SAN JOSÉ STATE
UNIVERSITY

EE 283- BROADBAND COMMUNICATION NETWORKS

Spring 2020

Dr. Nader F. Mir

Course Project

Simulation of SDN-Based Cloud Data Center Using Mininet Tool

by

RANJITHA SRINIVASA

014534351

A. Abstract:

The objectives of this project is to learn the working of the MININET, a widely-used computer network emulation (simulation) tool. Understanding the simulation of Software-Defined Network (SDN), virtual switches, and SDN controllers and understanding how a small-scale cloud data center hardware setup operates. Software-Defined Networking not just overcomes the limitations of the traditional network technologies by handling the persistent and immediate changes in cloud data centers but also makes network resource management open to encourage innovation related to it. To further enhance the acceleration of the innovation, easy-to-learn testbeds are utilized, to evaluate and measure the performance of network and capacity of the host simultaneously within a data center. This is often considered as expensive if accomplished in a physical environment. Hence, we need to use a network simulator to implement and analyze this complex network topology. With that in mind, I am using Mininet to simulate a data center cloud system and formulate a performance report on different virtual topologies.

Background

- **Cloud Computing and Data Centers and Network Virtualization**

A cloud computing or a cloud-based system is a network-based computing system in which clients use a shared pool of configurable computing resources. Cloud-based systems provide services from large data centers (DCs).
Cloud computing, on the other hand, is equivalent to "distributed computing over a network" that runs programs or applications on numerous connected host servers at the same time.
Network virtualization is the act of decoupling networking services from network infrastructures. The important role that networking plays in cloud computing calls for an improved, combined control and optimization of networking and computing resources in a cloud environment. The concept of network virtualization can be thought of as a virtual network that operates using software that is built on certain physical network devices.
Software-defined networking (SDN)

- **Software-defined networking (SDN)**

SDN is a networking paradigm by which a central software program known as a "controller" (or SDN controller) deter-mines and controls the overall network behavior, resulting in potential improvement in network performance. The hypervisor in the SDN environment is called Flow Visor. There are various controller software modules, each created for a certain objective such as Open Daylight and Floodlight.

The Open Daylight project is an industry coalition open source project hosted by the Linux Foundation. Open Daylight delivers a common open source framework and platform for SDN.

Floodlight Controller Floodlight is another controller that provides basic features such as routing, topology discovery, security, or even firewall rules. It provides the means to control or direct the network that lies beneath.

Network functions virtualization (NFV) is an alternative and to some extent complementary-to-SDN networking approach that aims to address these problems, especially for building networks with complex applications. NFV virtualizes the entire class of network node functions, from routing to billing, into building blocks that may be connected or chained together to create communication services.

- **Mininet**

Mininet is a network emulator targeted for SDN-based cloud network simulation. It uses simulated hosts, virtual switches, and links on a single Linux kernel. With Mininet, one can use a variety of networking protocols and send packets through what seems like a real Ethernet interface, with any typical given link speed and delay. Packets are processed by an emulated Ethernet switch or router and queueing packets can be analyzed. Since Mininet is an emulator, any real-time program including Web servers, TCP window monitoring programs, or Wireshark can run on it. Software-defined network (SDN) that are developed in Mininet can be transferred to hardware OpenFlow switches for packet forwarding. Servers in Mininet, can create virtual machines (VMs), and thus this emulator can be used in cloud computing. Any Mininet topology can be connected to an SDN controller like Open Daylight or Floodlight and the default port is 6633.

- **Wireshark:**

Wireshark is the world's foremost and widely used network protocol analyzer. It lets you see what is happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.

B. Downloading and running Software :

Downloaded Mininet open source from sdnhub http://sdnhub.org/tutorials/sdn-tutorial-vm/ and installed the tool. Mininet commands are preceded by a Mininet> prompt. To use virtualization in Mininet, tools like VMware was installed.

1. When Mininet runs on VirtualBox, for example, click open a virtual machine and load the Mininet .ova file installed.
2. Power on the virtual machine.
3. Open terminal1 and run commands as:
    - **cd mininetranjitha** # opens the predefined Mininet file
    - **cd custom/** # opens the predefined custom file
    - **ls** # to check the available files
    - **nano** # to create topology
    - press **Ctrl+O** to save file by name as filename.py(exit to terminal)

- **ls** # to check if the new file is created
- **nano filename.py** # to check the code
- open another terminal2 to activate controller and run commands:
    - **cd ryu**
    - **./bin/ryu-manager –verbose ryu/app/simple_switch_13.py**
- now run command in terminal one as **sudo mn –custom filename.py –topo mytopo –controller=remote –switch ovsk,protocols=OpenFlow13** # the command will execute the topology created.
- Run commands as: # to execute appropriate output and capture screenshots
    - Pingall
    - Pingallfull
    - Iperf hx hy
    - hx ping hy
    - Sudo wirehsark &

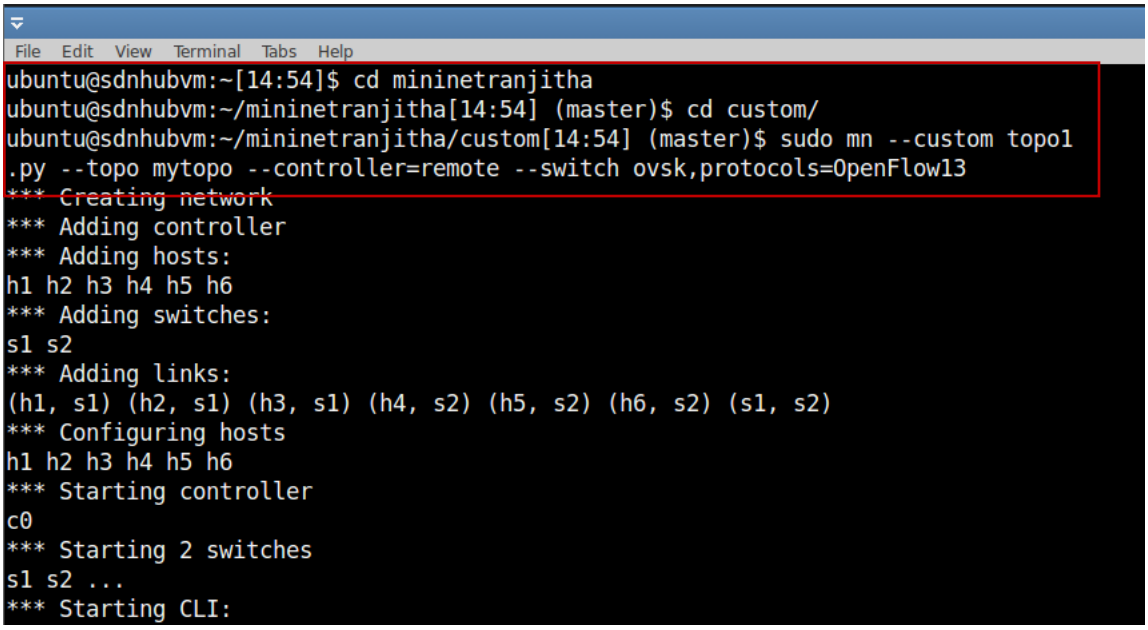


Fig1: Ran the basic commands to run the topo1.py

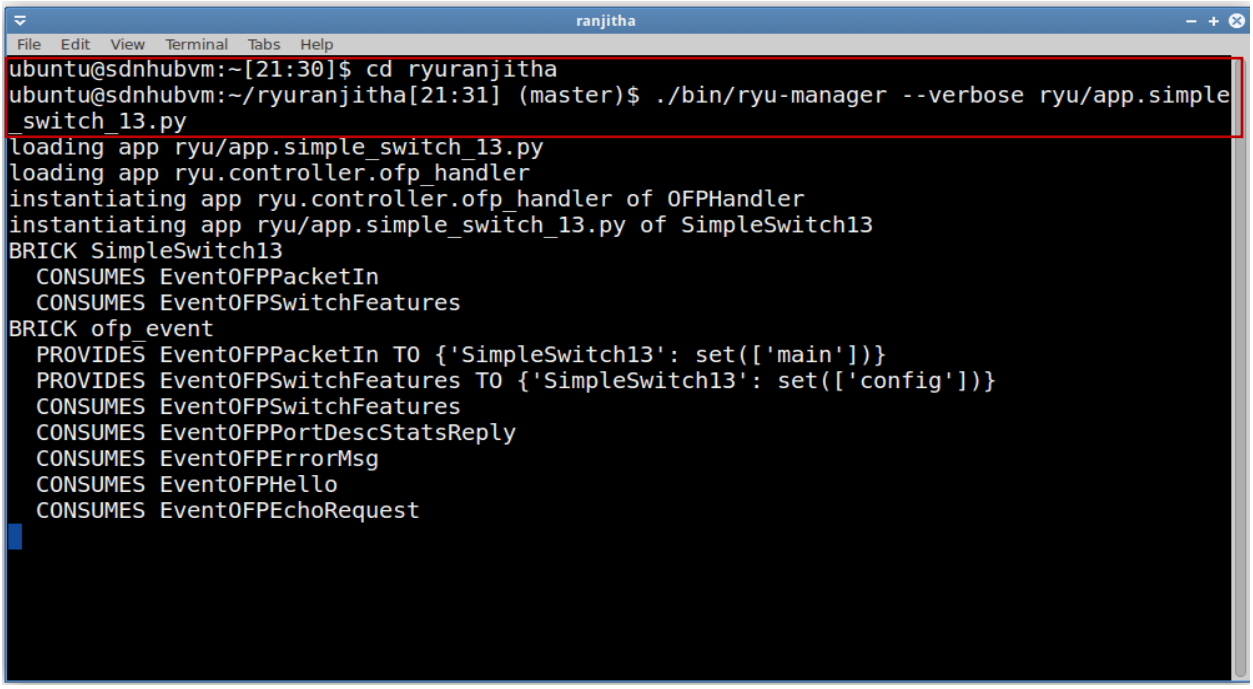


Fig2: Controller commands to execute simulation 1

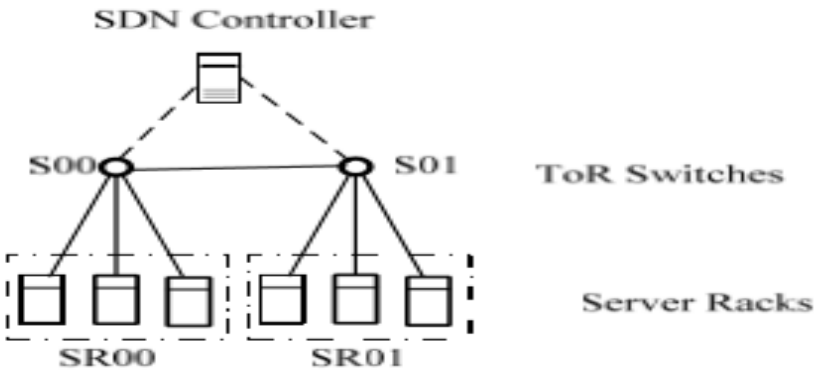
C. Experiment: Simulation of a Small Scale SDN-Based Cloud Data Center:

Simulation 1:

Implement 6 Servers, in 2 Racks, with 2 Switches and an SDN Controller

Implemented data center that includes 6 servers (hosts) connected in 2 server racks forming a small data center network and Ran Mininet and:

[a] Conducted a test using Pingall command and report the average round trip time (RTT). Shown the snapshot of the result for all hosts.



```
  GNU nano 2.2.6

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 'h2' )
        host3 = self.addHost( 'h3' )
        host4 = self.addHost( 'h4' )
        host5 = self.addHost( 'h5' )
        host6 = self.addHost( 'h6' )


        switch00 = self.addSwitch( 's1' )
        switch01 = self.addSwitch( 's2' )

        # Add links
        # Add links
        self.addLink( host1, switch00 )
        self.addLink( host2, switch00 )
        self.addLink( host3, switch00 )
        self.addLink( host4, switch01 )
        self.addLink( host5, switch01 )
        self.addLink( host6, switch01 )
        self.addLink( switch00, switch01 )



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

```
File  Edit  View  Terminal  Tabs  Help
ubuntu@sdnhubvm:~[14:54]$ cd mininetranjitha
ubuntu@sdnhubvm:~/mininetranjitha[14:54] (master)$ cd custom/
ubuntu@sdnhubvm:~/mininetranjitha/custom[14:54] (master)$ sudo mn --custom topo1
.py --topo mytopo --controller=remote --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
```

Fig3: Code for creating and executing topology for simulation 1 and saved as topo1.py

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
```

Fig4: Command **pingall** for simulation 1 of topo1.py

```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.352/0.352/0.352/0.000 ms
 h1->h3: 1/1, rtt min/avg/max/mdev 0.312/0.312/0.312/0.000 ms
 h1->h4: 1/1, rtt min/avg/max/mdev 0.084/0.084/0.084/0.000 ms
 h1->h5: 1/1, rtt min/avg/max/mdev 0.419/0.419/0.419/0.000 ms
 h1->h6: 1/1, rtt min/avg/max/mdev 0.324/0.324/0.324/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 0.046/0.046/0.046/0.000 ms
 h2->h3: 1/1, rtt min/avg/max/mdev 0.185/0.185/0.185/0.000 ms
 h2->h4: 1/1, rtt min/avg/max/mdev 0.213/0.213/0.213/0.000 ms
 h2->h5: 1/1, rtt min/avg/max/mdev 0.313/0.313/0.313/0.000 ms
 h2->h6: 1/1, rtt min/avg/max/mdev 0.218/0.218/0.218/0.000 ms
 h3->h1: 1/1, rtt min/avg/max/mdev 0.052/0.052/0.052/0.000 ms
 h3->h2: 1/1, rtt min/avg/max/mdev 0.104/0.104/0.104/0.000 ms
 h3->h4: 1/1, rtt min/avg/max/mdev 0.348/0.348/0.348/0.000 ms
 h3->h5: 1/1, rtt min/avg/max/mdev 0.202/0.202/0.202/0.000 ms
 h3->h6: 1/1, rtt min/avg/max/mdev 0.204/0.204/0.204/0.000 ms
 h4->h1: 1/1, rtt min/avg/max/mdev 0.084/0.084/0.084/0.000 ms
 h4->h2: 1/1, rtt min/avg/max/mdev 0.114/0.114/0.114/0.000 ms
 h4->h3: 1/1, rtt min/avg/max/mdev 0.067/0.067/0.067/0.000 ms
 h4->h5: 1/1, rtt min/avg/max/mdev 0.320/0.320/0.320/0.000 ms
 h4->h6: 1/1, rtt min/avg/max/mdev 0.322/0.322/0.322/0.000 ms
 h5->h1: 1/1, rtt min/avg/max/mdev 0.133/0.133/0.133/0.000 ms
 h5->h2: 1/1, rtt min/avg/max/mdev 0.062/0.062/0.062/0.000 ms
 h5->h3: 1/1, rtt min/avg/max/mdev 0.060/0.060/0.060/0.000 ms
 h5->h4: 1/1, rtt min/avg/max/mdev 0.054/0.054/0.054/0.000 ms
 h5->h6: 1/1, rtt min/avg/max/mdev 0.563/0.563/0.563/0.000 ms
 h6->h1: 1/1, rtt min/avg/max/mdev 0.082/0.082/0.082/0.000 ms
 h6->h2: 1/1, rtt min/avg/max/mdev 0.100/0.100/0.100/0.000 ms
 h6->h3: 1/1, rtt min/avg/max/mdev 0.082/0.082/0.082/0.000 ms
 h6->h4: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
 h6->h5: 1/1, rtt min/avg/max/mdev 0.079/0.079/0.079/0.000 ms
```

Fig5: Command **pingallfull** representing the average RTT for all the hosts

- Here we can observe that all hosts are communicating with every other hosts as showcased by the pingall command. Each host while communicating with other hosts in pingallfull command will showcase different RTT average. **RTT average differs for connection established between different hosts.**

[b] Comment specifically on an RTT over a connection from a server in S01 rack to a server in S02 rack.

