LPI 109.1 - Basic network troubleshooting

Curs 2021 - 2022

ASIX M01-ISO 109 Networking Fundamentals

Basic network troubleshooting

Description

Key concepts:

- ❏ Manually configure network interfaces, including viewing and changing the configuration of network interfaces using iproute2.
- ❏ Manually configure routing, including viewing and changing routing tables and setting the default route using iproute2.
- ❏ Debug problems associated with the network configuration.
- ❏ Awareness of legacy net-tools commands.

Commands and files:

- ❏ ip
- ❏ hostname
- ❏ ss
- ❏ ping
- ❏ ping6
- ❏ traceroute
- ❏ traceroute6
- ❏ tracepath
- ❏ tracepath6
- ❏ netcat
- ❏ ifconfig
- ❏ netstat
- ❏ route

Network Tools

When troubleshooting a network, it is useful to gather information about networking services such as open ports, interface statistics, routing tables, and network connections. The following sections will cover common legacy and updated Linux tools available for displaying network status.

netstat Command

The netstat command is used by the system administrator to monitor the traffic on the network, and check connections that are not trustworthy.

- netstat
- netstat -s -r -i
- netstat -t -u -x -n -p -a
- netstat -l

```
$ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
```

```
$ netstat -u
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 mylaptop.edt.org:bootpc _gateway:bootps         ESTABLISHED
```

```
$ netstat -x | head
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    51551    /run/user/1001/systemd/notify
unix  4      [ ]         DGRAM                    16197    /run/systemd/notify
unix  23     [ ]         DGRAM                    16209    /run/systemd/journal/dev-log
unix  16     [ ]         DGRAM                    16213    /run/systemd/journal/socket
unix  2      [ ]         DGRAM                    31327    /var/run/chrony/chronyd.sock
unix  2      [ ]         DGRAM                    32379    /run/systemd/home/notify
unix  2      [ ]         DGRAM                    87408
@userdb-4e259eaec8a74395f53e9d957a38b841
unix  2      [ ]         DGRAM                    38053    @0000c
```

```
$ netstat -i
Kernel Interface table
Iface             MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
br-38fb360b0b9f  1500        0      0      0 0             0      0      0      0 BMU
docker0          1500        0      0      0 0             0      0      0      0 BMU
enp1s0           1500        0      0      0 0             0      0      0      0 BMU
lo              65536       24      0      0 0            24      0      0      0 LRU
virbr0           1500        0      0      0 0             0      0      0      0 BMU
wlp0s20f3        1500   150012      0      0 0         76586      0      0      0 BMRU
```

ss Command

The ss command is designed to show socket statistics and supports all the major packet and socket types. Meant to be a replacement for and to be similar in function to the netstat command, it also shows a lot more information and has more features.

- ss
- ss -s
- ss -l -t -u -x

The output is very similar to the output of the netstat command with no options. The columns above are:

