A
REPORT ON
SETUP OF SDN NETWORK

Guided By:

Roberto Rojas-Cessa

Implemented By:

Ravinder Kumar

31462074

List of Content:

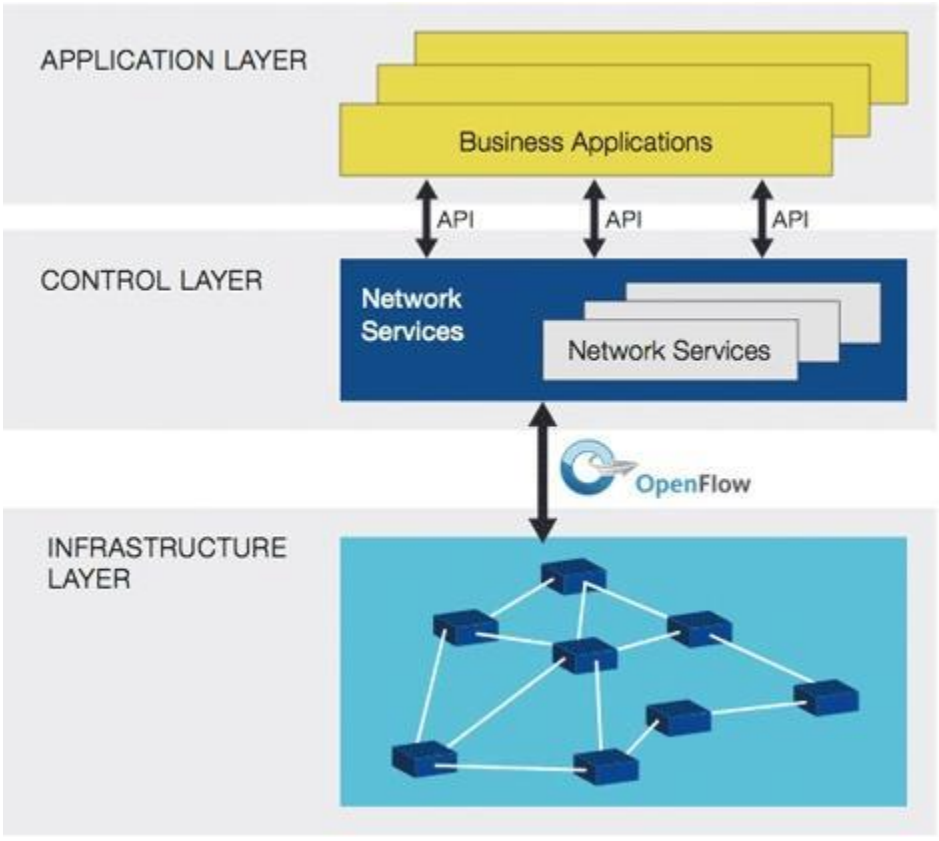
What is SDN: The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices. Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. We are using OpenFlow protocol for building the SDN solution. There are some critical area which can make a difference in technology for organizations:

1. Network Programmability
2. Logically centralize intelligence and control
3. Abstraction of the network
4. Openness

How SDN is different from Traditional networks: The biggest difference between a traditional network and SDN is that the latter is a **software-based network**. Traditional networks rely on physical infrastructure such as switches and routers to make connections and run properly. In contrast, a software-based network allows the user to control the allocation of resources at a virtual level through the control plane. Rather than interacting with physical infrastructure, the user is interacting with software to provision new devices.

In a traditional network, the data plane tells your data where it needs to go. Likewise, under the traditional network model, the control plane is located within a switch or router. The location of the control plane is particularly inconvenient because administrators don't have easy access to dictate traffic flow (especially when compared to an SDN). Under an SDN the control plane becomes software-based and can be accessed through a connected device. This means that an administrator can control the flow of traffic from a centralized user interface with greater scrutiny. This gives users more control over how their network functions. You can also change your network's configuration settings from the comfort of a centralized hub. Managing configurations in this way is particularly beneficial with regards to segmentation of the network as the user can process many configurations promptly.