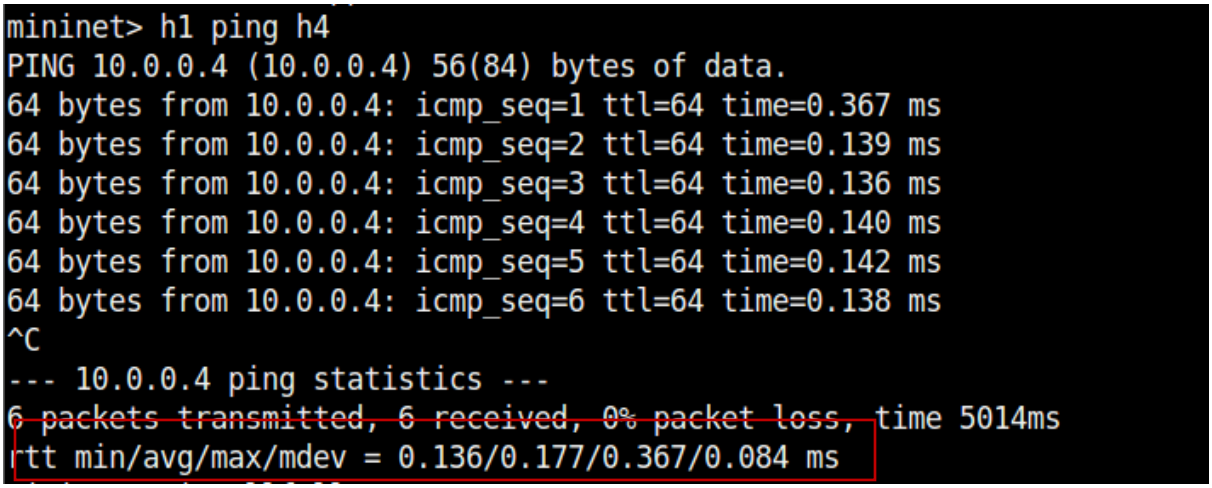


Fig6: Command **h1 ping h4** is showing the connection from h1 of s01 rack to h4 of s02 rack.

- Here we can observe the RTT for a particular connection from h1 to h4.
- We observe that **icmp** messages are sent as a result of successful connection established. 6packets are transmitted and 6 are received with **RTT average of 0.177 secs**. Here we can observe that all hosts are communicating with every other hosts. Each host while communicating with other hosts will showcase different RTT.

[c] Experience iperf a server in S01 rack to a server in S02 rack. Show the snapshot of your result.

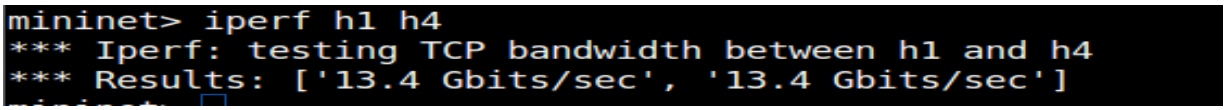


Fig7: Command **iperf h1 h4** showcases the bandwith between h1 and h4 as **13.4 gbps.**

[d] Experience Wireshark on a server in S01 rack to a server in S02 rack. Show the snapshot of your result.

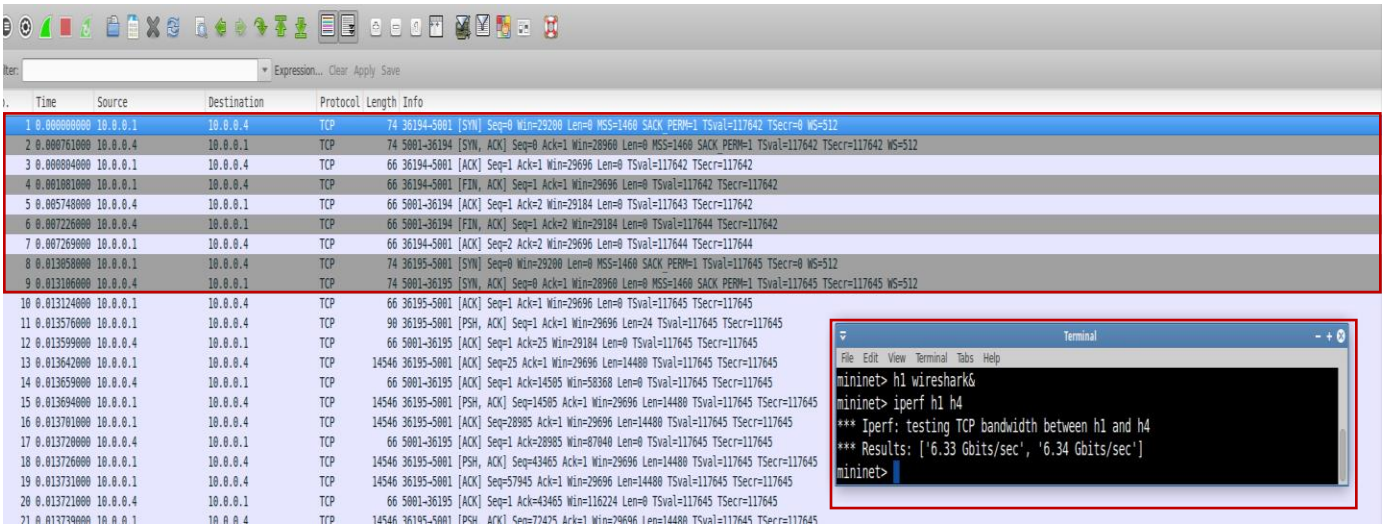
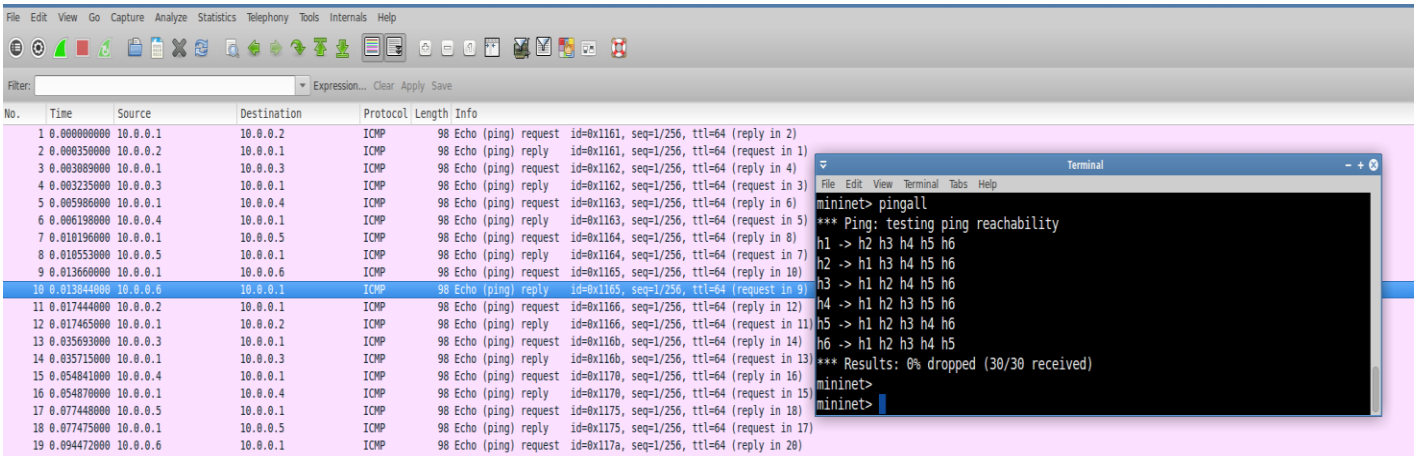
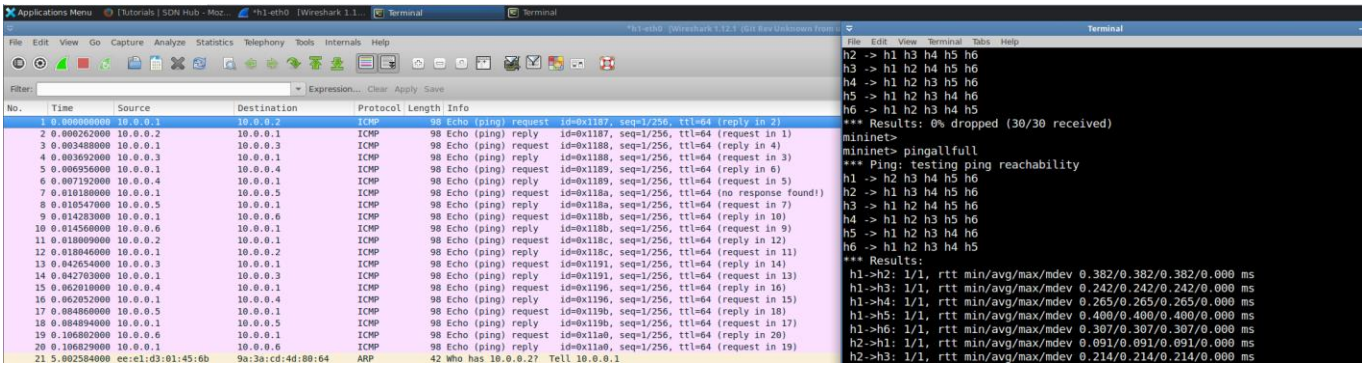


Fig8: Command **h1 wireshark &** to open wireshark and ran **iperf h1 h4** command to check the connection.

- It shows **TCP handshake** to indicate connection is established successfully and packets are received successfully.





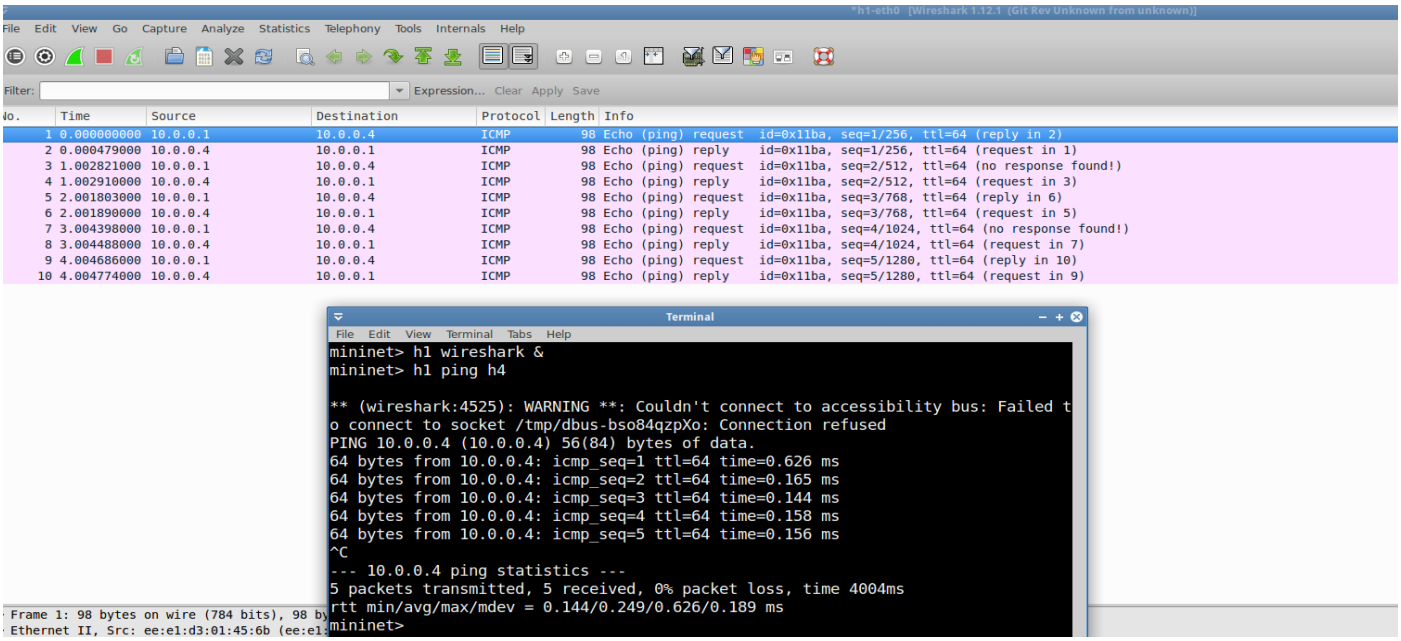


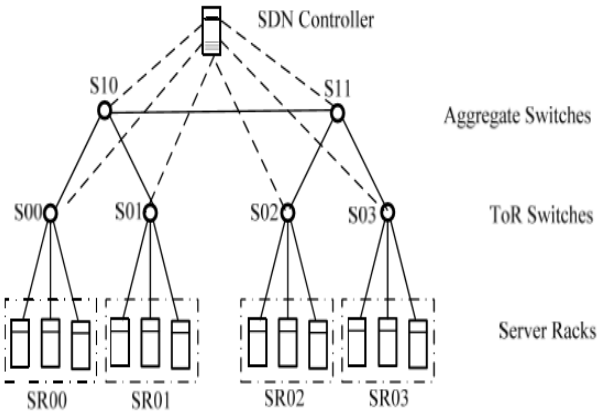
Fig9: Above three screenshots showing Wireshark for **pingall, pingallfull, and h1 ping h4** command showing **ICMP** messages

Simulation 2 :

Implement 12 Servers in 4 Racks, with 6 Switches, and an SDN Controller

Implement the following data center that includes 12 servers (hosts) in 4 server racks connected in a slightly different version of fully connected network. Run Mininet and:

[a] Conducted a test using Pingall command and report the average round trip time (RTT). Shown the snapshot of the result for all hosts.



