Class-6

1. **Linux Networking**
2. Explain basic networking concepts, including types of networks and addressing issues.
3. Configure network interfaces and use basic networking utilities, such as **ifconfig**, **ip**, **ping**, **route** and **traceroute**.
4. Use graphical and non-graphical browsers, such as Lynx, w3m, Firefox, Chrome and Epiphany.
5. *Transfer files to and from clients and servers using both graphical and text mode applications, such as FileZilla, ftp, sftp, curl and wget.*

Introduction to Networking

A network is a group of computers and computing devices connected together through communication channels, such as cables or wireless media. The computers connected over a network may be located in the same geographical area or spread across the world.

A network is used to:

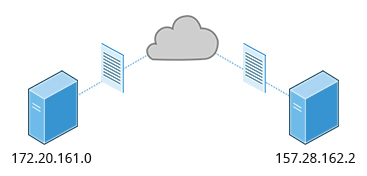
1. Allow the connected devices to communicate with each other
2. Enable multiple users to share devices over the network, such as music and video servers, printers and scanners.
3. Share and manage information across computers easily.

Most organizations have both an internal network and an Internet connection for users to communicate with machines and people outside the organization. The **Internet** is the largest network in the world and can be called *"the network of networks"*.

IP Addresses

Devices attached to a network must have at least one unique network address identifier known as the **IP** (**I**nternet **P**rotocol) address. The address is essential for routing packets of information through the network.

Exchanging information across the network requires using streams of small packets, each of which contains a piece of the information going from one machine to another. These packets contain data buffers, together with headers which contain information about where the packet is going to and coming from, and where it fits in the sequence of packets that constitute the stream. Networking protocols and software are rather complicated due to the diversity of machines and operating systems they must deal with, as well as the fact that even very old standards must be supported.



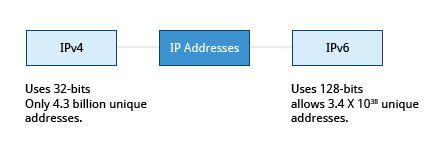
IPv4 and IPv6

There are two different types of IP addresses available: IPv4 (version 4) and IPv6 (version 6). IPv4 is older and by far the more widely used, while IPv6 is newer and is designed to get past limitations inherent in the older standard and furnish many more possible addresses.

IPv4 uses 32-bits for addresses; there are *only* 4.3 billion unique addresses available. Furthermore, many addresses are allotted and reserved, but not actually used. IPv4 is considered inadequate for meeting future needs because the number of devices available on the global network has increased enormously in recent years.

IPv6 uses 128-bits for addresses; this allows for 3.4 X 1038 unique addresses. If you have a larger network of computers and want to add more, you may want to move to IPv6, because it provides more unique addresses. However, it can be complex to migrate to IPv6; the two protocols do not always inter-operate well. Thus, moving equipment and addresses to IPv6 requires significant effort and has not been quite as fast as was originally intended. We will discuss IPv4 more than IPv6 as you are more likely to deal with it.

One reason IPv4 has not disappeared is there are ways to effectively make many more addresses available by methods such as NAT (Network Address Translation).  NAT enables sharing one IP address among many locally connected computers, each of which has a unique address only seen on the local network. While this is used in organizational settings, it also used in simple home networks. For example, if you have a router hooked up to your Internet Provider (such as a cable system) it gives you one externally visible address, but issues each device in your home an individual local address.



Decoding IPv4 Addresses

A 32-bit IPv4 address is divided into four 8-bit sections called [octets](https://en.wikipedia.org/wiki/Octet_(computing)).

Example:  
IP address →            172  .          16  .          31  .         46  
Bit format →     10101100.00010000.00011111.00101110

***NOTE****: Octet is just another word for byte.*

Network addresses are divided into five classes: A, B, C, D and E. Classes A, B and C are classified into two parts: Network addresses (Net ID) and Host address (Host ID). The Net ID is used to identify the network, while the Host ID is used to identify a host in the network. Class D is used for special multicast applications (information is broadcast to multiple computers simultaneously) and Class E is reserved for future use. In this section you will learn about classes A, B and C.

A 32-bit IPv4 address is divided into four 8-bit sections called [octets](https://en.wikipedia.org/wiki/Octet_(computing)).

Example:  
IP address →            172  .          16  .          31.         46  
Bit format →     10101100.00010000.00011111.00101110

***NOTE****: Octet is just another word for byte.*

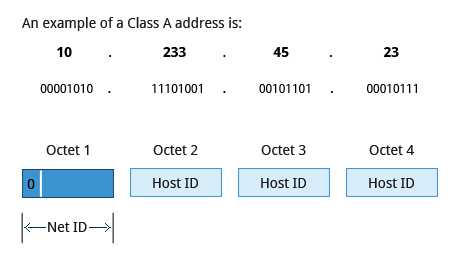
Network addresses are divided into five classes: A, B, C, D and E. Classes A, B and C are classified into two parts: Network addresses (Net ID) and Host address (Host ID). The Net ID is used to identify the network, while the Host ID is used to identify a host in the network. Class D is used for special multicast applications (information is broadcast to multiple computers simultaneously) and Class E is reserved for future use. In this section you will learn about classes A, B and C.

