Basic TCP/IP Networking

- Setting up TCP/IP on a machine consists of:
    - Setting the hostname
    - Assigning an IP address and subnet mask
    - Assigning a default gateway address
- All the above tasks involve manipulating the ***/etc/hosts*** and ***/etc/networks*** files either manually or using GUI-based tools such as ***netcfg*** or ***linuxconf***.

Setting the Hostname

- This is normally done during the bootup procedure by executing the ***hostname*** command.
- It can be invoked from command line as:

    hostname name

- The argument ***name*** is the unqualified hostname without the domain name.
- Since the local hostname is used to look up the host's IP address, you must ensure that the resolver will able to look up the host's IP address. This is accomplished by entering the name in ***/etc/hosts.***

Creating Subnets

- Large networks are typically segmented into smaller networks in the interests of efficiency, manageability and security.
- To accommodate the several networks a network administrator may use eight bits of the host part as additional subnet bits.
- This leaves another eight bits for the host part, allowing for 254 hosts on each of the subnets.
- Each individual network will now be assigned a subnet number. For example we can have two subnets within a larger network with network addresses: **142.232.66.0** and **142.232.132.0.** The subnet mask is **255.255.255.0.**

Assigning IP Addresses

- IP addresses can be assigned to a host either statically (***/etc/hosts***) or dynamically (DHCP).

- The following are the contents of a sample ***hosts*** file:

  # IP Address FQDN Aliases

```
10.42.24.5 ithaca.opus42.net ithaca
10.42.24.2 odyssey.opus42.net odyssey
10.42.24.11 magrathea.opus42.net magrathea
10.42.24.20 zeus.opus42.net zeus
10.42.24.10 valkyerie.opus42.net valkyerie
```

- To ensure that all applications use ***/etc/hosts*** first when looking up the IP address of a host, you have to edit the ***/etc/host.conf*** file.

- The following line in that file will ensure that the resolver will first check the **hosts** file, then use **bind** and finally **nis** to resolve a hostname.

  order hosts, bind, nis

- After the devices have been configured, they must be connected to the kernel networking software. This is done using the ***ifconfig*** (where "if" stands for interface) and ***route*** commands.

- The ***ifconfig*** command is a very powerful tool used to monitor and configure network devices.

- Its normal invocation is as follows:

  ifconfig interface [address] [parameters]

- ***interface*** is the interface name, and ***address*** is the IP address to be assigned to the interface. This may be either an IP address in dotted quad notation or a name that ***ifconfig*** will look up in ***/etc/hosts.***

- If *ifconfig* is invoked with only the interface name, it displays that interface's configuration. When invoked without any parameters, it displays all interfaces you have configured so far; an option of *-a* forces it to show the inactive ones as well.

- A sample invocation for the Ethernet interface **eth0** may look like this:

 # /sbin/ifconfig

```
eth0 Link encap:Ethernet HWaddr 00:50:04:87:2C:74
     inet addr:10.42.24.5 Bcast:10.42.255.255 Mask:255.255.0.0
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:24185 errors:0 dropped:0 overruns:0 frame:0
     TX packets:7503 errors:0 dropped:0 overruns:0 carrier:0
     collisions:628 txqueuelen:100
     Interrupt:16 Base address:0xe400

lo  Link encap:Local Loopback
    inet addr:127.0.0.1 Mask:255.0.0.0
    UP LOOPBACK RUNNING MTU:3924 Metric:1
    RX packets:40 errors:0 dropped:0 overruns:0 frame:0
    TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
```

- The MTU and metric fields show the current MTU and metric value for that interface. The metric value is traditionally used by some operating systems to compute the cost of a route. Linux doesn't use this value yet, but defines it for compatibility nevertheless.

- The Rx and Tx lines show how many packets have been received or transmitted error free, how many errors occurred, how many packets were dropped (probably because of low memory), and how many were lost because of an overrun.

- Receiver overruns usually happen when packets come in faster than the kernel can service the last interrupt. The flag values printed by ***ifconfig*** correspond more or less to the names of its command-line options; they will be explained below.

- The following is a list of parameters recognized by ifconfig with the corresponding flag names. Options are used turn features on or off by preceding the option name by a hyphen.

- **up**

    - This is used to enable a network interface. It may also be used to re-enable an interface that has been taken down temporarily using the down option. (This option corresponds to the flags *UP and RUNNING.)*

- **down**

    - This is used to disable a network interface. It effectively disables any IP traffic through the interface.

- **netmask** ***mask***

  Assigns a subnet mask to be used by the interface. It may be given as either a 32-bit hexadecimal number preceded by 0x, or as a dotted quad of decimal numbers.

- **pointopoint** ***address***

  - This option is used for point-to-point IP links that involve only two hosts. (If a point-to-point address has been set, ***ifconfig*** displays the **POINTOPOINT** flag.)

