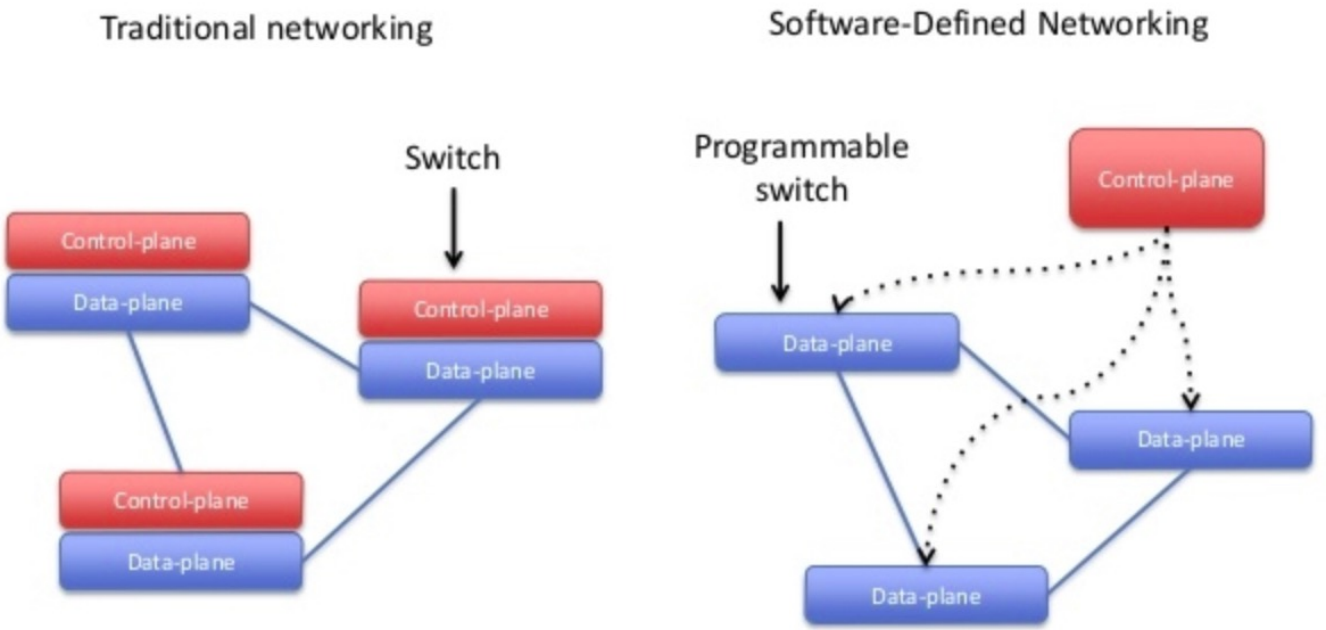
Software Defined Network
SDN

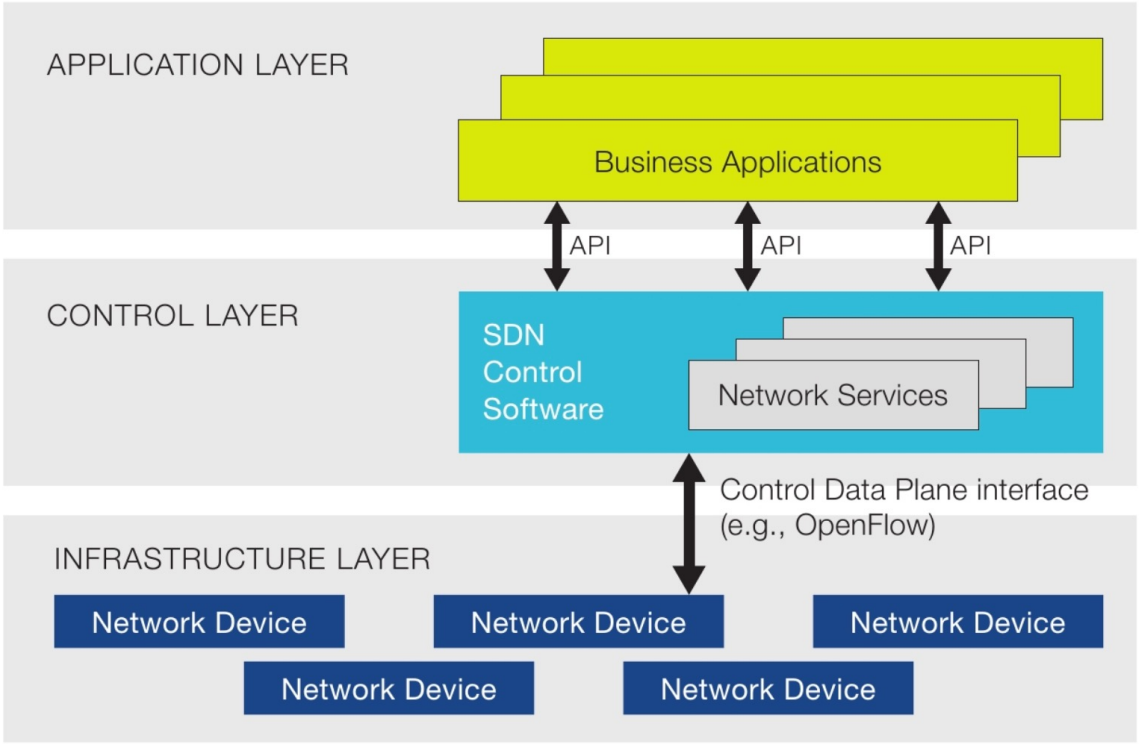
Ch 2

SDN Lab Setup

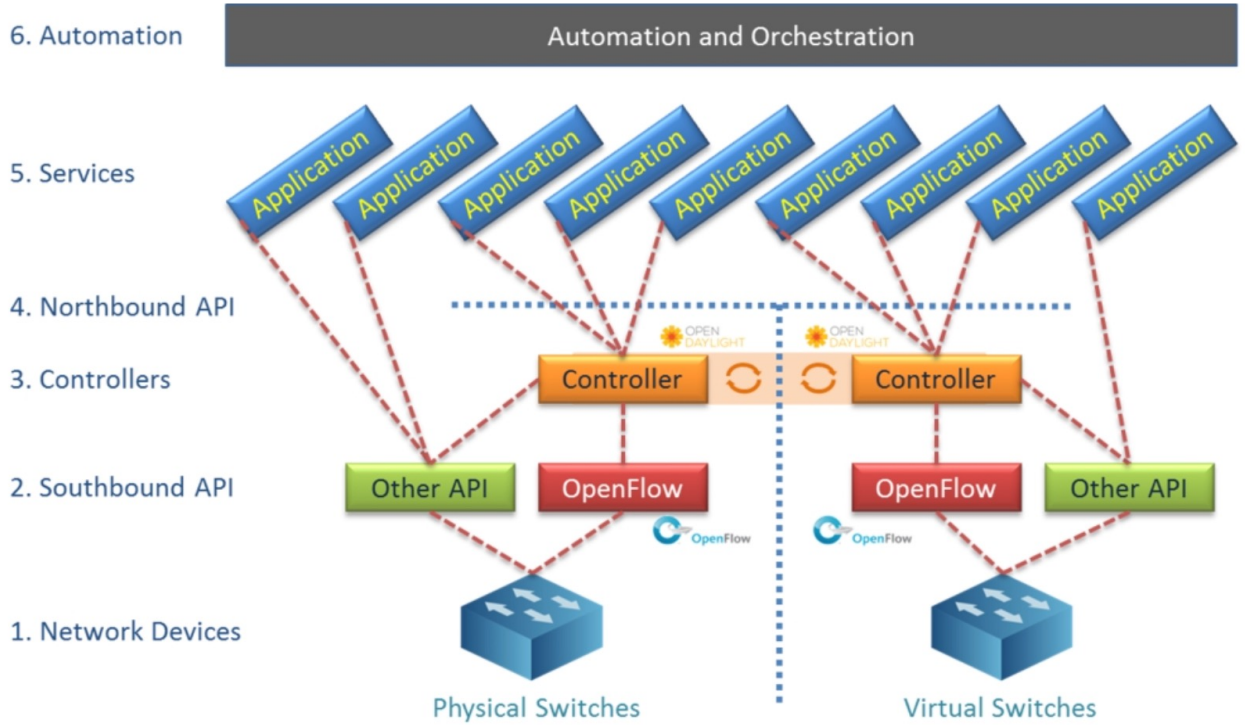
SDN Introduction



SDN Architecture



SDN Architecture



SDN Test bed Installation

- Option 1
  - Prebuilt VM Image (OVA Format), Size 3.3GB can be download from this below link,
  - https://drive.google.com/file/d/1_eC9O6EW7fh5YUp_LJMQ3_gIyFXBIc8k/view?usp=sharing
  - **username : test password : test**

SDN Test bed Installation

- **Option2: Fresh Installation**
  - You can download the ISO installer from the below link,
  - https://www.ubuntu.com/download/desktop

If you want to setup the testbed by yourself follow this

Requirements:

OS: Ubuntu 20.04

CPU: 2 Cores +

RAM: 4GB +

HDD: 15GB+

SDN Test bed Installation

```
sudo apt update
sudo apt install python3 python3-pip xterm iperf hping3 net-tools wireshark apache2-utils curl
sudo apt install mininet
sudo pip3 install ryu
sudo pip3 install mininet
sudo cp /usr/bin/python3 /usr/bin/python
ryu-manager --version
sudo mn --version
