



Red Hat System Administration I

Dr. Eng. Gamal Selim

Lecture 06

Agenda

- **Chapter 11: Managing Red Hat Enterprise Linux Networking**

MANAGING RED HAT ENTERPRISE LINUX NETWORKING

GOAL

To configure basic IPv4 networking on Red Hat Enterprise Linux systems.

OBJECTIVES

- Explain fundamental concepts of computer networking.
- Test and review current network configuration with basic utilities.
- Manage network settings and devices with **nmcli** and NetworkManager.
- Modify network settings by editing the configuration files.
- Configure and test system host name and name resolution.

SECTIONS

- Networking Concepts (and Practice)
- Validating Network Configuration (and Practice)
- Configuring Networking with **nmcli** (and Practice)
- Editing Network Configuration Files (and Practice)
- Configuring Host Names and Name Resolution (and Practice)

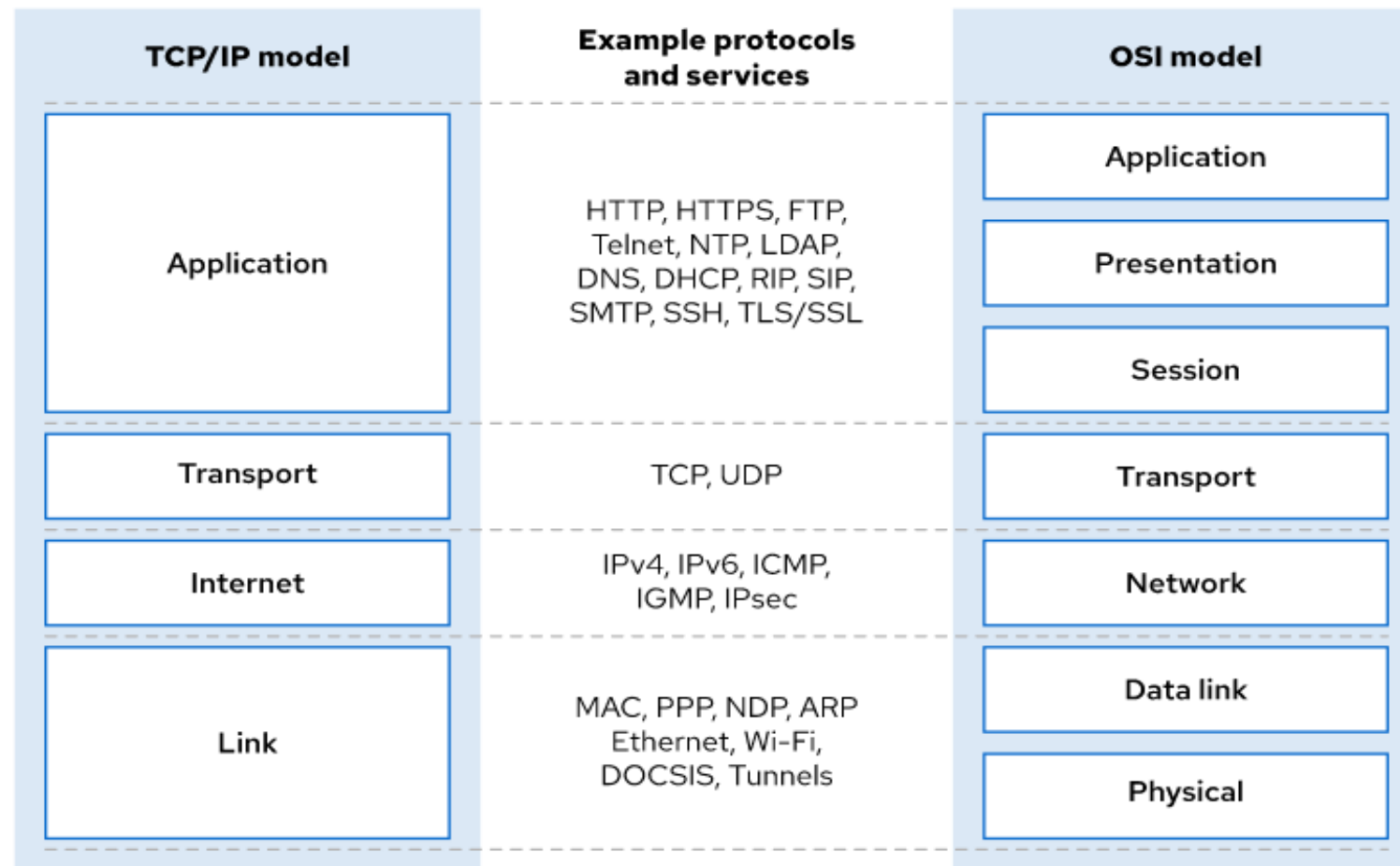
NETWORKING CONCEPTS

OBJECTIVES

After completing this section, students should be able to explain fundamental concepts of computer networking.

IPV4 NETWORKING

TCP/IP standards follow a four-layer network model specified in RFC1122.



- **Application**

Each application has specifications for communication so that clients and servers may communicate across platforms. Common protocols include SSH (remote login), HTTPS (secure web), NFS or CIFS (file sharing), and SMTP (electronic mail delivery).

- **Transport**

Transport protocols are TCP and UDP. TCP is a reliable connection-oriented communication, while UDP is a connectionless *datagram* protocol. Application protocols use TCP or UDP ports. A list of well-known and registered ports can be found in the **/etc/services** file.

When a packet is sent on the network, the combination of the service port and IP address forms a socket. Each packet has a source socket and a destination socket. This information can be used when monitoring and filtering.

- **Internet**

The Internet, or network layer, carries data from the source host to the destination host. Each host has an IP address and a prefix used to determine network addresses. Routers are used to connect networks.

ICMP is a control protocol at this layer. Instead of ports, it has types. The **ping** utility is an example of using ICMP packets to test connectivity. **ping** sends an ICMP ECHO_REQUEST packet. A successful **ping** receives an ICMP ECHO_REPLY acknowledgment. An unsuccessful **ping** may receive ICMP error messages such as "destination unreachable" or may not receive any response.

