Linux Networking Basics

Network Interfaces

List available ethernet interfaces

- List available ethernet interfaces
  ```
  $ lspci | egrep -i ethernet
  ```
- List available network card interfaces with detail information like *logical name* and *capabilities*.
  ```
  # lshw -class network
  ```
- List available interfaces with layer 3 information, packet and byte stats.
  ```
  $ ifconfig -a
  ```
  ifconfig without any option shows only active interfaces.
- using *ip* command.
  ```
  $ ip link show
  ```

Using *ethtool* to retrieve information about an ethernet interface

- Get driver information.
  ```
  $ ethtool -i eth0
  ```
- Get current settings like *speed* , *duplex*, *auto negotiation*, *link* and *port*.
  ```
  $ ethtool eth0
  ```
- Get statistics.
  ```
  $ ethtool -S eth0
  ```
-

- Make LED on the NIC blink.

```
$ ethtool -p eth0 <time in seconds>
```

Using *ethtool* to set ethernet interface settings

- Disable auto negotiation and set interface speed and duplex. speed value is in mbit/s.

```
# ethtool -s eth0 speed 10 duplex full autoneg off
```

- View original link speed.

```
# ethtool -d eth0 | grep -i "link speed"
```

- See if link is connected and up.

```
# ethtool eth0 | grep -i "link detected"
```

Get information about the kernel module for a NIC driver

e1000 is an example driver for Intel cards.

- See if the module is loaded.

```
$ lsmod | grep e1000
```

- Get information about the module for a NIC driver.

```
$ modinfo e1000
```

- See if a kernel module file (*.ko*) exists.

```
$ ls -R /lib/modules/`uname -r`/kernel/ | grep e1000
```

IP Configuration

IP configuration with *ifconfig* command

- Bringing down an interface.

  ```
  # ifconfig eth0 down
  ```

- Bringing up an interface.

  ```
  # ifconfig eth0 up
  ```

- Bringing up an interface and set IP address.

  ```
  # ifconfig eth0 192.168.1.15/24 up
  ```

  netmask 255.255.255.0 can be used instead of the prefix if preferred.

IP configuration with *ip* command

ip command is part of *iproute2* package and does a range of jobs like showing and manipulating link, ip address, routing table, and other things.

```
ip <object name> <command> [dev <device>]
```

Object names I cover are *link*, *addr* and *route*. In *show* commands if you don't specify *device* it will query all devices.

- Setting down an interface.

  ```
  # ip link set down dev eth0
  ```

- Setting up an interface.

  ```
  # ip link set up dev eth0
  ```

- Show link information about an interface.

  ```
  $ ip link show dev eth0
  ```

- Show ip address information about an interface.

  ```
  $ ip addr show dev eth0
  ```

- Add ip address to an interface.

  ```
  # ip addr add 192.168.1.110/24 dev eth0
  ```

  255.255.255.0 can be used instead of the prefix if preferred.

- Remove an ip address from an interface.

  ```
  # ip addr del 192.168.1.110/24 dev eth0
  ```

- Remove all ip addresses from an interface.

  `# ip addr flush dev eth0`

Managing routing table

- Get routing table.

  `$ ip route show` or `$ route -n`
- Adding static route.

  `# ip route add 172.16.0.0/16 via 192.168.1.1`
- Adding static default route.

  `# ip route add default via 192.168.1.1`

  0.0.0.0/0 can be used in place of *default*.
- Removing static route.

  `# ip route del 172.16.0.0/16`

Resolver

Linux get the name servers' ip address from */etc/resolv.conf* file. syntax is very easy. You add a line for each name server like this.

```
nameserver 192.168.1.1
nameserver 8.8.4.4
```

/etc/resolve.conf is very volatile and can be overwritten by *network scripts* and other configuration utilities like *dhclient*.

Getting IP from DHCP Server

dhclient is a utility to help you obtain ip address from a dhcp server.

