

UNIX network analysis

Understanding your UNIX system network configuration

Skill Level: Intermediate

[Martin Brown \(mc@mcslp.com\)](mailto:mc@mcslp.com)

Professional writer
Freelance

05 May 2009

You can find out a lot about your network by using a variety of different tools. If you want to understand the layout of your network, where packets are going, and what people are doing, then you need to use a variety of different tools that can help you to build up a picture of your network and what is going on. This tutorial examines techniques for monitoring the traffic and content of your UNIX® network and how to read and diagnose problems on your network.

Section 1. Before you start

This tutorial is for UNIX systems administrators who are looking for ways to discover and determine information about their network structure and configuration, including what services and systems are running on different machines. To get the best out of this tutorial, you should have a basic knowledge of the UNIX operating system, and a basic understanding of how networks and the Internet protocol (IP) operate.

About this tutorial

When accessing a new UNIX system, or even understanding an existing one, a key part of the puzzle to how the system operates is the network configuration. There are many aspects of the network that you need to know and understand to correctly identify problems and prevent future problems. By using some basic tools and

commands you can determine a lot about the configuration of a single system, and through this basic understanding, a good idea of the configuration of the rest of the network. With some additional tools, you can expand that knowledge to cover more systems and services within your network.

In this tutorial you will use some basic tools within the UNIX environment that can disclose information about the configuration of your system. By understanding these tools and the information they output, you will be able to gain a greater understanding of your system network configuration and how it works. You will also examine tools and solutions that can look at the wider network and gain more detailed information about your network, its potential security issues, and key points of information that will help you identify and diagnose problems when they do occur.

Section 2. Understanding networks on the host

The first step to understanding your network better is to understand the network configuration of the machine you are currently using. This will give you a number of frames of reference, such as the IP address of the current host, the DNS configuration, and what other machines you can connect to and communicate with.

Finding configuration information

Determining the current configuration of the machine you are working on gives you the base information about your environment. Your first task is to determine the IP address and network mask for the current machine. By using these two values, you can determine the address of your machine and what other machines you can connect to directly on your network (for instance, without the use of a router).

Before you determine the IP address, get the hostname for the system by using the `hostname` command (see Listing 1).

Listing 1. Getting the hostname

```
$ hostname  
sulaco
```

The `ifconfig` command will display the current configuration information for all your configured network devices when you use the `-a` option. For example, Listing 2 shows the output from the `ifconfig` command on a Solaris machine.

Listing 2. Output from ipconfig on Solaris

```
$ ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
pcn0: flags=201004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,CoS>
mtu 1500 index 2
    inet 192.168.1.25 netmask fffffc00 broadcast 192.168.3.255
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
    inet6 ::1/128
pcn0: flags=202004841<UP,RUNNING,MULTICAST,DHCP,IPv6,CoS> mtu 1500 index 2
    inet6 fe80::20c:29ff:fe7f:dc5/10
```

You can see from this output that there is a loopback device, lo0, with the normal address of 127.0.0.1 for localhost. You can also see that the same device also has an equivalent IPv6 address.

The pcn0 device is configured with a network address of 192.168.1.25, and with a netmask of fffffc00, equivalent to 255.255.252.0. You can also see that in this case the address was set using DHCP (from the list of DHCP flags).

The netmask is particularly important, because with the netmask alone you can tell the size (in terms of registered IP addresses) of your immediate network. In this case, 255.255.252.0 equates to four class C addresses, because 256 (the maximum number of hosts) minus 252 (the number of masked hosts) equals four.

By combining the netmask with the configured IP address, you can guess the range of the IP addresses in the local network. Because IP blocks are usually split by whole groups and in sequence, you can tell that the IP address span of the network is 192.168.0.0 through 192.168.3.255. You can determine this because with a netmask of four class C addresses you would normally split the entire range (192.168.0.0-192.168.255.255) into equal blocks -- with the address prefix of 192.168.1.x it must be in the first block of four addresses.

Different operating systems output the information (and the detail) in different ways. Listing 3 shows the output from a Linux® system.

Listing 3. Output on a Linux system

```
eth0      Link encap:Ethernet  HWaddr 00:1d:60:1b:9a:2d
          inet addr:192.168.0.2  Bcast:192.168.3.255  Mask:255.255.252.0
          inet6 addr: fe80::21d:60ff:fe1b:9a2d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2371085881  errors:36  dropped:0  overruns:0  frame:36
          TX packets:2861233776  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:913269364222 (850.5 GiB)  TX bytes:3093820025338 (2.8 TiB)
          Interrupt:23 Base address:0x4000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
```

```
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:279755697 errors:0 dropped:0 overruns:0 frame:0
TX packets:279755697 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:388038389807 (361.3 GiB) TX bytes:388038389807 (361.3 GiB)
```

Listing 4 shows the output from a Mac OS X™ system.

