Name : Md. Sohag Hossain

ID : IT-17060

Lab report no : 08

Lab report name : SDN Controllers

Objectives:

1. How to work SDN Controllers

Theory:

An SDN controller is an application in a software-defined networking architecture that manages flow control for improved network management and application performance. The SDN controller platform typically runs on a server and uses protocols to tell switches where to send packets.

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.

SDN controller (software-defined networking controller):

An SDN controller is an application in a software-defined networking (SDN) architecture that manages flow control for improved network management and application performance. The SDN controller platform typically runs on a server and uses protocols to tell switches where to send packets.

SDN controllers direct traffic according to forwarding policies that a network operator puts in place, thereby minimizing manual configurations for individual network devices. By taking the control plane off of the network hardware and running it instead as software, the centralized controller facilitates automated network management and makes it easier to integrate

and administer business applications. In effect, the SDN controller serves as a sort of operating system (OS) for the network.

SDN controller vendors:

Vendors that offer SDN controllers include the following:

- Big Switch Networks
- Cisco
- Cumulus Networks
- Hewlett Packard Enterprise
- Juniper Networks
- Nuage Networks
- Pica8
- Pluribus Networks
- VMware

Open source SDN controllers

SDN controllers are available in a variety of open source options, including:

- Floodlight
- OpenDaylight
- OpenContrail
- Open Network Operating System

SD-WAN controllers

Traditionally, SDN controllers are used in data center networks. As SDN technology evolved, however, the WAN became a compelling use case, driving the

growth of software-defined WAN (SD-WAN) technology. An SD-WAN controller performs many of the same duties as an SDN controller, following policy configurations to direct WAN traffic over the most efficient route. The SD-WAN market has fewer notable open source options than SDN, as most SD-WAN controllers typically come tied together with the vendor's proprietary SD-WAN platform.

Traffic Generators:

A traffic generators 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

Using traffic Generators:

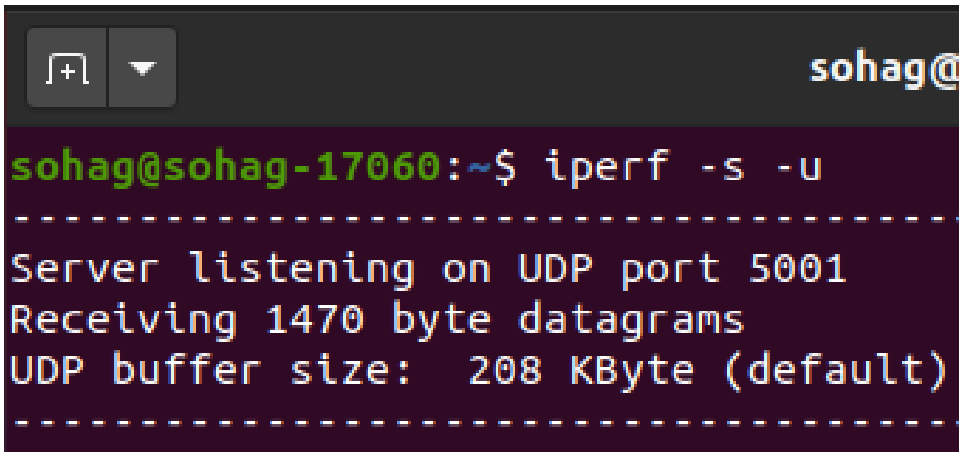
Execute the command line iperf --help.

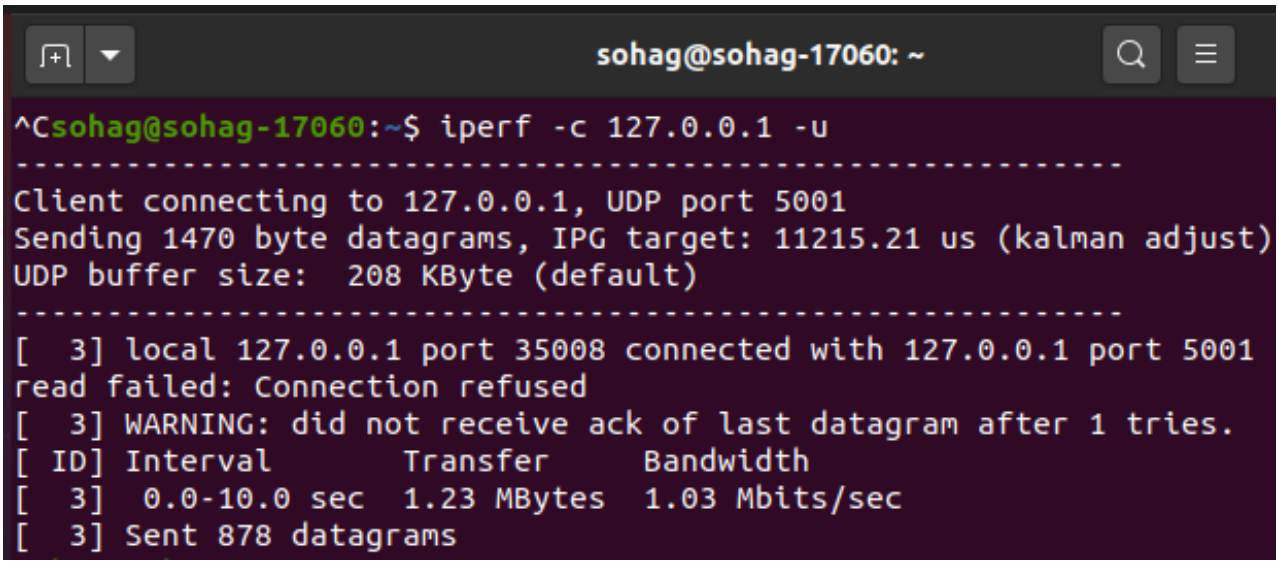
```
sohag@sohag-17060:~$ iperf --help
Usage: iperf [-s|-c host] [options]
       iperf [-h|--help] [-v|--version]

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 traff
ic 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 (Defaul
ts: 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 specifi
ed 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 multica
st 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
```

