NAME

scamper — parallel Internet measurement utility

SYNOPSIS

scamper [**-?Dv**] [**-c** *command*] [**-p** *pps*] [**-w** *window*] [**-M** *monitorname*]
[**-l** *listname*] [**-L** *listid*] [**-C** *cycleid*] [**-o** *outfile*] [**-F** *firewall*]
[**-H** *holdtime*] [**-n** *nameserver*] [**-d** *debugfile*] [**-e** *pidfile*]
[**-O** *options*]
[**-i** *IPs* | **-I** *cmds* | **-f** *file* | **-P** *[ip:]port* | **-R** *name:port* | **-U** *unix-dom*]

DESCRIPTION

The **scamper** utility provides the ability to execute Internet measurement techniques to IPv4 and IPv6 addresses, in parallel, to fill a specified packets-per-second rate. Currently, **scamper** supports the well-known traceroute and ping techniques, DNS, as well as MDA traceroute, six different alias resolution techniques, most tbit methods, sting, neighbour discovery, HTTP(S), and UDP probes.

scamper has five modes of operation. First, **scamper** can be supplied a list of one or more addresses on the command line with the **-i** option. **scamper** will then execute a command with each of the supplied addresses, in parallel, and output the results as each task completes. Second, **scamper** can be supplied a list of one or more addresses in a listfile, one address per line, using the **-f** option. Third, **scamper** can be supplied a list of one or more complete commands on the command line with the **-I** option. Fourth, **scamper** can be instructed to listen on an IP address and port specified with the **-P** option, or on a unix domain socket specified with the **-U** option, where it can take commands dynamically. Finally, **scamper** can be instructed to connect to a remote host and port specified with the **-R** option, where it will be supplied with commands dynamically.

For most modules, **scamper** must be started as root. It is recommended to set the setuid bit on the scamper binary, and set the ownership of the scamper binary to root. **scamper** will run privilege separated, forking a process that retains root privileges for tasks that require root, while processing network traffic in an unprivileged process running in a `chroot`(2).

The options are as follows:

-? prints a list of command line options and a synopsis of each.

-v causes **scamper** to output version information and exit.

-D With this option set, **scamper** will detach and become a daemon. Use with the **-P** or **-U** options.

-c *command*
specifies the command for **scamper** to use by default. The current choices for this option are:

- **dealias:** use Ally, Mercator, MIDAR, or Radargun-style probing to infer which IP addresses belong to the same system.
- **http:** perform an HTTP request.
- **host:** issue simple DNS queries to a domain name server.
- **neighbourdisc:** issue an IPv4 ARP or IPv6 Neighbour discovery query to determine the layer-2 address of an IP address on the same network.
- **ping:** conduct simple delay measurements with various probe types.
- **trace:** conduct classic and Paris-style traceroute probing, which infers a single path towards a destination.
- **tracelb:** use the multipath discovery algorithm (MDA) to infer the presence of load-balanced paths towards a destination.
- **sniff:** capture a subset of packets arriving at the host using a subset of tcpdump-style filter expressions.

- **sting:** use the sting method to infer one-way packet loss with a TCP receiver.
- **tbit:** use techniques from the TCP behavior inference tool (TBIT) to infer properties of a TCP receiver.
- **udpprobe:** sends UDP probes and waits for responses.

scamper uses trace by default. The options for each of these commands are documented in their own sections of this manual page.

-p *pps*
specifies the target packets-per-second rate for **scamper** to reach. By default, this value is 20.

-w *window*
specifies the maximum number of tasks that may be probed in parallel. A value of zero places no upper limit. By default, zero is used.

-M *monitorname*
specifies the canonical name of machine where **scamper** is run. This value is used when recording the output in a warts output file.

-l *listname*
specifies the name of the list when run from the command line. This value is used when recording the output in a warts output file.

-L *listid*
specifies the numerical id of the list when run from the command line. This value is used when recording the output in a warts output file.

-C *cycleid*
specifies the numerical cycle id to begin with when run from the command line. This value is used when recording the output in a warts output file.

-o *outfile*
specifies the default output file to write measurement results to. By default, stdout is used.

-F *firewall*
specifies that **scamper** may use the firewall in measurements that require it (tbit and sting). **scamper** supports two firewall types: IPFW, and PF. To use the IPFW firewall, pass ipfw:<start>-<end>, where <start> is the first rule **scamper** can use, and <end> is the last. To use the PF firewall, pass pf:<anchor>:<num>, where <anchor> is the anchor for **scamper** to use, and <num> specifies the number of rules **scamper** is allowed to use.

-H *holdtime*
specifies the length of time, in seconds, to prevent a followup task from using the same probe signature as a previous task. The default is five seconds, and can be disabled by passing zero.

-n *nameserver*
specifies the nameserver for **scamper** to use. By default, **scamper** uses the first nameserver specified in /etc/resolv.conf

-d *debugfile*
specifies a filename to write debugging messages to. By default, no debugfile is used, though debugging output is sent to stderr if scamper is built for debugging.

-e *pidfile*
specifies a file to write scamper's process ID to. If scamper is built with privilege separation, the ID of the unprivileged process is written.