GNU nano 2.2.6 File: topo2.py

```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 'h2' )
        host3 = self.addHost( 'h3' )
        host4 = self.addHost( 'h4' )
        host5 = self.addHost( 'h5' )
        host6 = self.addHost( 'h6' )
        host7 = self.addHost( 'h7' )
        host8 = self.addHost( 'h8' )
        host9 = self.addHost( 'h9' )
        host10 = self.addHost( 'h10' )
        host11 = self.addHost( 'h11' )
        host12 = self.addHost( 'h12' )

        # Add Switches
        switch00 = self.addSwitch( 's1' )
        switch01 = self.addSwitch( 's2' )
        switch02 = self.addSwitch( 's3' )
        switch03 = self.addSwitch( 's4' )
        switch10 = self.addSwitch( 's5' )
        switch11 = self.addSwitch( 's6' )


        # Add links
        self.addLink( host1, switch00 )
        self.addLink( host2, switch00 )
        self.addLink( host3, switch00 )
        self.addLink( host4, switch01 )
        self.addLink( host5, switch01 )
        self.addLink( host6, switch01 )
        self.addLink( host7, switch02 )
        self.addLink( host8, switch02 )
        self.addLink( host9, switch02 )
        self.addLink( host10, switch03 )
        self.addLink( host11, switch03 )
        self.addLink( host12, switch03 )

        self.addLink( switch00, switch10 )
        self.addLink( switch01, switch10 )
        self.addLink( switch02, switch11 )
        self.addLink( switch03, switch11 )
        self.addLink( switch10, switch11 )



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

```
ubuntu@sdnhubvm:~/mininetranjitha/custom[15:22] (master)$ ls
README  topo1.py  topo2.py  topo-2sw-2host.py
ubuntu@sdnhubvm:~/mininetranjitha/custom[15:22] (master)$ nano topo2.py
ubuntu@sdnhubvm:~/mininetranjitha/custom[15:22] (master)$ sudo mn --custom topo2.py --topo mytopo --controller=remote --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3) (h10, s4) (h11, s4) (h12, s4) (s1, s5) (s2, s5) (s3, s6) (s4, s6) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
```

Fig10: Code for creating and executing topology for simulation 2 and saved as topo2.py

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
```

Fig10b: code for **pingall** for simulation2 of topo2.py

```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.335/0.335/0.335/0.000 ms
 h1->h3: 1/1, rtt min/avg/max/mdev 0.326/0.326/0.326/0.000 ms
 h1->h4: 1/1, rtt min/avg/max/mdev 0.509/0.509/0.509/0.000 ms
 h1->h5: 1/1, rtt min/avg/max/mdev 0.443/0.443/0.443/0.000 ms
 h1->h6: 1/1, rtt min/avg/max/mdev 0.293/0.293/0.293/0.000 ms
 h1->h7: 1/1, rtt min/avg/max/mdev 0.530/0.530/0.530/0.000 ms
 h1->h8: 1/1, rtt min/avg/max/mdev 0.406/0.406/0.406/0.000 ms
 h1->h9: 1/1, rtt min/avg/max/mdev 0.649/0.649/0.649/0.000 ms
 h1->h10: 1/1, rtt min/avg/max/mdev 0.613/0.613/0.613/0.000 ms
 h1->h11: 1/1, rtt min/avg/max/mdev 0.623/0.623/0.623/0.000 ms
 h1->h12: 1/1, rtt min/avg/max/mdev 0.390/0.390/0.390/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 0.066/0.066/0.066/0.000 ms
 h2->h3: 1/1, rtt min/avg/max/mdev 0.389/0.389/0.389/0.000 ms
 h2->h4: 1/1, rtt min/avg/max/mdev 0.433/0.433/0.433/0.000 ms
 h2->h5: 1/1, rtt min/avg/max/mdev 0.295/0.295/0.295/0.000 ms
 h2->h6: 1/1, rtt min/avg/max/mdev 0.350/0.350/0.350/0.000 ms
 h2->h7: 1/1, rtt min/avg/max/mdev 0.431/0.431/0.431/0.000 ms
 h2->h8: 1/1, rtt min/avg/max/mdev 0.262/0.262/0.262/0.000 ms
 h2->h9: 1/1, rtt min/avg/max/mdev 0.483/0.483/0.483/0.000 ms
 h2->h10: 1/1, rtt min/avg/max/mdev 0.337/0.337/0.337/0.000 ms
 h2->h11: 1/1, rtt min/avg/max/mdev 0.197/0.197/0.197/0.000 ms
 h2->h12: 1/1, rtt min/avg/max/mdev 0.212/0.212/0.212/0.000 ms
 h3->h1: 1/1, rtt min/avg/max/mdev 0.093/0.093/0.093/0.000 ms
 h3->h2: 1/1, rtt min/avg/max/mdev 0.129/0.129/0.129/0.000 ms
 h3->h4: 1/1, rtt min/avg/max/mdev 0.590/0.590/0.590/0.000 ms
 h3->h5: 1/1, rtt min/avg/max/mdev 0.411/0.411/0.411/0.000 ms
 h3->h6: 1/1, rtt min/avg/max/mdev 0.301/0.301/0.301/0.000 ms
 h3->h7: 1/1, rtt min/avg/max/mdev 0.433/0.433/0.433/0.000 ms
 h3->h8: 1/1, rtt min/avg/max/mdev 0.397/0.397/0.397/0.000 ms
 h3->h9: 1/1, rtt min/avg/max/mdev 0.210/0.210/0.210/0.000 ms
 h3->h10: 1/1, rtt min/avg/max/mdev 0.395/0.395/0.395/0.000 ms
 h3->h11: 1/1, rtt min/avg/max/mdev 1.187/1.187/1.187/0.000 ms
 h3->h12: 1/1, rtt min/avg/max/mdev 0.385/0.385/0.385/0.000 ms
 h4->h1: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
 h4->h2: 1/1, rtt min/avg/max/mdev 0.128/0.128/0.128/0.000 ms
 h4->h3: 1/1, rtt min/avg/max/mdev 0.079/0.079/0.079/0.000 ms
 h4->h5: 1/1, rtt min/avg/max/mdev 0.226/0.226/0.226/0.000 ms
```

```
h4->h3: 1/1, rtt min/avg/max/mdev 0.079/0.079/0.079/0.000 ms
h4->h5: 1/1, rtt min/avg/max/mdev 0.226/0.226/0.226/0.000 ms
h4->h6: 1/1, rtt min/avg/max/mdev 0.184/0.184/0.184/0.000 ms
h4->h7: 1/1, rtt min/avg/max/mdev 0.844/0.844/0.844/0.000 ms
h4->h8: 1/1, rtt min/avg/max/mdev 0.535/0.535/0.535/0.000 ms
h4->h9: 1/1, rtt min/avg/max/mdev 0.373/0.373/0.373/0.000 ms
h4->h10: 1/1, rtt min/avg/max/mdev 0.441/0.441/0.441/0.000 ms
h4->h11: 1/1, rtt min/avg/max/mdev 0.464/0.464/0.464/0.000 ms
h4->h12: 1/1, rtt min/avg/max/mdev 0.328/0.328/0.328/0.000 ms
h5->h1: 1/1, rtt min/avg/max/mdev 0.075/0.075/0.075/0.000 ms
h5->h2: 1/1, rtt min/avg/max/mdev 0.084/0.084/0.084/0.000 ms
h5->h3: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
h5->h4: 1/1, rtt min/avg/max/mdev 0.092/0.092/0.092/0.000 ms
h5->h6: 1/1, rtt min/avg/max/mdev 0.340/0.340/0.340/0.000 ms
h5->h7: 1/1, rtt min/avg/max/mdev 0.477/0.477/0.477/0.000 ms
h5->h8: 1/1, rtt min/avg/max/mdev 0.500/0.500/0.500/0.000 ms
h5->h9: 1/1, rtt min/avg/max/mdev 0.571/0.571/0.571/0.000 ms
h5->h10: 1/1, rtt min/avg/max/mdev 0.281/0.281/0.281/0.000 ms
h5->h11: 1/1, rtt min/avg/max/mdev 0.218/0.218/0.218/0.000 ms
h5->h12: 1/1, rtt min/avg/max/mdev 0.209/0.209/0.209/0.000 ms
h6->h1: 1/1, rtt min/avg/max/mdev 0.116/0.116/0.116/0.000 ms
h6->h2: 1/1, rtt min/avg/max/mdev 0.121/0.121/0.121/0.000 ms
h6->h3: 1/1, rtt min/avg/max/mdev 0.099/0.099/0.099/0.000 ms
h6->h4: 1/1, rtt min/avg/max/mdev 0.119/0.119/0.119/0.000 ms
h6->h5: 1/1, rtt min/avg/max/mdev 0.074/0.074/0.074/0.000 ms
h6->h7: 1/1, rtt min/avg/max/mdev 1.049/1.049/1.049/0.000 ms
h6->h8: 1/1, rtt min/avg/max/mdev 0.277/0.277/0.277/0.000 ms
h6->h9: 1/1, rtt min/avg/max/mdev 0.214/0.214/0.214/0.000 ms
h6->h10: 1/1, rtt min/avg/max/mdev 0.239/0.239/0.239/0.000 ms
h6->h11: 1/1, rtt min/avg/max/mdev 0.203/0.203/0.203/0.000 ms
h6->h12: 1/1, rtt min/avg/max/mdev 0.359/0.359/0.359/0.000 ms
h7->h1: 1/1, rtt min/avg/max/mdev 0.125/0.125/0.125/0.000 ms
h7->h2: 1/1, rtt min/avg/max/mdev 0.114/0.114/0.114/0.000 ms
h7->h3: 1/1, rtt min/avg/max/mdev 0.236/0.236/0.236/0.000 ms
h7->h4: 1/1, rtt min/avg/max/mdev 0.113/0.113/0.113/0.000 ms
h7->h5: 1/1, rtt min/avg/max/mdev 0.057/0.057/0.057/0.000 ms
h7->h6: 1/1, rtt min/avg/max/mdev 0.093/0.093/0.093/0.000 ms
h7->h8: 1/1, rtt min/avg/max/mdev 0.322/0.322/0.322/0.000 ms
h7->h9: 1/1, rtt min/avg/max/mdev 0.235/0.235/0.235/0.000 ms
h7->h10: 1/1, rtt min/avg/max/mdev 0.451/0.451/0.451/0.000 ms
h7->h11: 1/1, rtt min/avg/max/mdev 1.155/1.155/1.155/0.000 ms
h7->h12: 1/1, rtt min/avg/max/mdev 0.272/0.272/0.272/0.000 ms
h8->h1: 1/1, rtt min/avg/max/mdev 0.076/0.076/0.076/0.000 ms
h8->h2: 1/1, rtt min/avg/max/mdev 0.129/0.129/0.129/0.000 ms
h8->h3: 1/1, rtt min/avg/max/mdev 0.137/0.137/0.137/0.000 ms
h8->h4: 1/1, rtt min/avg/max/mdev 0.114/0.114/0.114/0.000 ms
h8->h5: 1/1, rtt min/avg/max/mdev 0.113/0.113/0.113/0.000 ms
h8->h6: 1/1, rtt min/avg/max/mdev 0.124/0.124/0.124/0.000 ms
h8->h7: 1/1, rtt min/avg/max/mdev 0.079/0.079/0.079/0.000 ms
h8->h9: 1/1, rtt min/avg/max/mdev 0.327/0.327/0.327/0.000 ms
```

