NAME

traceroute - print the route packets trace to network host

SYNOPSIS

traceroute [**-46dFITUnreAV**] [**-f first_ttl**] [**-g gate,...**]
[**-i device**] [**-m max_ttl**] [**-p port**] [**-s src_addr**]
[**-q nqueries**] [**-N squeries**] [**-t tos**]
[**-l flow_label**] [**-w waittime**] [**-z sendwait**] [**-UL**] [**-D**]
[**-P proto**] [**--sport=port**] [**-M method**] [**-O mod_options**]
[**--mtu**] [**--back**]
host [**packet_len**]
traceroute6 [*options*]
tcptraceroute [*options*]
lft [*options*]

DESCRIPTION

traceroute tracks the route packets taken from an IP network on their way to a given host. It utilizes the IP protocol's time to live (TTL) field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to the host.

traceroute6 is equivalent to *traceroute* **-6**

tcptraceroute is equivalent to *traceroute* **-T**

lft , the Layer Four Traceroute, performs a TCP traceroute, like *traceroute* **-T** , but attempts to provide compatibility with the original such implementation, also called lft.

The only required parameter is the name or IP address of the destination **host** . The optional **packet_len**'gth is the total size of the probing packet (default 60 bytes for IPv4 and 80 for IPv6). The specified size can be ignored in some situations or increased up to a minimal value.

This program attempts to trace the route an IP packet would follow to some internet host by launching 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 (or TCP reset), which means we got to the host, or hit a max (which defaults to 30 hops). Three probes (by default) 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. The address can be followed by additional information when requested. If the probe answers come from different gateways, the address of each responding system will be printed. If there is no response within a 5.0 seconds (default), an * (asterisk) is printed for that probe.

After the trip time, some additional annotation can be printed: **!H**, **!N**, or **!P** (host, network or protocol unreachable), **!S** (source route failed), **!F** (fragmentation needed), **!X** (communication administratively prohibited), **!V** (host precedence violation), **!C** (precedence cutoff in effect), or **!<num>** (ICMP unreachable code <num>). If almost all the probes result in some kind of unreachable, traceroute will give up and exit.

We don't want the destination host to process the UDP probe packets, so the destination port is set to an unlikely value (you can change it with the **-p** flag). There is no such a problem for ICMP or TCP tracerouting (for TCP we use half-open technique, which prevents our probes to be seen by applications on the destination host).

In the modern network environment the traditional traceroute methods can not be always applicable, because of widespread use of firewalls. Such firewalls filter the unlikely UDP ports, or even ICMP echoes. To solve this, some additional tracerouting methods are implemented (including tcp), see **LIST OF AVAILABLE METHODS** below. Such methods try to use particular protocol and source/destination port, in order to bypass firewalls (to be seen by firewalls just as a start of allowed type of a network session).

OPTIONS

--help Print help info and exit.

-4, **-6** Explicitly force IPv4 or IPv6 tracerouting. By default, the program will try to resolve the name given, and choose the appropriate protocol automatically. If resolving a host name returns both IPv4 and IPv6 addresses, *traceroute* will use IPv4.

-I, --icmp
Use ICMP ECHO for probes

-T, --tcp
Use TCP SYN for probes

-d, --debug
Enable socket level debugging (when the Linux kernel supports it)

-F, --dont-fragment
Do not fragment probe packets. (For IPv4 it also sets DF bit, which tells intermediate routers not to fragment remotely as well).

Varying the size of the probing packet by the **packet_len** command line parameter, you can manually obtain information about the MTU of individual network hops. The **--mtu** option (see below) tries to do this automatically.

Note, that non-fragmented features (like **-F** or **--mtu**) work properly since the Linux kernel 2.6.22 only. Before that version, IPv6 was always fragmented, IPv4 could use the once the discovered final mtu only (from the route cache), which can be less than the actual mtu of a device.

-f *first_ttl*, **--first**=*first_ttl*
Specifies with what TTL to start. Defaults to 1.

-g *gateway*, **--gateway**=*gateway*
Tells traceroute to add an IP source routing option to the outgoing packet that tells the network to route the packet through the specified *gateway* (most routers have disabled source routing for security reasons). In general, several *gateway*'s is allowed (comma separated). For IPv6, the form of *num*,*addr*,*addr...* is allowed, where *num* is a route header type (default is type 2). Note the type 0 route header is now deprecated (rfc5095).

-i *interface*, **--interface**=*interface*
Specifies the interface through which *traceroute* should send packets. By default, the interface is selected according to the routing table.

