

Simple Network Diagnostic Tools

Ping and Traceroute

Hans Kruse and Carl Bruggeman

Jan 1, 2007

As we build networks in the lab we will need tools to check our work as we go. “Real” programs (like web browsers) are not very useful for this purpose; there are too many different causes that can make them not work, and they require a server operating on “the other end”. Fortunately, the Internet provides some diagnostic capabilities in the form of the Internet Control Message Protocol. Both `ping` and `traceroute` use ICMP to give us basic visibility into whether our network provides the desired connectivity.

These tools are so useful because every Internet host is required to implement ICMP, and all operating systems provide a version of the `ping` and `traceroute` programs (however, as we will see, the implementations are not all identical).

The ping command

The term “ping” comes from the way you “look out of” a submarine, which after all has no windows. Submarines use their sonar equipment to send out short bursts of sound (a “ping”) and listen to see if an echo comes back. If you hear an echo you know something is out there, and from the time it took the echo to come back you can guess how far away the object is. Not much information, but it keeps submarines from running into things.

The `ping` program does much the same thing as a submarine sonar. You give `ping` the address of the host to which you want to test connectivity. `ping` then creates an ICMP Echo Request packet, and sends it to that address. If the Echo Request packet reaches its destination, the target host is required to respond with an ICMP Echo Reply packet, which it sends to the address from which the Echo Request came. Assuming that the Echo Reply packet makes it back to your machine, `ping` lets you know that the reply came back, and tells you how much time elapsed between sending the Request and getting the Reply back (remember the submarine?).

What does the ping output mean?

The command `ping 132.235.64.1` might produce this output:

```
PING 132.235.64.1 (132.235.64.1): 56 data bytes
64 bytes from 132.235.64.1: icmp_seq=0 ttl=254 time=0.351 ms
64 bytes from 132.235.64.1: icmp_seq=1 ttl=254 time=0.346 ms
64 bytes from 132.235.64.1: icmp_seq=2 ttl=254 time=0.362 ms
64 bytes from 132.235.64.1: icmp_seq=3 ttl=254 time=0.359 ms
64 bytes from 132.235.64.1: icmp_seq=4 ttl=254 time=0.357 ms
64 bytes from 132.235.64.1: icmp_seq=5 ttl=254 time=0.337 ms
^C
--- 132.235.64.1 ping statistics ---
```

```
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.337/0.352/0.362/0.009 ms
```

The first line of output confirms the command you issued (you will see why there are two copies of the IP address in a bit); each line after that represents a Reply received. Lets look at the first such output line:

64 bytes is the size of the ICMP Echo reply

from 132.235.64.1 should normally be the address of the machine you pinged in the first place

icmp_seq=0 – each ICMP Echo Request packet is numbered, and the reply tells the ping program which of the numbered packets is being replied to. The numbers should increase by 1 each time, unless the network drops packets from time to time (either Requests or Replies); so this is a simple way to tell if there is packet loss, especially if you run ping for a while.

ttl=254 – “TTL” means Time to Live. This is a counter in every IP packet. The sender (ping in this case) sets it to whatever it thinks a reasonable number is (the maximum is 255). Each router subtracts one, and packets get thrown out if TTL ever reaches 0. Assuming we started with TTL=255, this report gives us an estimate of the number of routers we passed through (1 in our example).

time=... – This is the time between the Echo Request and Reply. Since networks are dynamic, this time will probably be different for each reply received.

Note the “^C ” at the bottom of the reply lines. Some ping programs run continuously until you stop them with Control-C. More about that below, when we talk about differences between ping implementations.

The statistics at the end of the command output are:

- How many packets (Requests) were sent out
- How many packets (Replies) were received. If fewer packets were received than sent, you get an estimate of packet loss. Note, however, that one or more reply may have been in transit when you stopped the ping, so this is not always accurate.
- The number of missed replies as a percentage (remember the note above).
- Statistics computed from the “time” values in the individual replies, namely

min – the smallest time value seen

avg – the average over all time values

max – the largest time value seen

stddev – the standard deviation computed over all time values received. This gives an estimate of how much variation there is in the time values. Note that not all versions of ping compute this.

