

SOFTWARE DEFINED NETWORKING

Lab 5: Configuring VXLAN to Provide Network Traffic Isolation

Document Version: 05-25-2020



Award 1829698 "CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview
Objectives
Lab settings
Lab roadmap3
1 Introduction
1.1 VXLAN architecture
1.2 VXLAN packet format
2 Lab topology
2.1 Lab settings
2.2 Loading a topology
2.3 Load the configuration file
2.4 Run the emulation
2.5 Verify the configuration
Configuring OSPF on routers r1 and r2 12
4 Configuring VXLAN 15
4.1 Run Mininet instances within the containers
4.2 Adding entries to the switches' flow tables
5 Verifying configuration
5.1 Performing connectivity test between end-hosts
5.2 Verifying VXLAN network identifiers by using Wireshark
References

Overview

This lab presents Virtual eXtensible Local Area Network (VXLAN), a network virtualization scheme that provides a solution for the scalability problems associated with datacenter and large cloud computing deployments. This lab aims to configure VXLAN to isolate network traffic within an emulated environment. Additionally, the user will inspect the packets to identify the fields corresponding to VXLAN network identifiers.

Objectives

By the end of this lab, the user will:

- 1. Understand the concept of VXLAN.
- 2. Assign IP addresses to a router interface.
- 3. Configure a routing protocol.
- 4. Emulate servers by using docker containers.
- 5. Push flow tables to configure VXLAN in a switch.
- 6. Isolate network traffic by using VXLAN.
- 7. Visualize VXLAN network identifiers by using Wireshark.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

- 1. Section 1: Introduction.
- 2. Section 2: Lab topology.
- 3. Section 3: Configuring OSPF router r1 and router r2.
- 4. Section 4: Configuring VXLAN.
- 5. Section 5: Verifying configuration.

1 Introduction

Data centers operate by hosting services for multiple tenants such as data servers and cloud computing services⁷. Those services require on-demand elastic provisioning of computing resources for multi-tenant environments. Such feature is supported by network virtualization, which provide an efficient way to host multiple tenants in the same server and traffic isolation, to avoid a tenant to have access to the data of another tenant.

Isolating network traffic could be done via Layer 2 or Layer 3 networks. For Layer 2 networks, VLANs are often used to segregate traffic. In such scenario a tenant could be identified by its own VLAN. However, VLAN-based network isolation suffers a limitation of 4094 VLANs, which is inadequate considering the high demand of cloud services. Additionally, a tenant could require multiple VLANs, which exacerbates the issue.

On the other hand, layer 3 networks do not provide an extensive solution for multi-tenant networks as well. Two tenants might use the same set of Layer 3 addresses within their networks, which requires the cloud provider to provide isolation in some other form. Further, requiring all tenants to use IP excludes customers relying on direct Layer 2 or non-IP Layer 3 protocols for inter VM communication.

1.1 VXLAN architecture

Hypervisor-based overlay networks is a novel use of Software-defined Network (SDN) capabilities³. This concept does not modify the physical network, which means that networking devices and their configurations remains unchanged. Hypervisor-based virtualized networks are built above such network⁵. The system at the edge of the network works as an interface to these virtual networks. In these networks, many details of the physical network from the devices that connect to the overlays are hidden.

VXLAN (Virtual eXtensible Local Area Network) addresses the above requirements of Layer 2 and Layer 3 data center network infrastructure in the presence of Virtual Machines (VMs) in a multi-tenant environment. VXLAN runs over the existing networking infrastructure and provides a means to increase the number of devices on a Layer 2 network. In summary, VXLAN is a Layer 2 overlay scheme build on the top of a Layer 3 network. Each overlay is unique within the tenant domain and is known as VXLAN segment. The communication is restricted just between VMs within the same VXLAN segment. Each VXLAN segment is identified by a 24-bit segment ID, called the VXLAN Network Identifier (VNI). This allows up to 16 million (2²⁴) VXLAN segments to coexist within the same administrative domain.

Consider Figure 1. VXLAN could be considered as a tunneling scheme to overlay Layer 2 networks on top of Layer 3 networks⁶. The tunnels are stateless, so each segment is encapsulated according to a set of rules. The end point of the tunnel (VXLAN Tunnel End Point or VTEP) is located within the hypervisor on the server that hosts the VM. The traffic is isolated according to the VNI and the end-hosts can communicate as they are located within the same network.



1.2 VXLAN packet format

Figure 2 illustrates the format of a VXLAN packet⁵. The outer header contains the MAC and IP addresses appropriate for sending a unicast packet to the destination switch, acting as a virtual tunnel end point. The VXLAN header follows the outer header and contains a VXLAN Network Identifier of 24 bits in length, sufficient for about 16 million networks.

Figure 2. VXLAN packet format.

2 Lab topology

Consider Figure 3. The topology consists of four end-hosts, two switches and two routers. The end hosts and switches are running inside Server 1 and Server 2. Those servers are implemented by Docker⁸ containers which run Mininet instances. Router r1 and router r2 are supported by Free-range Routing (FRR) engine.

Figure 3. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Device	Interface	IP Address	Subnet	
r1	r1-eth0	192.168.12.1	/30	
11	r1-eth1	192.168.1.1	/24	
rJ	r2-eth0	192.168.12.2	/30	
12	r2-eth1	192.168.2.1	/24	
h1	h1-eth0	10.0.0.1	/8	
h11	h11-eth0	10.0.0.1	/8	
h2	h2-eth0	10.0.0.2	/8	
h22	h22-eth0	10.0.0.2	/8	
d1	d1-eth0	192.168.1.10	/24	
d2	d2-eth0	192.168.2.10	/24	

Table 2. Topology information.

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), save the topology and export a layer 2 model.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type password.

Figure 4. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab5.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab5* and click on *Open*.

Figure 5. Opening topology.

Figure 6. MiniEdit's topology.

2.3 Load the configuration file

At this point the topology is loaded however, the interfaces are not configured. In order to assign IP addresses to the devices' interfaces, you will execute a script that loads the configuration to the routers and end devices.

Step 1. Click on the icon below to open Linux terminal.


```
Figure 7. Opening Linux terminal.
```

Step 2. Click on the Linux terminal and navigate into *SDN_Labs/lab5* directory by issuing the following command. This folder contains a configuration file and the script responsible for loading the configuration. The configuration file will assign the IP addresses to the routers' interfaces. The command is short for change directory followed by an argument that specifies the destination directory.

cd SDN_Labs/lab5

Figure 8. Entering the *SDN_Labs/lab5* directory.