```

SDN Test bed Installation

To check the python version:

> python3 --version or python --version

```
Python 3.8.5
```

To verify :

> ovs-vsctl --version

```
ovs-vsctl (Open vSwitch) 2.13.1
DB Schema 8.2.0
```

SDN Test bed Installation

Testing

Open 4 Terminals:

1. In Terminal1,

```
sudo wireshark
```

And start the capture for "loopback" or "any" interface.

2. In Terminal2,

```
ryu-manager ryu.app.simple_switch_13
```

3. In Terminal3,

```
sudo mn --controller=remote,ip=127.0.0.1 --mac --switch=ovsk,protocols=OpenFlow13 --top
```

you will get the mininet prompt. In mininet prompt, type pingall comamnd

```
pingall
```

ข้อ 3 In Terminal 3, พิมพ์ตามนี้ เนื่องจากด้านบนคำสั่งไม่ครบ

sudo mn --controller=remote,ip=127.0.0.1 --mac --switch=ovsk,protocols=OpenFlow13 --topo=single,4

SDN Test bed Installation

- เมื่อพิมพ์เสร็จ จะได้ดังภาพ

Logs:

```
suresh@suresh-vm:~$ sudo mn --controller=remote,ip=127.0.0.1 --mac --switch=ovsk,proto
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

SDN Test bed Installation

4. In Terminal 4,