A final note on the use of ping. For ping to work properly, the network must forward ICMP Echo Request and Echo Reply packets. In our ITL networks, that is usually the case when the network has been set up correctly. However, in the Internet in general, many things other than malfunctions can have an effect on the output from ping. Security devices (e.g. firewalls) and high performance routers are often configured to silently drop ICMP packets. When your ping fails, it may be encountering such a device. Therefore, if you are using ping to demonstrate that your local network can access “the Internet”, make sure you are pinging a target that is know to respond.

ping and Host *Names*

ping can of course resolve names to addresses, so typing `ping www.itl.ohiou.edu` will work fine:

```
PING www.itl.ohiou.edu (132.235.201.2): 56 data bytes
64 bytes from 132.235.201.2: icmp_seq=0 ttl=253 time=0.724 ms
64 bytes from 132.235.201.2: icmp_seq=1 ttl=253 time=0.710 ms
```

You can now see why the initial output line looked odd before, ping acknowledges your command by echoing the argument you gave it, followed by the address (which means you saw the address twice in the earlier example). The examples so far came from Mac OS X, lets try this on Ubuntu Linux, and we get (for the same command):

```
PING www.itl.ohiou.edu (132.235.201.2) 56(84) bytes of data.
64 bytes from www.itl.ohiou.edu (132.235.201.2): icmp_seq=1 ttl=253 time=0.633 ms
64 bytes from www.itl.ohiou.edu (132.235.201.2): icmp_seq=2 ttl=253 time=0.613 ms
64 bytes from www.itl.ohiou.edu (132.235.201.2): icmp_seq=3 ttl=253 time=0.637 ms
```

Ubuntu Linux not only understands a name as its argument, but it also helpfully looks up a name for the address in the reply packets. Pretty cool, but in the ITL lab, you **should not use this!** Here is why:

1. In our labs, the addresses you are given to use for your machines are not in the name server, you have to use numerical addresses for these machines.
2. In many cases you will be testing a partially completed network, your machines may not be able to reach a name server yet!
3. In the Ubuntu example above, your machine may not display the replies until it does the address to name conversion. If a name server is not available, that is a long wait! Until ping gives up, it will look as if your network is down (no replies shown), when the problem is really lack of access to a name server.

So, we recommend for the basic testing we do in the ITL that you turn off address to name conversion. How you do that differs on each OS, see the next section. By the way, this same discussion applies to `traceroute` just as much, maybe more.

ping on different operating systems

Unfortunately, ping does not quite behave the same on all systems, as we have already seen above. At the end of this note you will find the `man` page for ping from Mac OS X. This section contains some notes on different options and behaviors you need to be aware of.

Mac OS X

- We recommend you use `ping -n` to force numeric address output, but the current version of ping on OS X does not really do the reverse lookup anyway.
- ping will run indefinitely by default, use “`^C`” to stop the command (and get statistics). You can use `ping -t` to have ping run for a specific time, if you would like.

Ubuntu Linux

- On Linux you do need to use `ping -n`.
- `ping` runs indefinitely by default, just like Mac OS X.
- The first sequence number is 1 (on Mac OS X it is 0).
- The statistics lines look a bit different.

Windows

- `ping` send 4 probes by default. Use `ping -t` to have it run indefinitely as on the Unix system. When you do, “^C” will work the same way as on Unix.
- You can use `ping -n <number>` to send a defined number of probes. Note that this is a very different use of “-n” than on Unix.
- Address to name translation is off by default. You can use `ping -a` to turn it on when you are pinging an address, but we recommend that you *not* do this.
- The statistics output looks different and does not include the standard deviation.
- You can see the equivalent of the man page with the command `ping /?`.

Sun Solaris

On Solaris, you must use `ping -s` to get a continuous series of pings as described above.

