Chapter 35 Network Devices and Configuration - Notes

35.2 Introduction

Network devices such as Ethernet, wireless connections require careful configuration, especially when there are multiple devices of same type. Consistent, persistent device naming can become tricky in such circumstances. Recent adoption of new schemes -> naming more predictable. Number of important utilities used to bring devices up/down, configure properties, establish routes etc. System administrators must become adept at their use.

35.3 Learning Objectives:

- Identify network devices and understand how the operating system names them and binds them to specific duties.
- Use the **ip** utility to display and control devices, routing, policy-based routing, and tunneling.
- Use the older **ifconfig** to configure, control, and query network interface parameters from either the command line or from system configuration scripts.
- Understand the Predictable Network Interface Device Names scheme.
- Know the main network configuration files in `/etc`.
- Use **NetworkManager** (**nmtui** and **nmcli**) to configure network interfaces in a distribution-independent manner.
- Know how to set default routes and static routes.
- Configure name resolution as well as run diagnostic utilities.

35.4 Network Devices

Network devices *not* associated with **special device files** (also known as **device nodes**), unlike block/character devices. Know by their names rather than having associated entries in `/dev` directory.

Names consist of type identifier followed by number:

- `eth0`, `eth1`, `eno1`, `eno2`, etc. for Ethernet devices
- `wlan0`, `wlan1`, `wlan2`, `wlp3s0`, `wlp3s2`, etc. for wireless devices
- `br0`, `br1`, `br2`, etc. for bridge interfaces
- `vmnet0`, `vmnet1`, `vmnet2`, etc. for virtual devices for communicating with virtual clients

Historically, multiple virtual devices could be associated with single physical devices.

35.5 ip

ip: the command line utility used to configure, control, query interface parameters and control devices, routing, etc. Preferred to the venerable **ifconfig** discussed next, since more versatile + more efficient because it uses **netlink** sockets rather than **ioctl** system calls.

ip can be used for wide variety of tasks. Can be used to configure, control, query devices and interface parameters. Also manipulate routing, policy-based routing, tunneling.

Basic syntax:

```
ip [ OPTIONS ] OBJECT [ COMMAND | help ]
ip [ -force ] -batch filename
```

where second form can read commands from designated file.

ip a multiplex utility. `OBJECT` argument describes what kind of action going to be performed. Possible `COMMANDS` depend on which `OBJECT` selected.

Some of the main values of `OBJECT`:

Main ip `object`s

`OBJECT`	Function
`address`	**IPv4** or **IPv6** protocol device address
`link`	Network Devices
`maddress`	Multicast Address
`monitor`	Watch for netlink messages
`route`	Routing table entry
`rule`	Rule in the routing policy database
`tunnel`	Tunnel over IP

36.6 Examples of Using ip

ip utility can be used in many ways:

- Show information for all network interfaces:

```
$ ip link show
```

- Show information for `eth0` network interface, including statistics:

```
$ ip -s link show eth0
```

- Set IP address for `eth0`:

```
$ sudo ip addr add 192.168.1.7 dev eth0
```

- Bring `eth0` down:

```
$ sudo ip link set eth0 down
```

- Set **MTU** to 1480 bytes for `eth0`:

```
$ sudo ip link set eth0 mtu 1480
```

- Set netw orking route:

```
$ sudo ip route add 172.16.1.0/24 via 192.168.1.5
```

ipubuntu

35.7 ifconfig

ifconfig: system administration utility long found in UNIX-like operating systems, used to configure, control, query netw ork interface parameters from either command line or from system configuration scripts. Superseded by **ip**, some Linux distributions no longer install by default. Some usage examples:

- Display some information about all interfaces:

```
$ ifconfig
```

- Display information about only `eth0`:

```
$ ifconfig eth0
```

- Set IP address to `192.168.1.50` on interface `eth0`:

```
$ sudo ifconfig eth0 192.168.1.50
```

- Set the *netmask* to 24-bit:

```
$ sudo ifconfig eth0 netmask 255.255.255.0
```

- Bring interface `eth0` up:

```
$ sudo ifconfig eth0 up
```

- Bring interface `eth0` dow n:

```
$ sudo ifconfig eth0 down
```

- Set the **MTU** (**M**aximum **T**ransfer **U**nit)

```
$ sudo ifconfig eth0 mtu 1480
```

ifconfigrhel7

35.8 Problems with Network Device Names

Classic device naming conventions described earlier encountered difficulties, especially when multiple interfaces of same type present. Eg., if two network cards, one named `eth0` and other `eth1`, but which physical device should be associated with each name?