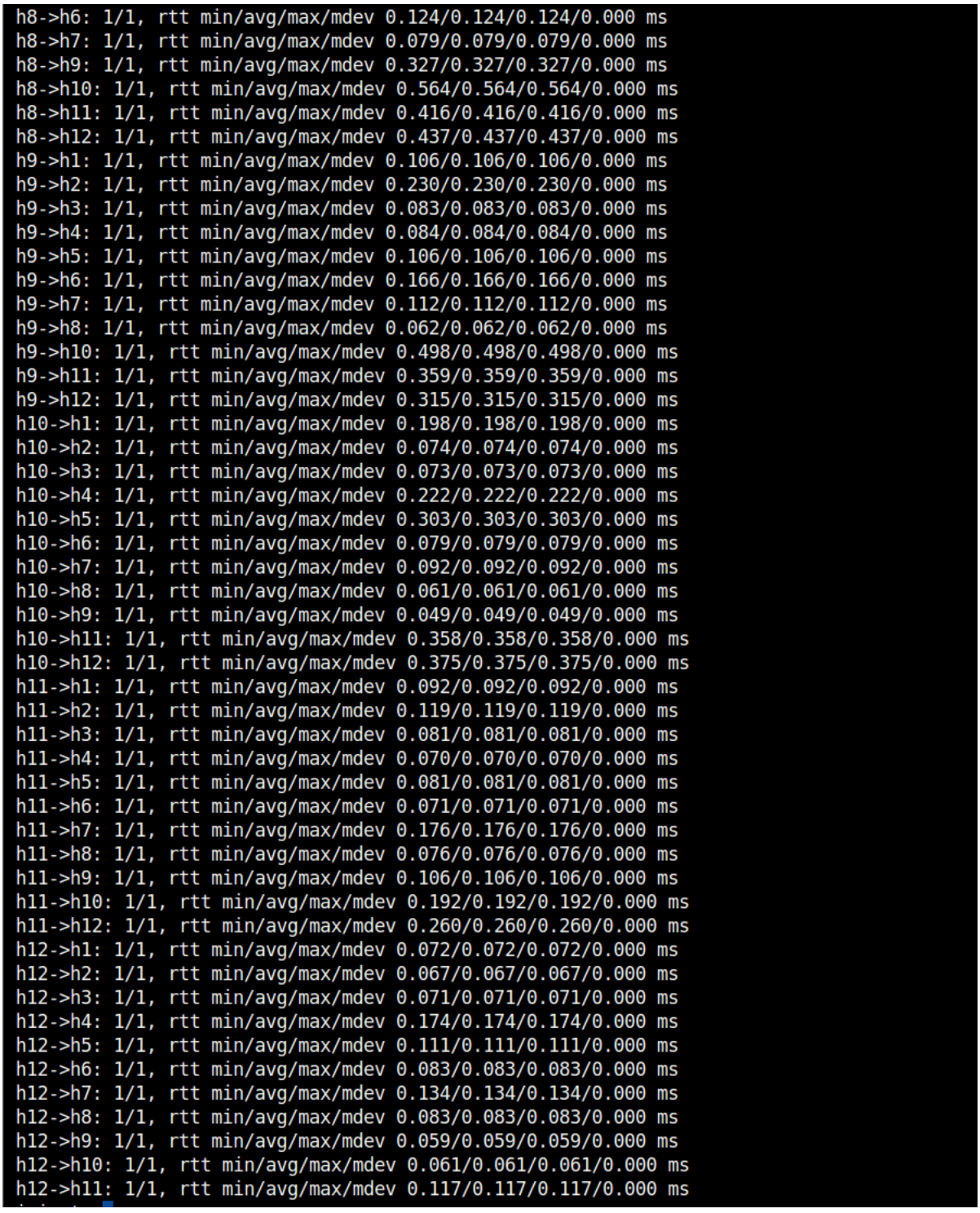


Fig11: Command **pingallfull** representing the average RTT for all the hosts for simulation 2 of topo2.py

- Here we can observe that all hosts are communicating with every other hosts as showcased by the pingall command. Each host while communicating with other hosts in

pingallfull command will showcase different RTT average. **RTT average differs for connection established between different hosts.**

[b] Comment specifically on an RTT over a connection from a server in S01 rack to a server in S04 rack.

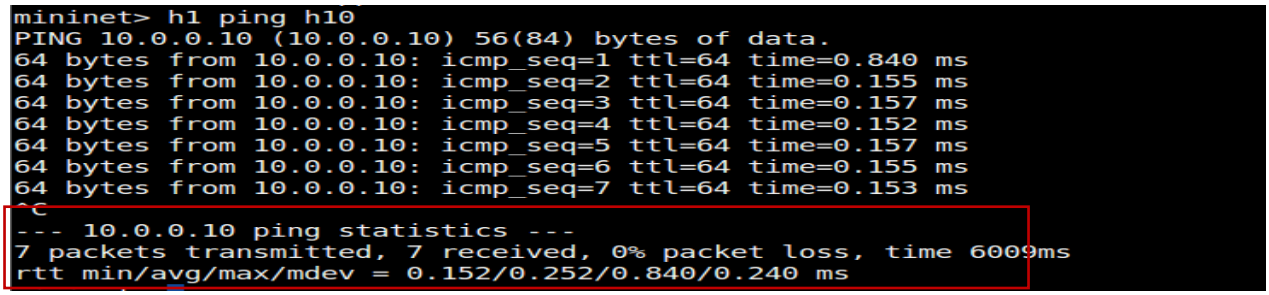


Fig12: Command **h1 ping h10** is showing the connection from h1 of s01 rack to h10 of s04 rack.

- Here we can observe the RTT for a particular connection from h1 to h10.
- We observe that **icmp** messages are sent as a result of successful connection established. 7packets are transmitted and 7 are received with **RTT average of 0.252** secs. Here we can observe that all hosts are communicating with every other hosts. Each host while communicating with other hosts will showcase different RTT.

[c] Experience iperf a server in S01 rack to a server in S04 rack. Show the snapshot of your result.

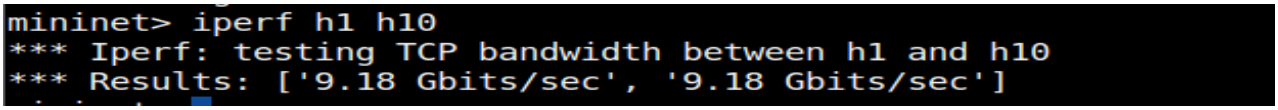


Fig13: Command **iperf h1 h10** showcases the bandwith between h1 and h10 as **9.18 gbps.**

[d] Experience Wireshark on a server in S01 rack to a server in S04 rack. Show the snapshot of your result.

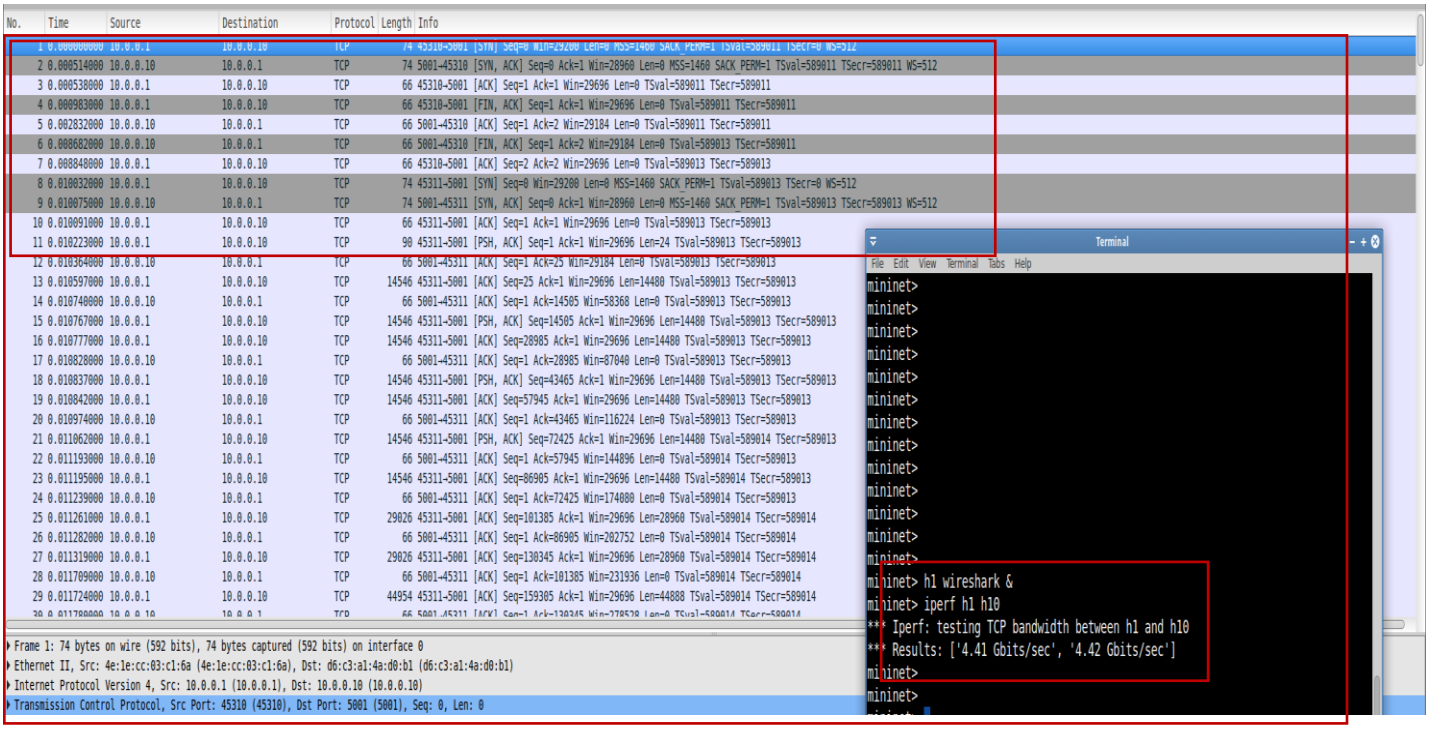
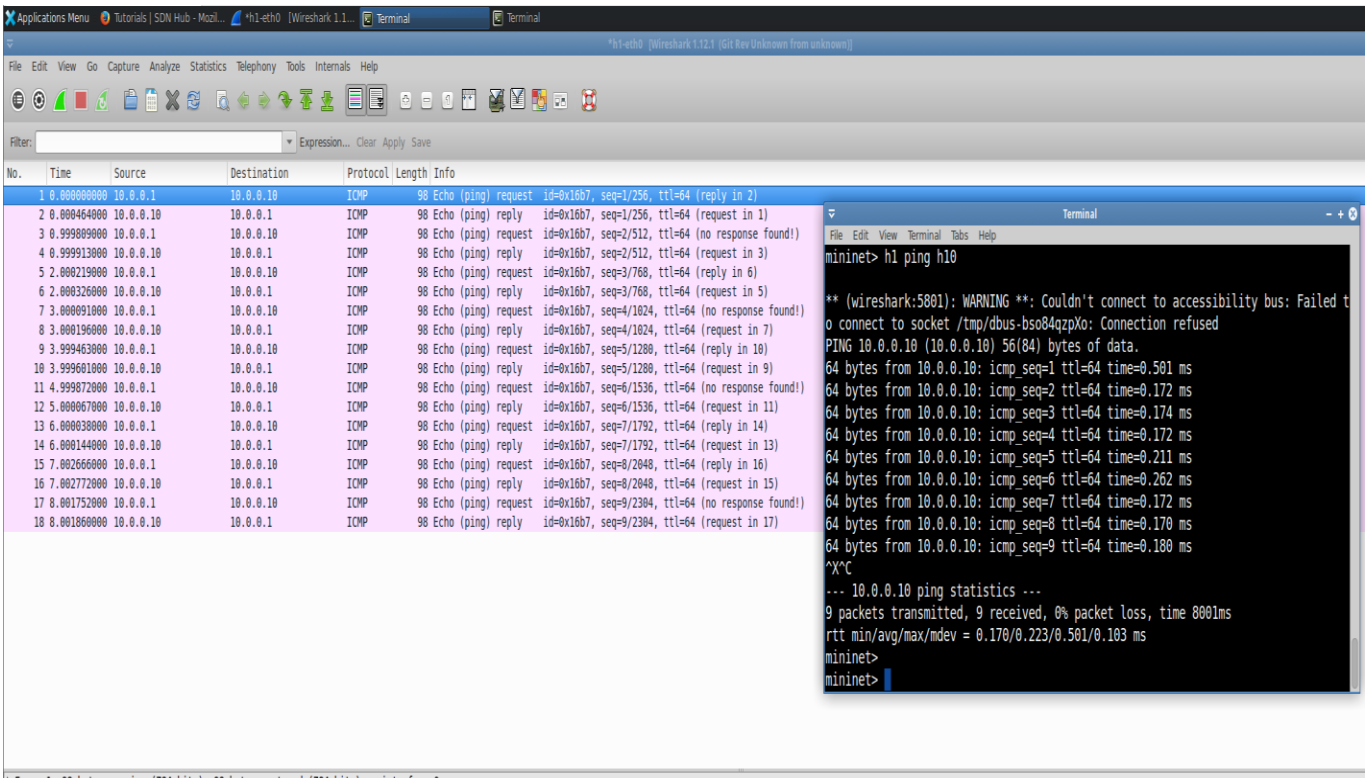
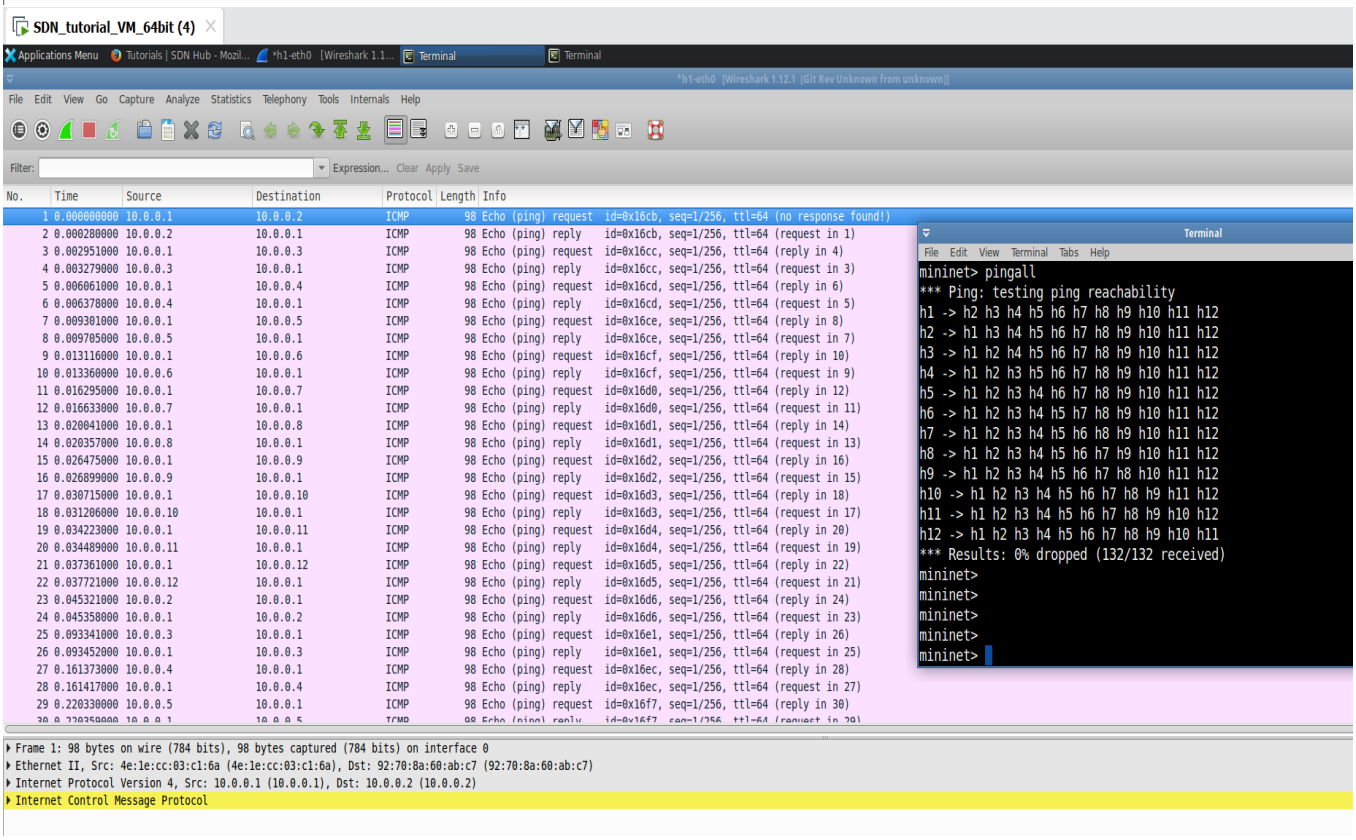


