by Peter Collinson, Hillside Systems

PAUL SCHULENBURG

What Machines are out There?

I've been engaged in some form of computer networking for about 20 years. My earliest experience, in the late '70s, was connecting two computers together using a serial line. I moved on in the early '80s to write the code that was used in the hosts on our campus-wide network based on the Cambridge Ring. In the late '80s, it was back to serial lines (and X25) for the UUCP network.

The '90s have seen the emergence of the Internet and its killer application, the World Wide Web. As we approach the end of the '90s, the Web seems to be turning into something run by big business for big business, whose glossy Shockwaved sites often contain little of interest. Advertising agencies earn their money by turning nothing into something, but because they start with nothing, the actual content of these sites is zero. Nothing in, nothing out. For me, the essential aspect of the '90s Web has been its ability to give you access to information that was not available before. I hope the delivery of real information doesn't completely disappear in a wave of commercially created electronic billboards.

A very common experience in these 20 years has been the helpful phone call from the person who tells you that your network is broken in some way: your mail system is down, your Web server isn't responding, or some other cataclysm has befallen you. The call usually occurs at an inconvenient moment, and the stress is compounded because you have been using the network connection for some considerable period and were convinced that all was well with the world.

Now, the call is actually well-intentioned, it's someone trying to be helpful, so you cannot be too rude when you discover that the fault is theirs. I think I have grown into a state where I always presume that they have the defective connection and whatever fault they have isn't my problem. I always try to be polite when I point the fickle finger of fate to the deficiencies in their setup. After all, humility dictates that sometimes, only sometimes, the presumption of my innocence is misplaced.

I recently had an experience where someone was trying to send me mail and called saying, "Your mail system is broken." This is not a terribly helpful thing to be told. I tried not to get cross. "How is it broken?" "It's bouncing the mail that I am sending." "What does it say in the bounced message?" "Oh, I haven't looked at that." The usual cause of this error is a mistyped address and the bounce is actually from the user's Internet service provider (ISP) and not from the target mail system. I presumed my innocence.

The caller found the bounced message and read the error out to me. Further investigation showed that my mail system was indeed bouncing his mail because his domain was in my list of spammers, which meant that at some point in the recent past, a user of his ISP sent me considerable volumes of unwanted mail. I informed him of this,

Listing 1. dig Lookup

```
$ dig www.hillside.co.uk

; <<>> DiG 2.0 <<>> www.hillside.co.uk
;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 6
;; flags: qr aa rd ra ; Ques: 1, Ans: 2, Auth: 4, Addit: 4
;; QUESTIONS:
;;      www.hillside.co.uk, type = A, class = IN

;; ANSWERS:
www.hillside.co.uk.     86400   CNAME   wooded.hillside.co.uk.
wooded.hillside.co.uk.  86400   A       194.205.42.3

;; AUTHORITY RECORDS:
hillside.co.uk. 86400   NS      ns.hillside.co.uk.
hillside.co.uk. 86400   NS      nsa.hillside.co.uk.
hillside.co.uk. 86400   NS      ns0.insnet.net.
hillside.co.uk. 86400   NS      ns1.insnet.net.

;; ADDITIONAL RECORDS:
ns.hillside.co.uk.      86400   A       194.205.42.3
nsa.hillside.co.uk.     86400   A       194.205.42.1
ns0.insnet.net. 110796  A       194.177.160.34
ns1.insnet.net. 110796  A       194.177.170.34

;; Sent 1 pkts, answer found in time: 2 msec
;; FROM: craggy to SERVER: default -- 127.0.0.1
;; WHEN: Wed Jun  2 11:17:55 1999
;; MSG SIZE  sent: 36  rcvd: 232
```

and asked him whether he should be using an ISP that supported spammers. He made no comment.

I relate this tale because if you are going to be helpful, then you should realize that the person you are calling will probably assume that their system is working well. You can be more helpful (or plant the seeds of guilt) by transmitting any error messages you have received. Now, I will agree that error messages from mail systems are not exactly user-friendly and often contain loads of gobbledegook that probably frightens novice users, but if you look at the message, there is usually one sentence that explains the error.