Listing 4. Output from a Mac OS X system

```
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.0.101 netmask 0xfffffc00 broadcast 192.168.3.255
    ether 00:16:cb:a0:3b:cb
    media: autoselect (1000baseT <full-duplex,flow-control>) status: active
    supported media: autoselect 10baseT/UTP <half-duplex> 10baseT/UTP
    <full-duplex> 10baseT/UTP <full-duplex,hw-loopback> 10baseT/UTP
    <full-duplex,flow-control> 100baseTX <half-duplex> 100baseTX
    <full-duplex> 100baseTX <full-duplex,hw-loopback> 100baseTX
    <full-duplex,flow-control> 1000baseT <full-duplex> 1000baseT
    <full-duplex,hw-loopback> 1000baseT <full-duplex,flow-control> none
fw0: flags=8822<BROADCAST,SMART,SIMPLEX,MULTICAST> mtu 2030
    lladdr 00:17:f2:ff:fe:7b:84:d6
    media: autoselect <full-duplex> status: inactive
    supported media: autoselect <full-duplex>
en1: flags=8822<BROADCAST,SMART,SIMPLEX,MULTICAST> mtu 1500
    ether 00:17:f2:9b:3d:38
    media: autoselect (<unknown type>)
    supported media: autoselect
en5: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::21c:42ff:fe00:8%en5 prefixlen 64 scopeid 0x7
    inet 10.211.55.2 netmask 0xffffffff00 broadcast 10.211.55.255
    ether 00:1c:42:00:00:08
    media: autoselect status: active
    supported media: autoselect
en6: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::21c:42ff:fe00:9%en6 prefixlen 64 scopeid 0x8
    inet 10.37.129.2 netmask 0xffffffff00 broadcast 10.37.129.255
    ether 00:1c:42:00:00:09
    media: autoselect status: active
    supported media: autoselect
```

In all cases, you can generally find the Internet address and netmask of the connected network devices. Obviously, if you have multiple network devices then you will get the information for each device in the output, and it may be that you can reach a wide range of different networks and systems from just one machine.

Finding name resolution services

Your next step in determining the configuration of the current machine should relate to the configuration of the name service system that will convert name and domain names on your system into an IP address when you access a service on another

machine.

The configuration of this on most machines is through the `/etc/nsswitch.conf` file, which contains a list of different naming services (hosts, users, and more) and the order in which the different services (DNS, NIS, or local files) should be used for resolution. You can see an example of this in Listing 5.

Listing 5. Resolving the name service system

```
passwd:      files
group:       files
hosts:       files dns
ipnodes:     files dns
networks:    files
protocols:   files
rpc:         files
ethers:      files
netmasks:    files
bootparams:  files
publickey:   files
netgroup:    files
automount:   files
aliases:     files
services:    files
printers:    user files
auth_attr:   files
prof_attr:   files
project:     files
tnrhttp:     files
tnrhdb:      files
```

In Listing 5, for example, the hostname information is resolved first by looking at the local files on the system (for example, `/etc/hosts`) and then the domain name system (DNS).

If the DNS has been configured, then the `/etc/resolv.conf` file will tell you which machines are being used to convert names into IP addresses. A sample of the file is shown here in Listing 6.

Listing 6. Which machines are being used to convert names into IP addresses

```
domain example.pri
nameserver 192.168.0.2
nameserver 192.168.0.3
```

This information can be useful if you want to query these machines directly for information. You can use tools such as `dig` and `nslookup` to extract information about the name service and resolution of names and IP addresses.

Checking routes

Hosts outside of your network (that is, beyond the scope of your network mask in comparison to your current IP address) are sent to a router to be forwarded on to another machine. Routers can be used at all levels of your network, including between departments, different physical sites, and to public and external sites such as the Internet.

The `netstat` command can tell you which machines or routers are contacted when your machine wants to communicate with machines outside the 'local' network. For example, Listing 7, below, is from a Solaris machine.

Listing 7. `netstat` command

```
$ netstat -r
```

```
Routing Table: IPv4
Destination          Gateway              Flags  Ref    Use      Interface
-----
default              voyager.example.pri UG      1    139    pcn0
192.168.0.0          solaris2.example.pri U       1    447    pcn0
solaris2             solaris2             UH      1     35    lo0

Routing Table: IPv6
Destination/Mask      Gateway              Flags  Ref    Use      If
-----
fe80::/10            fe80::20c:29ff:fe7f:dc5 U       1       0    pcn0
solaris2             solaris2             UH      1       0    lo0
```

The default route shows the gateway (router) used to route packets that are either outside of the current network, or that are not already covered by another route for a specific IP address or IP address range.

Because you might need to determine this information in a situation where your current nameservice is not working, or not returning the right information, you can also specify the `-n` option to show the information using IP addresses instead of names.

Checking supported services

The `netstat` command can also be used to determine what services are being shared and exposed on the current host. This includes all network services, including DNS, NFS, Web services, and other information. The information displayed is based upon the ports that are open and in the 'listening' state waiting for client connections, or ports that are already open and communicating with a client.

This information can prove invaluable, both to determine if a service is running, and as part of a standard security check to determine whether a machine is sharing or exposing itself to more risk than is necessary.

You can see an example of the output in Listing 8, here using `-a` to display all the

open ports and services, both established (open) and listening for new connections. By default, netstat also shows the open UNIX domain sockets, which are only accessible to the current machine. For brevity these have been removed from the output.