Netid - the socket type and transport protocol
State - Connected and Unconnected, depending on protocol
Recv-Q - Amount of data queued up for being processed having been received
Send-Q - Amount of data queued up for being sent to another host
Local Address - The address and port of the local host's portion of the connection
Peer Address - The address and port of the remote host's portion of the connection

```
$ ss -tp
State      Recv-Q      Send-Q            Local Address:Port                Peer Address:Port         Process
ESTAB      0           0                 192.168.1.45:54710           142.250.184.165:https        users:(("chrome",pid=3250,fd=40))
ESTAB      0           0                 192.168.1.45:60586               5.79.35.160:https        users:(("chrome",pid=3250,fd=100))
ESTAB      0           0                 192.168.1.45:59326            64.233.184.188:hpvroom      users:(("chrome",pid=3250,fd=46))
ESTAB      0           0                 192.168.1.45:59968           142.250.200.142:https        users:(("chrome",pid=3250,fd=48))
ESTAB      0           0                 192.168.1.45:53536             52.37.107.177:https
ESTAB      0           0                 192.168.1.45:41374             216.58.209.74:https        users:(("chrome",pid=3250,fd=67))
ESTAB      0           0                 192.168.1.45:40838            64.233.166.189:https        users:(("chrome",pid=3250,fd=75))
ESTAB      0           0                 192.168.1.45:57000            142.250.201.74:https        users:(("chrome",pid=3250,fd=39))
ESTAB      0           0                 192.168.1.45:49906           142.250.184.174:https        users:(("chrome",pid=3250,fd=63))
ESTAB      0           0                 192.168.1.45:52344            216.58.215.142:https        users:(("chrome",pid=3250,fd=57))
ESTAB      0           0                 192.168.1.45:41000           142.250.200.131:https        users:(("chrome",pid=3250,fd=68))
ESTAB      0           0                 192.168.1.45:41030           142.250.200.131:https        users:(("chrome",pid=3250,fd=85))
```

```
$ ss -lu
State              Recv-Q             Send-Q                        Local Address:Port                        Peer Address:Port             Process
UNCONN             0                  0                                0.0.0.0:52153                            0.0.0.0:*
UNCONN             0                  0                           192.168.122.1:domain                          0.0.0.0:*
UNCONN             0                  0                      0.0.0.0%virbr0:bootps                         0.0.0.0:*
UNCONN             0                  0                              127.0.0.1:323                            0.0.0.0:*
UNCONN             0                  0                          224.0.0.251:mdns                             0.0.0.0:*
UNCONN             0                  0                              0.0.0.0:mdns                             0.0.0.0:*
UNCONN             0                  0                                 [::1]:323                                [::]:*
UNCONN             0                  0                                 [::]:50172                               [::]:*
UNCONN             0                  0                                 [::]:mdns                                [::]:*
```

```
$ ss -s
Total: 1172
TCP:   21 (estab 11, closed 5, orphaned 0, timewait 1)
Transport Total     IP        IPv6
RAW       1         0         1
UDP       10        7         3
TCP       16        13        3
INET      27        20        7
FRAG      0         0         0
```

ip command

The ifconfig command is becoming obsolete (deprecated) in some Linux distributions and is being replaced with a form of the ip command, specifically ip address.

- ip neighbor
- ip link
- ip address
- ip [-s]
- ip route

ip neigh

- ip neighbor show
- ip neighbor show dev <device>

```
$ ip neigh show
192.168.1.1 dev wlp0s20f3 lladdr e0:41:36:0e:5a:f0 REACHABLE
fe80::e241:36ff:fe0e:5af0 dev wlp0s20f3 lladdr e0:41:36:0e:5a:f0 router STALE

$ ip neigh show dev wlp0s20f3
192.168.1.1 lladdr e0:41:36:0e:5a:f0 REACHABLE
fe80::e241:36ff:fe0e:5af0 lladdr e0:41:36:0e:5a:f0 router STALE
```

```
$ arp
```

ip link

- ip link show
- ip link show dev device
- ip link set dev <device> up
- ip link set dev <device> down

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN mode
DEFAULT group default qlen 1000
    link/ether c0:18:50:14:97:b3 brd ff:ff:ff:ff:ff:ff
3: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
DORMANT group default qlen 1000
    link/ether 84:1b:77:00:78:c8 brd ff:ff:ff:ff:ff:ff
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:6c:e0:04 brd ff:ff:ff:ff:ff:ff
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN
mode DEFAULT group default qlen 1000
    link/ether 52:54:00:6c:e0:04 brd ff:ff:ff:ff:ff:ff
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
DEFAULT group default
    link/ether 02:42:f0:26:60:cf brd ff:ff:ff:ff:ff:ff
7: br-38fb360b0b9f: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN mode DEFAULT group default
    link/ether 02:42:4c:19:93:7c brd ff:ff:ff:ff:ff:ff
```

```
$ ip -br link  show
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
enp1s0           DOWN           c0:18:50:14:97:b3 <NO-CARRIER,BROADCAST,MULTICAST,UP>
wlp0s20f3        UP             84:1b:77:00:78:c8 <BROADCAST,MULTICAST,UP,LOWER_UP>
virbr0           DOWN           52:54:00:6c:e0:04 <NO-CARRIER,BROADCAST,MULTICAST,UP>
virbr0-nic       DOWN           52:54:00:6c:e0:04 <BROADCAST,MULTICAST>
docker0          DOWN           02:42:f0:26:60:cf <NO-CARRIER,BROADCAST,MULTICAST,UP>
br-38fb360b0b9f  DOWN           02:42:4c:19:93:7c <NO-CARRIER,BROADCAST,MULTICAST,UP>
```

