Lab Report Name: SDN Controllers and Mininet
Lab report No:07
Name:Md.Abdul Mukit
ID:IT-17055
Theory:

Traffic Generator:

What is iPerf?: iPerf is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test it reports the bandwidth, loss, and other parameters.

Mininet: Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native) Because you can easily interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research.

Install iperf:

```
mahade@Virtual-Box:~$ sudo apt-get install iperf
[sudo] password for mahade:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic
  linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  iperf
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 76.5 kB of archives.
After this operation, 213 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 iperf amd64 2.0.13+d
fsg1-1build1 [76.5 kB]
Fetched 76.5 kB in 2s (38.0 kB/s)
Selecting previously unselected package iperf.
(Reading database ... 330579 files and directories currently installed.)
Preparing to unpack .../iperf_2.0.13+dfsg1-1build1_amd64.deb ...
Unpacking iperf (2.0.13+dfsg1-1build1) ...
Setting up iperf (2.0.13+dfsg1-1build1) ...
Processing triggers for man-db (2.9.1-1) ...
```

Install Mininet:

```
mahade@Virtual-Box:~$ sudo apt-get install mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic
  linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  cgroup-tools libcgroup1 libevent-2.1-7 libunbound8 openvswitch-common
  openvswitch-switch python-pkg-resources python3-openvswitch
  python3-sortedcontainers socat
Suggested packages:
  ethtool openvswitch-doc python-setuptools python-sortedcontainers-doc
The following NEW packages will be installed:
  cgroup-tools libcgroup1 libevent-2.1-7 libunbound8 mininet
  openvswitch-common openvswitch-switch python-pkg-resources
  python3-openvswitch python3-sortedcontainers socat
0 upgraded, 11 newly installed, 0 to remove and 31 not upgraded.
```

OVS controller installation:

4. Exercises

Exercise 4.1.1: Open a Linux terminal, and execute the command line *iperf --help*. Provide four configuration options of iperf .

```
mahade@Virtual-Box:~$ iperf --help
Usage: iperf [-s|-c host] [options]
       iperf [-h|--help] [-v|--version]

Client/Server:
  -b, --bandwidth #[kmgKMG | pps]  bandwidth to send at in bits/sec or packets p
er second
  -e, --enhancedreports    use enhanced reporting giving more tcp/udp and traffi
c information
  -f, --format    [kmgKMG]   format to report: Kbits, Mbits, KBytes, MBytes
  -i, --interval  #        seconds between periodic bandwidth reports
  -l, --len       #[kmKM]    length of buffer in bytes to read or write (Default
s: TCP=128K, v4 UDP=1470, v6 UDP=1450)
  -m, --print_mss          print TCP maximum segment size (MTU - TCP/IP header)
  -o, --output    <filename> output the report or error message to this specifie
d file
  -p, --port      #        server port to listen on/connect to
  -u, --udp                use UDP rather than TCP
      --udp-counters-64bit use 64 bit sequence numbers with UDP
  -w, --window    #[KM]    TCP window size (socket buffer size)
  -z, --realtime           request realtime scheduler
  -B, --bind <host>[:<port>][%<dev>] bind to <host>, ip addr (including multicas
t address) and optional port and device
  -C, --compatibility      for use with older versions does not sent extra msgs
```

Exercise 4.1.2: Open two Linux terminals, and configure terminal-1 as client (*iperf –c IPv4_server_address*) and terminal-2 as server (*iperf -s*).

For terminal -2:

```
mahade@Virtual-Box:~$ iperf -c 127.0.0.1
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 4.00 MByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 36114 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  55.2 GBytes  47.4 Gbits/sec
```

For terminal -1:

```
mahade@Virtual-Box:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 36114
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  55.2 GBytes  47.4 Gbits/sec
```

Exercise 4.1.3: Open two Linux terminals, and configure terminal-1 as client and terminal-2 as server for exchanging UDP traffic, which are the command lines? Which are the statistics are provided at the end of transmission?

```
mahade@Virtual-Box:~$ iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 34766
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.007 ms    0/  892 (0%)
```

```
mahade@Virtual-Box:~$ iperf -c 127.0.0.1 -u
------------------------------------------------------------
Client connecting to 127.0.0.1, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 34766 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 892 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.007 ms    0/  892 (0%)
```

Exercise 4.1.4: Open two Linux terminals, and configure terminal-1 as client and terminal-2 as server for exchanging UDP traffic, with:

- Packet length = 1000bytes
- Time = 20 seconds
- Bandwidth = 1Mbps
- Port = 9900

Which are the command lines?