```
sudo ovs-vsctl show

sudo ovs-ofctl -O OpenFlow13 dump-flows s1
```

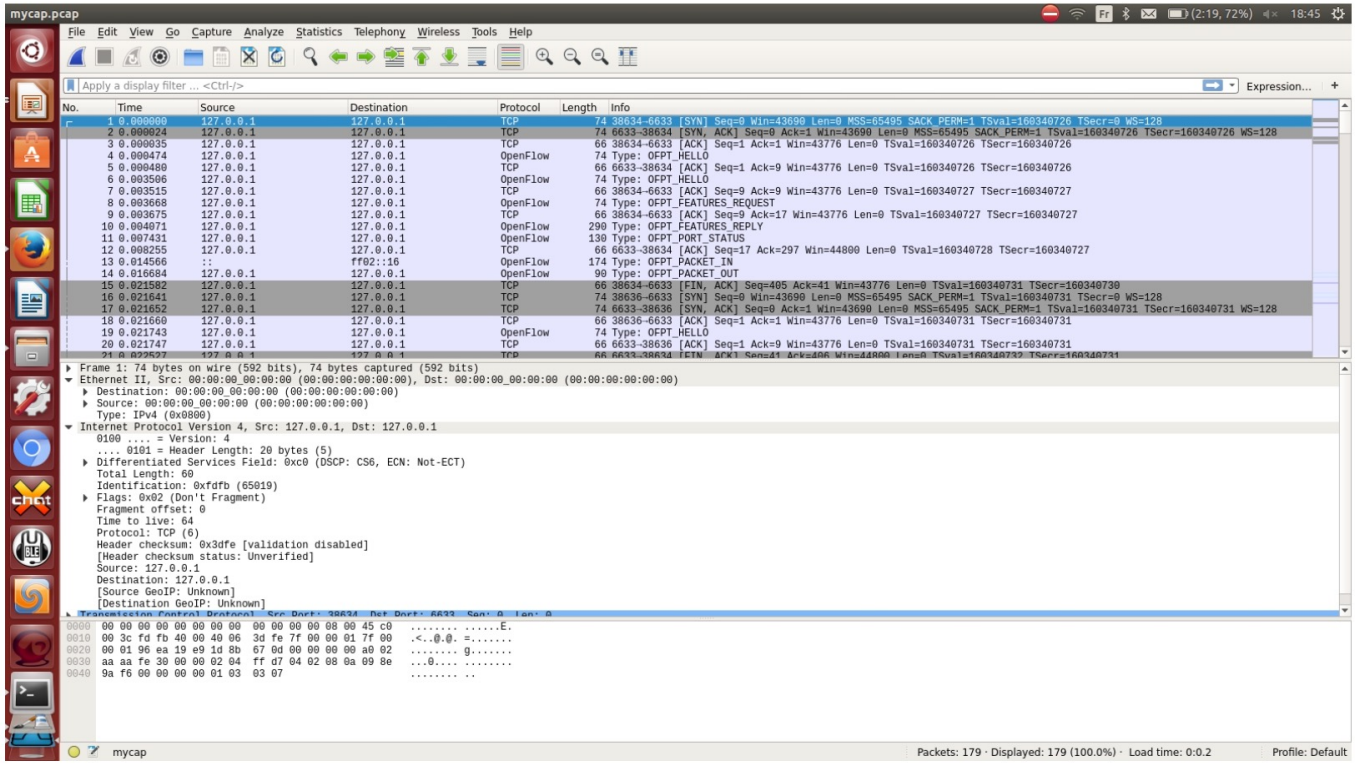
```
suresh@suresh-vm:~$ sudo ovs-vsctl show
[sudo] password for suresh:
a315e8b4-dd3f-42f6-b84a-e967e02660a4
    Bridge "s1"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        Controller "ptcp:6654"
        fail_mode: secure
        Port "s1-eth4"
            Interface "s1-eth4"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth3"
            Interface "s1-eth3"
        Port "s1-eth2"
            Interface "s1-eth2"
    ovs_version: "2.13.1"
suresh@suresh-vm:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=115.430s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="
 cookie=0x0, duration=115.421s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="
 cookie=0x0, duration=115.410s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="
 cookie=0x0, duration=115.404s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="
 cookie=0x0, duration=115.391s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="
 cookie=0x0, duration=115.380s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="
 cookie=0x0, duration=115.370s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="
 cookie=0x0, duration=115.368s, table=0, n_packets=2, n_bytes=140, priority=1,in_port="
 cookie=0x0, duration=115.361s, table=0, n_packets=3, n_bytes=238, priority=1,in_port="
```

SDN Test bed Installation

5. check the openflow messages in wireshark

Stop the Wireshark capture,

In the filter type "openflow_v4" to see the OPENFLOW Messages.



Networking Concept

TCP/IP Layers

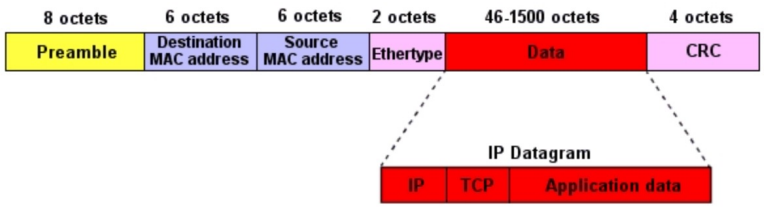
TCP/IP	OSI Model	Protocols
Application Layer	Application Layer	DNS, DHCP, FTP, HTTPS, IMAP, LDAP, NTP,POP3, RTP, RTSP, SSH, SIP, SMTP, SNMP, Telnet, TFTP
	Presentation Layer	JPEG, MIDI, MPEG, PICT, TIFF
	Session Layer	NetBIOS, NFS, PAP, SCP, SQL, ZIP
Transport Layer	Transport Layer	TCP, UDP
Internet Layer	Network Layer	ICMP, IGMP, IPsec, IPv4, IPv6, IPX, RIP
Link Layer	Data Link Layer	ARP, ATM, CDP, FDDI, Frame Relay, HDLC, MPLS, PPP, STP, Token Ring
	Physical Layer	Bluetooth, Ethernet, DSL, ISDN, 802.11 Wi-Fi

Ethernet (Layer 2)

Ethernet (Layer2) Concepts

- Ethernet is Layer2 Protocol,
- Majorly used as LAN connectivity
- Interface (eth0, eth1, ….)
- MAC Address is L2 Address, Associated with the interface, its 8 bytes (HWaddr 02:42:ac:11:00:02)

Ethernet Frame

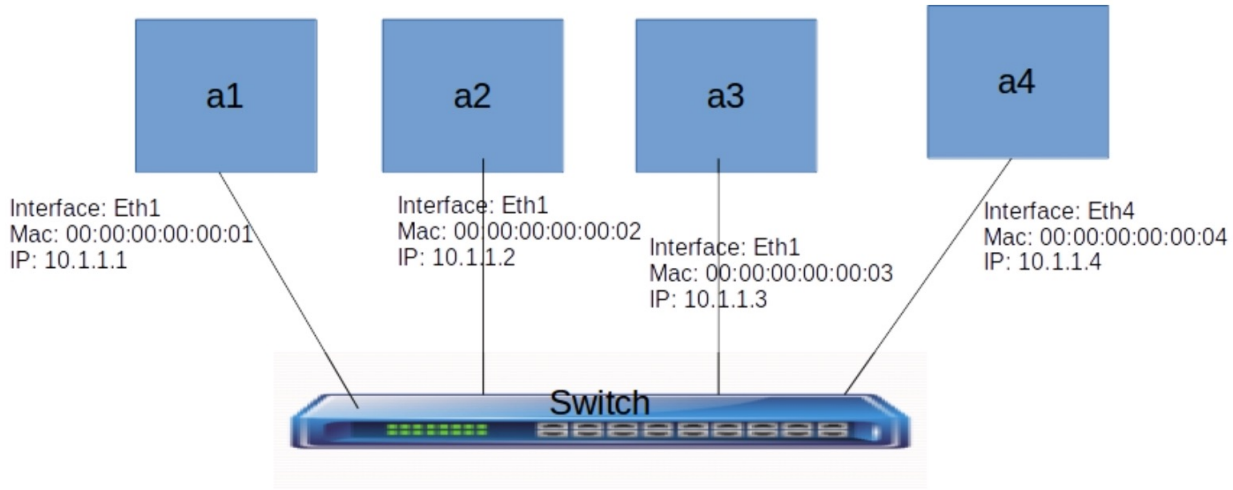