```
$ ip  link  show enp1s0
2: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN mode
DEFAULT group default qlen 1000
    link/ether c0:18:50:14:97:b3 brd ff:ff:ff:ff:ff:ff
$
$ ip  link  show wlp0s20f3
3: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
DORMANT group default qlen 1000
    link/ether 84:1b:77:00:78:c8 brd ff:ff:ff:ff:ff:ff
```

```
[root@mylaptop ~]# ls /sys/class/net/
br-38fb360b0b9f  docker0  enp1s0  lo  virbr0  virbr0-nic  wlp0s20f3
```

```
# ip link set dev enp0s8 down
# ip link set dev enp0s8 up

# ip link set enp0s8 mtu 2000
# ifconfig enp0s3 mtu 1500

# ip link show dev enp0s8
```

ip address

- ip address show
- ip address show dev <device>
- ip address add ip/mask dev <device>
- ip address del ip/mask dev <device>

```
$ ip address 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: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group
default qlen 1000
    link/ether c0:18:50:14:97:b3 brd ff:ff:ff:ff:ff:ff
3: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether 84:1b:77:00:78:c8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.45/24 brd 192.168.1.255 scope global dynamic noprefixroute wlp0s20f3
       valid_lft 39380sec preferred_lft 39380sec
    inet6 fe80::2062:d7b2:5a59:f895/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group
default qlen 1000
    link/ether 52:54:00:6c:e0:04 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN
group default qlen 1000
    link/ether 52:54:00:6c:e0:04 brd ff:ff:ff:ff:ff:ff
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group
default
    link/ether 02:42:f0:26:60:cf brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
7: br-38fb360b0b9f: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default
    link/ether 02:42:4c:19:93:7c brd ff:ff:ff:ff:ff:ff
    inet 192.168.49.1/24 brd 192.168.49.255 scope global br-38fb360b0b9f
       valid_lft forever preferred_lft forever
```

```
$ ip addr show  wlp0s20f3
3: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether 84:1b:77:00:78:c8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.45/24 brd 192.168.1.255 scope global dynamic noprefixroute wlp0s20f3
       valid_lft 39344sec preferred_lft 39344sec
    inet6 fe80::2062:d7b2:5a59:f895/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

$ ip addr show  docker0
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group
default
    link/ether 02:42:f0:26:60:cf brd ff:ff:ff:ff:ff:ff
```

```
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

```
# ip addr add 192.168.5.5/24 dev enp0s8
# ip addr add 2001:db8::10/64 dev enp0s8
```

```
# ifconfig eth2 192.168.50.50 netmask 255.255.255.0
# ifconfig eth2 192.168.50.50 netmask 0xffffff00
# ifconfig enp0s8 add 2001:db8::10/64
```

ip route

- ip route show
- ip route add net/mask dev <device>
- ip route del net/mask dev <device>
- ip route add default via ip-gateway
- ip route del default via ip-gateway

The output of ip route and ip -6 route reads as follows:

- Destination.
- Optional address followed by interface.
- The routing protocol used to add the route.
- The scope of the route. If this is omitted, it is global scope, or a gateway.
- The route's metric. This is used by dynamic routing protocols to determine the cost of the route. This isn't used by most systems.
- If it is an IPv6 route, the RFC4191 route preference.

```
$ ip route show
default via 192.168.1.1 dev wlp0s20f3 proto dhcp metric 600
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.1.0/24 dev wlp0s20f3 proto kernel scope link src 192.168.1.45 metric 600
192.168.49.0/24 dev br-38fb360b0b9f proto kernel scope link src 192.168.49.1 linkdown
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 linkdown
```

```
$ netstat -r

$ ip route

$ route
```

```
# ip route add 192.68.10..0/24 dev enp1s8

# ip route del 192.168.10.0/24 dev enp1s8

# ip route add default via 192.168.1.1

# ip route del default via 192.168.1.1
```

ping command

The packet internet groper ping command is used to check the connectivity to a host. It is a simple test that can be performed from the command prompt when a particular network service is not available. This utility sends the ICMP protocol's ECHO_REQUEST (ping) datagram to a host, and the host sends an ECHO_RESPONSE (pong) datagram in response. Each datagram will have an IP header, an ICMP header, a timeval structure, and some additional bytes used for padding.