Listing 8. Output using -a

```
$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 *:imaps                 *:                       LISTEN
tcp      0      0 *:nfs                   *:                       LISTEN
tcp      0      0 *:vmware-authd         *:                       LISTEN
tcp      0      0 localhost:10024         *:                       LISTEN
tcp      0      0 localhost:10025         *:                       LISTEN
tcp      0      0 *:mysql                 *:                       LISTEN
tcp      0      0 *:imap                 *:                       LISTEN
tcp      0      0 localhost:783          *:                       LISTEN
tcp      0      0 *:sunrpc                *:                       LISTEN
tcp      0      0 bear.example.pri:http   *:                       LISTEN
tcp      0      0 *:cisco-sccp           *:                       LISTEN
tcp      0      0 *:47506                 *:                       LISTEN
tcp      0      0 *:34452                 *:                       LISTEN
tcp      0      0 172.16.217.1:domain    *:                       LISTEN
tcp      0      0 192.168.92.1:domain    *:                       LISTEN
tcp      0      0 bear.example.pri:domain *:                       LISTEN
tcp      0      0 localhost:domain       *:                       LISTEN
tcp      0      0 *:53941                 *:                       LISTEN
tcp      0      0 *:3128                  *:                       LISTEN
tcp      0      0 localhost:rndc          *:                       LISTEN
tcp      0      0 *:smtp                  *:                       LISTEN
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65452   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65459   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65412   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65417   ESTABLISHED
tcp      0      0 bear.example.pri:mysq   bear.example.pri:35475   TIME_WAIT
tcp      0      0 bear.example.pri:http   sulaco.example.p:49603   FIN_WAIT2
tcp      0      0 bear.example.pri:nfs    sulaco.example.p:49552   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65433   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65431   ESTABLISHED
tcp      0      0 bear.example.pri:nfs    sulaco.example.p:51900   CLOSE_WAIT
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65415   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65475   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65472   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65429   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65430   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65438   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65443   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65418   ESTABLISHED
tcp      0      0 bear.example.pri:nfs    narcissus.exempl:62968   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65448   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65423   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65468   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65445   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65476   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65453   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65456   ESTABLISHED
tcp      0      0 bear.example.pri:nfs    sulaco.example.p:59172   CLOSE_WAIT
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65416   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65439   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65441   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65446   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65470   ESTABLISHED
tcp      0      0 bear.example.pri:imap   sulaco.example.p:65450   ESTABLISHED
tcp      0      0 bear.example.pri:nfs    sulaco.example.p:65320   ESTABLISHED
```



```

tcp      0      0 bear.example.pri:imap      sulaco.example.p:65465    ESTABLISHED
tcp      0      0 bear.example.pri:36230     solaris2.vmbear.mcs:ssh   ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65421    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65464    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65474    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:64955    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65473    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65461    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65454    ESTABLISHED
tcp      0      0 bear.example.pri:http      sulaco.example.p:49608    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65471    ESTABLISHED
tcp      0      0 localhost:50123            localhost:ssh              ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65420    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65466    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65463    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65451    ESTABLISHED
tcp      0      0 bear.example.pri:35471     bear.example.pri:mysql    TIME_WAIT
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65457    ESTABLISHED
tcp      1      0 bear.example.pri:nfs        sulaco.example.p:53877    CLOSE_WAIT
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65432    ESTABLISHED
tcp      0      0 bear.example.pri:mysql      bear.example.pri:35470    TIME_WAIT
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65467    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65414    ESTABLISHED
tcp      0      0 bear.example.pri:50112     bear.example.pri:imap      TIME_WAIT
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65462    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65460    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65469    ESTABLISHED
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65422    ESTABLISHED
tcp      0      0 bear.example.pri:50110     bear.example.pri:imap      TIME_WAIT
tcp      0      0 bear.example.pri:50111     bear.example.pri:imap      TIME_WAIT
tcp      0      0 bear.example.pri:imap      sulaco.example.p:65442    ESTABLISHED
tcp6     0      0 [::]:imaps                 [::]:*                     LISTEN
tcp6     0      0 [::]:11211                  [::]:*                     LISTEN
tcp6     0      0 [::]:imap                    [::]:*                     LISTEN
tcp6     0      0 [::]:cisco-sccp              [::]:*                     LISTEN
tcp6     0      0 [::]:ssh                      [::]:*                     LISTEN
tcp6     0      0 localhost:rndc                [::]:*                     LISTEN
tcp6     0      0 [::]:https                   [::]:*                     LISTEN
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:52786    ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:56220    ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:63895    ESTABLISHED
tcp6     0      0 localhost:ssh                 localhost:50123            ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:60914    ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:64669    ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:56053    ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:52268    ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:49528    ESTABLISHED
tcp6     0      0 bear.example.pri:ssh          sulaco.example.p:65408    ESTABLISHED
udp      0      0 *:nfs                         *:*                         *:*
udp      0      0 *:42498                      *:*                         *:*
udp      0      0 *:54680                      *:*                         *:*
udp      0      0 172.16.217.1:domain          *:*                         *:*
udp      0      0 192.168.92.1:domain          *:*                         *:*
udp      0      0 bear.example.p:domain         *:*                         *:*
udp      0      0 localhost:domain              *:*                         *:*
udp      0      0 *:45495                      *:*                         *:*
udp      0      0 *:icpv2                      *:*                         *:*
udp      0      0 *:bootps                     *:*                         *:*
udp      0      0 *:964                        *:*                         *:*
udp      0      0 *:11211                      *:*                         *:*
udp      0      0 *:sunrpc                      *:*                         *:*
udp      0      0 *:50042                      *:*                         *:*
raw      0      0 *:icmp                       *:*                         *:*

```

7

As you can see from this output, the machine is quite busy. The third column shows the hostname and port, separated by a colon, for each open connection or listening

connection. If the TCP or UDP service number matches a known port number (as defined within the `/etc/services` file), then the service name is displayed in the output. For the host, either the host's name, an alternative IP address, or the '*' symbol is displayed. The asterisk indicates that the service and ports are open and listening on all IP addresses.