On the whole, I try to diagnose an error before complaining about it. It doesn't take a guru to perform some basic checks on a remote machine or the service supplied by a remote machine. Quite often, these checks show there is something wrong at your end, and you can rectify it, or at least route an error report sensibly to get the problem fixed.

Verifying an Address

The most common fault is undoubtedly the use of an address that either doesn't exist or where the host machine has disappeared for some reason. As you probably know, domain names are mapped into IP addresses by the Domain Name System (DNS–the "S" can stand for "Service"). I discussed DNS in general terms in an article published in May 1998, "The Domain Name Service," *SunExpert*, Page 30 (`http://sw.expert.com/C2/SE.C2.MAY.98.pdf`). The DNS distributed database allows me to control my local address space using local files while telling you about it. Basically, let's assume you start an application that wants to look for one of my machines, `craggy.hillside.co.uk`. The application will ask your local DNS name server and that server will first reach out over the Internet for a server that supports the `.uk` domain. Then, with data based on the `.uk` information, it will look for a server that supports `.co.uk` and then for the address of my name server containing `hillside.co.uk`. Finally, your code will interrogate my name server to find details about the actual machine `craggy.hillside.co.uk`. The search can take some time, so the DNS code at your end will retain any results for some period so that a subsequent lookup will return local information, which may actually be out of date.

Also, some applications will give up and say, "Sorry I cannot find that information." This is particularly the case with Web browsers. If someone calls you and says, "Your Web site is down," when you know that it isn't, there's a good chance their browser has simply given up on the chore of looking up your address.

When you ask them to retype the address into the browser window, magically things will spring into life. In the interim, the needed DNS information has arrived and is present on their machine. At one time, I got several complaints about lack of Web service from my site from users of one ISP whose DNS system was simply overloaded. Asking them to type the address again sorted out the problem.

The DNS is a publicly available database, and there are user-level tools that allow you to interrogate it. Most machines will have the `nslookup` program, which is sometimes a little mysterious (it's hidden in `/usr/sbin` on Solaris). For quick lookups, I prefer the `dig` program that emanates from the University of Southern California. It's not a standard part of Solaris, but precompiled binaries are available. On my Red Hat Linux system, and also on my BSD system, `dig` is sitting there ready to use.

Listing 1 shows you how to lookup a machine with `dig`. Please pick another address when you try this for yourself, I do not need the extra traffic. The output is somewhat voluminous, and I don't intend to exhaustively explain it all here. The main reply to the request follows the `ANSWERS:` line. It shows that `www.hillside.co.uk` exists as a name in the DNS. This is an alias (a `CNAME`) to my machine `wooded.hillside.co.uk`, which has an IP address in an A record. The large numbers that follow the machine names are time-out values for the name, and I'll ignore them here. The remaining information tells you about the name servers that support my domain.

Looking up names using `dig` works for named machines like `www.hillside.co.uk`. You may think `www` implies a service, but this is only a convention, not part of the network. What about mail? The DNS supports a special type of record, the MX record, which tells mail systems where mail is to be sent for that address. MX records are used to advertise the names of a range of machines that are prepared to forward mail to the particular mail address. Again, if I use `dig` to inspect my mail address, I'll get the full answer as shown in Listing 1, but the `ANSWERS:` will be

```
$ dig mx hillside.co.uk
 ...
 ...
;; ANSWERS:
hillside.co.uk. 86400 MX 10 craggy.hillside.co.uk.
hillside.co.uk. 86400 MX 20 mxbackup.insnet.net.
 ...
 ...
```

which tells the mail system that it should first send mail to `craggy` and then to the backup mail system supplied by my ISP: `mxbackup.insnet.net`. The numbers before the machine names ("10" and "20") are priority values, the lower the better.

