

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 **Network Manager** (**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. Known 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 objects

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 networking route:

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

```
student@ubuntu: ~  
student@ubuntu:~$ ip -s link show ens33  
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000  
    link/ether 00:0c:29:c8:bf:ca brd ff:ff:ff:ff:ff:ff  
    RX: bytes    packets  errors  dropped overrun mcast  
    1582717      1166      0        0        0        0  
    TX: bytes    packets  errors  dropped carrier collsns  
    59159        746      0        0        0        0  
student@ubuntu:~$ ip addr show  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000  
    link/ether 00:0c:29:c8:bf:ca brd ff:ff:ff:ff:ff:ff  
    inet 172.16.2.128/24 brd 172.16.2.255 scope global dynamic ens33  
        valid_lft 1096sec preferred_lft 1096sec  
    inet6 fe80::bf00:6f80:9502:f589/64 scope link  
        valid_lft forever preferred_lft forever  
student@ubuntu:~$
```

35.7 ifconfig

ifconfig: system administration utility long found in UNIX-like operating systems, used to configure, control, query network 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` down:

```
$ sudo ifconfig eth0 down
```

- Set the **MTU** (Maximum Transfer Unit)

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

```
File Edit View Search Terminal Help
x7:/tmp>/sbin/ifconfig
enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.250 netmask 255.255.255.0 broadcast 192.168.1.255
    ether 54:ee:75:b5:f8:39 txqueuelen 1000 (Ethernet)
    RX packets 251841 bytes 305672016 (291.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 148108 bytes 105257478 (100.3 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0xf1200000-f1220000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 100 bytes 7820 (7.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 100 bytes 7820 (7.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp4s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 7e:6d:db:1e:16:9c txqueuelen 1000 (Ethernet)
    RX packets 140946 bytes 192702080 (183.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 44664 bytes 7435027 (7.0 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

x7:/tmp>
```

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 slot 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 wl
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.