For example, you can tell from this output that the machine is configured to support NFS, and has open (established) connections, as shown in Listing 9.

Listing 9. Machine is configured to support NFS

```
$ netstat -a|grep nfs
tcp        0      0 *:nfs                 :::*                    LISTEN
tcp        1      0 bear.example.pri:nfs  sulaco.example.p:51900 CLOSE_WAIT
tcp        0      0 bear.example.pri:nfs  narcissus.example.p:62968 ESTABLISHED
tcp        1      0 bear.example.pri:nfs  sulaco.example.p:59172 CLOSE_WAIT
tcp        0      0 bear.example.pri:nfs  sulaco.example.p:65320 ESTABLISHED
tcp        1      0 bear.example.pri:nfs  sulaco.example.p:53877 CLOSE_WAIT
udp        0      0 *:nfs                 :::*                    *
```

It is also possible using this output to see which machines are currently communicating with this machine. For example, you can extract a list of the machines connected to this one by looking at the fifth column, and then sorting and removing duplicates from the list (see Listing 10).

Listing 10. Extracting a list of connected machines

```
$ netstat -a|egrep 'tcp|udp'|grep ESTABLISHED|awk '{ print $5; }'|cut -d: -f1|sort|uniq
localhost
narcissus.mcslp.p
nautilus.wireless
polarbear.wireless
solaris2.vmbear.mcs
sulaco.mcslp.pri
```

This can be useful when you suspect there is a user or computer connected to the machine that you do not recognize or don't expect.

To find out about these other machines, you need to start looking at the other computers within your network.

Section 3. Finding information about other hosts

Once you have the basic information about your machine, you can start to spread out and look at other machines in your network to determine the available and

services that they provide. With the right tools, you can even try to determine what operating system these machines are running and what services the machines might be sharing.

Checking hosts

The easiest and most obvious tool for checking remote machines is to use the ping tool to check whether a particular host is up and available. The ping tool does something very simple. It sends a packet to the remote host requesting a response. When the response has been received, the ping tool calculates the time difference, and the time taken to send and receive the packet can be used as an indication of how near or far a machine is from its current location.

For example, if you ping a machine on your own network, you are likely to get a response to the ping packet very quickly (see Listing 11).

Listing 11. Pinging machine on your own network

```
$ ping bear
PING bear.mcslp.pri (192.168.0.2): 56 data bytes
64 bytes from 192.168.0.2: icmp_seq=0 ttl=64
time=0.154 ms
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64
time=0.162 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64
time=0.149 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64
time=0.161 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64
time=0.162 ms
64 bytes from 192.168.0.2: icmp_seq=5 ttl=64
time=0.161 ms
^C
--- bear.mcslp.pri ping statistics ---
6 packets transmitted, 6 packets received, 0% packet
loss
round-trip min/avg/max/stddev =
0.149/0.158/0.162/0.005 ms
```

Different implementations of the ping tool work in different ways. By default on Linux and Mac OS X, the tool will continually send packets and wait for a response until you force the application to terminate with Control-C.

On Solaris™, AIX®, and some other UNIX variants, without any additional arguments the ping tools will merely indicate if the remote host responded (see Listing 12).

Listing 12. Pinging without additional arguments on UNIX variants

```
$ ping bear
bear is alive
```

To perform the longer test, use the `-s` option, shown in Listing 13.

Listing 13. Using the `-s` option for pinging

```
$ ping -s bear
PING bear: 56 data bytes
64 bytes from bear.mcslp.pri (192.168.0.2): icmp_seq=0. time=0.288 ms
64 bytes from bear.mcslp.pri (192.168.0.2): icmp_seq=1. time=0.247 ms
64 bytes from bear.mcslp.pri (192.168.0.2): icmp_seq=2. time=0.208 ms
64 bytes from bear.mcslp.pri (192.168.0.2): icmp_seq=3. time=0.230 ms
^C
---bear PING Statistics---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev = 0.208/0.243/0.288/0.034
```

The time field for each line gives you an indication of the speed and latency (the delay before response, and often an indication of the level of activity) for each packet. When you stop the output, you get a summary of the number of packets sent, received, and the time statistics.

The further the distance that the ping packets have to travel, the longer the response time from the remote host. For example, if you try to ping a public server on the Internet, the time taken for the response packet can be significantly higher (see Listing 14).

Listing 14. Pinging a public server on the Internet

```
$ ping www.example.com
PING www.example.com (67.205.21.169) 56(84) bytes of data.
64 bytes from mcslp.com (67.205.21.169): icmp_seq=1 ttl=44 time=193 ms
64 bytes from mcslp.com (67.205.21.169): icmp_seq=2 ttl=44 time=194 ms
64 bytes from mcslp.com (67.205.21.169): icmp_seq=3 ttl=44 time=197 ms
64 bytes from mcslp.com (67.205.21.169): icmp_seq=4 ttl=44 time=194 ms
^C
--- www.example.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3039ms
rtt min/avg/max/mdev = 193.737/195.120/197.123/1.353 ms
```

Compare the times for this connection to an Internet service (193ms) with the time for a local host (0.23ms).