- **Link**

The link, or media access, layer provides the connection to physical media. The most common types of networks are wired Ethernet (802.3) and wireless WLAN (802.11). Each physical device has a hardware address (MAC) which is used to identify the destination of packets on the local network segment.

IPv4 addresses

An IPv4 address is a 32-bit number, normally expressed in decimal as four *octets* ranging in value from 0 to 255, separated by dots. The address is divided into two parts: the *network part* and the *host part*. All hosts on the same subnet, which can talk to each other directly without a router, have the same network part; the network part identifies the subnet. No two hosts on the same subnet can have the same host part; the host part identifies a particular host on a subnet.

In the modern Internet, the size of an IPv4 subnet is variable. To know which part of an IPv4 address is the network part and which is the host part, an administrator must know the *netmask* which is assigned to the subnet. The netmask indicates how many bits of the IPv4 address belong to the subnet. The more bits that are available for the host part, the more hosts can be on the subnet.

The lowest possible address on a subnet (host part is all zeros in binary) is sometimes called the *network address*. The highest possible address on a subnet (host part is all ones in binary) is used for broadcast messages in IPv4, and is called the *broadcast address*.

IP Address:

172.17.5.3 = 10101100.00010001.00000101.00000011

Prefix: /16

Netmask:

255.255.0.0 = 11111111.11111111.00000000.00000000

10101100.00010001.00000101.00000011

Network

Host

IP Address:

192.168.5.3 = 11000000.10101000.00000101.00000011

Prefix: /24

Netmask:

255.255.255.0 = 11111111.11111111.11111111.00000000

11000000.10101000.00000101.00000011

Network

Host

Figure 11.1: IPv4 addresses and netmasks

Network masks are expressed in two forms. The older syntax for a netmask which uses 24 bits for the network part would read `255.255.255.0`. A newer syntax, called CIDR notation, would specify a *network prefix* of `/24`. Both forms convey the same information; namely, how many leading bits in the IP address contribute to its network address.