Class A Network Addresses

Class A addresses use the first octet of an IP address as their Net ID and use the other three octets as the Host ID**.** The first bit of the first octet is always set to zero. So you can use only 7-bits for unique network numbers. As a result, there are a maximum of 126 Class A networks available (the addresses 0000000 and 1111111 are reserved). Not surprisingly, this was only feasible when there were very few unique networks with large numbers of hosts. As the use of the Internet expanded, Classes B and C were added in order to accommodate the growing demand for independent networks.

Each Class A network can have up to 16.7 million unique hosts on its network. The range of host address is from **1.0.0.0** to **127.255.255.255**.

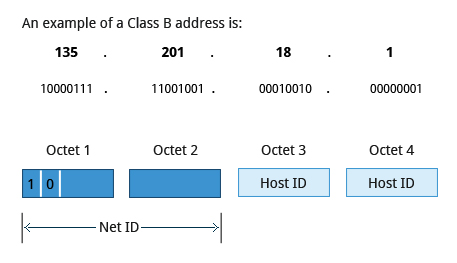
***NOTE****: The value of an octet, or 8-bits, can range from 0 to 255.*



Class B Network Addresses

Class B addresses use the first two octets of the IP address as their Net ID and the last two octets as the Host ID. The first two bits of the first octet are always set to binary 10, so there are a maximum of 16,384 (14-bits) Class B networks. The first octet of a Class B address has values from 128 to 191. The introduction of Class B networks expanded the number of networks but it soon became clear that a further level would be needed.

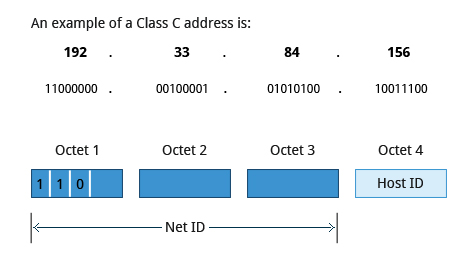
Each Class B network can support a maximum of 65,536 unique hosts on its network. The range of host addresses is from **128.0.0.0** to **191.255.255.255**.



Class C Network Addresses

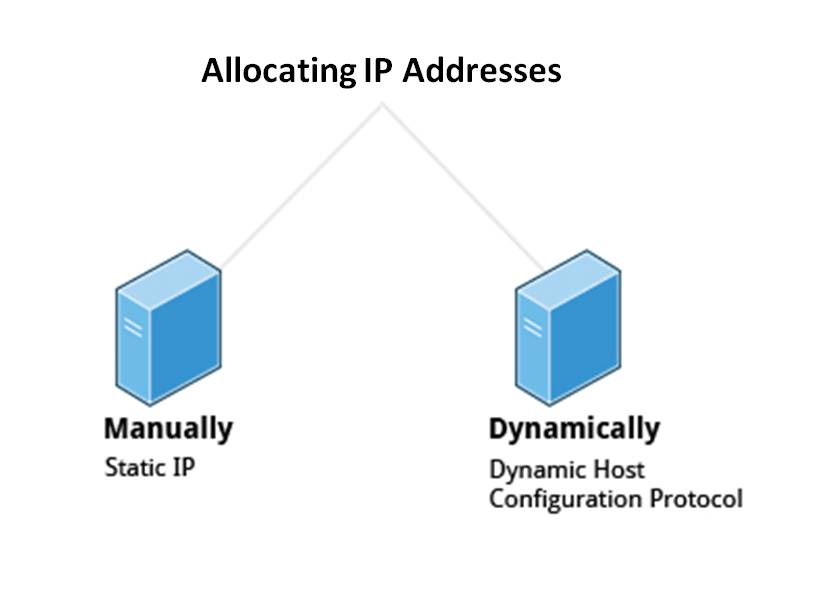
Class C addresses use the first three octets of the IP address as their Net ID and the last octet as their Host ID. The first three bits of the first octet are set to binary 110, so almost 2.1 million (21-bits) Class C networks are available. The first octet of a Class C address has values from 192 to 223. These are most common for smaller networks which don't have many unique hosts.

Each Class C network can support up to 256 (8-bits) unique hosts. The range of host addresses is from **192.0.0.0** to **223.255.255.255**.



IP Address Allocation

Typically, a range of IP addresses are requested from your Internet Service Provider (ISP) by your organization's network administrator. Often, your choice of which class of IP address you are given depends on the size of your network and expected growth needs. If NAT is in operation, such as in a home network, you only get one externally visible address!



You can assign IP addresses to computers over a network either manually or dynamically. Manual assignment adds static (never changing) addresses to the network. Dynamically assigned addresses can change every time you reboot or even more often; the **D**ynamic **H**ost **C**onfiguration **P**rotocol (**DHCP**) is used to assign IP addresses.

Name Resolution

Name Resolution is used to convert numerical IP address values into a human-readable format known as the hostname. For example, **104.95.85.15** is the numerical IP address that refers to the hostname whitehouse.gov. Hostnames are much easier to remember!

Given an IP address, you can obtain its corresponding hostname. Accessing the machine over the network becomes easier when you can type the hostname instead of the IP address.

You can view your system’s hostname simply by typing **hostname**with no argument.

***NOTE****: If you give an argument, the system will try to change its hostname to match it, however, only root users can do that.*

The special hostname localhost is associated with the IP address **127.0.0.1** and describes the machine you are currently on (which normally has additional network-related IP addresses).