```
Destination Mac address :  Target machine MAC address
Source Mac address:         Source machine MAC address
Ethertype:   Next layer protocol
                  0X0800 -  IP
                  0X0806 -  ARP
                  0X86DD  - IPv6
```

Neighbour Discovery

Neighbour Discovery

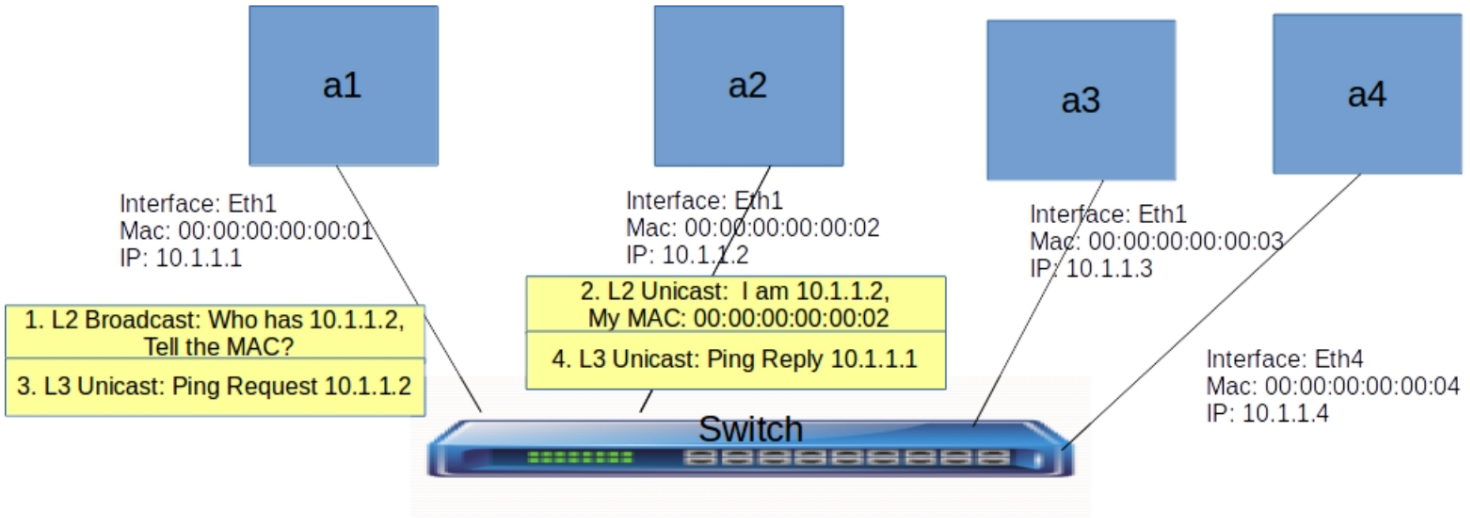
The MAC address is how machines on a subnet communicate. When machine A sends packets to another machine on its subnet, it sends it using the MAC address. When sending a packet to a machine on the public Internet, the packet is sent to the MAC address of the router interface that is the default gateway. IP addresses are used to figure out the MAC address to send to using ARP.



ARP Basics

ARP stands for Address Resolution Protocol. When you try to ping an IP address on your local network, say 192.168.1.1, your system has to turn the IP address 192.168.1.1 into a MAC address. This involves using ARP to resolve the address, hence its name.

Systems keep an ARP look-up table where they store information about what IP addresses are associated with what MAC addresses.



ARP Basics

1. When trying to send a packet to an IP address, the system will first consult this table to see if it already knows the MAC address. If there is a value cached, ARP is not used.

2. If the IP address is not found in the ARP table, the system will then send a broadcast packet to the network using the ARP protocol to ask "who has 192.168.1.1".

3. Because it is a broadcast packet, it is sent to a special MAC address that causes all machines on the network to receive it.

4. Any machine with the requested IP address will reply with an ARP packet that says "I am 192.168.1.1", and this includes the MAC address which can receive packets for that IP.

ARP Basics

ARP Table

List ARP Table

