Lab 3
Linux Networking Tools

Prof. Kredo

Due: Start of lab Friday, February 13

Name:	
Name:	

Introduction

In this lab you will accomplish several goals:

- Learn some common Linux networking diagnostic tools
- Build a small network and measure throughput

Work in pairs for this lab using the equipment at your desk. Distribute the work evenly to make sure both group members know the material, as you will be required to know the material for evaluation.

1 Networking on Linux [60 Points]

You will now expand upon the material learned in Lab 1 and become familiar with some common network-related commands on your Linux system.

1.1 `sudo`

You will often use the command `sudo(8)` to execute a command with root (Administrator) privileges. (`sudo` itself is not a networking command.) You only need `sudo` when you change the system configuration or access protected files, so you can typically execute commands without `sudo` if those commands only display system information. The machines in lab are setup so you can only run a subset of commands through `sudo`. If you wish experiment with other commands or system configurations, you are encouraged to install Linux on a partition or a virtual machine on your personal computer.

1.2 `ip`

`ip(8)` is the main command used to configure and manage the IP networking interfaces on your Linux system. You used this command in Lab 1 to configure your network interface, but we'll look at those commands in more detail now. The first argument to `ip` is the object to configure or display. The man page lists all possible objects, but we will only use the following in this class:

- `address`
- `link`
- `route`
- `neighbor`

Let's look at each of the commands used in Lab 1 to configure your networking interface. We'll cover `neighbor` objects in later labs. As you read through these instructions, enter the commands to configure your system.

1.2.1 `ip address add <IP>/16 dev eth0 broadcast +`

This command `add`s the `address` specified to the `dev`ice named `eth0` and sets the `broadcast` address appropriately, where `<IP>` is the address you wish to use. We haven't covered addresses yet, so don't worry if you don't understand the address or the format. You can also `change` and `del`ete addresses from interfaces. See `ip-address(8)` for more information on `address` objects of `ip`.

1.2.2 `ip link set dev eth0 up`

This command `set`s the `link` (`link` in this context is another name for interface) of a `dev`ice named `eth0` in the `up`, or on, state. By default, network interfaces are disabled (down) on our Linux systems, so you need to use this command to allow the OS to send and receive packets on the interface. You can find all the other `link` settings at `ip-link(8)`.

1.2.3 `ip route add default via 10.11.0.1`

This command `add`s the `default route` to (`via`) the device at address `10.11.0.1`. We haven't discussed routes or default routes yet, but they describe how packets should move through the network on their way to the destination. Without a default route your system would only be able to communicate with neighboring devices. `ip-route(8)` has more information on how you can modify routes in your system.

1.3 `ping`

The `ping(8)` command tests whether a path exists between a local device and a remote device by sending a short message. If the remote device receives the message, it responds by sending a return message to the local device. `ping` also keeps track of the time it takes to send the message and receive the reply, called the round trip time (RTT). You use the `ping` command by giving it a name or address to attempt to reach.

Ping the loopback address of your host by running:

```
ping -c 5 127.0.0.1
```

What is the minimum, average, and maximum delay to the loopback address?

How many pings were sent with the above command? Using the man page if necessary, write the command you would use to ping the loopback address with 20 ping messages.

Connect your host to the external network by using a patch cable. Verify you have joined the network by pinging the device at the address 10.11.0.1. What is the delay to that machine?

`ping` can also take hostnames as arguments. Pick some hostname (e.g., www.example.com) and ping the host. What hostname did you pick? What were the delay times?

1.4 `traceroute`

`ping` can tell you if you can reach another device, but it doesn't tell you much about how packets get there. `traceroute(8)` was designed to provide this information. It takes the same arguments as `ping` and displays which devices your packets traverse as they progress toward the remote device.

Using the hostname you selected in the previous question, use `traceroute` to determine the path to the host. `traceroute` may take a few seconds to find the complete path. If it gets stuck for longer than 10 seconds or produces many lines without information, then pick another hostname and try again.

What host did you use? How many intermediate devices did your packets traverse to get there?

What was the address of the host you selected?

1.5 Wireshark

The tools you've used so far send and receive packets on the network, but you will often want to examine those packets to more easily study running protocols. Wireshark allows you to do this by capturing all the packets in your part of the network and displaying them to you.

Open Wireshark and begin capturing packets by clicking on the `eth0` interface link under the Capture section in the upper-left portion of the application.

You should see an window with two main sections. The upper part lists the packets that Wireshark has captured and the lower part displays the content of selected packets. Ping some devices to generate traffic. After you have captured at least 10–12 packets click the stop capture button (the one with the red x). Use Wireshark to examine the packets you captured and answer the following questions.

Find a packet sent to or from your host. How do you know it came from your host or was destined for your host?

How large are the packets you captured?

What protocols were used in those packets? (HINT: Look at the bottom section that details the packet contents.)

2 Performance Measurements with `iperf` [40 Points]

`iperf(1)` is a simple tool to measure the throughput from one device, a client, to another device, a server. You will use `ipef` to measure throughput and examine how packet size influences throughput.

2.1 Basic Throughput

Start by connecting your PCs to a switch (the devices with many ports in your rack) and running `iperf` between your hosts. On one PC run `iperf` in server mode and on the other machine run `iperf` in client mode. For the test in this section, use the default values for `iperf`.

What was your measured throughput?

`iperf` can be used to perform basic throughput measurements and also to (crudely) generate background traffic for other analysis.

2.2 Packet Size

Throughout the semester we'll discuss how packet size can affect network performance. This section has you experimentally determine how much packet size affects throughput by performing throughput measurements on a small network (PC-switch-PC) while varying the packet size. You vary the packet size in `iperf` by using the `-M` argument. We'll discuss maximum segment size later in the semester; for now you can think of it as the amount of data in each packet, ignoring all headers.

Vary the packet size from 100 Bytes to 1500 Bytes in 100 Byte increments and record your results in the table below.

Data Size	**Throughput (Mbps)**
100 B	
200 B	
300 B	
400 B	
500 B	
600 B	
700 B	
800 B	
900 B	
1000 B	
1100 B	
1200 B	
1300 B	
1400 B	
1500 B	

Using your results, explain why the throughput changes as you increase and decrease packet size.

For fun, measure the throughput between an `iperf` server and an `iperf` client running on the same machine.

Submit your completed lab handout before the next lab.

3 Lab 3 Addresses

	Host Addresses	
Desk	**PC 1**	**PC 2**
A	10.11.50.101	10.11.50.201
B	10.11.50.102	10.11.50.202
C	10.11.50.103	10.11.50.203
D	10.11.50.104	10.11.50.204
E	10.11.50.105	10.11.50.205
F	10.11.50.106	10.11.50.206
G	10.11.50.107	10.11.50.207
H	10.11.50.108	10.11.50.208

	Host Addresses	
Desk	**PC 1**	**PC 2**
I	10.11.50.109	10.11.50.209
J	10.11.50.110	10.11.50.210
K	10.11.50.111	10.11.50.211
L	10.11.50.112	10.11.50.212
M	10.11.50.113	10.11.50.213
N	10.11.50.114	10.11.50.214
O	10.11.50.115	10.11.50.215