The Time To Live (TTL) value is the maximum number of IP routers that may attempt to route a packet. Every time a router attempts to route the packets, its TTL count is decremented by 1. If a router receives a packet and the TTL of a packet is zero, then the packet is discarded.

- ping
- ping -c

Some of the ping options:

-c count	Stop after sending count ECHO_REQUEST packets
-s packetsize	Specifies the number of data bytes to be sent
-t ttl	Sets the IP Time to Live
-w timeout	Sets the timeout in seconds for ping to exit

The ping6 command is similar to the ping command, but it uses ICMPv6 ECHO_REQUEST to verify network connectivity. The ping6 command can use either a hostname or an IPv6

```
$ ping www.google.com
PING www.google.com (142.250.201.68) 56(84) bytes of data.
64 bytes from mad07s25-in-f4.1e100.net (142.250.201.68): icmp_seq=1 ttl=115 time=13.3 ms
64 bytes from mad07s25-in-f4.1e100.net (142.250.201.68): icmp_seq=2 ttl=115 time=10.7 ms
^C
--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 10.701/11.997/13.294/1.296 ms


$ ping6  www.google.com
ping6: connect: Network is unreachable
$ ping google.com
PING google.com (172.217.17.14) 56(84) bytes of data.
64 bytes from mad07s09-in-f14.1e100.net (172.217.17.14): icmp_seq=1 ttl=115 time=9.99 ms
64 bytes from mad07s09-in-f14.1e100.net (172.217.17.14): icmp_seq=2 ttl=115 time=11.6 ms
^C64 bytes from 172.217.17.14: icmp_seq=3 ttl=115 time=170 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 9.988/63.995/170.409/75.248 ms
```

traceroute command

The traceroute command is used to trace the route of packets to a specified host. This utility uses the IP header's TTL field and tries to fetch an ICMP TIME_EXCEEDED response from

each router on the path to the host. The probing is done by sending ICMP ping packets with a small TTL value and then checking the ICMP TIME_EXCEEDED response.

The traceroute command, commonly used for seeing how a transmission travels between a local host machine to a remote system

-T	Probe using TCP SYN
-f first_ttl	Specifies the initial TTL value
-m max_ttl	Specifies the maximum number of hops to be probed
-w timeout	Sets the timeout in seconds to exit after waiting for a response to a probe

```
$ traceroute lms.pue.es
traceroute to lms.pue.es (51.15.184.105), 30 hops max, 60 byte packets
 1  _gateway (192.168.1.1)  1.767 ms  1.934 ms  3.208 ms
 2  136.red-81-46-38.customer.static.ccgg.telefonica.net (81.46.38.136)  6.596 ms  7.362
ms  7.557 ms
 3  33.red-81-46-34.customer.static.ccgg.telefonica.net (81.46.34.33)  9.070 ms  9.029
ms  12.602 ms
 4  * * *
 5  * * *
 6  * be8-400-grtbcnes1.net.telefonicaglobalsolutions.com (213.140.50.218)  3.524 ms *
 7  176.52.248.191 (176.52.248.191)  10.713 ms  14.024 ms  11.253 ms
 8  176.52.248.172 (176.52.248.172)  15.666 ms 176.52.248.174 (176.52.248.174)  16.119
ms 176.52.248.172 (176.52.248.172)  16.308 ms
 9  176.52.248.178 (176.52.248.178)  17.559 ms lag-34.ear1.Madrid1.Level3.net
(4.68.39.69)  24.611 ms  14.123 ms
10  * * *
11  212.3.235.198 (212.3.235.198)  24.855 ms ae-5.edge1.Madrid2.Level3.net (4.68.39.33)
12.252 ms  13.188 ms
12  51.158.8.71 (51.158.8.71)  25.369 ms  28.165 ms  24.423 ms
13  212.3.235.198 (212.3.235.198)  25.134 ms  26.505 ms  26.467 ms
14  siurana.pue.es (51.15.184.105)  26.436 ms !X  26.871 ms !X  23.521 ms !X

$ traceroute labs.pue.es
traceroute to labs.pue.es (212.83.191.190), 30 hops max, 60 byte packets
 1  _gateway (192.168.1.1)  1.508 ms  1.849 ms  2.695 ms
 2  136.red-81-46-38.customer.static.ccgg.telefonica.net (81.46.38.136)  7.717 ms  7.955
ms  9.074 ms
 3  33.red-81-46-34.customer.static.ccgg.telefonica.net (81.46.34.33)  11.196 ms  11.169
ms  11.141 ms
 4  133.red-81-46-34.customer.static.ccgg.telefonica.net (81.46.34.133)  24.399 ms
24.372 ms  24.344 ms
 5  46.red-80-58-81.staticip.rima-tde.net (80.58.81.46)  12.660 ms  12.913 ms  13.601 ms
 6  213.140.50.204 (213.140.50.204)  15.109 ms  3.606 ms  6.140 ms
...
```

