

SOFTWARE DEFINED NETWORKING

REPORT



*MEMBER*

1. *HỒ VIẾT TRUNG – ITITIU17098*
2. *LÊ VƯƠNG DUY – ITITIU17096*
3. *NGUYỄN ANH DŨNG – ITITIU17063*
4. *NGUYỄN TRẦN CHÍ HIẾU – ITITIU17110*
5. *LÊ TUẤN KIỆT – ITITIU17040*

December 1, 2020

international university – vietnam national university

HO CHI MINH CITY, VIETNAM

1. **THEORY:**
2. ***Introduction:***
   1. *Overview of traditional network “planes”:*

Normally, network is visualized as a cloud or number of straight lines connecting nodes (representing network devices. E.g: routers, switches,..). In reality, a network has three dimensions, often called as “planes”: the Data plane, the Control plane and the Management plane. Together, these three planes are responsible for handling and forwarding our data packets from one end to another.

1. The Data plane:

The data plane is responsible for handling the data packets and applying actions to them, based on rules that we program into lookup tables. The actions must happen at line speed, therefore we must be fast enough (e.g., 40Gbit/sec per port). Also called the data path or the forwarding plane, the data plane takes packets in one port of a switch and sends them out another port.

1. The Control plane:

The control plane is tasked with calculating and programming actions for the data plane. This is where the forwarding decisions are made and where other functions (e.g., Quality of Service, Virtual Local Area Networks, etc.) are implemented. The control plane is operating at a lower speed than the data plane. It does not operate — or need to operate — at wire speed.

1. The Management plane:

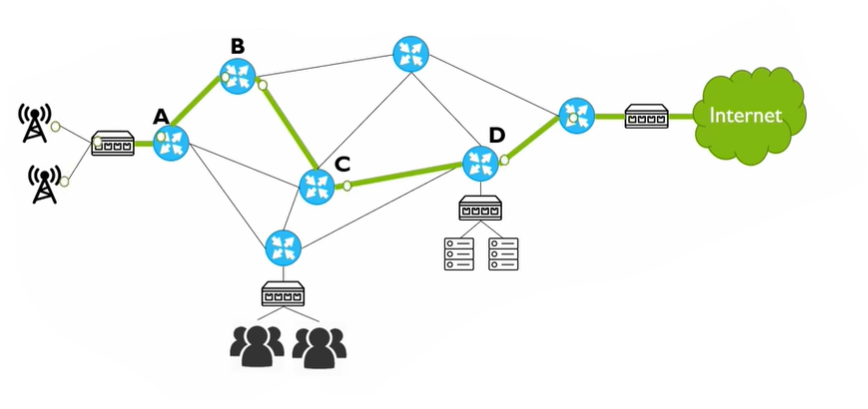
The management plane is where we can configure and monitor the network device (e.g., switch or router). The network device can be a shell, command-line interface (CLI) or web interface. The management plane usually runs on the same processor as the control plane.

Understanding the concept of the three planes in traditional network is the basis for development of the Software-defined Networking in modern days.

* 1. *Software-defined networking (SDN):*

In traditional network, each nodes of the network is a network device (e.g: routers, switches,..) that is integrated with data plane, control plane and management plane all in one device. The problem with this architecture is that each of the network device is internally different, has different configurations from each other and can only monitor or communicate with connected adjacent devices and cannot optimized the forwarding process through the whole network. Furthermore, scaling the network system is also a problem with the traditional network and the process of configurating the entire network system is very time and resource consuming.

