

Lab 6: Basic Network Setup

In this lab, you will learn about basic TCP/IP networking to add your machine to the lab network. You will also learn about your machine's networking hardware and setup. Perform the indicated steps, and write the answers to questions in the appropriate spaces on this hand-out. Turn-in one completed handout per team and be sure to include everyone's name

Boot the System

Boot the system into single-user mode

Step 1. Boot your system into single user mode. To accomplish this, reboot the system, and interrupt the Red Hat splash screen at the beginning of the boot process with Control-X. Then type **linux single** at the **boot:** prompt.

Configure basic networking

Learn about the basic system networking hardware and how to configure and activate the network interface.

Step 2. To enable networking, you need a device driver for the network interface card (NIC). Your machine has a 3Com NIC, either model **3c509** or **3c59x**. The server machines (the two machines in the middle of each row of tables) have two NICs, one of each type. Examine the NICs in your system to see which one(s) you have. The NIC with the round plug-like BNC connector is the older 3c509 model. The NIC without this connector is the newer 3c59x model.

Q1. How many NICs does your system have? _____

Q2. Which NIC(s) does your system have? _____

Q3. The RJ-45 connector on the Cat 5 Ethernet cable is plugged into which NIC? _____

Step 3. Manually load the 3c509 device driver module using **insmod** (recall the module loading utilities **insmod**, **lsmod**, and **rmmmod** from last week's lab). Ensure the module is loaded with **lsmod**. If your system has two NICs, try loading the **3c59x** device driver module. If you try to load the wrong device driver module, **insmod** will fail and the device will not function. If it loads successfully, you may see some initialization messages printed out by the driver. Unload the module using **rmmmod**.

Q4. Which device driver module(s) works for your NIC? _____

Step 4. In Linux, networking devices are known as **eth0**, **eth1**, **eth2**, etc. where the first loaded device is always **eth0**, the second **eth1**, etc. But the Linux kernel needs to know the name of the driver module, not the generic **ethN** name used by many startup scripts and networking utilities. To enable the use of generic names, and to allow the module(s) to auto-load upon first use, you need to add an **alias** in the file **/etc/modules.conf**. The kernel module auto-loader then knows which specific kernel device driver module is associated with the generic networking interface name **eth0**, etc. Edit the file **/etc/modules.conf**, add the line below, save the file and exit (your **modules.conf** file may already contain this line).

```
alias eth0 module-name
```

Manually Configuring a Network Interface

*Learn how to use the **ifconfig** command to configure a network interface.*

Step 5. Network interface parameters are examined using the **ifconfig** command. Examine the first two lines of output from the command:

```
$ ifconfig eth0
```

Q5. Write down the first two lines: _____

Step 6. Network interfaces are configured with the **ifconfig** command. The **ifconfig** command accepts the various IP address parameters as arguments, and many other interface-specific values. The command below will configure your **eth0** network interface (replace the **N** with your machine number):

```
# ifconfig eth0 10.0.0.N broadcast 10.0.0.255 netmask 255.255.255.0 up
```

Step 7. Use **ifconfig** to check that the IP networking parameters are correct:

```
$ ifconfig eth0
```

Q6. Compare the first three lines of output with the previous two lines of output you recorded previously. Describe the differences: _____

Step 8. Once you are certain the network parameters are correct, test the interfaces connectivity with the **ping** command. Try to ping another IP address on the network. Try to ping the system at 10.0.0.200 as well. You are looking for the response from **ping** that indicates the host is alive. This is a good first step to test connectivity.

```
$ ping destination-ip-address
```

Step 9. IP network values can be changed using the **ifconfig** command. Change your system's IP address by adding 100 to the final number of the IP address (the **host** part). The command to use is (where *NN* is your machine number using two digits):

```
# ifconfig eth0 10.0.0.1NN
```

Step 10. The **up** argument you supplied earlier to **ifconfig** activated the interface (brought it *up*). The interface can also be disabled (brought *down*). Use the command below to disable the interface:

```
# ifconfig eth0 down
```

Step 11. Test the **ping** command once again to verify that the interface is down.

Q7. What happens when you try to ping another system? _____

Step 12. Unload the NIC's device driver module.

Configuring Red Hat's Network Startup Scripts

Learn how to setup Red Hat's network startup scripts to configure the network automatically..