The examples which follow illustrate how the IP address, prefix (netmask), network part, and host part are related.

Calculating the network address for 192.168.1.107/24

Host addr	192.168.1.107	11000000.10101000.00000001.01101011
Network prefix	/24 (255.255.255.0)	11111111.11111111.11111111.00000000
Network addr	192.168.1.0	11000000.10101000.00000001.00000000
Broadcast addr	192.168.1.255	11000000.10101000.00000001.11111111

Calculating the network address for 10.1.1.18/8

Host addr	10.1.1.18	00001010.00000001.00000001.00010010
Network prefix	/8 (255.0.0.0)	11111111.00000000.00000000.00000000
Network addr	10.0.0.0	00001010.00000000.00000000.00000000
Broadcast addr	10.255.255.255	00001010.11111111.11111111.11111111

Calculating the network address for 172.16.181.23/19

Host addr	172.168.181.23	10101100.10101000.10110101.00010111
Network prefix	/19 (255.255.224.0)	11111111.11111111.11100000.00000000
Network addr	172.168.160.0	10101100.10101000.10100000.00000000
Broadcast addr	172.168.191.255	10101100.10101000.10111111.11111111

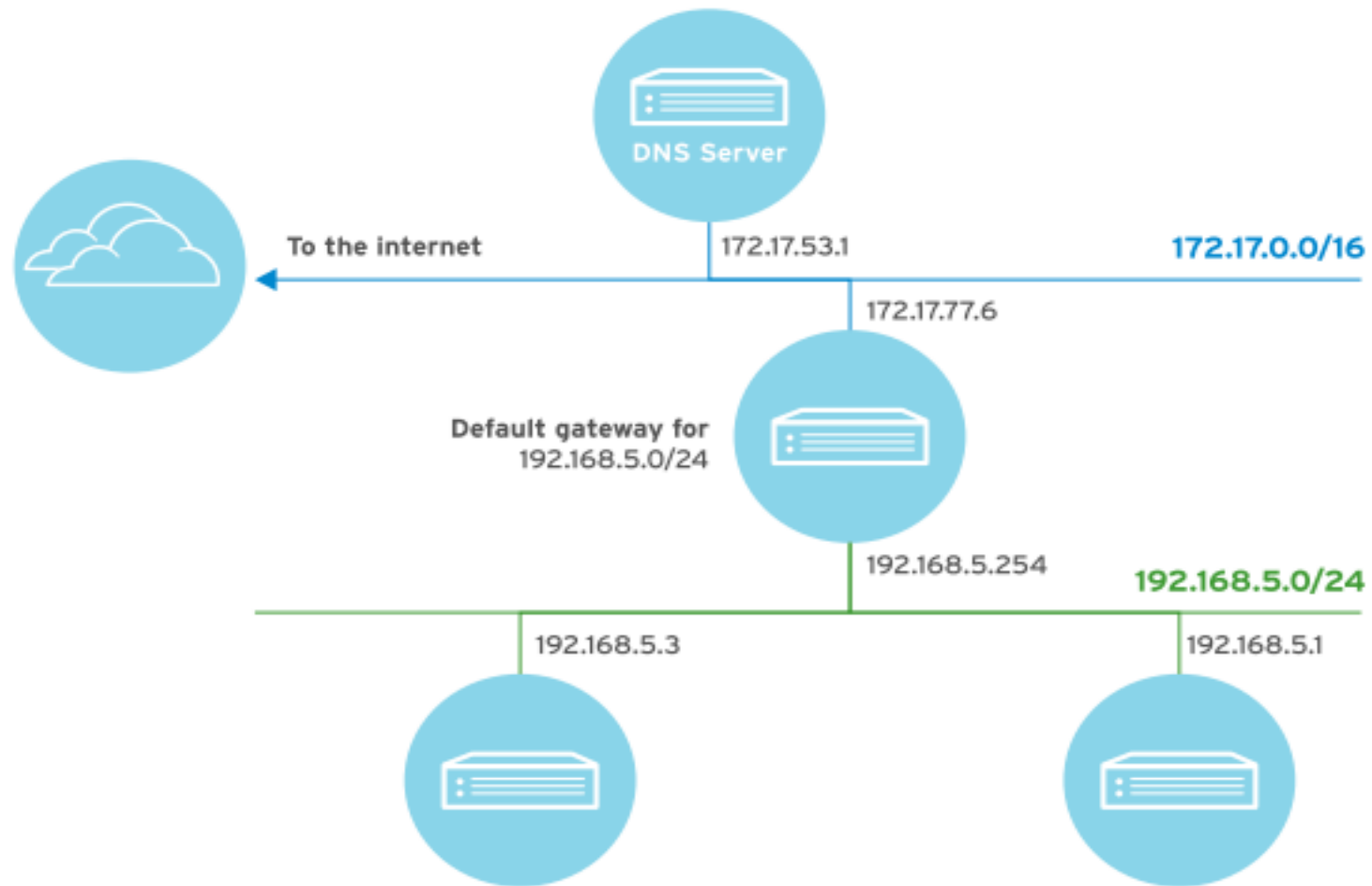


Figure 11.2: Network routing and DNS concepts

IPv4 routing

Whether using IPv4 or IPv6, network traffic needs to move from host to host and network to network. Each host has a *routing table*, which tells it how to route traffic for particular networks. The routing table entries will list a destination network, which interface to send the traffic out, and the IP address of any intermediate router that is required to relay the message to its final destination. The routing table entry which matches the destination of the network traffic is used to route it. If two entries match, the one with the longest prefix is used.

If the network traffic does not match a more specific route, the routing table usually has an entry for a *default route* to the entire IPv4 Internet, 0.0.0.0/0. This default route points to a *router* on a reachable subnet (that is, on a subnet that has a more specific route in the host's routing table).

If a router receives traffic that is not addressed to it, instead of ignoring it like a normal host, it *forwards* the traffic based on its own routing table. This may send the traffic directly to the destination host (if the router happens to be on the destination's subnet), or it may be forwarded on to another router. This process of forwarding continues until the traffic reaches its final destination.

Example routing table

DESTINATION	INTERFACE	ROUTER (IF NEEDED)
192.0.2.0/24	wlo1	