How SDN works: At the most basic level an Ethernet Switch consists of two main components, the switching hardware and some software, or firmware as it is often called. When a packet arrives at one of the ports the hardware asks the firmware what it should do with the packet. The firmware then looks up the destination address on a list (called a MAC table) which contains all the devices that are currently connected to the switch and then tells the hardware which port to send the packet out from. proposed the concept of separating the brains of the switch (software), called the Control Plane from the switching hardware, or Data Plane. By placing this Control Plane onto a totally separate server instead of inside the actual switch you can now modify that software to add additional functionality. This new Control Plane software became what is now known as a SDN Controller and the additional functionality is implemented as SDN application that "plug" into the controller.



The above diagram shows the separation between different layers of SDN deployment.

After the explanation of basic SDN network we are moving towards the setup of SDN network using required software's such as virtualizations or controllers.

Setup of SDN Network using Physical Computers:

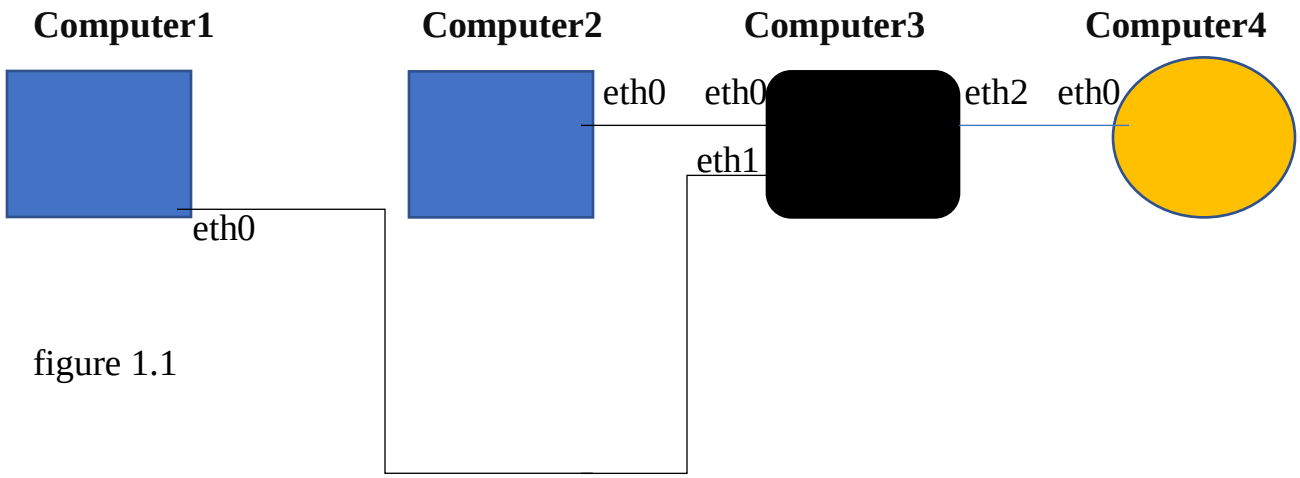


figure 1.1

I am taking computer as C:

We have four computers and going to setup a network using sdn. First two computers are taking as client1 and client2. Computer three is openVswitch and computer4 is ryu controller.

Let's see how to install openVswitch in computer3:

First update and upgrade the system:

1. sudo apt-get update & upgrade

2 sudo apt-get install openvswitch-switch

Install the dependencies because openVswitch requires some dependencies installed first. Run this command in the terminal

```
sudo apt install build-essential fakeroot graphviz autoconf automake bzip2 debhelper dh-autoreconf libssl-dev libtool openssl procps python-all python-qt4 python-twisted-conch python-zopeinterface module-assistant dkms make libc6-dev python-argparse uuid-runtime netbase kmod python-twisted-web iproute2 ipsec-tools openvswitch-switch racoon
```

These are all dependencies which we can download using one command.

