NETWORK SECURITY

LINUX BRIDGE

LINUX BRIDGE LAB

Overview: Virtual networking is one of the key enablers of enterprise and cloud networking. When dealing with virtual networks, we usually need virtual switches for creating complex network topology. Virtual switching solutions like Software Defined Networking (SDN) has gained lots of attention recently for its strength in managing large and ever-changing network infrastructures. But long before SDN came into existence, Linux bridge was the prevalent virtual switching tool, and it still is nowadays. A bridge, just like a physical switch, is a device to unite network segments transparently by forwarding the network traffic from the network devices connected to it. In this lab, we will first go through some Linux networking concepts and tools which can later help us understand the way Linux bridge works. Then, we will look into Linux bridge from both application and implementation perspective.

Objectives: Upon completing the experiments, students should:

- Understand the basic concepts of Linux networking;
- Get familiar with some popular Linux networking tools used to analyze network structure and traffic;
- Learn to use Linux bridge tools to manage Linux bridges, and understand the how the Linux bridge works.

Required Knowledge:

Students are required to have the following prerequisite knowledge for this lab:

1. The usage of the text-based terminal and basic Linux commands;
2. Knowledge of accessing a remote Linux server using SSH;
3. Basic understanding of the use of common network devices such as routers and switches.

Background:

EZSetup is a novel Web application capable of creating a variety of user-defined cybersecurity practice environments (e.g., labs and competition scenarios) in one or more computing clouds (e.g., OpenStack or Amazon AWS). EZSetup provides a Web user interface for practice designers to create a practice scenario by dragging and dropping icons visually and the links between them thus allows for customization and significantly reduces overhead in creating and using practice environments. Completely spared from the complexity of creating practice environments, end users can jump right in and fully concentrate on cybersecurity practice.

For more information about EZSetup, please see bundled user manual file.

Secure Shell (SSH) is a network protocol that allows data to be exchanged over a secure channel between two computers. Encryption provides confidentiality and integrity of data. SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

For more information about SSH, please see https://wiki.archlinux.org/index.php/Secure_Shell

tcpdump is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. Distributed under the BSD license, tcpdump is free software.

For more information about tcpdump, please see its manual page: https://www.tcpdump.org/manpages/tcpdump.1.html For tcpdump and wireshark filtering syntax, please see http://www.tcpdump.org/manpages/pcap-filter.7.html

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark is very similar to

tcpdump, but has a graphical front-end, plus some integrated sorting and filtering options.

For more information about Wireshark, please see its User's Manual: https://www.wireshark.org/docs/wsug_html_chunked/

Environment:

In this lab, students can access two virtual machines (VM) from EZSetup. The first one is for Linux bridge experiments, and the second one is for observing the network traffic. The network topology is given by Figure 1, VM properties and access information are provided in Table 1 below. Please refer to the EZSetup dashboard for the actual public IP addresses and passwords.

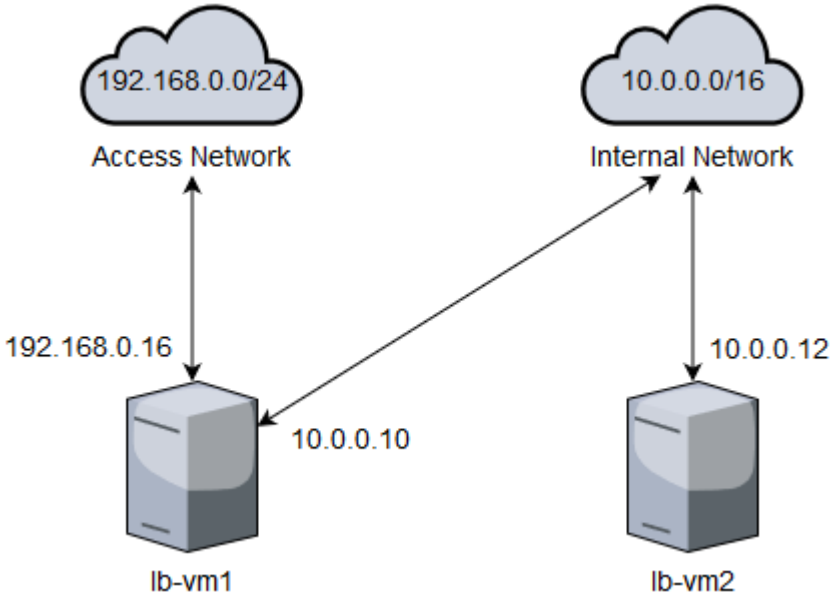

Figure 1 Lab network topology

Table 1 VM properties and access information

Name	Image	RAM	VCPU	Disk	Login account
lb-vm1	linuxbridge-vm-1	2GB	2	40GB	See EZSetup
lb-vm2	linuxbridge-vm-2	2GB	2	40GB	See EZSetup