-O *options*

allows scamper's behaviour to be further tailored. The options are case insensitive. The current choices for this option are:

- **text:** output results in plain text. Suitable for interactive use.
- **warts:** output results in `warts`(5) format. Suitable for archiving measurement results and for use by researchers as it records details that cannot be easily represented with the text option.
- **warts.gz:** output results in `warts`(5) format, compressed with `gzip`(1).
- **warts.bz2:** output results in `warts`(5) format, compressed with `bzip2`(1).
- **warts.xz:** output results in `warts`(5) format, compressed with `xz`(1).
- **json:** output results in JSON format. Suitable for processing measurement results with a scripting language. A better approach is to output results in `warts`(5) format, and to use `sc_warts2json`(1) to convert warts to JSON. The JSON format is documented in `sc_warts2json`(1).
- **planetlab:** tell scamper it is running on a planetlab system. Necessary to use planetlab's safe raw sockets.
- **rawtcp:** tell scamper to use IPPROTO_RAW socket to send IPv4 TCP probes, rather than a datalink socket.
- **ICMP-rxerr:** tell scamper to use IP_RECVERR or IPV6_RECVERR to receive ICMP responses, rather than raw sockets. This is useful on Linux systems that have these sockets, and scamper does not have the permissions to obtain a raw socket. This option currently only works with the trace command.
- **select:** tell scamper to use `select`(2) rather than `poll`(2)
- **kqueue:** tell scamper to use `kqueue`(2) rather than `poll`(2) on systems where `kqueue`(2) is available.
- **epoll:** tell scamper to use `epoll`(7) rather than `poll`(2) on systems where `epoll`(7) is available.
- **ring:** tell scamper to create a memory-mapped ring buffer to receive datalink responses. This significantly improves scamper performance and accuracy on Linux systems with other non-scamper traffic. See `packet`(7) for further details.
- **noring:** tell scamper to not create a memory-mapped ring buffer to receive datalink responses.
- **dl-any:** tell scamper to use a single socket to receive datalink responses on Linux. This prevents datalink transmission at this time. See `packet`(7) for further details.
- **dyn-filter:** tell scamper to use dynamic BPF filters to filter datalink responses in the kernel, rather than a static BPF filter that passes all traffic through. This facility is available on Linux, and BSD systems with BIOCSETFNR available, such as MacOS and FreeBSD. See `packet`(7) or `bpf`(4) for further details.
- **cmdfile:** the input file consists of complete commands.
- **noinitndc:** do not initialise the neighbour discovery cache.
- **debugfileappend:** append to the debugfile specified with the **-d** option. The default is to truncate the debugfile.
- **notls-remote:** do not use TLS when establishing a connection with the remove controller specified with the **-R** option.
- **notls:** do not use TLS anywhere in scamper, including tbit.
- **cafile=file:** load the CA certificates in the specified file into scamper, instead of the default certificates.
- **client-certfile=file:** load the certificate in the specified file into scamper and present it to the remote controller for client authentication.
- **client-privfile=file:** load the private key in the specified file into scamper and use it for client authentication with the remote controller.

-i *IP 1..N*
specifies a list of one or more addresses to probe, on the command line, using the command specified with the **-c** option.

-f *listfile*
specifies the input file to read for target addresses, one per line, and uses the command specified with the **-c** option on each.

-I *cmds*
specifies a list of one or more complete commands, including target addresses, for scamper to execute.

-P *[ip:]port*
specifies that **scamper** provide a control socket listening on the specified IP address and port on the local host. If an IP address is not specified, **scamper** will bind to the port specified on the loopback address.

-R *name:port*
specifies that **scamper** connects to a specified remote host and port to receive commands.

-U *unix domain socket*
specifies that **scamper** provide a control socket listening on the specified socket in the unix domain.

TRACE OPTIONS

The trace command is used for conducting classic and Paris-style traceroute probing, which infers a single path towards a destination. The following variations of the traceroute(8) options are available:

trace [**-MQT**] [**-c** *confidence*] [**-d** *dport*] [**-f** *firsthop*] [**-g** *gaplimit*] [**-G** *gapaction*] [**-H** *wait-probe-hop*] [**-l** *loops*] [**-m** *maxttl*] [**-N** *squeries*] [**-o** *offset*] [**-O** *option*] [**-p** *payload*] [**-P** *method*] [**-q** *attempts*] [**-r** *rtraddr*] [**-s** *sport*] [**-S** *srcaddr*] [**-t** *tos*] [**-U** *userid*] [**-w** *wait-timeout*] [**-W** *wait-probe*] [**-z** *gss-entry*] [**-Z** *lss-name*]

-c *confidence*
specifies that a hop should be probed to a specified confidence level (95% or 99%) to be sure the trace has seen all interfaces that will reply for that hop.

-d *dport*
specifies the base destination port value to use for UDP-based and TCP-based traceroute methods. For ICMP-Paris, this option sets the ICMP checksum value.

-f *firsthop*
specifies the TTL or HLIM value to begin probing with. By default, a first hop of one is used.

-g *gaplimit*
specifies the number of unresponsive hops permitted until a check is made to see if the destination will respond. By default, a gap limit of 5 hops is used. Setting the gap limit to 0 disables the gap limit, but doing this is not recommended.

-G *gapaction*
specifies what should happen if the gaplimit condition is met. A value of 1 (default) means halt probing, while a value of 2 means send last-ditch probes.

-H *wait-probe-hop*
specifies the minimum time to wait, in seconds, between sending consecutive probes that use the same TTL value. By default, the next probe is sent as soon as possible after receiving a response to a probe with a given TTL value. The limit is 2 seconds.

-l *loops*
specifies the maximum number of loops permitted until probing stops. By default, a value of one is used. A value of zero disables loop checking.

-m *maxttl*
specifies the maximum TTL or HLIM value that will be probed. By default, there is no restriction, apart from the 255 hops that the Internet protocols allow.

-M specifies that path MTU discovery (PMTUD) should be attempted for the path when the initial traceroute completes. **scamper** will not conduct PMTUD unless it is probing a responsive destination, as otherwise there is no way to distinguish all packets being lost from just big packets (larger than MTU) being lost.