Configure terminal-1 as client and terminal-2 as server for exchanging UDP traffic, which are the command lines?

First we need to create a server by the following command:
$ iperf -s –u

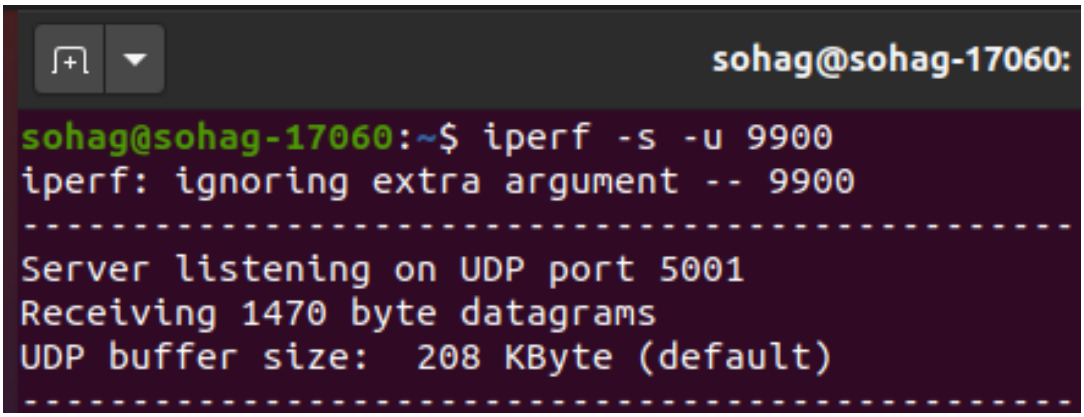
Then we need to create a client by the following command:
$ iperf -c 127.0.0.1 –u



