1

dnsperf

dnsperf is an authoritative-server-specific DNS performance testing tool.

Command Synopsis

```
dnsperf [ -1 ] [ -A ] [ -b bufsize ] [ -c ] [ -d datafile ] [ -D ] [ -e ]
   [ -f family ] [ -h ] [ -H histogram_buckets ] [ -l  limit ]
   [ -p port ] [ -q num_queries ] [ -Q max_qps ] [ -s server_addr ]
   [ -t timeout ] [ -T  histogram_seconds ] [ -u ] [ -v ] [ -y
   name:secret ]
```

Options

Option	Description
`-1`	Instructs *dnsperf* to run through the input file exactly once. This is the default if no time limit is set.
`-A`	Reports the command line arguments passed to *dnsperf* to standard output as part of the final statistics.
`-b bufsize`	Sets the size of the socket's send and receive buffers, in kilobytes. If not specified, the default value is 32 (kB).
`-c`	Prints a count of the number of responses with each DNS RCODE as part of the final statistics.
`-d datafile`	Specifies the input data file. If not specified, *dnsperf* reads from standard input.
`-D`	Sets the DNSSEC OK bit in all packets sent. This also enables EDNS 0, which is required for DNSSEC.
`-e`	Enables EDNS 0, by adding an OPT record to all packets sent.
`-f family`	Specifies the address family used for sending DNS packets. The possible values are • inet—Use IPv4. • inet6—Use IPv6. • any—Either IPv4 or IPv6 as appropriate. If "any" (the default value) is specified, dnsperf uses whichever address family is appropriate for the server to which it is sending packets.
`-h`	Prints a usage statement and exit.
`-H histogram_buckets`	When specified, instructs *dnsperf* to print a histogram showing response latency after completing the run, containing the number of buckets specified by `histogram_buckets`.
`-l limit`	Specifies a time limit for the run, in seconds. **NOTE:** This may cause the input to be read multiple times, or only some of the input to be read. The default behavior is to read the input once, and have no specific time limit.
`-p port`	Sets the port on which the DNS packets are sent. If not specified, the standard DNS port (53) is used.

Table 1-1 dnsperf options

Option	Description
`-q num_queries`	Sets the maximum number of outstanding requests. When this value is reached, *dnsperf* stops sending requests until either responses are received or its requests time out. The default value is 20.
`-Q max_qps`	Limits the number of requests per second. There is no default limit.
`-s server_addr`	Specifies the name or address of the server to which requests will be sent. The default is the loopback address, 127.0.0.1.
`-t timeout`	Specifies the request timeout value, in seconds. After `timeout` is reached, *dnsperf* will no longer wait for a response to a particular request.
`-T histogram_seconds`	When specified, dnsperf prints a histogram showing response latency after completing the run. The histogram will include latencies up to the number of seconds specified by `histogram_seconds`. **NOTE:** This should be used with the -H option.
`-u`	Instructs *dnsperf* to send DNS dynamic update messages, rather than queries. **NOTE:** The format of the input file is different in this case—see *"Constructing a Dynamic Update Input File"* on page 5.
`-v`	Enables verbose mode. The DNS RCODE of each response will be reported to standard output when the response is received.
`-y name:secret`	Adds a TSIG record to all packets sent, using the specified TSIG key `name` and `secret`, where `secret` is expressed as a base-64 encoded string.

Table 1-1 dnsperf options *(continued)*

Operational Considerations

Performance testing at the traffic levels involved is essentially a hard real-time application. At a query rate of 100,000 qps, a 1/100s delay translates into 1000 incoming UDP packets—this is far more than most operating systems can buffer.

Therefore—on the same LAN as the server under test—run *dnsperf* on *its own machine*, ensuring that:

- The machine running *dnsperf* is *at least* as fast as the server being tested—otherwise, it may become a performance bottleneck.
- There are no other applications running on the machine running *dnsperf*.

Bandwidth

dnsperf makes significant demands on bandwidth. The machine running *dnsperf* and the server under test should be connected by a network segment with Gigabit Ethernet (or greater) capacity.

Firewalls

Ensure that no stateful firewalls exist between the caching server and the Internet. As a rule, stateful firewalls can't handle the amount of UDP traffic generated by *dnsperf*, and may skew test results by:

- Dropping packets.
- Locking up.
- Crashing.