Simplest method: have first device found be `eth0`, second `eth1` etc. Unfortunately, probing for devices not deterministic for modern systems, and devices may be located/plugged in unpredictable order. Thus, one might wind up with Internet interface swapped with local interface. Even if no change in hardware, order in which interfaces located known to vary with kernel version and configuration.

Many system administrators solved this problem in a simple manner: by hardcoding associations between hardware (MAC) addresses and device names in system configuration files and startup scripts. While this method worked for years, required manual tuning + had other problems, eg., when MAC addresses not fixed. Can happen in both embedded and virtualized systems.

35.9 Predictable Network Interface Device Names

Predictable Network Interface Device Names (**PNIDN**): strongly correlated with use of **udev** and integration with **systemd**. 5 types of names that devices can be given:

1. Incorporating Firmware or BIOS provided index numbers for on-board devices:

   Example: `eno1`

2. Incorporating **Firmware** or **BIOS** provided **PCI Express** hotplug slow index numbers:

   Example: `ens1`

3. Incorporating physical and/or geographical location of the hardware connection:

   Example: `enp2s0`

4. Incorporating the `MAC` address:

   Example: `enx7837d1ea46da`

5. Using the old classic method:

   Example: `eth0`

35.10 Examples of the New Naming Scheme

For example, on machine with two onboard PCI network interfaces that would have been `eth0` and `eth1`:

```
$ ip link show | grep enp
2: enp4s2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo-fast state DOWN mode DEFAULT qlen 1000
3: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo-fast state UP mode DEFAULT qlen 1000
```

These names are correlated with physical locations of the hardware on the PCI system.

Likewise, for wireless device that previously would have been simply named `wlan0`:

```
17:/home/coop>ip link show | grep w1
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORMANT qlen 1000
```

See same pattern. Easy to turn off new scheme, go back to classic names. Left as research project. In following, will mostly use classic names for definiteness + simplicity.

35.11 NIC Configuration Files

While network interfaces can be configured on the fly using either **ip** or **ifconfig** utilities, settings not persistent. Thus, number of Linux distribution-dependent files store persistent network interface and device configuration information.

Each distribution has own set of files and/or directories. Depending on version, might be:

Red Hat

```
/etc/sysconfig/network
/etc/sysconfig/network-scripts/ifcfg-ethX
/etc/sysconfig/network-scripts/ifcfg-ethX:Y
/etc/sysconfig/network-scripts/route-ethX
```

Debian

```
/etc/network/interfaces
```

SUSE

```
/etc/sysconfig/network
```

When using **systemd**, preferable to use Network Manager, rather than try to configure underlying text files. In fact, in new Linux distributions, many of these files non-existent, empty, or much smaller, there only for backward compatibility reasons.

35.12 Network Manager

Once upon a time, network connections almost all wired (Ethernet), did not change unless there was significant change to either the hardware, software, or network configuration. During system boot, files in `/etc` consulted to establish all device configurations.

However, modern systems often have dynamic configurations:

- Networks may change as a device is moved from place to place
- Wireless devices may have a large choice of networks to hook into
- Devices may change as hardware such as wireless devices are plugged in or turned on and off

Previously discussed configuration files created to deal with more static situations, very distribution dependent.

Network Manager still uses configuration files, but administrator can avoid directly manipulating them. Its use hopefully almost the same on different systems.

35.13 Network Manager Interfaces

GUI (Graphical User Interface)

If using your laptop in a hotel room or a coffee shop, probably going to use whatever graphical interface your Linux distribution's desktop offers. Can use this to select between different networks, configure security/passwords, turn devices off/on etc.

nmtui

If making configuration change on system that is likely to last for a while, likely to use **nmtui** as it has almost no learning curve + will edit underlying configuration files for you.

nmcli

If need to run scripts that change network configuration, will want to use **nmcli**. Of, if command line junkie, may want to use this instead of **nmtui**.