-N *squeries*
specifies the number of consecutive hops that may have an outstanding probe. By default, only one hop may have an outstanding probe. Increasing the number of outstanding probes will allow traceroutes to complete faster, at the expense of sending unnecessary probes. The number of outstanding probes must be less than the gaplimit.

-o *offset*
specifies the fragmentation offset to use in probes. By default, no offset is used.

-O *option*
specifies optional arguments to use. The current choices for this option are:

- **const-payload:** specifies that the payload should not be altered in order to arrive at a desired UDP checksum value for probe/response matching.
- **dl:** specifies that the datalink socket should be used to timestamp sent and received packets. For apparent UDP responses to UDP-Paris traceroute probes, **scamper** will assume that any UDP response was for the last sent probe, as it has no other way of determining which probe the reply might be for.
- **dtree-noback:** specifies that the traceroute should not do backwards probing when using doubletree.
- **ptr:** lookup hostnames for intermediate traceroute hops.
- **raw:** send IPv4 TCP probes using a raw socket, rather than a datalink interface.

-p *payload*
specifies the payload, in hexadecimal, to use as a base in probes. The first 2 bytes of the payload may be modified to accomplish ICMP-Paris and UDP-Paris traceroute, unless the traceroute command was specified with the const-payload option.

-P *method*
specifies the traceroute method to use. **scamper** currently supports five different probe methods: UDP, ICMP, UDP-Paris, ICMP-Paris, TCP, and TCP-ACK. Note: scamper uses UDP-Paris by default, and these options are case insensitive.

-q *attempts*
specifies the maximum number of attempts to obtain a response per hop. By default, a value of two is used.

-Q specifies that all allocated probes are sent, regardless of how many responses have been received.

-r *rtraddr*
specifies the IP address of the router to use.

-s *sport*
specifies the source port value to use. For ICMP-based methods, this option specifies the ICMP identifier to use. By default, **scamper** uses a value it derives from the process ID, but can be told

to obtain a port from the operating system by specifying zero.

-S *srcaddr*
specifies the source address to use in probes. The address cannot be spoofed.

-t *tos*
specifies the value to set in the IP ToS/DSCP + ECN byte. By default, this byte is set to zero.

-T specifies that time exceeded messages from the destination do not cause the trace to be defined as reaching the destination.

-U *userid*
specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of traceroute.

-w *wait-timeout*
specifies how long to wait, in seconds, for a reply. By default, a value of 5 is used.

-W *wait-probe*
specifies the minimum time to wait, in 10s of milliseconds, between sending consecutive probes. By default the next probe is sent as soon as possible.

-z *gss-entry*
specifies an IP address to halt probing when encountered; used with the double-tree algorithm.

-Z *lss-name*
specifies the name of the local stop set to use when determining when to halt probing backwards; used with the double-tree algorithm.

PING OPTIONS

The ping command is used for conducting simple delay measurements with various probe types. The following variations of the ping(8) options are available:

ping [**-R**] [**-A** *TCP-ack*] [**-b** *payload-size*] [**-B** *payload*] [**-c** *probecount*] [**-C** *ICMP-sum*] [**-d** *dport*] [**-F** *sport*] [**-i** *wait-probe*] [**-m** *ttl*] [**-M** *MTU*] [**-o** *replycount*] [**-O** *options*] [**-p** *pattern*] [**-P** *method*] [**-r** *rtraddr*] [**-s** *size*] [**-S** *srcaddr*] [**-T** *timestamp*] [**-U** *userid*] [**-W** *wait-timeout*] [**-z** *tos*]

-A *TCP-ack*
specifies the number to use in the acknowledgement field of the TCP header when using the TCP-ack, tcp-ack-sport, and TCP-synack methods, or the number to use in the sequence number field of the TCP header when using the TCP-syn, TCP-syn-sport, and TCP-rst methods.

-B *payload-size*
specifies, in bytes, the size of the payload to include in each probe. By default, **scamper** uses 56 bytes for ICMP echo probes, 12 bytes for UDP probes, 44 bytes for ICMP time probes, and zero bytes for TCP probes.

-B *payload*
specifies, in a hexadecimal string, the payload to include in each probe.

-c *probecount*
specifies the number of probes to send before exiting. By default, a value of 4 is used.

-C *ICMP-sum*
specifies the ICMP checksum to use when sending a probe. The payload of each probe will be manipulated so that the checksum is valid.

−d *dport*
specifies the destination port to use in each TCP/UDP probe, and the first ICMP sequence number to use in ICMP probes.

−F *sport*
specifies the source port to use in each TCP/UDP probe, and the ICMP ID to use in ICMP probes. By default, **scamper** uses a value it derives from the process ID, but can be told to obtain a port from the operating system by specifying zero.

−i *wait-probe*
specifies the length of time to wait, in seconds, between probes. By default, a value of 1 is used.

−m *ttl*
specifies the TTL value to use for outgoing packets. By default, a value of 64 is used.

−M *MTU*
specifies a pseudo MTU value. If the response packet is larger than the pseudo MTU, an ICMP packet too big (PTB) message is sent.

−o *replycount*
specifies the number of replies required at which time probing may cease. By default, all probes are sent.

−O *options*
The current choices for this option are:

- **dl:** specifies that the datalink socket should be used to timestamp sent and received packets.
- **nosrc:** specifies that the real address of the host should not be embedded in the payload of the packet when the spoof option is used.
- **raw:** send IPv4 TCP probes using a raw socket, rather than a datalink interface.
- **sockrx:** specifies that responses should be collected from a socket, rather than from a datalink interface.
- **spoof:** specifies that the source address is to be spoofed according to the address specified with the **−S** option. The address scamper would otherwise use as the source address is embedded in the payload of the probe unless the nosrc option is used.
- **tbt:** specifies that the goal of the ping is to obtain fragmented responses, so that the **−c** option specifies how many packets to send, and the **−o** option specifies how many fragmented responses are desired.

