



Mawlana Bhashani Science And Technology University

Lab-Report

Lab Report No:

Lab Report Name: SDN Controllers and Mininet

Group member ID: IT-18019 and IT-18037

Date of Performance:

Date of Submission:

Submitted by

Name: Md.Shamim

ID: IT-18019

3rd Year 2nd Semester

Session: 2017-2018

Dept. of ICT, MBSTU

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

1. Objectives:

The objective of the lab 4 is to:

Install and use traffic generators as powerful tools for testing network performance.

Install and configure SDN Controller

Install and understand how the mininet simulator works

Implement and run basic examples for understanding the role of the controller and how it interact with mininet

2. Theory:

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.

iPerf features

TCP and SCTP

- o Measure bandwidth
- o Report MSS/MTU size and observed read sizes.
- o Support for TCP window size via socket buffers.

UDP

- o Client can create UDP streams of specified bandwidth.
- o Measure packet loss
- o Measure delay jitter
- o Multicast capable

Cross-platform: Windows, Linux, Android, MacOS X, FreeBSD, OpenBSD

Cross-platform: Windows, Linux, Android, MacOS X, FreeBSD, OpenBSD,

NetBSD, VxWorks, Solaris,...

Client and server can have multiple simultaneous connections (-P option).

Server handles multiple connections, rather than quitting after a single test.

Can run for specified time (-t option), rather than a set amount of data to transfer (-n or -k option).

Print periodic, intermediate bandwidth, jitter, and loss reports at specified intervals (-i option).

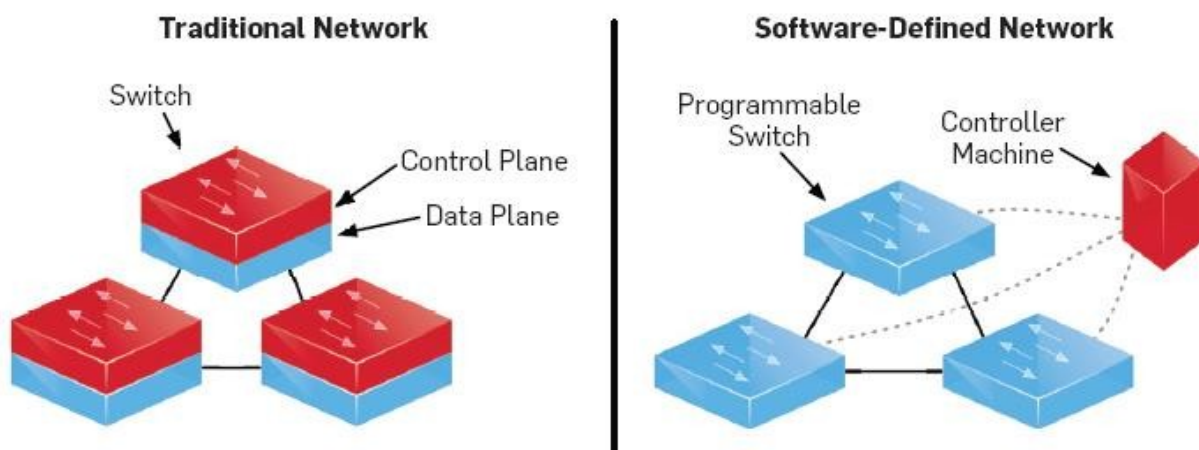
Run the server as a daemon (-D option)

Use representative streams to test out how link layer compression affects your achievable bandwidth (-F option).

2.2. Software Defined Networking:

Software-defined networking was pioneered between 2008 and 2011 by work done at Stanford University and the Nicira Company (now part of VMware). The basic premise behind SDN is

that by separating control of network functions from hardware devices, administrators acquire more power to route and direct traffic in response to changing requirements.



Software-Defined vs. Traditional Networking: The key difference between traditional and software-defined networking is how SDNs handle data packets. In a traditional network, the way a switch handles an incoming data packet is written into its firmware. Most switches —

particularly those used in commercial data centers rather than enterprise environments —

respond to and route all packets the same way. SDN provides admins with granular control over the way switches handle data, giving them the ability to automatically prioritize or block certain types of packets. This, in turn, allows for greater efficiency without the need to invest in expensive, application-specific network switches.