If GUI properly done, should be able to accomplish any task using any of these three methods. However, will focus on **nmtui** and **nmcli** because essentially distribution independent + hide any differences in underlying configuration files.

35.14 nmtui

nmtui straightforward to use. Can navigate with either arrow keys or tab key.

Besides activating/editing connections, also set system hostname. However, some operations, such as this, cannot be done by normal users, will be prompted for root password to go forward.

nmtui-main **nmtui Main Screen**

nmtui-edit **nmtui Edit Screen**

35.15 nmtui Wireless Configuration

nmtui-config **nmtui Wireless Configuration**

35.16 nmcli

nmcli: command line interface to **Network Manager**. Can issue direct commands, but also has interactive mode.

For many details/examples, can visit Networking/CLI Fedora wiki webpage or can type:

```
$ man nmcli-examples
```

Will explore use of **nmcli** in lab exercises.

nmcli

35.17 Routing

Routing: process of selecting paths in network along which to send network traffic.

Routing table: list of routes to other networks managed by system. Defines paths to all networks and hosts, sending remote traffic to routers.

To see the current routing table, can use **route** or **ip**:

```
$ route -n
$ ip route
```

routeubuntu

35.18 Default Route

Default route: the way packets are send when there is no other match in routing table for reaching specified network.

Can be obtained dynamically using DHCP. However, can also be manually configured (static). With **nmcli** can be done via:

```
$ sudo nmcli con mod virbr0 ipv4.routes 192.168.10.0/24 \
             +ipv4.gateway 192.168.122.0
$ sudo nmcli con up virbr0
```

or can modify configuration files directly. On Red Hat-based systems, can modify `/etc/sysconfig/network` putting in the line:

```
GATEWAY=x.x.x.x
```

or alternatively in `/etc/sysconfig/network-scripts/ifcfg-ethX` on device-specific basis in configuration file for individual NIC. On Debian-based systems, equivalent is putting:

```
gateway=x.x.x.x
```

in `etc/network/interfaces`.

On either system can set default gateway at runtime with:

```
$ sudo route add default gw 192.168.1.10 enp2s0
$ route
Kernel IP routing table
Destination   Gateway      Genmask       Flags Metric Ref Use Iface
default       192.168.1.10 0.0.0.0       UG    0      0     0 enp2s0
default       192.168.1.1  0.0.0.0       UG    1024   0     0 enp2s0
172.16.132.0  0.0.0.0      255.255.255.0 U     0      0     0 vmnet1
192.168.1.0   0.0.0.0      255.255.255.0 U     0      0     0 enp2s0
192.168.113.0 0.0.0.0      255.255.255.0 U     0      0     0 vmnet8
```

Note: this might wipe out your network connection! Can restore either by resetting network, or in above example by doing:

```
$ sudo route add default gw 192.168.1.1 enp2s0
```

These changes not persistent, will not survive system restart.

35.19 Static Routes

Static routes: used to control packet flow when there is more than one router or route. Defined for each interface and can be either persistent or non-persistent.

When system can access more than one router, or perhaps there are multiple interfaces, useful to selectively control which packets go to which router.

Either **route** or **ip** command can be used to set a non-persistent route as in:

```
$ sudo ip route add 10.5.0.0/16 via 192.168.1.100
$ route
Destination  Gateway      Genmask        Flags Metric Ref Use Iface
default      192.168.1.1  0.0.0.0        UG    0      0   0   eth0
10.5.0.0     quad         255.255.0.0    UG    0      0   0   eth0
192.168.1.0  *            255.255.255.0  U     1      0   0   eth0
```

On Red Hat-bases system, persistent route can be set up editing `/etc/sysconfig/network-scripts/route-ethX` as shown by:

```
$ cat /etc/sysconfig/network-scripts/route-eth0
10.5.0.0/16 via 172.17.9.1
```

On Debian-based systems, need to add lines to `/etc/network/interfaces` such as:

```
iface eth1 inet dhcp
    post-up route add -host 10.1.2.51 eth1
    post-up route add -host 10.1.2.52 eth1
```

On SUSE-based system, need to add or create file such as `/etc/sysconfig/network/ifroute-eth0` with lines like:

```
# Destination Gateway Netmask Interface [Type] [Options]
192.168.1.150 192.168.1.1 255.255.255.255 eth0
10.1.1.150 192.168.233.1.1 eth0
10.1.1.0/24 192.168.1.1 - eth0
```

where each field is separated by tabs.