tracepath command

The tracepath command is used to trace the path to a network host, discovering MTU (maximum transmission unit) along the path. The functionality is similar to traceroute. It sends ICMP and UDP messages of various sizes to find the MTU size on the path. Using UDP messages to trace the path can be useful when routers are configured to filter ICMP traffic.

```
$ tracepath lms.pue.es
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                              4.125ms
 1:  _gateway                                              6.719ms
```

```
 2:  _gateway                                              7.006ms pmtu 1492
 2:  136.red-81-46-38.customer.static.ccgg.telefonica.net  14.672ms
 3:  33.red-81-46-34.customer.static.ccgg.telefonica.net   8.155ms asymm  8
 4:  no reply
 5:  no reply
 6:  be8-400-grtbcnes1.net.telefonicaglobalsolutions.com  11.160ms asymm  7
 7:  176.52.248.191                                       13.968ms asymm  8
 8:  5.53.6.64                                            69.992ms asymm  9
 9:  lag-34.ear1.Madrid1.Level3.net                       17.411ms asymm  8
10:  176.52.248.247                                       18.191ms asymm  7
11:  212.3.235.198                                        27.129ms asymm 10
12:  51.158.8.71                                          31.315ms asymm 11
13:  212.3.235.198                                        26.832ms asymm 10
14:  51.158.8.71                                          27.029ms asymm 11
15:  195.154.2.7                                          28.436ms asymm 11
16:  siurana.pue.es                                       25.865ms !H
     Resume: pmtu 1492
```

ethtool command

The ethtool utility is useful for configuring and troubleshooting network devices such as Ethernet cards and their device drivers, and other useful troubleshooting information, such as the speed of an interface.

```
$ ethtool enp1s0
Settings for enp1s0:
        Supported ports: [ TP  MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: Symmetric Receive-only
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: Symmetric Receive-only
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Speed: Unknown!
        Duplex: Unknown! (255)
        Auto-negotiation: on
        master-slave cfg: preferred slave
        master-slave status: unknown
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: external
        MDI-X: Unknown
netlink error: Operation not permitted
        Link detected: no
```

```
$ ethtool -i enp1s0
driver: r8169
version: 5.11.22-100.fc32.x86_64
firmware-version: rtl8168h-2_0.0.2 02/26/15
expansion-rom-version:
bus-info: 0000:01:00.0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: yes
supports-priv-flags: no
```

```
$ ethtool -i wlp0s20f3
driver: iwlwifi
version: 5.11.22-100.fc32.x86_64
firmware-version: 59.601f3a66.0 Qu-c0-hr-b0-59.uc
expansion-rom-version:
bus-info: 0000:00:14.3
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no

# ethtool wlp0s20f3
Settings for wlp0s20f3:
       Link detected: yes
```

```
$ iwconfig
lo        no wireless extensions.

enp1s0    no wireless extensions.

wlp0s20f3  IEEE 802.11  ESSID:"MOVISTAR_5AF0"
          Mode:Managed  Frequency:2.437 GHz  Access Point: E2:41:36:0E:5A:F0
          Bit Rate=48 Mb/s   Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
          Link Quality=53/70  Signal level=-57 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:38  Invalid misc:22   Missed beacon:0

virbr0    no wireless extensions.

virbr0-nic  no wireless extensions.

docker0   no wireless extensions.

br-38fb360b0b9f  no wireless extensions.
```

```
$ iwconfig wlp0s20f3
wlp0s20f3  IEEE 802.11  ESSID:"MOVISTAR_5AF0"
          Mode:Managed  Frequency:2.437 GHz  Access Point: E2:41:36:0E:5A:F0
          Bit Rate=48 Mb/s   Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=51/70  Signal level=-59 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:38  Invalid misc:22   Missed beacon:0
```