Step 13. Red Hat Linux uses a series of shell scripts to configure the network and various network related files and settings. Your system's hostname will be **hostN**, where *N* is your machine's number (e.g. **host4** or **host18**). Set the hostname for the machine by editing the file **/etc/sysconfig/network** and adding the lines shown below. The **HOSTNAME** variable is used by the Red Hat Linux startup scripts to assign as the hostname during startup. The **NETWORKING** variable indicates that networking should be enabled. Make the changes, save the file and exit

```
HOSTNAME=your-hostname
NETWORKING=yes
```

Q8. What system command is ultimately responsible for setting the hostname on a UNIX/Linux system? _____

Step 14. You can also create or update the **/etc/hostname** file, a file used for backwards compatibility (it is also commonly used on other UNIX systems). The file should contain only a single line, your system's hostname.

Step 15. The network configuration files are stored in **/etc/sysconfig/network-scripts**, and the files read by the network startup scripts are named **ifcfg-ethX**, where *X* is the interface number. Change directories into **/etc/sysconfig/network-scripts** and create the file named **ifcfg-eth0** (if the file already exists, you can delete all lines, or comment them out

with a # character). This file is used to store network settings for the **eth0** interface. The **DEVICE** variable is used to specify the interface name, **NETMASK**, **NETWORK**, and **BROADCAST** are all standard IP values, and **ONBOOT** indicates that networking should be enabled during multi-user boot. Comment out any existing lines with the # character, and add the lines below. Where you see the *N*, replace it with the value of your machine number:

```
DEVICE=eth0
NETMASK=255.255.255.0
IPADDR=10.0.0.N
NETWORK=10.0.0.0
BROADCAST=10.0.0.255
ONBOOT=yes
```

Step 16. The Red Hat Linux shell script **ifup** uses the configuration files you just setup to configure your system's network interface. This program will indirectly cause the loading of the NIC's device driver, and use the settings from the file **ifcfg-eth0** to configure IP. If there are any errors or problems bring up the interface, they are most likely being caused by incorrect variable names or values in the **ifcfg-eth0** file are problematic, or you specified the wrong device driver in **/etc/modules.conf**. Bring the interface up using the **ifup** shell script:

```
# ifup eth0
```

Step 17. Use **ifconfig** to check that your network settings look correct. Look for the **inet addr**, **Bcast**, and **Mask** values. Also look for the word **UP** at the beginning of the next line; if **UP** is missing, the interface is not up and running.

```
# ifconfig -a
```

Step 18. Once again use the ping command to test that the network interface is functional. You can also examine the lights for the port on the hub to which your machine is attached, and see if your Tx/Rx light is flashing. If it is flashing, the hub is receiving packets, which means they are leaving your **eth0** interface and arriving at the hub. Kill the ping with Control-C. Again, if there are problems, fix them before moving on.

Step 19. Try **ping**'ing your system's loopback interface. You can use either the IP address 127.0.0.1 or the name **localhost**.

Q9. The ping should not have succeeded – did it fail? _____

Step 20. Now try to ping your **eth0** interface's IP address (10.0.0.*N*).

Step 21. You probably found that the **ping** failed. This is because your loopback interface is not up. Run **ifconfig -a** again, and notice the absence of the word **UP** in the **lo** interface section. This means the interface is down (not running). Bring up the **loopback** interface with:

```
# ifup lo
```

Step 22. Now try to **ping** both your **loopback** interface and your own IP address again. This time both should succeed.

Step 23. It is very important to understand that pinging your own IP address does not prove that you have network connectivity to the network to which your card is connected. This is easily proven by pinging your own IP address (not the loopback address) and unplugging the network cable. You will notice that the ping continues to report responses! This means that packets are reaching their destination without going out over the wire. This occurs because IP intercepts the packets, and sends them instead through the loopback interface. And this is exactly why your ping did not work earlier when the loopback interface **lo** was down.

Step 24. Bring down the **eth0** and **lo** network interfaces with the commands:

```
# ifdown eth0
# ifdown lo
```

Step 25. You can now bring the machine up to multi-user level 3 (do not start a graphical session – level 5) and networking should be enabled. Use the command below and watch for the network-specific messages in the boot process:

```
# init 3
```

Step 26. Use the **ping** command again to test connectivity from your machine to other machines. Correct any problems before proceeding, making sure that your network interfaces are working correctly.

