**Linux Networking**

**IP Adddressing**

**Basic Terminology**

* **Address**—The unique number ID assigned to one host or interface in a network.
* **Subnet**—A portion of a network sharing a particular subnet address.
* **Subnet mask**—A 32-bit combination used to describe which portion of an address refers to the subnet and which part refers to the host.
* **Interface**—A network connection.

An IP address is an address used in order to uniquely identify a device on an IP network. The address is made up of 32 binary bits, which can be divisible into a network portion and host portion with the help of a subnet mask. The 32 binary bits are broken into four octets (1 octet = 8 bits). Each octet is converted to decimal and separated by a period (dot). For this reason, an IP address is said to be expressed in dotted decimal format (for example, 172.16.81.100). The value in each octet ranges from 0 to 255 decimal, or 00000000 - 11111111 binary.

Here is how binary octets convert to decimal: The right most bit, or least significant bit, of an octet holds a value of 20. The bit just to the left of that holds a value of 21. This continues until the left-most bit, or most significant bit, which holds a value of 27. So if all binary bits are a one, the decimal equivalent would be 255 as shown here:

1 1 1 1 1 1 1 1

128 64 32 16 8 4 2 1 (128+64+32+16+8+4+2+1=255)

Here is a sample octet conversion when not all of the bits are set to 1.

0 1 0 0 0 0 0 1

0 64 0 0 0 0 0 1 (0+64+0+0+0+0+0+1=65)

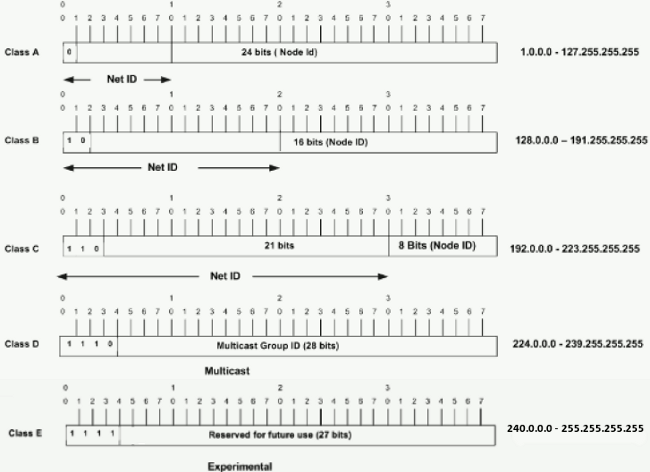
And this is sample shows an IP address represented in both binary and decimal.

10. 1. 23. 19 (decimal)

00001010.00000001.00010111.00010011 (binary)

These octets are broken down to provide an addressing scheme that can accommodate large and small networks. There are five different classes of networks, A to E. This document focuses on addressing classes A to C, since classes D and E are reserved and discussion of them is beyond the scope of this document.

Given an IP address, its class can be determined from the three high-order bits. [Figure 1](http://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html#figone) shows the significance in the three high order bits and the range of addresses that fall into each class. For informational purposes, Class D and Class E addresses are also shown.

**Figure 1**

In a Class A address, the first octet is the network portion, so the Class A example in [Figure 1](http://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html#figone) has a major network address of 1.0.0.0 - 127.255.255.255. Octets 2, 3, and 4 (the next 24 bits) are for the network manager to divide into subnets and hosts as he/she sees fit. Class A addresses are used for networks that have more than 65,536 hosts (actually, up to 16777214 hosts!).

In a Class B address, the first two octets are the network portion, so the Class B example in [Figure 1](http://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html#figone) has a major network address of 128.0.0.0 - 191.255.255.255. Octets 3 and 4 (16 bits) are for local subnets and hosts. Class B addresses are used for networks that have between 256 and 65534 hosts.

In a Class C address, the first three octets are the network portion. The Class C example in [Figure 1](http://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html#figone) has a major network address of 192.0.0.0 - 223.255.255.255. Octet 4 (8 bits) is for local subnets and hosts - perfect for networks with less than 254 hosts.

## Network Masks

A network mask helps you know which portion of the address identifies the network and which portion of the address identifies the node. Class A, B, and C networks have default masks, also known as natural masks, as shown here:

Class A: 255.0.0.0

Class B: 255.255.0.0

Class C: 255.255.255.0

An IP address on a Class A network that has not been subnetted would have an address/mask pair similar to: 8.20.15.1 255.0.0.0. To see how the mask helps you identify the network and node parts of the address, convert the address and mask to binary numbers.

8.20.15.1 = 00001000.00010100.00001111.00000001

255.0.0.0 = 11111111.00000000.00000000.00000000

Once you have the address and the mask represented in binary, then identifying the network and host ID is easier. Any address bits which have corresponding mask bits set to 1 represent the network ID. Any address bits that have corresponding mask bits set to 0 represent the node ID.

8.20.15.1 = 00001000.00010100.00001111.00000001

255.0.0.0 = 11111111.00000000.00000000.00000000

-----------------------------------

net id | host id

netid = 00001000 = 8

hostid = 00010100.00001111.00000001 = 20.15.1

## Understanding Subnetting

Subnetting allows you to create multiple logical networks that exist within a single Class A, B, or C network. If you do not subnet, you are only able to use one network from your Class A, B, or C network, which is unrealistic.

Each data link on a network must have a unique network ID, with every node on that link being a member of the same network. If you break a major network (Class A, B, or C) into smaller subnetworks, it allows you to create a network of interconnecting subnetworks. Each data link on this network would then have a unique network/subnetwork ID. Any device, or gateway, connecting *n* networks/subnetworks has *n* distinct IP addresses, one for each network / subnetwork that it interconnects.

