Basic Network Configuration

Only a few steps are involved in adding a new machine to an existing local area network. The basic steps to add a new machine are as follows:

- Assign a unique IP address and hostname
- Make sure the network interfaces are properly configured at boot time
- Set up a default route and perhaps fancier routing
- Point to a DNS name server to allow access to the rest of the Internet.

If you rely on DHCP for basic provisioning, most of the configuration chores for a new machine are performed on the DHCP server rather than on the new machine itself. New OS installations typically default to getting their configuration through DHCP, so new machines may require no network configuration at all.

After any change that might affect booting, you should always reboot to verify that the machine comes up correctly. Six months later when the power has failed and the machine refuses to boot, it is hard to remember what changes you made that might have caused the problem.

As you work through basic network configuration on any machine, you will find it helpful to reset your connectivity with basic tools such as **ping** and **traceroute**.

**ping**: Check To See If a Host Is Alive

The **ping** command is embarrassingly simple, but in many situations it is the only command you need for network debugging. It sends an ICMP ECHO_REQUEST packet to a target host and waits to see if the host answers back.

You can use **ping** to check the status of individual hosts and to test segments of the network. Routing tables, physical networks, and gateways are all involved in processing a ping, so the network must be more or less working for **ping** to succeed. If **ping** does not work, you can be pretty sure that nothing more sophisticated will work either.

However, this rule does not apply to networks that block ICMP echo requests with a firewall. Make sure that a firewall is not interfering with your debugging before you conclude that the target host is ignoring a **ping**. You might consider disabling a meddlesome firewall for a short period of time to facilitate debugging.

If your network is in bad shape, chances are that DNS is not working. Simplify the situation by using numeric IP addresses when pinging, and use **ping**’s **-n** option to prevent **ping** from attempting to do reverse lookups on IP addresses -- these lookups also trigger DNS requests.

Be aware of the firewall issue if you are using **ping** to check your Internet connectivity. Some well-known sites answer **ping** packets and other do not. google.com has been a consistent responder.
Most versions of **ping** run in an infinite loop unless you supply a packet count argument. On Solaris, **ping** 

`-s` provides the extended output that other versions use by default. Once you have had your fill of pinging,

**e.g.**

```bash
$ /usr/sbin/ping -s google.com
PING google.com: 56 data bytes
64 bytes from yx-in-f147.1e100.net (74.125.45.147): icmp_seq=0. time=5.50 ms
64 bytes from yx-in-f147.1e100.net (74.125.45.147): icmp_seq=1. time=5.04 ms
64 bytes from yx-in-f147.1e100.net (74.125.45.147): icmp_seq=2. time=4.89 ms
64 bytes from yx-in-f147.1e100.net (74.125.45.147): icmp_seq=3. time=4.98 ms
^C
----google.com PING Statistics-----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max/stddev = 4.89/5.10/5.50/0.27
```

The output for google.com shows the host’s IP address, the ICMP sequence number of each response packet, and the round trip travel time. The most obvious thing that the output above tells us is that the server google.com is alive and connected to the network.

The ICMP sequence number is a particularly valuable piece of information. Discontinuities in the sequence number indicate dropped packets. They are normally accompanied by a message for each missing packet.

Despite the fact that IP does not guarantee the delivery of packets, a healthy network should drop very few packets. Lost-packet problems are important to track down because they tend to be masked by higher-level protocols. The network may appear to function correctly, but it will be slower than it ought to be, not only because of the retransmitted packets but also because of the protocol overhead needed to detect and manage them.

To track down the cause of disappearing packets, first run **traceroute** (covered next) to discover the route that packets are taking to the target host. Then **ping** the intermediate gateways in sequence to discover which link is dropping packets. To pin down the problem, you need to send a fair number of packets. The fault generally lies on the link between the last gateway you can ping without loss of packets and the gateway beyond that.