−p *pattern*
specifies the pattern, in hex, to use in probes. Up to 16 bytes may be specified. By default, each probe's bytes are zeroed.

−P *method*
specifies the type of ping packets to send. By default, ICMP echo requests are sent. Choices are: ICMP-echo, ICMP-time, TCP-syn, TCP-ack, TCP-ack-sport, TCP-synack, TCP-rst, TCP-syn-sport, UDP, UDP-dport, and UDP-sport, and these options are case insensitive.

−r *rtraddr*
specifies the IP address of the router to use.

−R specifies that the record route IP option should be used.

−s *size*
specifies the size of the probes to send. The probe size includes the length of the IP and ICMP headers.

-S *srcaddr*
specifies the source address to use in probes. The address can be spoofed if **-O** *spoof* is included.

-T *timestamp*
specifies that an IP timestamp option be included. The timestamp option can either be: tsprespec where IP addresses of devices of interest can be specified; tsonly, where timestamps are embedded by devices but no IP addresses are included; and tsandaddr, where timestamps and IP addresses are included by devices in the path. See the examples section for more information.

-U *userid*
specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of ping.

-W *wait-timeout*
specifies how long to wait for responses after the last ping is sent. By default this is one second.

-z *tos*
specifies the value to use in the IPv4 ToS/DSCP + ECN byte. By default, this byte is set to zero.

DEALIAS OPTIONS

The dealias command is used to send probes for the purpose of alias resolution. It supports mercator, where aliases are inferred if a router uses a different address when sending an ICMP response; ally, where aliases are inferred if a sequence of probes sent to alternating IP addresses yields responses with incrementing, interleaved IP-ID values; radargun, midarest, and midardisc, where probes are sent to a set of IP addresses in multiple rounds and aliases are inferred by post-processing the results; prefixscan, where an alias is searched in a prefix for a specified IP address; and bump, where two addresses believed to be aliases are probed in an effort to force their IP-ID values out of sequence. The following options are available for the **scamper** dealias command:

dealias [**-@** *start-time*] [**-f** *fudge*] [**-m** *method*] [**-o** *replyc*] [**-O** *option*] [**-p** *probedef*] [**-q** *attempts*] [**-r** *wait-round*] [**-S** *schedule*] [**-U** *userid*] [**-w** *wait-timeout*] [**-W** *wait-probe*] [**-x** *exclude*]

-@ *start-time*
specifies the time, in seconds since the Unix epoch, when the measurement should start. Valid for midardisc.

-f *fudge*
specifies a fudge factor for alias matching. Defaults to 200. Valid for ally, bump, and prefixscan.

-m *method*
specifies which method to use for alias resolution. Valid options are: ally, bump, mercator, midardisc, midarest, prefixscan, and radargun. These options are case insensitive.

-o *replyc*
specifies how many replies to wait for. Only valid for prefixscan.

-O *option*
allows alias resolution behaviour to be further tailored. The current choices for this option are:

- **inseq:** where IP-ID values are required to be strictly in sequence (with no tolerance for packet reordering)
- **shuffle:** randomise the order of probes sent each round; only valid for radargun probing.
- **nobs:** do not allow for byte swapped IP-ID values in responses. Valid for ally and prefixscan.

-p *probedef*
specifies a definition for a probe. Possible options are:

-c *sum*
specifies what ICMP checksum to use for ICMP probes. The payload of the probe will be altered appropriately.

-d *dst-port*
specifies the destination port of the probe. Defaults to 33435.

-F *src-port*
specifies the source port of the probe. By default, **scamper** uses a value it derives from the process ID.

-i *IP*
specifies the destination IP address of the probe.

-M *mtu*
specifies the pseudo MTU to use when soliciting fragmented responses.

-P *method*
specifies which method to use for the probe. Valid options are: UDP, UDP-dport, TCP-ack, TCP-ack-sport, TCP-syn-sport, and ICMP-echo, and these options are case insensitive.

-s *size*
specifies the size of the probes to send.

-t *ttl*
specifies the IP time to live of the probe.

The mercator method accepts one probe definition; ally accepts up to two probe definitions; prefixscan expects one probe definition; radargun expects at least one probe definition; midardisc and midarest expect at least two probe definitions; bump expects two probe definitions.

-q *attempts*
specifies how many times a probe should be retried if it does not obtain a useful response.

-r *wait-round*
specifies how many milliseconds to wait between probing rounds with radargun.

-S *schedule*
specifies information for a single round in midardisc. A schedule item is colon separated, as follows: seconds-since-start:begin-index:end-index. The seconds-since-start field can include fractions of a second. The begin-index and end-index identify the begin and end probedef indexes for the round. There must be at least two rounds in midardisc. The length of each round is determined by when the next round is due to begin; the length of the final round is the same as the length of the penultimate round.

-U *userid*
specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of dealias.

-w *wait-timeout*
specifies how long to wait in seconds for a reply from the remote host.

-W *wait-probe*
specifies how long to wait in milliseconds between probes.

-x *exclude*
specifies an IP address to exclude when using the prefixscan method. May be specified multiple times to exclude multiple addresses.

HOST OPTIONS

The host command can issue requests to a domain name server. The following options are available for the **scamper** host command:

host [**-r**] [**-R** *retry-count*] [**-s** *server-ip*] [**-t** *type*] [**-U** *userid*] [**-W** *wait*]

-r specifies that this query is a non-recursive query. The default is to issue a recursive query.