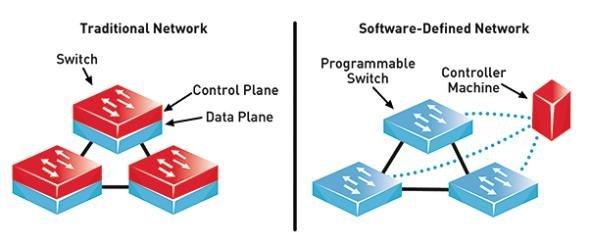
For example: we have a traditional network consists of nodes A, B, C and D (figure 1.1). If the device at node A decided to take the route through router B through C through D to connect to the internet but problem occurs with the device at node D, then A cannot know about that information because it is only adjacent to B and only knows the information at B. We can configure the devices to take other routes but it will inevitably takes time and resource.



*Figure 1.1 Traditional network*

Solution? How about we have a centralized controller or so called a “brain” that monitors and can does configuration on the entire network. With a centralized controller that is connected to every device on the network and overseas the entire network, any configuring jobs on the devices or network itself are done much faster, less costly and open the door to more options for scaling the network system or managing security. This is the basic concept of the Software-defined Network: having a separate controller that is centralized and connected to all network devices in the network.

SDN overcome traditional network challenges by separating the control plane and data plane. SDN assigns the control plane to an external device called controller, which manages all the routing decisions. Here routers do not have to do computationally expensive tasks. All protocols execute in the central controller that takes all the routing decisions. The controller always monitors the global view of the whole network, which can possibly through some well-deﬁned APIs. *(see figure 1.2 of the visualization of the two networks architecture).*



*Figure 1.2 Traditional network vs SDN*

1. ***Architectural design of SDN:***
   1. *The Openflow protocol:*

OpenFlow is todays best explored and is a standard protocol that exchange information between the control plane and OpenFlow devices present in the data plane in an SDN environment. This protocol is a multivendor standard and maintained by the Open Networking Foundation (ONF). The ONF is a joint initiative by major software giants like Google, Microsoft, and Microsoft etc. In the year 2011, the ﬁrst version V 1.1 came out to the market. Many vendors, including Bell, IBM, CISCO, Juniper, etc., are using the OpenFlow standard and add this functionality to their network devices to make the device SDN compatible. Most of the commodity networking hardware devices have started using SDN enabling technology including HP, IBM, NEC, Pronto, Juniper, Extreme, etc.

* 1. *Layers of SDN:*

1. Infrastructure layer:

* Consists of forwarding devices like router, switch, ….. These can be traditional switches hardware as long as they support a programmable interface such as Openflow or software switches such as Open Vswitch.

+ Hardware switches: High performance

+ Software switches: Greate flexibility for new behaviors

* These forwarding devices receive packets, take actions (dropping packets, modifying packet headers, sending out packets) on packets and update counters.

1. Control Layer: (Controller)

* Has a global view of all the devices in the Infrastructure Layer and provide the SDN Applications a simplified view of the network
* Serves as a translator between the application layer and the forwarding devices
* Run with key core-services (Topology, Inventory, Statistics, Host Tracker )

+ Topology Service: Determines how forwarding devices are connected

+ Inventory Service: Used to track all SND enabled devices and record their basic information (Openflow version, Capability, ….. )

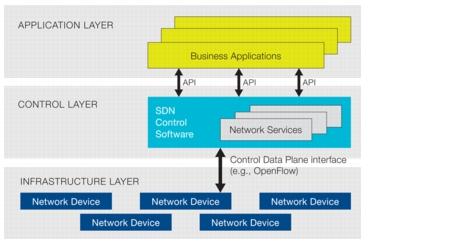
+ Statistics Service: Reads counter information off of forwarding devices ( Traffic counters on flows, Interfaces, Flow table, ….)