-m *max_ttl*, **--max-hops**=*max_ttl*
Specifies the maximum number of hops (max time-to-live value) *traceroute* will probe. The default is 30.

-N *squeries*, **--sim-queries**=*squeries*
Specifies the number of probe packets sent out simultaneously. Sending several probes concurrently can speed up *traceroute* considerably. The default value is 16.
Note that some routers and hosts can use ICMP rate throttling. In such a situation specifying too large number can lead to loss of some responses.

-n Do not try to map IP addresses to host names when displaying them.

-p *port*, **--port**=*port*
For UDP tracing, specifies the destination port base *traceroute* will use (the destination port number will be incremented by each probe).
For ICMP tracing, specifies the initial ICMP sequence value (incremented by each probe too).
For TCP and others specifies just the (constant) destination port to connect. When using the tcptraceroute wrapper, -p specifies the source port.

-t *tos*, **--tos=***tos*
For IPv4, set the Type of Service (TOS) and Precedence value. Useful values are 16 (low delay) and 8 (high throughput). Note that in order to use some TOS precedence values, you have to be super user.
For IPv6, set the Traffic Control value.

-l *flow_label*, **--flowlabel=***flow_label*
Use specified flow_label for IPv6 packets.

-w *waittime*, **--wait=***waittime*
Set the time (in seconds) to wait for a response to a probe (default 5.0 sec).

-q *nqueries*, **--queries=***nqueries*
Sets the number of probe packets per hop. The 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.

-s *source_addr*, **--source=***source_addr*
Chooses an alternative source address. Note that you must select the address of one of the interfaces. By default, the address of the outgoing interface is used.

-z *sendwait*, **--sendwait=***sendwait*
Minimal time interval between probes (default 0). If the value is more than 10, then it specifies a number in milliseconds, else it is a number of seconds (float point values allowed too). Useful when some routers use rate-limit for ICMP messages.

-e, --extensions
Show ICMP extensions (rfc4884). The general form is *CLASS/TYPE*: followed by a hexadecimal dump. The MPLS (rfc4950) is shown parsed, in a form: **MPLS:L=***label*,**E=***exp_use*,**S=***stack_bottom*,**T=***TTL* (more objects separated by /).

-A, --as-path-lookups
Perform AS path lookups in routing registries and print results directly after the corresponding addresses.

-V, --version
Print the version and exit.

There are additional options intended for advanced usage (such as alternate trace methods etc.):

--sport=*port*
Chooses the source port to use. Implies **-N 1**. Normally source ports (if applicable) are chosen by the system.

--fwmark=*mark*
Set the firewall mark for outgoing packets (since the Linux kernel 2.6.25).

-M *method*, **--module=***name*
Use specified method for traceroute operations. Default traditional udp method has name *default*, icmp (**-I**) and tcp (**-T**) have names *icmp* and *tcp* respectively.
Method-specific options can be passed by **-O .** Most methods have their simple shortcuts, (**-I** means **-M icmp**, etc).

-O *option*, **--options=***options*
Specifies some method-specific option. Several options are separated by comma (or use several **-O** on cmdline). Each method may have its own specific options, or many not have them at all. To print information about available options, use **-O help**.

-U, --udp
Use UDP to particular destination port for tracerouting (instead of increasing the port per each probe). Default port is 53 (dns).

-UL Use UDPLITE for tracerouting (default port is 53).

-D, --dccp
Use DCCP Requests for probes.

-P *protocol*, **--protocol**=*protocol*
Use raw packet of specified protocol for tracerouting. Default protocol is 253 (rfc3692).

--mtu Discover MTU along the path being traced. Implies **-F -N 1**. New*mtu* is prin ted once in a form of **F**=*NUM* at the first probe of a hop which requires such *mtu* to be reached. (Actually, the correspond frag needed icmp message normally is sent by the previous hop).

Note, that some routers might cache once the seen information on a fragmentation. Thus you can receive the final mtu from a closer hop. Try to specify an unusual *tos* by **-t** , this can help for one attempt (then it can be cached there as well).
See **-F** option for more info.

--back Print the number of backward hops when it seems different with the forward direction. This number is guessed in assumption that remote hops send reply packets with initial ttl set to either 64, or 128 or 255 (which seems a common practice). It is printed as a negate value in a form of '-NUM' .

LIST OF AVAILABLE METHODS

In general, a particular traceroute method may have to be chosen by **-M name**, but most of the methods have their simple cmdline switches (you can see them after the method name, if present).

default
The traditional, ancient method of tracerouting. Used by default.