-R *retry-count*
specifies the number of retries until giving up. The default is to not send any retries.

-s *server-ip*
specifies the IP address of the name server to query instead of the default nameserver.

-t *type*
specifies the DNS query type. The type argument can be one of the following: A, AAAA, PTR, MX, NS, SOA. The default is A if a name is queried, or a PTR if an IP address is queried.

-U *userid*
specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of host.

-W *wait*
specifies the number of seconds to wait for a response. The default is to wait for five seconds.

NEIGHBOUR DISCOVERY OPTIONS

The neighbourdisc command attempts to find the layer-2 address of a given IP address using IPv4 ARP or IPv6 Neighbour Discovery. The following options are available for the **scamper** neighbourdisc command:

neighbourdisc [**-FQ**] [**-i** *interface*] [**-o** *reply-count*] [**-q** *attempts*] [**-w** *wait*]

-F specifies that we only want the first response.

-Q specifies that we want to send all attempts.

-i *interface*
specifies the name of the interface to use for neighbour discovery.

-o *reply-count*
specifies how many replies we wait for.

-q *attempts*
specifies how many probes we send out.

-w *wait*
specifies how long to wait between probes in milliseconds. Defaults to 1000.

TBIT OPTIONS

The tbit command can be used to infer TCP behaviour of a specified host. At present, it implements tests to check the ability of the host to respond to ICMP Packet Too Big messages, respond to Explicit Congestion Notification, test Selective Acknowledgement behaviour, the Initial Congestion Window, and resilience to Blind Attacks. The following options are available for the **scamper** tbit command:

tbit [**-t** *type*] [**-p** *app*] [**-d** *dport*] [**-s** *sport*] [**-b** *ASN*] [**-f** *cookie*] [**-m** *mss*] [**-M** *mtu*] [**-o** *offset*] [**-O** *option*] [**-P** *ptbsrc*] [**-q** *attempts*] [**-S** *srcaddr*]

[**-T** *ttl*] [**-u** *url*] [**-U** *userid*] [**-w** *wscale*]

-t *type* specifies which type of testing to use. Valid options are: pmtud, ecn, null, sack-rcvr, icw, blind-rst, blind-syn, blind-data, blind-fin.

-p *app* specifies what kind of traffic to generate for testing. Destination port defaults the application standard port. Valid applications are: http, bgp.

-d *dport* specifies the destination port for the packets being sent. Defaults are application-specific.

-s *sport* specifies the source port for the packets being sent. By default, **scamper** uses a value it derives from the process ID.

-b *ASN* specifies the autonomous system number (ASN) that should be used when establishing a BGP session.

-f *cookie* specifies the TCP fast open cookie that should be used when establishing a TCP connection.

-m *mss* specifies the maximum segment size to advertise to the remote host.

-M *mtu* specifies the MTU to use in a Packet Too Big message.

-o *offset* specifies the sequence number offset to use when conducting blind-syn and blind-rst tests, and the acknowledgement number offset to use when conducting a blind-data test.

-O *option* allows tbit behaviour to be further tailored. The current choices for this option are:

- **blackhole:** for PMTUD testing, do not send Packet Too Big messages; this tests to ability of a host to infer a PMTUD blackhole and work around it.
- **tcpts:** advertise support for TCP timestamps when establishing a TCP connection. If the peer supports TCP timestamps, embed timestamps in data packets.
- **ipts-syn:** use the timestamp IP option in a SYN packet when attempting to establish a TCP connection.
- **iprr-syn:** use the record-route IP option in a SYN packet when attempting to establish a TCP connection.
- **ipqs-syn:** use the quick-start IP option in a SYN packet when attempting to establish a TCP connection.
- **sack:** advertise support for TCP selective acknowledgements (SACK) when establishing a TCP connection.
- **fo:** advertise support for TCP fast open using the official IANA number assigned for fast open.
- **fo-exp:** advertise support for TCP fast open using the testing number assigned by IANA for fast open.

-P *ptbsrc* specifies the source address that should be used to send Packet Too Big messages in the pmtud test.

-q *attempts*
specifies the number of attempts to make with each packet to reduce false inferences caused by packet loss.

-S *srcaddr*
specifies the source address that should be used in TCP packets sent by the tbit test.

-T *ttl* specifies the IP time-to-live value that should be used in TCP packets sent by the tbit test.

-u *url* specifies a url to use when using the http application method. If the url starts with https, the tbit test begins with a TLS handshake.

-U *userid* specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of tbit.

-w *wscale* specifies the window scale option to use when establishing the TCP connection.

TRACELB OPTIONS

The tracelb command is used to infer all per-flow load-balanced paths between a source and destination using the multipath discovery algorithm (MDA). The following options are available for the **scamper** tracelb command:

tracelb [**-c** *confidence*] [**-d** *dport*] [**-f** *firsthop*] [**-g** *gaplimit*] [**-O** *option*] [**-P** *method*] [**-q** *attempts*] [**-r** *rtraddr*] [**-Q** *maxprobec*] [**-s** *sport*] [**-t** *tos*] [**-U** *userid*] [**-w** *wait-timeout*] [**-W** *wait-probe*]

-c *confidence*
specifies the level of confidence we want to attain that there are no more parallel load balanced paths at a given hop. Valid values are 95 (default) and 99, for 95% confidence and 99% confidence respectively.

-d *dport* specifies the base destination port to use. Defaults to 33435, the default used by traceroute(8).

-f *firsthop*
specifies how many hops away we should start probing.

-g *gaplimit*
specifies how many consecutive unresponsive hops are permitted before probing down a branch halts. Defaults to three.