The round trip time reported by **ping** gives you insight into the overall performance of a path through a network. Moderate variations in round trip time do not usually indicate problems. Packets may occasionally be delayed by tens or hundreds of milliseconds for no apparent reason. That is just the way IP works. You should expect to see a fairly consistent round trip time for the majority of packets, with occasional lapses. Many of today’s routers implement rate-limited or lower-priority responses to ICMP packets, which means that a router may delay responding to your ping if it is already dealing with a lot of other traffic.

The **ping** program can send echo request packets of any size, so by using a packet larger than the MTU of the network (1,500 bytes for Ethernet), you can force fragmentation. This practice helps you identify media errors or other low-level issues such as problems with a congested network or VPN.
Note that even a simple command like ping can have dramatic effects. In 1998, the so-called Ping of Death attack crashed large numbers of UNIX and Windows systems. It was launched simply by transmission of an overly large ping packet. When the fragmented packet was reassembled, it filled the receiver’s memory buffer and crashed the machine. The Ping of Death issue has long since been fixed, but several other caveats should be kept in mind regarding ping.

First, it is hard to distinguish the failure of a network from the failure of a server with only the ping command. In an environment where ping tests normally work, a failed ping just tells you that something is wrong.

Second, a successful ping does not guarantee much about the target machine’s state. Echo request packets are handled within the IP protocol stack and do not require a server process to be running on the probed host. A response guarantees only that a machine is powered on and has not experienced a kernel panic. You will need higher-level methods to verify the availability of individual services such as HTTP and DNS.

traceroute: Trace IP Packets

traceroute, originally written by Van Jacobson, uncovers the sequence of gateways through which an IP packet travels to reach its destination. The syntax is

$ traceroute hostname

There are a variety of options, most of which are not important in daily use. As usual, the hostname can be specified as either a DNS name or an IP address. The output is simply a list of hosts, starting with the first gateway and ending at the destination. From SPARC1 we have

$ traceroute pan.ces.clemson.edu
traceroute to pan.ces.clemson.edu (130.127.200.19), 30 hops max, 40 byte packets
1 cs-gateway (130.127.48.1) 0.824 ms 0.622 ms 0.739 ms
2 130-127-1-29.generic.clemson.edu (130.127.1.29) 0.534 ms 0.467 ms 0.477 ms
3 130.127.7.38 (130.127.7.38) 0.685 ms 0.696 ms 4.199 ms
4 pan.ces.clemson.edu (130.127.200.19) 0.551 ms 0.537 ms 0.476 ms

From this output, we can tell that pan.ces.clemson.edu is four hops away from sparc1.cs.clemson.edu and we can see which gateways are involved in the connection. The round trip time for each gateway is also shown – three samples for each hop are measured and displayed. A typical traceroute between Internet hosts often includes more than 15 hops, even if the two sites are just across town.

traceroute works by setting the time-to-live field (TTL, actually “hop count to live”) of an outbound packet to an artificially low number. As packets arrive at a gateway, their TTL is decreased. When a gateway decreases the TTL to 0, it discards the packet and sends an ICMP “time exceeded” message back to the original host.

The first three traceroute packets have their TTL set to 1. The first gateway to see such a packet (cs-gateway in the example above) determines that the TTL has been exceeded and notifies SPARC1 of the
dropped packet by sending back an ICMP message. The sender’s IP address in the header of the error packet identifies the gateway, and **traceroute** looks up this address with DNS to find the gateway’s hostname.

To identify the second-hop gateway, **traceroute** sends out a second round of packets with TTL fields set to 2. The first gateway routes the packets and decreases their TTL by 1. At the second gateway, the packets are then dropped, and an ICMP message is generated as before. This process continues until the TTL is equal to the number of hops to the destination host, and the packets reach their destination successfully.

Most routers send their ICMP messages from the interface “closest” to the destination. If you run **traceroute** backward from the destination host, you may see different IP addresses being used to identify the same set of routers. You might also discover that packets flowing in the reverse direction take a completely different path, a configuration known as **asymmetric routing**.