+ Host Tracker: Discovers where IP addresses and MAC addresses of hosts are located on the network by intercepting packets in the network

* Interacting with network nodes and allowing the network administrator to apply custom policies to the physical layer devices

1. Application Layer:

* Manages all the routing decisions and determine what to do when forwarding devices receive packets. Here forwarding devices do not have to do computationally expensive tasks.
* Serves network-focused purposes: energy eﬃcient networking, security monitoring, network virtualization, …...
* Provides end-to-end solutions for real world enterprise and data center networks

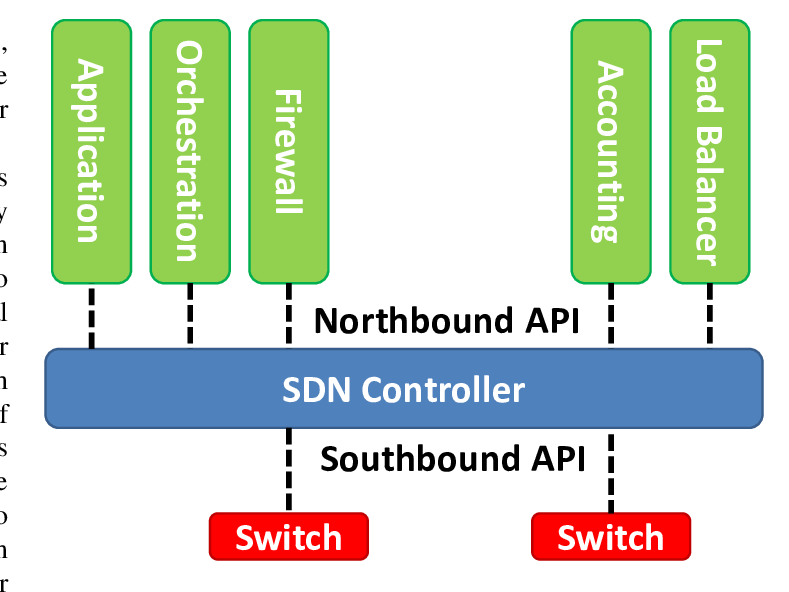


*Figure 2.1 Layers of SDN*

* 1. *Interfaces of controller:*

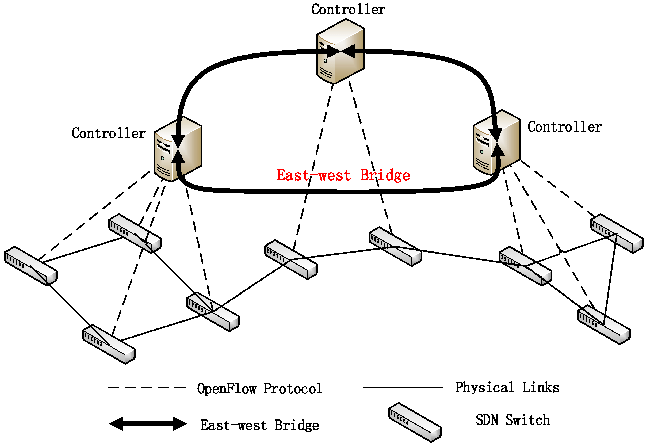
The SDN controller can interact with these three layers, through some standard open interfaces:

1. Southbound interface: Southbound interface creates a channel to interact with the controller and underlying forwarding elements. OpenFlow is the standardized protocol supported by ONF, is the widely used southbound interface, which establishes a secured link between the controller and forwarding devices. The remote updating of the switches′ ﬂow table by the controller is called programming in the SDN perspective. Another protocol called ForCES is used for a long time for communication between controller and data plane components. Though ForCES and SDN have some common goal, but they diﬀer in many aspects.
2. Northbound interface: The north bound APIs represents are interfaces between the controller and the applications application layer. This interface helps the application developers to manage the network through the program. Since the network policies are dynamic in nature in an SDN environment, hence the traditional languages fail to achieve this. To program the network devices, Frenetic is used, that provides standard libraries and support modular programming that help to design a high-level packet-forwarding policies for switches. Similar to Frenetic, Pyretic is another programming platform, which provides modular programming and has increased abstraction layer that allow the application developer to develop a more challenging application. A control architecture for SDN called Procera, which uses abstraction layers to hide the details that are unimportant and reveal only the relevant information to upper layers.