Benefits of Software-Defined Networking: There are several benefits to the more advanced level of control afforded by implementing SND in a multi-tenant network environment:

- 📖 Automation: SND allows for automation of complex operational tasks that make networks faster, more efficient and easier to manage.

- 📖 Increased uptime: SDN has proven effective in reducing deployment and configuration errors that can lead to service disruptions.

- 📖 Less drain on resources: SDN gives administrators control over how their routers and switches operate from a single, virtual workflow. This frees up key staff to focus on more important tasks.

- 📖 Better visibility: With SDN, system administrator's gain improved visibility into overall network function, allowing them to allocate resources more effectively.

- 📖 Cost savings: SND can lead to significant overall costs savings. It also reduces the amount of spending required on infrastructure by allowing data centers to get the most use of their existing devices.

2.2.1. Controller:

OVS-testcontroller is a simple OpenFlow controller that manages any number of switches over the OpenFlow protocol, causing them to function a

Ryu is a component-based software defined networking framework. Ryu provides software components with well-defined API that make it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, Ryu supports fully 1.0, 1.2, 1.3, 1.4, 1.5 and Nicira Extensions. All of the code is freely available under the Apache 2.0 license.

2.2.2. 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. Mininet is also a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems.

3. Methodology

TIP: For getting extra space in your USB-please use the following tips:

- Empty the trash
- Delete the Android related stuff
- Delete the extras for other courses
- Delete the already installed package sources

Install iperf

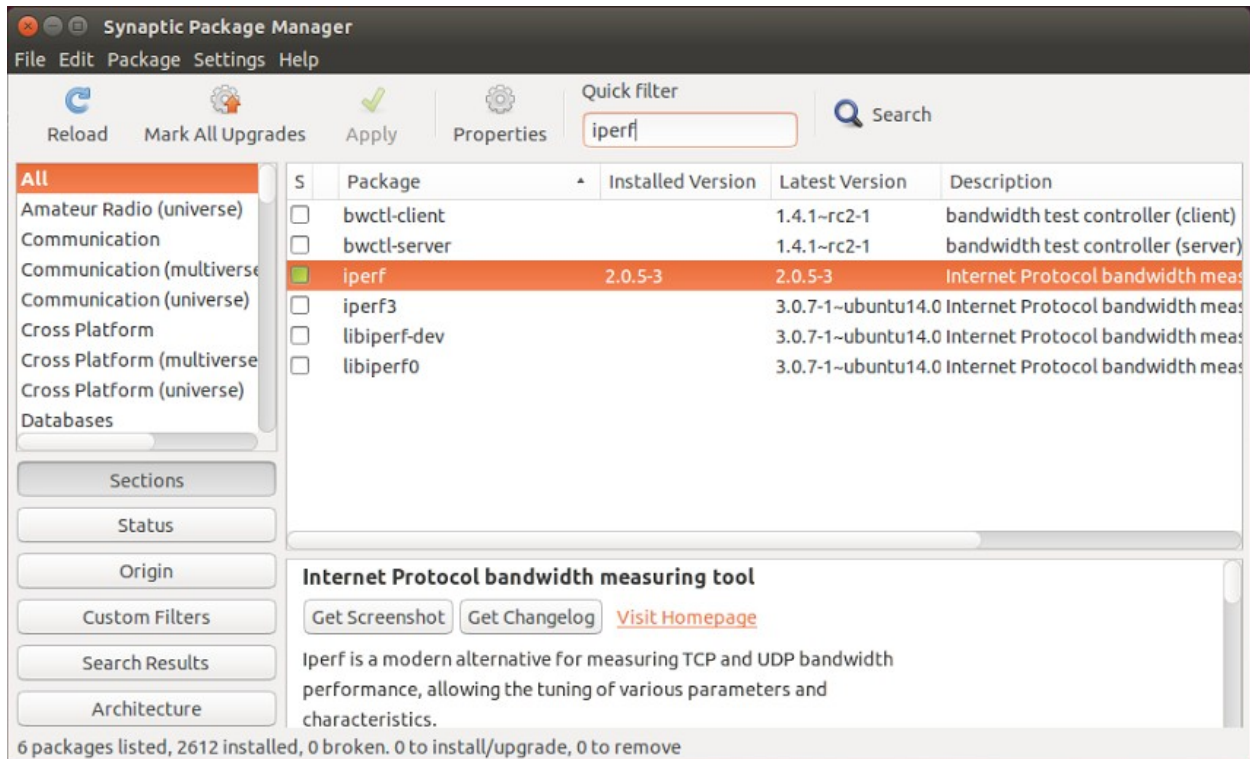
1. Open the Synaptic Package Manager (Navigator ->System->Synaptic Package Manager)

