Chapter 1

Configuring Advanced Networking

1.1 Networking Basics Resumed

1.1.1 Network configuration tools

Terms	Description
ip addr show	Shows address information about all network interfaces.
ip -s link show ens33	Shows statistics about packets but for interface `ens33`. Same as `ip -s link`, but for a specific interface.
ip route	Shows routing information
traceroute / **tracepath**	For analysing a particular route or path.
netstat / **ss**	Analyse ports and services currently listening for incoming connections.

1.1.2 Network Manager

NetworkManager is used to both manage and monitor network settings. While the settings made with the IP tool act directly on the NICs, they're temporary and wiped with every boot or even bringing the interface down and up again. The network manager uses config scripts in `/etc/sysconfig/network-scripts` to store our configs and use them after every boot. The settings can be managed using either `nmcli` or `nmtui`. The former is preferred for scripts while `nmtui` is preferred for manual configs.

nmcli concepts

- A **device** or an **interface** is a network interface, corresponding to the hardware NIC (Network Interface Card).
- A **connection** is a collection of configuration settings for a *device*.
- Multiple connections can exist for the same device, but since they operate on the same settings for the device, only one of them can be active.

- All the connections (and some details) can be shown with the command `nmcli con show`.
- To show all the details for a particular connection, we have to use the command `nmcli con show <interface name>` like `nmcli con show wlo1` (where *wlo1* is the name of the connection).
- To see the connection status for a device, we use `nmcli dev status`. This shows us which devices are connected and which connection they're presently using.
- To see the details of the actual NIC device, we use `nmcli dev show <deviceName>`.

1.1.3 Creating Network Interfaces with nmcli

To add a new connection using `nmcli` that has the name *dhcp* that auto-connects using dynamic IP on interface *eno1*, we use:

```
# nmcli con add con-name "dhcp" type ethernet ifname eno1
```

To add a new connection *static* that uses a static ip that doesn't connect automatically, we use:

```
# nmcli con add con-name "static" type ethernet ifname eno1 autoconnect no ip4
↪  192.168.122.102 gw4 192.168.122.1
```

Now, the available connections can be checked with `nmcli dev status`. The we can connect the *static* connection using `nmcli con up static` and then switch back to the original connection *dhcp* using `nmcli con up dhcp`.

1.1.4 Modifying Network Interfaces using nmcli

To see the details of the *static* connection, we use `nmcli con show static`. Then, to add/modify the DNS server address for that connection, we use the `con mod` keywords, which makes the command:

```
# nmcli con mod "static" ipv4.dns 192.168.122.1
```

Note that the modification requires the `ipv4` keyword instead `ip4`. To define a second IPv4 DNS for the *static* connection, we use the + symbol to denote that a new value for the item should be added and the old value shouldn't be overwritten. The command then becomes:

```
# nmcli con mod "static" +ipv4.dns 8.8.8.8
```

An existing static IP address and gateway can be edited using:

```
# nmcli con mod "static" ipv4.addresses "192.168.100.10/24 192.168.100.1"
```

A secondary IPv4 address can be added using:

```
# nmcli con mod "static" +ipv4.addresses "10.0.0.10/24"
```

Finally, to activate all the above settings, we use: `nmcli con up static`.

1.1.5 Working directly with Configuration Files

All the `nmcli` tool really does while adding or modifying settings is write the changes to the configuration files in `/etc/sysconfig/network-scripts/ifcfg-<interfaceName>`. We may choose to edit them directly if needed. Then, after making the necessary modifications, we ask the NetworkManager service to reload the configuration using `nmcli con reload`.

1.1.6 Managing Hostname and DNS

The hostname is stored in the file `/etc/hostname` and can be edited directly or using the `hostnamectl set-hostname <newHostName>` command. The current hostname can then be viewed using `hostnamectl status`.