netcat command

The netcat utility is a cross-platform tool it can be used for monitoring and debugging network connections. The netcat command can also be used in the short form, which is nc.

```
$ netcat -z localhost 1-100
Connection to localhost (::1) 21 port [tcp/ftp] succeeded!
Connection to localhost (::1) 80 port [tcp/http] succeeded!

$ netcat -zv localhost 1-100
...
```

```
(host1)$ sudo netcat -l 5001

(host2)$ netcat host1 5001
```

```
(tty1)$ sudo netcat -l 5001

(tty2)$ netcat localhost 5001
```

```
$ netcat smtp.gmail.com 25
220 smtp.gmail.com ESMTP be3sm596522wmb.1 - gsmtp
EHLO
501-5.5.4 Empty HELO/EHLO argument not allowed, closing connection.
501 5.5.4  https://support.google.com/mail/?p=helo be3sm596522wmb.1 - gsmtp
...
```

Network interface & Layers

OSI network layers:

7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data-Link
1 Physical

Physical Layer

The physical layer of the OSI model defines hardware connections and turns binary data into physical pulses (electrical, light, or radio waves).

“Is the device on?”, “Is my network card detected?”, and “Is the network card connected?”

- lspci
- lsusb
- lsmod
- ip link show

```
# lspci  | grep -E "Network|Ethernet"
00:14.3 Network controller: Intel Corporation Killer Wi-Fi 6 AX1650i 160MHz Wireless
Network Adapter (201NGW) (rev 30)
01:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI
Express Gigabit Ethernet Controller (rev 15)

# lspci -vs  00:14.3
00:14.3 Network controller: Intel Corporation Killer Wi-Fi 6 AX1650i 160MHz Wireless
```

```
Network Adapter (201NGW) (rev 30)
        Subsystem: Intel Corporation Device 0074
        Flags: bus master, fast devsel, latency 0, IRQ 16
        Memory at 81430000 (64-bit, non-prefetchable) [size=16K]
        Capabilities: [c8] Power Management version 3
        Capabilities: [d0] MSI: Enable- Count=1/1 Maskable- 64bit+
        Capabilities: [40] Express Root Complex Integrated Endpoint, MSI 00
        Capabilities: [80] MSI-X: Enable+ Count=16 Masked-
        Capabilities: [100] Latency Tolerance Reporting
        Capabilities: [164] Vendor Specific Information: ID=0010 Rev=0 Len=014 <?>
        Kernel driver in use: iwlwifi
        Kernel modules: iwlwifi

# lspci -vs  01:00.0
01:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI
Express Gigabit Ethernet Controller (rev 15)
        Subsystem: Lenovo Device 3852
        Flags: bus master, fast devsel, latency 0, IRQ 16
        I/O ports at 2000 [size=256]
        Memory at 81304000 (64-bit, non-prefetchable) [size=4K]
        Memory at 81300000 (64-bit, non-prefetchable) [size=16K]
        Capabilities: [40] Power Management version 3
        Capabilities: [50] MSI: Enable- Count=1/1 Maskable- 64bit+
        Capabilities: [70] Express Endpoint, MSI 01
        Capabilities: [b0] MSI-X: Enable+ Count=4 Masked-
        Capabilities: [100] Advanced Error Reporting
        Capabilities: [140] Virtual Channel
        Capabilities: [160] Device Serial Number 00-00-00-00-00-00-00-00
        Capabilities: [170] Latency Tolerance Reporting
        Capabilities: [178] L1 PM Substates
        Kernel driver in use: r8169
        Kernel modules: r8169
```

```
# modinfo r8169

# modinfo iwlwifi
```

```
$ ip link show dev enp1s0
2: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN mode
DEFAULT group default qlen 1000
    link/ether c0:18:50:14:97:b3 brd ff:ff:ff:ff:ff:ff

$ ip link show dev wlp0s20f3
3: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
DORMANT group default qlen 1000
    link/ether 84:1b:77:00:78:c8 brd ff:ff:ff:ff:ff:ff
```

```
# ip link set eth0 up

# ip link set eth0 down
```