Another **traceroute** example is shown below:

```
$ traceroute google.com
traceroute: Warning: google.com has multiple addresses; using 74.125.45.99
traceroute to google.com (74.125.45.99), 30 hops max, 40 byte packets
1 cs-gateway (130.127.48.1)  0.912 ms  0.698 ms  0.746 ms
2 130.127.0.29 (130.127.0.29)  0.523 ms  0.595 ms  0.444 ms
3 130.127.3.30 (130.127.3.30)  0.819 ms  0.728 ms  0.751 ms
4 poole-border-01.clemson.edu (130.127.3.1)  1.599 ms  1.481 ms  1.267 ms
5 205-186-62-9.generic.c-light.net (205.186.62.9)  1.485 ms  1.603 ms  1.534 ms
6 205-186-63-2.generic.c-light.net (205.186.63.2)  4.410 ms  4.403 ms  4.479 ms
7 74.125.48.33 (74.125.48.33)  4.395 ms  4.307 ms  4.329 ms
8 72.14.239.100 (72.14.239.100)  4.424 ms  126.321 ms  64.233.174.2
   (64.233.174.2)  17.508 ms
   5.232 ms  209.85.254.243 (209.85.254.243)  4.986 ms
   12.428 ms  14.690 ms
11 * * *
12 *^C
```

If we see stars in all the time fields for a given gateway, no “time exceeded” messages are arriving from that machine. Perhaps the gateway is simply down. Sometimes a gateway or firewall is configured to silently discard packets with expired TTLs. Some firewalls block ICMP “time exceeded” messages entirely. If such firewall lies along the path, you will not get information about any of the gateways beyond it. Also some firewalls may block the outbound UDP datagrams that **traceroute** sends to trigger the ICMP responses. This problem causes **traceroute** to report no useful information at all.

**Hostname and IP address assignment**

Administrators have various heartfelt theories about how the mapping from hostnames to IP addresses is best maintained: through the **hosts** file, LDAP, the DNS system, or perhaps some combination of those
options. The conflicting goals are scalability, consistency, and maintainability versus a system that is flexible enough to allow machines to boot and function when not all services are available.

Another consideration you might take into account when designing your addressing system is the possible need to renumber your hosts in the future. Unless you are using private addresses, your site’s IP addresses may change when you switch ISPs. Such a transition becomes daunting if you must visit each host on the network to reconfigure its address. To expedite renumbering, you can use hostnames in configuration files and confine address mappings to a few centralized locations such as the DNS database and your DHCP configuration files.

The `/etc/hosts` file is the oldest and simplest way to map names to IP addresses. Each line starts with an IP address and continues with the various symbolic names by which that address is known. Here is the file from SPARC2:

```
#  # Internet host table  
#  ::1     localhost
127.0.0.1   localhost
130.127.48.2 citron.cs.clemson.edu
130.127.48.6 plato.cs.clemson.edu
130.127.48.252 sparc2.cs.clemson.edu  # Added by DHCP
```

Because `/etc/hosts` contains only local mappings and must be maintained on each client system, it is best reserved for mappings that are needed at boot time, e.g. the host itself, the default gateway, and name servers. Use DNS and LDAP to find mappings for the rest of the local network and the rest of the world. You can also use `/etc/hosts` to specify mappings that you do not want the rest of the world to know about and therefore do not publish in DNS.

**ifconfig: Configure Network Interfaces**

`ifconfig` enables or disables a network interface, sets IP address and subnet mask, and sets various other options and parameters. It is usually run at boot time with command-line parameters taken from config files, but you can also run it by hand to make changes on the fly. Be careful if you are making `ifconfig` changes and are logged in remotely – many a sysadmin has been locked out this way and had to drive in to fix things.