The traceroute command

The `ping` command lets us quickly determine if two hosts can exchange (ICMP) packets. However, if the answer is no, we do not get much information about the location or nature of the fault. Because complex diagnostics can themselves increase network load, the Internet only defines basic error reporting in ICMP. The `traceroute` command makes use of these basic error reporting functions to give us more visibility into the network. When a router cannot forward a packet, it sends an ICMP error message back to the machine that sent the original packet. A host machine will do the same for packets that it is not prepared to handle. `traceroute` triggers these errors on purpose, to provide a simple “map” of the network between you and your target. Lets look at an example:

```
> traceroute www.osc.edu
traceroute to www.osc.edu (192.148.248.50), 64 hops max, 40 byte packets
 1  gwv-232-ald6600.cns.ohiou.edu (132.235.232.254)  1.049 ms  0.343 ms  0.316 ms
 2  132.235.195.14 (132.235.195.14)  0.617 ms  0.370 ms  0.320 ms
 3  132.235.195.25 (132.235.195.25)  4.026 ms  7.717 ms  9.459 ms
 4  athna-r0-ge-0-0-0s395.bb.oar.net (199.18.17.249)  5.712 ms  4.620 ms  12.781 ms
 5  clmbn-r0-ge7-3s392.core.oar.net (199.18.17.17)  14.362 ms  7.839 ms  4.863 ms
 6  clmbn-r1-pc3-0.core.oar.net (199.18.145.2)  5.273 ms  3.951 ms  10.618 ms
 7  clmbs-r0-ge-0-1-0s857.bb.oar.net (199.18.0.25)  12.492 ms  5.026 ms  3.772 ms
 8  clmbo-r0-ge-0-0-0s855.bb.oar.net (199.18.0.17)  5.766 ms  4.408 ms  14.311 ms
 9  osc-r0-ge-1-3-0s68.cpe.oar.net (199.18.22.6)  11.341 ms  6.328 ms  7.161 ms
10  osc-r2-vl65.cpe.oar.net (192.153.41.10)  4.472 ms  17.073 ms  7.026 ms
11  clmbs-r10-vl654.osc.oar.net (192.153.41.54)  6.953 ms  6.544 ms  9.646 ms
12  osc.edu (192.148.248.50)  8.542 ms  9.769 ms  12.739 ms
```

Here is how `traceroute` does this:

1. It creates a packet to `www.osc.edu` and set the TTL to zero (you *do* remember the TTL discussion from the section on `ping`, right?). The first router, i.e. `gwv-232-ald6600.cns.ohiou.edu`, discards that packet and sends an ICMP Time to Live Exceeded error message. That generates the output in the line starting with “1” above. Actually, `traceroute` does this three times by default, and reports the time between sending the packet and receiving the ICMP message for each probe. Note that these time can vary quite a bit.
2. `traceroute` now sends the same packet with a TTL=1. That makes it through the first router, but fails on the second one, which sends the ICMP Time Exceeded message. That identifies the hop in line “2”.
3. `traceroute` keeps sending the same packet with increasing TTLs. Eventually the packet will get to the host we specified, `www.osc.edu`. By default, `traceroute` sends UDP packets with a random port number, meaning that the receiving host probably does not know what to do with the packet. The host will respond to the packet with an ICMP Port Unreachable error, which tells `traceroute` that it is done.

Things to remember when using traceroute

The description above describes how `traceroute` is supposed to work. However, it rarely works this smoothly. The man page for `traceroute`, which is included in this note, describes a number of them. For our purposes, here are the important ones:

- In the ITL, always turn the address to name translation off. (The section on operating system differences shows the details). `traceroute` can appear to fail (produce no output) for a long time if a name server is unavailable.
- For many reasons, some of the probes will not return responses. If at least one succeeds for a given hop, the corresponding output line will be filled in, with some of the time values replaced by “*”. If all probes fail, the output line will only contain three “*”s.
- If the last ICMP Port Unreachable error is not received — either because the host did not send it, or because the ICMP messages are dropped by a firewall or router — `traceroute` will continue to probe up to the maximum number of hops requested (typically between 30 and 64). This can take quite some time. You can stop `traceroute` with “^C”.
- Occasionally a router downstream will be mis-configured, and the probe packet will trigger an unusual error message. `traceroute` displays these in place of the times, for example !H means “Host Unreachable”, !N means “Network Unreachable”. These can give you valuable diagnostic information when your network is not working properly.