Step 3. To execute the shell script, type the following command. The argument of the program corresponds to the configuration zip file that will be loaded in all the routers in the topology.

./config_loader.	sh lab5_conf.zip	p	
•		sdn@adm	in: ~/SDN_Labs/lab5
File A	Actions Edit View	Help	
	sdn@admin:	~/SDN_Labs/lab5	\otimes
sdn@ad sdn@ad sdn@ad	lmin:~\$ cd SDN_La lmin:~/SDN_Labs/l lmin:~/SDN_Labs/l	bs/lab5 ab5\$ <mark>.</mark> /config_loado ab5\$	er.sh lab5_conf.zip
Fi	gure 9. Executing t	he shell script to load	the configuration.

Step 4. Type the following command to exit the Linux terminal.

•		sdn@ac	lmin: ~/SDN_Labs/lab5
File Ad	ctions Edit	View Help	
	sdi	@admin: ~/SDN_Labs/lab5	0
sdn@adr sdn@adr sdn@adr	nin:~\$ cd nin:~/SDN nin:~/SDN	SDN_Labs/lab5 _ Labs/lab5 \$./config_loa _ Labs/lab5 \$ exit	ader.sh lab5_conf.zip

Figure 10. Exiting from the terminal.

2.4 Run the emulation

In this section, you will run the emulation and check the links and interfaces that connect the devices in the given topology.

Step 1. To proceed with the emulation, click on the *Run* button located in lower left-hand side.

j	Run]
	Stop	

Figure 11. Starting the emulation.

Step 2. Issue the following command on Mininet terminal to display the interface names and connections.

links	
1 -	Shell No. 1
File Actions Edit View He	lp
Shell	No. 1 🛞
containernet> links	
r1-eth0<->r2-eth0 (OK OK))
d1-eth0<->r1-eth1 (OK OK))
d2-eth0<->r2-eth1 (OK OK))
containernet>	
Eigure 12 Displayir	g network interfaces

Figure 12. Displaying network interfaces.

In Figure 12, the link displayed within the gray box indicates that interface *ethO* of router r1 connects to interface eth0 of router r2 (i.e., r1-eth0<->r2-eth0).

2.5 Verify the configuration

You will verify the IP addresses listed in Table 2 and inspect the routing table of routers r1, r2, and r3.

Step 1. In order to verify router r1, hold right-click on router r1 and select *Terminal*.

Figure 13. Opening a terminal on router r1.

Step 2. In this step, you will start zebra daemon, which is a multi-server routing software that provides TCP/IP based routing protocols. The configuration will not be working if you Page 10 do not enable zebra daemon initially. In order to start the zebra, type the following command:

zebra		
X	"Host: r1"	- 0 X
<pre>root@admin:/etc/routers/rl# ze root@admin:/etc/routers/rl#</pre>	bra	

Step 3. After initializing zebra, vtysh should be started in order to provide all the CLI commands defined by the daemons. To proceed, issue the following command:

 vtysh

 X
 "Host: r1"

 root@admin:/etc/routers/r1# zebra

 root@admin:/etc/routers/r1# vtysh

 Hello, this is FRRouting (version 7.5-dev).

 Copyright 1996-2005 Kunihiro Ishiguro, et al.

 admin#

 Figure 15. Starting vtysh on router r1.

Step 4. Type the following command on router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks. The routing table of router r1 does not contain any route to external networks as there is no routing protocol configured yet.

```
show ip route
```

X	"Host: r1"	-	9	×
root@adı root@adı	nin:/etc/routers/rl# zebra nin:/etc/routers/rl# vtysh			
Hello, Copyrig	this is FRRouting (version 7.2-dev). Nt 1996-2005 Kunihiro Ishiguro, et al.			
admin#	show ip route			
Codes:	<pre>< + kernel route, C - connected, S - static, R - RIP,) - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP, 7 - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP, 5 - PBR, f - OpenFabric, > - selected route. * - FIB route. g - gueued route. r - reject</pre>	ed	ro	ut
e				
C>* 192 C>* 192 admin#	168.1.0/24 is directly connected, r1-eth1, 00:00:11 168.12.0/30 is directly connected, r1-eth0, 00:00:11			

Figure 16. Displaying routing table of router r1.

The output in the figure above shows that the networks 192.168.1.0/24 and 192.168.12.0/30 are directly connected through the interfaces *r1-eth1* and *r1-eth0*, respectively.

Step 5. Hold right-click on router r2 and select Terminal.

Figure 17. Opening a terminal on router r2.

Step 6. Router r2 is configured similarly to router r1 but, with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r2 terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r2.

X	"Host: r2" – 0	×
root@admin:/etc/routers/r2# zebra root@admin:/etc/routers/r2# vtysh		
Hello, this is FRRouting (version Copyright 1996-2005 Kunihiro Ishig	7.2-dev). guro, et al.	
admin# show ip route		
Codes: K - kernel route, C - conne 0 - OSPF, I - IS-IS, B - BO T - Table, v - VNC, V - VNO F - PBR, f - OpenFabric, > - selected route, * - FIE	ected, S - static, R - RIP, GP, E - EIGRP, N - NHRP, C-Direct, A - Babel, D - SHARP, B route, q - queued route, r - rejected ro	out
e		
C>* 192.168.2.0/24 is directly con C>* 192.168.12.0/30 is directly co admin#	nnected, r2-eth1, 00:00:04 onnected, r2-eth0, 00:00:04	

Figure 18. Displaying routing table of router r2.

3 Configuring OSPF on routers r1 and r2

In this section, you will configure OSPF routing protocol in router r1 and router r2. First, you will enable the OSPF daemon on routers r3 and r4. Second, you will a establish single area OSPF, which is classified as area 0 or backbone area. Finally, you will advertise all the connected networks.

Step 1. To configure OSPF routing protocol, you need to enable the OSPF daemon first. In router r1, type the following command to exit the vtysh session.

exit

Figure 19. Exiting the vtysh session.

Step 2. Type the following command on router r1 terminal to enable OSPF daemon.

ospfd

Figure 20. Starting OSPF daemon.