192.168.5.0/24	enp3s0	
0.0.0.0/0 (<i>default</i>)	enp3s0	192.168.5.254

Names and IP addresses

The IP protocol uses addresses to communicate, but human beings would rather work with names than long and hard-to-remember strings of numbers. *DNS*, the Domain Name System, is a distributed network of servers that maps host names to IP addresses. In order for name service to work, the host needs to be pointed at a *nameserver*. This nameserver does not need to be on the same subnet; it just needs to be reachable by the host.

DHCP or static network configuration

Many systems are configured to obtain network settings automatically at boot time. The local configuration files indicate that DHCP should be used and a separate client service queries the network for a server and obtains a lease for network settings.

If a DHCP server is not available, the system must use a *static* configuration where the network settings are read from a local configuration file. The correct network settings are obtained from the network administrator or architecture team to ensure there are no conflicts with other systems.

Since DHCP uses the hardware address to track assignments, only one address may be assigned per interface with DHCP. Multiple static addresses may be assigned to a single interface. This practice is common in systems hosting services for multiple clients, such as HTTP IP-based hosting. Red Hat Enterprise Linux interfaces typically have an IPv4 address and an IPv6 local-link address, but may have more addresses assigned.

NETWORK INTERFACE NAMES

Traditionally, network interfaces in Linux are enumerated as **eth0**, **eth1**, **eth2**, and so on. However, the mechanism which sets these names can cause changes to which interface gets which name as devices are added and removed. The default naming behavior in Red Hat Enterprise Linux 7 is to assign fixed names based on firmware, device topology, and device type. Interface names have the following characters:

- Ethernet interfaces begin with *en*, WLAN interfaces begin with *wl*, and WWAN interfaces begin with *ww*.
- The next character(s) represents the type of adapter with an *o* for on-board, *s* for hotplug slot, and *p* for PCI geographic location. Not used by default but also available to administrators, an *x* is used to incorporate a MAC address.
- Finally, a number *N* is used to represent an index, ID, or port.
- If the fixed name cannot be determined, the traditional names such as *ethN* will be used.

