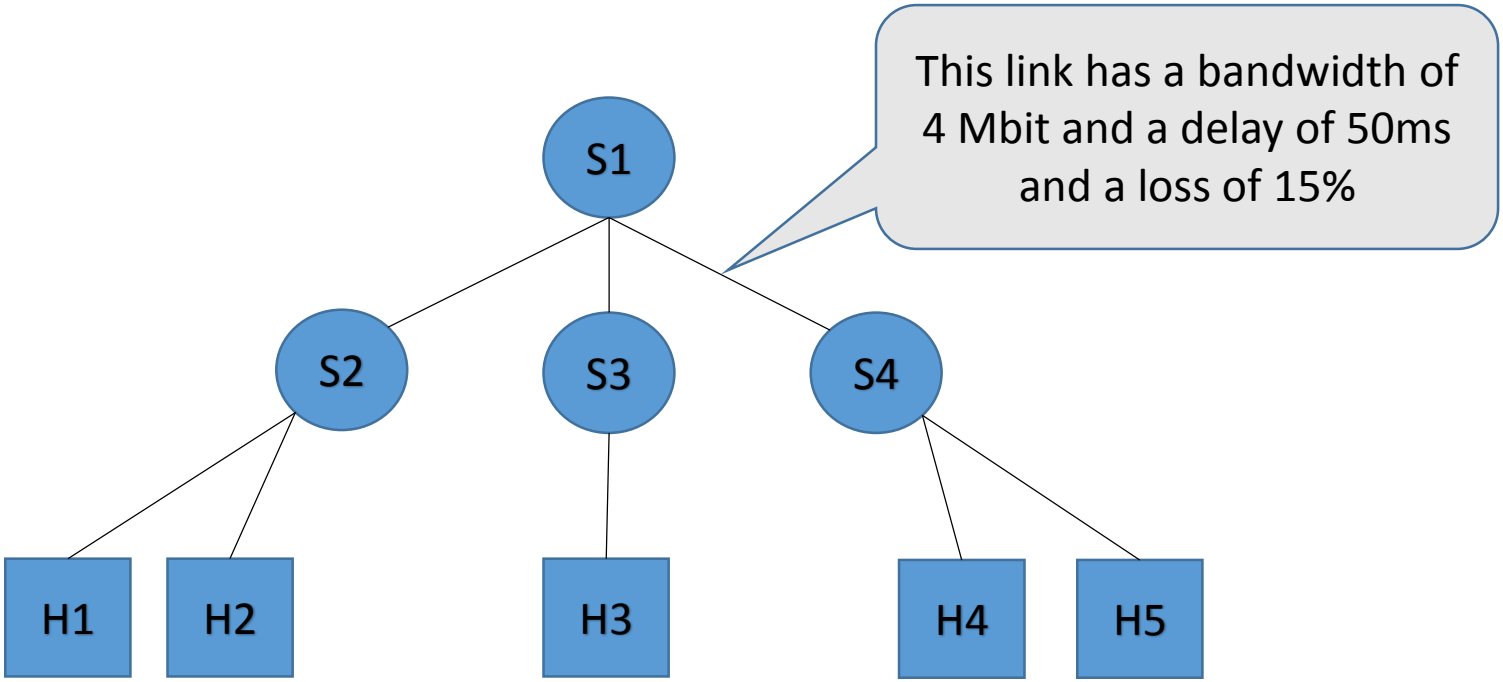
```
# Each host gets 50%/n of system CPU
host = self.addHost('h%s' % (h + 1), cpu=.5/n)

# 10 Mbps, 5ms delay, 10% loss, 1000 packet queue
self.addLink(host, switch, bw=10, delay='5ms',
                           loss=10, max_queue_size=1000, use_htb=True)
```

```
sudo mn --custom ~/mininet/custom/topo-2sw-2host.py
--topo mytopo –-link tc --test pingall
```

Exercise 1

- Build the following topology, execute a ping between all the hosts and measure the bandwidth between host 1 and host 4



Control Commands: `ping`

- Used to test the reachability of a host on an IP network
- It also measures the round-trip-time
- Operates by sending ICMP Echo Request packets to the target host and waiting for an ICMP Echo Reply

Control Commands: `ping`

```
lsd@sampei:~$ ping www.google.it
PING www.google.it (216.58.205.67) 56(84) bytes of data.
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=1 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=2 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=3 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=4 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=5 ttl=55 time=12.1 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=6 ttl=55 time=12.3 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=7 ttl=55 time=12.1 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=8 ttl=55 time=11.9 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=9 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=10 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=11 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=12 ttl=55 time=12.0 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=13 ttl=55 time=12.1 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=14 ttl=55 time=12.1 ms
64 bytes from mil04s25-in-f3.1e100.net (216.58.205.67): icmp_seq=15 ttl=55 time=12.1 ms
^C
--- www.google.it ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14010ms
rtt min/avg/max/mdev = 11.980/12.093/12.399/0.155 ms
lsd@sampei:~$
```

Control Commands: `traceroute`

- Used for displaying the route (path) and measuring transit delays of packets across an IP network
- Works incrementing the TTL field of the IP packet
- Note: sometimes packets can follow different paths and the output could be misleading…

Control Commands: `traceroute`

```
lsd@sampei:~$ sudo traceroute -T www.repubblica.it
traceroute to www.repubblica.it (23.12.106.210), 30 hops max, 60 byte packets
 1  salaria-gw.di.uniroma1.it (151.100.17.1)  0.839 ms  1.229 ms  1.684 ms
 2  151.100.254.249 (151.100.254.249)  0.837 ms  0.836 ms  0.836 ms
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  a23-12-106-210.deploy.static.akamaitechnologies.com (23.12.106.210)  1.562 ms  1.541 ms  1.553 ms
lsd@sampei:~$
```

Control Commands: `iperf`

- Tool used to measure the bandwidth and the quality of a network ‘‘link’’
- The network ‘‘link’’ is delimited by two hosts running `iperf`
- `iperf` uses both TCP or UPD
  - TCP is mainly used to measure the bandwidth
  - UDP is mainly used to measure the packet loss

Control Commands: `iperf`

Example: TCP

- Sul server: `iperf –s`
- Sul client: `iperf –c <ip_server>`

Example: UDP

- Sul server: `iperf -u –s`
- Sul client: `iperf –c <ip_server> -u`