Linux Networking Overview – Thales November 2018

The following applies to most if not all linux distribution, but the commands and configuration files may differ.

This session concentrates on CentOS 7.

Topics.

1. Basic Concepts.
2. Connectivity.
3. Configuration files
4. IP-Binding
5. Bonding
6. VLANS
7. Some commands
8. Kernel Parameters

1. Basic concepts.

1.1 NIC: Network interface card, the device that offers physical connectivity to a network. The device has a hardware (MAC) address. This mac address is used by all physical devices connected to the same network segment to communicate. The hardware address is at level 2 of the tcp/ip stack: datalink layer. Example: `00:0c:29:46:d2:94`

1.2 IP-address: A 32 bits or 128 bits numerical address that helps a machine connect to other machines on the same physical network or other physical network. The ip-address is at level three of the tcp/ip stack: internetwork layer. Example: `192.168.4.1`

1.3 Netmask: A mask used to divide an IP address into subnets and specify the network's available hosts. Example: `255.255.255.0`

1.4 Broadcast address: Network address at which all devices connected to a multiple-access communications network are enabled to receive datagrams. Example: `192.168.4.255`

1.5 IP-binding: Configure multiple ip-addresses on the same nic.

1.6 Bonding: Group multiple network interface cards to form a channel or link aggregate to increase the bandwidth and redundancy.

1.7 Vlans: Logical broadcast domains which can span local area networks.

1.8 Gateway: The ip-address of a machine in the subnet that offers access to other subnets. These subnets may be on the same physical segment or on another physical segment.

1.9 Nameserver: System that translates domain-names and host-names into ip addresses.

2. Connectivity

2.1 The ip command

The ip command can be used to list your interfaces, ip addresses, routes, etc.

examples:

```
[root@linux1 ~]# ip address show
[root@linux1 ~]# ip link show
[root@linux1 ~]# ip route show
[root@linux1 ~]# ip neigh show
```

If you are used to using the old set of commands like ifconfig and netstat, you can install '*net-tools*', and as you do that you might just as well install '*bind-utils*', for nslookup and other deprecated commands.

The following table lists the deprecated commands and the replacements.

deprecated	new
arp	ip n (ip neighbor)
ifconfig	ip a (ip addr), ip link, ip -s (ip -stats)
iptunnel	ip tunnel
iwconfig	iw
nameif	ip link, ifrename
netstat	Ss, ip route (for netstat-r), ip -s link (for netstat -i), ip maddr (for netstat-g)
route	ip r (ip route)

If you are accustomed to commands like ifconfig and netstat, you can still use them if you install the tools mentioned.

To view the current ethernet ports, you could run the command 'lshw'.

```
[root@linux1 ~]# yum install -y lshw
[root@linux1 ~]# lshw -short | grep -i ether
/0/100/16/0          ens192      network        VMXNET3      net Controller
/0/100/17/0          ens224      network        VMXNET3      net Controller
```

To view the current configuration of your network card:

```
[root@linux1 ~]# ifconfig ens192
ens192: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.4.131  netmask 255.255.255.0  broadcast 192.168.4.255
        inet6 fe80::6ea9:de5d:ea0d:4b78  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:46:d2:8a  txqueuelen 1000  (Ethernet)
        RX packets 1157987  bytes 81005190 (77.2 MiB)
        RX errors 0  dropped 489  overruns 0  frame 0
        TX packets 26041  bytes 29592652 (28.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

To view the current routing table:

```
[root@linux1 ~]# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.4.1     0.0.0.0         UG        0 0          0 ens192
192.168.4.0     0.0.0.0         255.255.255.0   U         0 0          0 ens192
```

All this information is stored in Memory. You can view this for example by looking at **/proc/net.** For example, to show the contents of your arp cache, which holds the hardware addresses of the cards recently connected to you can run 'cat arp'.