*Figure 2.2 Northbound and Southbound interfaces*

1. East-west Bridge: In practice, a large-scale enterprise network is partitioned into sub-networks, where each sub-network is handled by a diﬀerent controller. In this multi-controller-based architecture, each network has the global network view to communicate with the other domain and exchange topology among the sub-domains. Directly intra domain communicate cannot possible without the help of a proper interface. East-west protocols solve this problem by exchanging network view among the controllers.



*Figure 2.3: East-west bridge*

1. ***Packet flow:***

When a packet arrives at a forwarding device, it either already knows what to do with it, or queries the Network Operating System. The applications on the Controller determine what action to take on the packet and push this information down to the forwarding device. Using the Controller as a translator, the forwarding device then take the assigned action on the packet. The forwarding devices will usually also cache instructions so that future packets don’t require checking with the Controller. This same process continues device by device until the packet leaves the SDN network to a destination.

1. ***Benefits of SDN:***

With SDN, an administrator can change any network switch's rules when necessary -- prioritizing, deprioritizing or even blocking specific types of packets with a granular level of control and security. This is especially helpful in a cloud computing multi-tenant architecture, because it enables the administrator to manage traffic loads in a flexible and more efficient manner. Essentially, this enables the administrator to use less expensive commodity switches and have more control over network traffic flow than ever before.

Other benefits of SDN are network management and end-to-end visibility. A network administrator need only deal with one centralized controller to distribute policies to the connected switches, instead of configuring multiple individual devices. This capability is also a security advantage because the controller can monitor traffic and deploy security policies. If the controller deems traffic suspicious, for example, it can reroute or drop the packets.

SDN also virtualizes hardware and services that were previously carried out by dedicated hardware, resulting in the touted benefits of a reduced hardware footprint and lower operational costs.

Additionally, software-defined networking contributed to the emergence of software-defined wide area network (SD-WAN) technology. SD-WAN employs the virtual overlay aspect of SDN technology, abstracting an organization's connectivity links throughout its WAN and creating a virtual network that can use whichever connection the controller deems fit to send traffic.

1. ***References:***

* A Comprehensive Tutorial on Software Deﬁned Network: The Driving Force for the Future Internet Technology: <https://www.researchgate.net/publication/309259552_A_Comprehensive_Tutorial_on_Software_Defined_Network_The_Driving_Force_for_the_Future_Internet_Technology?fbclid=IwAR1LIyv_fZZJ2be2DAcPOhtaP6hb1oel9MhN4yVhRxvY5dGuZeU9POZJ7Ng>
* Introduction to SDN (Software defined network) - SDN and Openflow Architecture:

<https://www.youtube.com/watch?v=TQVl5-G3u2U>

* software-defined networking (SDN):

<https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN>

* Software Defined Networking Fundamentals Part 1: Intro to Networking Planes:

<https://www.linux.com/tutorials/software-defined-networking-fundamentals-part-1-intro-networking-planes/>

* Software Defined Networking Fundamentals Part 2: Switches and Network Architecture:

<https://www.linux.com/tutorials/software-defined-networking-fundamentals-part-2-switches-and-network-architecture/>

* Software Defined Networking Fundamentals Part 3: Transformation of Network Architecture:

<https://www.linux.com/tutorials/software-defined-networking-fundamentals-part-3-transformation-network-architecture/>

* What is software-defined networking (SDN)?:

<https://www.youtube.com/watch?v=Z5Gi2Bpd82M>

* Introduction to SDN (Software-defined Networking):

<https://www.youtube.com/watch?v=DiChnu_PAzA>

* What is a Software Defined Network:

<https://www.youtube.com/watch?v=lPL_oQT9tmc>

1. **Lab set up tutorial:**
2. OpenDaylight SDN Controller:

* Download: <https://docs.opendaylight.org/en/latest/downloads.html>
* Choosing OpenDaylight Sodium Zip/Tar is accepted.