Task 1: Linux Networking Basics

A computer network is a telecommunication network for connected computer nodes exchanging data through data links. When talking about the composition of a computer network, people usually use conceptual models to describe the characteristics of the network and its functions. Network models like IBM System Network Architecture (SNA), Open Systems Interconnection (OSI) model, and Internet protocol suite (TCP/IP) model make it easier for people to standardize diverse communication systems and protocols by abstract the network into several layers.

The seven layers of an OSI model, as described in ISO/IEC 7498-1, are listed below in Table 2. These layers are identified by their functions and roles in data processing, like dealing with electric signals or routing data across the network. Each layer provides necessary interfaces to its upper layer, the higher the layer, the more abstract the data representation will be.

Table 2 Layers of OSI Model

Layer	Function	Example protocols
7. Application	Provide an interface to end users or other applications like web browser, and electronic mail	HTTP, HTTPS, IMAP, AMPQ
6. Presentation	Data translation and formatting between network services and applications like data compression and decryption	TLS, SSL, AFP
5. Session	Manage dialogues between local and remote applications like connection establishment and termination	SOCKS, SSH, RPC
4. Transport	Provide data segmentation, acknowledgment mechanisms for transmitting data segments reliably over layer 3 network protocols	TCP, UDP, iSCSI
3. Network	Provide a logical addressing and routing system to transfer data over multiple layer 2 networks	ICMP, IPv4, IPv6

2. Data link	Handle node-to-node data transfer connected to a physical layer	ARP, RARP, VLAN
1. Physical	Define physical network devices and transmit bit streams over them	USB, Wi-Fi, Bluetooth

A network interface is a software representation of physical or virtual network device. On Linux systems, you can find available network interfaces in `/proc/net/dev`, and access device configuration and statistics under `/sys/class/net/` folder. On Debian-based Linux systems like Ubuntu and Linux Mint, you can also configure network interfaces in `/etc/network/interfaces` and files under `/etc/network/interfaces.d/` folder. If you only want a quick look at the interfaces you have, please use the `ifconfig` command to list all interfaces along with their detailed information.

```
$ ifconfig -a
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:160 errors:0 dropped:0 overruns:0 frame:0
            TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)
```

We can see from the output that network interface names are listed in the left column, configurations and statistics are in the right. For each interface, ifconfig will give its device type (like Ethernet or Local Loopback), hardware address (or MAC address), IPv4 and IPv6 addresses, a network mask of each address, running status, input/output packet numbers and sizes.

Since the ens4 interface is down. let's set the IP address and bring it up following the command lines below.

```
$ sudo ip addr add 10.0.0.10/16 dev ens4
```

```
$ sudo ifconfig ens4 up
```

Like sending a mail to someone, if we want to transmit data to a network interface over a network, we need some addressing mechanisms. A MAC address, or media access control address, is a unique identifier assigned to network interfaces for data link layer (layer 2) communications in a network segment, which consists of 48 bits or 6 octets written in hexadecimal and separated by colons. By looking up the MAC address, network devices like switch or router can forward the data to the right receiver interface. However, MAC addresses are typically used only to direct data from one device to its next device and will be replaced by the device which forwards your data. This means MAC addresses are usually used in the local network and do not travel far. Also, the MAC address binds to the physical network interface, so it will be difficult for others to find you if you switch network interface frequently.

That's where Internet Protocol (IP) addresses come into play. An IPv4 (version 4) address is a 32-bit number, which is usually written in human-readable notations with four 8-bit decimal numbers separated by dots. An IPv6 (version) address is much longer, using a 128 bits number and usually written in hexadecimal with colon separators. IP addresses work in the network layer (layer 3) and are required most of the time if a host wants to communicate with other hosts. IP addresses within a segment of network, or subnet, start with the same prefix, like 192.168.0.1 and 192.168.0.2, where 192.168 is the common prefix. To define the prefix and the size of the subnet, people often use netmasks or CIDR (Classless Inter-Domain Routing) notations. For example, for a subnet 192.168.0.0 to 192.168.255.255, its netmask is 255.255.0.0 and CIDR is 192.168.0.0/16. Here, the netmask defines how many bits from the left in an IP address is the prefix, and we can do the bitwise AND operation of the IP address on the netmask to get the subnet prefix. CIDR is just a compact representation of the count of leading 1 bits in the prefix.

```
          11000000.10101000.00000001.00000010    192.168.1.2
AND       11111111.11111111.00000000.00000000    255.255.0.0
------------------------------------------------------------
=         11000000.10101000.00000000.00000000    192.168.0.0
```