Fig14: Command **h1 wireshark &** to open wireshark and ran **iperf h1 h4** command to check the connection.

- It shows **TCP handshake** to indicate connection is established successfully and packets are received successfully.


Applications Menu
Tutorials | SDN Hub - Mozil...
*h1-eth0 [Wireshark 1.1...
Terminal
Terminal
*h1-eth0 [Wireshark 1.12.1 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
1 0.000000000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=1/256, ttl=64 (reply in 2)
2 0.000464000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=1/256, ttl=64 (request in 1)
3 0.999809000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=2/512, ttl=64 (no response found!)
4 0.999913000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=2/512, ttl=64 (request in 3)
5 2.000219000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=3/768, ttl=64 (reply in 6)
6 2.000326000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=3/768, ttl=64 (request in 5)
7 3.000091000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=4/1024, ttl=64 (no response found!)
8 3.000196000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=4/1024, ttl=64 (request in 7)
9 3.999463000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=5/1288, ttl=64 (reply in 10)
10 3.999601000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=5/1288, ttl=64 (request in 9)
11 4.999872000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=6/1536, ttl=64 (no response found!)
12 5.000067000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=6/1536, ttl=64 (request in 11)
13 6.000038000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=7/1792, ttl=64 (reply in 14)
14 6.000144000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=7/1792, ttl=64 (request in 13)
15 7.002666000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=8/2048, ttl=64 (reply in 16)
16 7.002772000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=8/2048, ttl=64 (request in 15)
17 8.001752000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16b7, seq=9/2304, ttl=64 (no response found!)
18 8.001860000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16b7, seq=9/2304, ttl=64 (request in 17)
Terminal
File Edit View Terminal Tabs Help
mininet> h1 ping h10
** (wireshark:5801): WARNING **: Couldn't connect to accessibility bus: Failed t
o connect to socket /tmp/dbus-bso84qzpXo: Connection refused
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.501 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.172 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.174 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=64 time=0.172 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=64 time=0.211 ms
64 bytes from 10.0.0.10: icmp_seq=6 ttl=64 time=0.262 ms
64 bytes from 10.0.0.10: icmp_seq=7 ttl=64 time=0.172 ms
64 bytes from 10.0.0.10: icmp_seq=8 ttl=64 time=0.170 ms
64 bytes from 10.0.0.10: icmp_seq=9 ttl=64 time=0.180 ms
^X^C
--- 10.0.0.10 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8001ms
rtt min/avg/max/mdev = 0.170/0.223/0.501/0.103 ms
mininet>
mininet>
Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 4e:1e:cc:83:c1:6a (4e:1e:cc:83:c1:6a), Dst: d6:c3:a1:4a:d0:b1 (d6:c3:a1:4a:d0:b1)
Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.10 (10.0.0.10)
Internet Control Message Protocol


SDN_tutorial_VM_64bit (4)
Applications Menu
Tutorials | SDN Hub - Mozil...
*h1-eth0 [Wireshark 1.1...
Terminal
Terminal
*h1-eth0 [Wireshark 1.12.1 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
1 0.000000000 10.0.0.1 10.0.0.2 ICMP 98 Echo (ping) request id=0x16cb, seq=1/256, ttl=64 (no response found!)
2 0.000280000 10.0.0.2 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16cb, seq=1/256, ttl=64 (request in 1)
3 0.002951000 10.0.0.1 10.0.0.3 ICMP 98 Echo (ping) request id=0x16cc, seq=1/256, ttl=64 (reply in 4)
4 0.003279000 10.0.0.3 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16cc, seq=1/256, ttl=64 (request in 3)
5 0.006061000 10.0.0.1 10.0.0.4 ICMP 98 Echo (ping) request id=0x16cd, seq=1/256, ttl=64 (reply in 6)
6 0.006378000 10.0.0.4 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16cd, seq=1/256, ttl=64 (request in 5)
7 0.009301000 10.0.0.1 10.0.0.5 ICMP 98 Echo (ping) request id=0x16ce, seq=1/256, ttl=64 (reply in 8)
8 0.009705000 10.0.0.5 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16ce, seq=1/256, ttl=64 (request in 7)
9 0.013116000 10.0.0.1 10.0.0.6 ICMP 98 Echo (ping) request id=0x16cf, seq=1/256, ttl=64 (reply in 10)
10 0.013360000 10.0.0.6 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16cf, seq=1/256, ttl=64 (request in 9)
11 0.016295000 10.0.0.1 10.0.0.7 ICMP 98 Echo (ping) request id=0x16d0, seq=1/256, ttl=64 (reply in 12)
12 0.016633000 10.0.0.7 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16d0, seq=1/256, ttl=64 (request in 11)
13 0.020041000 10.0.0.1 10.0.0.8 ICMP 98 Echo (ping) request id=0x16d1, seq=1/256, ttl=64 (reply in 14)
14 0.020357000 10.0.0.8 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16d1, seq=1/256, ttl=64 (request in 13)
15 0.026475000 10.0.0.1 10.0.0.9 ICMP 98 Echo (ping) request id=0x16d2, seq=1/256, ttl=64 (reply in 16)
16 0.026899000 10.0.0.9 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16d2, seq=1/256, ttl=64 (request in 15)
17 0.030715000 10.0.0.1 10.0.0.10 ICMP 98 Echo (ping) request id=0x16d3, seq=1/256, ttl=64 (reply in 18)
18 0.031206000 10.0.0.10 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16d3, seq=1/256, ttl=64 (request in 17)
19 0.034223000 10.0.0.1 10.0.0.11 ICMP 98 Echo (ping) request id=0x16d4, seq=1/256, ttl=64 (reply in 20)
20 0.034489000 10.0.0.11 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16d4, seq=1/256, ttl=64 (request in 19)
21 0.037361000 10.0.0.1 10.0.0.12 ICMP 98 Echo (ping) request id=0x16d5, seq=1/256, ttl=64 (reply in 22)
22 0.037721000 10.0.0.12 10.0.0.1 ICMP 98 Echo (ping) reply id=0x16d5, seq=1/256, ttl=64 (request in 21)
23 0.045321000 10.0.0.2 10.0.0.1 ICMP 98 Echo (ping) request id=0x16d6, seq=1/256, ttl=64 (reply in 24)
24 0.045358000 10.0.0.1 10.0.0.2 ICMP 98 Echo (ping) reply id=0x16d6, seq=1/256, ttl=64 (request in 23)
25 0.093341000 10.0.0.3 10.0.0.1 ICMP 98 Echo (ping) request id=0x16e1, seq=1/256, ttl=64 (reply in 26)
26 0.093452000 10.0.0.1 10.0.0.3 ICMP 98 Echo (ping) reply id=0x16e1, seq=1/256, ttl=64 (request in 25)
27 0.161373000 10.0.0.4 10.0.0.1 ICMP 98 Echo (ping) request id=0x16ec, seq=1/256, ttl=64 (reply in 28)
28 0.161417000 10.0.0.1 10.0.0.4 ICMP 98 Echo (ping) reply id=0x16ec, seq=1/256, ttl=64 (request in 27)
29 0.220330000 10.0.0.5 10.0.0.1 ICMP 98 Echo (ping) request id=0x16f7, seq=1/256, ttl=64 (reply in 30)
Terminal
File Edit View Terminal Tabs Help
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
mininet>
mininet>
mininet>
mininet>
mininet>
Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 4e:1e:cc:03:c1:6a (4e:1e:cc:03:c1:6a), Dst: 92:70:8a:60:ab:c7 (92:70:8a:60:ab:c7)
Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.2 (10.0.0.2)
Internet Control Message Protocol

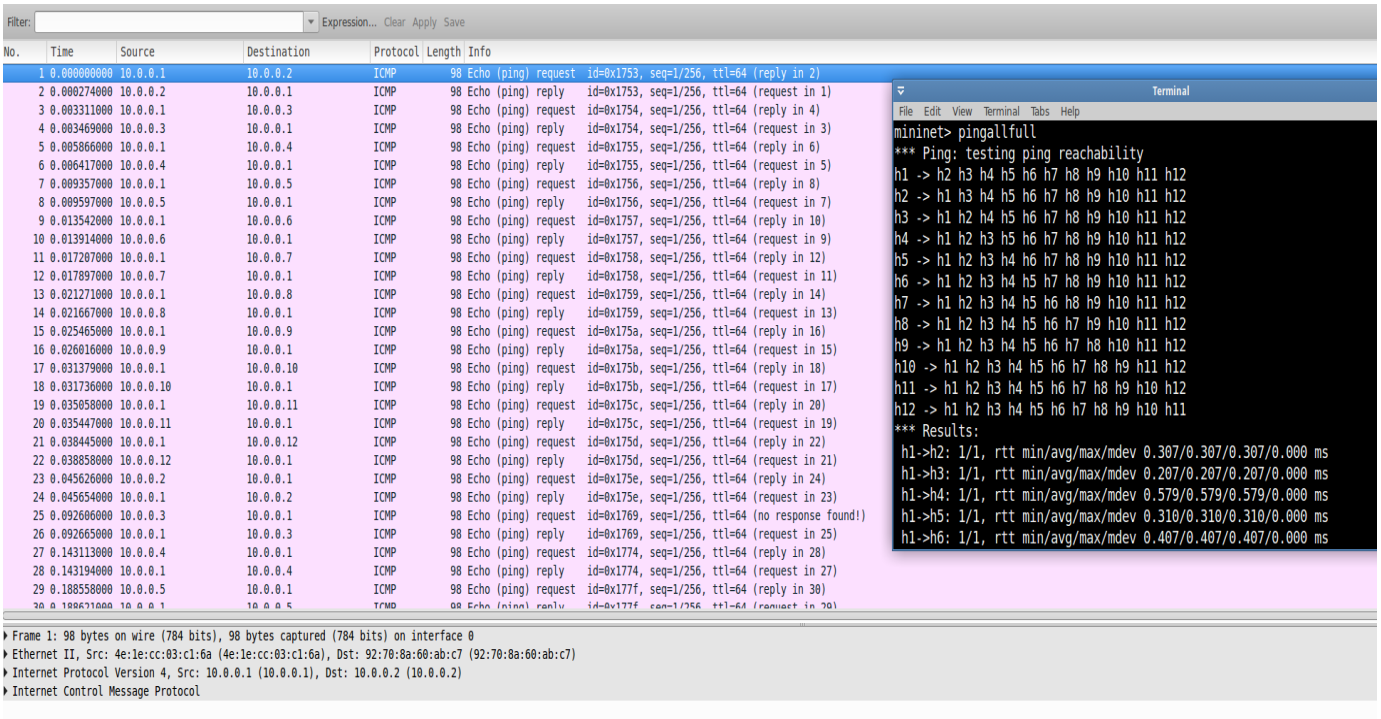


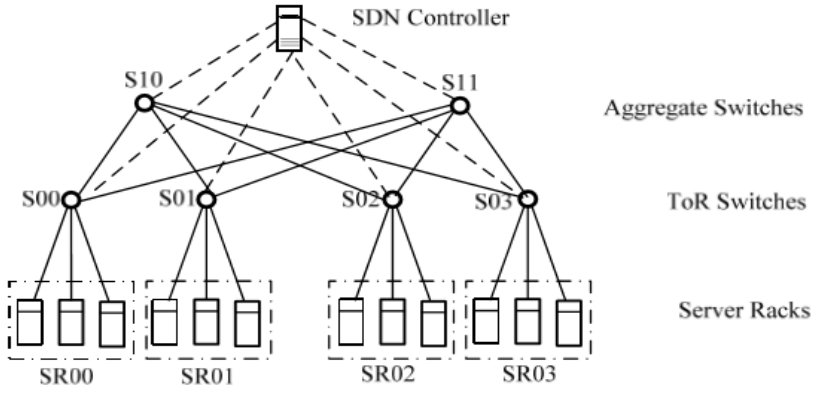
Fig15: Above three screenshots showing Wireshark for **pingall, pingallfull, and h1 ping h10** command **showing ICMP messages**