# Screenshot Showing Server IP Address of The Linux Foundation Website

# **Screenshot Showing Server IP Address of The Linux Foundation Website**

Using Domain Name System (DNS) and Name Resolution Tools

Let's get a feel for how the domain nameserver works on a recent Ubuntu system.

So first we need to log in.

And when we log in, we'll open up a command line terminal because we're going to do all our operations from the command line.

So I have done that by right-clicking on the desktop.

And then let me make a bigger font.

I might as well go full screen while I'm at it.

So there are a couple of important files on your system, one is called "/etc/resolv.conf".

Now on older Linux systems, you didn't have all this information at the top.

You just had important thing here, nameserver 127.0.0.53.

You're more likely to find your nameserver is not a 127.1, which all 127 addresses are on the actual machine, but 192.1.6.8.1, for instance, would be a common thing.

It might point your wireless modem, which is attached to your Internet service provider, for instance.

Recent systems use a systemd service called "systemd-resolved", which works in a more complicated way.

It makes a DNS server on the local machine which caches the results of previous searches.

So we're not going to get into the details of that, but that's a recent development that you'll find on newer machines.

Another important file is "/etc/hosts".

This file is consulted before the domain nameserver is consulted.

So you'll see, for instance, here at the bottom, we've got two machines on the local network.

We give their IP addresses and a name that we can use to get at them.

And you'll notice you can have more than one name tied to an IP address.

So for instance, I could do "ping theordore" and that's going to take me to 200 or I could do "ping beaver" and it takes me to the same machine.

Now, if want to look at something which isn't specified in that file, or in fact, I can do that "host theordore", and it will go look at the file.

But if I want to get something out on the Internet, I could do "host linuxfoundation.org", and I'll get both the IPv4 and the IPv6 addresses here of the Linux Foundation.

And also information about the mail services provided by the Linux Foundation.

I can do another utility for similar information, it's "nslookup".

So I'll do that, "nslookup linuxfoundation.org".

You see, you got the same basic information, a little more compactly.

Here is the IPv4 address, and then the two IPv6 addresses.

Another utility I can use is "dig".

So let's do "dig linuxfoundation.org".

And you'll see, I once again got the information about the IP address and a lot more information about the search to find that server.

So that's just a little bit of what you can do with DNS and some of the files that are associated with it.

Network Configuration Files

Network configuration files are essential to ensure that interfaces function correctly. They are located in the **/etc** directory tree. However, the exact files used have historically been dependent on the particular Linux distribution and version being used.

For Debian family configurations, the basic network configuration files could be found under **/etc/network/**, while for Red Hat and SUSE family systems one needed to inspect **/etc/sysconfig/network**.

Modern systems emphasize the use of Network Manager, which we briefly discussed when we considered graphical system administration, rather than try to keep up with the vagaries of the files in **/etc**. While the graphical versions of Network Manager do look somewhat different in different distributions, the **nmtui** utility (shown in the screenshot) varies almost not at all, as does the even more sparse **nmcli**(command line interface) utility. If you are proficient in the use of the GUIs, by all means, use them. If you are working on a variety of systems, the lower level utilities may make life easier.

# Network Manager

# **Network Manager**

Recent Ubuntu distributions include **netplan**, which is turned on by default, and supplants Network Manager. Since no other distribution has shown interest, and since it can easily be disabled if it bothers you, we will ignore it.

Network Interfaces

Network interfaces are a connection channel between a device and a network. Physically, network interfaces can proceed through a network interface card (NIC), or can be more abstractly implemented as software. You can have multiple network interfaces operating at once. Specific interfaces can be brought up (activated) or brought down (de-activated) at any time.

Information about a particular network interface or all network interfaces can be reported by the **ip** and **ifconfig** utilities, which you may have to run as the superuser, or at least, give the full path, i.e. **/sbin/ifconfig**, on some distributions. **ip** is newer than **ifconfig**and has far more capabilities, but its output is uglier to the human eye. Some new Linux distributions do not install the older **net-tools** package to which **ifconfig**belongs, and  so you would have to install it if you want to use it.

# Network Interfaces

# **Network Interfaces**

The ip Utility

To view the IP address:

**$ /sbin/ip addr show**

To view the routing information:

**$ /sbin/ip route show**

**ip** is a very powerful program that can do many things. Older (and more specific) utilities such as **ifconfig** and **route** are often used to accomplish similar tasks. A look at the relevant man pages can tell you much more about these utilities.

# ip utility

# **ip Utility**

ping

**ping** is used to check whether or not a machine attached to the network can receive and send data; i.e. it confirms that the remote host is online and is responding.

To check the status of the remote host, at the command prompt, type **ping <hostname>**.

**ping** is frequently used for network testing and management; however, its usage can increase network load unacceptably. Hence, you can abort the execution of **ping** by typing **CTRL-C**, or by using the **-c** option, which limits the number of packets that **ping** will send before it quits. When execution stops, a summary is displayed.

# ping

# **Ping**

**route**

A network requires the connection of many nodes. Data moves from source to destination by passing through a series of routers and potentially across multiple networks. Servers maintain routing tablescontaining the addresses of each node in the network. The IP routing protocols enable routers to build up a forwarding table that correlates final destinations with the next hop addresses.