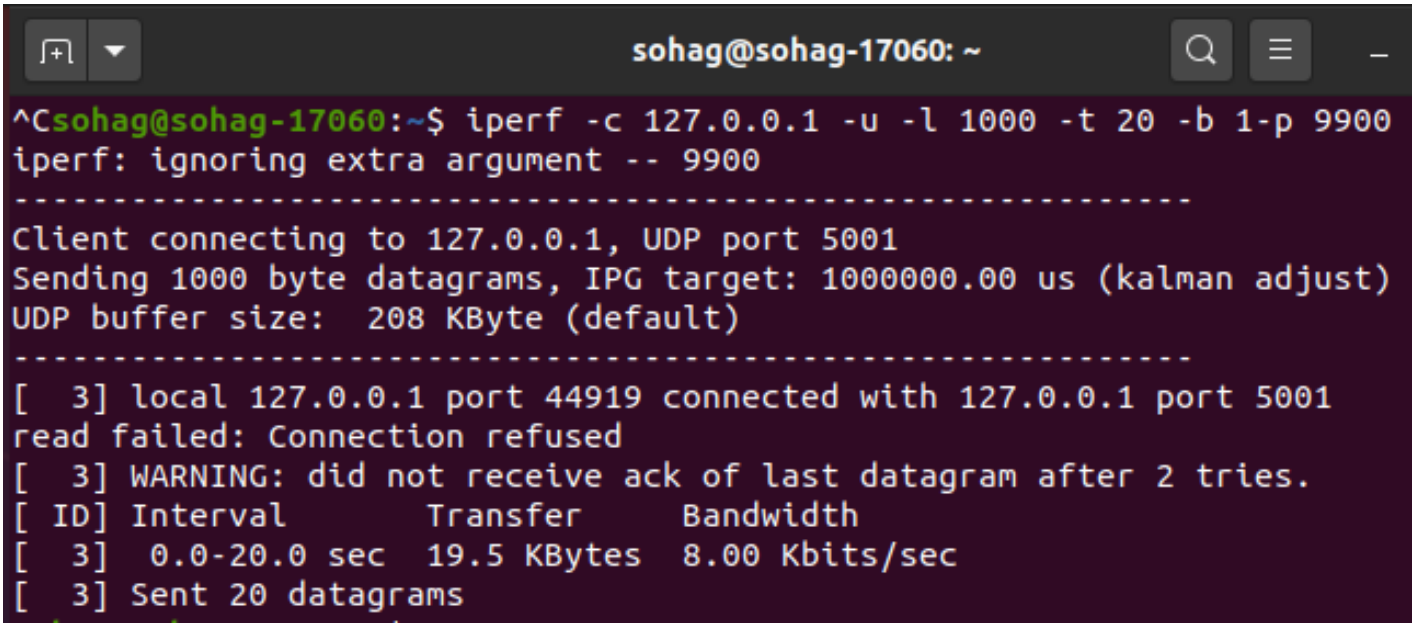


Configure terminal-1 as client and terminal-2 as server for exchanging UDP traffic:

The server command is:
iperf -s -u -p 9900



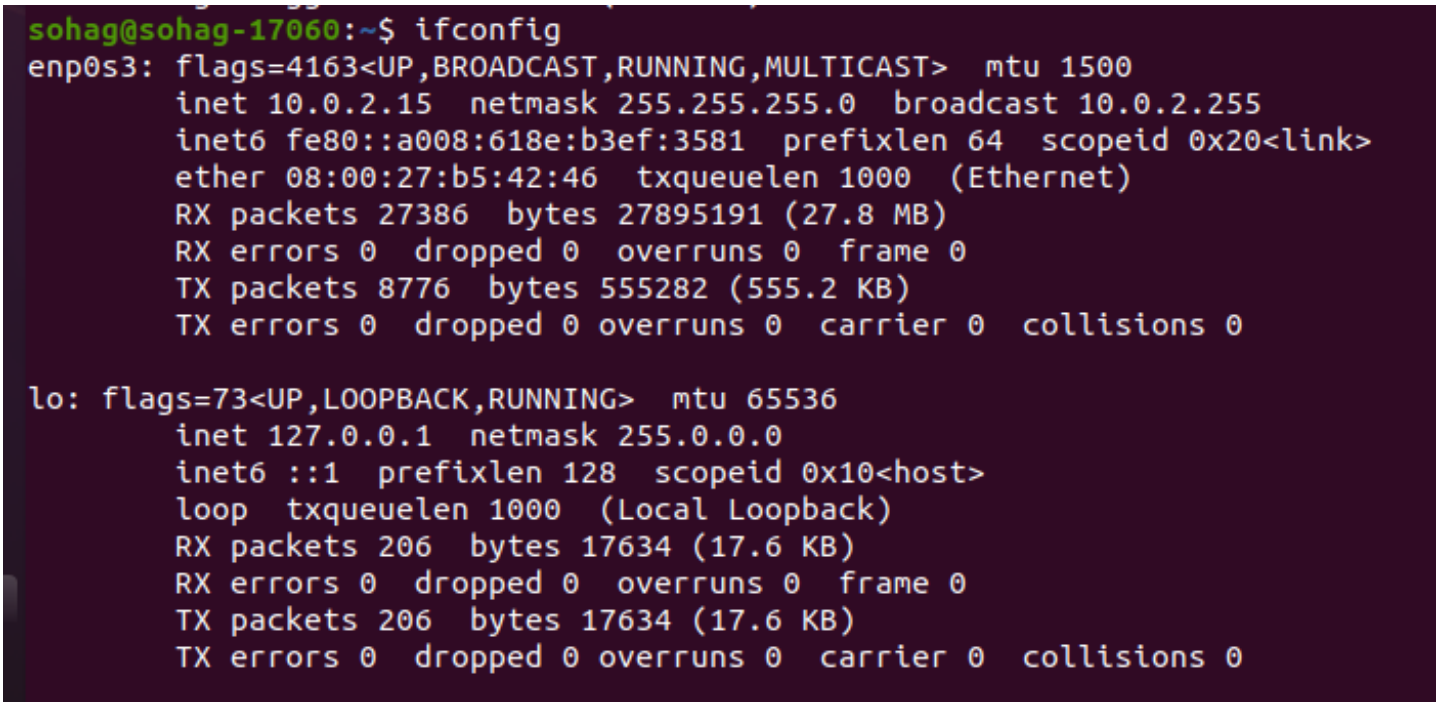
The client command is:
iperf -c 127.0.0.1 -u -l 1000 -t 20 -b 1 -p 9900



Using Mininet:

Execute the command line ifconfig in terminal

terminal-1.



Conclusion: I have paid a lot of efforts to do all the lab task that are given above. Although I give so many error when I run this program. After all, I finally able to run these program .