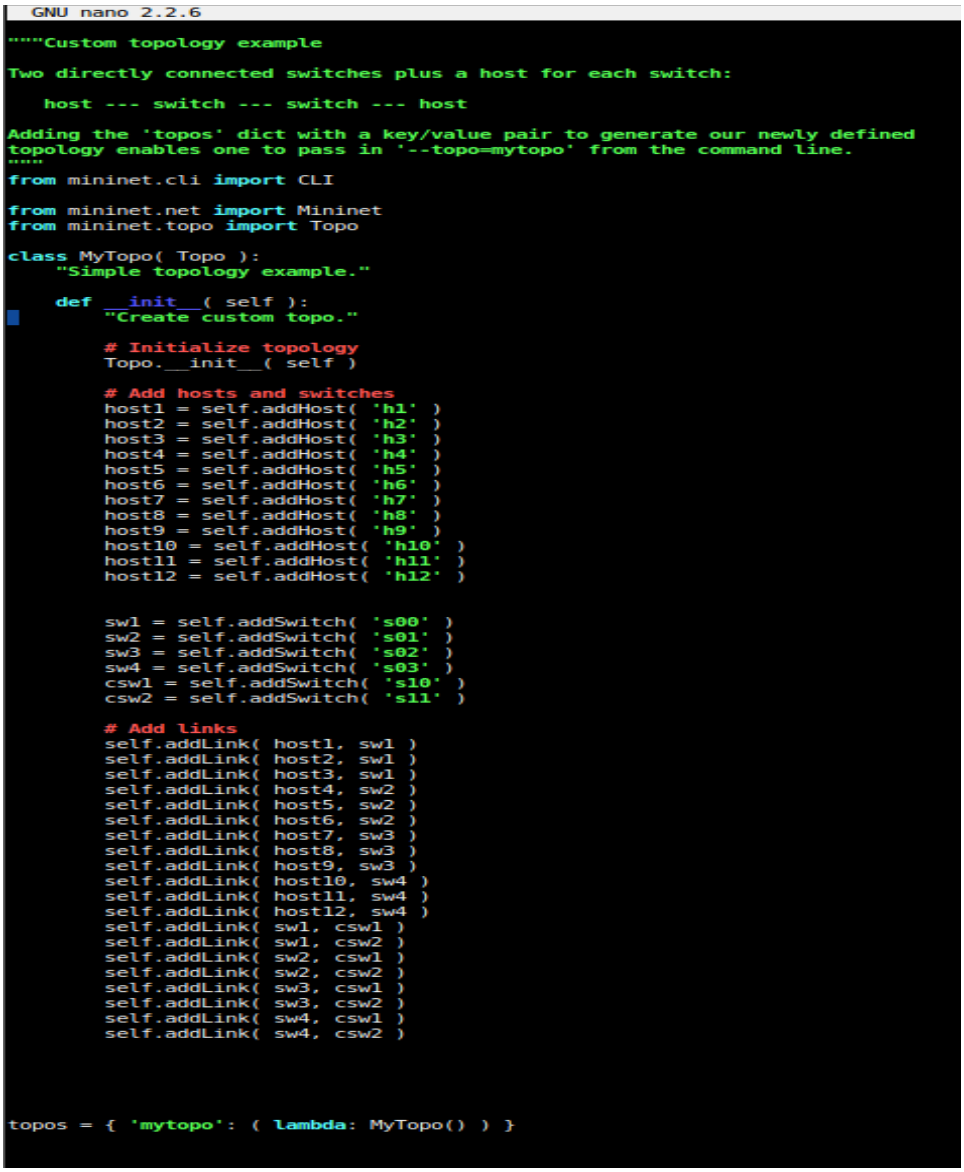
Simulation 3

Implement 12 Servers in 4 Racks, with 6 Switches, and an SDN Controller (Fully Connected).

Implement the following data center that includes 12 servers (hosts) in 4 server racks connected in a fully connected network. Run Mininet and:

[a] Conducted a test using Pingall command and report the average round trip time (RTT). Shown the snapshot of the result for all hosts.





```
  GNU nano 2.2.6

"""Custom topology example

Two directly connected switches plus a host for each switch:

   host --- switch --- switch --- host

Adding the 'topos' dict with a key/value pair to generate our newly defined
topology enables one to pass in '--topo=mytopo' from the command line.
"""
from mininet.cli import CLI

from mininet.net import Mininet
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 'h2' )
        host3 = self.addHost( 'h3' )
        host4 = self.addHost( 'h4' )
        host5 = self.addHost( 'h5' )
        host6 = self.addHost( 'h6' )
        host7 = self.addHost( 'h7' )
        host8 = self.addHost( 'h8' )
        host9 = self.addHost( 'h9' )
        host10 = self.addHost( 'h10' )
        host11 = self.addHost( 'h11' )
        host12 = self.addHost( 'h12' )


        sw1 = self.addSwitch( 's00' )
        sw2 = self.addSwitch( 's01' )
        sw3 = self.addSwitch( 's02' )
        sw4 = self.addSwitch( 's03' )
        csw1 = self.addSwitch( 's10' )
        csw2 = self.addSwitch( 's11' )

        # Add links
        self.addLink( host1, sw1 )
        self.addLink( host2, sw1 )
        self.addLink( host3, sw1 )
        self.addLink( host4, sw2 )
        self.addLink( host5, sw2 )
        self.addLink( host6, sw2 )
        self.addLink( host7, sw3 )
        self.addLink( host8, sw3 )
        self.addLink( host9, sw3 )
        self.addLink( host10, sw4 )
        self.addLink( host11, sw4 )
        self.addLink( host12, sw4 )
        self.addLink( sw1, csw1 )
        self.addLink( sw1, csw2 )
        self.addLink( sw2, csw1 )
        self.addLink( sw2, csw2 )
        self.addLink( sw3, csw1 )
        self.addLink( sw3, csw2 )
        self.addLink( sw4, csw1 )
        self.addLink( sw4, csw2 )



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

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Adding switches:
s00 s01 s02 s03 s10 s11
*** Adding links:
(h1, s00) (h2, s00) (h3, s00) (h4, s01) (h5, s01) (h6, s01) (h7, s02) (h8, s02) (h9, s02) (h10, s03) (h11, s03) (h12, s03) (s00, s10) (s00, s11) (s01, s10) (s01, s11) (s02, s10) (s02, s11) (s03, s10) (s03, s11)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controller
c0
*** Starting 6 switches
s00 s01 s02 s03 s10 s11 ...
*** Starting CLI:
```

Fig16: Code for creating and executing topology for simulation 3 and saved as topology3.py

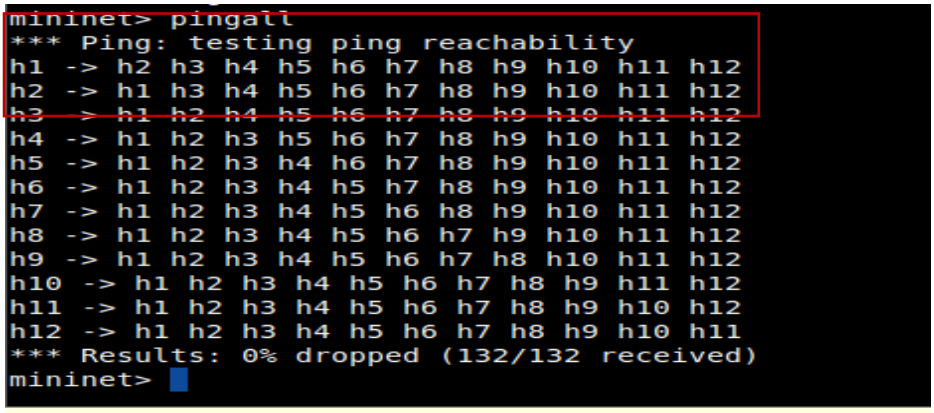
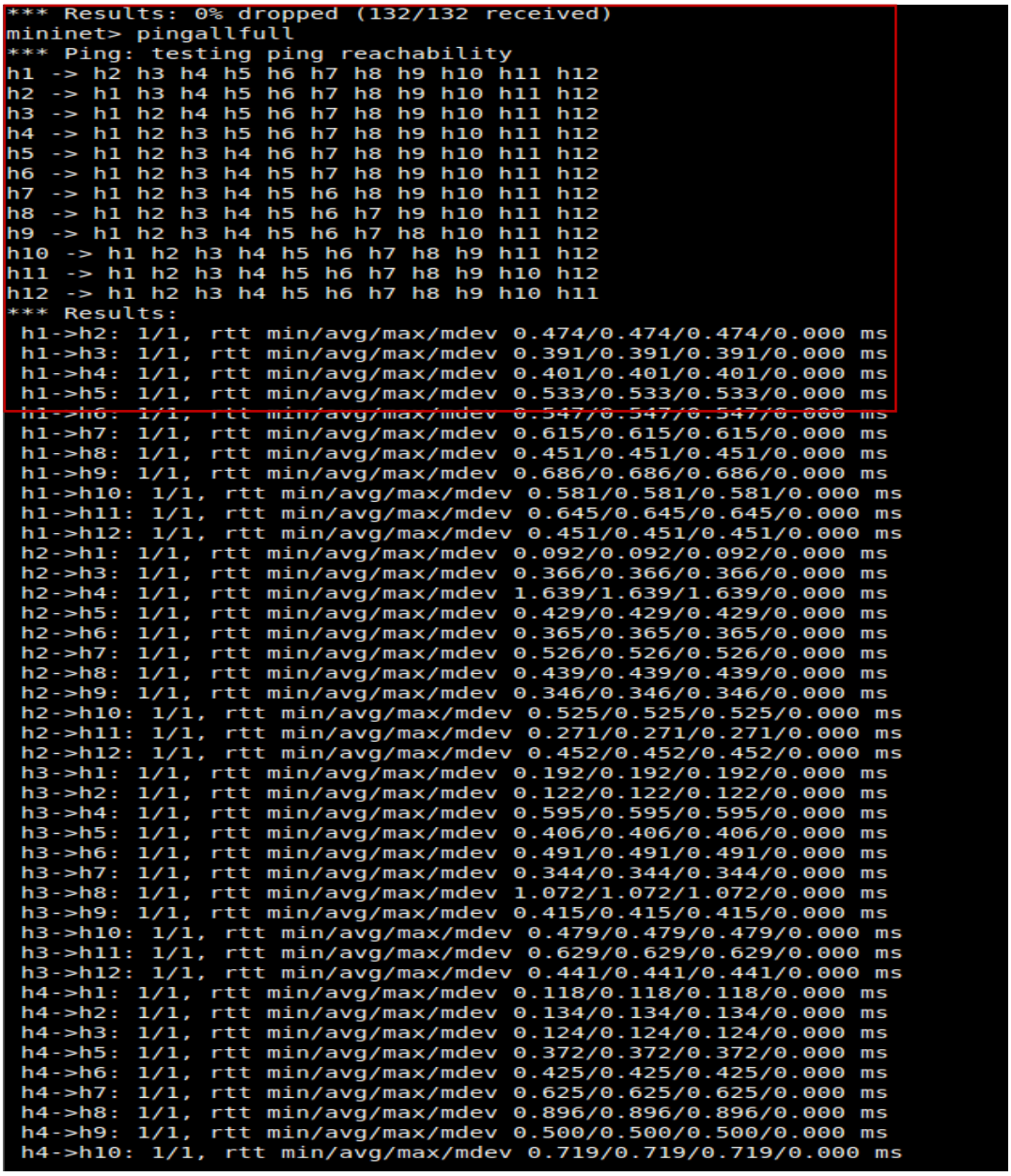