```
arp -a
```

Delete ARP entry

```
arp -d 10.1.1.2
```

ARP Basics

ARP Demo

1. Start the mininet topology (linear topology , not connected to SDN Controller)

```
sudo python traditional_switch.py
```

2. in mininet console, run 'links' command to identify the link names

```
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
h3-eth0<->s1-eth3 (OK OK)
h4-eth0<->s1-eth4 (OK OK)
mininet>
```

   So if we want to capture wireshark traces for h2, we have to capture the s1-eth2 interface.

3. capture wireshark traces for h2

4. in mininet console run 'h1 ping h2'

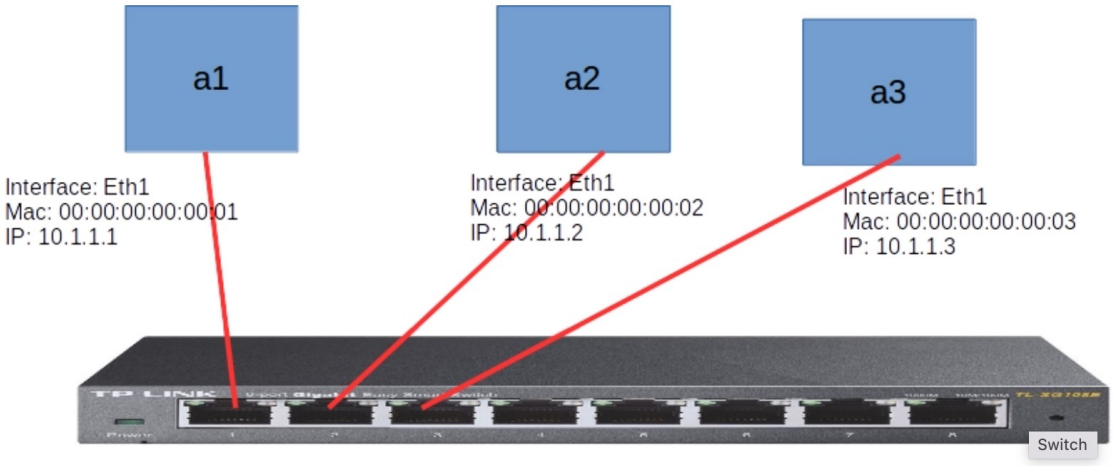
5. Analyze the ARP Packets from Wireshark traces

6. check the ARP entries in h1 and h2

```
mininet>h1 arp -a
```

Traditional L2 Switch

A network switch (MAC bridge) is a computer networking device that connects devices together on a computer network by using packet switching to receive, process, and forward data to the destination device.



Connections:
PORT1 – Connected to a1
PORT2 – Connected to a2
PORT3 - Connected to a3

MAC Table:
PORT1 – 00:00:00:00:00:01
PORT2 – 00:00:00:00:00:02
PORT3 - 00:00:00:00:00:03

Traditional L2 Switch

Control Plane

Learns the MAC Address from the incoming packet and populate the MAC Table

Data Plane

- Receives the Packet
- Read the Destination MAC from the Packet
- Look up the MAC Table for the destination Port
- Forward the Packet to the destination Port

Openvswitch is softswitch, which works as normal switch(traditional) as well as SDN(openflow)switch

Demo

Demo

Same steps as Neighbour discovery Demo

Lets run OVS commands to check the switch

```
sudo ovs-vsctl   --version
sudo ovs-vsctl show
sudo ovs-appctl fdb/show s1
sudo ovs-dpctl dump-flows
```

How it works

1. Traditional Switch have built in Control Plane + Data Plane.
2. Mac Table(a.k.a Forwarding table) is Empty when the switch starts.
3. Control Plane updates the Mac Table with the MAC and the PORT Number. This information is extracted from incoming Packet.
4. Control Plane keeps on building/updating the Mac Table.
5. When the packet arrives, Switch Data plane looks the Mac table, if the destination MAC matches in the Mac table, it forwards the packet to the respective Port.

SDN Switch(Openflow Switch)

- Run the Linear Mininet Topology,
  - sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch=ovsk,protocols=OpenFlow13 --topo=linear,4

2. Start Wireshark Capture

3. RYU L3 Application

```
ryu-manager ryu.app.simple_switch_13
```

Copy

4. In the mininet cli, ping h1 to h2.