  - **broadcast** ***address***

  - The broadcast address is usually made up from the network number by setting all bits of the host part on (If a broadcast address has been set, ***ifconfig*** displays the **BROADCAST** flag).

- **metric** ***number***

  - This option may be used to assign a metric value to the routing table entry created for the interface. This metric is used by the Routing Information Protocol (RIP) to build routing tables for the network.

  - The default metric used by RIP chooses the optimal route to a given host based on the "length" of the path. It is computed by summing up the individual metric values of each host-to-host link.

  - By default, a hop has length 1, but this may be any positive integer less than 16. (A route length of 16 is equal to infinity. Such routes are considered unusable.) The *metric* parameter sets this hop cost, which is then broadcast by the routing daemon.

- **mtu** ***bytes***

  - Sets the **Maximum Transmission Unit**, which is the maximum number of octets the interface is able to handle in one transaction. For Ethernet the MTU defaults to 1500.

- **arp**

  - Specific to broadcast networks such as Ethernet. It enables the use of ARP, the **Address Resolution Protocol**, to detect the physical addresses of hosts attached to the network.

  - For broadcast networks, it is on by default (If ARP is disabled, ***ifconfig*** displays the flag NOARP).

- **-arp**

  - Disables the use of ARP on this interface.

- **promisc**
    - Puts the interface in promiscuous mode. On a broadcast network, this makes the interface receive all packets, regardless of whether they were destined for another host or not.
    - This allows an analysis of network traffic using packet filters and such, also called *Ethernet snooping.* Usually, this is a good technique of hunting down network problems that are otherwise hard to detect.
    - This also allows attackers to intercept the traffic on your network for passwords and other information. (This option corresponds to the flag *PROMISC.)*
- **-promisc**
    - Turns off promiscuous mode.
- **allmulti**
    - Enables the interface to receive all multicast packets on the network (this option corresponds to the MULTICAST flag).
- **-allmulti**
    - Disables multicast mode.

- The following command will assign the **eth0** interface the IP address of **ithaca** (10.42.24.5) and a subnet mask of **255.255.255.0**.

    #/sbin/ifconfig eth0 ithaca netmask 255.255.255.0

- The next step is to install a routing entry that informs the kernel about the network that can be reached through ***eth0.***

We use the ***route*** command to accomplish this as follows:

    # ***/sbin/route add -net 10.42.24.0***

- Note that the **-net** option is used because ***route*** can handle both routes to networks and routes to individual hosts.
- When being given an address in dotted quad notation, ***route*** attempts to guess whether it is a network or a hostname by examining the host field.
- If the address's host field is zero, ***route*** assumes it denotes a network; otherwise it takes it as a host address.

- The kernel routing table can be viewed as follows:

 # /sbin/route -n

```
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
142.232.66.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
127.0.0.1 0.0.0.0 255.0.0.0 UH 0 0 0 lo
0.0.0.0 142.232.66.253 0.0.0.0 UG 0 0 0 eth0
```

- The Flags column contains a list of flags set for each interface. U is always set for active interfaces, and H says the destination address denotes a host.

The Loopback Interface

The very first interface to be activated during bootup is the loopback interface:

 # /sbin/ifconfig lo 127.0.0.1

- Occasionally, you will also see the dummy hostname **localhost** being used instead of the IP address.

- The loopback interface is useful not only as a test interface during development, but is actually used by some applications during normal operation. For example, all applications based on RPC use the loopback interface to register themselves with the ***portmapper*** daemon at startup.

- Therefore, you always have to configure it, regardless of whether your machine is attached to a network or not.

- An entry in the routing table that tells IP that it may use this interface as a route to destination 127.0.0.1 is accomplished as follows:

  # /sbin/route add 127.0.0.1

Routing through a Gateway

- Networks connected to other networks by gateways. Gateways may link two or more networks as well as provide a link to the outside world, the Internet.
- In order to use the service of a gateway, you have to provide additional routing information to the networking layer.
- For example, two networks called **valhalla** and **ithaca** can be linked through a gateway called **milliways**.
-
- Assuming that **milliways** has already been configured, we only have to add another entry to routing table on host **valkyerie** in network **valhalla** that tells the kernel it can reach all hosts on network **ithaca** through **milliways**.
- The appropriate invocation of ***route*** on host **valkyerie** is shown below; the **gw** keyword tells it that the next argument denotes a **gateway**.

  # **route add ithaca gw milliways**

- Of course, any host on the **ithaca** network you wish to talk to must have a corresponding routing entry for that network otherwise you would only be able to send data one way.
- Assume that **milliways** also has a connection to the Internet. Then we would want datagrams to **any** destination network other than **ithaca** to be handed to **milliways**.
- This can be accomplished by making it the default gateway for **valkyerie**:

  # **route add default gw milliways**

- The network name **default** is shorthand for 0.0.0.0, which denotes the default route.

