Cloud-native Software-Defined Mobile Networks
Assignment #3
Virtual Network Developments based on SDN

Sharif University of Technology

Department of Electrical Engineering

Due Date: 25.04.2020

Problem 1

The objective of this problem is getting familiar with some open source tools for software-defined networking. It is suggested do parts in order while taking care of the details for each part.

Deliverable

- You should submit an individual Python code for each part.

Part 1- Network Emulation in Mininet

Create the network topology shown in Fig. 1 in Mininet using python API and make sure the hosts are able to ping each other.

Figure 1: LAN topology

You may arbitrarily apply the following template for creating topology with Python:

```
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch, UserSwitch,
    OVSController
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink


def topology():
  net = Mininet( controller=OVSController, switch=OVSKernelSwitch )
  # Add hosts and switches
  #
  #
  # Add links between nodes
  #
  #
  net.build()
  net.start()
  # *****************
  # Write your code Here
  # *****************

  print "*** Running CLI"
  CLI( net )
  print "*** Stopping network"
  net.stop()



if __name__ == '__main__':
setLogLevel( 'info' )
topology()

```

Listing 1: code template

Part 2- Adding Layer 3 Functionality

In the previous part, all network components are in the same sub-net. What if h1 and h2 be in different networks? In order to establish the connection between different hosts in different sub-networks we need to rely on layer-3 forwarding principals realized through routers. In this part, modify the previous network implementation by adding a router as depicted in Fig. 2. Please note that the address of interfaces have already been changed.

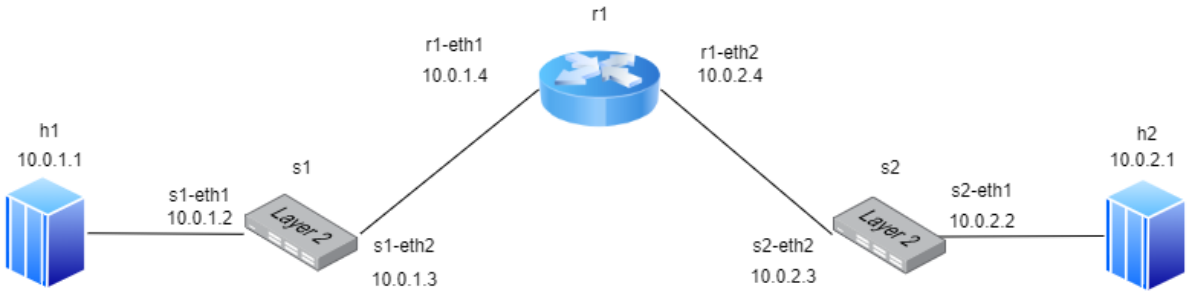

Figure 2: Layer-3 topology

To implement the router applying either a host with Linux IP-forward facilities or an Open Virtual Switch are acceptable. Again, perform ping tests between hosts to make sure that your topology works.

Part 3- Adding a Controller

In the previous part, we used the command line for adding flows to the switches manually. Here, applying an OpenDaylight (ODL) controller it is supposed to add the flows through RestConf API to the switches. Hence, it is needed to install the ODL controller with appropriate features and also modify your network topology creator file to add the controller into it.