Data link layer

The data-link layer of the OSI model defines interface to the physical layer. It also monitors and corrects for errors that may occur in the physical layer by using a frame check sequence (FCS). The data-link layer keeps a table of IP address to MAC address translations.

“Does this computer see any devices on the network?”

- ip neighbor
- ethtool
- arp [-a]

```
$ ip neigh show
192.168.1.1 dev wlp0s20f3 lladdr e0:41:36:0e:5a:f0 REACHABLE
fe80::e241:36ff:fe0e:5af0 dev wlp0s20f3 lladdr e0:41:36:0e:5a:f0 router STALE


$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   e0:41:36:0e:5a:f0   C                     wlp0s20f3

$ arp -a
_gateway (192.168.1.1) at e0:41:36:0e:5a:f0 [ether] on wlp0s20f3

$ ethtool eth0
```

Network Layer

The network layer of the OSI model performs network routing functions, defines logical addresses, and uses a hierarchical addressing scheme. Various protocols specify packet structure and processing used to carry data from host to host.
“Does this device have an IP address?” and “Is the gateway address set on this device?”

- ip address
- ip route
- dhclient -v etho
- dhclient -r eth0

```
$ ip addr show lo
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

# ip a show docker0
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group
default
    link/ether 02:42:f0:26:60:cf brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

```
# dhclient -v wlp0s20f3
Internet Systems Consortium DHCP Client 4.4.2b1
Copyright 2004-2019 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlp0s20f3/84:1b:77:00:78:c8
Sending on   LPF/wlp0s20f3/84:1b:77:00:78:c8
Sending on   Socket/fallback
DHCPDISCOVER on wlp0s20f3 to 255.255.255.255 port 67 interval 3 (xid=0xd6cb6850)
DHCPOFFER of 192.168.1.45 from 192.168.1.1
DHCPREQUEST for 192.168.1.45 on wlp0s20f3 to 255.255.255.255 port 67 (xid=0xd6cb6850)
```

```
DHCPACK of 192.168.1.45 from 192.168.1.1 (xid=0xd6cb6850)
bound to 192.168.1.45 -- renewal in 19339 seconds.
```

Routing Table

When checking network connectivity, ensure that your system can get to the assigned gateway. The network gateway, as defined in your network interface configuration, is the “first hop” or the first place your computer will go to when looking for resources beyond the local network.

```
$ ip route
default via 192.168.1.1 dev wlp0s20f3 proto dhcp metric 600
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.1.0/24 dev wlp0s20f3 proto kernel scope link src 192.168.1.45 metric 600
192.168.49.0/24 dev br-38fb360b0b9f proto kernel scope link src 192.168.49.1 linkdown
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 linkdown
```

Domain Name System

Beyond network connectivity, uniform resource locator (URL) addresses need to be resolved to IP addresses using DNS.

- nslookup
- host
- dig
- /etc/hosts
- /etc/resolv.conf

```
# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.122.28  centos20

# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
search Home edt.org
nameserver 80.58.61.250
nameserver 80.58.61.254
```

```
# nslookup pue.es
Server:         80.58.61.250
Address:        80.58.61.250#53

Non-authoritative answer:
Name:   pue.es
Address: 176.34.150.171

# host -t A pue.es
pue.es has address 176.34.150.171


# dig -t NS pue.es

; <<>> DiG 9.11.28-RedHat-9.11.28-1.fc32 <<>> -t NS pue.es
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15466
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 9
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;pue.es.                                        IN          NS

;; ANSWER SECTION:
pue.es.                                 86400   IN          NS          ns-769.awsdns-32.net.
pue.es.                                 86400   IN          NS          ns-1988.awsdns-56.co.uk.
pue.es.                                 86400   IN          NS          ns-1113.awsdns-11.org.
pue.es.                                 86400   IN          NS          ns-224.awsdns-28.com.