-O *option* allows tracelb behaviour to be further tailored. The current choices for this option are:

- **ptr:** do Domain Name System pointer (PTR) record lookups for IP addresses.

-P *method* specifies which method we should use to do the probing. Valid options are: UDP-dport, ICMP-echo, UDP-sport, TCP-sport, and TCP-ack-sport. Note: scamper uses UDP-dport by default, and these options are case insensitive.

-q *attempts*
specifies how many probes we should send in an attempt to receive a reply. Defaults to 2.

-Q *maxprobec*
specifies the maximum number of probes we ever want to send. Defaults to 3000.

-r *rtraddr*
specifies the IP address of the router to use.

-s *sport* specifies to the source port to use when sending probes. By default, **scamper** uses a value it derives from the process ID.

-t *tos* specifies the value for the IP Type-of-service field for outgoing probes. Defaults to 0.

-U *userid* specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of tracelb.

-w *wait-timeout*
specifies in seconds how long to wait for a reply to a probe. Defaults to 5.

-W *wait-probe*
specifies in 1/100ths of seconds how long to wait between probes. Defaults to 25 (i.e. 250ms).

STING OPTIONS

The sting command is used to infer one-way loss using an algorithm with TCP probes. It requires the firewall be enabled in scamper using the **-F** option. The following options are available for the **scamper** sting command:

sting [**-c** *count*] [**-d** *dport*] [**-f** *distribution*] [**-h** *request*] [**-H** *hole*] [**-i** *inter*] [**-m** *mean*] [**-s** *sport*]

-c *count* specifies the number of samples to make. By default 48 samples are sent, as this value is the current default of the FreeBSD TCP reassembly queue length. Sting 0.7 uses 100 samples.

-d *dport* specifies the base destination port to use. Defaults to 80, the default port used by the HTTP protocol.

-f *distribution*
specifies the delay distribution of samples. By default a uniform distribution is constructed. Other distributions are currently not implemented in scamper's implementation of sting.

-h *request*
specifies the default request to make. Currently not implemented.

-H *hole* specifies the size of the initial hole left in the request. The default is 3 bytes, the same as sting-0.7.

-i *inter* specifies the inter-phase delay between data seeding and hole filling, in milliseconds. By default, sting waits 2000ms between phases.

-m *mean* specifies the mean rate to send packets in the data phase, in milliseconds. By default, sting waits 100ms between probes.

-s *sport* specifies to the source port to use when sending probes. By default, **scamper** uses a value it derives from the process ID.

SNIFF OPTIONS

The sniff command is used to capture packets matching a specific signature. At present, the only supported signature is ICMP echo packets with a specific ID value, or packets containing such a quote. The following options are available for the **scamper** sniff command:

sting [**-c** *limit-pktc*] [**-G** *limit-time*] [**-S** *ipaddr*] [**-U** *userid*] <expression>

-c *limit-pktc*
specifies the maximum number of packets to capture. By default, sniff will return after it has captured 100 packets; the maximum value is 5000 packets.

-G *limit-time*
specifies the maximum time, in seconds, to capture packets. By default, sniff will return after it has listened for 60 seconds; the limit is 20 minutes.

-S *ipaddr* specifies the IP address that packets must arrive using. scamper uses the IP address to identify the appropriate interface to listen for packets.

-U *userid* specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of sniff.

The sole supported expression is icmp[icmpid] == X, where X is the ICMP-ID to select.

HTTP OPTIONS

The http command is used to conduct an HTTP or HTTPS query. It saves the results of an HTTPS query unencrypted. The following options are available for the **scamper** http command:

http [**-H** *header*] [**-m** *max-time*] [**-O** *option*] [**-u** *url*] [**-U** *userid*]

-H *header*
: specifies a header to include in the request. You can specify multiple headers by including multiple header options. You cannot override the Host header, which is derived from the URL parameter.

-m *max-time*
: specifies the maximum length of time, in seconds, for the http measurement to run. By default, this value is 60 seconds. The max-time value cannot be less than 1 second or greater than 60 seconds.

-O *option*
: specifies optional arguments to use. The current choices for this option are:

- **insecure:** specifies that no validation of a presented TLS certificate should take place.

-u *url*
: specifies the URL for the HTTP request. This is a mandatory option for the http command. If the url starts with https, the http test begins with a TLS handshake. The URL can override the default destination port for the scheme. Note that the http test will not do a DNS lookup for any name embedded in the URL; the http test must include the IP address to connect to as a mandatory parameter after specifying other http options.

-U *userid*
: specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of http.

UDPPROBE OPTIONS

The udpprobe command is used to send UDP packets and wait for UDP responses. The following options are available for the **scamper** udpprobe command:

udpprobe [**-c** *probec*] [**-d** *dport*] [**-o** *stopc*] [**-O** *option*] [**-p** *payload*] [**-S** *srcaddr*] [**-U** *userid*] [**-w** *wait-timeout*] [**-W** *wait-probe*]

-c *probec*
: specifies the number of probes to send before exiting. By default, one probe is sent.

-d *dport*
: specifies the destination port to send the probe to. This is a mandatory parameter.

-o *stopc*
: specifies the number of replies required at which time probing may cease. By default, probing stops after the first reply.

-O *option*
: specifies optional arguments to use. The current choices for this option are:

- **exitfirst:** specifies that udpprobe should exit when it receives the first response, rather than when the wait-timeout period is over.

-p *payload*
: specifies the payload of the probe to include. This is a mandatory parameter. The payload is specified in hexadecimal.

-S *srcaddr*
: specifies the source address to use in probes. The address cannot be spoofed.