An example of sending a packet from your personal computer to a remote web server is shown below in Figure 2. During the transmission, we also need to find the physical location (MAC address) of a machine by its IP address, which can be achieved by using Address Resolution Protocol (ARP). You probably see an ARP request and response when you initiate a connection to another machine, which you haven't connected before.

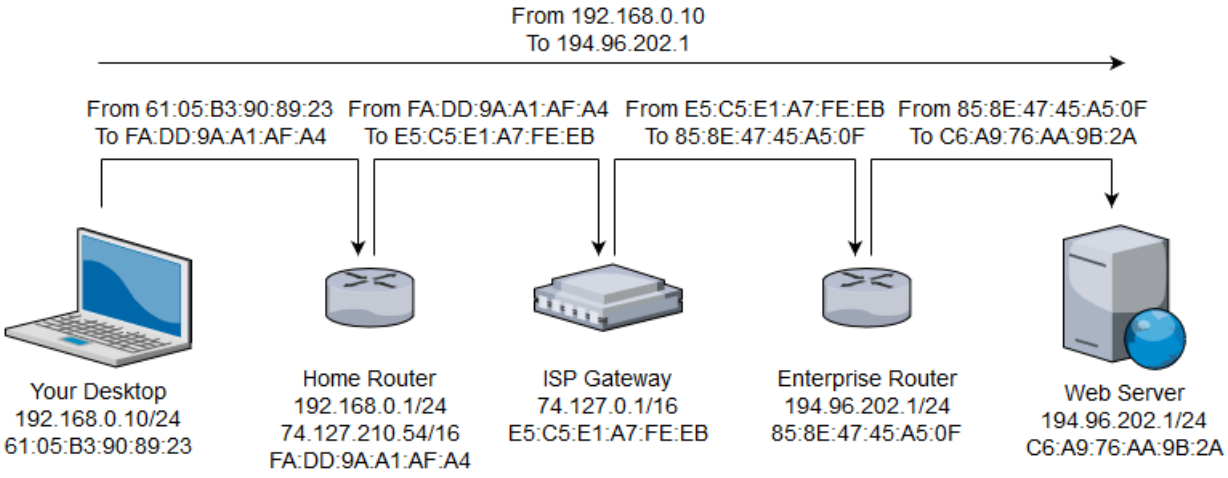

Figure 2 Packet flow from user to the remote server

Linux provides a set of network tools for developers to test their network setups. To view interface information, we can use the `ip` command apart from the `ifconfig` command we talked about earlier.

```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1
      link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
      inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
      inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
```

To test the connectivity between your machine and another host, you can use the `ping` command to send Internet Control Message Protocol (ICMP) packets.

```
$ ping www.google.com -c 3
```

```
PING www.google.com (74.125.30.103) 56(84) bytes of data.
64 bytes from of-in-f103.1e100.net (74.125.30.103): icmp_seq=1 ttl=42
time=36.4 ms
64 bytes from of-in-f103.1e100.net (74.125.30.103): icmp_seq=2 ttl=42
time=34.9 ms
64 bytes from of-in-f103.1e100.net (74.125.30.103): icmp_seq=3 ttl=42
time=34.2 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 34.248/35.211/36.429/0.933 ms
```

Sometimes, you may want to learn more about what's going on under the hood, such as, what the network frames or packets are like when you visit a website. You can use a sniffer tool to intercept and filter the network traffic passing through your interfaces. A command-line tool `tcpdump` is used for the network traffic sniffing. You can assign the interface you want to inspect, and a set of rules to match the packets. For example, if you want to listen on interface ens3, you can run the following command

```
$ sudo tcpdump -i ens3
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on ens3, link-type EN10MB (Ethernet), capture size 262144
bytes
05:53:33.604965 IP 192.168.0.16.ssh > 192.168.97.6.57405: Flags [P.],
seq 241650537:241650645, ack 3972362028, win 459, length 108
05:53:33.605016 IP 192.168.0.16.ssh > 192.168.97.6.57405: Flags [P.],
seq 108:144, ack 1, win 459, length 36
05:53:33.605071 IP 192.168.0.16.ssh > 192.168.97.6.57405: Flags [P.],
seq 144:252, ack 1, win 459, length 108
05:53:33.605106 IP 192.168.0.16.ssh > 192.168.97.6.57405: Flags [P.],
seq 252:288, ack 1, win 459, length 36
```