In order to subnet a network, extend the natural mask using some of the bits from the host ID portion of the address to create a subnetwork ID. For example, given a Class C network of 204.17.5.0 which has a natural mask of 255.255.255.0, you can create subnets in this manner:

204.17.5.0 - 11001100.00010001.00000101.00000000

255.255.255.224 - 11111111.11111111.11111111.11100000

--------------------------|sub|----

By extending the mask to be 255.255.255.224, you have taken three bits (indicated by "sub") from the original host portion of the address and used them to make subnets. With these three bits, it is possible to create eight subnets. With the remaining five host ID bits, each subnet can have up to 32 host addresses, 30 of which can actually be assigned to a device *since host ids of all zeros or all ones are not allowed* (it is very important to remember this). So, with this in mind, these subnets have been created.

204.17.5.0 255.255.255.224 host address range 1 to 30

204.17.5.32 255.255.255.224 host address range 33 to 62

204.17.5.64 255.255.255.224 host address range 65 to 94

204.17.5.96 255.255.255.224 host address range 97 to 126

204.17.5.128 255.255.255.224 host address range 129 to 158

204.17.5.160 255.255.255.224 host address range 161 to 190

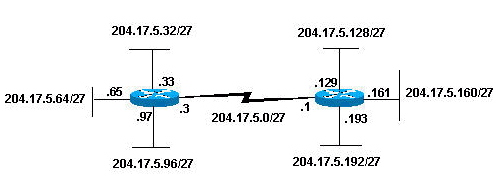
204.17.5.192 255.255.255.224 host address range 193 to 222

204.17.5.224 255.255.255.224 host address range 225 to 254

**Note:**There are two ways to denote these masks. First, since you are using three bits more than the "natural" Class C mask, you can denote these addresses as having a 3-bit subnet mask. Or, secondly, the mask of 255.255.255.224 can also be denoted as /27 as there are 27 bits that are set in the mask. This second method is used with [CIDR](http://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html#cidr). With this method, one of these networks can be described with the notation prefix/length. For example, 204.17.5.32/27 denotes the network 204.17.5.32 255.255.255.224. When appropriate the prefix/length notation is used to denote the mask throughout the rest of this document.

The network subnetting scheme in this section allows for eight subnets, and the network might appear as:

**Figure 2**



Notice that each of the routers in [Figure 2](http://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html#figtwo) is attached to four subnetworks, one subnetwork is common to both routers. Also, each router has an IP address for each subnetwork to which it is attached. Each subnetwork could potentially support up to 30 host addresses.

This brings up an interesting point. The more host bits you use for a subnet mask, the more subnets you have available. However, the more subnets available, the less host addresses available per subnet. For example, a Class C network of 204.17.5.0 and a mask of 255.255.255.224 (/27) allows you to have eight subnets, each with 32 host addresses (30 of which could be assigned to devices). If you use a mask of 255.255.255.240 (/28), the break down is:

204.17.5.0 - 11001100.00010001.00000101.00000000

255.255.255.240 - 11111111.11111111.11111111.11110000

--------------------------|sub |---

Since you now have four bits to make subnets with, you only have four bits left for host addresses. So in this case you can have up to 16 subnets, each of which can have up to 16 host addresses (14 of which can be assigned to devices).

Take a look at how a Class B network might be subnetted. If you have network 172.16.0.0 ,then you know that its natural mask is 255.255.0.0 or 172.16.0.0/16. Extending the mask to anything beyond 255.255.0.0 means you are subnetting. You can quickly see that you have the ability to create a lot more subnets than with the Class C network. If you use a mask of 255.255.248.0 (/21), how many subnets and hosts per subnet does this allow for?

172.16.0.0 - 10101100.00010000.00000000.00000000

255.255.248.0 - 11111111.11111111.11111000.00000000

-----------------| sub |-----------

You are using five bits from the original host bits for subnets. This allows you to have 32 subnets (25). After using the five bits for subnetting, you are left with 11 bits for host addresses. This allows each subnet so have 2048 host addresses (211), 2046 of which could be assigned to devices.

**Note:**In the past, there were limitations to the use of a subnet 0 (all subnet bits are set to zero) and all ones subnet (all subnet bits set to one). Some devices would not allow the use of these subnets. Cisco Systems devices allow the use of these subnets when the **ip subnet zero** command is configured.

## CIDR

Classless Interdomain Routing (CIDR) was introduced to improve both address space utilization and routing scalability in the Internet. It was needed because of the rapid growth of the Internet and growth of the IP routing tables held in the Internet routers.

CIDR moves way from the traditional IP classes (Class A, Class B, Class C, and so on). In CIDR , an IP network is represented by a prefix, which is an IP address and some indication of the length of the mask. Length means the number of left-most contiguous mask bits that are set to one. So network 172.16.0.0 255.255.0.0 can be represented as 172.16.0.0/16. CIDR also depicts a more hierarchical Internet architecture, where each domain takes its IP addresses from a higher level. This allows for the summarization of the domains to be done at the higher level. For example, if an ISP owns network 172.16.0.0/16, then the ISP can offer 172.16.1.0/24, 172.16.2.0/24, and so on to customers. Yet, when advertising to other providers, the ISP only needs to advertise 172.16.0.0/16.