traceroute on different operating systems

Mac OS X and Linux

- Use `traceroute -n` to turn off address to name resolution.
- Mac OS X will probe for 64 hops by default.
- Linux will probe for 30 hops by default.

Windows

- The `tracert` command is called `tracert`.
- Use `tracert /?` to get usage information.
- Use `tracert -d` to turn off address to name resolution.
- `tracert` probes for 30 hops by default.
- The `tracert` program uses a different packet size and type than `tracert`. A virus author decided to use the `tracert` packet format to hide communications to/from the virus, and succeeded in getting many sites (including OU) to block `tracert` probes. The program is still useful in the ITL, but can't "see" outside the OU network. We will discuss alternatives in the class and/or the lab.

The ping man page from Mac OS X

PING(8) BSD System Manager's Manual PING(8)

NAME

ping -- send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

```
ping [-AaDdfnoQqRrv] [-c count] [-i wait] [-l preload] [-M mask | time]
[-m ttl] [-P policy] [-p pattern] [-S src_addr] [-s packetsize]
[-t timeout] [-z tos] host
ping [-AaDdfLnoQqRrv] [-c count] [-I iface] [-i wait] [-l preload]
[-M mask | time] [-m ttl] [-P policy] [-p pattern] [-S src_addr]
[-s packetsize] [-T ttl] [-t timeout] [-z tos] mcast-group
```

DESCRIPTION

The ping utility uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ("pings") have an IP and ICMP header, followed by a "struct timeval" and then an arbitrary number of "pad" bytes used to fill out the packet. The options are as follows:

-A Audible. Output a bell (ASCII 0x07) character when no packet is received before the next packet is transmitted. To cater for round-trip times that are longer than the interval between transmissions, further missing packets cause a bell only if the maximum number of unreceived packets has increased.

-a Audible. Include a bell (ASCII 0x07) character in the output when any packet is received. This option is ignored if other format options are present.

-c count
Stop after sending (and receiving) count ECHO_RESPONSE packets. If this option is not specified, ping will operate until interrupted.

-D Set the Don't Fragment bit.

-d Set the SO_DEBUG option on the socket being used.

-f Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. For every ECHO_REQUEST sent a period "." is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. Only the super-user may use this option. This can be very hard on a network and should be used with caution.

-I iface
Source multicast packets with the given interface address. This

flag only applies if the ping destination is a multicast address.

`-i wait`

Wait wait seconds between sending each packet. The default is to wait for one second between each packet. The wait time may be fractional, but only the super-user may specify values less than 1 second. This option is incompatible with the `-f` option.

`-L` Suppress loopback of multicast packets. This flag only applies if the ping destination is a multicast address.

`-l preload`

If preload is specified, ping sends that many packets as fast as possible before falling into its normal mode of behavior. Only the super-user may use this option.

`-M mask | time`

Use `ICMP_MASKREQ` or `ICMP_TSTAMP` instead of `ICMP_ECHO`. For mask, print the netmask of the remote machine. Set the `net.inet.icmp.maskrepl` MIB variable to enable `ICMP_MASKREPLY`. For time, print the origination, reception and transmission time-stamps.

`-m ttl` Set the IP Time To Live for outgoing packets. If not specified, the kernel uses the value of the `net.inet.ip.ttl` MIB variable.

`-n` Numeric output only. No attempt will be made to lookup symbolic names for host addresses.

`-o` Exit successfully after receiving one reply packet.

`-P policy`

policy specifies IPsec policy for the ping session. For details please refer to `ipsec(4)` and `ipsec_set_policy(3)`.