# route

# **Route**

One can use the **route** utilityor the newer **ip route** command to view or change the IP routing table to add, delete, or modify specific (static) routes to specific hosts or networks. The table explains some commands that can be used to manage IP routing:

|  |  |
| --- | --- |
| **Task** | **Command** |
| Show current routing table | **$ route –n** or **ip route** |
| Add static route | **$ route add -net address** or **ip route add** |
| Delete static route | **$ route del -net address** or **ip route del** |

traceroute

**traceroute** is used to inspect the route which the data packet takes to reach the destination host, which makes it quite useful for troubleshooting network delays and errors. By using **traceroute**, you can isolate connectivity issues between hops, which helps resolve them faster.

To print the route taken by the packet to reach the network host, at the command prompt, type **traceroute <address>**.

# traceroute

# **Traceroute**

More Networking Tools

Now, let’s learn about some additional networking tools. Networking tools are very useful for monitoring and debugging network problems, such as network connectivity and network traffic.

|  |  |
| --- | --- |
| **Networking Tools** | **Description** |
| **ethtool** | Queries network interfaces and can also set various parameters such as the speed |
| **netstat** | Displays all active connections and routing tables; useful for monitoring performance and troubleshooting |
| **nmap** | Scans open ports on a network; important for security analysis |
| **tcpdump** | Dumps network traffic for analysis |
| **iptraf** | Monitors network traffic in text mode |
| **mtr** | Combines functionality of ping and traceroute and gives a continuously updated display |
| **dig** | Tests DNS workings; a good replacement for host and nslookup |

Graphical and Non-Graphical Browsers

Browsers are used to retrieve, transmit, and explore information resources, usually on the World Wide Web. Linux users commonly use both graphical and non-graphical browser applications.

The common graphical browsers used in Linux are:

1. [Firefox](https://www.mozilla.org/en-US/firefox/)
2. [Google Chrome](https://www.google.com/chrome/)
3. [Chromium](https://www.chromium.org/Home)
4. [Konqueror](https://kde.org/applications/internet/org.kde.konqueror)
5. [Opera](https://www.opera.com/)

Sometimes, you either do not have a graphical environment to work in (or have reasons not to use it) but still need to access web resources. In such a case, you can use non-graphical browsers, such as the following:

|  |  |
| --- | --- |
| **Non-Graphical Browsers** | **Description** |
| [lynx](http://lynx.browser.org/) | Configurable text-based web browser; the earliest such browser and still in use |
| [elinks](http://www.elinks.cz/) | Based on Lynx; it can display tables and frames |
| [w3m](http://w3m.sourceforge.net/) | Another text-based web browser with many features |

wget

Sometimes, you need to download files and information, but a browser is not the best choice, either because you want to download multiple files and/or directories, or you want to perform the action from a command line or a script. **wget** is a command line utility that can capably handle the following types of downloads:

1. Large file downloads
2. Recursive downloads, where a web page refers to other web pages and all are downloaded at once
3. Password-required downloads
4. Multiple file downloads.

To download a web page, you can simply type **wget <url>**, and then you can read the downloaded page as a local file using a graphical or non-graphical browser.

# wget

curl

Besides downloading, you may want to obtain information about a URL, such as the source code being used. **curl** can be used from the command line or a script to read such information. **curl** also allows you to save the contents of a web page to a file, as does **wget.**

You can read a URL using **curl <URL>**. For example, if you want to read [http://www.linuxfoundation.org](https://www.linuxfoundation.org/), type **curl http://www.linuxfoundation.org**.

To get the contents of a web page and store it to a file, type **curl -o saved.html http://www.mysite.com**. The contents of the main index file at the website will be saved in **saved.html**.

# curl

FTP (File Transfer Protocol)

When you are connected to a network, you may need to transfer files from one machine to another. **F**ile **T**ransfer **P**rotocol (**FTP**) is a well-known and popular method for transferring files between computers using the Internet. This method is built on a client-server model. FTP can be used within a browser or with stand-alone client programs.

# File Transfer Protocol

# FTP is one of the oldest methods of network data transfer, dating back to the early 1970s. As such, it is considered inadequate for modern needs, as well as being intrinsically insecure. However, it is still in use and when security is not a concern (such as with so-called anonymous FTP) it can make sense. However, many websites, such as [kernel.org](https://www.kernel.org/), have abandoned its use.

FTP Clients

FTP clients enable you to transfer files with remote computers using the FTP protocol. These clients can be either graphical or command line tools. Filezilla, for example, allows use of the drag-and-drop approach to transfer files between hosts. All web browsers support FTP, all you have to do is give a URL like **ftp://ftp.kernel.org** where the usual **http://** becomes **ftp://**.

Some command line FTP clients are:

* + - **ftp**
    - **sftp**
    - **ncftp**
    - **yafc** (Yet Another FTP Client).

FTP has fallen into disfavor on modern systems, as it is intrinsically insecure, since passwords are user credentials that can be transmitted without encryption and are thus prone to interception. Thus, it was removed in favour of using **rsync** and web browser https access for example. As an alternative, **sftp** is a very secure mode of connection, which uses the Secure Shell (**ssh**) protocol, which we will discuss shortly. **sftp** encrypts its data and thus sensitive information is transmitted more securely. However, it does not work with so-called anonymous FTP (guest user credentials).