```
[root@linux1 net]# cd /proc/net
[root@linux1 net]# cat arp
IP address       HW type     Flags       HW address            Mask     Device
192.168.4.227    0x1         0x2         00:0c:29:f7:ef:02     *        ens192
192.168.4.101    0x1         0x2         00:0c:29:f7:ef:02     *        ens192
192.168.4.156    0x1         0x2         b8:27:eb:29:76:f9     *        ens192
192.168.4.1      0x1         0x2         4c:9e:ff:5a:bd:31     *        ens192
192.168.4.15     0x1         0x2         f4:5c:89:8d:d4:47     *        ens192
```

To add a static route:

```
[root@linux1 net]# route add -net 192.16.7.0 gw 192.168.4.133 netmask 255.255.255.0
dev ens192
```

To check whether your machine is forwarding packets to other networks you can check /proc/sys/net/ipv4/

```
[root@linux1 /]# cat /proc/sys/net/ipv4/ip_forward
0
```
(0=not forwarding, 1=forwarding)

You can explore ***/proc*** for yourself. There's a lot more than just networking.

2.2 SELINUX and Firewalld

2.2.1 SELINUX

The current hosts that run CentOS 7 can choose to run firewalld or iptables, for firewalling. You cannot run both at the same time. If your network is setup correctly connectivity wise, but you are unable to perform certain networking tasks, two things may cause these problems: firewall or selinux.

In this session we are not going to go into selinux, other than stating that it enforces mandatory access control. This results in that certain processes can only write to certain directories and connect to certain networking ports, base on a configuration type. If selinux is the problem, you can *disable* it or put it into *permissive* mode.

To check whether selinux is active, run getenforce.

```
[root@linux1 etc]# getenforce
Disabled
```

This will tell you that selinux is either active 'enforcing', not active 'disabled', or reporting-only 'permissive'. This last state will log the errors in the audit.log config file, but it will not deny anything.

To change from disabled to permissive or from disable to enforcing, you will need a reboot. Changing from permissive or enforcing to disabled also needs a reboot. To change from enforcing to permissive or the other way around, you can use the setenforce command.

```
[root@linux1 ~]# getenforce
Permissive
[root@linux1 ~]# setenforce enforcing
[root@linux1 ~]# getenforce
Enforcing
[root@linux1 ~]# setenforce 0
[root@linux1 ~]# getenforce
Permissive
[root@linux1 ~]# setenforce 1
[root@linux1 ~]# getenforce
Enforcing
```

The config file is /etc/sysconfig/selinux.conf.

```
[root@linux1 /]# cat /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     permissive - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of permissive.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are
protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2.2.2 Firewalld

Firewalld works with so-called *zones*.

A zone holds a white-list of hosts, services and ports that it allows to connect to the host. The service that runs is managed by the firewall daemon: firewalld. The default zone is usually the *public* zone. This public zone allows only allows dhcpv6client and ssh.

```
[root@linux1 /]# firewall-cmd --zone=public --list-services
dhcpv6-client ssh
```

To list all available zones:

```
[root@linux1 /]#  firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

To list the active zones, and the interfaces that are connected:

```
[root@linux1 /]# firewall-cmd --get-active-zones
public
  interfaces: ens192 ens224
```

To add a service or other functionality to a zone:

```
[root@linux1 /]# firewall-cmd --zone=public --add-service=http
success

[root@linux1 /]# firewall-cmd --zone=public --add-port=5000/tcp
success
```

To make the changes permanent, add **`--permanent`** to the command.

To stop and disable firewalld:

```
[root@linux1 /]# systemctl stop firewalld
[root@linux1 /]# systemctl disable firewalld
```

3. Configuration files.

Many config files in /etc/sysconfig. The files to configure your network interfaces on CentOS, can be found in /etc/sysconfig/network-scripts.

Each network interface has its own config-file. So if you have ens192 and ens224, you will have to default config-files. One for each card.

3.1 NIC Configuration.

The file to configure your network interface: */etc/sysconfig/network-scripts/ifcfg-<nicname>*.
So, for example:

```
[root@linux1 /]# cat /etc/sysconfig/network-scripts/ifcfg-ens192
TYPE="Ethernet"
BOOTPROTO="none"
NAME="ens192"
DEVICE="ens192"
ONBOOT="yes"
IPADDR="192.168.4.131"
PREFIX="24"
GATEWAY="192.168.4.1"
DNS1="192.168.4.1"
```

BOOTPROTO is usually set to *dhcp* or *none*, but can also be *static* or *bootp*. Static and none mean that you will have to supply the ip information yourself. If it is dhcp or bootp, then the address will be retrieved from a dhcp- or bootpserver. Another important switch is the ONBOOT parameter. If very often defaults to “no”. This means that the ip address will not be configured at boot time. In this file you could also enter e.g. the default gateway and dns nameservers.

Note: if you set the dns server in the ifcfg-ens192 file, it will automatically update the central configuration **/etc/resolv.conf** file with that information. If you do not enter a DNS entry in the ifcfg-ens192 file, you will manually have to update the /etc/resolv.conf file. This resolv.conf file is the central file to determine your dns name server, to the least. Also, if you add multiple different DNS entries to different ifcfg-<nicname> files, all these will be added to the /etc/resolv.conf file.

3.2 Resolve order.

If your host has to resolve an ip address to a hostname or a user id to a user name or various other resolve issues, there is a file to determine the order in which the possibilities to resolve are checked. This file is **/etc/nsswitch.conf**

For the networking part of your host, the **host:** entry is relevant.

grep hosts /etc/nsswitch.conf

```
[root@linux1 /~]# grep hosts /etc/nsswitch.conf
hosts:      files dns
```

If you would want to first resolve using the dns name server, you should change the order into *dns files*.

3.3 DNS name server

The file /etc/resolv.conf, contains at least the name server that should be queried if the hosts needs an ip address or hostname. Also the dns domain name is often added.

As seen in 3.1, if the name server is added in the ifcfg-<nic-name>, then the file will be edited by the network manager.

An example:

```
search example.com local.lan
name server 1.1.1.1
name server 8.8.8.8
name server 9.9.9.9
```

3.4 Changing Network configuration.

To change your configuration you can choose to manually change configs, but that will not update the configuration files, so after a reboot or restart of a particular service, you configuration will be lost. To make things permanent, you will have to change the config files.

After having changed the config files, you can still set things manually, if you like.

4. IP binding.

You can add more than one ip address to a network interface. This can be done on the fly or by creating a separate ifcfg- file.

The reason why you might want to do this is for example in a cluster environment. You may want to be able to (automatically) migrate an ip address to another node in the cluster.

4.1 IP binding on the fly.

To add an ip address to an already existing nic, you can use the ip or ifconfig command.

Example with ifconfig

```
[ root@linux1 /~] # ifconfig ens224:1 192.168.1.10
```

Example with ip

```
[ root@linux1 /~]# ip a add 192.168.10.11 dev ens224
```

You can also do it in a the config file.

Copy the ifcfg-ens224 file to ifcfg-ens224:1
Edit all entries that refer to ens224 to ens224:1
Edit the ip address
Save the file
Restart the service

Example:

```
[root@linux2]# cd /etc/sysconfig/network-scripts
[root@linux2]# cp ifcfg-ens224 ifcfg-ens224:1
[root@linux2]# sed -i s/ens224/ens224:1/ ifcfg-ens224:1
[root@linux2]# sed -i s/192.168.4.132/192.168.4.133/ ifcfg-ens224:1
[root@linux2]# systemctl restart network
```

5. Bonding.

5.1 What is bonding?

Linux allows binding of multiple network interfaces into a single channel/NIC using special kernel module called bonding. The Linux bonding driver provides a method for aggregating multiple network interfaces into a single logical “bonded” interface.

5.2 Modes

Round-robin (balance-rr) or 0

Network packets transmit in sequential order from the first available network interface (NIC) slave towards the last one.This way every Ethernet card used for send and receive network packets.So this mode has load balancing and fault tolerance feature.

Active-backup (active-backup) or 1

In this only one slave Ethernet card in the bond is active.Another slave Ethernet cards only becomes active only when the active slave fails. The single logical bonded interface’s MAC address is externally visible on only one NIC (port) to avoid distortion in the network switch. This mode provides fault tolerance.

XOR (balance-xor) or 2

Network packets transmit based on hash of network packet’s source and destination. Default algorithm considers only MAC addresses (layer2), In this case we can only use one Ethernet card at a time. Newer versions allow selection of additional policies based on IP addresses (layer2+3), In this case we can use multiple Ethernet cards combine to form Bond with help of alias over virtual Bond interface and TCP/UDP port numbers (layer3+4), This used when your application used multiple port to transmit data over Bond channel. This select same NIC slave for destination MAC address, IP address, or IP address and port combination, respectively, This also should have same capability over switch level at same time. This mode provides load balancing and fault tolerance feature.

Broadcast (broadcast) or 3

This transmits network packets over all slave network interfaces. This mode provides fault tolerance feature.

IEEE 802.3ad Dynamic link aggregation (802.3ad, LACP) or 4

This creates aggregation groups that share with same speed and duplex settings and also Utilizes all slave network interfaces in the active aggregator group. This mode is behave like the XOR mode mentioned above and also supports the same balancing policies. Link set up dynamically between two LACP-supporting peers.

Adaptive transmit load balancing (balance-tlb) or 5
This mode does not require any special network-switch support. Outgoing network packet traffic distributed according to the current load on each slave interface. Incoming traffic is received by one currently designated slave network interface. If receiving slave fails, another slave takes over the MAC address of the failed slave.

Adaptive load balancing (balance-alb) or 5
This has balance-tlb mentioned above plus receive load balancing (rlb) for IPV4 traffic only, and not require any special network switch support. Receive load balancing is achieved through ARP negotiation. Bonding driver intercepts ARP Replies sent by local system on their way out and will overwrites source hardware address with unique hardware address of one of slaves interface in the single logical bonded interface such that different network-peers use different MAC addresses for their network packet traffic.

5.3 Setup

To setup Bonding with two slave interfaces in round robin mode:

First Ethernet File — /etc/sysconfig/network-scripts/ifcfg-bond0

```
DEVICE=bond0
NAME=bond0
TYPE=Bond
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.4.132
NETMASK=255.255.255.0
GATEWAY=192.168.4.1
BONDING_MASTER=yes
BONDING_OPT="mode=balance-rr"
```

First Ethernet File — /etc/sysconfig/network-scripts/ifcfg-ens192

```
DEVICE=ens192
ONBOOT=yes
TYPE=Ethernet
BOOTPROTO=none
MASTER=bond0
SLAVE=yes
```

Second Ethernet File — /etc/sysconfig/network-scripts/ifcfg-ens224

```
DEVICE=ens224
BOOTPROTO=none
ONBOOT=yes
TYPE=Ethernet
MASTER=bond0
SLAVE=yes
```

Now we just need to restart network service with below command.

```
#systemctl restart network
```

To check your bond0, you can use /proc

```
# cat /proc/net/bonding/bond0
```

6. VLANS.

6.1 What are vlans?

A VLAN (virtual LAN) abstracts the idea of the local area network (LAN) by providing data-link connectivity for a subnet. One or more network switches may support multiple, independent VLANs, creating Layer 2 (data link) implementations of subnets. A VLAN is associated with a broadcast domain. It is usually composed of one or more Ethernet switches.

6.2 VLANS CentOS

```
~]# modprobe --first-time 8021
 modprobe: ERROR: could not insert '8021q': Module already in kernel
~]# modinfo 8021q
(output skipped)
```

Configure the parent interface in `/etc/sysconfig/network-scripts/ifcfg-ens224`:

```
DEVICE=ens224
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
```

1. Configure the VLAN interface configuration in the `/etc/sysconfig/network-scripts/` directory. The configuration file name should be the parent interface plus a `.` character plus the VLAN ID number. For example, if the VLAN ID is 192, and the parent interface is eth0, then the configuration file name should be `ifcfg-eth0.192`:

```
DEVICE=ens224.10
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.4.131
PREFIX=24
NETWORK=192.168.4.1
VLAN=yes
```

If there is a need to configure a second VLAN, with for example, VLAN ID 20, on the same interface, eth0, add a new file with the name `ens224.20` with the VLAN configuration details.

2. Restart the networking service in order for the changes to take effect. As `root` issue the following command:

```
# systemctl restart network
```

Note: to setup vms with vlans, it may be easier to create a trunk between an ESXi port and the switch port. Then simply connect the vm-interfaces to the vlan network in ESXi.

8. Some commands

If you run CentOS 7 but still like the use the deprecated commands:

```
yum install net-tools
yum install bind-utils
```

ip (link and ip management)

```
ip addr add 192.168.50.5 dev eth1
ip addr show

ip link show
ip link set eth1 down

ip route show

ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0
```

ifconfig (link and ip management)

```
ifconfig ens224 down
ifconfig
```

netstat (statistics)

```
netstat -rn
netstat -an | grep -i listen
```

ss (statistics)

```
ss | head -n 5
ss -lt
ss -pl
```

nslookup (name resolution)

```
nslookup oracle.com
```

dig (name resolution)

```
dig oracl.com
```

nmap (port scan)

```
nmap 192.168.4.1
```

ethtool (query or control network driver and hardware settings)

```
ethtool -P ens192
```

traceroute (check routing path)

```
traceroute oracle.com
```

tcpdump (trace network traffic)

```
tcpdump -i ens192 udp
```

8. Kernel Parameters.

There are many kernel parameters that have to do with security and performance.

To list all settings:

```
# sysctl -a
```

To change parameters: sysctl -w <param>=”value”
or edit /etc/sysctl.conf

For example:

```
# sysctl -w net/ipv4/ip_forward=1
(this will change your machine into a router)

 sysctl net.ipv4.icmp_echo_ignore_all
1

# sysctl -w net.ipv4.icmp_echo_ignore_all=1
(this will stop replying to icpm reply requests)

# sysctl net.ipv4.icmp_echo_ignore_all
1
```

Some important Networking parameters.

Change TCP keepalive parameters

•TCP keepalive is a mechanism for TCP connections that help to determine whether the other end has stopped responding or not.
•TCP will send the keepalive probe contains null data to the network peer several times after a period of idle time. If the peer does not respond, the socket will be closed automatically.
•By default, TCP keepalive process waits for two hours (7200 secs) for socket activity before sending the first keepalive probe, and then resend it every 75 seconds. As long as there is TCP/IP socket communications going on and active, no keepalive packets are needed.

Note: With the following settings, your application will detect dead TCP connections after 120 seconds (60s + 10s + 10s + 10s + 10s + 10s + 10s)

```
net.ipv4.tcp_keepalive_time = 60
net.ipv4.tcp_keepalive_intvl = 10
net.ipv4.tcp_keepalive_probes = 6
```

Increase the maximum connections

The upper limit on how many connections the kernel will accept (default 128):

```
net.core.somaxconn = 1024
```

Warning: Increasing this value may only increase performance on high-loaded servers and may cause as slow processing rate (e.g. a single threaded blocking server) or insufficient number of worker threads/processes [1].

Increasing the size of the receive queue.

The received frames will be stored in this queue after taking them from the ring buffer on the network card.

Increasing this value for high speed cards may help prevent losing packets:

```
net.core.netdev_max_backlog = 100000
net.core.netdev_budget = 50000
net.core.netdev_budget_usecs = 5000
```

Note: In real time application like SIP routers, this option requires a high speed CPU otherwise the data in the queue will be out of date.

Tweak the pending connection handling

`tcp_max_syn_backlog` is the maximum queue length of pending connections 'Waiting Acknowledgment'.

In the event of a synflood DOS attack, this queue can fill up pretty quickly, at which point tcp_syncookies will kick in allowing your system to continue to respond to legitimate traffic, and allowing you to gain access to block malicious IPs.

If the server suffers from overloads at peak times, you may want to increase this value a little bit:

```
net.ipv4.tcp_max_syn_backlog = 30000
```

`tcp_max_tw_buckets` is the maximum number of sockets in 'TIME_WAIT' state.

After reaching this number the system will start destroying the socket that are in this state.

Increase this to prevent simple DOS attacks:

```
net.ipv4.tcp_max_tw_buckets = 2000000
```

TCP SYN cookie protection

Helps protect against SYN flood attacks. Only kicks in when `net.ipv4.tcp_max_syn_backlog` is reached:

```
net.ipv4.tcp_syncookies = 1
```

Enable Ignoring to ICMP Request

To disable ICMP echo 'ping' requests:

```
net.ipv4.icmp_echo_ignore_all = 1
```