Fig17: Command **pingall** for simulation 3 of topology3.py



```
h4->h10: 1/1, rtt min/avg/max/mdev 0.719/0.719/0.719/0.000 ms
h4->h11: 1/1, rtt min/avg/max/mdev 0.790/0.790/0.790/0.000 ms
h4->h12: 1/1, rtt min/avg/max/mdev 0.599/0.599/0.599/0.000 ms
h5->h1: 1/1, rtt min/avg/max/mdev 0.123/0.123/0.123/0.000 ms
h5->h2: 1/1, rtt min/avg/max/mdev 0.114/0.114/0.114/0.000 ms
h5->h3: 1/1, rtt min/avg/max/mdev 0.110/0.110/0.110/0.000 ms
h5->h4: 1/1, rtt min/avg/max/mdev 0.228/0.228/0.228/0.000 ms
h5->h6: 1/1, rtt min/avg/max/mdev 0.339/0.339/0.339/0.000 ms
h5->h7: 1/1, rtt min/avg/max/mdev 0.542/0.542/0.542/0.000 ms
h5->h8: 1/1, rtt min/avg/max/mdev 0.862/0.862/0.862/0.000 ms
h5->h9: 1/1, rtt min/avg/max/mdev 0.393/0.393/0.393/0.000 ms
h5->h10: 1/1, rtt min/avg/max/mdev 1.980/1.980/1.980/0.000 ms
h5->h11: 1/1, rtt min/avg/max/mdev 0.612/0.612/0.612/0.000 ms
h5->h12: 1/1, rtt min/avg/max/mdev 0.351/0.351/0.351/0.000 ms
h6->h1: 1/1, rtt min/avg/max/mdev 0.105/0.105/0.105/0.000 ms
h6->h2: 1/1, rtt min/avg/max/mdev 0.110/0.110/0.110/0.000 ms
h6->h3: 1/1, rtt min/avg/max/mdev 0.564/0.564/0.564/0.000 ms
h6->h4: 1/1, rtt min/avg/max/mdev 0.104/0.104/0.104/0.000 ms
h6->h5: 1/1, rtt min/avg/max/mdev 0.113/0.113/0.113/0.000 ms
h6->h7: 1/1, rtt min/avg/max/mdev 0.472/0.472/0.472/0.000 ms
h6->h8: 1/1, rtt min/avg/max/mdev 0.391/0.391/0.391/0.000 ms
h6->h9: 1/1, rtt min/avg/max/mdev 0.340/0.340/0.340/0.000 ms
h6->h10: 1/1, rtt min/avg/max/mdev 0.294/0.294/0.294/0.000 ms
h6->h11: 1/1, rtt min/avg/max/mdev 0.409/0.409/0.409/0.000 ms
h6->h12: 1/1, rtt min/avg/max/mdev 0.389/0.389/0.389/0.000 ms
h7->h1: 1/1, rtt min/avg/max/mdev 0.092/0.092/0.092/0.000 ms
h7->h2: 1/1, rtt min/avg/max/mdev 0.094/0.094/0.094/0.000 ms
h7->h3: 1/1, rtt min/avg/max/mdev 0.122/0.122/0.122/0.000 ms
h7->h4: 1/1, rtt min/avg/max/mdev 0.156/0.156/0.156/0.000 ms
h7->h5: 1/1, rtt min/avg/max/mdev 0.097/0.097/0.097/0.000 ms
h7->h6: 1/1, rtt min/avg/max/mdev 0.072/0.072/0.072/0.000 ms
h7->h8: 1/1, rtt min/avg/max/mdev 0.358/0.358/0.358/0.000 ms
h7->h9: 1/1, rtt min/avg/max/mdev 0.431/0.431/0.431/0.000 ms
h7->h10: 1/1, rtt min/avg/max/mdev 0.820/0.820/0.820/0.000 ms
h7->h11: 1/1, rtt min/avg/max/mdev 0.447/0.447/0.447/0.000 ms
h7->h12: 1/1, rtt min/avg/max/mdev 0.437/0.437/0.437/0.000 ms
h8->h1: 1/1, rtt min/avg/max/mdev 0.096/0.096/0.096/0.000 ms
h8->h2: 1/1, rtt min/avg/max/mdev 0.099/0.099/0.099/0.000 ms
h8->h3: 1/1, rtt min/avg/max/mdev 0.188/0.188/0.188/0.000 ms
h8->h4: 1/1, rtt min/avg/max/mdev 0.106/0.106/0.106/0.000 ms
h8->h5: 1/1, rtt min/avg/max/mdev 0.116/0.116/0.116/0.000 ms
h8->h6: 1/1, rtt min/avg/max/mdev 0.085/0.085/0.085/0.000 ms
h8->h7: 1/1, rtt min/avg/max/mdev 0.086/0.086/0.086/0.000 ms
h8->h9: 1/1, rtt min/avg/max/mdev 0.314/0.314/0.314/0.000 ms
h8->h10: 1/1, rtt min/avg/max/mdev 0.325/0.325/0.325/0.000 ms
h8->h11: 1/1, rtt min/avg/max/mdev 0.394/0.394/0.394/0.000 ms
h8->h12: 1/1, rtt min/avg/max/mdev 0.304/0.304/0.304/0.000 ms
h9->h1: 1/1, rtt min/avg/max/mdev 0.114/0.114/0.114/0.000 ms
h9->h2: 1/1, rtt min/avg/max/mdev 0.099/0.099/0.099/0.000 ms
h9->h3: 1/1, rtt min/avg/max/mdev 0.110/0.110/0.110/0.000 ms
h9->h4: 1/1, rtt min/avg/max/mdev 0.121/0.121/0.121/0.000 ms
h9->h5: 1/1, rtt min/avg/max/mdev 0.127/0.127/0.127/0.000 ms
h9->h6: 1/1, rtt min/avg/max/mdev 0.089/0.089/0.089/0.000 ms
h9->h7: 1/1, rtt min/avg/max/mdev 0.077/0.077/0.077/0.000 ms
h9->h8: 1/1, rtt min/avg/max/mdev 0.109/0.109/0.109/0.000 ms
h9->h10: 1/1, rtt min/avg/max/mdev 0.267/0.267/0.267/0.000 ms
h9->h11: 1/1, rtt min/avg/max/mdev 0.314/0.314/0.314/0.000 ms
```

```
h9->h8: 1/1, rtt min/avg/max/mdev 0.109/0.109/0.109/0.000 ms
h9->h10: 1/1, rtt min/avg/max/mdev 0.267/0.267/0.267/0.000 ms
h9->h11: 1/1, rtt min/avg/max/mdev 0.314/0.314/0.314/0.000 ms
h9->h12: 1/1, rtt min/avg/max/mdev 0.201/0.201/0.201/0.000 ms
h10->h1: 1/1, rtt min/avg/max/mdev 0.290/0.290/0.290/0.000 ms
h10->h2: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
h10->h3: 1/1, rtt min/avg/max/mdev 0.085/0.085/0.085/0.000 ms
h10->h4: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
h10->h5: 1/1, rtt min/avg/max/mdev 0.100/0.100/0.100/0.000 ms
h10->h6: 1/1, rtt min/avg/max/mdev 0.167/0.167/0.167/0.000 ms
h10->h7: 1/1, rtt min/avg/max/mdev 0.100/0.100/0.100/0.000 ms
h10->h8: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
h10->h9: 1/1, rtt min/avg/max/mdev 0.084/0.084/0.084/0.000 ms
h10->h11: 1/1, rtt min/avg/max/mdev 0.421/0.421/0.421/0.000 ms
h10->h12: 1/1, rtt min/avg/max/mdev 0.294/0.294/0.294/0.000 ms
h11->h1: 1/1, rtt min/avg/max/mdev 0.230/0.230/0.230/0.000 ms
h11->h2: 1/1, rtt min/avg/max/mdev 0.163/0.163/0.163/0.000 ms
h11->h3: 1/1, rtt min/avg/max/mdev 0.098/0.098/0.098/0.000 ms
h11->h4: 1/1, rtt min/avg/max/mdev 0.102/0.102/0.102/0.000 ms
h11->h5: 1/1, rtt min/avg/max/mdev 0.161/0.161/0.161/0.000 ms
h11->h6: 1/1, rtt min/avg/max/mdev 0.121/0.121/0.121/0.000 ms
h11->h7: 1/1, rtt min/avg/max/mdev 0.089/0.089/0.089/0.000 ms
h11->h8: 1/1, rtt min/avg/max/mdev 0.090/0.090/0.090/0.000 ms
h11->h9: 1/1, rtt min/avg/max/mdev 0.103/0.103/0.103/0.000 ms
h11->h10: 1/1, rtt min/avg/max/mdev 0.200/0.200/0.200/0.000 ms
h11->h12: 1/1, rtt min/avg/max/mdev 0.311/0.311/0.311/0.000 ms
h12->h1: 1/1, rtt min/avg/max/mdev 0.085/0.085/0.085/0.000 ms
h12->h2: 1/1, rtt min/avg/max/mdev 0.211/0.211/0.211/0.000 ms
h12->h3: 1/1, rtt min/avg/max/mdev 0.131/0.131/0.131/0.000 ms
h12->h4: 1/1, rtt min/avg/max/mdev 0.102/0.102/0.102/0.000 ms
h12->h5: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
h12->h6: 1/1, rtt min/avg/max/mdev 0.310/0.310/0.310/0.000 ms
h12->h7: 1/1, rtt min/avg/max/mdev 0.084/0.084/0.084/0.000 ms
h12->h8: 1/1, rtt min/avg/max/mdev 0.088/0.088/0.088/0.000 ms
h12->h9: 1/1, rtt min/avg/max/mdev 0.106/0.106/0.106/0.000 ms
h12->h10: 1/1, rtt min/avg/max/mdev 0.076/0.076/0.076/0.000 ms
h12->h11: 1/1, rtt min/avg/max/mdev 0.165/0.165/0.165/0.000 ms
mininet>
```

Fig18: Command **pingallfull** representing the average RTT for all the hosts for simulation 3 of topology3.py

- Here we can observe that all hosts are communicating with every other hosts as showcased by the pingall command. Each host while communicating with other hosts in pingallfull command will showcase different RTT average. **RTT average differs for connection established between different hosts.**

[b] Comment specifically on an RTT over a connection from a server in S00 rack to a server in S03 rack.

```
mininet> h1 ping h10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=1.00 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.118 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.114 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=64 time=0.117 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=64 time=0.148 ms
64 bytes from 10.0.0.10: icmp_seq=6 ttl=64 time=0.196 ms
64 bytes from 10.0.0.10: icmp_seq=7 ttl=64 time=0.264 ms
64 bytes from 10.0.0.10: icmp_seq=8 ttl=64 time=0.156 ms
64 bytes from 10.0.0.10: icmp_seq=9 ttl=64 time=0.278 ms
64 bytes from 10.0.0.10: icmp_seq=10 ttl=64 time=0.172 ms
64 bytes from 10.0.0.10: icmp_seq=11 ttl=64 time=0.133 ms
^C
--- 10.0.0.10 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10017ms
rtt min/avg/max/mdev = 0.114/0.245/1.002/0.245 ms
mininet>
```

Fig19: Command **h1 ping h10** is showing the connection from h1 of s01 rack to h10 of s04 rack.

- Here we can observe the RTT for a particular connection from h1 to h10.
- We observe that icmp messages are sent as a result of successful connection established. 11 packets are transmitted and 11 are received with **RTT average of 0.245** secs. Here we can observe that all hosts are communicating with every other hosts. Each host while communicating with other hosts will showcase different RTT.

[c] Experience iperf a server in S00 rack to a server in S03 rack. Show the snapshot of your result.

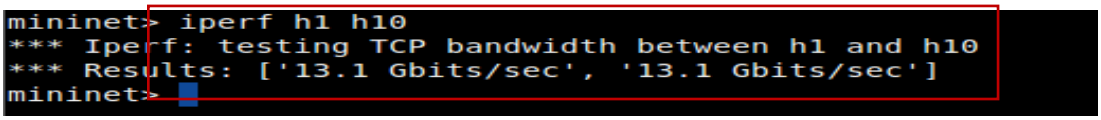


Fig20: Command **iperf h1 h10** showcases the bandwith between h1 and h10 as **13.1 gbps**.

[d] Experience Wireshark on a server in S00 rack to a server in S03 rack. Show the snapshot of your result.

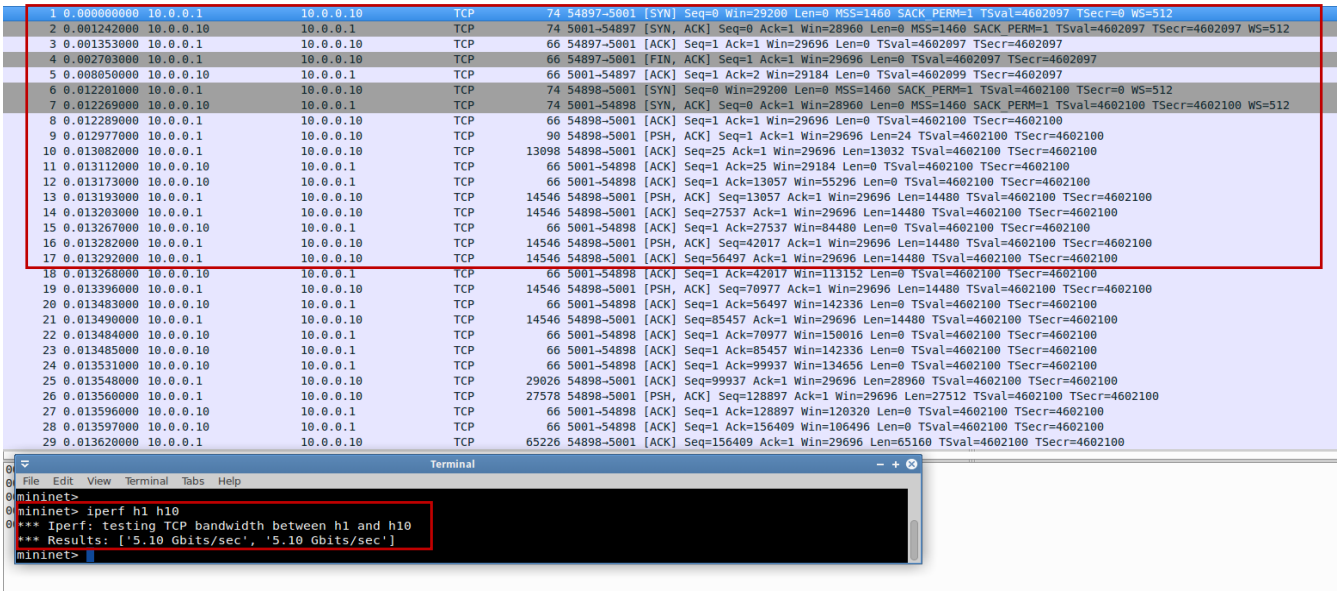
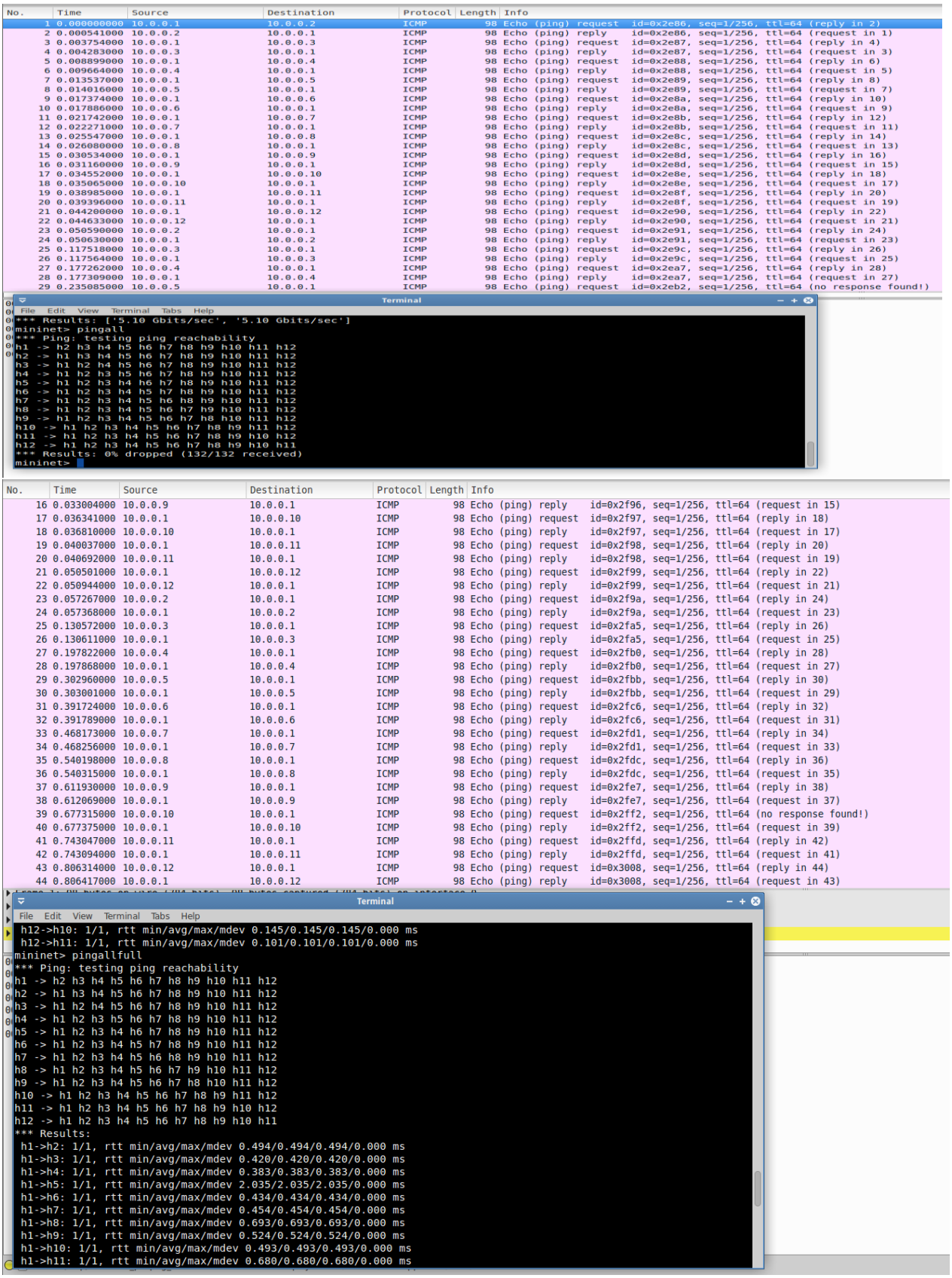


