The Mininet network emulator

Mininet

- Network emulator widely used in academy and industry
- A lot of videos already available in Internet
  - https://github.com/mininet/mininet/wiki/Videos
  - Presentation by one of the Mininet creators http://www.youtube.com/watch?v=UWImN9IDpZQ

Mininet built-in topologies

- There are 3 built-in topologies in Mininet: single, linear, tree
- The “--topo” parameter allows to choose one of the built-in topologies. Ex.
  - $ sudo mn --topo single,4
    - One switch connected to 4 hosts
  - $ sudo mn --topo tree,depth=3,fanout=2
    - Tree of depth 3, with two children per node
  - $ sudo mn --topo linear,3
    - 3 switches, one after the other, and one host per switch

Main Mininet commands

- After the execution of Mininet in interactive mode (e.g. $ sudo mn), you will get the Command Line Interface (CLI)
  - mininet>
- To exit from mininet, just execute "quit" or "exit »
- If for any reason, Mininet did not execute correctly, or something finished with an error status, remove any configuration with the command "mn –c"

Main Mininet commands

- While in the CLI, you can issue the
  - “help” to list all the available CLI commands
  - “node” to list all the network devices deployed by mininet
  - “dump” to display the network information related to each deployed network device
  - “net” to show how the devices are interconnected
- You can open a terminal which belongs to one of the deployed clients. For instance, to open the h1 client terminal, you can execute “xterm h1” from the CLI

Custom Topologies

- If you want to play with a network topology other than the built-in topologies, you need to write a Python script
- Example of script testmininet.py
- # mn --custom /path/to/testmininet.py --topo toptest

```
; #Some headers here
class Test_Topo(Topo):
    def __init__(self):
        "Create P2P topology”

        # Initialize topology
        Topo.__init__(self)

        # Add hosts and switches
        h1 = self.addNode('h1’)
        s1 = self.addSwitch('s1’)

        # Add links
        self.addLink(h1, s1)

topos = {
    'toptest': (lambda: Test_Topo())
}
```

Execution from Python

- If you want to play with a network topology other than the built-in topologies, you need to write a Python script
- Example of script testmininet-2.py
- # python /path/to/testmininet-2.py

```
class Test_Topo(Topo):
        …

topos = {
        'toptest': (lambda: Test_Topo())
}

def topTest():
        topo = Test_Topo()
        net = Mininet(topo=topo, link=TCLink)
        net.start()
        h1 = net.get('h1')
        print h1.cmd('ping -c3 10.0.0.2')
        CLI(net) ; # to land into the CLI

        # clean up everything at the end
        net.stop() ;

if __name__ == '__main__':
        setLogLevel('info')
        topTest()
```

Customizing links

- It’s also possible to add some characteristics to a link. In this part, we are going to focus only on two main parameters, delay and bandwidth
- For instance, to add a 10 ms delay between s1 and h1
  - self.addLink(s1, h1, delay='10ms')
- Or to limit the bandwidth to only 10Mbps
  - self.addLink(s1, h1, bw=10)
- If you customize the links in a topology created in your python file, you must lunch Mininet with the “--link tc” parameter. Eg.
  - # mn --custom /path/to/you/python/file.py --topo toponame --link tc