Configure and use telnet

Step 27. Try using the **telnet** command to connect to your own machine using the **student** account (it should be exist already).

```
$ telnet your-ip-address
```

Step 28. The telnet above should have failed to connect. This is because **telnet** requires the telnet daemon to be running on the remote machine. Your system may not have the telnet daemon software installed (if it was installed using the **workstation** configuration). To test for this software, run the command:

```
$ rpm -qa | grep telnet-server
```

Step 29. If the above command produced a single line (the package name of the telnet server software) skip to Step 31. Otherwise, the telnet software is not installed, so you will need to install the telnet server package from the Red Hat Linux CD-ROM (or any place you can find RPMs). Insert a Red Hat Linux Installer Disk 1 into the CD-ROM drive and mount the disk using:

```
# mount /mnt/cdrom
```

Step 30. The Red Hat software is available in RPM package formats in the directory **/mnt/cdrom/RedHat/RPMS**. Change to that directory and install the telnet server package with the command:

```
# rpm -ivh telnet-server*
```

Step 31. Now that you have the telnet server software installed, you can configure the system to start up the server. The telnet daemon is configured to be controlled with **xinetd**, the networking super-daemon. To allow **xinetd** to run the telnet daemon so that others can use **telnet** to connect to your machine, you need to edit the **xinetd** configuration file for the telnet daemon. The file is named **/etc/xinetd.d/telnet**. Change the line in the file **disabled = yes** to **disabled = no**.

Step 32. Now try using **telnet** again as you did above.

Step 33. Changing a configuration file is not enough to cause **xinetd** to re-read its configuration files. The currently running **xinetd** is unaware that you have changed the file **/etc/xinetd.d/telnet**. In fact, and this is very important to understand, *no* daemons will notice changes to their configuration files until you force them to re-read their configuration files. Many daemons will re-read their configuration files when they receive a particular signal (which signal depends on the daemon). To force **xinetd** to re-read its configuration file, review the bottom of the **xinetd** man page to learn how to tell **xinetd** to re-read its configuration files. Hint: Search for the word **signal**.

Step 34. Deliver the appropriate signal to **xinetd** daemon using the **kill** command – make sure you send the correct signal, or you may kill **xinetd**! Check that it is still running using the **ps -ef** command.

Q10. What is the exact command that causes **xinetd** to re-read its configuration file: _____

Step 35. Once again, try to **telnet** into your own machine. Be sure that you are able to connect and login before proceeding.

Step 36. Now try to **telnet** into one of your classmate's systems. Once you have logged into their system, **telnet** from their system back to your own system. Exit both **telnet** sessions when you have succeeded. You will not be able to telnet as **root** (a security measure) – use your system's **student** account.

Step 37. Typing IP addresses is tedious and host names are much easier. To associate a host name with an IP address, edit the file **/etc/hosts** and add a new line that looks like the line below, where *IP-address* is the IP address of the machine you want to call *hostname*. If you are connecting to host **host20**, then your entry would like 192.168.10.120 host20. Never remove or change the **localhost** entry!

```
IP-address hostname
```

Step 38. Now try to **ping** and **telnet** again using the name of the host instead of its IP address. When you want to use a host's name instead of an IP address, you must have an entry in the **/etc/hosts** file (unless of course you are running DNS or NIS which you will learn about in due course). Add names whenever you want.

Configure and use rsh and rlogin

Step 39. Try logging into your own machine using the **rlogin** (remote login) command:

```
$ rlogin -l student your-hostname
```

Step 40. The **rlogin** should have failed to connect. Again, you must install and enable the appropriate daemons, which in this case are the **rsh** and **rlogin** services. Install the **rsh-server*** package, as you did previously for the **telnet** package.

Step 41. The **rsh** and **rlogin** servers are also controlled via **xinetd**. Configure the **rsh** and **rlogin** configuration files for **xinetd** as you did earlier for **telnet**.

Step 42. Try to **rlogin** again. Exit the **rlogin** session just like you exit any shell. Once this succeeds, try to **rlogin** to another host.

Step 43. Try doing the same, this time with **rsh** instead of **rlogin**.

Q11. What is the difference between these two commands? _____