Step 3. In order to enter to router r1 terminal, issue the following command.

vtysh

25	"Host: r1"
admin# exit root@admin:/etc/routers/rl# ospfd root@admin:/etc/routers/rl# vtysh	
Hello, this is FRRouting (version Copyright 1996-2005 Kunihiro Ishi	7.2-dev). guro, et al.
admin#	
Figure 21. Star Starting vtyshon	router r1.

Step 4. To enable router r1 configuration mode, issue the following command:

configure terminal

Figure 22. Enabling configuration mode on router r1.

Step 5. In order to configure OSPF routing protocol, type the command shown below. This command enables OSPF configuration mode where you advertise the networks directly connected to router r1.

Figure 23. Configuring OSPF on router r1.

Step 6. In this step, you will enable all the interfaces of router r1 to participate in the OSPF routing process, i.e., all the attached networks will be advertised to OSPF neighbors. The advertised networks are associated with area 0. To advertise all connected networks in the same command, the network 0.0.0/0 will be used. This network address matches all IP addresses.

```
network 0.0.0.0/0 area 0
```

X	"Host: r1"
admin# exit	
root@admin:/etc/rou	uters/rl# ospfd
root@admin:/etc/rou	uters/rl# vtysh
Hello, this is FRR	outing (version 7.2-dev).
Copyright 1996-200	5 Kunihiro Ishiguro, et al.
admin# configure te	erminal
admin(config)# rout	ter ospf
admin(config-route	r)# network 0.0.0.0/0 area 0
admin(config-route	r)#

Figure 24. Enabling all the interfaces of router r1 to participate in the OSPF routing process.

Step 7. Type the following command to exit from the configuration mode.

end

Figure 25. Exiting from the configuration mode.

Step 8. Router r2 is configured similarly to router r1. Those steps are summarized in the following figure. To proceed, on route r2 terminal, issue the commands depicted below.

Figure 26. Exiting from the configuration mode.

Step 9. Type the following command to verify the routing table of router r1.

Figure 27. Verifying the routing table of router r1.

Consider Figure 27. The network 192.168.2.0/24 is learned via OSPF ($\bigcirc *$) and it is reachable via the next hop 192.168.12.2 (route r2).

4 Configuring VXLAN

In this section, the user will start the networks within the containers d1 and d2. Both containers run a Mininet topology as depicted in Figure 3. In container d1, the topology consists in two hosts (h1 and h11) connected to a switch (s1). Similarly, container d2 runs a topology with two hosts (h2 and h22) connected to a switch (s2). The end-hosts within the containers will be isolated by using VXLAN.

Note that the containers d1 and d2 emulates a multitenant environment. Multi-tenancy is a mode of operation where multiple independent instances such as end-hosts (see Figure 3) of a tenant operate in a shared environment, while ensuring logical segmentation between the instances. A tenant could be a business entity, user group,

python start_server1.py

applications, or cloud services. The tenant instances such as h1, h11, h2 and h22 are logically isolated but physically operate on the same fabric.

4.1 Run Mininet instances within the containers

The following section shows the steps to run a Mininet topology within the containers and how to navigate through the configuration files.

Step 1. In container d1 terminal, type the following python script to start a Mininet instance that consists of two hosts connected to a switch.

```
root@d1: ~
                                                                         – ø x
root@d1:~# python start_server1.py
* Starting ovsdb-server
* Configuring Open vSwitch system IDs
* Starting ovs-vswitchd
* Enabling remote OVSDB managers
*** Error setting resource limits. Mininet's performance may be affected.
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h11
*** Starting controller
*** Starting 1 switches
s1 ..
Host ',
       h1.name, 'has IP address = 10.0.0.1 and MAC address = 00:00:00:00:00
:01
       h11.name, 'has IP address = 10.0.0.1 and MAC address = 00:00:00:00:0
Host ',
0:01
*** Running CLI
*** Starting CLI:
               Figure 28. Starting a Mininet instance within container d1.
```

The figure above starts a Mininet instance in the container d1. Also, the information about the hosts are summarized after starting switch s1.

Notice that host h1 and host h11 have the same IP addresses and MAC addresses. These hosts will be isolated b using VXLAN.

Step 2. In container d1, run the following command to verify the devices in the topology:

```
links
```


Figure 29. Verifying the links between the devices in container d1.

The figure above shows that the host h1 and switch s1 are connected via the interface pair *h1-eth0<->s1-eth1*. Similarly, host h11 is connected to the switch s1 (*h11-eth0<->s1-eth2*).

Step 3. Similarly, in container d2 terminal, type the following python script to start a Mininet instance that consists in two hosts connected to a switch as well.

python start_server2.py			
Х	root@d2: ~		- ø ×
root@d2:~# python start_server2.	ру		
* Starting ovsdb-server			
* Configuring Open vSwitch syst	iem IDs		
* Starting ovs-vswitchd	227		
* Enabling remote UVSDB manager	5 Mininotle ne	rformonco mo	w bo offected
*** Adding controller	.s. mininet s pe	errormance ma	y be arrected.
*** Adding hosts			
*** Adding switch			
*** Creating links			
*** Starting network			
*** Configuring hosts			
h2 h22			
*** Starting controller			
www.pricitics.g. indialbox			
s2			
Host h2 has TP address = 10.0	0.2 and MAC a	address = 00	:00:00:00:00:02
Host h22 has IP address = 10.	0.0.2 and MAC	address = 0	0:00:00:00:00:02
*** Running CLI			
*** Starting CLI:			
mininet>			

Figure 30. Starting a Mininet instance within container d2.

The figure above starts a Mininet instance in the container d2. Also, the information about the hosts are summarized after starting switch s2.

Notice that host h2 and host h22 have the same IP addresses and MAC addresses. These hosts will be isolated b using VXLAN.

Step 4. In container d2 terminal, run the following command to verify the devices in the topology:

links

Figure 31. Verifying the links between the devices in container d2.

The figure above shows that the host h2 and switch s2 are connected via the interface pair *h2-eth0<->s2-eth1*. Similarly, host h22 is connected to the switch s2 (*h22-eth0<->s2-eth2*).

4.2 Adding entries to the switches' flow tables

In this section you will add entries to the flow tables of switch s1 and switch s2. These entries are added to a table that is responsible for traffic processing. In this lab, the flow tables specify the VXLAN tags and the actions to forward the packets to their right destination.

The main purpose of configuring VXLAN in this lab is to isolate the traffic from h1 to h2 and from h11 to h22.