The value of the search domain and preferred nameserver (i.e., the one that the NetworkManager uses by default) is auto-pushed from `/etc/sysconfif/network-scripts/ifcfg-<connectionName>` to the file `/etc/resolv.conf`.

1.2 Understanding Routing

Let us consider the following network:

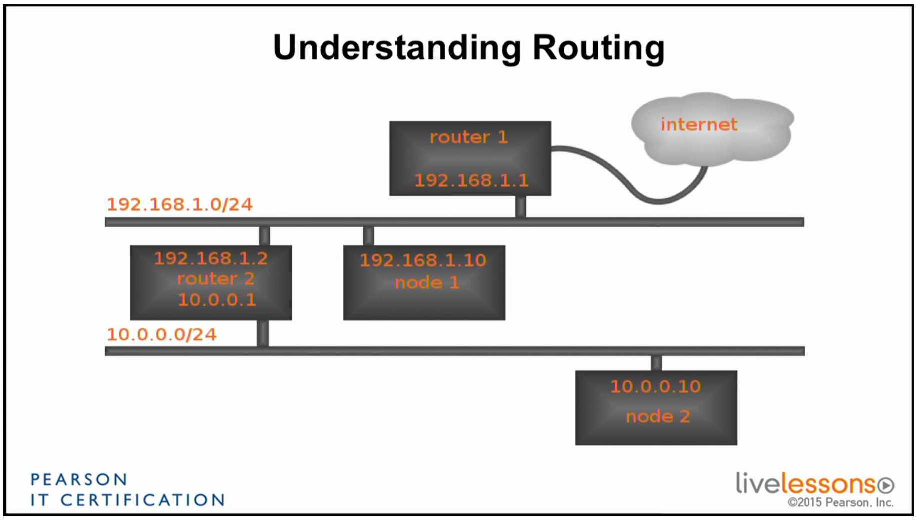


Figure 1.1: Sample Network

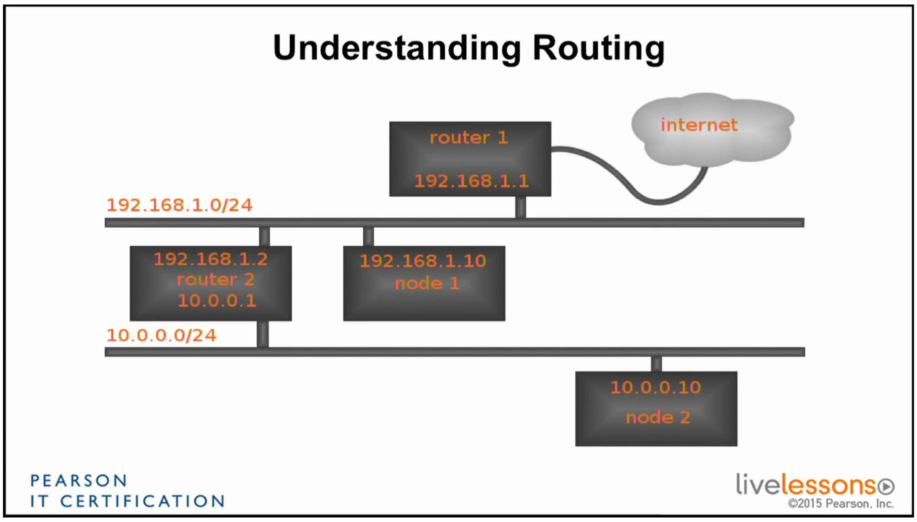
Here, we see two different networks - the `10.0.0.0/24` network connected to the inner `192.168.1.0/24` network via *router 2* (`10.0.0.1`), which in turn connects to the internet via the edge router with IP `192.168.1.1` - *router 1*.

For any packet headed to the internet on network 2, i.e., any packet originating from *node 2*, the default gateway will have to be *router 2* (`10.0.0.1`). This gets the packet on to the `192.168.1.0/24` network, where the default gateway is *router 1* (`192.168.1.1`), which passes it on to the internet.