An `ifconfig` command most commonly has the form:

```
    ifconfig interface [family] address options
```

For example, the command

```
    ifconfig eth0 192.168.1.13 netmask 255.255.255.0 up
```

sets the IPv4 address and netmask associated with the interface eth0 and readies the interface for use.
interface identifies the hardware interface to which the command applies. It is usually a two- or three-character name followed by a number. However, Solaris interface names can be longer. On Solaris systems, ifconfig --a lists the system’s network interfaces and summarizes their current settings. From White we have

```bash
$ /sbin/ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,Running,MULTICAST,IPv4,VIRTUAL> mtu 8232
    index 1
    inet 127.0.0.1 netmask ff000000

e1000g0: flags=1004843<UP,BROADCAST,Running,MULTICAST,DHCP,IPv4> mtu 1500 index 2
    inet 172.19.48.213 netmask ffffff00 broadcast 172.19.48.255
    ether 8:0:27:f2:1e:ba
```

The family parameter tells ifconfig which network protocol, i.e. “address family”, you want to configure. You can set up multiple protocols on an interface and use them all simultaneously, but they must be configured separately. The main options here are inet for IPv4 and inet6 for IPv6. inet is assumed if you leave the parameter out.

The address parameter specifies the interface’s IP address. A hostname is also acceptable here, but the hostname must be resolved to an IP address at boot time. For a machine’s primary interface, this means that the hostname must appear in the local hosts file, since other name resolution methods depend on the network having been initialized.

The keyword up turns the interface on, and down turns it off. When an ifconfig command assigns an IP address to an interface, as in the example above, the up parameter is implicit and does not need to be mentioned by name.

The netmask option sets the subnet mask for the interface and is required if the network is not subnetted according to its address class. The mask can be specified in dotted decimal notation or as a 4-byte hexadecimal number beginning with 0x. As usual, bits set to 1 are part of the network number, and bits set to 0 are part of the host number.

route: Configure Static Routes

The route command defines static routes, explicit routing table entries that never change, even if you run a routing daemon. When you add a new machine to a local area network, you usually need to specify only a default route.

Routing is performed at the IP layer. When a packet bound for some other host arrives, the packet’s destination IP address is compared with the route in the kernel’s routing table. If the address matches a route in the table, the packet is forwarded to the “next-hop gateway” IP address associated with that route.

There are two special cases. First, a packet may be destined for some host on a directly connected network. In this case, the “next-hop gateway” address in the routing table is one of the local host’s own interfaces, and the packet is sent directly to its destination. This type of route is added to the routing table for you by the ifconfig command when you configure an interface.
Second, it may be that no route matches the destination address. In this case, the default route is invoked if one exists. Otherwise, an ICMP “network unreachable” or “host unreachable” message is returned to the sender.

Many local area networks have only one way out, so all they need is a single default route that points to the exit. On the Internet backbone, the routers do not have default routes. If there is no routing entry for a destination, that destination cannot be reached.

Each `route` command adds or removes one route. A typical command is

```
    # route add -net 192.168.45.128/25 zulu-gw.atrust.net
```

This command adds a route to the 192.168.45.128/25 network through the gateway router zulu-gw.atrust.net, which must be either an adjacent host or one of the local host's own interfaces.

Solaris has a nifty `-p` option to `route` that makes your changes persistent across reboots. In addition to being entered in the kernel's routing tables, the changes are recorded in `/etc/inet/static_routes` and restored at boot time.

**DNS Configuration**

To configure a machine as a DNS client, you need only set up the `/etc/resolv.conf` file. DNS service is not, strictly speaking, required, but it is hard to imagine a situation in which you would want to eliminate it completely.

The `resolv.conf` file lists the DNS domains that should be searched to resolve names that are incomplete, i.e. that are not fully qualified, such as sparc1 instead of sparc1.cs.clemson.edu and the IP address of the name servers to contact for name lookups. A sample file from SPARC1 is

```
domain cs.clemson.edu
nameserver 130.127.255.250
nameserver 130.127.255.251
```

`/etc/resolv.conf` should list the “closest” stable name server first. Servers are contacted in order, and the timeout after which the next server in line is tried can be quite long. You can have up to three `nameserver` entries. If possible you should always have more than one.

If the local host obtains the addresses of its DNS servers through DHCP, the DHCP client software stuffs these addresses into the `resolv.conf` file for you when it obtains the leases. Since DHCP configuration is the default for most systems, you generally do not need to configure the `resolv.conf` file manually if your DHCP server has been set up correctly.