- Getting ip, net mask, name servers and default route with dhclient.

  ```
  # dhclient -v4 eth0
  ```

- Release the current lease and stop the running DHCP client.

  ```
  # dhclient -r eth0
  ```

Network Scripts

All the things done in the previous section is in the *memory*. all ip configuration will vanish the moment you reboot your machine. So major linux distributions offer what is called *network scripts* which is used by network service to configure interfaces.

Network scripts are among those topics which is different from distro to distro. I will cover *Debian* and *RedHat* network scripts for all other distros are using either on of them.

Debian Network Scripts

Debian reads the file **/etc/network/interfaces** for instructions how to configure network interfaces. Use your preferred editor to modify the contents.

```
auto eth0
iface eth0 inet static
address 192.168.1.95/24
#netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 192.168.1.1 8.8.4.4

auto eth1
iface eth1 inet dhcp
up ip route add 20.0.0.0/8 via 172.16.1.1
down ip route del 20.0.0.0/8
```

- All network interfaces are configured in on single file.
- **auto eth0**: Makes networking service to configure this interface at boot.
- **iface eth0 inet static**: Configure interface with static ipv4. (inter is used for configuring ipv6)
- **iface eth0 inet dhcp**: Configure interface with dynamic ipv4 using dhcp. in this case address, gateway and dns-nameservers are not needed.
- **address**: IP address/prefix
- **netmask**: If you prefer net mask over prefix, you may use it instead.
- **gateway**: *next hop* for the default route.
- **up**: Run command right after interface is configured.
- **down**: Run command right after interface is down.

When an interface is configured there are two commands that can be used to either bring up or down a configured interface. **ifup** and **ifdown**.

- Bring up a previously configured interface.
  ```
  # ifup eth0
  ```
- Bring down a previously configured interface.
  ```
  # ifdown eth0
  ```

RedHat Network Scripts

RedHat reads the file **/etc/sysconfig/network-scripts/ifcfg-<devicename>** files for instructions how to configure network interfaces. each file contain configuration for one interface.

The second part of the file name which corresponds device name is really not important. It could be anything because redhet network scripts load all **ifcfg-*** files and looks inside for a directive named **DEVICE=** and **HWADDR=** to match the exact device to configure.

```
DEVICE=eth0
HWADDR=08:00:27:36:38:94
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.1.95
PREFIX=24
#NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS1=192.168.1.1
DNS2=8.8.4.4
```

- **DEVICE and HWADDR**: Device name and mac address of the interface. Networking service checks these two, to select which interface to configure.
- **ONBOOT**: Makes networking service to configure this interface at boot.
- **BOOTPROTO**: *none* for static ip config or *dhcp* for using dhcp.
- **IPADDR**: IP address
- **PREFIX**: Network Prefix
- **GATEWAY**: *next hop* for the default route.
- **DNS1 and DNS2** : Nameservers are configured from DNS1 to DNSn.

When an interface is configured **ifup** and **ifdown** can be used to bring up pr down the interface.

Hostname

- Get system hostname
  `$ hostname`
  `$ uname -n`
  `$ cat /proc/sys/kernel/hostname`
  `$ sysctl kernel.hostname`

- Change hostname immediately

  `# hostname foo`

  `# sysctl kernel.hostname=foo`
- Change hostname permanently on **Debian**

  `# echo foo > /etc/hostname` then `# hostname -F /etc/hostname`
- Change hostname permanently on **RedHat**

  Edit **/etc/sysconfig/network** file. and change **HOSTNAME=**. then run the following command. then `# hostname -F /etc/hostname`
- Changing bash prompt to show FQDN
  - \H in PS1 environment is replaced by FQDN
  - \h in PS1 environment is replaced by hostname
  - example: `$ export PS1="\[\u@\H: \w\]\$ "_`
  - You may also edit your .bashrc file to permanently change PS1

ARP cache

arp command manipulates or displays the kernel's IPv4 network arp cache. It can add entries to the table, delete one, or display the current content.

- Get current content of the table.