35.20 Name Resolution

Name Resolution: act of translating hostnames to IP addresses of their hosts. Eg., browser or email client will take `training.linuxfoundation.org` and resolve he name to the IP address of the server (or servers) that serve `training.linuxfoundation.org` in order to transmit to and from that location.

There are two facilities for doing this translation:

- Static name resolution (using `/etc/hosts`).
- Dynamic name resolution (using DNS servers).

There are several command line tools that can be used to resolve IP address of hostname:

```
$ [dig | host | nslookup] linuxfoundation.org
```

- **dig**: generates the most information, has many options

- **host**: more compact
- **nslookup**: older

dig is the newest and the other are sometimes considered deprecated, but the output for host is easiest to read and contains the basic information.

One sometimes also required **reverse resolution**: converint IP address to host name. Try feeding these three utilities a known IP address instead of hostname, and examine output.

35.21 /etc/hosts

`/etc/hosts` holds local database of hostnames and IP addresses. Contains set of records (each taking one line) which map IP addresses with corresponding hostnames and aliases.

Typical `/etc/hosts` file looks like:

```
$ cat /etc/host
127.0.01      localhost localhost.localdomain localhost4 localhost4 localdomain4
::1           localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.1.100 hans hans7 hans64
192.168.1.150 bethe bethe7 bethe64
192.168.1.2   hp-printer
192.168.1.10  test32 test64 oldpc
```

Such static name resolution primarily used for local, small, isolated networks. Generally checked before DNS attempted to reolve address; however, priority can be controlled by `/etc/nsswitch.conf`.

The other host-related files in `/etc` are `/etc/hosts.deny` and `/etc/hosts.allow`. These are self-documenting and their purpose is obvious from their names. The `allow` file searched first, and `deny` file only searched if query is not found there.

`/etc/host.conf` contains general configuration information; rarely used.

35.22 DNS

If name resolution cannot be done locally using `/etc/hosts`, then system will query a DNS (**D**omain **N**ame **S**erver) server.

DNS dynamic and consists of network of servers which client uses to look up names. Service distributed; any one DNS server has only information about its **zone of authority**; however, all of them together can cooperate to resolve any name.

Machine's usage of DNS configured in `/etc/resolv.conf`, which historically has looked like:

```
search example.com aps.org
nameserver 192.168.1.1
nameserver 8.8.8.8
```

which:

- Can specify particular domains to search
- Defines a strict order of nameservers to query

DNS_zone_large **DNS**

35.23 Network Diagnostics

Number of basic network utilities in every system administrator's toolbox, including:

- **ping**

  Send 64-byte test packets to designated network hosts and (if it finds them) tries to report back on the time required to reach it (in milliseconds), any lost packets, and some other parameters. Note that the exact output will vary according to the host being targeted, but you can at least see that the network is working and the host is reachable.

- **traceroute**

  Is used to display a network path to a destination. Shows the routers packets flow through to get to a host, as well as the time it takes for each **hop**

- **mtr**

  Combines functionality of **ping** and **traceroute** and creates a continuously updated display, like **top**

- **dig**

  Is useful for testing DNS functionality. Note that one can also use **host** or **nslookup**, older programs that also try to return DNS information about a host.

Note: some recent distributions (such as RHEL 7) require root privilege (as with **sudo**) in order to run the first three diagnostic utilities.

Examples:

```
$ ping -c 10 linuxfoundation.org
$ traceroute linuxfoundation.org
$ mtr linuxfoundation.org
```

35.24 ping Example

pingrhel7

35.25 traceroute Example

tracerouterhel7

35.26 mtr Example

mtrhel7

##

Back to top

Previous Chapter - Table of Contents - Next Chapter