Now we are taking computer4 as controller and let's install ryu controller in computer4:

Steps:

1. pip install ryu

We are installing ryu controller from source code:
2. git clone git://github.com/osrg/ryu.git
3. cd ryu
4. pip install . (here . means install in current directory)

Some functions of ryu requires extra packages so we are installing those packages also:
5. pip install -r tools/optional-requires

If you got some error messages at the installation stage, please confirm dependencies for building the required Python packages
6. apt install gcc python-dev libffi-dev libssl-dev libxml2-dev libxslt1-dev zlib1g-dev

Now we have all computers ready with installation and we are setting up the sdn, before setting up network let me describe all ports and how to make connection between all computers.

Computer1(client1): ports: eth0

Computer2(client2): ports: eth0

Computer3(switch): ports: eth0, eth1,eth2

Computer4(controller): ports: eth0

Connection:

Steps: (we have to make connection as shown in figure 1.1)

1. First connect computer3(eth2) to computer4(eth0) using ethernet cable.

2. Connect computer2(eth0) to computer3(eth0). This is the connection between clinet2 and openVswitch.

3. Connect computer1(eth0) to computer3(eth1).

Our connection are completed, I am going to explain which commands we have to run in each computer.

1. Now, We have to make tcp connection between computer3(switch) and computer4(controller):

Configure the ip address in computer3(eth2):

sudo ifconfig eth2 10.0.1.2 netmask 255.255.255.0 up

after that configure ip address in computer4(eth0):

sudo ifconfig eth0 10.0.1.3 netmask 255.255.255.0 up

Note: both computers ip addresses should be in same network.

Connect c3(eth2) and c4(eth0) and ping each other. If pinging works means connection is fine. If connect: host is unreachable shows, then remove all connection setting from GUI and make manual entry.

Now, run these commands in computer3:

i. Create a bridge using command

ovs-vsctl add-br br0(bridge_name)

ovs-vsctl means openVswitch controlling commands, the above command will create bridge.

ii. ovs-vsctl set bridge br0 protocols=OpenFlow10

using this command bridge br0 will follow the openflow protocol version 1, we have need to set this because many versions are available for openVswitch.

Now start controller in computer 4, we are already install ryu controller in our system:

I. cd ryu;
II. run the command to start controller

PYTHONPATH = ./bin/ryu-manager /ryu/app/simple_switch.py

Run the following commands in computer3:

ovs-vsctl set controller br0 tcp:10.0.1.16:6653(default_port)

this command set the controller with switch and forward the rules to switch for operation.

Run following command in computer4:

```
netstat -an | grep 6653
```

It will tcp connection listen and connection is established or not.

Our tcp connection between computer3 and computer4 are done. After the tcp connection we have to add our ports in bridge which we have created in computer3.

Run the following commands to add the ports in bridge br0:

```
ovs-vsctl add-port br0 eth0
ovs-vsctl add-port bro eth1
```

eth0,eth1 are ports of computer3(switch)

Configure the ip addresses in computer1(eth0) and computer2(eth0):

Configure the ip address in computer1(eth0):

sudo ifconfig eth0 10.0.1.5 netmask 255.255.255.0 up

after that configure ip address in computer2(eth0):

sudo ifconfig eth0 10.0.1.6 netmask 255.255.255.0 up

Note: Make sure all interfaces are in same network

Now run following command in computer1:

ping 10.0.1.6

Check the rule flows in computer3(switch) using command:

ovs-ofctl dump-flows br0

This command tells the flow-table entries for the bridge.

Note: We can ping the system using name instead of ip address, like we have to ping computer2 from computer1. For this open terminal and run following command:

Vim /etc/hosts

And make entry of ip address of computer2 as name you want. For instance,

Client2 10.0.1.6

And save that file, after that we can ping using ping client2 instead of ping 10.0.1.6. because dns will resolve the ip address for name client2