2. Setup the proxy:

- o Click on settings-> Preference -> Network
- o Click on manual proxy configuration
- o HTTP and FTP Proxy: proxy.rmit.edu.au Port: 8080

3. Search for Quick filter `iperf`

4. Click on Mark for installation
5. Then click on Apply and wait until the package is installed



Install mininet

1. In Synaptic Package
2. Search for Quick filter `mininet`
3. Click on Mark for installation
4. Then click on Apply and wait until the package is installed

Install Controller

OVS controller:

1. In Synaptic Package
2. Search for Quick filter `openvswitch-controller`
3. Click on Mark for installation
4. Then click on Apply and wait until the package is installed

RYU controller:

NOTE: Installation valid for Internet access without Proxy

1. Run the command line: `sudo pip install ryu`

2. In some cases the system will not install the required packages by default, command line in

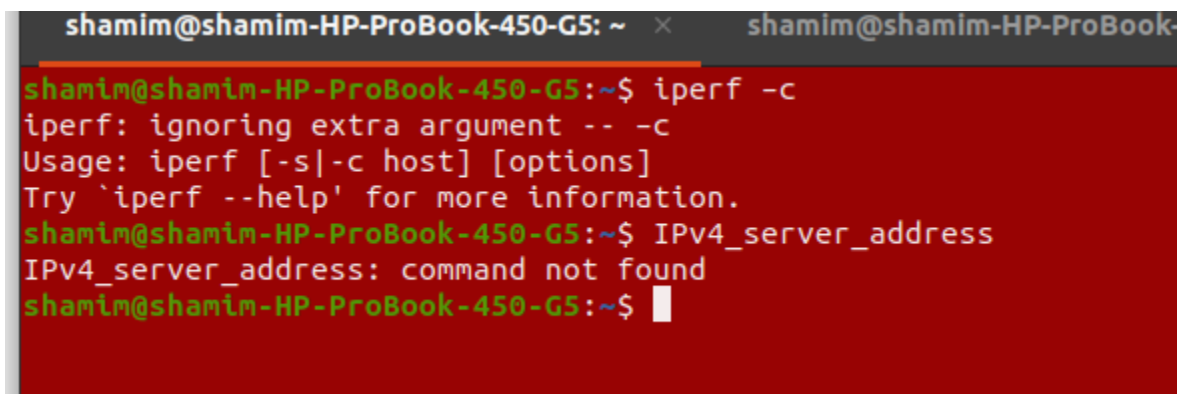
this case is: `sudo pip install ryu eventlet routes webob paramiko lxml netaddr`