## NAT (Network Address Translation)

Primarily NAT was introduced to the world of IT and networking due to the lack of IP addresses, or looking at it from another view, due to the vast amount of growing IT technology relying on IP addresses. To add to this, NAT adds a layer of security, by hiding computers, servers and other IT equipment from the outside world.

**How NAT works**

When computers and servers within a network communicate, they need to be identified to each other by a unique address, in which resulted in the creation of a 32 bit number, and the combinations of these 32 bits would accommodate for over 4 billion unique addresses, known as IP address. This was named IPv4, and although over 4 billion addresses sounds a lot, it really is not considering how fast the world of computers and the internet has grown.

To circumvent this problem, a temporary solution was produced known as NAT. NAT resulted in two types of IP addresses, public and private. A range of private addresses were introduced, which anyone could use, as long as these were kept private within the network and not routed on the internet. The range of private addresses known as RFC 1918 are;

Class A 10.0.0.0 - 10.255.255.255

Class B 172.16.0.0 - 172.31.255.255

Class C 192.168.0.0 - 192.168.255.255

NAT allows you to use these private IP address on the internal network. So within your private network you would assign a unique IP address to all your computers, servers and other IP driven resources, usually done via DHCP. Another company can use the same private IP addresses as well, as long as they are kept internal to their network. So two companies maybe using the same range of IP addresses but because they are private to their network, they are not conflicting with each other.

However when internal hosts do need to communicate to the public network (Internet) then this is where a public address comes into the equation. This address usually purchased from an ISP is a routable public address everyone can see, which would represent your network gateway. This public address would be unique, no one else would use this address.

Now getting to the point; When a host on the internal network with an internal IP address does need to communicate outside it's private network, it would use the public IP address on the network's gateway to identify itself to the rest of the world, and this translation of converting a private IP address to public is done by NAT. For example a computer on an internal address of 192.168.1.10 wanted to communicate with a web server somewhere on the internet, NAT would translate the address 192.168.1.10 to the company's public address, lets call this 1.1.1.1 for example. so that the internal address is identified as the public address when communicating with the outside world. This has to be done because when the web server somewhere on the internet was to reply to this internal computer, it needs to send this to a unique and routable address on the internet, the public address. It can not use the original address of 192.168.1.10, as this is private, none routable and hidden from the outside world. This address, of 1.1.1.1 would be the address of the public address for that company and can be seen by everyone. Now the web server would reply to that public address, 1.1.1.1. NAT would then use its records to translate the packets received from the web server that was destined to 1.1.1.1 back to the internal network address of 192.168.1.10, and though the computer who requested the original info, will receive the requested packets.

Now you can obviously see the two benefits of NAT. Firstly it would save on the IP addresses we use, as every single computer does not need a public address, and also it would hide these private computers from the outside world. Everyone can only see the public address, the rest is hidden behind this public address. So from the internet only the public address on the external interface of the firewall or router can be seen, and nothing beyond it.

**Types of NAT**

Three main types of NAT rules are used today depending on what needs to be accomplished;

**Static NAT**

A pool of public IP addresses are assigned to the NAT device. A private IP address can then be statically mapped to anyone of these public addresses. This type of NATTING scheme is usually used for servers requiring the same IP address always, hence the name "static", so server 1 will always have the same IP address assigned to it, server 2 will have a different public IP address assigned to it and so on.

**Dynamic NAT**

Again the NAT device will consist of a pool of IP addresses. This time though the pool of IP addresses will be used when needed and then given back to the pool. So if computer A needed a public address, it would take one from the pool, then hand it back when done. The next time the same computer wanted an IP address it may be assigned a different public address from the pool, because the one used previously may be in use by another computer, hence the name "dynamic". So users who want to communicate on the internet at any one time will be limited by how many public IP addresses are available in the NAT pool. A company would purchase a number of public IP's depending on their need.

**Port Address Translation (PAT)**

In this type of setup, a company would only have one public IP address assigned to their network, and so everyone would share this one public address when using the internet, browsing the web for example. Yes, you may be asking how can everyone share one address, well the clue lies within the name, Port address translation. When a computer wants to use the internet, the NAT device, using the PAT method will remember the IP address and source port of the internal host. For example 192.168.1.10 with a source port of 55331 wanted to browse Amazon.com. The NAT device will keep a note of this, and when Amazon replies to the public address and the port number of 55331, the NAT device will use the PAT method and look up the port information which maps to the internal computer requesting it. So it would be saying, this information Amazon has sent back to the public address and port number 55331, maps to the IP address 192.168.1.10 who originally requested it, though the information is for that computer. So the connections are uniquely identified by a source port, all using the same public IP but with unique source ports to identify who requested what information.

A company would save a reasonable amount of money and IP addresses using this method because it is only using one IP address. This has been a major factor to why IPv6 has been mentioned for some years now but still not required in most countries.

NAT is also implemented in home based routers and hardware firewalls such as the Netgear's and the Linksys of this world as well as the high end hardware firewalls such as the likes of Cisco and Juniper.

This has proved a valuable feature on hardware firewalls for saving public IP addresses and also a countermeasure for some types of attacks such as a reconnaissance attack.

**Disadvantages of NAT**

As with everything, NAT does have it's drawbacks. Some applications and services such as VPN and video conferencing struggle to process via NAT (Not entirely true as you can most of the time get them configured to work with NAT, but can get a little messy when setting rules up in applications,, routers and firewalls).

Check out <http://www.9tut.com/network-address-translation-nat-tutorial> for more details on NAT.

# What is DHCP?

Dynamic Host Configuration Protocol (DHCP) is a network protocol that enables a server to automatically assign an IP address to a computer from a defined range of numbers (i.e., a scope) configured for a given network.