Step 1. To visualize the entries to be added to the flow table of switch s1, in container d1, type the following command:

```
sh cat flow1.txt | nl
```

20	root@d1: ~
mininet>	sh cat flows1.txt nl
1	<pre>table=0,in port=1,actions=set field:100->tun id,resubmit(,1)</pre>
2	<pre>table=0,in port=2,actions=set field:200->tun id,resubmit(,1)</pre>
3	table=0, actions=resubmit(,1)
4	table=1,tun_id=100,dl_dst=00:00:00:00:00:01,actions=output:1
5	table=1,tun_id=200,dl_dst=00:00:00:00:00:01,actions=output:2
6	table=1,tun_id=100,dl_dst=00:00:00:00:00:02,actions=output:10
7	table=1,tun_id=200,dl_dst=00:00:00:00:00:02,actions=output:10
8	<pre>table=1,tun_id=100,arp,nw_dst=10.0.0.1,actions=output:1</pre>
9	table=1,tun_id=200,arp,nw_dst=10.0.0.1,actions=output:2
10	<pre>table=1,tun_id=100,arp,nw_dst=10.0.0.2,actions=output:10</pre>
11	table=1,tun_id=200,arp,nw_dst=10.0.0.2,actions=output:10
12	table=1,priority=100,actions=drop
mininet>	

Figure 32. Flow table in container d1.

Step 2. In container d1, Issue the following command to add entries to the flow table of switch s1.

```
sh ovs-ofctl add-flows s1 flows1.txt
```


Figure 33. Adding flow entries to switch s1.

Step 3. In this step, you will configure a VXLAN tunnel endpoint (VTEP) that will enable outgoing traffic from switch s1 to the outer network. A script is written to facilitate this process. To execute the script, type the following command.

Figure 34. Enabling outgoing traffic in switch s1.

VTEP is the device responsible for encapsulating and de-encapsulating layer 2 traffic. This device is the connection between the overlay and the underlay network. In this case, the VTEP is configured to provide connectivity between the switches and the containers' egress interfaces.

Step 4. In container d2, issue the following command to add entries to the flow table of switch s2.

```
sh ovs-ofctl add-flows s2 flows2.txt
```


Figure 35. Adding flow entries to switch s2.

Step 5. Similarly, in container d2, type the command below to configure a VXLAN tunnel endpoint (VTEP) in order that enables outgoing traffic from switch s2 to the outer network, issue the following command:

```
sh ./vxlan cmd2.cmd
```


Figure 36. Enabling outgoing traffic in switch s2.

5 Verifying configuration

In this section, the user will verify that the VXLAN tags were applied accordingly. Notice that the traffic between h1 and h2 has the VXLAN tag 100 and, the traffic between h11 and h22 corresponds to the VXLAN tag 200. This tag is known as the VXLAN Network Identifier (VNI). The VNI is used to identify VXLAN traffic.

5.1 Performing connectivity test between end-hosts

The following steps aim to verify the connectivity between end-hosts. This means that there should be connectivity between h1 and h2, also between h11 and h22.

Step 1. In container d1 terminal, issue the following command to verify the connectivity between host h1 and host h11. Notice that h_1 specifies host 1 as the source.

```
h1 ping 10.0.0.2
```

X					ro	oot@d1:	-		
mir	ninet>	h1 pi	ing 10.0.0.	.2					
PI	NG 10.0	0.0.2	(10.0.0.2)	56(8	4) byt	es of o	lata.		
64	bytes	from	10.0.0.2:	icmp_	seq=3	ttl=64	time=0.309	ms	
64	bytes	from	10.0.0.2:	icmp	seq=4	ttl=64	time=0.104	ms	
64	bytes	from	10.0.0.2:	icmp_	seq=5	ttl=64	time=0.104	ms	
64	bytes	from	10.0.0.2:	icmp_	seq=6	ttl=64	time=0.178	ms	
64	bytes	from	10.0.0.2:	icmp_	seq=7	ttl=64	time=0.245	ms	
64	bytes	from	10.0.0.2:	icmp_	seq=8	ttl=64	time=0.224	ms	
64	bytes	from	10.0.0.2:	icmp_	seq=9	ttl=64	time=0.176	ms	
64	bytes	from	10.0.0.2:	icmp_	seq=10	ttl=64	1 time=0.533	ms	
64	bytes	from	10.0.0.2:	icmp	seq=11	ttl=64	1 time=0.107	ms	
64	bytes	from	10.0.0.2:	icmp	seq=12	ttl=64	1 time=0.085	ms	
64	bytes	from	10.0.0.2:	icmp	seq=13	ttl=64	1 time=0.082	ms	
64	bytes	from	10.0.0.2:	icmp_	seq=14	ttl=64	1 time=0.083	ms	

Figure 37. Performing a connectivity test between host h1 and host h2.

Consider Figure 37. The results show a successful connectivity test.

Step 2. In container d2 terminal, issue the command shown below to disable the network interface of host h2.

Figure 38. Disabling h2 network interface.

Step 3. Click on container d1 terminal. The user will verify that the connectivity is lost. Press Ctrl+c to stop the test.

X			root@	d1: ~		
From	10.0.0.1	icmp seq=320	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=321	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=322	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=323	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=324	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=325	Destination	Host	Unreachable	
From	10.0.0.1	<pre>icmp_seq=326</pre>	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=327	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=328	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=329	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=330	Destination	Host	Unreachable	
From	10.0.0.1	<pre>icmp_seq=331</pre>	Destination	Host	Unreachable	
From	10.0.0.1	<pre>icmp_seq=332</pre>	Destination	Host	Unreachable	
From	10.0.0.1	icmp_seq=333	Destination	Host	Unreachable	

Figure 39. Verifying connectivity between host h1 and host h2.

Step 4. In container d1 terminal, issue the following command to test the connectivity between host h11 and host h22. Notice that h11 specifies host 11 as the source.

h11	ping	10.0.0.2
	Prid	10.0.0.2

20				n	oot@d1	~		
mi	ninet>	h11 p	oing 10.0.0	9.2				
PI	NG 10.0	0.0.2	(10.0.0.2) 56(84) by	tes of (lata.		
64	bytes	from	10.0.0.2:	icmp seq=3	ttl=64	time=0.258	ms	
64	bytes	from	10.0.0.2:	icmp seq=4	ttl=64	time=0.094	ms	
64	bytes	from	10.0.0.2:	icmp seq=5	ttl=64	time=0.091	ms	
64	bytes	from	10.0.0.2:	icmp seq=6	ttl=64	time=0.090	ms	
64	bytes	from	10.0.0.2:	icmp seq=7	ttl=64	time=0.093	ms	
64	bytes	from	10.0.0.2:	icmp_seq=8	ttl=64	time=0.108	ms	

Figure 40. Performing a connectivity test between host h11 and host h22.