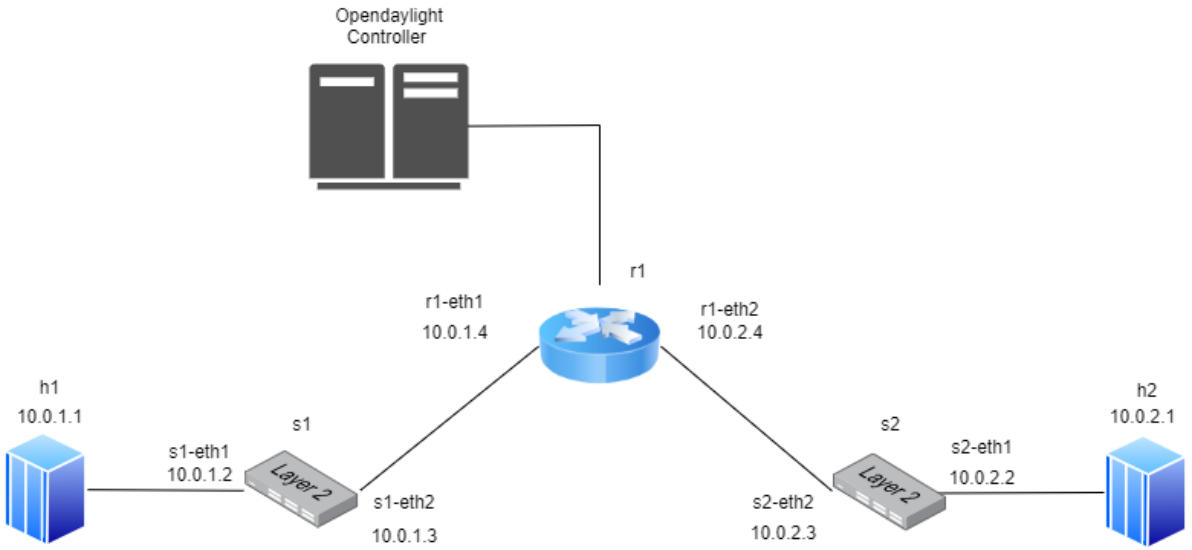

Figure 3: Layer-3 topology with controller

Problem 1 continued on next page...

For the implementation of this part please consider the following notes:

- The router has to be implemented lonely using an Open vSwitch (OVS) and NOT through a routing host.
- Adding flows has to be necessarily done via the installed controller and NOT through the command line in the router.
- Make sure that there exists no default flows on the router. To this end, use the the following command in the router console before adding any flow.
  "sh ovs-ofctl del-flows ¡router-name¿"
- Hosts should be able to ping each other, and try to exploit Wireshark for debugging.
- It is suggested to use individual virtual machines (VMs), namely, one for the controller and one for other network components.
- It is also highly recommended to use ODL version 7.3 and also strictly prevent installing redundant features.

Part 4- Development of Shortest Path Routing Application via SDN

Suppose a network consists of n nodes (switches, routers and hosts) and any *directed* link between two desired nodes is defined through a positive weight, then the network flow graph can be fully determined according to an asymmetric adjacency matrix $(A_{n\times n})$, where

- if $A_{ij} > 0$, it determines the weight (cost) of link from switch v_i to the switch v_j.
- if $A_{ij} = 0$, there isn't a link between switches v_i, v_j, i.e. these nodes are disconnected.

Moreover, s and d refer to the switches hosts $h1$ and $h2$ are connected to, respectively, such that $1 \leq s, d \leq n$ (see Fig. 4).

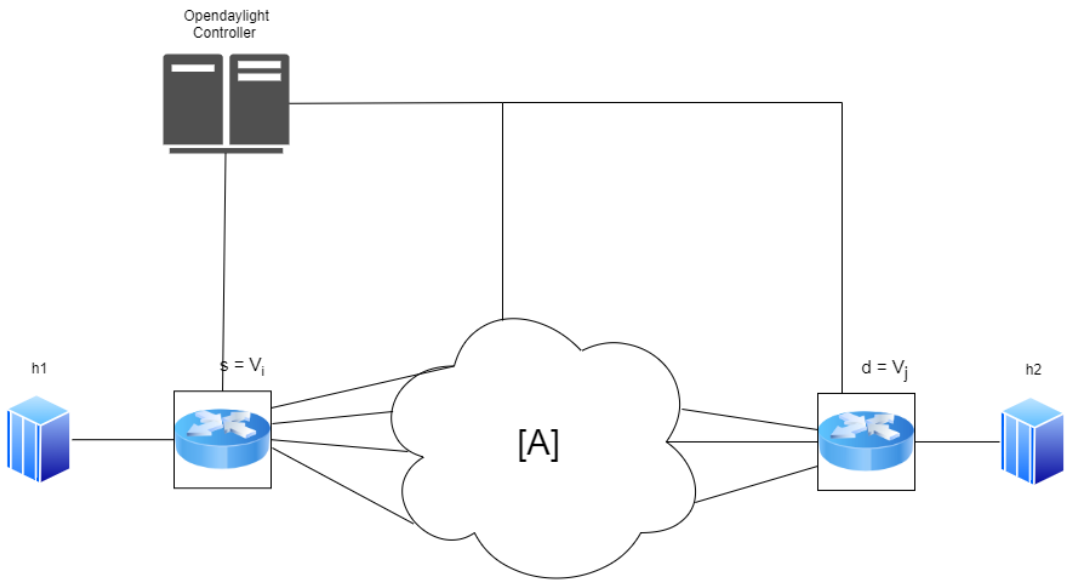

Figure 4: Layer-3 topology

Develop an application which following the calculation of the shortest path (minimum weight) between source and destination, instructs the ODL controller in order to forward the ping packets (request and responses) throughout the shortest paths at the corresponding directions by adding the flows to the routers .

Sample Input (for $n = 4$):

$$A = \begin{pmatrix} 0 & 2 & 3 & 4 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}, s = 1, d = 4$$

Output:
ping request path: $h_1 \to v_1 \to v_2 \to v_4 \to h_2$.
ping reply path: $h_2 \to v_4 \to v_1 \to h_1$.