```
mininet>h1 ping h2
```

5. Check the OpenFlow flows

```
sudo ovs-ofctl -O OpenFlow13 dump-flows s1
```

SDN Switch(Openflow Switch)

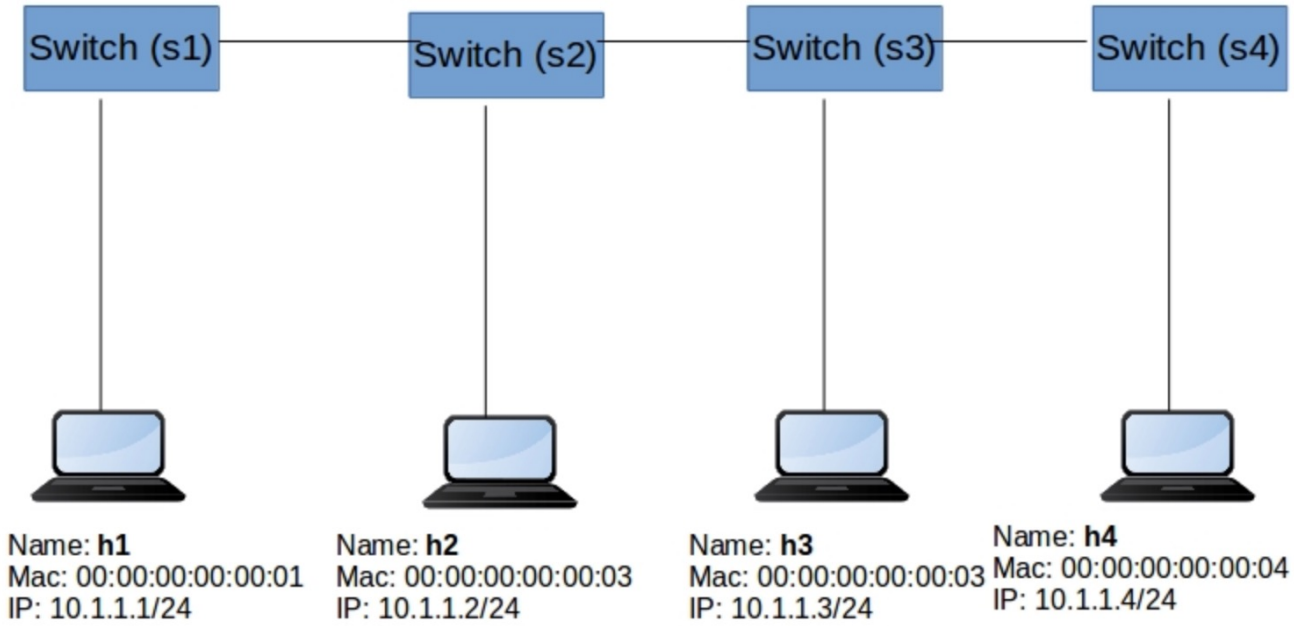
How it works

1. Switch is configured with SDN Controller IP and Openflow protocol version.
2. Switch establishes the communication with SDN Controller.
3. SDN Controller installs the default Openflow rule (TABLE MISS ENTRY) in the switch Flow table.
4. TABLE MISS Entry openflow rule matches with all the packets and send it to the CONTROLLER. The priority is lowest in the table(0)
5. When the Host Data Packet arrives in the Switch, It will be matched with TABLE MISS ENTRY , and the packet will be sent it to Controller (PACKET IN Message)
6. Controller receives the packet, and build the Switch Logic with the packets.
7. Controller adds the OPENFLOW flows to the switch.
8. Now, Switch data path is built with flows. So next time, when the Packet arrives it will be matched with the Flow table and forward the packet to respective port.

Mininet

5. Linear Topology

linear topology (where each switch has one host, and all switches connect in a line)



sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch=ovsk,protocols=OpenFlow13 --topo=linear,4

Mininet

Basic Operations

1. To check the mininet version