The MX records in my DNS record allow you to address mail to my site using my domain name. Actually, some people don't use this mechanism, and should. One option is to associate an IP address with your domain name and then hope that the sender's mail system will behave like `sendmail`. If `sendmail` cannot find an MX record for the address, but can find an IP address, it will send the mail to the IP address. The thinking here is that you should always be able to send mail to a specific machine. However, people have perverted this thoughtful mechanism to allow the transmission of mail to domains that happen to have IP addresses. Associating an IP address with a domain name is actually discouraged by the Internet standards, but people still do it.

The final point to make about `dig` is that it can be used to painlessly find the reverse mapping, translating an IP address into a machine name:

```
$ dig -x 194.205.42.1
 ...
 ...
;; ANSWERS:
1.42.205.194.in-addr.arpa.
                    86400 PTR craggy.hillside.co.uk.
 ...
 ...
```

This shows that my domain has reverse mapping entries (PTR records) that map onto the fake domain that is used for reverse lookup. Again, I am finding that some ISPs don't set up the PTR records properly and, increasingly, you cannot find a machine name for a particular IP address. My FTP system refuses to deal with such anonymous IP addresses, and I encourage you to do the same. I support anonymous access, meaning you

Listing 2. Using ping

```
$ ping -s craggy.hillside.co.uk
PING craggy.hillside.co.uk: 56 data bytes
64 bytes from craggy.hillside.co.uk (194.205.42.1): icmp_seq=0. time=169. ms
64 bytes from craggy.hillside.co.uk (194.205.42.1): icmp_seq=1. time=165. ms
64 bytes from craggy.hillside.co.uk (194.205.42.1): icmp_seq=2. time=272. ms
64 bytes from craggy.hillside.co.uk (194.205.42.1): icmp_seq=3. time=236. ms
64 bytes from craggy.hillside.co.uk (194.205.42.1): icmp_seq=4. time=331. ms
64 bytes from craggy.hillside.co.uk (194.205.42.1): icmp_seq=5. time=348. ms
^C
```

don't have to have an account, therefore access is anonymous. But I don't support your anonymity. I don't see why I should not know who has visited my system and pulled files.

The ping Command

Well, as you can see, `dig` allows you to discover how names map onto a particular IP address, and as a side effect, allows you to check that you're using a legal address. What if the address you are using is correct, but things are still not functioning? What next?

Well, is the machine that you need to access up and running, and can you get to it? The `ping` command can help. The command sends a message to the remote host and looks for a reply. If the remote machine replies, then it's alive. So on a Solaris or SunOS system, you'll see something like the following:

```
$ ping craggy.hillside.co.uk
craggy.hillside.co.uk is alive
```

You can give an IP address as an argument should you wish. Actually, the command is again hidden away; you'll find it in `/usr/sbin` on Solaris and `/usr/etc` on SunOS. The command needs superuser privilege and is `setuid` to root on all the systems I've looked at.

The `ping` command started life on BSD systems and its default action has been changed somewhat by Sun Microsystems. The original command sent a stream of packets (one per second) until terminated by the user with Control-C. You'll find this original behavior on Linux and BSD systems, and it can be invoked on Solaris or SunOS by supplying the `-s` switch. Listing 2 shows that I'm logged into a machine in California and `ping`ing my system in the United Kingdom.

The message that is sent by `ping` is a standard Internet Control Message Protocol (ICMP) packet. When received by a target machine, it elicits a response that is usually sent from the kernel of the machine's operating system, so you can tell the remote machine is alive. However, the machine may not be supporting users or may be online for other services.

The ICMP message contains a sequence number that is incremented by one for every message that is sent. The returned packet will contain the sequence number, and looking at the increasing sequence numbers can tell you whether or not any packets have been dropped in the round-trip from the sending machine to the remote machine and back. At busy times, you'll find that all networks exercise their right to throw away packets. Any TCP protocol connection that's carrying important information will cope by retrying to provide a reliable data stream.