The sudo command above is to make sure you have enough privilege to execute the following command. If you are only interested in a particular

set of packets, e.g., a packet to www.google.com, you can apply some filter rules behind the `tcpdump` command.

```
$ sudo tcpdump -i ens3 dst host www.google.com
```

For a comprehensive list of filter syntax, you can refer to the pcap-filter man-page listed in the "References" section. If you like to work in a desktop environment, Wireshark is also a great choice. Wireshark is a network protocol analyzer which lets you live capture network traffic and provide easy-to-use offline analysis functions. First, let's open Wireshark by typing the command below in terminal window.

```
$ sudo wireshark-gtk
```

Or, you can start Wireshark by right-clicking on your VNC screen and choose the Applications > Internet > Wireshark.

Before starting the capture, we need to tell Wireshark which interfaces we want to listen on. Clicking the first button in the toolbar can check one or more interfaces in the interface selection dialog. Then, press the "start" button to initiate the capturing process.

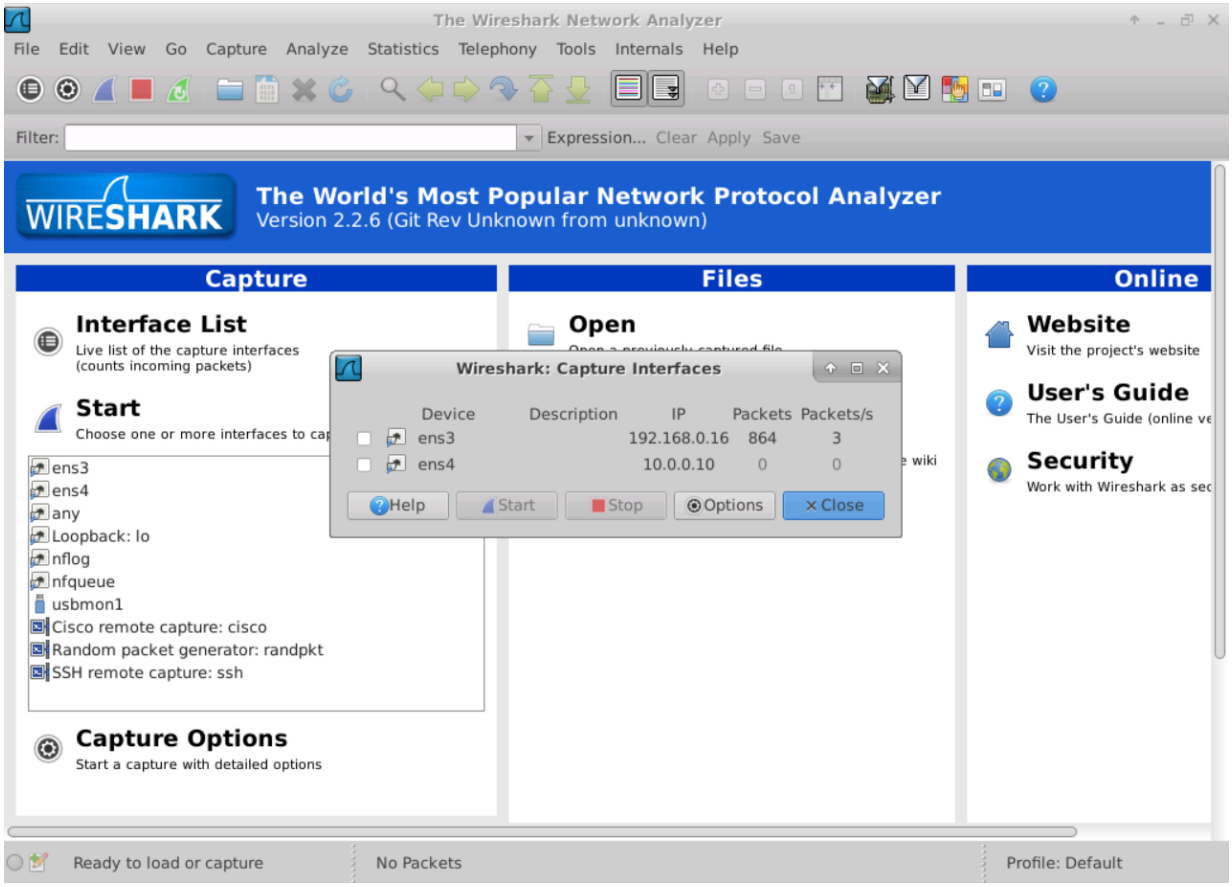

Figure 3 Interface selection dialog in Wireshark

Once the capture begins, we can see all the network traffic in the packet list pane. There are several columns available, including the packet source and destination IP or MAC address, the protocol used and the packet length. Each line corresponds to one packet in the capture file or stream. If you select a line in this pane, packet details will be displayed in the packet detail and packet bytes panes. In the packet detail pane, protocols used by the packet will show in an expandable tree, with lower layer protocols to the top and higher layer protocols to the bottom. You can click on any protocol to view its fields and values, or raw bytes in the packet bytes pane.

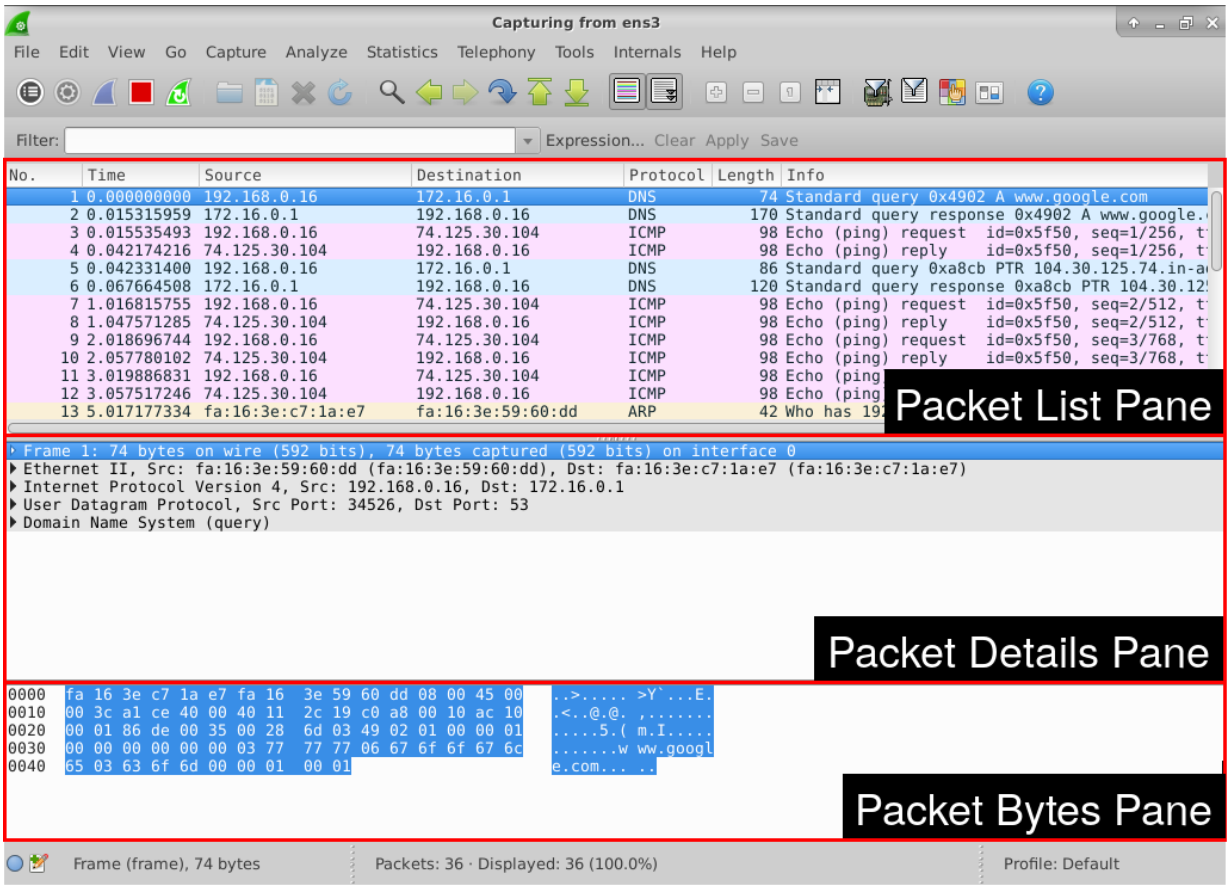

Figure 4 Wireshark network traffic capturing window

LAB EXERCISE 1

Log in to the first VM (lb-vm1) and answer the following questions.