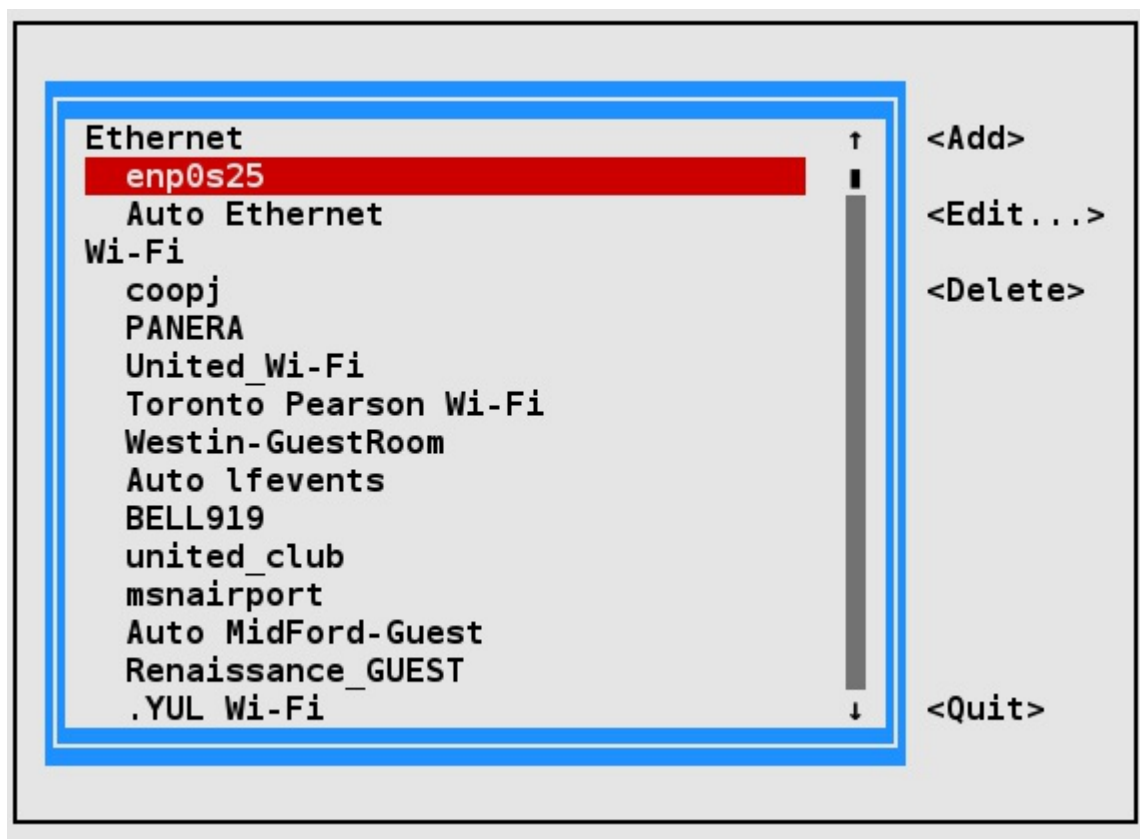
35.14 nmtui

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 Screen



nmtui

Edit Screen

35.15 nmtui Wireless Configuration

Edit Connection

Profile name
Device

= WI-FI

SSID
Mode

<Hide>

Security
Password

<Show>

BSSID
Cloned MAC address

<Show>

MTU

<Show>

= IPv4 CONFIGURATION
= IPv6 CONFIGURATION

<Show>
<Show>

☒ Automatically connect
☒ Available to all users

<Cancel> <OK>

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.


```

student@ubuntu: ~
student@ubuntu:~$ nmcli --help
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }

OPTIONS
  -t[erse]                terse output
  -p[retty]               pretty output
  -m[ode] tabular|multiline output mode
  -c[olors] auto|yes|no   whether to use colors in output
  -f[ields] <field1,field2,...>|all|common specify fields to output
  -e[scape] yes|no        escape columns separators in values
  -a[sk]                  ask for missing parameters
  -s[how-secrets]         allow displaying passwords
  -w[ait] <seconds>       set timeout waiting for finishing operations
  -v[ersion]              show program version
  -h[elp]                 print this help

OBJECT
  g[eneral]              NetworkManager's general status and operations
  n[etworking]           overall networking control
  r[adio]                NetworkManager radio switches
  c[onnection]           NetworkManager's connections
  d[evice]               devices managed by NetworkManager
  a[gent]                NetworkManager secret agent or polkit agent
  m[onitor]              monitor NetworkManager changes

student@ubuntu:~$

```

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

```

```

student@ubuntu: /etc
student@ubuntu:/etc$ route -n
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0        192.168.1.1    0.0.0.0         UG    100    0      0 ens33
169.254.0.0    0.0.0.0        255.255.0.0     U    1000   0      0 ens33
192.168.1.0    0.0.0.0        255.255.255.0   U    100    0      0 ens33
192.168.122.0  0.0.0.0        255.255.255.0   U    0      0      0 virbr0
student@ubuntu:/etc$ ip route
default via 192.168.1.1 dev ens33 proto static metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.1.0/24 dev ens33 proto kernel scope link src 192.168.1.24 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 linkdown