DHCP assigns an IP address when a system is started, for example:

1. A user turns on a computer with a DHCP client.
2. The client computer sends a broadcast request (called a DISCOVER or DHCPDISCOVER), looking for a DHCP server to answer.
3. The router directs the DISCOVER packet to the correct DHCP server.
4. The server receives the DISCOVER packet. Based on availability and usage policies set on the server, the server determines an appropriate address (if any) to give to the client. The server then temporarily reserves that address for the client and sends back to the client an OFFER (or DHCPOFFER) packet, with that address information. The server also configures the client's DNS servers, WINS servers, NTP servers, and sometimes other services as well.
5. The client sends a REQUEST (or DHCPREQUEST) packet, letting the server know that it intends to use the address.
6. The server sends an ACK (or DHCPACK) packet, confirming that the client has a been given a lease on the address for a server-specified period of time.

**Ethernet**

The most common type of LAN hardware is known as *Ethernet*. In its simplest form, it consists of a single cable with hosts attached to it through connectors, taps, or transceivers. Simple Ethernets are relatively inexpensive to install, which together with a net transfer rate of 10, 100, or even 1,000 Megabits per second, accounts for much of its popularity.

Ethernets come in three flavors: *thick*, *thin*, and *twisted pair*.

**Linux Network Devices :**

The Linux kernel supports a number of hardware drivers for various types of equipment. This section gives a short overview of the driver families available and the interface names they use.

There is a number of standard names for interfaces in Linux, which are listed here. Most drivers support more than one interface, in which case the interfaces are numbered, as in eth0 and eth1:

lo

This is the local loopback interface. It is used for testing purposes, as well as a couple of network applications. It works like a closed circuit in that any datagram written to it will immediately be returned to the host's networking layer. There's always one loopback device present in the kernel, and there's little sense in having more.

eth0, eth1, …

These are the Ethernet card interfaces. They are used for most Ethernet cards, including many of the parallel port Ethernet cards.

tr0, tr1, …

These are the Token Ring card interfaces. They are used for most Token Ring cards, including non-IBM manufactured cards.

sl0, sl1, …

These are the SLIP interfaces. SLIP interfaces are associated with serial lines in the order in which they are allocated for SLIP.

ppp0, ppp1, …

These are the PPP interfaces. Just like SLIP interfaces, a PPP interface is associated with a serial line once it is converted to PPP mode.

plip0, plip1, …

These are the PLIP interfaces. PLIP transports IP datagrams over parallel lines. The interfaces are allocated by the PLIP driver at system boot time and are mapped onto parallel ports. In the *2.0.x* kernels there is a direct relationship between the device name and the I/O port of the parallel port, but in later kernels the device names are allocated sequentially, just as for SLIP and PPP devices.

ax0, ax1, …

These are the AX.25 interfaces. AX.25 is the primary protocol used by amateur radio operators. AX.25 interfaces are allocated and mapped in a similar fashion to SLIP devices.

There are many other types of interfaces available for other network drivers. We've listed only the most common ones.

**Network Configuration Files**

/etc/resolve.conf → List servers for internet domain name resolution.

/etc/hosts → Lists hosts to be resolved locally

Red Hat/Fedora/CentOS

/etc/sysconfig/network → Specify network configuration. eg. Static , , , etc.

/etc/sysconfig/network-scripts/ifcfg-***device →*** Specify TCP network information.

/etc/network/interfaces ***→*** Specify network configuration and devices. eg. Static IP and info, DHCP, etc.

**Configuring Hostname**

This is a three step process:

1. Issue the command: hostname *new-host-name*
2. Change network configuration file: /etc/sysconfig/network  
    Edit entry: HOSTNAME=*new-host-name*
3. Restart systems which relied on the hostname (or reboot):
   * Restart network services: service network restart  
      (or: /etc/init.d/network restart)
   * Restart desktop:
     + Bring down system to console mode: [init](http://man.yolinux.com/cgi-bin/man2html?cgi_command=init) 3
     + Bring up X-Windows: init 5

One may also want to check the file /etc/hosts for an entry using the system name which allows the system to be self aware.

**Activating & De-activating your NIC**

Commands for starting and stopping TCP/IP network services on a Network Interface Card (NIC):

* **Activate:** /sbin/ifup eth0  
   (Also: ifconfig eth0 up ***- Note: Even if no IP address is assigned you can listen.***)
* **De-Activate:** /sbin/ifdown eth0  
   (Also: ifconfig eth0 down)

These scripts use the scripts and NIC config  
 files in /etc/sysconfig/network-scripts/

**Configuring Static IP on Redhat Machine**

You will need the following information in order to complete this configuration:

* IP Address
* Gateway Address
* Broadcast Address
* Netmask
* Network Address
* DNS Server Addresses
* System Hostname

The first thing is to turn off NetworkManager.This service is responsible for requesting DHCP addresses and configuring the network interfaces. Since we are setting them statically we do not need it.  
  
First let's stop NetworkManager:  
service NetworkManager stop  
or /etc/init.d/NetworkManager stop  
  
Now let's make sure it does not start at boot:  
chkconfig NetworkManager off  
  
Let's also make sure the network service is set to start at boot:  
chkconfig network on  
  
Now that NM (NetworkManager) is out of the way we need to edit three files.  
  