For example, the first embedded network interface may be named **eno1** and a PCI card network interface may be named **enp2s0**. The new names make it easier to distinguish the relationship between a port and its name if the user knows both, but the trade off is that users cannot assume a system with one interface calls that interface **eth0**.

VALIDATING NETWORK CONFIGURATION

OBJECTIVES

After completing this section, students should be able to test and review current network configuration with basic utilities.

DISPLAYING IP ADDRESSES

The **/sbin/ip** command is used to show device and address information.

```
[student@desktopX ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST, ❶UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    ❷link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
    ❸inet 172.25.0.10/24 brd ❹172.25.0.255 scope global eth0
        valid_lft forever preferred_lft forever
    ❺inet6 fe80::5054:ff:fe00:b/64 scope link
        valid_lft forever preferred_lft forever
```

- ❶ An active interface has the status of **UP**.
- ❷ The link line specifies the hardware (MAC) address of the device.
- ❸ The inet line shows the IPv4 address and prefix.
- ❹ The broadcast address, scope, and device name are also on this line.
- ❺ The inet6 line shows IPv6 information.

The **ip** command may also be used to show statistics about network performance. The received (RX) and transmitted (TX) packets, errors, and dropped counters can be used to identify network issues caused by congestion, low memory, and overruns.

```
[student@desktopX ~]$ ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    269850    2931      0       0         0         0
    TX: bytes  packets  errors  dropped carrier collsns
    300556    3250      0       0         0         0
```

TROUBLESHOOTING ROUTING

The **/sbin/ip** command is also used to show routing information.

```
[student@desktopX ~]$ ip route
default via 172.25.0.254 dev eth0 proto static metric 1024
172.25.X.0/24 dev eth0 proto kernel scope link src 172.25.X.10
10.0.0.0/8 dev eth1 proto kernel scope link src 10.0.0.11
```

To trace the path to a remote host, use either **tracert** or **tracert**. Both commands can be used to trace a path with UDP packets; however, many networks block UDP and ICMP traffic. The **tracert** command has options to trace the path with UDP (default), ICMP (-I), or TCP (-T) packets, but may not be installed by default.

```
[student@desktopX ~]$ tracert access.redhat.com
...
 4:  71-32-28-145.rcmt.qwest.net                48.853ms  asymm   5
 5:  dcp-brdr-04.inet.qwest.net                  100.732ms  asymm   7
 6:  206.111.0.153.ptr.us.xo.net                  96.245ms  asymm   7
 7:  207.88.14.162.ptr.us.xo.net                   85.270ms  asymm   8
 8:  ae1d0.cir1.atlanta6-ga.us.xo.net              64.160ms  asymm   7
 9:  216.156.108.98.ptr.us.xo.net                 108.652ms
10:  bu-ether13.atlengamq46w-bcr00.tbone.rr.com   107.286ms  asymm  12
...
```

TROUBLESHOOTING PORTS AND SERVICES

TCP services use sockets as end points for communication and are made up of an IP address, protocol, and port number. Services typically listen on standard ports while clients use a random available port. Well-known names for standard ports are listed in the `/etc/services` file.

The `ss` command is used to display socket statistics. The `ss` command is meant to replace the older tool `netstat`, included in the `net-tools` package, which may be more familiar to some system administrators but which may not always be installed.

```
[student@desktopX ~]$ ss -ta
State      Recv-Q Send-Q      Local Address:Port      Peer Address:Port
LISTEN     0      128             *:sunrpc                 *:*LISTEN     0      128             ① *:ssh                    *:*LISTEN     0      100             ② 127.0.0.1:smtp           *:*LISTEN     0      128             *:36889                  *:*ESTAB      0      0             ③ 172.25.X.10:ssh          172.25.254.254:59392
LISTEN     0      128             :::sunrpc                 :::*
LISTEN     0      128             ④ :::ssh                   :::*
```

- ① The port used for SSH is listening on all IPv4 addresses. The "*" is used to represent "all" when referencing IPv4 addresses or ports.
- ② The port used for SMTP is listening on the 127.0.0.1 IPv4 loopback interface.
- ③ The established SSH connection is on the 172.25.X.10 interface and originates from a system with an address of 172.25.254.254.
- ④ The port used for SSH is listening on all IPv6 addresses. The "::" syntax is used to represent all IPv6 interfaces.