`oslo.config msgpack-python greenlet repoze.lru ecdsa six`

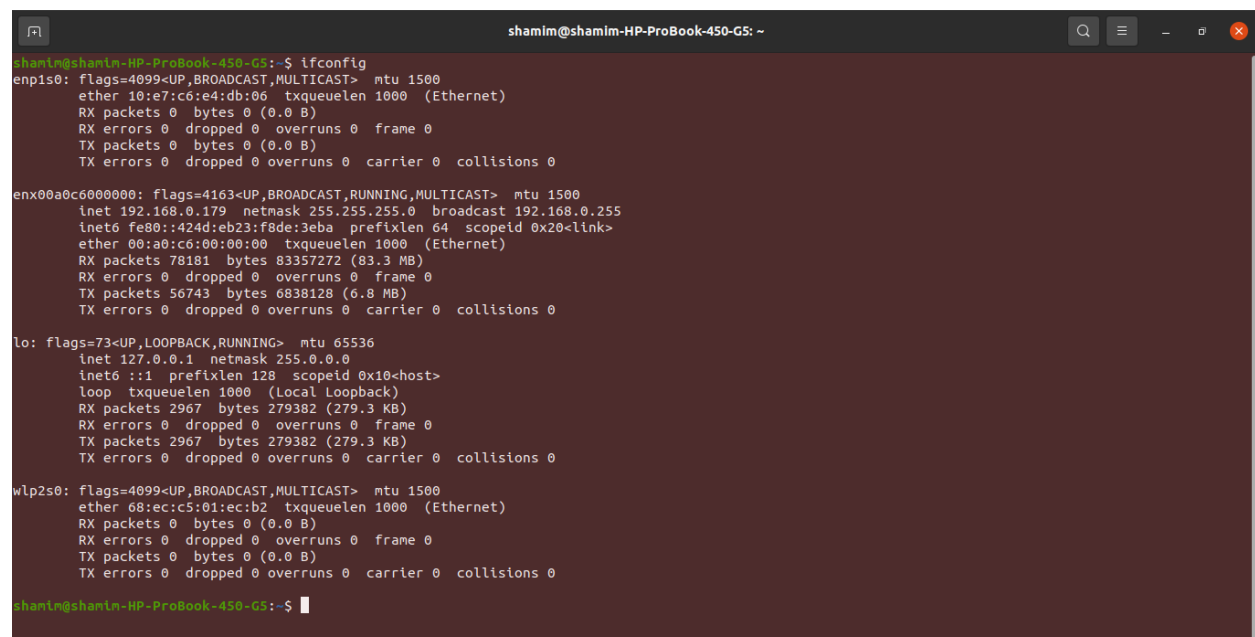
`stevedore rfc3986`

`debtcollector`

4. Exercises:



```
shamim@shamim-HP-ProBook-450-G5: ~  
shamim@shamim-HP-ProBook-450-G5:~$ iperf -c  
iperf: ignoring extra argument -- -c  
Usage: iperf [-s|-c host] [options]  
Try 'iperf --help' for more information.  
shamim@shamim-HP-ProBook-450-G5:~$ IPv4_server_address  
IPv4_server_address: command not found  
shamim@shamim-HP-ProBook-450-G5:~$
```



```
shamim@shamim-HP-ProBook-450-G5:~$ ifconfig  
enp1s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
ether 10:e7:c6:e4:db:06 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  
  
enx00a0c6000000: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 192.168.0.179 netmask 255.255.255.0 broadcast 192.168.0.255  
inet6 fe80::424d:eb23:f8de:3eba prefixlen 64 scopeid 0x20<link>  
ether 00:a0:c6:00:00:00 txqueuelen 1000 (Ethernet)  
RX packets 78181 bytes 83357272 (83.3 MB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 56743 bytes 6838128 (6.8 MB)  
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 2967 bytes 279382 (279.3 KB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 2967 bytes 279382 (279.3 KB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlp2s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
ether 68:ec:c5:01:ec:b2 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  
  
shamim@shamim-HP-ProBook-450-G5:~$
```

```
shamim@shamim-HP-ProBook-450-G5:~$ iperf -s
```

```
-----  
Server listening on TCP port 5001  
TCP window size: 128 KByte (default)  
-----  
█
```

```
shamim@shamim-HP-ProBook-450-G5: ~  
Client specific:  
-c, --client <host> run in client mode, connecting to <host>  
-d, --dualtest Do a bidirectional test simultaneously  
--ipg set the the interpacket gap (milliseconds) for packets within an isochronous frame  
--isochronous <frames-per-second>:<mean>,<stddev> send traffic in bursts (frames - emulate video traffic)  
-n, --num #[kmgKMG] number of bytes to transmit (instead of -t)  
-r, --tradeoff Do a bidirectional test individually  
-t, --time # time in seconds to transmit for (default 10 secs)  
-B, --bind [<ip> | <ip:port>] bind ip (and optional port) from which to source traffic  
-F, --fileinput <name> input the data to be transmitted from a file  
-I, --stdin input the data to be transmitted from stdin  
-L, --listenport # port to receive bidirectional tests back on  
-P, --parallel # number of parallel client threads to run  
-R, --reverse reverse the test (client receives, server sends)  
-T, --ttl # time-to-live, for multicast (default 1)  
-V, --ipv6_domain Set the domain to IPv6 (send packets over IPv6)  
-X, --peer-detect perform server version detection and version exchange  
-Z, --linux-congestion <algo> set TCP congestion control algorithm (Linux only)  
  
Miscellaneous:  
-x, --reportexclude [CDMSV] exclude C(connection) D(data) M(multicast) S(settings) V(server) reports  
-y, --reportstyle C report as a Comma-Separated Values  
-h, --help print this message and quit  
-v, --version print version information and quit  
  