`-p pattern`

You may specify up to 16 ‘‘pad’’ bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, ‘‘-p ff’’ will cause the sent packet to be filled with all ones.

`-Q` Somewhat quiet output. Don’t display ICMP error messages that are in response to our query messages. Originally, the `-v` flag was required to display such errors, but `-v` displays all ICMP error messages. On a busy machine, this output can be overbearing. Without the `-Q` flag, ping prints out any ICMP error messages caused by its own `ECHO_REQUEST` messages.

`-q` Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

`-R` Record route. Includes the `RECORD_ROUTE` option in the `ECHO_REQUEST` packet and displays the route buffer on returned packets. Note that the IP header is only large enough for nine such routes; the `tracert(8)` command is usually better at determining the route packets take to a particular destination. If more routes come back than should, such as due to an illegal spoofed packet, ping will print the route list and then truncate it at the correct spot. Many hosts ignore or discard the `RECORD_ROUTE` option.

`-r` Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by `routed(8)`).

`-S src_addr`

Use the following IP address as the source address in outgoing packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the IP address of the interface the probe packet is sent on. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent.

`-s packetsize`

Specify the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

`-T ttl` Set the IP Time To Live for multicasted packets. This flag only applies if the ping destination is a multicast address.

`-t timeout`

Specify a timeout, in seconds, before ping exits regardless of how many packets have been received.

`-v` Verbose output. ICMP packets other than `ECHO_RESPONSE` that are received are listed.

`-z tos` Use the specified type of service.

When using ping for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be 'pinged'. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the round-trip time statistics. When the specified number of packets have been sent (and received) or if the program is terminated with a `SIGINT`, a brief summary is displayed, showing the number of packets sent and received, and the minimum, mean, maximum, and standard deviation of

the round-trip times.

If ping receives a SIGINFO (see the status argument for stty(1)) signal, the current number of packets sent and received, and the minimum, mean, and maximum of the round-trip times will be written to the standard error output.

This program is intended for use in network testing, measurement and management. Because of the load it can impose on the network, it is unwise to use ping during normal operations or from automated scripts.

ICMP PACKET DETAILS

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. When a packetsize is given, this indicated the size of this extra piece of data (the default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header).

If the data space is at least eight bytes large, ping uses the first eight bytes of this space to include a timestamp which it uses in the computation of round trip times. If less than eight bytes of pad are specified, no round trip times are given.

DUPLICATE AND DAMAGED PACKETS

The ping utility will report duplicate and damaged packets. Duplicate packets should never occur when pinging a unicast address, and seem to be caused by inappropriate link-level retransmissions. Duplicates may occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates may not always be cause for alarm. Duplicates are expected when pinging a broadcast or multicast address, since they are not really duplicates but replies from different hosts to the same request.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the ping packet's path (in the network or in the hosts).

TRYING DIFFERENT DATA PATTERNS

The (inter)network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases the particular pattern that will have problems is something that does not have sufficient 'transitions', such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It is not necessarily enough to specify a data pattern of all zeros (for example) on the command line because the pattern that is of interest is at the data link level, and the relationship between what you type and what the controllers transmit can be complicated.

This means that if you have a data-dependent problem you will probably

have to do a lot of testing to find it. If you are lucky, you may manage to find a file that either cannot be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the `-p` option of ping.

TTL DETAILS

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. In current practice you can expect each router in the Internet to decrement the TTL field by exactly one.

The TCP/IP specification recommends setting the TTL field for IP packets to 64, but many systems use smaller values (4.3BSD uses 30, 4.2BSD used 15).

The maximum possible value of this field is 255, and most UNIX systems set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you will find you can “ping” some hosts, but not reach them with telnet(1) or ftp(1).