student@ubuntu:/etc$

```

35.18 Default Route

Default route: the way packets are sent 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-based 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 the 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:** convert 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/hosts
127.0.0.1      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 resolve 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.

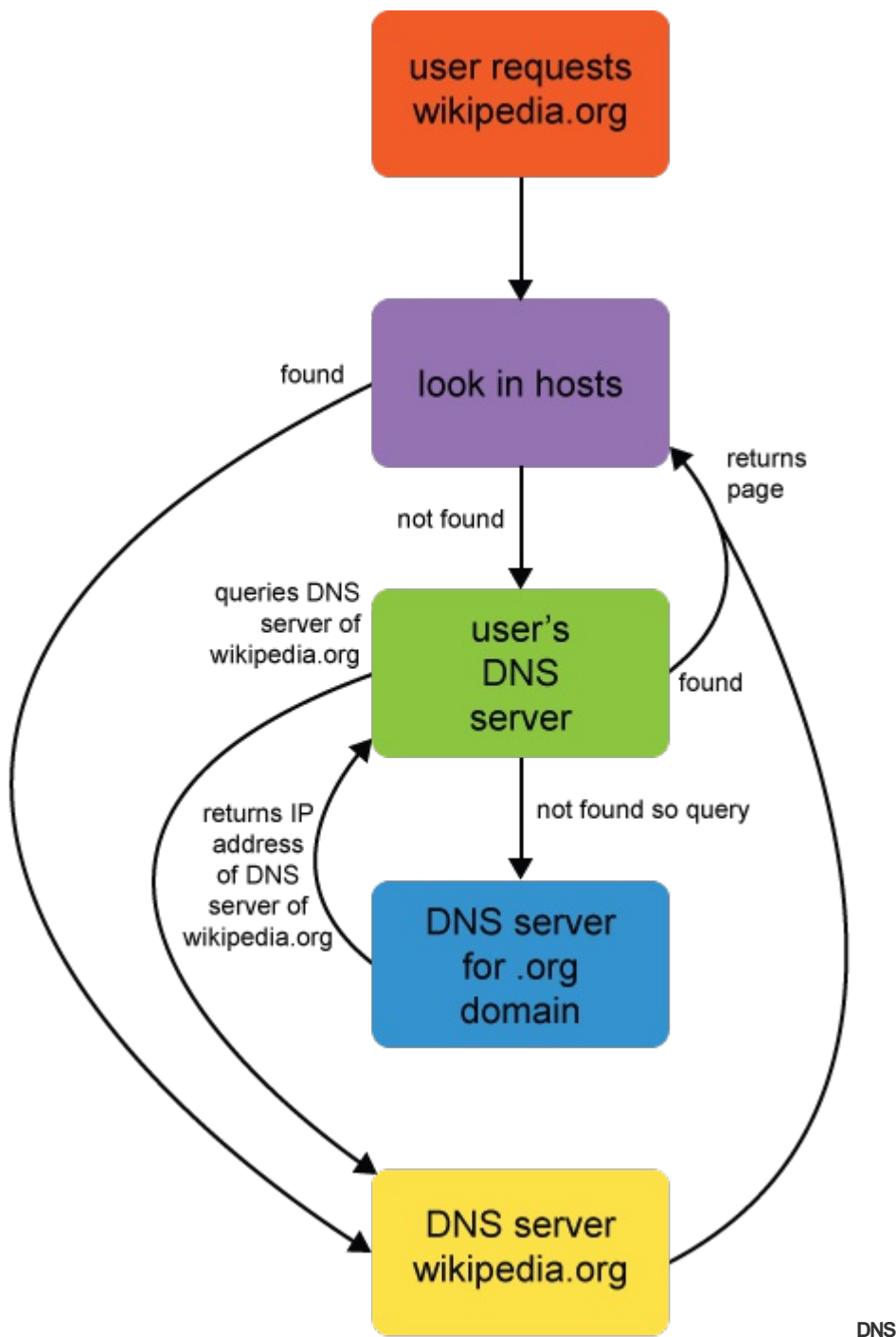
DNS dynamic and consists of network of servers which client uses to look up names. Service distributed; any one DNS server has only information about its **zone of authority**; however, all of them together can cooperate to resolve any name.

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



35.23 Network Diagnostics

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

- **ping**

Sends 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.

- **tracert**

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

```
File Edit View Search Terminal Help
c7:/tmp>ping -c 10 linuxfoundation.com
PING linuxfoundation.com (140.211.169.4) 56(84) bytes of data.
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=1 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=2 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=3 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=4 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=5 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=6 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=7 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=8 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=9 Packet filtered
From corv-car1-gw.nero.net (207.98.64.17) icmp_seq=10 Packet filtered

--- linuxfoundation.com ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9011ms