/etc/sysconfig/network  
/etc/sysconfig/network-scripts/ifcfg-eth0  
/etc/resolv.conf  
  
First let's edit /etc/sysconfig/network. Here we will need to tell the system to turn on networking, the hostname of the machine and the gateway. Open the file in your favorite text editor and add or change the following lines. Of course you will need to make sure the configuration matches your system, this is just an example.  
NETWORKING=yes  
HOSTNAME=server.domain.com  
GATEWAY=192.168.1.1  
  
Now save and close that file. Let's move on to editing the /etc/sysconfig/network-scripts/ifcfg-eth0 file. Open the file in your favorite text editor and add or change the following lines.  
DEVICE=eth0  
BOOTPROTO=static  
BROADCAST=192.168.1.255  
IPADDR=192.168.1.10  
NETMASK=255.255.255.0  
NETWORK=192.168.1.0  
ONBOOT=yes  
  
Now that the network interface is taken care of we need the final important piece of the puzzle, DNS.Without DNS you will not be able to go to a website by name, on connect to anything else on the network unless you know the IP address. To tell the system what DNS servers to use we edit the /etc/resolv.conf file. Open the file in your favorite editor and add or change the following lines (search line is optional).  
search domain.com  
 nameserver 192.168.1.2  
nameserver 192.168.1.3  
  
Now that we have given the system the necessary information all we need to do is restart the network service.  
service network restart

**Configuring dynamic via DHCP**

Edit the following two files.  
  
/etc/sysconfig/network  
/etc/sysconfig/network-scripts/ifcfg-eth0  
  
First let's edit /etc/sysconfig/network. Here we will need to tell the system to turn on networking, the hostname of the machine and the gateway. Open the file in your favorite text editor and add or change the following lines. Of course you will need to make sure the configuration matches your system, this is just an example.  
NETWORKING=yes  
HOSTNAME=server.domain.com  
  
Now save and close that file. Let's move on to editing the /etc/sysconfig/network-scripts/ifcfg-eth0 file. Open the file in your favorite text editor and add or change the following lines.  
DEVICE=eth0  
BOOTPROTO=dhcp  
ONBOOT=yes  
  
Now that we have given the system the necessary information all we need to do is restart the network service.  
service network restart

**Configuring Network Card**

Red hat Enterprise Linux version 5.x and 4.x provides the following tools to make changes to network configuration such as add new card, assign/change IP address/subnet/gateway, and change DNS server and more.

[a] GUI tool - system-config-network

[b] Command line text based GUI tool (No X Windows/Gnome/KDE required) - system-config-network

[c] Edit configuration files stored in /etc/sysconfig/network-scripts/ directory. This method works with remote server over the ssh based session.

## 1. GUI tool system-config-network

Open the X terminal or login using ssh over X based session command (ssh -X user@server-name-here). Type the following command at shell prompt:

Warning: It is important that you configure network interface cards correctly over ssh -X based session; otherwise, you will be locked out due to wrong network configuration.

$ system-config-network &  
Sample outputs:

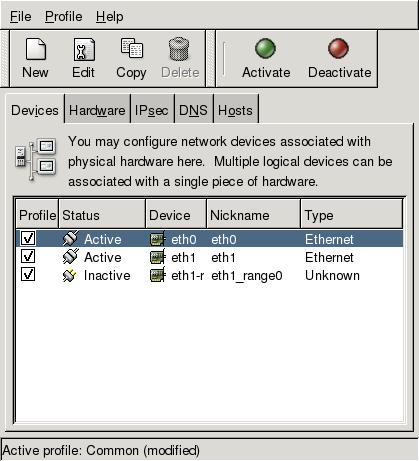


Fig.01: Configuring the RHEL/CentOS server network card using GUI tool

You will see a Window as above. Next, select your Ethernet card (such as eth0 or eth1) and click on the Edit button. You can now setup/modify IP address, netmask, default gateway and other properties. Here is an example from my personal RHEL 5.x server:

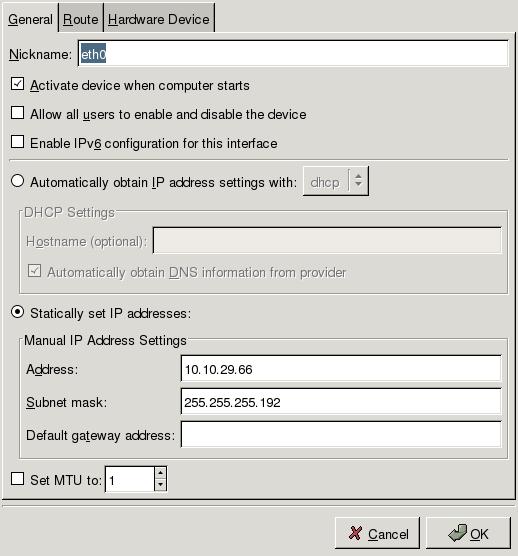


Fig.02: Setting or modifying IPv4 and IPv6 properties on RHEL/CentOS based server

You can obtain IP address using DHCP or setup manually. Once IP address assigned, click on Ok button to save the changes. You can activate card by clicking on Activate button.

## 2. Command line tool system-config-network-tui

If you don't have X windows GUI (gnome/kde desktop) installed on RHEL/CentOS/Fedora based system, than type the following command at shell prompt (this method also works on remote server using ssh based session):

Warning: It is important that you configure network interface cards correctly over ssh based session; otherwise, you will be locked out due to wrong network configuration.