  `$ arp -n`

  `$ ip neighbor show`

  `-n` is used to prevent reverse lookup for ip addresses in the table.
- Remove an entry

  `# arp -d 192.168.1.1`
- Remove all entries from an interface.

  `# ip -s neigh flush dev eth0`

  all can be used instead of ethic

- Adding static arp entry.

  ```
  # arp -s 192.168.0.10 02:AC:99:00:93:BF
  ```

- Adding static arp entries from a file.

  ```
  # arp -f file
  ```

- Disabling arp on an interface.

  ```
  # ip link set arp off dev eth0
  ```

ICMP

ping

ping is used to send icmp packets to a host and calculate time elapse for the round trip.

- ping a host.

  ```
  $ ping 192.168.1.1
  ```

  ping continues until you hit ^c

- ping a host 4 times.

  ```
  $ ping 192.168.1.1 -c 4
  ```

- ping flood. when you use `-f` option with ping, it means ping will send as much he requests as possible to the host. For every echo request sent a **dot** is printed and for every echo reply received a **backspace** is printed. So accumulating dots represents drops.

  For a stable link you may see a only a dot.

  ```
  # ping 192.168.1.1 -f
  ```

- Other options:

  - `-n` Disable reverse lookup.
  - `-i` Interval between echo requests.
  - `-t` Send echo request with modified TTL
  - `-w` Timeout in seconds.

    - `-s` Packet size. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

traceroute

traceroute is used to discover the hops between a host and a target. it utilizes the imp **TTL** to discover hops.

- General use.
  ```
  $ traceroute -n 8.8.4.4
  ```
  `-n` is used to prevent reverse lookup for ip addresses of the hops.
- *traceroute* with different methods.
    - UDP
      ```
      $ traceroute -n -M default
      ```
    - TCP
      ```
      $ traceroute -n -M tcp
      ```
    - ICMP
      ```
      $ traceroute -n -M icmp
      ```
- Other options:
    - `-4` Force ipv4.
    - `-m` Max TTL. Default is 30.
    - `-s` Choose alternate source address.
    - `-q` Sets the number of probe packets per hop. The default is 3.

mtr

mtr combines the functionality of the *traceroute* and *ping*. mtr first discovers hops then start to ping each hop separately.

It is the best tool to discover **packet loss**.

- General use.

  ```
  $ mtr -n 8.8.4.4
  ```

  `-n` is used to prevent reverse lookup for ip addresses of the hops.
- Other options:
  - `-4` Force ipv4.
  - `-c` Number of pings sent to each hop.
  - `-s` Packet size.
  - `-u` Use UDP instead of icmp.
  - `-T` Use TCP instead of icmp.

Basic Monitoring

There are many tools to monitor network traffic like *mrtg* and *cacti*, but I want to introduce some command line tools.

bmon for monitoring traffic

bandwidth monitor and rate estimator. it has the ability to print ASCII graphs.

```
$ bmon
```

iftop for monitoring traffic

iftop is like bmon but it shows RX/TX traffic per host. when you have a linux internet gateway this comes very handy.

```
# iftop -ni eth0
```

`-n` Disables reverse lookup and `-i` specifies the interface.

netstat

netstat is tool that fits both in *monitoring* and *security* category. *netstat* is used to provide troubleshooting and insights into protocols, ports and connections.

- Get IP socket status.
  ```
  $ netstat --inet -n
  ```
- Get IP socket status details.
  ```
  # netstat --inet -p -e
  ```
  - `-p` include process name and PID
  - `-e` Additional information like user
  - `-p` include process name and PID
- Get open TCP ports (Listening ports).
  ```
  # netstat -lnpt
  ```
  - `-t` TCP use `-u` for UDP
  - `-l` Listening
- Get top 10 hosts with most established connections to you with number of connections per hosts. It is best way to detect DOS attacks.
  ```
  # netstat -ant | grep ESTAB | awk {'print $5'} | cut -d: -f1 | sort -n | uniq -c | sort -nr | head -n1
  ```