CONFIGURING NETWORKING WITH `nmccli`

OBJECTIVES

After completing this section, students should be able to manage network settings and devices with `nmccli` and NetworkManager.

NETWORKMANAGER

NetworkManager is a daemon that monitors and manages network settings. In addition to the daemon, there is a GNOME Notification Area applet that provides network status information. Command-line and graphical tools talk to NetworkManager and save configuration files in the `/etc/sysconfig/network-scripts` directory.

VIEWING NETWORK INFORMATION WITH `nmcli`

To display a list of all connections, use `nmcli con show`. To list only the active connections, add the `--active` option.

```
[root@desktopX ~]# nmcli con show
```

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	eth0
System eth0	5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03	802-3-ethernet	--
guest	f601ca8a-6647-4188-a431-dab48cc63bf4	802-11-wireless	wlp3s0

```
[root@desktopX ~]# nmcli con show --active
```

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	eth0
guest	f601ca8a-6647-4188-a431-dab48cc63bf4	802-11-wireless	wlp3s0

Specify a connection ID (name) to see the details of that connection. The lowercase settings represent the configuration of the connection. Setting and property names are defined in the **nm-settings(5)** man page. The uppercase settings are active data.

```
[root@desktopX ~]# nmcli con show "static-eth0"
...
ipv4.method:                                manual
ipv4.dns:                                    172.25.254.254, 8.8.8.8
ipv4.dns-search:
ipv4.addresses:                             { ip = 172.25.X.10/24, gw = 172.25.X.254 }
ipv4.routes:
ipv4.ignore-auto-routes:                    no
ipv4.ignore-auto-dns:                      no
ipv4.dhcp-client-id:                        --
ipv4.dhcp-send-hostname:                    yes
ipv4.dhcp-hostname:                         --
```

The **nmcli** command can also be used to show device status and details.

```
[root@desktopX ~]# nmcli dev status
DEVICE   TYPE      STATE      CONNECTION
eth0     ethernet  connected  static-eth0
wlp3s0   wifi      connected  guest
lo       loopback  unmanaged  --
[root@desktopX ~]# nmcli dev show eth0
GENERAL.DEVICE:                eth0
GENERAL.TYPE:                  ethernet
GENERAL.HWADDR:                52:54:00:00:00:0A
GENERAL.MTU:                   1500
GENERAL.STATE:                 100 (connected)
GENERAL.CONNECTION:            static-eth0
GENERAL.CON-PATH:              /org/freedesktop/NetworkManager/
ActiveConnection/1
WIRED-PROPERTIES.CARRIER:     on
IP4.ADDRESS[1]:                ip = 172.25.X.10/24, gw = 172.25.X.254
IP4.DNS[1]:                    172.25.254.254
IP6.ADDRESS[1]:                ip = fe80::5054:ff:fe00:b/64, gw = ::
```

CREATING NETWORK CONNECTIONS WITH **nmcli**

When creating a new connection with **nmcli**, the order of the arguments is important. The common arguments appear first and must include the type and interface. Next, specify any type-specific arguments and finally specify the IP address, prefix, and gateway information. Multiple IP addresses may be specified for a single device. Additional settings such as a DNS server are set as modifications once the connection exists.