The ping tool can also be a quick way of determining whether you can even reach the remote host that you want to connect to. Running ping on a host that does not exist returns a very specific error (see Listing 15).

Listing 15. Pinging a host that does not exist

```
$ ping notinhere
PING notinhere (192.168.0.110) 56(84) bytes of data.
```

```
>From bear.mcslp.pri (192.168.0.2) icmp_seq=1 Destination Host Unreachable
>From bear.mcslp.pri (192.168.0.2) icmp_seq=2 Destination Host Unreachable
>From bear.mcslp.pri (192.168.0.2) icmp_seq=3 Destination Host Unreachable
^C
--- notinhere ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4039ms
```

The ping tool relies on knowing what other machines are available on your network. Let's see how you can determine what hosts might be on the network without knowing their names or IP addresses

Discovering hosts on your network

Within the Ethernet network system (and others), all devices on your network have a unique address associated with the hardware network device. The Media Access Control (MAC) number uniquely identifies the network device and, through the use of higher-level protocols such as the Internet Protocol you can associate the MAC address with the host name.

This is used (in reverse) by the operating system when sending packets out on the network. When you send packets to a specific hostname, the operating system attempts to resolve the hostname into a MAC address so it can construct the hardware (Ethernet) packet to be sent out on the network.

The Address Resolution Protocol (ARP) handles this mapping, and you can use the arp tool to display the currently held information about the hosts and their host names or IP addresses.

Because any machine on the network that wants to communicate with another must have sent out a packet with the MAC address and the IP address, the information gleaned by your system in the ARP cache can be a useful way to find out what other machines are on the network (see Listing 16).

Listing 16. Using the arp command

```
$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
gendarme.mcslp.pri ether    00:1B:2F:F0:39:6A C             eth0
narcissus.mcslp.pri ether    00:16:CB:85:2D:15 C             eth0
solaris2.vmbear.mcslp.p ether    00:0C:29:7F:0D:C5 C             eth0
nautilus.wireless.mcslp ether    00:17:F2:40:4D:1B C             eth0
sulaco.mcslp.pri ether    00:16:CB:A0:3B:CB C             eth0
```

With modern Ethernet switches, in place of the older hub structure, the information output by arp may be limited to the packets sent and received to or from a particular host. If you can run arp on a server you will get a longer list of information, but this isn't always possible or practical.

On some network switches you have a network management or monitoring port where all packets are echoed, and which you can use to gain information about the other network devices and therefore the network structure. If you don't have access to this information, you may need a more brute force approach to finding hosts on your network.

Finding other hosts on your network

The nmap tool is a utility that can perform a variety of different scans across your network to find and determine different levels of information. At a basic level, it can be used to find all of the hosts within a given network.

Earlier the article examined how to get the current IP address and netmask information for a host. You can use this information to set the basic search parameters for nmap to try and find all of the hosts on the network. To specify this information, you have to use the CIDR style addresses. The CIDR format uses the IP address of the host, and the number of bits in the network mask, to determine the span of the network.

From the example host, 192.168.1.25 was the IP address, and the network mask was 255.255.252.0. This is equivalent to 22 bits -- 8 bits for the first part, 8 bits for the second, and 6 bits for the third part.

Running nmap with this address will scan every single IP address within the range (for instance, every address between 192.168.0.0 and 192.168.3.255) and determine which hosts reply.

You can perform a number of different tests, including a test using the standard ping protocol, or a more extensive test that tries other network ports in case the ping protocol has been disabled. For example, the ping test displays the list of hosts in Listing 17.

Listing 17. Running nmap to scan range of IP addresses

```
$ nmap -sP 192.168.1.25/22

Starting Nmap 4.76 ( http://nmap.org ) at 2009-03-24 15:59 GMT
Host 192.168.0.1 appears to be up.
Host bear.mcslp.pri (192.168.0.2) appears to be up.
Host narcissus.mcslp.pri (192.168.0.3) appears to be up.
Host 192.168.0.10 appears to be up.
Host 192.168.0.27 appears to be up.
Host sulaco.mcslp.pri (192.168.0.101) appears to be up.
Host nautilus.wireless.mcslp.pri (192.168.0.109) appears to be up.
Host 192.168.1.1 appears to be up.
Host 192.168.1.25 appears to be up.
Host gentool.vmbear.mcslp.pri (192.168.1.52) appears to be up.
Host gentoo2.vmbear.mcslp.pri (192.168.1.53) appears to be up.
Nmap done: 1024 IP addresses (11 hosts up) scanned in 5.78 seconds
```

The ping check can give you a very quick idea of what other machines are on the network. In this case, 11 hosts have been discovered, but not all of them can be resolved back to a name. This is a fault in the DNS configuration that should be fixed, as some systems use the reverse lookup (IP address to name) as a security check to ensure the client IP address has not been faked.