1. VirtualBox – Ubuntu:

* Download:
  + VirtualBox: <https://www.virtualbox.org/wiki/Downloads>
  + Ubuntu for PC: <https://ubuntu.com/download/desktop>
    - How to install Ubuntu in VirtualBox: <https://gocinfo.com/cai-dat-ubuntu-tren-may-ao-virtualbox-tren-windows.html>
* You have to download extension pack if you want to run virtual machine such as: mininet or opendaylight

1. Mininet Emulator:

* Download: <http://mininet.org/download/>
* Get familiar with mininet: <https://github.com/mininet/mininet>

1. **Lab Guide**: <https://github.com/hocchudong/thuctap032016/blob/master/ThaiPH/SDN/ThaiPH_opendaylight_mininet_lab.md>

<http://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/>

**Using OpenDaylight SDN Controller with Mininet Network Emulator:**

1. **Using Virtual Machines:**

We will use 2 virtual machines:

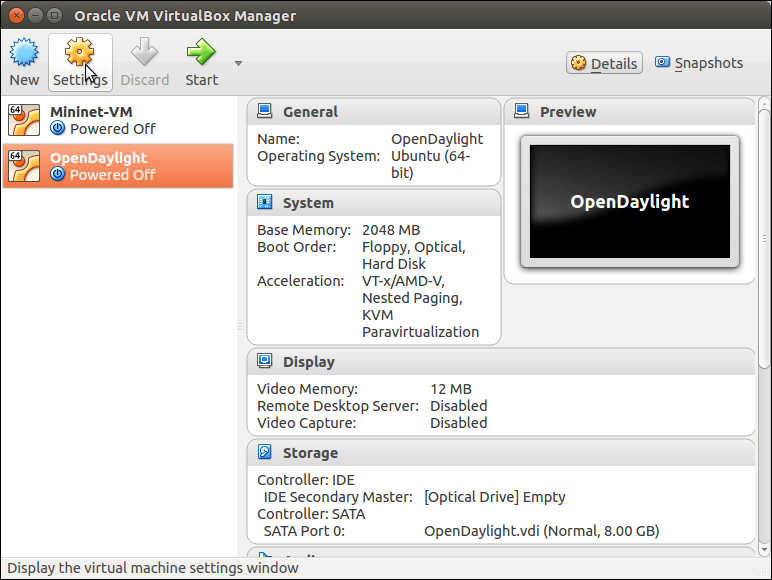
* Mininet emulator network <http://www.brianlinkletter.com/set-up-mininet/>
* OpenDaylight Controller

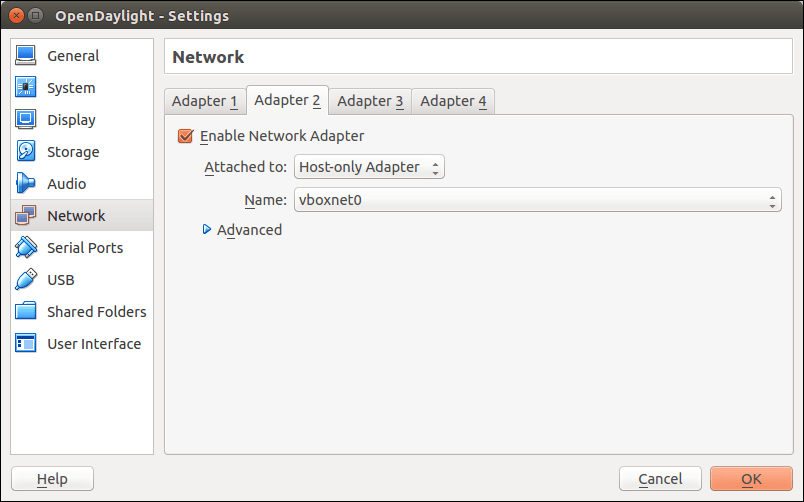
We also use VirtualBox to run the Mininet VM and install, run the OpenDaylight SDN Controller on a new VM we create in VirtualBox