Fig21: Command **h1 wireshark &** to open wireshark and ran **iperf h1 h4** command to check the connection.

- It shows **TCP handshake** to indicate connection is established successfully and packets are received successfully.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x2e86, seq=1/256, ttl=64 (reply in 2)
2	0.000541000	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e86, seq=1/256, ttl=64 (request in 1)
3	0.003754000	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) request id=0x2e87, seq=1/256, ttl=64 (reply in 4)
4	0.004283000	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e87, seq=1/256, ttl=64 (request in 3)
5	0.008899000	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) request id=0x2e88, seq=1/256, ttl=64 (reply in 6)
6	0.009664000	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e88, seq=1/256, ttl=64 (request in 5)
7	0.013537000	10.0.0.1	10.0.0.5	ICMP	98	Echo (ping) request id=0x2e89, seq=1/256, ttl=64 (reply in 8)
8	0.014016000	10.0.0.5	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e89, seq=1/256, ttl=64 (request in 7)
9	0.017374000	10.0.0.1	10.0.0.6	ICMP	98	Echo (ping) request id=0x2e8a, seq=1/256, ttl=64 (reply in 10)
10	0.017886000	10.0.0.6	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e8a, seq=1/256, ttl=64 (request in 9)
11	0.021742000	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request id=0x2e8b, seq=1/256, ttl=64 (reply in 12)
12	0.022271000	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e8b, seq=1/256, ttl=64 (request in 11)
13	0.025547000	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request id=0x2e8c, seq=1/256, ttl=64 (reply in 14)
14	0.026080000	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e8c, seq=1/256, ttl=64 (request in 13)
15	0.030534000	10.0.0.1	10.0.0.9	ICMP	98	Echo (ping) request id=0x2e8d, seq=1/256, ttl=64 (reply in 16)
16	0.031160000	10.0.0.9	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e8d, seq=1/256, ttl=64 (request in 15)
17	0.034552000	10.0.0.1	10.0.0.10	ICMP	98	Echo (ping) request id=0x2e8e, seq=1/256, ttl=64 (reply in 18)
18	0.035065000	10.0.0.10	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e8e, seq=1/256, ttl=64 (request in 17)
19	0.038985000	10.0.0.1	10.0.0.11	ICMP	98	Echo (ping) request id=0x2e8f, seq=1/256, ttl=64 (reply in 20)
20	0.039396000	10.0.0.11	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e8f, seq=1/256, ttl=64 (request in 19)
21	0.044200000	10.0.0.1	10.0.0.12	ICMP	98	Echo (ping) request id=0x2e90, seq=1/256, ttl=64 (reply in 22)
22	0.044633000	10.0.0.12	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2e90, seq=1/256, ttl=64 (request in 21)
23	0.050590000	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x2e91, seq=1/256, ttl=64 (reply in 24)
24	0.050630000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x2e91, seq=1/256, ttl=64 (request in 23)
25	0.117518000	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) request id=0x2e9c, seq=1/256, ttl=64 (reply in 26)
26	0.117564000	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) reply id=0x2e9c, seq=1/256, ttl=64 (request in 25)
27	0.177262000	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) request id=0x2ea7, seq=1/256, ttl=64 (reply in 28)
28	0.177309000	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) reply id=0x2ea7, seq=1/256, ttl=64 (request in 27)
29	0.235085000	10.0.0.5	10.0.0.1	ICMP	98	Echo (ping) request id=0x2eb2, seq=1/256, ttl=64 (no response found!)

Terminal

File Edit View Terminal Tabs Help

```
*** Results: ['5.10 Gbits/sec', '5.10 Gbits/sec']
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
mininet>
```

No.	Time	Source	Destination	Protocol	Length	Info
16	0.033004000	10.0.0.9	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2f96, seq=1/256, ttl=64 (request in 15)
17	0.036341000	10.0.0.1	10.0.0.10	ICMP	98	Echo (ping) request id=0x2f97, seq=1/256, ttl=64 (reply in 18)
18	0.036810000	10.0.0.10	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2f97, seq=1/256, ttl=64 (request in 17)
19	0.040037000	10.0.0.1	10.0.0.11	ICMP	98	Echo (ping) request id=0x2f98, seq=1/256, ttl=64 (reply in 20)
20	0.040692000	10.0.0.11	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2f98, seq=1/256, ttl=64 (request in 19)
21	0.050501000	10.0.0.1	10.0.0.12	ICMP	98	Echo (ping) request id=0x2f99, seq=1/256, ttl=64 (reply in 22)
22	0.050944000	10.0.0.12	10.0.0.1	ICMP	98	Echo (ping) reply id=0x2f99, seq=1/256, ttl=64 (request in 21)
23	0.057267000	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x2f9a, seq=1/256, ttl=64 (reply in 24)
24	0.057368000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x2f9a, seq=1/256, ttl=64 (request in 23)
25	0.130572000	10.0.0.3	10.0.0.1	ICMP	98	Echo (ping) request id=0x2fa5, seq=1/256, ttl=64 (reply in 26)
26	0.130611000	10.0.0.1	10.0.0.3	ICMP	98	Echo (ping) reply id=0x2fa5, seq=1/256, ttl=64 (request in 25)
27	0.197822000	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) request id=0x2fb0, seq=1/256, ttl=64 (reply in 28)
28	0.197868000	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) reply id=0x2fb0, seq=1/256, ttl=64 (request in 27)
29	0.302960000	10.0.0.5	10.0.0.1	ICMP	98	Echo (ping) request id=0x2fbb, seq=1/256, ttl=64 (reply in 30)
30	0.303001000	10.0.0.1	10.0.0.5	ICMP	98	Echo (ping) reply id=0x2fbb, seq=1/256, ttl=64 (request in 29)
31	0.391724000	10.0.0.6	10.0.0.1	ICMP	98	Echo (ping) request id=0x2fc6, seq=1/256, ttl=64 (reply in 32)
32	0.391789000	10.0.0.1	10.0.0.6	ICMP	98	Echo (ping) reply id=0x2fc6, seq=1/256, ttl=64 (request in 31)
33	0.468173000	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) request id=0x2fd1, seq=1/256, ttl=64 (reply in 34)
34	0.468256000	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) reply id=0x2fd1, seq=1/256, ttl=64 (request in 33)
35	0.540198000	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) request id=0x2fdc, seq=1/256, ttl=64 (reply in 36)
36	0.540315000	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) reply id=0x2fdc, seq=1/256, ttl=64 (request in 35)
37	0.611930000	10.0.0.9	10.0.0.1	ICMP	98	Echo (ping) request id=0x2fe7, seq=1/256, ttl=64 (reply in 38)
38	0.612069000	10.0.0.1	10.0.0.9	ICMP	98	Echo (ping) reply id=0x2fe7, seq=1/256, ttl=64 (request in 37)
39	0.677315000	10.0.0.10	10.0.0.1	ICMP	98	Echo (ping) request id=0x2ff2, seq=1/256, ttl=64 (no response found!)
40	0.677375000	10.0.0.1	10.0.0.10	ICMP	98	Echo (ping) reply id=0x2ff2, seq=1/256, ttl=64 (request in 39)
41	0.743047000	10.0.0.11	10.0.0.1	ICMP	98	Echo (ping) request id=0x2ffd, seq=1/256, ttl=64 (reply in 42)
42	0.743094000	10.0.0.1	10.0.0.11	ICMP	98	Echo (ping) reply id=0x2ffd, seq=1/256, ttl=64 (request in 41)
43	0.806314000	10.0.0.12	10.0.0.1	ICMP	98	Echo (ping) request id=0x3008, seq=1/256, ttl=64 (reply in 44)
44	0.806417000	10.0.0.1	10.0.0.12	ICMP	98	Echo (ping) reply id=0x3008, seq=1/256, ttl=64 (request in 43)

Terminal

File Edit View Terminal Tabs Help

```
 h12->h10: 1/1, rtt min/avg/max/mdev 0.145/0.145/0.145/0.000 ms
 h12->h11: 1/1, rtt min/avg/max/mdev 0.101/0.101/0.101/0.000 ms
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.494/0.494/0.494/0.000 ms
 h1->h3: 1/1, rtt min/avg/max/mdev 0.420/0.420/0.420/0.000 ms
 h1->h4: 1/1, rtt min/avg/max/mdev 0.383/0.383/0.383/0.000 ms
 h1->h5: 1/1, rtt min/avg/max/mdev 2.035/2.035/2.035/0.000 ms
 h1->h6: 1/1, rtt min/avg/max/mdev 0.434/0.434/0.434/0.000 ms
 h1->h7: 1/1, rtt min/avg/max/mdev 0.454/0.454/0.454/0.000 ms
 h1->h8: 1/1, rtt min/avg/max/mdev 0.693/0.693/0.693/0.000 ms
 h1->h9: 1/1, rtt min/avg/max/mdev 0.524/0.524/0.524/0.000 ms
 h1->h10: 1/1, rtt min/avg/max/mdev 0.493/0.493/0.493/0.000 ms
 h1->h11: 1/1, rtt min/avg/max/mdev 0.680/0.680/0.680/0.000 ms
```

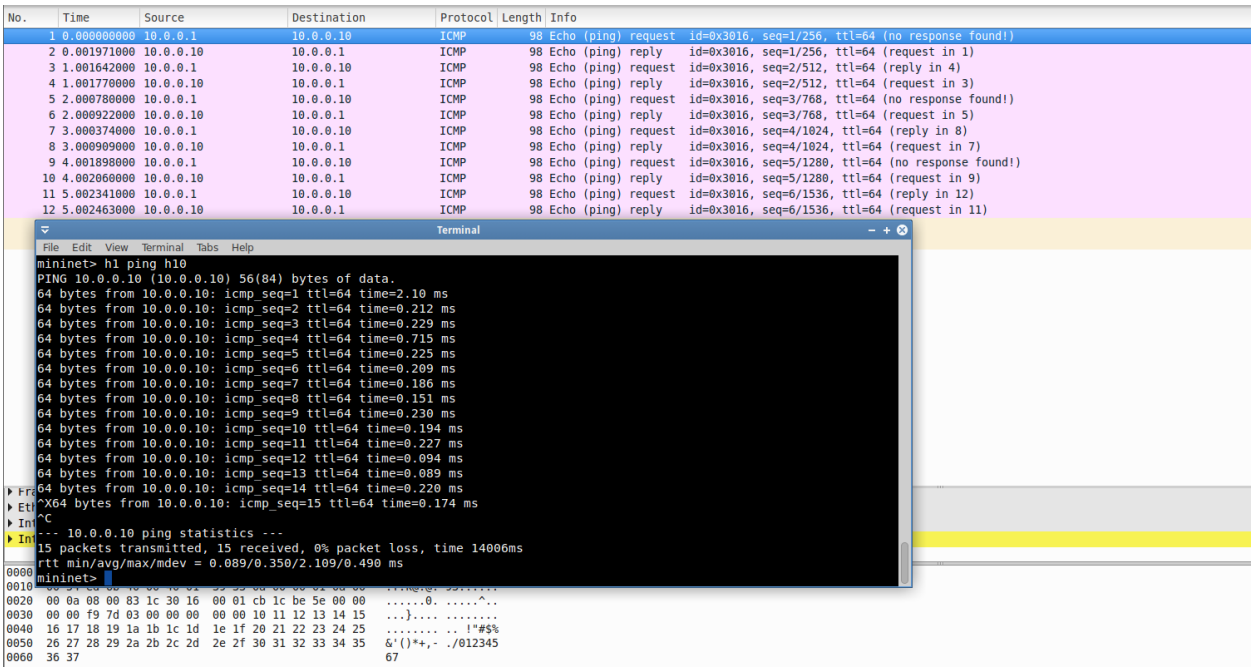


Fig22: Above three screenshots showing Wireshark for **pingall, pingallfull, and h1 ping h10** command showing **ICMP messages**

----------xx----------