Finding other services on your network

The ping check is useful, but if you want to know the services an individual machine is actually exposing itself to, use the TCP check. A TCP check takes longer, as nmap will try to open ports using the TCP/IP protocol from each host within the list. This can be more effective at displaying what hosts are on your network, and at providing detailing information about the open ports for each host. You can see this in Listing 18.

Listing 18. Using a TCP check

```
$ nmap -sT 192.168.1.25/22

Starting Nmap 4.76 ( http://nmap.org ) at 2009-03-24 16:03 GMT
Interesting ports on 192.168.0.1:
Not shown: 997 closed ports
PORT      STATE SERVICE
80/tcp    open  http
8080/tcp  open  http-proxy
49153/tcp open  unknown

Interesting ports on bear.mcslp.pri (192.168.0.2):
Not shown: 987 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
143/tcp   open  imap
443/tcp   open  https
902/tcp   open  iss-realsecure
993/tcp   open  imaps
2000/tcp  open  callbook
2049/tcp  open  nfs
3128/tcp  open  squid-http
3306/tcp  open  mysql

Interesting ports on narcissus.mcslp.pri (192.168.0.3):
Not shown: 982 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
88/tcp    open  kerberos-sec
106/tcp   open  pop3pw
111/tcp   open  rpcbind
311/tcp   open  asip-webadmin
389/tcp   open  ldap
548/tcp   open  afp
625/tcp   open  apple-xsrvr-admin
749/tcp   open  kerberos-adm
1021/tcp  open  unknown
```

```
1022/tcp open  unknown
3659/tcp open  unknown
3689/tcp open  rendezvous
4111/tcp open  unknown
5900/tcp open  vnc
8086/tcp open  unknown
8087/tcp open  unknown

Interesting ports on 192.168.0.10:
Not shown: 997 closed ports
PORT      STATE SERVICE
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https

Interesting ports on 192.168.0.27:
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Interesting ports on sulaco.mcslp.pri (192.168.0.101):
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
88/tcp    open  kerberos-sec
548/tcp   open  afp
631/tcp   open  ipp
2170/tcp  open  unknown

Interesting ports on nautilus.wireless.mcslp.pri (192.168.0.109):
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
88/tcp    open  kerberos-sec
111/tcp   open  rpcbind
1001/tcp  open  unknown
5900/tcp  open  vnc

Interesting ports on 192.168.1.1:
Not shown: 995 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
5431/tcp  open  unknown

Interesting ports on 192.168.1.25:
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
4045/tcp  open  lockd

Interesting ports on gentool.vmbear.mcslp.pri (192.168.1.52):
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
3128/tcp  open  squid-http

Interesting ports on gentoo2.vmbear.mcslp.pri (192.168.1.53):
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind

Nmap done: 1024 IP addresses (11 hosts up) scanned in 32.27 seconds
```


From this output, you can see that there are a number of servers on the network providing a variety of services. The device at 192.168.0.1, for example, provides HTTP and HTTP-proxy services. Sp does bear.mcslp.pri, in addition to smtp imap, nfs, and MySQL services.

To determine more specific information about these services, you can use nmap again with the version argument to get a more specific list of the version information about the protocols and ports that are open on a specific host.

For example, checking what appears to be the main server (bear), you can get a very good idea of exactly what is running behind each of these ports (see Listing 19).

Listing 19. Using nmap with the version argument

```
$ nmap -sT -sV bear

Starting Nmap 4.76 ( http://nmap.org ) at 2009-03-24 16:17 GMT
Interesting ports on localhost (127.0.0.1):
Not shown: 985 closed ports
PORT      STATE SERVICE          VERSION
22/tcp    open  ssh              OpenSSH 5.1 (protocol 2.0)
25/tcp    open  smtp             Postfix smtpd
53/tcp    open  domain          ISC BIND 9.4.3-P1
111/tcp   open  rpcbind
143/tcp   open  imap            Cyrus IMAP4 2.3.13-Gentoo
443/tcp   open  ssl/http        Apache httpd
783/tcp   open  spamassassin    SpamAssassin spamd
902/tcp   open  ssl/vmware-auth VMware Authentication Daemon 1.10 (Uses VNC)
993/tcp   open  ssl/imap        Cyrus imapd
2000/tcp  open  sieve           Cyrus timsieved 2.3.13-Gentoo (included w/cyrus imap)
2049/tcp  open  rpcbind
3128/tcp  open  http-proxy      Squid webproxy 2.7.STABLE6
3306/tcp  open  mysql           MySQL 5.0.60-log
10024/tcp open  smtp            amavisd smtpd
10025/tcp open  smtp            Postfix smtpd
Service Info: Hosts: gendarme.mcslp.com, bear, 127.0.0.1

Service detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.12 seconds
```

In this case, you can see a number of specific services, this time showing the version and even application information that is provided in each case.

Determining unidentified hosts on your network

When you have found a host on your network, especially one that you may not immediately recognize, you may want to know more about the host. The TCP port scan shows you what services are being supported by the host, but this may not necessarily tell you the whole story. Some devices and systems may or may not expose ports in a manner that doesn't make it immediately obvious what is on your network.

The nmap operating system scan examines the open ports and tries to work out what the system is behind the different services. This can make the difference between identifying a server with open ports and a new device on your network.

For example, if you run the operating system identification on the server bear, you can identify the system as running a traditional version of Linux, which probably indicates a standard computer, as shown in Listing 20.