# system-config-network-tui &  
Sample outputs:

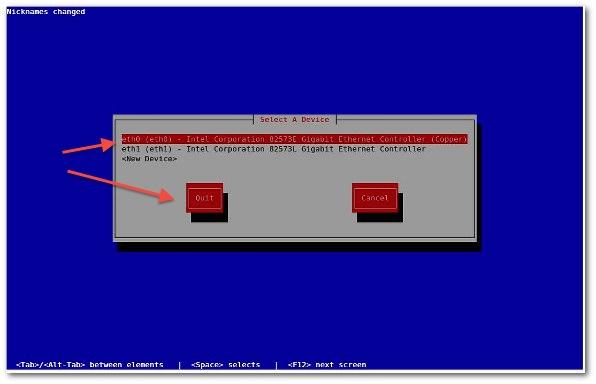


Fig.03: RHEL/CentOS command line network config tool (click to large)

Select your Ethernet card such as eth0 or eth1 and hit [Enter] or [F12] special key to configure IP properties for selected NIC:

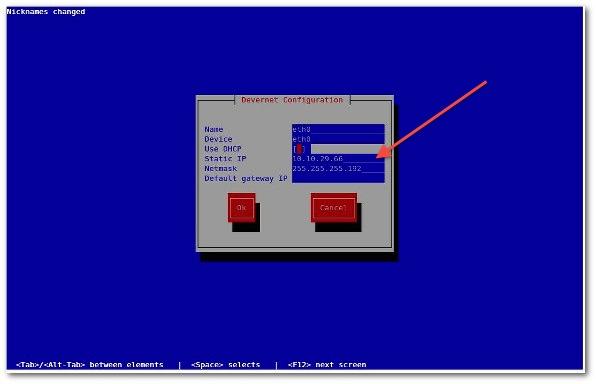


Fig.04: Setting up IPv4 properties on RHEL/CentOS (click to large)

You can obtain an IP address using DHCP or setup IP address manually. Once an IP address assigned, click on the Ok button to save the changes.

## Edit configuration files stored in /etc/sysconfig/network-scripts/ directory

You can configure network card by editing text files stored in /etc/sysconfig/network-scripts/ directory. Open the terminal or login using ssh. Next, change directory to /etc/sysconfig/network-scripts/:  
# cd /etc/sysconfig/network-scripts/  
You need to edit / create files as follows using a text editor such as vi:

* /etc/sysconfig/network-scripts/ifcfg-eth0 : First Ethernet card configuration file.
* /etc/sysconfig/network-scripts/ifcfg-eth1 : Second Ethernet card configuration file.

Examples: Edit eth0 configuration file

To edit/create first NIC file, type the following command in /etc/sysconfig/network-scripts/ directory:

# vi ifcfg-eth0  
OR  
# vi /etc/sysconfig/network-scripts/ifcfg-eth0  
Edit or modify as follows for static ip configuration:

# eth0 - Intel Corporation 82573E Gigabit Ethernet Controller (Copper) on server1.cyberciti.biz by nixCraft on 10/4/2007

DEVICE=eth0

BOOTPROTO=static

DHCPCLASS=

HWADDR=00:30:48:56:A6:2E

IPADDR=10.10.29.66

NETMASK=255.255.255.192

ONBOOT=yes

Save and close the file.

Setting up default gateway and server (host) name

You need to define a default gateway (router IP) and hostname in /etc/sysconfig/network file. Edit /etc/sysconfig/network, enter:  
# vi /etc/sysconfig/network  
Append or modify configuration as follows:

NETWORKING=yes

HOSTNAME=www1.nixcraft.in

GATEWAY=10.10.29.65

Save and close the file. Finally, you need to restart the network service, run:  
# /etc/init.d/network restart  
OR  
# /sbin/service network restart

Setting up dns server

Make sure you have correct DNS server defined in /etc/resolv.conf file:  
# vi /etc/resolv.conf  
Setup DNS server IP address as follows (feel free to replace 10.0.80.11, 10.0.80.12 as per your setup):  
nameserver 10.0.80.11  
nameserver 10.0.80.12  
nameserver 202.67.222.222

Save and close the file. Now you can ping the gateway/other hosts using the ping command:  
$ ping 10.0.80.12

**Firewall**

A firewall is a program running on a Gateway, Bridge or PC/Laptop/Smartphone that is capable of filtering incoming, outgoing, and forwarded network packets. A firewall is essentially a tool that lets you restrict you or your network's access to the Internet, and someone else's access from the Internet to your network.

Linux comes with a host based firewall called Netfilter. According to the official project site:

netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.

This Linux based firewall is controlled by the program called iptables to handles filtering for IPv4, and ip6tables handles filtering for IPv6.

IP Tables is a Lower Level Firewall Implementation Solution available in Linux Systems.

Most Distro’s have other Firewalll options as well:

For examples,

RHEL/ CentOS has Firewalld. Firewalld is a easier to use/implement IP Tables but some programs are not compatible, and would direct interaction with IP Tables.

SELinux ( Security Enhanced Linux) : Here is more info of you are curious about this topic:

https://www.digitalocean.com/community/tutorials/an-introduction-to-selinux-on-centos-7-part-1-basic-concepts

IP Tables provides basic low level routing system for how Linux deals with IP Traffic/ Packets.

Contains Several Important Tables:

1) NAT - NW Address Translation

Network address translation (NAT) is a methodology of remapping one IP address space into another by modifying network address information in Internet Protocol (IP) packet headers while they are in transit across a traffic routing device

This is the basic method through which several VMs running on same Hardware Server (Hypervisor) can get unique IP/MAC Addresses etc. The Virtualization programs like VMWare ESXi/ KVM would modify the IP Packets that are received by Hypervisors NIC and map them to appropriate IP Addresses of VM’s the program is managing.