In normal operation ping prints the ttl value from the packet it receives. When a remote system receives a ping packet, it can do one of three things with the TTL field in its response:

- o Not change it; this is what BSD systems did before the 4.3BSD-Tahoe release. In this case the TTL value in the received packet will be 255 minus the number of routers in the round-trip path.
- o Set it to 255; this is what current BSD systems do. In this case the TTL value in the received packet will be 255 minus the number of routers in the path from the remote system to the pinging host.
- o Set it to some other value. Some machines use the same value for ICMP packets that they use for TCP packets, for example either 30 or 60. Others may use completely wild values.

RETURN VALUES

The ping utility returns an exit status of zero if at least one response was heard from the specified host; a status of two if the transmission was successful but no responses were received; or another value (from `<syssexits.h>`) if an error occurred.

SEE ALSO

netstat(1), ifconfig(8), routed(8), traceroute(8)

HISTORY

The ping utility appeared in 4.3BSD.

AUTHORS

The original ping utility was written by Mike Muuss while at the US Army

Ballistics Research Laboratory.

BUGS

Many Hosts and Gateways ignore the RECORD_ROUTE option.

The maximum IP header length is too small for options like RECORD_ROUTE to be completely useful. There's not much that can be done about this, however.

Flood pinging is not recommended in general, and flood pinging the broadcast address should only be done under very controlled conditions.

The -v option is not worth much on busy hosts.

BSD October 2, 2002 BSD

The traceroute man page from Mac OS X

TRACEROUTE(8) BSD System Manager's Manual TRACEROUTE(8)

NAME

traceroute -- print the route packets take to network host

SYNOPSIS

```
traceroute [-dFISdnrvx] [-f first_ttl] [-g gateway] [-i iface]
[-M first_ttl] [-m max_ttl] [-P proto] [-p port] [-q nqueries]
[-s src_addr] [-t tos] [-w waittime] [-z pausesecs] host
[packetsize]
```

DESCRIPTION

The Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking the route one's packets follow (or finding the miscreant gateway that's discarding your packets) can be difficult. Traceroute utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host.

The only mandatory parameter is the destination host name or IP number. The default probe datagram length is 40 bytes, but this may be increased by specifying a packet size (in bytes) after the destination host name.

Other options are:

-f first_ttl

Set the initial time-to-live used in the first outgoing probe packet.

-F Set the "don't fragment" bit.

-d Enable socket level debugging.

-g gateway

Specify a loose source route gateway (8 maximum).

-i iface

Specify a network interface to obtain the source IP address for outgoing probe packets. This is normally only useful on a multi-homed host. (See the **-s** flag for another way to do this.)

-I Use ICMP ECHO instead of UDP datagrams. (A synonym for "**-P icmp**").

-M first_ttl

Set the initial time-to-live value used in outgoing probe packets. The default is 1, i.e., start with the first hop.

-m max_ttl

Set the max time-to-live (max number of hops) used in outgoing probe packets. The default is `net.inet.ip.ttl` hops (the same default used for TCP connections).

`-n` Print hop addresses numerically rather than symbolically and numerically (saves a nameserver address-to-name lookup for each gateway found on the path).

`-P proto`

Send packets of specified IP protocol. The currently supported protocols are: UDP , TCP , GRE and ICMP Other protocols may also be specified (either by name or by number), though traceroute does not implement any special knowledge of their packet formats. This option is useful for determining which router along a path may be blocking packets based on IP protocol number. But see BUGS below.

`-p port`

Protocol specific. For UDP and TCP, sets the base port number used in probes (default is 33434). Traceroute hopes that nothing is listening on UDP ports base to base+n hops-1 at the destination host (so an ICMP PORT_UNREACHABLE message will be returned to terminate the route tracing). If something is listening on a port in the default range, this option can be used to pick an unused port range.

`-q nqueries`

Set the number of probes per ‘‘ttl’’ to nqueries (default is three probes).

`-r` Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by `routed(8)`).

`-s src_addr`

Use the following IP address (which must be given as an IP number, not a hostname) as the source address in outgoing probe packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the IP address of the interface the probe packet is sent on. If the IP address is not one of this machine’s interface addresses, an error is returned and nothing is sent. (See the `-i` flag for another way to do this.)

`-S` Print a summary of how many probes were not answered for each hop.