c7:/tmp>
```

35.25 traceroute Example

```

File Edit View Search Terminal Help
c7:/tmp>tracert www.linuxfoundation.org
tracert to www.linuxfoundation.org (140.211.169.4), 30 hops max, 60 byte packets
 1 gateway (192.168.1.1) 0.386 ms 0.455 ms 0.510 ms
 2 * * *
 3 dtr02ftbgwi-tge-0-6-0-3.ftbg.wi.charter.com (96.34.24.122) 24.020 ms 24.476 ms 24.555 ms
 4 crr01ftbgwi-bue-101.ftbg.wi.charter.com (96.34.30.246) 24.273 ms 24.371 ms 24.623 ms
 5 crr01euclwi-bue-400.eucl.wi.charter.com (96.34.24.224) 26.629 ms 26.521 ms 26.707 ms
 6 crr02sgnwm-tge-0-0-0-4.sgnw.mi.charter.com (96.34.2.153) 32.672 ms 17.489 ms 18.903 ms
 7 bbr02euclwi-bue-5.eucl.wi.charter.com (96.34.0.7) 21.910 ms 22.064 ms 21.976 ms
 8 bbr01chcgil-bue-1.chcg.il.charter.com (96.34.0.9) 47.642 ms 47.696 ms 47.540 ms
 9 prr01chcgil-bue-2.chcg.il.charter.com (96.34.3.9) 50.501 ms 50.580 ms 50.366 ms
10 ge-7-2-3.0.rtr.eqch.net.internet2.edu (64.57.20.49) 50.277 ms 56.285 ms ge-7-2-0.0.rtr.eqch.
net.internet2.edu (162.252.69.153) 40.472 ms
11 ae-1.80.rtsw.chic.net.internet2.edu (64.57.20.150) 39.146 ms 39.147 ms 39.155 ms
12 ae-0.80.rtsw.kans.net.internet2.edu (64.57.20.148) 50.416 ms 48.018 ms 52.376 ms
13 ae-0.80.rtsw.salt.net.internet2.edu (64.57.20.146) 81.953 ms 79.413 ms 84.802 ms
14 ae-2.80.rtsw.losa.net.internet2.edu (64.57.20.144) 86.829 ms 86.982 ms 84.276 ms
15 nero-cps1.ptck-core1-gw.oregon-gigapop.net (198.32.165.198) 93.184 ms 93.248 ms 94.575 ms
16 ptck-p2-gw.nero.net (207.98.64.170) 96.579 ms ptck-p1-gw.nero.net (207.98.64.168) 97.037 ms
ptck-p2-gw.nero.net (207.98.64.170) 95.485 ms
17 corv-p2-gw.nero.net (207.98.64.27) 97.880 ms corv-p1-gw.nero.net (207.98.64.25) 98.413 ms co
rv-p2-gw.nero.net (207.98.64.27) 102.088 ms
18 corv-car1-gw.nero.net (207.98.64.19) 98.220 ms 95.977 ms corv-car1-gw.nero.net (207.98.64.17
) 98.659 ms
19 corv-car1-gw.nero.net (207.98.64.17) 95.829 ms !X 93.986 ms !X *
c7:/tmp>

```

35.26 mtr Example

```

File Edit View Search Terminal Help
My traceroute [v0.85]
c7 (0.0.0.0) Thu May 25 09:51:00 2017
Unable to allocate IPv6 socket for nameserver communication: Address family not supported by pro
tocol
Packets
Host Loss% Snt Last Avg Best Wrst StDev
0.0% 16 0.3 0.3 0.2 0.3 0.0
 2. ???
 3. dtr02ftbgwi-tge-0-6-0-3.ftbg.wi.charter.com 80.0% 16 11.9 9.4 7.9 11.9 2.1
 4. crr01ftbgwi-bue-101.ftbg.wi.charter.com 80.0% 16 8.2 8.0 7.0 8.7 0.7
 5. crr01euclwi-bue-400.eucl.wi.charter.com 80.0% 16 15.0 14.0 13.3 15.0 0.7
 6. crr02sgnwm-tge-0-0-0-4.sgnw.mi.charter.com 86.7% 16 21.7 20.2 18.7 21.7 2.0
 7. bbr02euclwi-bue-5.eucl.wi.charter.com 86.7% 16 25.6 22.7 19.7 25.6 4.1
 8. bbr01chcgil-bue-1.chcg.il.charter.com 86.7% 16 37.7 40.9 37.7 44.2 4.5
 9. prr01chcgil-bue-2.chcg.il.charter.com 81.8% 12 37.0 36.3 35.7 37.0 0.0
10. xe-8-1-2.0.rtr.eqch.net.internet2.edu 81.8% 12 35.4 35.9 35.4 36.3 0.0
11. ae-1.80.rtsw.chic.net.internet2.edu 81.8% 12 37.3 36.5 35.7 37.3 1.0
12. et-3-1-0.4072.rtsw.kans.net.internet2.edu 81.8% 12 47.8 47.8 47.8 47.8 0.0
13. ae-0.80.rtsw.salt.net.internet2.edu 81.8% 12 79.4 79.9 79.4 80.3 0.0
14. ae-2.80.rtsw.losa.net.internet2.edu 81.8% 12 74.0 75.1 74.0 76.3 1.4
15. nero-cps1.ptck-core1-gw.oregon-gigapop.net 81.8% 12 97.5 94.2 90.9 97.5 4.7
16. ptck-p1-gw.nero.net 81.8% 12 93.7 93.9 93.7 94.2 0.0
17. corv-p1-gw.nero.net 80.0% 11 95.0 97.7 95.0 100.4 3.7
18. corv-car1-gw.nero.net 80.0% 11 94.8 94.7 94.7 94.8 0.0
19. ???

```

##

[Back to top](#)

[Previous Chapter](#) - [Table of Contents](#) - [Next Chapter](#)