However, when the packets originate from node 1 (`192.168.1.10`), there are two possible routes - if the packet is destined for the `10.0.0.0/24` network, then the gateway should be *router 2* (`192.168.1.2`). But if the packet is for any other network, then the default gateway of *router 1* (`192.168.1.1`) should be used. Thus, a static route should be defined on node 1 for the `10.0.0.0/24` network.

1.3 Setting up Static Routing

The most convenient way to set up static routes is to use `nmtui`. Let's assume we're setting up static routing for node 2 in our last example.



Understanding Routing

Figure 1.2: Network Diagram

We need to edit the existing connection to include the new static route. For this, we select the options: `Edit a Connection` → Select the connection to use → `Edit...` → `Routing` section → `Edit...` → `Add...` → Type the address of the network for which the static route will be defined in `Destination/Prefix` (with the Network ID and prefix, like, `10.0.0.0/24`) → Add the IP address of the router that leads to the network in the `Next Hop` section (`192.168.1.2` in our case).

The **metric** of the connection is how a router chooses which route to take when there are multiple routes available to another network. Thus, it's only useful when there are multiple routes available for the same network, and is irrelevant to us right now. We now choose `<Ok>` → `<Ok>` → `<Quit>`.

Note however, that the new route won't be added to the network configuration till either the connection is *refreshed* (by reactivating the connection) or the NetworkManager service is restarted. We could do this by `nmtui` → `Activate a Connection` → Select the connection which we edited → `Activate`. Now the output of `ip route show` will show the static route as well.

If the interface name was *ens33*, The `/etc/sysconfig/network-scripts` directory now has a new file called : `route-ens33` with the following contents:

```
ADDRESS0=10.0.0.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.1.2
```

Note that the nmtui utility has translated the `/24` prefix from the **CIDR** (Classless Inter-Domain Routing) notation `10.0.0.0/24` to the standard Network IP and Network Masks, where `/24` translates to the network mask of `255.255.255.0`.

1.4 Understanding Network Bridges

A network bridge is a device that connects two or more networks to form one extended network. For example, an Ethernet bridge connects two or more LANs to create a unified, extended LAN. Virtual bridges are special purpose network interfaces used in virtualized environments.

Let us consider that the physical host has a NIC called `eno1`. The entire virtualized network in the diagram then has to communicate with any external networks via this interface. However, they can't all just send their packets to the driver of the NIC. Thus, they need a virtual bridge `virbr0`. There can be multiple virtual bridges too.

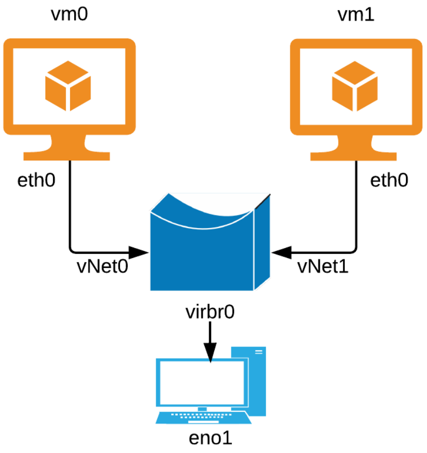


Figure 1.3: A virtualized network

The virtual bridge acts like a physical switch in the network and merely passes data between the networks. Note that it is incapable of routing decisions. All network traffic - even the traffic that originates from the physical KVM host are handled by it and thus, the virtual bridge decides who can send their packets at a specific moment.

Each of the virtual machines have their own virtualized Ethernet interface called `eth0` which have to be connected to an interface (port) on the virtual bridge. The virtual bridge names them `vNet0` and `vNet1` accordingly.

1.4.1 Working with Network Bridges

1.5 Setting up Network Bridges

1.6 Understanding Network Bonds and Teams

1.7 Configuring Network Teams

1.8 Configuring IPv6