[kmgKMG] Indicates options that support a k,n,g,K,M or G suffix  
Lowercase format characters are 10^3 based and uppercase are 2^n based  
(e.g. 1k = 1000, 1K = 1024, 1m = 1,000,000 and 1M = 1,048,576)  
  
The TCP window size option can be set by the environment variable  
TCP_WINDOW_SIZE. Most other options can be set by an environment variable  
IPERF_<long option name>, such as IPERF_BANDWIDTH.  
  
Source at <http://sourceforge.net/projects/iperf2/>  
Report bugs to <iperf-users@lists.sourceforge.net>  
shamim@shamim-HP-ProBook-450-G5:~$
```



```
shamim@shamim-HP-ProBook-450-G5: ~  
shamim@shamim-HP-ProBook-450-G5:~$ 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 per second  
-e, --enhancedreports use enhanced reporting giving more tcp/udp and traffic information  
-f, --format [kmgKMG] format to report: Kbits, Mbits, KBytes, MBytes  
-i, --interval # seconds between periodic bandwidth reports  
-l, --len #[kmgKMG] length of buffer in bytes to read or write (Defaults: 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 specified 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 multicast address) and optional port and device  
-C, --compatibility for use with older versions does not sent extra msgs  
-M, --mss # set TCP maximum segment size (MTU - 40 bytes)  
-N, --nodelay set TCP no delay, disabling Nagle's Algorithm  
-S, --tos # set the socket's IP_TOS (byte) field  
  
Server specific:  
-s, --server run in server mode  
-t, --time # time in seconds to listen for new connections as well as to receive traffic (default not set)  
--udp-histogram #,# enable UDP latency histogram(s) with bin width and count, e.g. 1,1000=1(ms),1000(bins)  
-B, --bind <ip>[%<dev>] bind to multicast address and optional device  
-H, --ssm-host <ip> set the SSM source, use with -B for (S,G)  
-U, --single_udp run in single threaded UDP mode  
-D, --daemon run the server as a daemon  
-V, --ipv6_domain Enable IPv6 reception by setting the domain and socket to AF_INET6 (Can receive on both IPv4 and IPv6)  
  
Client specific:  
-c, --client <host> run in client mode, connecting to <host>  
-d, --dualtest Do a bidirectional test simultaneously
```

```
shamim@shamim-HP-ProBook-450-G5: ~ x shamim@shamim-HP-ProBook-450-G5: ~  
shamim@shamim-HP-ProBook-450-G5:~$ nc localhost 2399  
Hello Server
```

```
shamim@shamim-HP-ProBook-450-G5: ~ x shamim@shamim-HP-ProBook-450-G5: ~  
shamim@shamim-HP-ProBook-450-G5:~$ nc -l 2399  
Hello Server
```

```
shamim@shamim-HP-ProBook-450-G5: ~ x shamim@shamim-HP-ProBook-450-G5: ~  
shamim@shamim-HP-ProBook-450-G5:~$ nc -u -l 2399
```

```
shamim@shamim-HP-... x shamim@shamim-HP-... x shamim@shamim-HP-... x
shamim@shamim-HP-ProBook-450-G5:~$ nc -u localhost 2399
```

```
shamim@shamim-HP-ProBook-450-G5:~$ netstat | grep 2399
tcp        0      0 localhost:2399      localhost:43118     TIME_WAIT
udp        0      0 localhost:48806     localhost:2399     ESTABLISHED
shamim@shamim-HP-ProBook-450-G5:~$
```

Dear Sir ,i can not install mininet in my linux pc.i try to install this but my system many time fail to run this software .