# FTP Clients

# **FTP Clients**

SSH: Executing Commands Remotely

Secure Shell (SSH) is a cryptographic network protocol used for secure data communication. It is also used for remote services and other secure services between two devices on the network and is very useful for administering systems which are not easily available to physically work on, but to which you have remote access.

# SSH: Executing Commands Remotely

# **SSH: Executing Commands Remotely**

To login to a remote system using your same user name you can just type **ssh some\_system** and press **Enter**. **ssh** then prompts you for the remote password. You can also configure ssh to securely allow your remote access without typing a password each time.

If you want to run as another user, you can do either **ssh -l someone some\_system** or **ssh someone@some\_system**. To run a command on a remote system via SSH, at the command prompt, you can type **ssh some\_system my\_command**.

Copying Files Securely with scp

We can also move files securely using Secure Copy (scp) between two networked hosts. scp uses the SSH protocol for transferring data.

To copy a local file to a remote system, at the command prompt, type **scp <localfile> <user@remotesystem>:/home/user/** and press **Enter**.

You will receive a prompt for the remote password. You can also configure **scp** so that it does not prompt for a password for each transfer.

# Copying Files Securely with scp

**Copying Files Securely with scp**

**Using SSH Between Two Virtual Machines**

**Let's do a simple demonstration of using secure shell and secure copy between two virtual machines: an Ubuntu machine and a CentOS machine.**

**So, first, from the Ubuntu machine, let's try to log into the CentOS machine.**

**Well, first I need to know the IP address of both machines. So, I can do that with "ip", and I'll say "--brief", to get a condensed output, "addr" for address, "show" [ip --brief add show].**

**And I can see my address on this machine is 172.16.249.133.**

**I'll do the same command on the CentOS machine, and to see the address is the same except for the last byte or octet, which is 129.**

**So first, let's login to the CentOS machine from Ubuntu.**

**I do "ssh student@172.16.249.129", and it's the first time I'm doing this. So it wants to make sure that I'm authentic. So, I say yes.**

**And now, I have to give the password, and it's fine. I'm on the CentOS machine, as you can see from the prompt.**

**Now, in that command, I really didn't have to give student@, because we're using the student account on both machines, but it never hurts to do that.**

**So, let me exit.**

**Now let's do from the CentOS machine. Let's copy over a directory using scp over to the Ubuntu machine.**

**So, let me do "scp -r" for recursive to get the whole directory and everything underneath it, and this time, I want /home/student, I'll copy the /home/student directory.**

**I won't bother saying student@, I'll just give the address 172.16.249.133, and I'll put it in the /tmp directory.**

**Once again, it wants to make sure that it's authentic. So, I'll say yes and once again I have to give the password, and it's copied over.**

**And if I go over to the Ubuntu machine and I look in the /tmp directory, I'll see the account is now there, that the directory is now there.**

**So that's all there is to do. A pretty simple demonstration of using secure shell and secure copy.**

Lab 14.1: Network Troubleshooting

Troubleshooting network problems is something that you will often encounter if you haven't already. We are going to practice some of the previously discussed tools, that can help you isolate, troubleshoot and fix problems in your network.

The solution file contains a step-by-step procedure for exercising many of the tools we have studied. Please repeat the steps, substituting your actual network interface names, alternative network addresses and web sites, etc.

Click the link below to view a solution to the Lab exercise.

[Lab Solution](https://courses.edx.org/asset-v1:LinuxFoundationX+LFS101x+2T2021+type@asset+block/labsol-network.html)

**Lab: Network Troubleshooting**

Troubleshooting network problems is something that you will often encounter if you haven't already. We are going to practice some of the previously discussed tools, that can help you isolate, troubleshoot and fix problems in your network.

Suppose you need to perform an Internet search, but your web browser can not find [google.com](http://google.com/), saying the host is unknown. Let's proceed step by step to fix this.

1. First make certain your network is properly configured. If your Ethernet device is up and running, running **ifconfig** should display something like:

**student:/tmp> /sbin/ifconfig**

**eno167777 Link encap:Ethernet HWaddr 00:0C:29:BB:92:C2**

**inet addr:192.168.1.14 Bcast:192.168.1.255 Mask:255.255.255.0**

**inet6 addr: fe80::20c:29ff:febb:92c2/64 Scope:Link**

**UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1**

**RX packets:3244 errors:0 dropped:0 overruns:0 frame:0**

**TX packets:2006 errors:0 dropped:0 overruns:0 carrier:0**

**collisions:0 txqueuelen:1000**

**RX bytes:4343606 (4.1 Mb) TX bytes:169082 (165.1 Kb)**

**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:0 errors:0 dropped:0 overruns:0 frame:0**

**TX packets:0 errors:0 dropped:0 overruns:0 carrier:0**

**collisions:0 txqueuelen:0**

**RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)**

On older systems you probably will see a less cryptic name than **eno167777**, like **eth0**, or for a wireless connection, you might see something like **wlan0** or **wlp3s0**. You can also show your **IP** address with:

**student:/tmp> ip addr show**

**1: lo: mtu 65536 qdisc noqueue state UNKNOWN group default**

**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**

**2: eno16777736: mtu 1500 qdisc pfifo\_fast state \**

**UP group default qlen 1000**

**link/ether 00:0c:29:bb:92:c2 brd ff:ff:ff:ff:ff:ff**

**p inet 192.168.1.14/24 brd 192.168.1.255 scope global dynamic eno16777736**

**valid\_lft 84941sec preferred\_lft 84941sec**

**inet 192.168.1.15/24 brd 192.168.1.255 scope global secondary dynamic eno16777736**

**valid\_lft 85909sec preferred\_lft 85909sec**

**inet6 fe80::20c:29ff:febb:92c2/64 scope link**

**valid\_lft forever preferred\_lft forever**

Does the IP address look valid? Depending on where you are using this from, it is most likely a Class C IP address; in the above this is **192.168.1.14**

If it does not show a device with an **IP** address, you may need to start or restart the network and/or **NetworkManager**. Exactly how you do this depends on your system. For most distributions one of these commands will accomplish this:

**student:/tmp> sudo systemctl restart NetworkManager**

**student:/tmp> sudo systemctl restart network**

**student:/tmp> sudo service NetworkManager restart**

**student:/tmp> sudo service network restart**

If your device was up but had no **IP** address, the above should have helped fix it, but you can try to get a fresh address with:

**student:/tmp> sudo dhclient eth0**

substituting the right name for the Ethernet device.

1. If your interface is up and running with an assigned **IP** address and you still can not reach [google.com](http://google.com/), we should make sure you have a valid hostname assigned to your machine, with **hostname**:

**student:/tmp> hostname**

**openSUSE**

It is rare you would have a problem here, as there is probably always at least a default hostname, such as **localhost**.

1. When you type in a name of a site such as [google.com](http://google.com/), that name needs to be connected to a known IP address. This is usually done employing the **DNS** sever (**D**omain **N**ame **S**ystem)

First, see if the site is up and reachable with **ping**:

**student:/tmp> sudo ping -c 3 google.com**

**PING google.com (216.58.216.238) 56(84) bytes of data.**

**64 bytes from ord31s22-in-f14.1e100.net (216.58.216.238): icmp\_seq=1 ttl=51 time=21.7 ms**

**64 bytes from ord31s22-in-f14.1e100.net (216.58.216.238): icmp\_seq=2 ttl=51 time=23.8 ms**

**64 bytes from ord31s22-in-f14.1e100.net (216.58.216.238): icmp\_seq=3 ttl=51 time=21.3 ms**

**--- google.com ping statistics ---**

**3 packets transmitted, 3 received, 0% packet loss, time 2002ms**

**rtt min/avg/max/mdev = 21.388/22.331/23.813/1.074 ms**

Note:

* + We have used **sudo** for **ping**; recent **Linux** distributions have required this to avoid clueless or malicious users from flooding systems with such queries.
  + We have used **-c 3** to limit to 3 packets; otherwise **ping** would run forever until forcibly terminated, say with **CTRL-C**.

If the result was:

**ping: unknown host google.com**

It is likely that something is wrong with your DNS set-up. (Note on some systems you will never see the unknown host message, but you will get a suspicious result like:

**student:/tmp> sudo ping l89xl28vkjs.com**

**PING l89xl28vkjs.com.site (127.0.53.53) 56(84) bytes of data.**

**64 bytes from 127.0.53.53: icmp\_seq=1 ttl=64 time=0.016 ms**

**...**

where the **127.0.x.x** address is a loop feeding back to the host machine you are on. You can eliminate this as being a valid address by doing:

**student:/tmp> host l89xl28vkjs.com**

**Host l89xl28vkjs.com not found: 3(NXDOMAIN)**

whereas a correct result would look like:

**student:/tmp> host google.com**

**google.com has address 216.58.216.206**

**google.com has IPv6 address 2607:f8b0:4009:80b::200e**

**google.com mail is handled by 20 alt1.aspmx.l.google.com.**

**google.com mail is handled by 10 aspmx.l.google.com.**

**google.com mail is handled by 30 alt2.aspmx.l.google.com.**

**google.com mail is handled by 40 alt3.aspmx.l.google.com.**

**google.com mail is handled by 50 alt4.aspmx.l.google.com.**

The above command utilizes the **DNS** server configured in **/etc/resolv.conf** on your machine. If you wanted to override that you could do:

**host 8.8.8.8**

**8.8.8.8.in-addr.arpa domain name pointer google-public-dns-a.google.com.**

**student@linux:~> host google.com 8.8.8.8**

**Using domain server:**

**Name: 8.8.8.8**

**Address: 8.8.8.8#53**

**Aliases:**

**google.com has address 216.58.216.110**

**google.com has IPv6 address 2607:f8b0:4009:804::1002**

**...\**

where we have used the publicly available **DNS** server provided by **Google** itself. (Using this or another public server can be a good trick sometimes if your network is up but **DNS** is ill; in that case you can also enter it in **resolv.conf**.)

Note that there is another file, **/etc/hosts**, where you can associate names with **IP** addresses, which is used **before** the **DNS** server is consulted. This is most useful for specifying nodes on your local network.