The command lines are:

For terminal 1:

Iperf –c 127.0.0.1 –u –l 1000 –t 20 –b 1 –p 9900

```
mahade@Virtual-Box:~$ iperf -c 127.0.0.1 -u -l 1000 -t 20 -b 1 -p 9900
WARNING: delay too large, reducing from 8000.0 to 1.0 seconds.
------------------------------------------------------------
Client connecting to 127.0.0.1, UDP port 9900
Sending 1000 byte datagrams, IPG target: 8000000000.00 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 47011 connected with 127.0.0.1 port 9900
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-21.0 sec  20.5 KBytes  8.00 Kbits/sec
[  3] Sent 21 datagrams
[  3] Server Report:
[  3]  0.0-21.0 sec  20.5 KBytes  8.00 Kbits/sec  62.505 ms    0/   21 (0%)
```

For terminal 2:

Iperf –s –u –p 9900

```
mahade@Virtual-Box:~$ iperf -s -u -p 9900
------------------------------------------------------------
Server listening on UDP port 9900
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 9900 connected with 127.0.0.1 port 47011
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0-21.0 sec  20.5 KBytes  8.00 Kbits/sec  62.505 ms    0/   21 (0%)
```

Using Mininet

Exercise 4.2.1: Open two Linux terminals, and execute the command line *ifconfig* in terminal-1. How many interfaces are present?
In terminal-2, execute the command line *sudo mn*, which is the output?
In terminal-1 execute the command line *ifconfig*. How many real and virtual interfaces are present now?

```
mahade@Virtual-Box:~$ ifconfig
eno1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether c8:d3:ff:e5:7e:6f  txqueuelen 1000  (Ethernet)
        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

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 1996457  bytes 59325279184 (59.3 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1996457  bytes 59325279184 (59.3 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.115  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f933:70:571a:f357  prefixlen 64  scopeid 0x20<link>
        ether 30:e3:7a:b1:ed:24  txqueuelen 1000  (Ethernet)
        RX packets 156264  bytes 141737221 (141.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 131709  bytes 70895978 (70.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
mahade@Virtual-Box:~$ sudo mn
[sudo] password for mahade:
*** 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>
```

Exercise 4.2.2: Interacting with mininet; in terminal-2, display the following command lines and explain what it does:

mininet> help

```
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> nodes

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

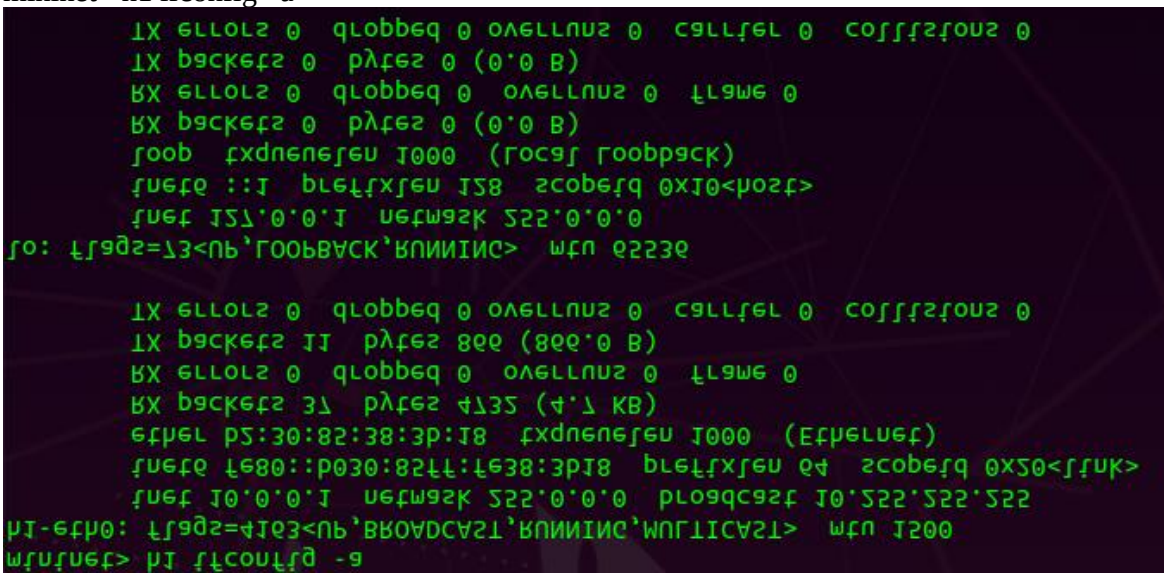
mininet> net

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

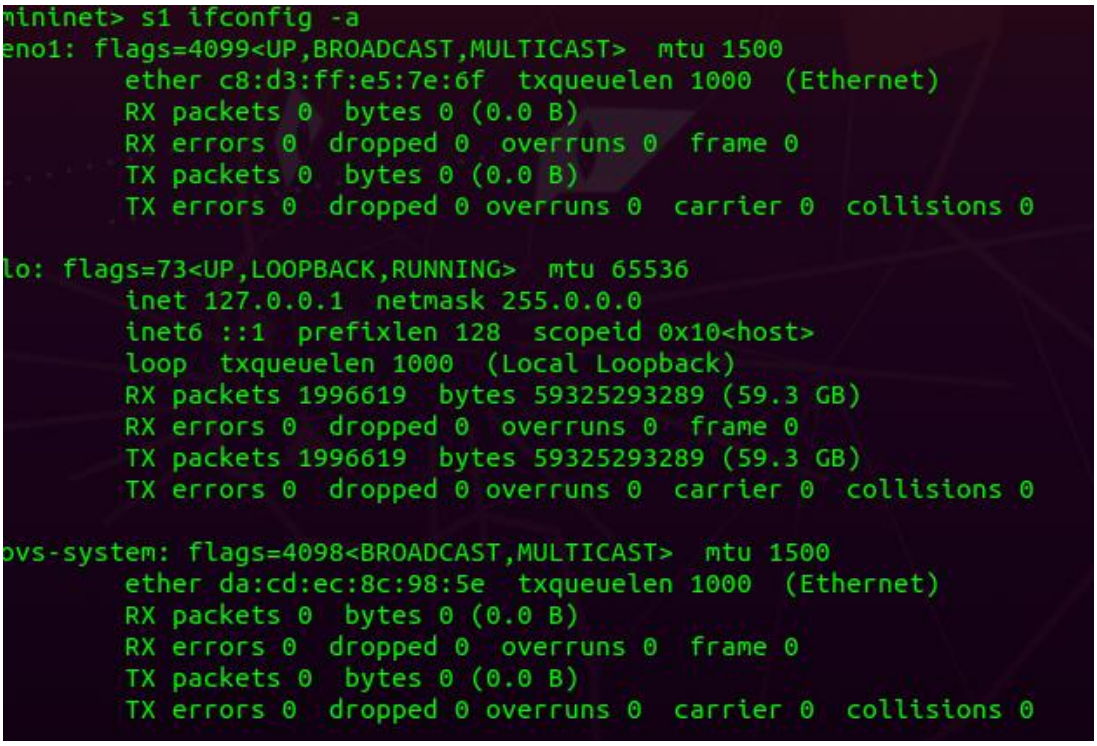
mininet> dump

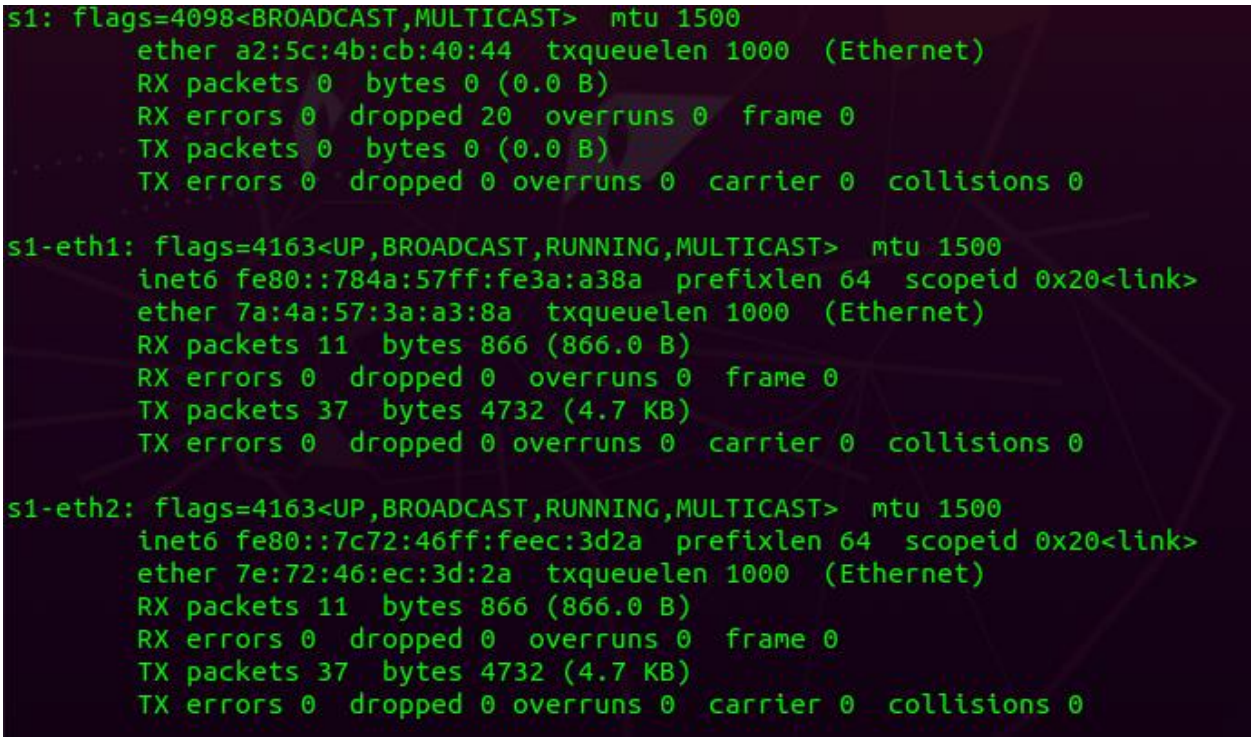
```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=18439>
<Host h2: h2-eth0:10.0.0.2 pid=18441>
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=18446>
```

mininet> h1 ifconfig –a



mininet> s1 ifconfig –a





mininet> h1 ping -c 5 h2