Examples of creating new connections

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Define a new connection named "default" which will autoconnect as an Ethernet connection on the eth0 device using DHCP.

```
[root@desktopX ~]#  
nmcli con add con-name "default" type ethernet ifname eth0
```

2. Create a new connection named "static" and specify the IP address and gateway. Do not autoconnect.

```
[root@desktopX ~]#  
nmcli con add con-name "static" ifname eth0 autoconnect no type ethernet  
ip4 172.25.X.10/24 gw4 172.25.X.254
```

3. The system will autoconnect with the DHCP connection at boot. Change to the static connection.

```
[root@desktopX ~]#  
nmcli con up "static"
```

4. Change back to the DHCP connection.

```
[root@desktopX ~]#  
nmcli con up "default"
```

Type options

Type options depend on the type used. An ethernet-type connection may optionally specify a MAC address for the connection. A wifi-type connection must specify the SSID and may specify additional options. Many other types are available, including bridge, bond, team, VPN, and VLAN. To view all the options, use **nmcli con add help**.

```
[root@desktopX ~]# nmcli con add help
Usage: nmcli connection add { ARGUMENTS | help }

ARGUMENTS := COMMON_OPTIONS TYPE_SPECIFIC_OPTIONS IP_OPTIONS

COMMON_OPTIONS:
    type <type>
    ifname <interface name> | ""
    [con-name <connection name>
    [autoconnect yes|no]

    [save yes|no]

TYPE_SPECIFIC_OPTIONS:
    ethernet:    [mac <MAC address>
                  [cloned-mac <cloned MAC address>
                  [mtu <MTU>
...

```

MODIFYING NETWORK INTERFACES WITH `nmcli`

An existing connection may be modified with `nmcli con mod` arguments. The arguments are sets of key/value pairs. The key includes a setting name and a property name. Use `nmcli con show "<ID>"` to see a list of current values for a connection. The `nm-settings(5)` man page documents the setting and property names and usage.

```
[root@desktopX ~]# nmcli con show "static"
connection.id:                static
connection.uuid:              f3e8dd32-3c9d-48f6-9066-551e5b6e612d
connection.interface-name:    eth0
connection.type:              802-3-ethernet
connection.autoconnect:       yes
connection.timestamp:         1394905322
connection.read-only:         no
...
```

Examples of connection modifications

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Turn off autoconnect.

```
[root@desktopX ~]#  
nmcli con mod "static" connection.autoconnect no
```

2. Specify a DNS server.

```
[root@desktopX ~]#  
nmcli con mod "static" ipv4.dns 172.25.X.254
```

3. Some configuration arguments may have values added or removed. Add a +/- symbol in front of the argument. Add a secondary DNS server.

```
[root@desktopX ~]#  
nmcli con mod "static" +ipv4.dns 8.8.8.8
```

4. Replace the static IP address and gateway.

```
[root@desktopX ~]#  
nmcli con mod "static" ipv4.addresses "172.25.X.10/24 172.25.X.254"
```

5. Add a secondary IP address without a gateway.

```
[root@desktopX ~]#  
nmcli con mod "static" +ipv4.addresses 10.10.10.10/16
```

SUMMARY OF nmcli COMMANDS

Basic device and connection commands for **nmcli**:

nmcli commands

COMMAND	USE
nmcli dev status	List all devices.
nmcli con show	List all connections.
nmcli con up "<ID>"	Activate a connection.
nmcli con down "<ID>"	Deactivate a connection. The connection will restart if autoconnect is yes.
nmcli dev dis <DEV>	Bring down an interface and temporarily disable autoconnect.
nmcli net off	Disable all managed interfaces.
nmcli con add ...	Add a new connection.
nmcli con mod "<ID>" ...	Modify a connection.
nmcli con del "<ID>"	Delete a connection.

EDITING NETWORK CONFIGURATION FILES

OBJECTIVES

After completing this section, students should be able to modify network settings by editing the configuration files.

MODIFYING NETWORK CONFIGURATION

It is also possible to configure the network by editing interface configuration files. Interface configuration files control the software interfaces for individual network devices. These files are usually named `/etc/sysconfig/network-scripts/ifcfg-<name>`, where `<name>` refers to the name of the device or connection that the configuration file controls. The following are standard variables found in the file used for static or dynamic configuration.