Probe packets are udp datagrams with so-called unlikely destination ports. The unlikely port of the first probe is 33434, then for each next probe it is incremented by one. Since the ports are expected to be unused, the destination host normally returns icmp unreach port as a final response. (Nobody knows what happens when some application listens for such ports, though).

This method is allowed for unprivileged users.

icmp –I
Most usual method for now, which uses icmp echo packets for probes.
If you can ping(8) the destination host, icmp tracerouting is applicable as well.

This method may be allowed for unprivileged users since the kernel 3.0 (IPv4 only), which supports new *dgram icmp* (or *ping*) sockets. To allow such sockets, sysadmin should provide *net/ipv4/ping_group_range* sysctl range to match any group of the user.
Options:

raw Use only raw sockets (the traditional way).
This way is tried first by default (for compatibility reasons), then new dgram icmp sockets as fallback.

dgram
Use only dgram icmp sockets.

tcp –T
Well-known modern method, intended to bypass firewalls.
Uses the constant destination port (default is 80, http).

If some filters are present in the network path, then most probably any unlikely udp ports (as for *default* method) or even icmp echoes (as for *icmp*) are filtered, and whole tracerouting will just stop at such a firewall. To bypass a network filter, we have to use only allowed protocol/port combinations. If we trace for some, say, mailserver, then more likely **-T -p 25** can reach it, even when **-I** can not.

This method uses well-known half-open technique, which prevents applications on the destination

host from seeing our probes at all. Normally, a tcp syn is sent. For non-listened ports we receive tcp reset, and all is done. For active listening ports we receive tcp syn+ack, but answer by tcp reset (instead of expected tcp ack), this way the remote tcp session is dropped even without the application ever taking notice.

There is a couple of options for *tcp* method:

syn,ack,fin,rst,psh,urg,ece,cwr
Sets specified tcp flags for probe packet, in any combination.

flags=*num*
Sets the flags field in the tcp header exactly to *num*.

ecn Send syn packet with tcp flags ECE and CWR (for Explicit Congestion Notification, rfc3168).

sack,timestamps,window_scaling
Use the corresponding tcp header option in the outgoing probe packet.

sysctl Use current sysctl (*/proc/sys/net/**) setting for the tcp header options above and **ecn**. Always set by default, if nothing else specified.

mss=*num*
Use value of *num* for maxseg tcp header option (when **syn**).

info Print tcp flags of final tcp replies when the target host is reached. Allows to determine whether an application listens the port and other useful things.

Default options is **syn,sysctl**.

tcpconn
An initial implementation of tcp method, simple using connect(2) call, which does full tcp session opening. Not recommended for normal use, because a destination application is always affected (and can be confused).

udp **–U**
Use udp datagram with constant destination port (default 53, dns).
Intended to bypass firewall as well.

Note, that unlike in *tcp* method, the correspond application on the destination host **always** receive our probes (with random data), and most can easily be confused by them. Most cases it will not respond to our packets though, so we will never see the final hop in the trace. (Fortunately, it seems that at least dns servers replies with something angry).

This method is allowed for unprivileged users.

udplite **–UL**
Use udplite datagram for probes (with constant destination port, default 53).

This method is allowed for unprivileged users.
Options:

coverage=*num*
Set udplite send coverage to *num*.

dccp **–D**
Use DCCP Request packets for probes (rfc4340).

This method uses the same half-open technique as used for TCP. The default destination port is 33434.

Options:

service=*num*
Set DCCP service code to *num* (default is 1885957735).

raw –P proto

Send raw packet of protocol *proto*.
No protocol-specific headers are used, just IP header only.
Implies **-N 1**.
Options:

protocol=*proto*
Use IP protocol *proto* (default 253).

NOTES

To speed up work, normally several probes are sent simultaneously. On the other hand, it creates a storm of packages, especially in the reply direction. Routers can throttle the rate of icmp responses, and some of replies can be lost. To avoid this, decrease the number of simultaneous probes, or even set it to 1 (like in initial traceroute implementation), i.e. **-N 1**

The final (target) host can drop some of the simultaneous probes, and might even answer only the latest ones. It can lead to extra looks like expired hops near the final hop. We use a smart algorithm to auto-detect such a situation, but if it cannot help in your case, just use **-N 1** too.

For even greater stability you can slow down the program's work by **-z** option, for example use **-z 0.5** for half-second pause between probes.

If some hops report nothing for every method, the last chance to obtain something is to use **ping -R** command (IPv4, and for nearest 8 hops only).

SEE ALSO

ping(8), ping6(8), **tcpdump(8),** netstat(8)