1. What are the interfaces on your VM? What's their MAC and IP addresses? What's their MTU (Maximum Transmission Unit) value?
2. What are the netmask and IP range for the subnet 10.0.0.0/23?
3. Can you connect to the host 192.168.0.1? What about 192.168.0.9? What command do you use? If you cannot connect to a host, what does the reply say?
4. Before we move on, clear **all** of your browser history by opening the Application Menu in Firefox > History > Clear Recent History... > Set time range to clear: Everything, and check all boxes. Close your

browser when this is completed. Now, open Wireshark and start capture on the ens3 interface. Then visit http://ualr.edu/. How do you filter out all the HTTP request to ualr.edu? What is the IP address of the Web server? What is the "`Accept-Encoding`" field in the Hypertext Transfer Protocol in the first HTTP request? (Hint: use http.host display filter)

Task 2: Managing Linux Bridge

Bridging is "plugging" one or more network interfaces into another network interface which has the connectivity to a larger network. Just imagine the dorm room in your college only has a limited number of ethernet jacks and no router, so you and your roommate can bridge your computers to a single ethernet port to join the Internet. The job of the bridge is to examine the data packet destination and decide where should it be passed to, or whether it should drop the packet, by looking at the MAC address of plugged network interfaces.

Linux bridge is a software bridge which emulates the behavior of a hardware bridge and provides additional powerful features like firewalling. You can check if your system has the bridge kernel module by the following command

```
$ modinfo bridge
filename:       /lib/modules/4.4.0-97-
generic/kernel/net/bridge/bridge.ko
alias:          rtnl-link-bridge
version:        2.3
license:        GPL
srcversion:     E093FB4EB1EE49C1B0F6A2F
depends:        stp,llc
intree:         Y
vermagic:       4.4.0-97-generic SMP mod_unload modversions
```

If it shows no errors, you can manage bridges either using the `brctl` command in the bridge-utilities package or the `ip` command. We will be using `brctl` in the rest of the instructions.

First, let's see all subcommands `brctl` offers.

```
$ brctl
Usage: brctl [commands]
commands:
    addbr       <bridge>             add bridge
    delbr       <bridge>             delete bridge
    addif       <bridge> <device>    add interface to bridge
    delif       <bridge> <device>    delete interface from bridge
    hairpin      <bridge> <port> {on|off}    turn hairpin on/off
    setageing    <bridge> <time>      set ageing time
    setbridgeprio    <bridge> <prio>  set bridge priority
    setfd       <bridge> <time>       set bridge forward delay
    sethello    <bridge> <time>       set hello time
    setmaxage    <bridge> <time>      set max message age
    setpathcost  <bridge> <port> <cost>      set path cost
    setportprio  <bridge> <port> <prio>      set port priority
    show         [ <bridge> ]         show a list of bridges
    showmacs     <bridge>             show a list of mac addrs
    showstp      <bridge>             show bridge stp info
    stp          <bridge> {on|off}    turn stp on/off
```

To create a bridge, you can use the `addbr` subcommand followed by the bridge name

```
$ sudo brctl addbr br0
```

The above command will add a bridge named `br0`. The bridge will show as a network interface, and you can check it by command `ifconfig br0`, or you can show all the bridges using

```
$ brctl show
bridge name    bridge id           STP enabled    interfaces
br0            8000.000000000000    no
```

Deleting a bridge is just as simple as creating a bridge. You can use brctl delbr with the bridge name

```
$ sudo brctl delbr br0
```

Now, let's try to plug an interface to the bridge. **Please be careful with the interface name when you enter the command.** Adding an interface to a bridge will make the interface lose network connection temporarily, and the first ethernet interface, which is ens3, is the channel for the VM to communicate with you. So, make sure you don't operate on the ens3 interface, or you will lose the connection to your VM. Enter the following command to add the second ethernet interface ens4 to bridge br0

```
$ sudo brctl addif br0 ens4
```

You can check the bridge details once again using brctl show. Also, if you want to remove an interface from a bridge, you can run the following command

```
$ sudo brctl delif br0 ens4
```

After you add the interface to the bridge, we may test the interface connectivity to its subnet by ping the other VM (lb-vm2)

```
$ ping -I ens4 10.0.0.1 -c 3
PING 10.0.0.1 (10.0.0.1) from 10.0.0.10 ens4: 56(84) bytes of data.
From 10.0.0.10 icmp_seq=1 Destination Host Unreachable
From 10.0.0.10 icmp_seq=2 Destination Host Unreachable
From 10.0.0.10 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time
2014ms
pipe 3
```

Right now, you should receive "Destination Host Unreachable" error, which means ens4 cannot send or receive packets from other hosts on the same subnet. This can be explained with the first two figures in Figure 5.