In Cloud Computing world, we also use NAT to provide Outbound Internet Access to VMs running in a private subnet.

2) MANGLE - for modifying packets

3) Filter - Decide which packets gets processed

IP Tables deals with Tables

Tables have Chains

A Chain is a sequential list of rules

A Rule defines particular kinds of packets and where they go.

Filter Table: Default Chains

INPUT

OUTPUT

FORWARD

Each Rule has 2 things:

1) Specification of type of packet - criteria for matching

2) Routing policy - ACCEPT, DROP, REJECT etc

## Iptables Config File

The default config files for RHEL / CentOS / Fedora Linux are:

* /etc/sysconfig/iptables - The system scripts that activate the firewall by reading this file.

## Turn On Firewall

Type the following two commands to turn on firewall:

chkconfig iptables on *#starts the service on boot*

service iptables start

service iptables restart *# restart the firewall*

service iptables stop *# stop the firewall*

## Understanding Firewall

There are total 3 chains:

1. INPUT - The default chain is used for packets addressed to the system. Use this to open or close incoming ports (such as 80,25, and 110 etc) and ip addresses / subnet (such as 202.54.1.20/29).
2. OUTPUT - The default chain is used when packets are generating from the system. Use this open or close outgoing ports and ip addresses / subnets.
3. FORWARD - The default chains is used when packets send through another interface. Usually used when you setup Linux as router. For example, eth0 connected to ADSL/Cable modem and eth1 is connected to local LAN. Use FORWARD chain to send and receive traffic from LAN to the Internet.

**Packet Matching Rules**

1. Each packet starts at the first rule in the chain .
2. A packet proceeds until it matches a rule.
3. If a match found, then control will jump to the specified target (such as REJECT, ACCEPT, DROP).

**Target Meanings**

1. The target ACCEPT means allow packet.
2. The target REJECT means to drop the packet and send an error message to remote host.
3. The target DROP means drop the packet and do not send an error message to remote host or sending host.

**Iptables Match Criteria**

|  |  |
| --- | --- |
| -t <-table-> | If you don't specify a table, then the filter table is assumed. The possible built-in tables include: filter, nat, mangle |
| -j <target> | Jump to the specified target chain when the packet matches the current rule. |
| -A | Append rule to end of a chain |
| -F | Flush. Deletes all the rules in the selected table |
| -p <protocol-type> | Match protocol. Types include, icmp, tcp, udp, and all |
| -s <ip-address> | Match source IP address |
| -d <ip-address> | Match destination IP address |
| -i <interface-name> | Match "input" interface on which the packet enters. |
| -o <interface-name> | Match "output" interface on which the packet exits |

Examples to Understand How To Write a Firewall Rule

1. Rule to accept tcp packets coming on interface eth0 from any address to my machine

**iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1 -p TCP -j ACCEPT**

0/0 – Means any IP address

2. Rule to allow ssh connection from a network

**iptables -A INPUT -i eth0 -p tcp -s 192.168.1.0/2**5 **--dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT**

**iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT**

3. Rule to restrict ICMP echo requests no more than one per second. This will allow the system to filter traffice that characterize DOS.

**iptables -A INPUT -p tcp --syn -m limit --limit 5/s -i eth0 -j ACCEPT**

--syn – used to identify a new tcp connection

A distributed **denial-of-service** (**DDoS**) **attack** occurs when multiple systems flood the bandwidth or resources of a targeted system, usually one or more web servers. Such an **attack** is often the result of multiple compromised systems (for example a botnet) flooding the targeted system with traffic.

4. Rule to Forward packets.

iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.2 -o eth1 -p TCP **--sport 1024:65535 -m multiport --dports 80,443 -j ACCEPT**

**iptables -A FORWARD -d 0/0 -o eth0 -s 192.168.1.**2 **-i eth1 -p TCP -m state --state ESTABLISHED -j ACCEPT**

5. Rule to block an IP address range

**iptables -A INPUT -s 192.168.1.0/24 -j DROP**

There are many blogs that are available online about iptable rules. Following are some of thee references:

- <http://www.linuxhowtos.org/Security/iptables.htm>

- <http://www.tecmint.com/basic-guide-on-iptables-linux-firewall-tips-commands/>

- <http://www.thegeekstuff.com/2011/06/iptables-rules-examples/>

**Miscellaneous**

**Configuring DHCP Server**

The DHCP protocol lets a DHCP client to lease network configuration parameters such as an IP address. In fact parameters to lease are not limited to IP address only and they also include:

* IP addresses and network masks
* Domain Names servers ( DNS )
* Default Gateways
* WINS servers
* Syslog hosts
* Proxy servers
* NTP servers
* X Font servers
* Syslog hosts

Each host set to obtain an IP address dynamically will upon boot send a DHCP request over the network ( by definition this is a broadcast of all 1's ) to discover whether there is a DHCP server available on the network and consequently ask for an network configuration. DHCP client is then obligated to maintain a communication with DHCP server and renew its IP address regularly as dictated by IP address's lease time expiry. In case that DHCP client fails to renew its IP address ( disconnection, client is turned off and etc. ) its IP address expires and DHCP server is free to lease this IP address to another DHCP client.

DHCP server keeps a record of all leased IP addresses and stores them into a file called dhcpd.leases which can be found in /var/lib/dhcp directory ( location of this file may vary depending on Linux system in use ). Having such a file allows DHCP server to keep track of all IP address leases even after its reboot or power failure.  
  