Configuring a Gateway

- Configuring a machine to switch packets between two networks is relatively simple. Assume that milliways is equipped with two Ethernet cards, each being connected to one of the two networks.

- We now have to configure both interfaces separately, giving them their respective IP addresses.

- It is quite useful to add information on the two interfaces to the ***hosts*** file as shown below, so we have names for them too:

  milliways.bcit.ca milliways milliways-if1
  milliways-if2

- The sequence of commands to set up the two interfaces is then:

  # /sbin/ifconfig eth0 milliways-if1
  # /sbin/ifconfig eth1 milliways-if2
  # /sbin/route add ithaca
  # /sbin/route add valhalla

- Note that in order for this to work the kernel must have been compiled with support for IP forwarding enabled.

The netstat Command

- ***netstat*** is a very useful tool for checking network activity and configuration.
- Invoking ***netstat*** with the -r flag displays the kernel routing table similar to the way we've been doing with ***route.***
- Here is an example:

#netstat -nr

```
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
142.232.66.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
127.0.0.1 0.0.0.0 255.0.0.0 UH 0 0 0 lo
0.0.0.0 142.232.66.253 0.0.0.0 UG 0 0 0 eth0
```

- The -n option makes *netstat* print addresses as dotted quad IP numbers rather than the symbolic host and network names.
- This is especially useful when you want to avoid address lookups over the network (e.g., to a DNS or NIS server).
- The second column of ***netstat***'s output shows the gateway the routing entry points to. If no gateway is used, an asterisk is printed instead.
- Column three shows the "generality" of the route. When given an IP address to find a suitable route for, the kernel goes through all routing table entries, performing a bitwise AND of the address and the Genmask before comparing it to the target of the route.
- The fourth column displays various flags that describe the route:

G: The route uses a gateway.

U: The interface to be used is up.

H: Only a single host can be reached through the route. For example, this is the case for the loopback entry 127.0.0.1.

D: This is set if the table entry has been generated by an ICMP redirect message.

M: This is set if the table entry was modified by an ICMP redirect message.

!: This is a reject route and the datagrams will be dropped.

- The next three two columns show the MSS, Window and irtt values that will be applied to TCP connections established via this route.

- The **MSS** is the **Maximum Segment Size** and is the largest datagram that the kernel will send via this route.

- **Window** is the maximum amount of data the system will accept in a single burst from a remote host. This is used for flow control.

- The **irtt** (**Initial Round Trip Time**) is the value that the TCP protocol will use when a connection is first established. This value is used to time the retransmissions.

- Values of 0 in these fields indicate that the default values are being used.

- ***netstat*** also supports a set of options to display active or passive sockets. The options *-t, -u, -w, and -x* show active TCP, UDP, RAW, or UNIX socket connections.

- If you provide the -a flag in addition, sockets that are waiting for a connection (i.e., listening) are displayed as well.

Checking the ARP Tables

- Sometimes it is useful to view or even alter the contents of the kernel's ARP tables, for example when you suspect a duplicate Internet address is the cause for some intermittent network problem.

- The ***arp*** tool is used for this:

  arp [-v] [-t hwtype] -a [hostnamel

  arp [-v] [-t hwtype] -s hostname hwaddr

  arp [-vl -d hostname [hosLname...]

- All ***hostname*** arguments may be either symbolic hostnames or IP addresses in dotted quad notation.

- Here is an example of invoking arp on one machine:

  # arp -a

  ? (142.232.66.253) at 00:00:0C:47:B1:17 [ether] on eth0

- The **-s** option is used to permanently add ***hostname***'s Ethernet address to the ARP tables. The **hwaddr** argument specifies the hardware address, which is by default the MAC address.

- One problem that may require you to manually add an IP address to the ARP table is when for some reason ARP queries for the remote host fail, when the ARP driver is buggy or there is another host in the network that erroneously identifies itself with that host's IP address.

- Hard-wiring IP addresses in the ARP table is also an extreme measure to protect yourself from hosts on your network that pose as someone else.

- The **-s** option may also be used to implement **proxy ARP**. This is a special technique where a host, say **gate,** acts as a gateway to another host named **fnord** by pretending that both addresses refer to the same host, namely **gate.**

- It does so by publishing an ARP entry for **fnord** that points to its own Ethernet interface. Now when a host sends out an ARP query for **fnord, gate** will return a reply containing its own Ethernet address.

- The querying host will then send all datagrams to **gate,** which forwards them to **fnord.**

- The proper invocation to provide proxy ARP for **fnord** is given below; the MAC address given must be that of **gate.**

    arp -s fnord 00:00:OC:47:B1:17 pub