You could also use the **dig** utility if you prefer:

**student:/tmp> dig google.com**

**; <<>> DiG 9.9.5-rpz2+rl.14038.05-P1 <<>> google.com**

**;; global options: +cmd**

**;; Got answer:**

**;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29613**

**;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 1**

**;; OPT PSEUDOSECTION:**

**; EDNS: version: 0, flags:; MBZ: 1c20 , udp: 1280**

**;; QUESTION SECTION:**

**;google.com. IN A**

**;; ANSWER SECTION:**

**google.com. 244 IN A 173.194.46.67**

**google.com. 244 IN A 173.194.46.65**

**google.com. 244 IN A 173.194.46.71**

**google.com. 244 IN A 173.194.46.73**

**google.com. 244 IN A 173.194.46.69**

**google.com. 244 IN A 173.194.46.68**

**google.com. 244 IN A 173.194.46.64**

**google.com. 244 IN A 173.194.46.72**

**google.com. 244 IN A 173.194.46.70**

**google.com. 244 IN A 173.194.46.66**

**google.com. 244 IN A 173.194.46.78**

**;; Query time: 22 msec**

**;; SERVER: 192.168.1.1#53(192.168.1.1)**

**;; WHEN: Mon Apr 20 08:58:58 CDT 2015**

**;; MSG SIZE rcvd: 215**

1. Suppose **host** or **dig** fail to connect the name to an **IP** address. There are many reasons **DNS** can fail, some of which are:
   * The **DNS** server is down. In this case try **ping**ing it to see if it is alive (you should have the **IP** address in **/etc/resolv.conf**.
   * The server can be up and running, but **DNS** may not be currently available on the machine.
   * Your **route** to the **DNS** server may not be correct.

How can we test the route? Tracing the route to one of the public name server we mentioned before:

**student@linux:~> sudo traceroute 8.8.8.8**

**traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets**

**1 192.168.1.1 (192.168.1.1) 0.405 ms 0.494 ms 0.556 ms**

**2 10.132.4.1 (10.132.4.1) 15.127 ms 15.107 ms 15.185 ms**

**3 dtr02ftbgwi-tge-0-6-0-3.ftbg.wi.charter.com (96.34.24.122)**

**15.243 ms 15.327 ms 17.878 ms**

**4 crr02ftbgwi-bue-3.ftbg.wi.charter.com (96.34.18.116) 17.667 ms 17.734 ms 20.016 ms**

**5 crr01ftbgwi-bue-4.ftbg.wi.charter.com (96.34.18.108) 22.017 ms 22.359 ms 22.052 ms**

**6 crr01euclwi-bue-1.eucl.wi.charter.com (96.34.16.77) 29.430 ms 22.705 ms 22.076 ms**

**7 bbr01euclwi-bue-4.eucl.wi.charter.com (96.34.2.4) 17.795 ms 25.542 ms 25.600 ms**

**8 bbr02euclwi-bue-5.eucl.wi.charter.com (96.34.0.7) 28.227 ms 28.270 ms 28.303 ms**

**9 bbr01chcgil-bue-1.chcg.il.charter.com (96.34.0.9) 33.114 ms 33.072 ms 33.175 ms**

**10 prr01chcgil-bue-2.chcg.il.charter.com (96.34.3.9) 36.882 ms 36.794 ms 36.895 ms**

**11 96-34-152-30.static.unas.mo.charter.com (96.34.152.30) 42.585 ms 42.326 ms 42.401 ms**

**12 216.239.43.111 (216.239.43.111) 28.737 ms 216.239.43.113 (216.239.43.113)**

**24.558 ms 23.941 ms**

**13 209.85.243.115 (209.85.243.115) 24.269 ms 209.85.247.17 (209.85.247.17)**

**25.758 ms 216.239.50.123 (216.239.50.123) 25.433 ms**

**14 google-public-dns-a.google.com (8.8.8.8) 25.239 ms 24.003 ms 23.795 ms**

Again, this should likely work for you, but what if you only got the first line in the **traceroute** output?

If this happened, most likely your default route is wrong. Try:

**student:/tmp> ip route show**

**efault via 192.168.1.1 dev eno16777736 proto static metric 1024**

**192.168.1.0/24 dev eno16777736 proto kernel scope link src 192.168.1.14**

Most likely this is set to your network interface and the IP address of your router, **DSL**, or Cable Modem. Let's say that it is blank or simply points to your own machine. Here's your problem! At this point, you would need to add a proper default route and run some of the same tests we just did.

Note, an enhanced version of **traceroute** is supplied by **mtr**, which runs continuously (like **top**). Running it with the **--report-cycles** option to limit how long it runs:

**student:/tmp> sudo mtr --report-cycles 3 8.8.8.8**

**My traceroute [v0.85]**

**c7 (0.0.0.0) Mon Apr 20 09:30:41 2015**

**Unable to allocate IPv6 socket for nameserver communication: Address family not supported**

**by protocol Packets Pings**

**Host Loss% Snt Last Avg Best Wrst StDev**

**0.0% 3 0.3 0.3 0.2 0.3 0.0**

**2. 10.132.4.1 0.0% 3 6.3 7.1 6.3 8.4 0.7**

**3. dtr02ftbgwi-tge-0-6-0-3.ftbg.wi. 0.0% 3 6.2 7.5 6.2 10.0 2.1**

**4. dtr01ftbgwi-bue-1.ftbg.wi.charte 0.0% 3 8.9 8.5 6.2 10.4 2.0**

**5. crr01ftbgwi-bue-4.ftbg.wi.charte 0.0% 3 8.9 9.7 8.9 10.4 0.0**

**6. crr01euclwi-bue-1.eucl.wi.charte 0.0% 3 16.5 17.4 14.2 21.5 3.7**

**7. bbr01euclwi-bue-4.eucl.wi.charte 0.0% 3 23.5 22.0 18.2 24.2 3.2**

**8. bbr02euclwi-bue-5.eucl.wi.charte 0.0% 3 18.9 22.7 18.1 31.1 7.2**

**9. bbr01chcgil-bue-1.chcg.il.charte 0.0% 3 22.9 23.0 22.9 23.1 0.0**

**10. prr01chcgil-bue-2.chcg.il.charte 0.0% 3 21.4 24.1 20.8 30.2 5.2**

**11. 96-34-152-30.static.unas.mo.char 0.0% 3 22.6 21.9 20.0 23.3 1.6**

**12. 216.239.43.111 0.0% 3 21.2 21.7 21.2 22.0 0.0**

**13. 72.14.237.35 0.0% 3 21.2 21.0 19.8 21.9 1.0**

**14. google-public-dns-a.google.com 0.0% 3 26.7 23.0 21.0 26.7 3.2**

Hopefully, running through some of these commands helped. It actually helps to see what the correct output for your system looks like. Practice using these commands; it is very likely that you will need them someday.

Lab 14.2: Non-Graphical Browsers

There are times when a graphical browser is not available, but you need to look up or download a resource. In this exercise, we are going to experiment with using non-graphical web browsers.

The solution file contains a step-by-step procedure for exercising the tools discussed. Please repeat the steps, substituting web sites, etc.

Click the link below to view a solution to the Lab exercise.

Lab: Non-graphical browsers

We have discussed non-graphical browsers:

* **lynx**
* **links** and **elinks**
* **w3m**

There are times when you will not have a graphical window interface running on your Linux machine and you need to look something up on the web or download a driver (like a graphics driver in order to bring up a graphical window interface). So, it is a good idea to practice using a non-graphical web browser to do some work.

With **links**, you can use your mouse to click on the top line of the screen to get a menu. In this case, we want to go to [google.com](http://google.com/) (or your favorite search engine), so you can just type **g** to go to a typed-in URL.

Pressing the **TAB** key will move your cursor to the **OK** button. You can then press the **ENTER** key.

You should now be at [google.com](http://google.com/) (or your favorite search engine). Use the down-arrow key to move through the choices until you reach the blank line used to enter your search query. Now type **Intel Linux graphics drivers** in the search box. Use the down-arrow key to move you to the **Google Search** button. With that highlighted, press the **ENTER** key.

Use your down-arrow key to move to the entry: **Intel(R) Graphics Drivers for Linux - Download Center**. It may take several presses of the down-arrow key. You can press the space-bar to move down the page or the 'b' key to move back up the page if needed. Once this line is highlighted, press the **ENTER** key. You will now go to the Intel Graphics Driver for Linux page. If you want, you can read the page. Remember, the space-bar will page you down the page while the 'b' key will move you back up the page. The Page Down and Page Up keys will do the same thing if you prefer. Find the URL under the line

**URL Location:**

Position your cursor at this line using the up-arrow or down-arrow key. Press the **ENTER** key to go to this location.

Page down this page until you see the line:

**Latest Releases**

If you move your cursor with the arrow keys, find the latest version (with the most recent release date) under this section. If using your arrow-keys, you should highlight **Release Notes**. Press the **ENTER** key.

This has installers for versions of **Ubuntu** and **Fedora**, along with the source code. You will need to page down a page or two depending on the size of your screen.

Select one of the installers, perhaps for the version of Linux that you are running, or just a random one, and press the **ENTER** key.

You should see a text dialog box with choices of what to do. Save the package wherever you want to.

You can now quit your non-graphical browser. If you used **links**, then click on the top line of the screen, select the **File** drop-down menu item, and click on **Exit**. Confirm that you really want to exit Links. You should now see your shell prompt.

**Linux Network Operations**

**Server-Client Configurations (FTP, SFTP AND HTTP)**

1. Deploying FTP Server & Interacting with Client

2. Deploying SFTP Server & Interacting with Client

3. Deploying HTTP Server & Interacting with Client

**FTP Server:**

**Step 1:** Update your repository and install VSFTPD package.

**Yum check-update;** **yum -y install vsftpd**

**Step2:** After installation you can find /etc/vsftpd/vsftpd.conf file

**Root#** mv /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.org

**SFTP Server:**

**Step1: create a separate group for FTP access**

**Root # yum -y install openssh-server**

**Root# groupadd ftpaccess**

**Step2: Now open /etc/ssh/sshd\_config file and make changes as below:**

**Subsystem sftp internal-sftp**

**Match group ftpaccess.**

**FTP server:**

**Step3: Now restart vsftpd service and make it start automatically after reboot**

**Root#** systemctl restart vsftpd.service

**Root#** systemctl enable vsftpd. service

**Step4:** Add FTP service firewall to allow ftp ports.

**Root#** Firewall -cmd –permanent –add-service=ftp

# firewall-cmd –reload

**Step5:** setup SEinux to allow ftp access to the users home directories:

**#** setsebiik -P ftp\_home\_dir on

**Step6:** Now create an User for ftp access. Here /**sbin/nologin** shell is used to prevent shell access to the server

**Root#** useradd -m dave -s /sbin/nologin

# Passwd dave

**HTTP Server**

# yum -y install https