The results will display a successful connectivity test. Do not stop the connectivity test.

5.2 Verifying VXLAN network identifiers using Wireshark

The following steps show how to verify VXLAN network identifiers using Wireshark network analyzer. The identifiers are used by the switch to isolate network traffic.

Step 1. Click on router r1 terminal and issue the following command to exit the vtysh session.

exit **Host: r1**" admin# exit root@admin:/etc/routers/r1# Figure 41. Exiting from vtysh.

Step 2. In router r1 terminal, start Wireshark dissector by issuing the following command. A new window will emerge.

wireshark

Figure 42. Starting Wireshark network analyzer.

After executing the above command on router r1 terminal, Wireshark window will open, where you monitor different interfaces related to router r1.

Step 3. Click on interface *r1-ethO* then on the icon located on upper left-hand side to start capturing packets on this interface.

					Th	e Wire	shar	k Ne	etwo	rk A	naly	zer					-	a x
<u>F</u> ile	Edit	<u>V</u> iew	<u>G</u> o <u>C</u> a	apture	<u>A</u> nalyze	e <u>S</u> tati	stics	Telep	hony	<u>W</u> ir	eless	<u>T</u> oo	ls <u>H</u> e	elp				
					X X	9				•	₫[Ð	Q		3.6	
Ap	oply a	display	filter	<ctrl-></ctrl->	>												-	• +
		W	elcome	e to W	ireshar	k												
		Ca	pture	е														
		us	ing this	filter: [Enter	a captu	re filte	er				•	All int	erface	es sho	wn *		
			r1 otb0															
			r1-eth1 Loopba any bluetoo nflog nfqueue Cisco re Display Randon system SSH rer	ck: lo th-mon e emote c Port AU n packe d Journa mote ca	itor :apture: X chann et genera al Export pture: s:	ciscodur el monit tor: ran : sdjour shdump	mp for cap dpkt nal	oture:	dpaux	rmon		-y_M_ y	\	~M				
		Le	arn															
		Use	er's Guie	de∙V	Viki · (Questio	ns an	d Ans	swers	· N	lailing	g List	ts					
		You	are runr	ning Wi	reshark 3	3.2.3 (Gi	t v3.2	3 рас	kageo	as 3	.2.3-1)).						

Figure 43. Starting packet capturing on interface r1-eth0.

Step 4. In the filter box located in upper left-hand side, type *vxlan* in order to filter the packets that contains VXLAN tags.

6									ŝ	*r1	-eth0	6						(222)	a x
<u>F</u> ile	<u>E</u> dit	Viev	v <u>G</u> o	<u> </u>	pture	Ana	lyze	Stati	stics	Te	ephon	<u>/</u> Wir	eless	Tools	<u>H</u> elp	ļ.			
		Z	۲			X	3	9	¢) 🕸		₹[•			
	klan																	\times	- +
No.	T 34 1 35 1 36 1 37 1 38 1 40 1 41 1 41 1	ime 4.336 5.359 5.366 6.384 7.408 7.408	60770 99926 90550 40071 40727 80366 30991	26 44 45 22 85 75 79 es or	50urce 10.0. 10.0. 10.0. 10.0. 10.0. 10.0.	e 0.2 0.1 0.2 0.1 0.2 0.1 0.2 0.2 e (11	84 bi	ts),	Des 10. 10. 10. 10. 10. 10. 10.	stina 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	tion 1 2 1 2 1 2 1 2 1 2 1 2 5 5 5 5 5 5 5 5	ured	Pro ICI ICI ICI ICI ICI ICI ICI	tocol MP MP MP MP MP MP MP	Lengt 148 148 148 148 148 148	Info Echo Echo Echo Echo Echo Echo Echo	(ping) (ping) (ping) (ping) (ping) (ping) (ping) ace r1-	reply request reply request reply request reply eth0, id	10 10 10 10 10 10 10 10
 Et Ir Us Vi Et Ir Ir 	herne iterne er Da rtual herne iterne	t II, t Pro tagra eXte t II, t Pro t Cor	Src otoco am Pr ensib Src otoco otrol	: 76 l Ver otoco le Lo : 00 l Ver Mess	8a:d rsion ol, S ocal 00:0 rsion sage	0:19: 4, S rc Po Area 0_00: 4, S Proto	52:77 rc: 1 rt: 4 Netwo 00:01 rc: 1 col	(76) 92.10 0742, rk (00) 0.0.0	:8a:d 58.1. Dst :00:0 9.1,	10:19 10, : Por : 00:00 Dst:	0:52:77 Dst: 1 t: 478 0:00:02 10.0	7), Ds L92.16 39 L), Ds .0.2	t: 92: 8.2.10 t: 00:	:b7:2i) :00:00	0:4a:5	1:ac 0:02	(92:b7: (00:00:	2b:4a:51 00:00:00	:ac) :02)
0000 0010 0020 0030 0040 0050 0060 0060 0070 0080 0090	92 00 02 02 03 03 00 14 24 34	b7 2b 86 6e 9a 9f 90 00 90 00 90 00 90 00 90 00 15 16 25 26 35 36	4a 5 ca 4 26 1 00 0 54 d 02 0 00 4 17 1 27 2 37	1 ac 0 00 2 b5 0 00 9 91 8 00 1 6a 8 19 8 29	76 8 3f 1 00 7 00 0 40 0 6f f 00 0 1a 1 2a 2	a d0 1 48 2 00 2 00 0 40 0 01 0 00 b 1c b 2c	19 5 38 c 00 0 00 0 01 4 1b 0 00 0 1d 1 2d 2	2 77 0 a8 8 00 0 00 d 15 4 d8 0 00 e 1f e 2f	08 0 01 0 00 0 00 0 00 0 00 0 00 0 00 0	0 45 a c0 0 00 1 08 0 00 0 cc 1 12 1 22 1 32	00 a8 00 00 01 5e 13 23 33	+JQ n @ & E · · T - Aj \$%&'() 4567	v F ? H8 r 0 @ № o	₹₩ · E · 1					

Figure 44. Filtering network traffic.