Listing 20. nmap operating system scan

```
# nmap -sT -O bear

Starting Nmap 4.76 ( http://nmap.org ) at 2009-03-24 16:20 GMT
Interesting ports on localhost (127.0.0.1):
Not shown: 985 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
53/tcp    open  domain
111/tcp   open  rpcbind
143/tcp   open  imap
443/tcp   open  https
783/tcp   open  spamassassin
902/tcp   open  iss-realsecure
993/tcp   open  imaps
2000/tcp  open  callbook
2049/tcp  open  nfs
3128/tcp  open  squid-http
3306/tcp  open  mysql
10024/tcp open  unknown
10025/tcp open  unknown
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17 - 2.6.25
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.71 seconds
```

The OS scan is not perfect, and it relies on finger printing techniques to determine what the open ports and returned version information means. For example, the scan below in Listing 21 has identified a number of potential operating systems that might be behind the port types.

Listing 21. Scan indentifying a number of potential operating systems

```
# nmap -sT -O some.faroffhost.com

Starting Nmap 4.76 ( http://nmap.org ) at 2009-03-24 16:23 GMT
Interesting ports on some.faroffhost.com (205.196.217.20):
Not shown: 976 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
110/tcp   open  pop3
111/tcp   filtered rpcbind
113/tcp   open  auth
```

```

135/tcp    filtered msrpc
139/tcp    filtered netbios-ssn
143/tcp    open      imap
548/tcp    open      afp
554/tcp    open      rtsp
555/tcp    open      dsf
587/tcp    open      submission
687/tcp    open      unknown
993/tcp    open      imaps
995/tcp    open      pop3s
1720/tcp   filtered H.323/Q.931
5222/tcp   open      unknown
5269/tcp   open      unknown
5666/tcp   open      unknown
7070/tcp   open      realserver
8000/tcp   open      http-alt
8001/tcp   open      unknown
8649/tcp   open      unknown
Device type: print server|general purpose|storage-misc|WAP|switch|specialized
Running (JUST GUESSING) : HP embedded (92%), Linux 2.6.X|2.4.X (92%), Buffalo embedded
(91%), Acorp embedded (89%), Actiontec Linux 2.4.X (89%), Linksys embedded (89%),
Netgear embedded (89%), Infoblox NIOS 4.X (89%)
Aggressive OS guesses: HP 4200 PSA (Print Server Appliance) model J4117A (92%),
Linux 2.6.20 (Ubuntu 7.04 server, x86) (92%), Linux 2.6.9 (92%), Buffalo TeraStation NAS
device (91%), Linux 2.6.18 (CentOS 5.1, x86) (91%), OpenWrt 7.09 (Linux 2.4.34) (90%),
Acorp W400G or W422G wireless ADSL modem (MontaVista Linux 2.4.17) (89%), HP Brocade
4100 switch; or Actiontec MI-424-WR, Linksys WRVS4400N, or Netgear WNR834B wireless
broadband router (89%), HP Brocade 4Gb SAN switch (89%), Infoblox NIOS Release
4.1r2-5-22263 (89%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 18 hops

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.66 seconds

```

It is worth noting that the nmap scan can be used on both local and remote networks. In the remote test above, nmap determined how different systems the packets had to go through before they reached their destination. Understanding the different devices between you and other machines on your network are often the final part of understanding and determining your network layout.

Determining the network structure

Whenever an IP network packet is sent out on the network, a special counter is incremented each time a system forwards the packets on to another network or system. Forwarding of packets happens within a number of different systems. If you have multiple network switches connected together, each hub can identify itself as a new device. In addition, wireless access points and traditional routers are all examples of devices that forward packets and are therefore considered part of the network route of the packet.

In most network environments, hubs, switches, and other components within your local network do not increment this value, but as you stretch out wider across your network, the network gets larger and more complex, so understanding the route taken by individual packets can help you to identify performance and connectivity problems.

The primary tool for displaying the route information for communicating with a host is traceroute. This determines the IP address of each host within a given path from the current host to the destination. If the host is immediately local, then the route is obviously direct (see Listing 22).

Listing 22. Using traceroute

```
$ traceroute solaris2
traceroute to solaris2 (192.168.1.25), 30 hops max, 40 byte packets
 1  solaris2.mcslp.pri (192.168.1.25)  0.651 ms  0.892 ms  0.969 ms
```

For hosts in the local network that might be accessible through a local router or bridge, see Listing 23.

Listing 23. Hosts in the local network

```
$ traceroute gentool
traceroute to gentool (192.168.1.52), 30 hops max, 40 byte packets
 1  gendarme.mcslp.pri (192.168.0.1)  3.163 ms  3.159 ms  6.618 ms
 2  gentool.mcslp.pri (192.168.1.52)  34.336 ms  34.341 ms  34.341 ms
```

Connections to distant networks may show each router and step the packets have taken (see Listing 24).