;; ADDITIONAL SECTION:
ns-1113.awsdns-11.org.          65950   IN          A           205.251.196.89
ns-1113.awsdns-11.org.          66226   IN          AAAA        2600:9000:5304:5900::1
ns-1988.awsdns-56.co.uk. 66066  IN          A       205.251.199.196
ns-1988.awsdns-56.co.uk. 66665  IN          AAAA        2600:9000:5307:c400::1
ns-224.awsdns-28.com.           66119   IN          A           205.251.192.224
ns-224.awsdns-28.com.           66440   IN          AAAA        2600:9000:5300:e000::1
ns-769.awsdns-32.net.           65732   IN          A           205.251.195.1
ns-769.awsdns-32.net.           66271   IN          AAAA        2600:9000:5303:100::1

;; Query time: 55 msec
;; SERVER: 80.58.61.250#53(80.58.61.250)
;; WHEN: Mon Nov 15 20:33:17 CET 2021
;; MSG SIZE  rcvd: 351
```

Firewalls

Linux uses IP tables to manage network traffic, and in RH systems firewalld.

- ipchains (old)
- iptables
- ip6tables
- firewalld

```
# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  anywhere             anywhere             ADDRTYPE match dst-type
LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  anywhere            !127.0.0.0/8          ADDRTYPE match dst-type
LOCAL

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.17.0.0/16        anywhere
MASQUERADE  all  --  192.168.49.0/24      anywhere
LIBVIRT_PRT  all  --  anywhere             anywhere

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain LIBVIRT_PRT (1 references)
target     prot opt source               destination
RETURN     all  --  192.168.122.0/24     base-address.mcast.net/24
RETURN     all  --  192.168.122.0/24     255.255.255.255
MASQUERADE  tcp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-65535
MASQUERADE  udp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-65535
MASQUERADE  all  --  192.168.122.0/24    !192.168.122.0/24
```

```
# systemctl  status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset:
enabled)
     Active: active (running) since Mon 2021-11-15 19:02:09 CET; 1h 34min ago
       Docs: man:firewalld(1)
   Main PID: 826 (firewalld)
      Tasks: 2 (limit: 8916)
     Memory: 42.6M
        CPU: 1.069s
```

```
     CGroup: /system.slice/firewalld.service
             └─826 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
```

up/down and delete interfaces

The ifup and ifdown legacy commands are used to bring up and bring down a network interface, respectively. Now the ip link command is used.

- ifup [deprecated]
- ifdown [deprecated
- ip link set eth0 up
- ip link set eth0 down

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

To delete an interface and make the change permanent, the configuration file for the corresponding interface should be deleted. After modifying the file the networking service should be restarted

- /etc/sysconfig/network-scripts/ifcfg-eth1 (Red Hat)
- /etc/network/interfaces

```
# rm /etc/sysconfig/network-scripts/ifcfg-eth1
```

Transport Layer

The transport layer of the OSI model performs transparent transfer of data between end users. It is responsible for error recovery and flow control and ensures complete data transfer.

To find out if the service is running and if it can be reached use the commands:

- ss
- netstat
- nmap
- ncat

```
# nmap localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2021-11-15 20:48 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 997 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
80/tcp  open  http
631/tcp open  ipp
```

Session & Presentation & Application Layers

The session, presentation, and application layers of the OSI model are all handled by software.

To get a good look at the inner workings of network communications, capture some data using tcpdump, iptrace or Wireshark.

Example Exercises

1. Which commands can be used to list network interfaces?
2. How would you temporarily disable an interface? How would you re-enable it?
3. Which commands can you use to show your default route?
4. How would add a second IP address to an interface?
5. How would you configure a default route?

6. What command(s) would you use to send an ICMP echo to learning.lpi.org?
7. How could you determine the route to 8.8.8.8?
8. What command would show you if any processes are listening on TCP port 80?
9. How could you determine the max MTU of a network path?

10. Realitza els exercicis indicats a: 109.3 Basic network troubleshooting
11. Realitza els exercicis del Question-Topics 108.3