Step 5. Click on the arrow located on leftmost of the field called *Virtual eXtensible Local Area Network.* A list will be displayed. Verify that the *VXLAN Network Identifier* is 200. Notice that such tag corresponds to the traffic from h11 to h22.

6											*1	r 1 -	eth	0									- 1	a x	
File	Edit	Vie	N G	io <u>C</u>	aptur	e <u>A</u>	Analyz	ze	<u>S</u> tati	stics	5 7	Tele	phor	ער אַר	<u>V</u> irele	ess <u>T</u> oo	ols	Help)						Ĩ
			۲			5		5	٩	¢		-		1	5 3	Ł			Ð,	e					
v:	xlan																					×	3	- +	
No.	254 255 256 258 259 261 262 262 ame thern tern tern trua Flag	Time 106.4 106.4 107.5 107.5 108.5 108.5 108.5 108.5 11: 14 et II et Pr atagr 1 ext gs: 0:	3205 9598 9603 1999 2005 4399 4405 4399 4405 8 by , Sr otoc am P ens1 (0800	2431 0407 0490 2978 0755 1786 1058 tes o c: 76 ol Ve rotoc ble D, VX	Sour 00:6 10.6 10.6 10.6 10.6 10.6 10.6 10.6 10.	ce 10:00 1.0.2 1.0.1 1.0.2 1.0	0_00: 2 2 1184 9:52 Src Port a Ne ork 1	bit :77 : 19 : 46 Ewor	01 (76 02.10 0742, VNI)	De 000 100 100 100 100 100 100 100 100 10	estir 2:00 2:0. 2:0. 2:0. 2:0. 2:0. 2:0. 2:0.	nati):0(.0 .0 .0 .0 .0 .0 .0	on 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	:00:0 77), 192. 789	92 ed (1 Dst: 168.	Protocc ARP ICMP ICMP ICMP ICMP ICMP ICMP ICMP 2:b7: 2.10	ol (5) (2)	engt 92 148 148 148 148 148 148 148	h Info 2 10. 3 Ech 3 Ech 3 Ech 3 Ech 3 Ech 3 Ech 1:ac) 0. 10 10 10 10 10 10	0.1 i (ping (ping (ping (ping (ping (ping ce r1- 92:b7:	s at 0) requ) rep1) requ) rep1) requ) rep1 eth0, 2b:4a	0:00 est y est y id :51:	0:00 ic ic ic ic ic ac)	
	VXL	AN Net	tworl	k Ide	ntif.	ier	(VNI): 2	00																
4	·	erveu			550																			1	
0000 0010 0020 0030 0040 0050 0050 0050 0060 0070 0080 0090	92 00 02 c8 45 0a 00 14 24 34	b7 21 86 66 0a 9f 00 00 00 00 00 00 15 16 25 26 35 36	4a 26 00 54 02 00 17 27 37	51 ac 40 00 12 b5 00 00 d9 91 08 00 41 6a 18 19 28 29	76 3f 00 00 40 6f 00 1a 2a	8a 11 72 02 00 f0 00 1b 2b	d0 19 48 30 00 00 40 00 01 11 00 00 1c 10 2c 20	9 52 8 c0 9 08 9 00 1 4d 9 00 d 1e d 2e	77 a8 00 00 15 d8 00 1f 2f	08 0 00 0 00 0 00 0 b5 8 10 1 20 2 30 3	00 (00 (00 (00 (00 (00 (00 (00 (45 c0 00 08 00 cc 12 22 32	00 a8 00 00 01 5e 13 23 33	+. • n • · · · · E · · · \$%& 456	JQ v @ ? ↓ r Γ @ Aj '()*+	Rw H8 @ M !	E 								

Figure 45. Verifying VXLAN network identifier.

Step 6. To stop packet capturing, click on the red button located on the upper left-hand side.

6												*r	1-6	ethO	W.								a x
File	Edit	Viev	v <u>G</u>	io <u>c</u>	aptu	re	Ana	lyze	St	atis	tics	Т	ele	phon	<u>w</u> i	rele	ss <u>T</u> ools	s <u>H</u> el	р				
		J	۲				X	6	(٩	¢		•		•	5			Ð,		2		
() v	dan																					$\times \rightarrow$	• +
No.	m	Time	-		Sou	rce					De	stin	atio	on			Protocol	Lengt	th Info		-		-
1	1283	533.50	9404	3402	10.	0.0	.2				10	.0.	0.1				ICMP	14	8 Echo	(pin	g)	reply	ic
1	1284	534.5	2802	2070	10.	0.0	.1				10	.0.	0.2				ICMP	14	8 Echo	(pin	g)	request	10
	1285	534.5	2807	9032	10.	0.0	.2				10	.0.	0.1				TCMP	14	8 Echo	(pin	g)	reply	10
	1287	535 5	5200	8410	10.	0.0	2				10	0	0.2				TCMP	14	8 Echo	(pin	g)	renly	ic
	1288	536.5	7599	2209	10.	0.0	.1				10	.0.	0.2				ICMP	14	8 Echo	(pin	a	request	ic
3	1289	536.5	7604	9459	10.	0.0	.2				10	.0.	0.1				ICMP	14	8 Echo	(pin	g)	reply	ic
																		-					-
•																							•
▶ Fr	ame :	L: 148	3 by	tes d	on wa	ire	(118	34 b	its), :	.48	by	tes	capt	ured	(11	184 bits) on i	Interf	ace r:	1-e	th0, id	0 🔺
▶ Et	herne	et II,	, Sr	C: /(5:8a	:d0:	19:	52:7	1 (16:1	a:0	10::	19:	52://), D	st:	92:b/:2	b:4a:5	ol:ac	(92:b	1:2	b:4a:51	ac)
► Hs	or Da	at anra	am D	or ve	201	Src	+, 31 • Doi	c.	192	12	Det	- D/	, U ort	· 479	192.1	08.4	2.10						
- Vi	rtua	L eXte	ensi	ble l	oca.		ea l	Vetw	ork	121	031		01 6	. 413									
•	Flag	s: 0)	080	e, vx	LAN	Net	work	ID	(VI	(IV													
	Grou	ip Pol	icy	ID:	0					- 53													
	VXLA	N Net	wor	k Ide	entif	fier	(VN	II):	200	9													
	Rese	rved:	0													12							
4																							•
0000	92	b7 2b	4a	51 a	c 76	8a	d0	19	52 7	77 0	8 0	04	5 0	00	· · + JQ	٠V	· · Rw · · E						
0010	00	86 6e	ca	40 0	0 3f	11	48	38	c0 a	a8 0	10	a c	0 a	18	• • n • @	. ? .	H8 · · · · ·						
0020	02	0a 9f	26	12 b	5 00	72	00	00	08 0	00 0	0 0	0 0	0 0	00	&.	۰·r	* * * * * * * *	÷.)					
0030	C8	00 00	00	00 00	00 00	02	00	00	00 0		00	10	0 8 6	00	- · · · ·		A M	*					
0040	45		02	08 0	1 40 9 6f	f0	40	1h	4u 1 94 c	18 h	a U 5 9	0 0	CE	1	E	.6	@.m	^					
0060	00	00 00	00	41 6	a 00	00	00	00	00 0	0 1	0 1	1 1	2 1			1							
0070	14	15 16	17	18 1	9 1a	1b	10	1d	1e 1	lf 2	0 2	1 2	2 2	3				#					
0080	24	25 26	27	28 2	9 2a	2b	2c	2d	2e 2	2f 3	0 3	1 3	2 3	33	5%&'()*+	, /012	3					
0090	34	35 36	37												4567			24					