Listing 24. Connections to distant networks

```
$ traceroute www.ibm.com
traceroute to www.ibm.com (129.42.58.216), 30 hops max, 40 byte packets
 1  gendarme.mcslp.pri (192.168.0.1)  3.163 ms  3.159 ms  6.618 ms
 2  gauthier-dsl1.hq.zen.net.uk (62.3.82.17)  34.336 ms  34.341 ms  34.341 ms
 3  lotze-ge-0-0-1-136.hq.zen.net.uk (62.3.80.137)  37.581 ms  47.276 ms  50.548 ms
 4  nietzsche-ae2-0.ls.zen.net.uk (62.3.80.70)  43.945 ms  47.239 ms  50.529 ms
 5  nozick-ge-3-1-0-0.ls.zen.net.uk (62.3.80.74)  55.343 ms  55.341 ms  55.339 ms
 6  lorenz-ge-3-0-0-0.te.zen.net.uk (62.3.80.78)  66.347 ms  63.118 ms  63.105 ms
 7  82.195.188.13 (82.195.188.13)  146.039 ms  118.175 ms  124.532 ms
 8  sl-bb22-lon-8-0.sprintlink.net (213.206.128.60)  50.460 ms  47.273 ms  40.991 ms
 9  sl-bb20-lon-12-0.sprintlink.net (213.206.128.52)  47.107 ms  47.094 ms  43.711 ms
10  sl-crs2-nyc-0-5-3-0.sprintlink.net (144.232.9.164)  111.579 ms  113.173 ms
    113.159 ms
11  144.232.18.238 (144.232.18.238)  116.353 ms  111.633 ms  111.619 ms
12  0.xe-5-0-1.XL3.NYC4.ALTER.NET (152.63.3.125)  114.812 ms  111.788 ms  115.000 ms
13  0.so-7-1-0.XT3.STL3.ALTER.NET (152.63.0.6)  151.969 ms  142.573 ms  142.574 ms
14  POS6-0.GW8.STL3.ALTER.NET (152.63.92.37)  142.552 ms  253.001 ms  252.986 ms
15  ibm-gw.customer.alter.net (65.206.180.74)  179.655 ms  228.775 ms  228.751 ms
16  10.16.255.10 (10.16.255.10)  145.847 ms  139.310 ms  142.509 ms
17  * * *
18  129.42.58.216 (129.42.58.216)  143.118 ms  141.181 ms  141.152 ms
```

Using this method, in combination with nmap to determine the list of hosts, you can gain a better understanding about the hosts on your network and which routers and systems are used to reach these systems.

Section 4. Summary

Conclusion

In this tutorial, you have learned about a number of different UNIX tools and techniques that can be used to determine different information about the hosts on your network, how accessible they are, what machines and other systems they are connected to, and the services and systems that they provide.

Using these techniques together, you should be able to enter any UNIX environment and work out the network configuration and, by recording the information, determine how and why things have gone wrong and how they can be fixed.

Resources

Learn

- [System Administration Toolkit: Network Scanning](#) (Martin Brown, developerWorks, December 2007): Get more tips on network scanning.
- [Solve application problems with tracing](#) (developerWorks): Get information on using truss, trace, and similar tools.
- Read [System Administration Toolkit: Standardizing your UNIX command-line tools](#) (Martin Brown, developerWorks, May 2006) to learn how to use the same command across multiple machines.
- For an article series that will teach you how to program in bash, see [Bash by example, Part 1: Fundamental programming in the Bourne again shell \(bash\)](#) (Daniel Robbins, developerWorks, March 2000), [Bash by example, Part 2: More bash programming fundamentals](#) (Daniel Robbins, developerWorks, April 2000), and [Bash by example, Part 3: Exploring the ebuild system](#) (Daniel Robbins, developerWorks, May 2000).
- [System Administration Toolkit](#): Check out other parts in this series.
- [Making Unix and Linux work together](#) (Martin Brown, developerWorks, April 2006) is a guide to getting traditional Unix distributions and Linux working together.
- Different systems use different tools, and the IBM Redbook [Solaris to Linux Migration: A Guide for System Administrators](#) will help you identify some key tools.
- [New to AIX and UNIX](#): Visit the New to AIX and UNIX page to learn more about AIX and UNIX.
- The [developerWorks AIX and UNIX zone](#) hosts hundreds of informative articles and introductory, intermediate, and advanced tutorials.
- [AIX Wiki](#): A collaborative environment for technical information related to AIX.
- [Technology bookstore](#) Browse this site for books and other technical topics.
- [developerWorks technical events and webcasts](#): Stay current with developerWorks technical events and webcasts.
- [Podcasts](#): Tune in and catch up with IBM technical experts.

Discuss

- Participate in the AIX and UNIX forums:
 - [AIX Forum](#)

- [AIX Forum for developers](#)
- [Cluster Systems Management](#)
- [IBM Support Assistant Forum](#)
- [Performance Tools Forum](#)
- [Virtualization Forum](#)
- More [AIX and UNIX Forums](#)

About the author

Martin Brown

Martin Brown has been a professional writer for over eight years. He is the author of numerous books and articles across a range of topics. His expertise spans myriad development languages and platforms -- Perl, Python, Java, JavaScript, Basic, Pascal, Modula-2, C, C++, Rebol, Gawk, Shellscrip, Windows, Solaris, Linux, BeOS, Mac OS/X and more -- as well as Web programming, systems management and integration. Martin is a regular contributor to ServerWatch.com, LinuxToday.com and IBM developerWorks, and a regular blogger at Computerworld, The Apple Blog and other sites, as well as a Subject Matter Expert (SME) for Microsoft. He can be contacted through his Web site at <http://www.mcslp.com>.