-U *userid*
: specifies an unsigned integer to include with the data collected; the meaning of the user-id is entirely up to the user and has no effect on the behaviour of udpprobe.

-w *wait-timeout*
specifies how long to wait, in seconds, for responses. By default, a value of 2 seconds is used.

-W *wait-probe*
specifies the minimum time to wait, in seconds, between sending consecutive probes. By default, a value of 1 second is used.

DATA COLLECTION FEATURES

scamper has three data output formats. The first is a human-readable format suitable for one-off data collection and measurement. The second, known as **warts**, is a binary format that records much more meta-data and is more precise than the human-readable format. The third format is json, which contains most of the meta-data and is almost as precise as the warts(5) format. **scamper** produces text output by default, but will produce the other formats if the user specifies them using **-O** *warts* or **-O** *json*, or if the output file name's suffix is one of these strings. **scamper** can also produce warts files compressed with gz, bz2, or xz at run-time, if **scamper** is linked against zlib(3), libbz2, or liblzma.

scamper is designed for Internet-scale measurement, where large lists of targets are supplied for probing. **scamper** has the ability to probe multiple lists simultaneously, with each having a mix rate that specifies the priority of the list.

When writing output to a warts(5) file, **scamper** records details of the list and cycle that each measurement task belongs to.

CONTROL SOCKET

When started with a **-P** or **-U** option, **scamper** allows inter-process communication via a TCP socket bound to the supplied port on the local host, or a unix domain socket bound to the supplied location in the file system. These sockets are useful for controlling the operation of a long-lived **scamper** process. A client may interact with scamper by using telnet(1) to open a connection to the supplied port, or nc(1) to open a unix domain socket.

The following control socket commands are available.

exit
The exit command closes the current control socket connection.

attach *argument ...*
The attach command changes how **scamper** accepts and replies to commands, returning results straight over the control socket. See **ATTACH** section below for details on which commands **scamper** accepts.

cycle_id *uint32_t*
The cycle identifer value to use in the cycle record. By default, **scamper** uses 1.

descr *string*
The descriptive string to use in the list record. By default, **scamper** does not include a descriptive string.

format *string*
The data format requested. The two options are warts and json. The warts binary data is uuencoded. The json is plain json text. By default, **scamper** uses warts.

list_id *uint32_t*
The list identifer value to use in the list record. By default, **scamper** uses 0.

monitor *string*
The monitor string to use in the list record. By default, **scamper** uses the value passed using the **-M** *monitorname* parameter to **scamper**, or the hostname if the user did not supply a moni-

tor name.

name *string*
The name string to use in the list record. By default, **scamper** uses a string derived from the socket connected to **scamper**.

priority *uint32_t*
The mixing priority of this source, relative to other scamper sources. By default, **scamper** uses a priority of 1 -- all sources are mixed equally.

get *argument*
The get command returns the current setting for the supplied argument. Valid argument values are: command, monitorname, nameserver, pid, pps, version, window.

set *argument ...*
The set command sets the current setting for the supplied argument. Valid argument values are: command, monitorname, nameserver, pps, window.

source *argument ...*

list ...
The **source list** command provides a listing of all currently defined sources. The optional third *name* parameter restricts the listing to the source specified.

shutdown *argument*
The shutdown argument allows the **scamper** process to be exited cleanly. The following arguments are supported

done
The **shutdown done** command requests that **scamper** shuts down when the current tasks have completed.

flush
The **shutdown flush** command requests that **scamper** flushes all remaining tasks queued with each list, finishes all current tasks, and then shuts down.

now The **shutdown now** command causes **scamper** to shutdown immediately. Unfinished tasks are purged.

cancel
The **shutdown cancel** command cancels any pending shutdown.

ATTACH MODE

In attach mode, none of the usual interactive mode commands are usable. Instead, commands may be entered directly and results will be sent back directly over the control socket. Commands are specified just as they would be with the -I flag for a command-line invocation of **scamper**. Replies are split into lines by single \n characters and have one of the following formats:

ERR ...
A line starting with the 3 characters "ERR" indicates an error has occurred. The rest of the line will contain an error message.

OK *id-num*
A line with the 2 characters "OK" indicates that scamper has accepted the command. **scamper** versions after 20110623 return an id number associated with the command, which allow the task to be halted by subsequently issuing a "halt" instruction.

MORE

A line with just the 4 characters "MORE" indicates that scamper has the capacity to accept more probing commands to run in parallel.

DATA *length id-num*

A line starting with the 4 characters "DATA" indicates the start of result. *length* specifies the number of characters of the data, including newlines. The data is in binary warts format and uuencoded before transmission, unless the user specified the json format in the attach command. *id-num* is the id number associated with the command returned in the OK statement when a command was accepted by **scamper** versions after 20230224.

To exit attached mode the client must send a single line containing "done". To halt a command that has not yet completed, issue a "halt" instruction with the id number returned when the command was accepted as the sole parameter.