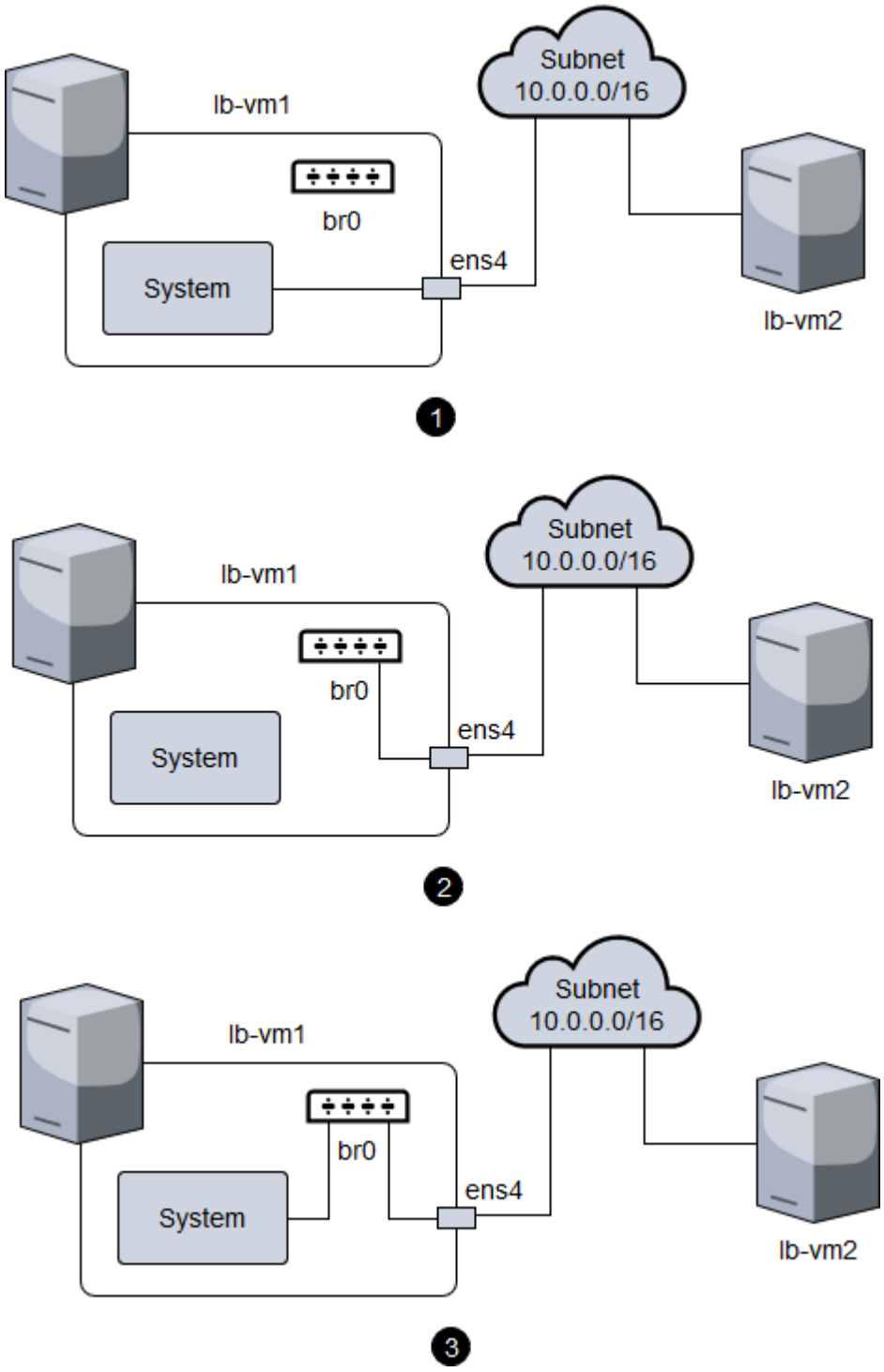

Figure 5 Steps of connecting a virtual host to the Linux bridge

At first, the system connects to the subnet via ens4 directly. When we add

ens4 to the bridge, we cut down the connection between the system and ens4, and right now the system still tries to send packets to ens4 instead of br0. To re-establish the connection, we should tell the system to send all traffic within the subnet to br0. To do this, we need to move the IP address of ens4 to br0 and bring up the bridge interface.

```
$ sudo ip addr del 10.0.0.10/16 dev ens4
$ sudo ip addr add 10.0.0.10/16 dev br0
$ sudo ip link set br0 up
```

Now, you should be able to ping other hosts in the subnet. Let's try to add some more interfaces to the bridge and see if it works. Move to the ~/labs/linux_bridge directory and run the following command to add two virtual hosts h1 and h2 to your VM.

```
$ ./start.sh
=======================================
        Virtual Hosts Activated!
           Type exit to quit.
=======================================
```

You can use "<host name> <command>" to execute network-related commands as if they are executed in another machine. No sudo is need here. For example, you can list interfaces of h1 using

```
$ h1 ifconfig -a
```

Or ping another host using

```
$ h2 ping 10.0.0.12 -c 3
```

Note here, however, that you should still receive "Destination Host Unreachable" because although running ./start.sh has added h1 and h2 to the VM, we still need to connect them to the subnet.

Also, you can still execute commands in the host VM by simply typing commands without host name.