The information at the end of each line is the trip time from the sending machine to remote machine and back. It's variable because of network congestion and the other network users. Actually, the average time from California to the United Kingdom two years ago was roughly 250 msec, it's about 180 msec now, reflecting the improved trans-continental links in the United States, and also the faster links used by my ISP across the Atlantic Ocean.

The `ping` command tells you four things. First, it tells you that you can reach the remote machine. Second, that the machine can get packets back to you. Third, that the remote machine is alive, has power and is running some operating system. And finally, it gives you an indication of the speed of the connection between the remote machine and you.

It's a little harder to find out whether or not services are functioning on the remote machine. One approach with some Internet protocols is to use `telnet` to tickle the server. This is viable with FTP, SMTP (mail) and HTTP (Web access). For example,

```
$ telnet machine ftp
```

The `ftp` keyword is looked up in `/etc/services` and translated into a port number. You can always do that yourself and supply a port number as the second argument to the command.

The trick here is to connect and then type `QUIT` to stop the connection. You may also need to know how to crash out of `telnet`: type Control-] to get the `telnet` prompt and then type `close`. If you are bold, before you leave you can also type in some commands from the relevant protocol to make the server do some work. However, do look at the appropriate protocol specification first.

Incidentally, Sun has long provided a command called `mconnect`, which is intended to connect to a mail server and tell you if it's working.

Routes

Sometimes the use of the DNS and `ping` can fool you into thinking that a remote machine is down. I recently had a situation where mail addressed to a customer was hanging around my machine for some time. Simply `ping`ing the machine gave no response, looking up the MX record and the machine name in the DNS showed that the machine didn't exist. Because my

customer had reported poor experiences with his ISP, I suspected that their system had died and the situation was bad. I left it, hoping it would burst into life.

Around 24 hours later, the situation had not altered and further enquiries told me that other people could see the name server and the remote network. When I checked from the United States, I too could see the name server and the network; it looked fine from outside my network, but was invisible from inside.

I used the `traceroute` command to see the routes that packets from my machine took to get to the machine and discovered there was a router problem. My packets were traveling to Germany (rather than staying in the United Kingdom) and were ending up in a black hole. The problem was caused by a routing failure. I contacted my ISP, who sorted things out.

The `traceroute` command was written by Van Jacobsen (of Lawrence Berkeley Laboratories). It's not part of the Solaris release (I haven't managed to look at Solaris 7 yet), but is standard on BSD and Linux. You can find precompiled binaries for Solaris on the Internet. Again, it needs superuser access.

The command makes use of a property of the IP protocol. Each IP packet contains a TTL (time to live) value, which is intended to stop packets traveling endlessly around the world. The TTL is set by the sender and is decremented by every router that the packet passes through. If it reaches one (or zero), then the router is allowed to throw the packet away. However, when doing so, it will send a ICMP message back to the transmitting host saying that it's a time-exceeded packet.

The `traceroute` program uses this behavior to work out the routes packets are taking. First, it sends a packet to the destination host with a TTL of one. The first router will see this, and send back an ICMP error message that can be used to determine its address. Then, `traceroute` sends a message with a TTL of two, getting the next router, and so on until the destination machine is reached. The result is a trace of the route packets have taken to get to the destination and back.

The `traceroute` command allows you to detect loops and black holes, and can sometimes be used to determine where the packets of an unknown IP address emanated from. Beware that some routers block these messages so that you cannot peer into their network.

Further Reading

Much of the basic Internet stuff in this article comes from the definitive, *TCP/IP Illustrated, Volume 1–The Protocols*, by W. Richard Stevens (published by Addison-Wesley Publishing Co., 1994, ISBN 0-201-63346-9). You can get binaries of `dig` and `traceroute` for Solaris from the Solaris Freeware Project at `http://sunfreeware.com`.

Peter Collinson *runs his own UNIX consultancy, dedicated to earning enough money to allow him to pursue his own interests: doing whatever, whenever, wherever... He writes, teaches, consults and programs using Solaris running on a SPARCstation 2. Email:* `pc@cpg.com`.