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** *pausemsecs*]
host [*packetlen*]

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 length (in bytes) after the destination host name.

Other options are:

−f 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 Specify a loose source route gateway (8 maximum).

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

−m 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 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 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 + nhops - 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 Set the number of probes per hop (default is 3).

−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*(8C)).

−s Use the following IP address (which usually is given as an IP number, not a hostname) as the source address in outgoing probe packets. On multi-homed hosts (those 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 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 4.4bsd, 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 defi nitions. 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 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 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 (change 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 4.3BSD). 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−<pmtu>** (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.

SEE ALSO

pathchar(8), netstat(1), ping(8)

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.

The current version is available via anonymous ftp:

ftp://ftp.ee.lbl.gov/traceroute.tar.gz

BUGS

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

Please send bug reports to traceroute@ee.lbl.gov.