If you list interfaces on the host VM, you will find there are two new interfaces added, which are `p-h1` and `p-h2`. Each of the `p-*` interfaces are one end of a tunnel which connects the virtual host and the VM. Packets from the virtual hosts can reach the VM through this tunnel, and vice versa. Figure 6 depicts current network topology.

Now, to connect h1 and h2 to the subnet, we need to add the VM side `p-h1` to the bridge. This should be done in the Virtual Host application.

```
$ sudo brctl addif br0 p-h1
```

This will connect `p-h1` to the bridge br0, and also make it possible for virtual host h1 to communicate with other hosts in the subnet. Now you should be able to ping other hosts using the command listed above.

Remember to use the exit command once you are done with tinkering the virtual hosts.

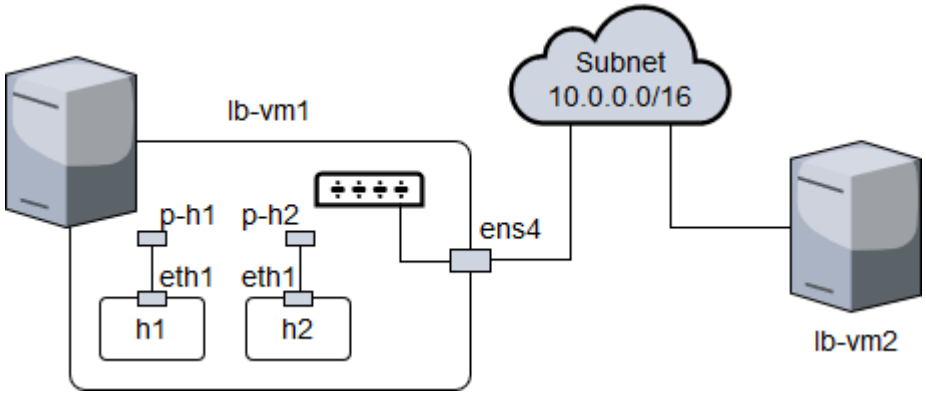

Figure 6 Network structure of the host VM and virtual hosts

LAB EXERCISE 2

Activate virtual hosts using the `start.sh` script and try to connect both hosts to the network. Then answer the following questions.

1. What's the MAC address of the bridge interface?

2. What commands do you use to connect the virtual hosts to the network? What are the IP and MAC addresses of the virtual hosts?
3. Open Wireshark on the second VM (lb-vm2) and listen on the ens3 interface. Then, try to ping it from virtual host h1. What's the IP and MAC address of the incoming packets do you see in Wireshark? What does this mean?
4. On the first VM (lb-vm1), use tcpdump to capture traffic on ens4. Then, try to ping the second VM (lb-vm2) and h2 from h1 successively. What commands do you use? What do you observe and what does it mean?

WHAT TO SUBMIT

Save your answers (with screenshots) to the above questions into a PDF file and name the file as "linux-bridge-ans.pdf".