```
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.372 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.122 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.101 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4097ms
rtt min/avg/max/mdev = 0.096/0.158/0.372/0.107 ms
```

Exercise 4.2.3: In terminal-2, display the following command line: *sudo mn --link tc,bw=10,delay=500ms*

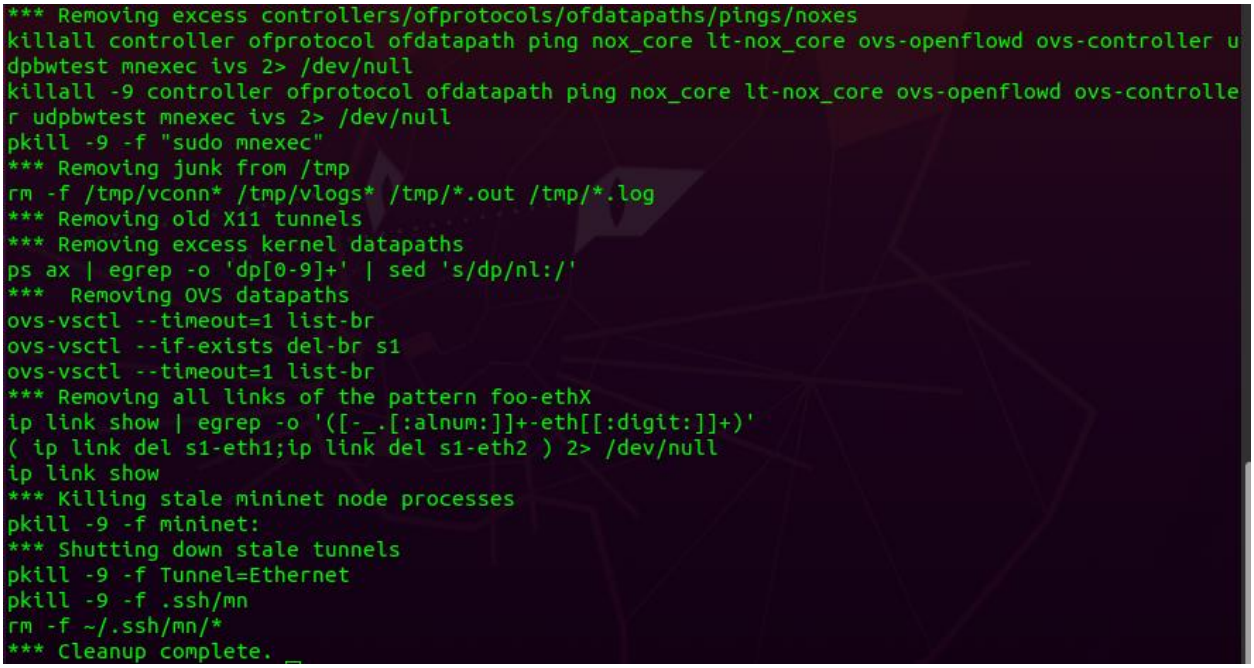
- mininet> h1 ping -c 5 h2, What happen with the link?
- mininet> h1 iperf –s –u &
- mininet> h2 iperf –c IPv4_h1 -u, Is there any packet loss?