Configuring the server

The nameserver under test should be configured as an authoritative server, serving one or more zones that are similar in size (and number) to those the server is expected to serve in production.

Recursion should be disabled for authoritative servers that support it—otherwise, the server's attempts to retrieve glue information from the Internet during testing will slow the server by an unpredictable amount of time. Recursion can be disabled as follows:

- BIND8 and BIND9—Specify `recursion no;` in the options block.
- BIND8 only—Specify `fetch-glue no;` in the options block.

The Query Input File

You need to construct a *dnsperf* input file containing a large and realistic set of queries, on the order of ten thousand to a million. The input file contains one line per query, consisting of a domain name and an RR type name separated by a space. The class of the query is implicitly IN.

When measuring the performance serving non-terminal zones such as the root zone or TLDs, note that such servers spend most of their time providing referral responses, not authoritative answers. Therefore, a realistic input file might consist mostly of queries for type A for names *below*, not at, the delegations present in the zone. For example, when testing the performance of a server config-

ured to be authoritative for the top-level domain *fi.*, which contains delegations for domains like *helsinki.fi* and *turku.fi*, the input file could contain lines like

```
www.turku.fi A
www.helsinki.fi A
```

where the *www* prefix ensures that the server will respond with a referral. Ideally, a realistic proportion of queries for nonexistent domains should be mixed in with those for existing ones, and the lines of the input file should be in a random order.

Constructing a Dynamic Update Input File

To test dynamic update performance, *dnsperf* is run with the `-u` option, and the input file is constructed of blocks of lines describing dynamic update messages. The first line in a block contains the zone name:

```
foo.com
```

Subsequent lines contain prerequisites, if any are required. Prerequisites can specify that a name may or may not exist, an RRset may or may not exist, or an RRset exists and its RDATA matches all specified rdata for that name and type. The keywords "require" and "prohibit" are followed by the appropriate information. All relative names are considered to be relative to the zone name. The following lines show the 5 types of prerequisites.

```
require a
require a A
require a A 1.2.3.4
prohibit x
prohibit x A
```

Subsequent lines contain records to be added, records to be deleted, RRsets to be deleted, or names to be deleted. The keywords "add" or "delete" are followed by the appropriate information. All relative names are considered to be relative to the zone name. The following lines show the 4 types of updates.

```
add x 3600 A 10.1.2.3
delete y A 10.1.2.3
delete z A
delete w
```

Each update message is terminated by a line containing the command:

```
send
```

Running Tests

dnsperf is run specifying the input file using the "-d" option, as in

```
# dnsperf -d input_file -s server
```

Monitoring the Test

The output of *dnsperf* is mostly self-explanatory. Pay attention to the number of dropped packets reported - when running the test over a local Ethernet connection, it should be zero. If one or more packets has been dropped, there may be a problem with the network connection.

In that case, the results should be considered suspect and the test repeated.

Measuring latency

When you specify the `-H` option, the statistics output includes a histogram (bar chart) showing the distribution of response latencies. Latencies for authoritative servers are typically negligible by comparison to network and queueing delays.

A typical histogram might contain 50 buckets representing latencies from 0 to 1 second in 20 millisecond increments.

To print such a histogram, execute *dnsperf* with the options

```
# dnsperf -d input_file -s server -H 50 -T 1
```

If you are interested in responses that arrive several seconds late, you can get a 10-second histogram using

```
dnsperf -d input_file -s server -H 50 -T 10
```

The lengths of the bars in the bar chart are normalized such that the widest bar is 60 characters wide, to allow the chart to be displayed in an 80-column window or printed on an 80-column printer. Responses are classified into successes and failures; successes are represented by "#" characters and failures by "-" in the bars. The number of success/failure responses is also printed next to each bar.

The average latency is also printed; it takes into account both successes and failures. Note that requests that got no response at all will not be included in the latency graph—this may unfairly skew the average latency in favor of servers that drop requests (or respond with an error later than the *dnsperf* timeout) over those from which an error response is received.

TSIG support

When the `-y` option is used, TSIG records is added to all requests. The command-line option specifies the key *`name`* and *`secret`*, where the *`secret`* is encoded in base64.

Throttling support

The `-Q` option limits the numbers of requests per second to a specific number, which is computed over the entire run of *dnsperf*.