`-t tos` Set the type-of-service in probe packets to the following value (default zero). The value must be a decimal integer in the range

0 to 255. This option can be used to see if different types-of-service result in different paths. (If you are not running a 4.4BSD or later system, this may be academic since the normal network services like telnet and ftp don't let you control the TOS). Not all values of TOS are legal or meaningful - see the IP spec for definitions. Useful values are probably '-t 16' (low delay) and '-t 8' (high throughput).

-v Verbose output. Received ICMP packets other than TIME_EXCEEDED and UNREACHABLEs are listed.

-w Set the time (in seconds) to wait for a response to a probe (default 5 sec.).

-x Toggle IP checksums. Normally, this prevents traceroute from calculating IP checksums. In some cases, the operating system can overwrite parts of the outgoing packet but not recalculate the checksum (so in some cases the default is to not calculate checksums and using -x causes them to be calculated). Note that checksums are usually required for the last hop when using ICMP ECHO probes (-I). So they are always calculated when using ICMP.

-z pausesecs

Set the time (in milliseconds) to pause between probes (default 0). Some systems such as Solaris and routers such as Ciscos rate limit ICMP messages. A good value to use with this is 500 (e.g. 1/2 second).

This program attempts to trace the route an IP packet would follow to some internet host by launching UDP probe packets with a small ttl (time to live) then listening for an ICMP "time exceeded" reply from a gateway. We start our probes with a ttl of one and increase by one until we get an ICMP "port unreachable" (which means we got to "host") or hit a max (which defaults to net.inet.ip.ttl hops & can be changed with the -m flag). Three probes (changed with -q flag) are sent at each ttl setting and a line is printed showing the ttl, address of the gateway and round trip time of each probe. If the probe answers come from different gateways, the address of each responding system will be printed. If there is no response within a 5 sec. timeout interval (changed with the -w flag), a "*" is printed for that probe.

We don't want the destination host to process the UDP probe packets so the destination port is set to an unlikely value (if some clod on the destination is using that value, it can be changed with the -p flag).

A sample use and output might be:

```
[yak 71]% traceroute nis.nsf.net.  
traceroute to nis.nsf.net (35.1.1.48), 64 hops max, 38 byte packet  
1 helios.ee.lbl.gov (128.3.112.1) 19 ms 19 ms 0 ms  
2 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
```

```

3 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 39 ms 40 ms 39 ms
5 ccn-nerif22.Berkeley.EDU (128.32.168.22) 39 ms 39 ms 39 ms
6 128.32.197.4 (128.32.197.4) 40 ms 59 ms 59 ms
7 131.119.2.5 (131.119.2.5) 59 ms 59 ms 59 ms
8 129.140.70.13 (129.140.70.13) 99 ms 99 ms 80 ms
9 129.140.71.6 (129.140.71.6) 139 ms 239 ms 319 ms
10 129.140.81.7 (129.140.81.7) 220 ms 199 ms 199 ms
11 nic.merit.edu (35.1.1.48) 239 ms 239 ms 239 ms

```

Note that lines 2 & 3 are the same. This is due to a buggy kernel on the 2nd hop system - lbl-csam.arpa - that forwards packets with a zero ttl (a bug in the distributed version of 4.3 BSD). Note that you have to guess what path the packets are taking cross-country since the NSFNet (129.140) doesn't supply address-to-name translations for its NSSes.