Figure 46. Stopping packet capturing.

Step 7. In container d1, press ctrl+c to stop the test.

Step 8. In container d2 terminal, re-enable the network interface in host h2 by issuing the following command:

Figure 47. Enabling interface h2-eth0.

Step 9. Perform a connectivity test between h1 and h2 by issuing the following command:

h1 ping 10.0.0.2

30	Ê			root@d1: ~
mi	ninet>	h1 pi	ing 10.0.0	.2
PI	NG 10.0	0.0.2	(10.0.0.2)) 56(84) bytes of data.
64	bytes	from	10.0.0.2:	<pre>icmp_seq=4 ttl=64 time=0.338 ms</pre>
64	bytes	from	10.0.0.2:	<pre>icmp_seq=5 ttl=64 time=0.276 ms</pre>
64	bytes	from	10.0.0.2:	icmp_seq=6 ttl=64 time=0.154 ms
64	bytes	from	10.0.0.2:	icmp_seq=7 ttl=64 time=0.130 ms
64	bytes	from	10.0.0.2:	icmp_seq=8 ttl=64 time=0.117 ms
64	bytes	from	10.0.0.2:	icmp_seq=9 ttl=64 time=0.125 ms
64	bytes	from	10.0.0.2:	icmp_seq=10 ttl=64 time=0.127 ms
64	bytes	from	10.0.0.2:	icmp_seq=11 ttl=64 time=0.132 ms
64	bytes	from	10.0.0.2:	<pre>icmp_seq=12 ttl=64 time=0.147 ms</pre>

Figure 48. Performing a connectivity test between host h1 and host h2.

Consider Figure 48. The results show a successful connectivity test.

Step 10. In Wireshark window, start packet capturing by clicking on the button located on upper left-hand side.

	*r1-eth0													-	ø	×									
File	<u>E</u> dit	t <u>V</u> iev	v <u>e</u>	io <u>C</u>	aptu	re	Ana	lyze	<u>S</u>	tatis	stic	s	Tel	eph	ony	Wir	eles	s <u>T</u> ools	5 <u>H</u> e	р					
		ß	0			1111	X	6		٩	<				\geq	•	<u>.</u>			Ð	6				
	vxlan																						$ \times \rightarrow$	-	+
No.		Time			Sou	rce					D	esti	inat	ion				Protocol	Leng	th Inf	0			-	-
	1347	559.1	9406	2984	10.	0.0	. 2				10	9.0	.0.	1				ICMP	14	8 Ec	ho	(ping)	reply		
	1348	560.1	2800	8572	10.	0.0	.1				10	9.0	.0.	2				ICMP	14	8 Ec	ho	(ping)	reques		
	1349	560.1	2806	8029	10.	0.0	. 2				10	9.0	.0.	1				ICMP	14	8 Ec	ho	(ping)	reply		
	1350	561.1	5205	2461	10.	0.0	.1				10	9.0	.0.	2				ICMP	14	8 Ec	ho	(ping)	reques		
	1351	561.1	5211	4173	10.	0.0	.2				10	9.0	.0.	1				ICMP	14	8 Ec	ho	(ping)	reply		
	1352	562.1	/599	7065	10.	0.0	.1				10	9.0	.0.	2				ICMP	14	8 EC	ho	(ping)	reques	1	
	1353	562.1	1605	1914	10.	0.0	.2				10	9.0	.0.	1				TCMP	14	B EC	no	(ping)	reply		-
4	1304	303.2	9000	3/80	10,1	0.0.	. 1				10	0.0		4				TCHP	14	O EC	no	(prind)	reques		
		4. 440		+			1440	34 h		、	4.40) h.			antu	rad	144	04 bita	\	inte			th0 i	4.0	
	thorn	of TT	Sr	c · 76	3 · 8 a	. dQ.	10.1	52.7	7 (76.	240 89'	d G b	·10	.52	•77)	De	(± ±	07 . h7 . 2) Uli	51.0	c (02·h7·	2h·/a·5	u u 1 ac	1
	ntern	et Pro	toc	ol Ve	ersi	on 4	L SI	C:	192	.16	8.1	1.1	0.	Dst	: 19	2.16	8.2	.10	D. 44.	01.a	~ (52.01.1		1.40	.,
1 i	Jser D	atagra	am P	roto	:01.	Src	Por	rt:	407	42.	Ds	st	Por	t:	4789										
- 1	/irtua	1 eXte	ensi	ble l	oca.	l Ar	ea I	Vetw	ork																
	> Fla	gs: 0)	080	D, VX	LAN	Net	work	ID	(V	NI)															
	Gro	up Pol	icy	ID:	0																				
	VXL	AN Net	worl	k Ide	entif	fier	(VN	I):	20	0															
	Res	erved:	0																						-
4	2002 C		-		-	-	-			-	-	-		-	-	1.12	2		-	-					
000	0 92	b7 2b	4a	51 a	c 76	8a	d0	19	52	77 (98	00	45	00	(E.)	+J0 ·	V ·	· · Rw · · E	6.						
001	0 00	86 6e	ca	40 0	0 3f	11	48	38	c0 ;	a8 (91	0a	c0	a8		n à.	?.	H8 · · · ·							
002	0 02	0a 9f	26	12 b!	5 00	72	00	00	08	00 (90	00	00	00		.8.	٠r								
003	0 c8	00 00	00	00 00	00 0	02	00	00	00	00 (90	01	08	00	÷				ě.						
004	0 45	00 00	54	d9 9:	1 40	00	40	01	4d :	15 (Ja	00	00	01	E	·T··	0.	@ · M · · · ·	6						
005	0 0a	00 00	02	08 0	0 6f	f0	01	1b	04	d8	50	80	СС	5e			۰0		٨						
006	0 00	00 00	00	41 6	a 00	00	00	00	00	00 :	10	11	12	13	•	··Aj	¥.18								
007	0 14	15 16	17	18 1	9 1a	1b	1c	1d	1e :	1f 2	20	21	22	23	1		11	· · · · [";	#						
008	0 24 0 34	25 26 35 36	27 37	28 2	9 2a	2b	2c	2d	2e .	2f 3	30	31	32	33	\$9 45	&'() 567	*+	,/0123	3						