Mininet setting:

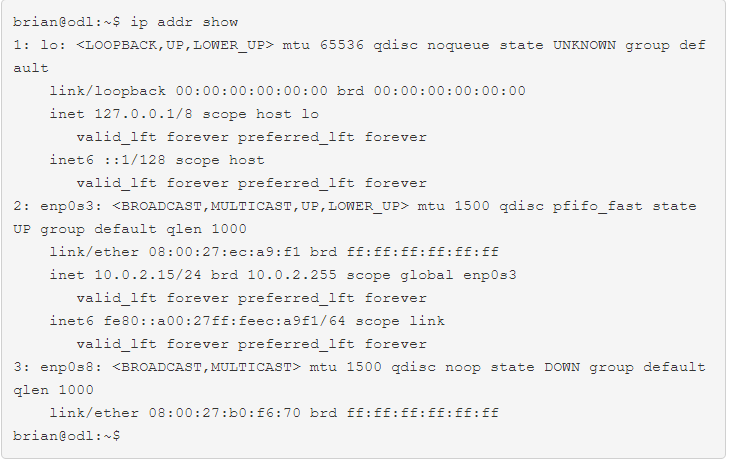
|  |  |
| --- | --- |
| Host-only network address | 192.168.x.0/24 |
| Host IP address on host-only network | 192.168.x.1/24 |
| Server Address: | 192.168.x.2/24 |
| Virtual Machine’s virtual interface IP address on host-only network (Lower Address Bound) | 192.168.x.3/24 |

1. **Setting up the OpenDaylight Virtual Machine:**

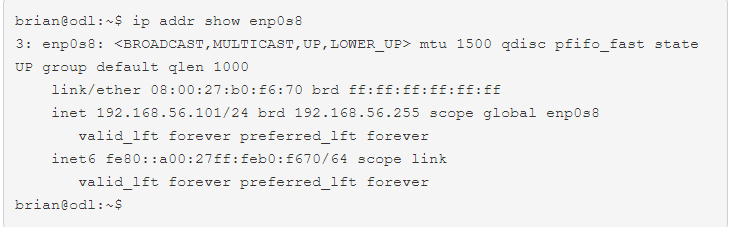




1. **Configure OpenDaylight VM interfaces:**







#### Connect to the OpenDaylight VM using SSH:

Open a terminal on host computer and login using SSH:

#### 

#### Install Java:

#### Code: (ODL – VM)

#### Sudo apt-get openjdk-7-jdk

#### Sudo apt-get install openjdk-7-jdk

#### Sudo apt-get install openjdk-7-jre

#### Sudo apt-get install maven

#### Install OpenDaylight:

#### From ODL-VM use browser to download OpenDaylight software from the OpenDaylight website

#### Install OpenDaylight by extracting the tar file:

#### Code:

#### Cd ODL/bin

#### ./karaf –of13

#### 

1. **Install OpenDaylight features:**

We installed the following features. Click on each feature to learn more about it:

* [odl-restconf](https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:Restconf): Allows access to RESTCONF API
* [odl-l2switch-switch](https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:L2_Switch): Provides network functionality similar to an Ethernet switch
* [odl-mdsal-apidocs](https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:Restconf_API_Explorer): Allows access to Yang API
* [odl-dlux-all](https://wiki.opendaylight.org/view/OpenDaylight_DLUX:DLUX_Karaf_Feature): OpenDaylight graphical user interface

To list all available optional features, run the command:

**Feature:list**

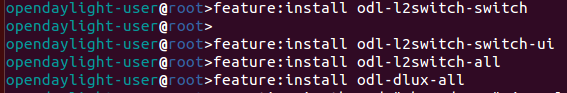
To list all installed features, run the command:

**Feature:list –installed**

\*\*Stop OpenDaylight by: enter the “Crt-d” or use command: system:shutdown / logout