A more interesting example is:

```

[yak 72]% traceroute allspice.lcs.mit.edu.
traceroute to allspice.lcs.mit.edu (18.26.0.115), 64 hops max
1 helios.ee.lbl.gov (128.3.112.1) 0 ms 0 ms 0 ms
2 lilac-dmc.Berkeley.EDU (128.32.216.1) 19 ms 19 ms 19 ms
3 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 19 ms 19 ms
4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 19 ms 39 ms 39 ms
5 ccn-nerif22.Berkeley.EDU (128.32.168.22) 20 ms 39 ms 39 ms
6 128.32.197.4 (128.32.197.4) 59 ms 119 ms 39 ms
7 131.119.2.5 (131.119.2.5) 59 ms 59 ms 39 ms
8 129.140.70.13 (129.140.70.13) 80 ms 79 ms 99 ms
9 129.140.71.6 (129.140.71.6) 139 ms 139 ms 159 ms
10 129.140.81.7 (129.140.81.7) 199 ms 180 ms 300 ms
11 129.140.72.17 (129.140.72.17) 300 ms 239 ms 239 ms
12 * * *
13 128.121.54.72 (128.121.54.72) 259 ms 499 ms 279 ms
14 * * *
15 * * *
16 * * *
17 * * *
18 ALLSPICE.LCS.MIT.EDU (18.26.0.115) 339 ms 279 ms 279 ms

```

Note that the gateways 12, 14, 15, 16 & 17 hops away either don't send ICMP "time exceeded" messages or send them with a ttl too small to reach us. 14 - 17 are running the MIT C Gateway code that doesn't send "time exceeded"s. God only knows what's going on with 12.

The silent gateway 12 in the above may be the result of a bug in the 4.[23] BSD network code (and its derivatives): 4.x (x <= 3) sends an unreachable message using whatever ttl remains in the original datagram. Since, for gateways, the remaining ttl is zero, the ICMP "time exceeded" is guaranteed to not make it back to us. The behavior of this bug is slightly more interesting when it appears on the destination system:

```

1 helios.ee.lbl.gov (128.3.112.1) 0 ms 0 ms 0 ms
2 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 19 ms 39 ms
3 lilac-dmc.Berkeley.EDU (128.32.216.1) 19 ms 39 ms 19 ms
4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 39 ms 40 ms 19 ms
5 ccn-nerif35.Berkeley.EDU (128.32.168.35) 39 ms 39 ms 39 ms
6 csgw.Berkeley.EDU (128.32.133.254) 39 ms 59 ms 39 ms
7 * * *
8 * * *
9 * * *
10 * * *
11 * * *
12 * * *
13 rip.Berkeley.EDU (128.32.131.22) 59 ms ! 39 ms ! 39 ms !

```

Notice that there are 12 "gateways" (13 is the final destination) and exactly the last half of them are "missing". What's really happening is that rip (a Sun-3 running Sun OS3.5) is using the ttl from our arriving datagram as the ttl in its ICMP reply. So, the reply will time out on the return path (with no notice sent to anyone since ICMP's aren't sent for ICMP's) until we probe with a ttl that's at least twice the path length. I.e., rip is really only 7 hops away. A reply that returns with a ttl of 1 is a clue this problem exists. Traceroute prints a "!" after the time if the ttl is ≤ 1 . Since vendors ship a lot of obsolete (DEC's Ultrix, Sun 3.x) or non-standard (HPUX) software, expect to see this problem frequently and/or take care picking the target host of your probes.

Other possible annotations after the time are !H, !N, or !P (host, network or protocol unreachable), !S (source route failed), !F (fragmentation needed - the RFC1191 Path MTU Discovery value is displayed), !X (communication administratively prohibited), !V (host precedence violation), !C (precedence cutoff in effect), or !<num> (ICMP unreachable code <num>). These are defined by RFC1812 (which supersedes RFC1716). If almost all the probes result in some kind of unreachable, traceroute will give up and exit.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use traceroute during normal operations or from automated scripts.

AUTHOR

Implemented by Van Jacobson from a suggestion by Steve Deering. Debugged by a cast of thousands with particularly cogent suggestions or fixes from C. Philip Wood, Tim Seaver and Ken Adelman.

SEE ALSO

netstat(1), ping(8)

BUGS

When using protocols other than UDP, functionality is reduced. In par-

ticular, the last packet will often appear to be lost, because even though it reaches the destination host, there's no way to know that because no ICMP message is sent back. In the TCP case, traceroute should listen for a RST from the destination host (or an intermediate router that's filtering packets), but this is not implemented yet.