EXAMPLES

To use the default traceroute command to trace the path to 192.0.2.1:

```
scamper -i 192.0.2.1
```

To infer Path MTU changes in the network and associate them with a traceroute path:

```
scamper -I "trace -P udp-paris -M 192.0.2.1"
```

To use paris traceroute with ICMP probes, using 3 probes per hop, sending all probes, writing to a specified warts file:

```
scamper -O warts -o file.warts -I "trace -P icmp-paris -q 3 -Q 192.0.2.1"
```

To conduct a traceroute and a ping to two different addresses using the default traceroute and ping parameters, writing to a specified warts file:

```
scamper -O warts -o file.warts -I "trace 192.0.2.1" "ping 192.0.2.2"
```

To ping a series of addresses defined in *filename*, probing each address 10 times:

```
scamper -c "ping -c 10" filename
```

Care must be taken with shell quoting when using commands with multiple levels of quoting, such as when giving a probe description with a dealias command. The following sends UDP probes to alternating IP addresses, one second apart, and requires the IP-ID values returned to be strictly in sequence.

```
scamper -O warts -o ally.warts -I "dealias -O inseq -W 1000 -m ally -p '-P udp -i 192.0.2.1' -p '-P udp -i 192.0.2.4'"
```

Alternatively, the following accomplishes the same, but without specifying the UDP probe method twice.

```
scamper -O warts -o ally.warts -I "dealias -O inseq -W 1000 -m ally -p '-P udp' 192.0.2.1 192.0.2.4"
```

The following command scans 198.51.100.0/28 for a matching alias to 192.0.2.4, but skips 198.51.100.3.

```
scamper -O warts -o prefixscan.warts -I "dealias -O inseq -W 1000 -m prefixscan -p '-P udp' -x 198.51.100.3 192.0.2.4 198.51.100.0/28"
```

The following uses UDP probes to enumerate all per-flow load-balanced paths towards 192.0.2.6 to 99% confidence; it varies the source port with each probe.

```
scamper -I "tracelb -P udp-sport -c 99 192.0.2.6"
```

The following command connects to the remote controller running `sc_remoted`(1) at foo.example.com:31337, loading the CA certificates specified in the file.

```
scamper -R foo.example.com:31337 -O cafile=/etc/ssl/certs/ca-certificates.crt
```

SEE ALSO

`ping`(8), `traceroute`(8), `libscamperctrl`(3), `libscamperfile`(3), `sc_ally`(1), `sc_analysis_dump`(1), `sc_attach`(1), `sc_ipiddump`(1), `sc_filterpolicy`(1), `sc_remoted`(1), `sc_speedtrap`(1), `sc_tbitblind`(1), `sc_tracediff`(1), `sc_uptime`(1), `sc_wartscat`(1), `sc_wartsdump`(1), `sc_warts2json`(1), `sc_warts2pcap`(1), `sc_warts2text`(1), `warts`(5),

S. Savage, *Sting: a TCP-based Network Measurement Tool*, 1999 USENIX Symposium on Internet Technologies and Systems.

R. Govindan and H. Tangmunarunkit, *Heuristics for Internet Map Discovery*, Proc. IEEE INFOCOM 2000.

N. Spring, R. Mahajan, and D. Wetherall, *Measuring ISP topologies with Rocketfuel*, Proc. ACM SIGCOMM 2002.

A. Medina, M. Allman, and S. Floyd, *Measuring the evolution of transport protocols in the Internet*, ACM/SIGCOMM Computer Communication Review.

M. Luckie, K. Cho, and B. Owens, *Inferring and Debugging Path MTU Discovery Failures*, Proc. ACM/SIGCOMM Internet Measurement Conference 2005.

B. Donnet, P. Raoult, T. Friedman, and M. Crovella, *Efficient algorithms for large-scale topology discovery*, Proc. ACM SIGMETRICS 2005.

B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, *Avoiding traceroute anomalies with Paris traceroute*, Proc. ACM/SIGCOMM Internet Measurement Conference 2006.

B. Augustin, T. Friedman, and R. Teixeira, *Measuring Load-balanced Paths in the Internet*, Proc. ACM/SIGCOMM Internet Measurement Conference 2007.

A. Bender, R. Sherwood, and N. Spring, *Fixing Ally's growing pains with velocity modeling*, Proc. ACM/SIGCOMM Internet Measurement Conference 2008.

M. Luckie, *Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet*, Proc. ACM/SIGCOMM Internet Measurement Conference 2010.

R. Beverly, W. Brinkmeyer, M. Luckie, and J.P. Rohrer, *IPv6 Alias Resolution via Induced Fragmentation*, Proc. Passive and Active Measurement Conference 2013.

M. Luckie, R. Beverly, W. Brinkmeyer, and k claffy, *Speedtrap: Internet-scale IPv6 Alias Resolution*, Proc. ACM/SIGCOMM Internet Measurement Conference 2013.

M. Luckie, R. Beverly, T. Wu, M. Allman, and k. claffy, *Resilience of Deployed TCP to Blind Attacks*, Proc. ACM/SIGCOMM Internet Measurement Conference 2015.

J. Czyz, M. Luckie, M. Allman, and M. Bailey, *Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy*, Proc. Network and Distributed Systems Security (NDSS) Conference 2016.

M. Luckie, A. Dhamdhere, B. Huffaker, D. Clark, and k. claffy, *bdrmap: Inference of Borders Between IP Networks*, Proc. ACM/SIGCOMM Internet Measurement Conference 2016.

M. Luckie and R. Beverly, *The Impact of Router Outages on the AS-level Internet*, Proc. ACM/SIGCOMM Conference 2017.

AUTHORS

scamper was written by Matthew Luckie <mjl@luckie.org.nz>. Alistair King contributed an initial implementation of Doubletree and support for memory-mapped ring buffers on Linux; Ben Stasiewicz contributed an initial implementation of TBIT's PMTUD test; Stephen Eichler contributed an initial implementation of TBIT's ECN test; Boris Pfahringer adapted **scamper** to use GNU autotools, modularised the tests, and updated this man page. Brian Hammond of Internap Network Services Corporation provided an initial implementation of scamper's json output format. Tiange Wu contributed an initial implementation of the blind in-window TBIT test, and Robert Beverly contributed BGP protocol support for TBIT.

ACKNOWLEDGEMENTS

scamper development was initially funded by the WIDE project in association with CAIDA. Boris' work was funded by the University of Waikato's Centre for Open Source Innovation.