Figure 49. Starting packet capturing.

Step 11. A notification window will be prompted. Click on *Continue without Saving* to proceed.

Figure 50. Closing without saving previous packet capture.

Step 12. Verify that the VXLAN Network Identifier is 100. Notice that such tag corresponds to the traffic from h1 to h2.

	Capturing from r1-eth0														-	a x					
<u>F</u> ile	<u>E</u> dit	Viev	v <u>G</u>	o <u>C</u>	aptur	e <u>A</u>	nalyz	e	Stati	stics	ίТ	elep	hony	Wire	less	Tools	i <u>H</u> elp)			
			۲) [5	<	2	٩	¢		۲						•			
v	dan																			$\times \rightarrow$	• +
No.	26 28 29 31 32 33 34 ame 5 herro	Time 9.689- 10.71: 10.71: 11.70: 11.70: 11.73 11.73 11.73 2: 148 et II, et Pro	43997 33943 34564 55792 59717 73702 74220 3 byt	79 845 126 260 761 217 580 :: 76	Sour 10.0 10.0 00:0 00:0 10.0 10.0 10.0	re (d0:1	_00:0 _00:0 1184 9:52:	00:0 00:0 bit 77	s), (76:	De 10 10 00 00 10 10 10 148 8a: 8a:	stin .0.0 .0.0 :00 .0.0 .0.0 .0.0 .0.0	atio 0.1 0.2 0.1 :00 :00 0.2 0.1 tes 19:5	n _00:0 _00:0 capti 52:77	0:02 0:01 ured (), Dst	Pr I(I(Al Al I(I(1184 : 92	totocol CMP CMP CMP RP RP CMP CMP CMP CMP	Lengt 148 148 92 92 148 148) on i b:4a:5	r Info 3 Echo 3 Echo 3 Echo 2 Who 2 10.0 3 Echo 3 Echo 1:ac	(ping) (ping) has 10. .0.2 is (ping) (ping) ace r1- (92:b7:	reply request reply 0.0.2? To at 00:00 request reply eth0, id 2b:4a:51:	ic ic ic ell 0:00 ic ic ic ic ac)
Vi	er Da rtua Flag Grou VXL/ Rese	atagra 1 eXte js: 0x up Pol AN Net erved:	am Pr nsil 0800 icy work	otoc le l , VX ID: Ide	ol, ocal LAN 0 ntif	Src Are Netw ier	Port: a Net ork I (VNI)	40 wor D (' : 10	742, k VNI) 00	Ds	t Po	ort:	478	9							
 0000 0010 0020 0030 0040 0050 0060 0070 0080 0090 	92 00 02 64 45 0a 00 14 24 34	b7 2b 86 0e 0a 9f 00 00 00 00 00 00 15 16 25 26 35 36	4a 13 26 54 02 00 17 27 37	51 ac 40 00 12 b5 90 00 a5 24 98 00 7c 59 18 19 28 29	: 76) 3f 5 00) 00 1 40) ba) 08) 1a) 2a	8a 11 72 02 00 3a 00 1b 2b	d0 19 a8 ef 00 00 00 00 40 01 01 1d 00 00 1c 1d 2c 2d	52 c0 08 00 81 00 1e 2e	77 a8 00 82 99 00 1f 2f	08 0 01 0 00 0 00 0 2c 8 10 1 20 2 30 3	00 4 0a c 00 0 01 0 00 0 34 c 11 1 21 2 31 3	5 0 0 0 0 0 8 0 0 0 5 2 1 2 2 3	0 · · · · · · · · · · · · · · · · · · ·	+JQ · v @ · ? & T · \$@ Y · %& ' () ^ 567	r	Rw E	• • • •				

Figure 51. Verifying VXLAN network identifier.

This concludes Lab 5. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

- 1. Mininet walkthrough, [Online]. Available: http://mininet.org.
- 2. Peuster, Manuel, Johannes Kampmeyer, and Holger Karl. "Containernet 2.0: A rapid prototyping platform for hybrid service function chains." 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). IEEE, 2018.
- 3. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "*OpenFlow: enabling innovation in campus networks*." ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.
- 4. P. Goransson, C. Black, T. Culver. "Software defined networks: a comprehensive approach". Morgan Kaufmann, 2016.
- 5. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, "ONOS: towards an open, distributed SDN OS," In Proceedings of the third workshop on Hot topics in software defined networking, pp. 1-6, 2014.
- 6. Mahalingam, Mallik, et al. "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks." RFC 7348 (2014): 1-22.
- 7. Juniper Networks, "Understanding EVPN with VXLAN Data Plane Encapsulation", [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics /concept/evpn-vxlan-data-plane-encapsulation.html.

- 8. Qu, Xiaorong, Weiguo Hao, and Yuanbin Yin. "*L3 gateway for VXLAN.*" U.S. Patent No. 8,923,155. 30 Dec. 2014.
- 9. Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." *Linux journal* 2014.239 (2014): 2.
- 10. Linux foundation collaborative projects, "FRRouting: what's in your router", [Online]. https://frrouting.org/