Please note that:

- Hosts should be able to ping each other, use wireshark to check path of packets.
- The Adjacency matrix is not necessarily symmetric, so request and response paths can be different.

Problem 2

Virtual Extensible LAN (VXLAN) technology is a network virtualization approach designed to address the challenges of scaling networks in large cloud computing deployments. At its most basic level, VXLAN is a tunneling protocol.
VXLAN is a formal internet standard, specified in RFC 7348, is another application layer-protocol based on UDP that runs on port 4789. For further explanation and details refer to the following link:
https://medium.com/@NTTICT/vxlan-explained-930cc825a51
In case of software-defined networking (SDN), VXLAN is often used as a basis for larger orchestration tooling that synchronizes state and configuration across multiple network devices. It also assists APIs with integration and automation. NSX, Contrail and Openstack Neutron are examples of systems that provide a logical front end to configure VXLAN across multiple devices.

Part 1- Emulating a 2-tenant network based on VXLAN without controller

Bring up two VMs with the means of Oracle VM VirtualBox and connect them so that each of them is pingable from another one. Then build the given topology shown in Fig. 5 using Mininet, and install the VXLAN tunnels by configuring the switches properly. It is expected to be able to ping each host from the corresponding host at the other end of the tunnel.

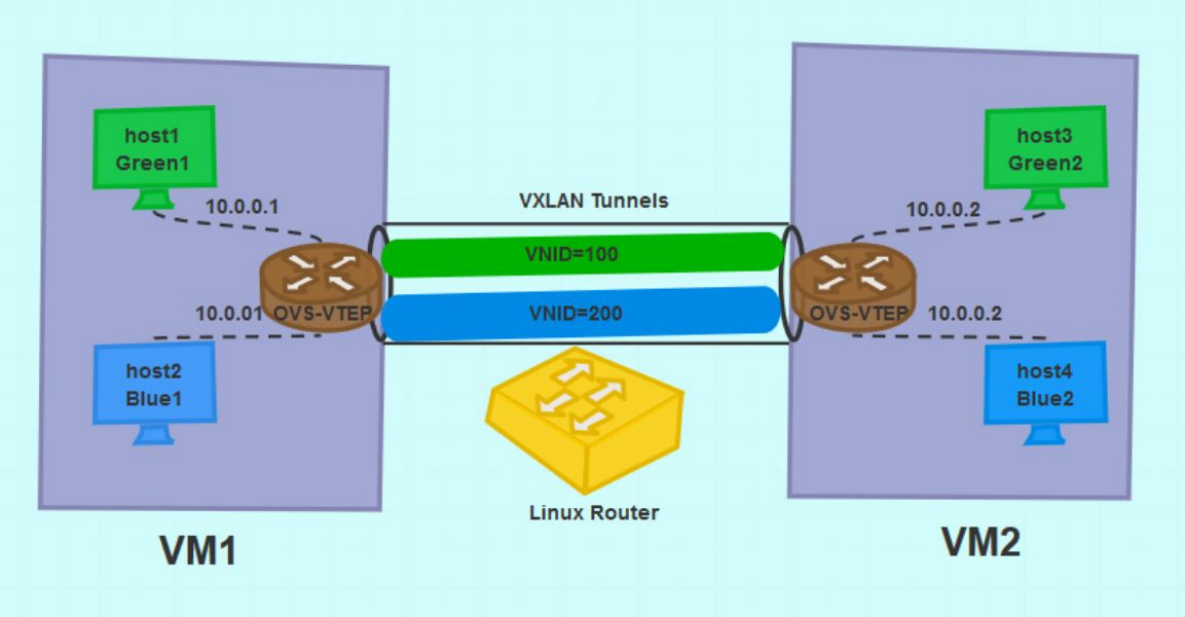

Figure 5: 2-tenant topology

Deliverable
The following files have to be submitted:

- Two Python scripts which build the network on each VM.
- Two .text files including appropriate rules for switches.
- Wireshark PCAP files capturing the packets inside and outside of the tunnel it using while observing appropriate network interfaces.

Part 2- Emulating a 2-tenant network based on VXLAN with controller

Repeat the previous part but this time configure the switches using OpenDaylight.

Hint
Make use of OpenDaylight Dlux to see the topology and Yang UI to get insight into RESTful API. Moreover, it's recommended to use Postman to send HTTP requests to the controller to configure the switch. We also suggest to run the controller through your host Linux machine.

Deliverable
The following files have to be submitted:

- Two python scripts which build the network on each VM. (Note that it's different from the previous part)
- Screen-shot of the topology from the controller's view.
- PCAP files capturing the packets inside/outside of the tunnel using Wireshark.
- A single Python script sends those HTTP requests to the controller using "requests" Python package (work with Postman is not acceptable).