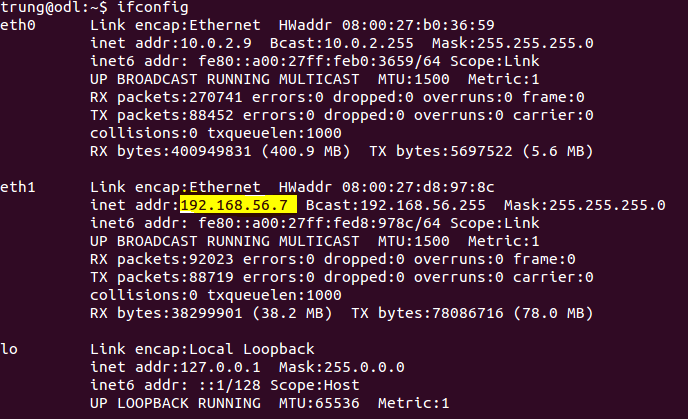




**\*\*In the ODL virtual machine:**

* **Create another terminal: (install nmap: which could show fully the localhost ip)**





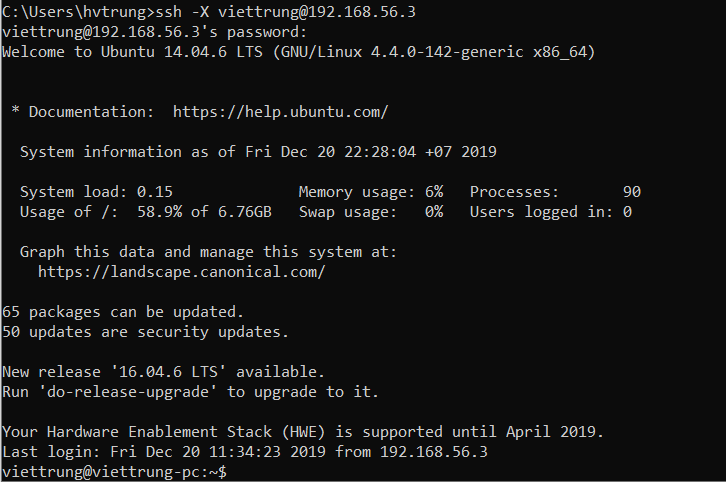
* + - **This ip address will host the ODL. In the browser: 192.168.56.7:8181/index.html**
    - **Login web site: user:admin \\ pass:admin**

1. **Set up the Mininet Virtual Machine:**

Use command: “ip addr show” to see ip address of host-only interface.

We need to use that IP address to access applications running on this virtual machine

1. **Connect to the Mininet VM using SSH:**



1. Start Mininet:

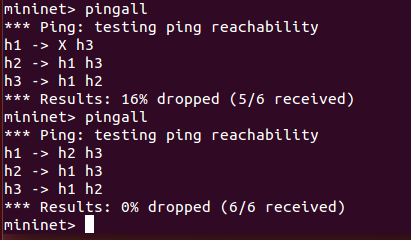
On the Mininet VM, start a simple network topology. In this case, we will do the following:

* + - Set up three switches in a linear topology
    - Each switch will be connected to one host
    - The MAC address on each host will be set to a simple number
    - The remote controller, OpenDaylight, is at IP address 192.168.56.101:6633
    - We will use OpenFlow version 1.3

The command: sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.56.8,port=6633 --switch ovs,protocols=OpenFlow13

1. **Test the network:**

Mininet> pingall



The OpenDaylight Graphical User Interface:

1. Topology:
2. Nodes:
3. Yang UI:
4. **Shut down the project:**

* On the Mininet VM, stop Mininet and clean up nodes, then shut down the VM:

From the mininet: exit

Code: sudo mn –c

Sudo shutdown –h now

* On the OpenDaylight VM, stop OpenDaylight and shut down the VM:

System: shutdown

Sudo shutdown –h now