```
mn --version
```

2. To clean up the existing ovs bridges and namespaces

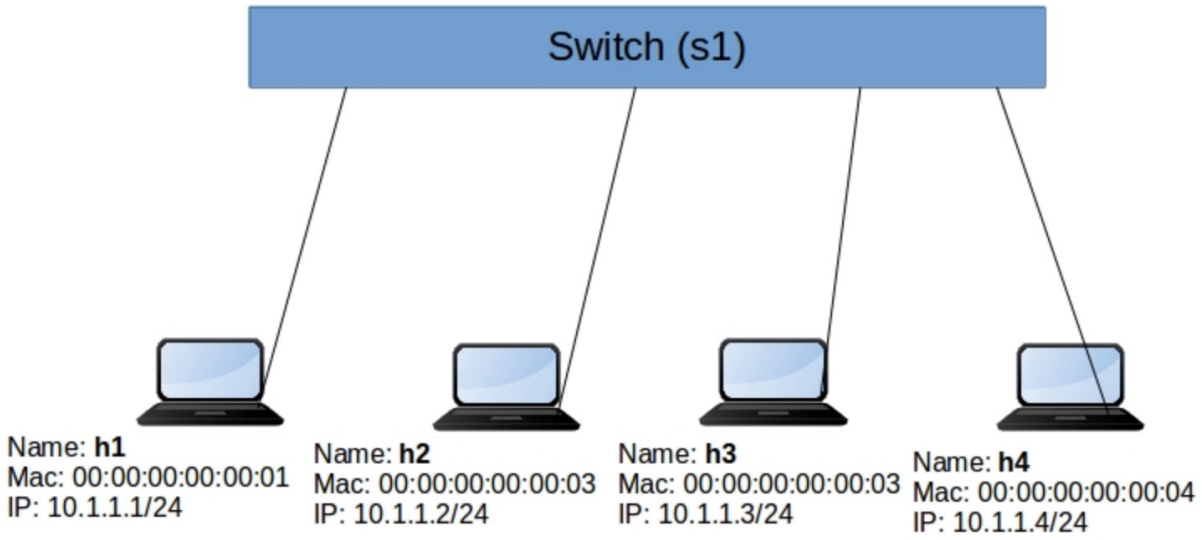
Note: sometime we mistakenly closed the mininet shell, or mininet crashed. But the topology components will continue to exists. To clean such stuff, cleanup command is used.

```
mn -c
```

Mininet

3. Our First Topology (Single)

Topology with Single Switch and 4 Nodes.



Mininet

RYU SDN Controller

```
ryu-manager ryu.app.simple_switch_13
```

Mininet Topology

- sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch=ovsk,protocols=OpenFlow13 --topo=single,4

Mininet

options	Description
--controller	type of controller local/remote and remote controller ip.
--mac	mac address starts with 00:00:00:00:00:01
-i	IP Subnets for the Topology
--switch	Switch type (ovsk - openvswitch kernel module), and openflow version.
--topo	topology type(linear,minimal,reversed,single,torus,tree) and params.

Once you are done, make sure "exit" the mininet shell,

```
exit
```

Mininet

Example:

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 4 links
....
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 2.604 seconds
suresh@suresh-vm:~$
```

Copy

Mininet

4. Mininet Basic Shell Commands

Informative commands

```
help
dump
net
links
```

Action commands

```
pingall
```

Mininet

Execute the commands in HOST/Node:

Option1:

we can login to each host using 'xterm' command

> xterm h1

```
mininet>xterm h1
```

It will open a xterm terminal for the host. Now we can execute the command inside that terminal

Option2:

we can directly execute from the mininet shell.

```
mininet><hostname> command
```

Copy

Mininet

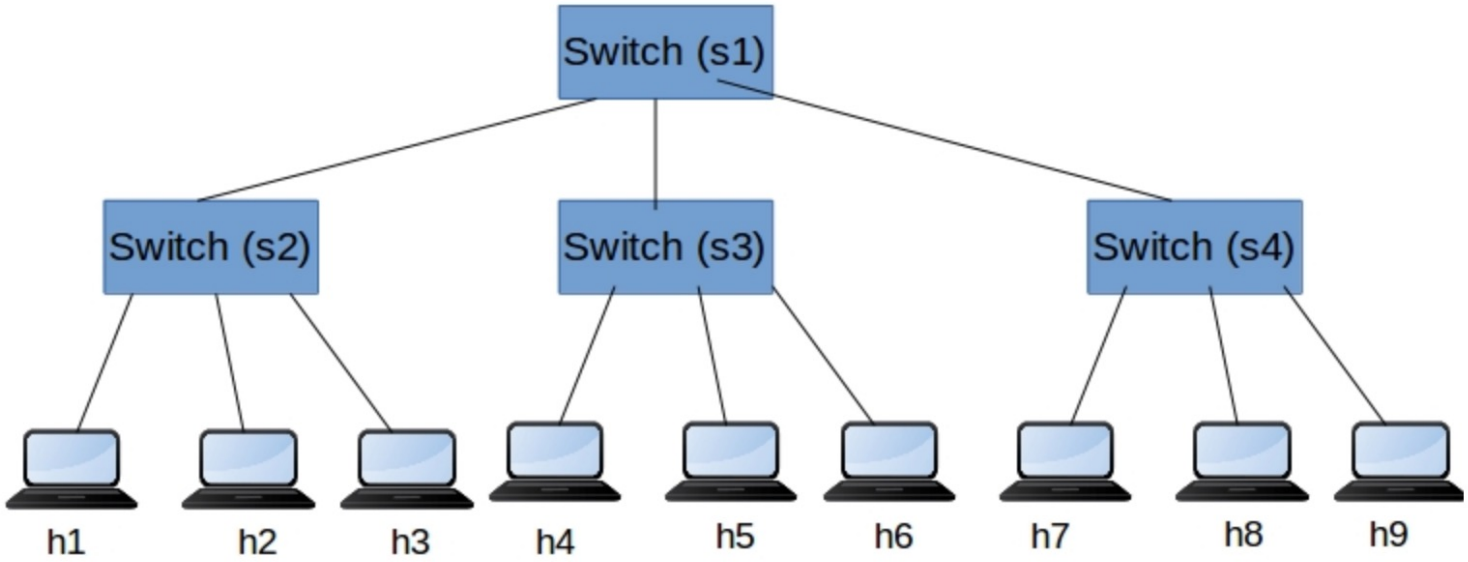
Example:

```
mininet>h1 ifconfig
mininet>h1 ping h2
mininet>h1 ip route
```

we can use either method to run the Traffic tests or executing the commands.