```
mahade@Virtual-Box:~$ sudo mn --link tc,bw=10,delay=500ms
[sudo] password for mahade:
*** 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)
Caught exception. Cleaning up...

Exception: Error creating interface pair (h1-eth0,s1-eth1): RTNETLINK answers: File exist
s

*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
```



Conclusion:

Mininet is a *network emulator* which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.

Software defining networking:

1.A ***traffic generator*** is used to put traffic onto a network for other machines to consume. Logically, a traffic generator has a physical layer address (and usually higher-level address), because it is supposed to look like a machine on the network to the target machines receiving the traffic. In terms of implementation, they are attached to the network using the same interfaces as other machines, because they are creating new packets that would not otherwise exist.

2.Transmission Control Protocol (TCP) is a connection-oriented protocol that computers use to communicate over the internet. It is one of the main protocols in TCP/IP networks. TCP provides error-checking and guarantees delivery of data and that packets will be delivered in the order they were sent.

*User Datagram Protocol (UDP) is a connectionless protocol that works just like TCP but assumes that error-checking and recovery services are not required. Instead, UDP continuously sends datagrams to the recipient whether they

receive them or not.

**TCP is best suited to be used for applications that require high reliability where timing is less of a concern.

- World Wide Web (HTTP, HTTPS)
- Secure Shell (SSH)
- File Transfer Protocol (FTP)
- Email (SMTP, IMAP/POP)

UDP is best suited for applications that require speed and efficiency.

- VPN tunneling
- Streaming videos
- Online games
- Live broadcasts
- Domain Name System (DNS)
- Voice over Internet Protocol (VoIP)
  Trivial File Transfer Protocol (TFTP)

iv.The advantages of PLC(programmable controller) are as follows:

1. ***Flexible in Nature:*** One model of PLC can be used for different operations as per requirement.
2. ***Easy to install and trouble shooting:*** In hard wired relay based systems, installation time is more as compared to the PLC based control panels.
3. ***Availability of Large contacts:*** PLC programming tools contain internal large number of contacts that can be used for any change induced in different applications.
4. ***Cost effective:*** Advanced technology and large production of PLC makes it cheaper than the other controller or relay based systems.
5. ***Simulation feature:*** PLC programming software comes with the simulation features by default.
6. ***Simple programming methods:*** PLC is provided with simple programming methods to program the PLC like Ladder or Boolean type of programming.

7. **Ease of maintenance:** As compared with the control systems like relay based or micro-controller based systems, maintenance cost of PLC is low.
8. **Documentation:** The programmer can program and print easily the programs of PLC for future use.

•