Configuration Options for ifcfg File

<i>STATIC</i>	<i>DYNAMIC</i>	<i>EITHER</i>
BOOTPROTO=none IPADDR0=172.25.X.10 PREFIX0=24 GATEWAY0=172.25.X.254 DEFROUTE=yes DNS1=172.25.254.254	BOOTPROTO=dhcp	DEVICE=eth0 NAME="System eth0" ONBOOT=yes UUID=f3e8dd32-3... USERCTL=yes

In the static settings, variables for IP address, prefix, and gateway have a number at the end. This allows multiple sets of values to be assigned to the interface. The DNS variable also has a number which is used to specify the order of lookup when multiple servers are specified.

After modifying the configuration files, run **nmcli con reload** to make NetworkManager read the configuration changes. The interface still needs to be restarted for changes to take effect.

```
[root@serverX ~]#  
nmcli con reload  
[root@serverX ~]#  
nmcli con down "System eth0"  
[root@serverX ~]#  
nmcli con up "System eth0"
```

CONFIGURING HOST NAMES AND NAME RESOLUTION

OBJECTIVES

After completing this section, students should be able to configure and test system host name and name resolution.

CHANGING THE SYSTEM HOST NAME

The **hostname** command displays or temporarily modifies the system's fully qualified host name.

```
[root@desktopX ~]# hostname  
desktopX.example.com
```

```
[root@desktopX ~]# hostnamectl set-hostname desktopX.example.com
[root@desktopX ~]# hostnamectl status
  Static hostname: desktopX.example.com
    Icon name: computer
    Chassis: n/a
  Machine ID: 9f6fb63045a845d79e5e870b914c61c9
    Boot ID: aa6c3259825e4b8c92bd0f601089ddf7
  Virtualization: kvm
  Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)
    CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:GA:server
    Kernel: Linux 3.10.0-97.el7.x86_64
  Architecture: x86_64
[root@desktopX ~]# cat /etc/hostname
desktopX.example.com
```

CONFIGURING NAME RESOLUTION

The *stub resolver* is used to convert host names to IP addresses or the reverse. The contents of the file **/etc/hosts** are checked first.

```
[root@desktopX ~]# cat /etc/hosts
127.0.0.1      localhost localhost.localdomain localhost4 localhost4.localdomain4
::1           localhost localhost.localdomain localhost6 localhost6.localdomain6

172.25.254.254 classroom.example.com
```

The **getent hosts hostname** command can be used to test host name resolution with the **/etc/hosts** file.

If an entry is not found in that file, the stub resolver looks for the information from a DNS nameserver. The **/etc/resolv.conf** file controls how this query is done:

- **nameserver**: the IP address of a nameserver to query. Up to three nameserver directives may be given to provide backups if one is down.
- **search**: a list of domain names to try with a short host name. Both this and **domain** should not be set in the same file; if they are, the last instance wins. See **resolv.conf(5)** for details.

```
[root@desktopX ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager will update the **/etc/resolv.conf** file using DNS settings in the connection configuration files. Use the **nmcli** to modify the connections.

```
[root@desktopX ~]# nmcli con mod ID ipv4.dns IP
[root@desktopX ~]# nmcli con
down ID
[root@desktopX ~]# nmcli con up ID
[root@desktopX ~]# cat /etc/sysconfig/
network-scripts/ifcfg-ID
...
DNS1=8.8.8.8
```

The default behavior of **nmcli con mod ID ipv4.dns IP** is to replace any previous DNS settings with the new IP list provided. A +/- symbol in front of the **ipv4.dns** argument will add or remove an individual entry.

```
[root@desktopX ~]#  
nmcli con mod ID +ipv4.dns IP
```

The **host HOSTNAME** command can be used to test DNS server connectivity.

```
[root@desktopX ~]# host classroom.example.com  
classroom.example.com has address 172.25.254.254  
[root@desktopX ~]# host 172.25.254.254  
254.254.25.172.in-addr.arpa domain name pointer classroom.example.com.
```

End of Chapter 11

*Thank
you*