Mininet

6. Tree Topology



sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --topo=tree,depth=2,fanout=3

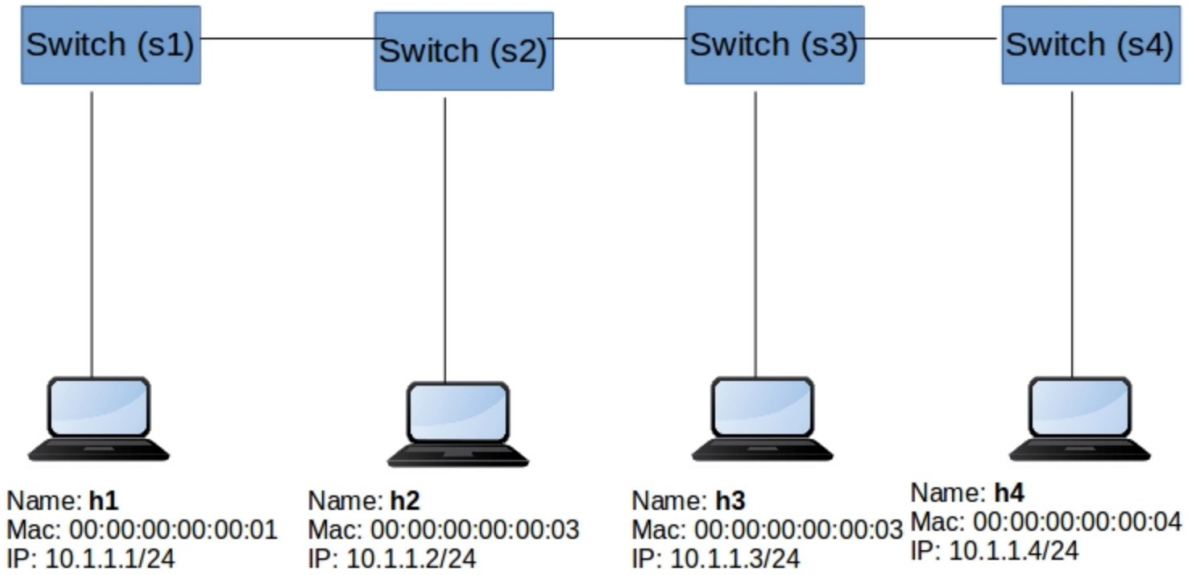
fanout : each switch is connected to these many childs depth : depth of the tree

Mininet

Running Traffic Tests

1. TCP/UDP Traffic Tests

a. Setup the Topology with xterms options (open terminal for each Node)



sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch=ovsk,protocols=OpenFlow13 --topo=linear,4 -x

Mininet

b. TCP Traffic Test between h1 to h4

Run IPERF TCP Server in h4

```
iperf -s
```

-s means server mode

RUN IPERF TCP Client in h1

```
iperf -c 10.1.1.4 -i 10 -t 30
iperf -c 10.1.1.4 -i 10 -b 10m -t 30
iperf -c 10.1.1.4 -i 10 -P 10 -t 30
```

Mininet

-c means client mode.

-i means reporting interval

-t means test duration in seconds

-b means bandwidth 10m means 10Mbps

-P means parallel connections

c. UDP Traffic Test between h1 to h4

Mininet

Run IPERF UDP Server in h4

```
iperf -u -s
```

-u means udp

RUN IPERF UDP Client in h1

```
iperf -u -c 10.1.1.4 -b 10m -i 10 -t 30
iperf -u -c 10.1.1.4 -b 10m -i 10 -P 10 -t 30
```

-b means bandwidth 10m means 10Mbps

Mininet

HTTP Traffic Tests

Run Python Simple Web Server in h4

```
python -m SimpleHTTPServer 80
```

Copy

From H1, Access the Web server

```
curl http://10.1.1.4/
```

curl utility used as web client to access the web server.

Mininet

If we want to simulate the 1000s users accessing the web server on the same time (load), we can use ab(apache bench) tool

```
ab -n 500 -c 50 http://10.1.1.4/
```

-c 50 means parallel request per second (50 Requests per second)

-n 500 means total request for this test (500 requests)

Apache bench tool have lot more options , More details can be found from the below link

https://httpd.apache.org/docs/2.4/programs/ab.html

Writing Custom Topology in Mininet

How to write Custom Topology in Mininet

Steps are below.

1. Import the python required libraries

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

2. Write the Topology definition class

```
class SingleSwitchTopo(Topo):
   def build(self):
       s1 = self.addSwitch('s1')

       h1 = self.addHost('h1')
       h2 = self.addHost('h2')
       h3 = self.addHost('h3')
       h4 = self.addHost('h4')

       self.addLink(h1, s1)
       self.addLink(h2, s1)
       self.addLink(h3, s1)
       self.addLink(h4, s1)
```

Writing Custom Topology in Mininet

Important Topology definition APIs:

```
addSwitch
addHost
addLink
```

3. Run the Topology as below,

- Create the Topology object
- Create the Mininet with Topology object
- Start the Mininet

```
if __name__ == '__main__':
    topo = SingleSwitchTopo()
    c1 = RemoteController('c1', ip='127.0.0.1')
    net = Mininet(topo=topo, controller=c1)
    net.start()
```

Writing Custom Topology in Mininet

How to run

1. start the RYU SDN Controller

```
ryu-manager ryu.app.simple_switch_13
```

2. Run the Mininet topology file

```
sudo python <topology file name>
```

3. Perform your tests/operation etc.