 Here are some advantages of having a DHCP server connected to network:

* No IP address conflicts. DHCP can guarantee that all hosts on the network will have unique IP address. DHCP server keeps a record of all IP addresses assigned and cross reference them with host's MAC addresses.
* Based on the MAC address DHCP allows for a fixed parameter configuration for a specific host
* Efficiency with minimum local client configuration

# 1.DHCP Server Installation

Standard DHCP server implementation available in various Linux distributions is an Open source version maintained by ISC ( Internet System Consortium ). There are currently 3 major versions 2, 3, 4 where version 3 supports backup servers, and version 4 supports IPv6. This article deals only with ISC DHCP v3.

Use the following commands to install DHCP on your Linux server:

Debian and Ubuntu:

# apt-get install dhcp3-server

Redhat & Fedora:

# yum install dhcp

# 2.Basic DHCP Configuration

By default DHCP server configuration does not include any subnets on which DHCP server should lease IP addresses. Therefore, depends on your Linux system you may get a following error message when you attempt to start DHCP with default dhcpd.conf configuration file.

Starting ISC DHCP server: dhcpdcheck syslog for diagnostics. ... failed!

Checking a log files and particularly /var/log/syslog reveals this problem in more detail:

No subnet declaration for eth0 (some IP address).

As it happens very often your server may be connected to multiple network subnets. In order to start DHCP server at least one subnet must be defined in DHCP configuration file /etc/dhcp/dhcpd.conf.  
  
 NOTE: if your server has access to more than one subnet, DHCP requires all subnets to be defined even though there isn't immediate intention to enable DHCP service on that subnet.  
  
 Below is the simplest example of DHCP configuration file:

subnet 10.1.1.0 netmask 255.255.255.0 {

range 10.1.1.3 10.1.1.254;

}

subnet 192.168.0.0 netmask 255.255.0.0 {

}

This configuration file instructs DHCP server to listen for DHCP client requests on subnet 10.1.1.0 with netmask 255.255.255.0. Furthermore, it will assign IP addresses in range 10.1.1.3 - 10.1.1.254. It also defines an empty definition of subnet with network ID 192.168.0.0.  
  
 Alter above code with your subnet and insert it into /etc/dhcp/dhcpd.conf. When ready restart your DHCP server with ( restart command may vary ) :

# /etc/init.d/isc-dhcp-server restart

# 3.DHCP default and max lease time

At this point we can add to our DHCP configuration another setting and that is to set default and max lease time expiry.

* default-lease-time is a value in seconds in which a leased IP address expiry will be set to if DHCP client does not ask for any other specific expiry lease time
* max-lease-time is a value in seconds which defines a maximum expiry time for an IP address leased by DHCP server

default-lease-time 600;

max-lease-time 7200;

subnet 10.1.1.0 netmask 255.255.255.0 {

range 10.1.1.3 10.1.1.254;

}

subnet 192.168.0.0 netmask 255.255.0.0 {

}

# 4.Define DNS server

Another configuration parameter possible to be set by DHCP server to its client is a definition of DNS server. If you want your clients to use DNS server with an IP address 8.8.8.8 and 10.1.1.1 you can do it by including an option "domain-name-servers" to DHCP's configuration file.

default-lease-time 600;

max-lease-time 7200;

subnet 10.1.1.0 netmask 255.255.255.0 {

range 10.1.1.3 10.1.1.254;

option domain-name-servers 10.1.1.1, 8.8.8.8;

}

subnet 192.168.0.0 netmask 255.255.0.0 {

}

subnet 10.1.1.0 netmask 255.255.255.0 {

range 10.1.1.3 10.1.1.254;

option routers 10.1.1.1;

}

# 5.Set default gateway

DHCP also allows for client's gateway configuration.To set any client on the local network to use default gateway 10.1.1.1, add line "option routers 10.1.1.1" into dhcpd.conf file as demonstrated below:

default-lease-time 600;

max-lease-time 7200;

subnet 10.1.1.0 netmask 255.255.255.0 {

range 10.1.1.3 10.1.1.254;

option domain-name-servers 10.1.1.1, 8.8.8.8;

option routers 10.1.1.1;

}

subnet 192.168.0.0 netmask 255.255.0.0 {

}

subnet 10.1.1.0 netmask 255.255.255.0 {

range 10.1.1.3 10.1.1.254;

option routers 10.1.1.1;

}

DHCP will now set DHCP client with gateway 10.1.1.1.

# 6.Host specific configuration

There maybe a need to set static IP address to a particular host on the network such as printer, web server and etc. In this case it is posible to amend DHCP server configuration to lease a choosen IP address to a specific host defined by its MAC address.

default-lease-time 600;

max-lease-time 7200;

subnet 10.1.1.0 netmask 255.255.255.0 {

range 10.1.1.3 10.1.1.254;

option domain-name-servers 10.1.1.1, 8.8.8.8;

option routers 10.1.1.1;

}

subnet 192.168.0.0 netmask 255.255.0.0 {

}

host printer {

hardware ethernet 00:16:d3:b7:8f:86;

fixed-address 10.1.1.100;

}

host web-server {

hardware ethernet 00:17:a4:c2:44:22;

fixed-address 10.1.1.200;

}

The above DHCP configuration file will permanently assign the IP address 10.1.1.100 to a host "printer" with a MAC address 00:16:d3:b7:8f:86 and IP address 10.1.1.200 to host "web-server" with MAC address 00:17:a4:c2:44:22.