A TCP DNS perf tool

fdupont@isc.org

DNS-OARC, Dublin, May 2013

History

- December 2011: urgent development of a DHCP perf tool in 2 weeks
- ICANN TCP DNS performance measurement for new gTLDs
- No suitable tool (only UDP)
- Ixia box ordered but not delivered
- Conclusion: need a TCP DNS perf tool for Yesterday!

ICANN ? requirements

- New gTLD Applicant Guidebook 4 June 2012, Module 5, 5.2.2, TCP support, pages 5-6 and 5-7
- http://newgtlds.icann.org/en/applicants/agb/guidebook-full-04jun12-en.pdf
- others: rate limiting, no EDNS0/TC fallback

Requirements

Self-certification documentation shall include data on load capacity, latency and external network reachability.

Load capacity shall be reported using a table, and a corresponding graph, showing percentage of queries that generated a valid (zone data, NODATA, or NXDOMAIN) response against an increasing number of queries per second generated from local (to the name servers) traffic generators. The table shall include at least 20 data points and loads that will cause up to 10% query loss (either due to connection timeout or connection reset) against a randomly selected subset of servers within the applicant's DNS infrastructure.

Query latency will be reported in milliseconds as measured by DNS probes located just outside the border routers of the physical network hosting the name servers, from a network topology point of view.

Work start

- DHCP is over UDP, not TCP
- 2 phases (DISCOVER/SOLICIT, REQUEST)
- use Linux Real-Time library (vs BSD)
- TCP DNS connection close by the client (vs HTTP by the server)

Kernel tuning

- On the server, usual listen() queue (aka backlog) and port reuse (TIME_WAIT) tuning
- On the client, port reuse tuning and increase the file descriptor limit

Summary (system calls)

- socket(), set not-blocking, connect()
- select()/poll()/epoll()
- getsocketopt() SO_ERROR, send()
- select()/poll()/epoll()
- recv(), close()

Summary (queues)

- Preallocate connection table
- Allocation mark + free queue
- Connecting queue
- Connected queue
- Sent queue

Summary (threads)

- Doesn’t work well in one thread
- poll() works a bit better but only epoll() can scale
- 3 threads:
  - connector thread
  - sender thread
  - receiver thread

Summary (scheduling)

- sender and receiver loop on epoll_wait() and manage timeouts
- scheduling in the connector loop:
  - compute the due date
  - wait (clock_nanosleep) if due in future
  - compute the number of new connections
  - min() with the aggressiveness parameter
  - create new connections

Findings

- The limit is new TCP connections
- The content of query/response doesn't matter
- Not-blocking connect() takes a lot of time!
- Worked hard to put the kernel at its limit
- Interactions between client and server kernels:
  - back pressure from the server on client(s)
  - higher connection rate with no service?

Conclusions

- Get an IXIA!
- No good reasons for the kernel behavior
- Should try with newer kernels (i.e., different threading model inside kernel)

Example

```
% perftcpdns -xaei -r 60000 -d.1,.5 -a 3 149.20...
....
connect: 334677, sent: 270501, received: 260625
embryonics: 64176 (19.2%)
drops: 9876 (3.7%)
total losses: 74052 (22.1%)
local limits: 0, bad connects: 0, connect timeouts: 63634
bad sends: 0, bad recvs: 0, recv timeouts: 8832
too shorts: 0, bad IDs: 0, not responses: 0
rcode counters: noerror: 260625, ...
rates: 31580,25524,24592 (target 60000)
loops: 9782,112891,156683,129366
shortwait: 0,0,3
compconn: 334680, lateconn: 2350
badconn: 0, collconn: 63634, recverr: 0, collsent: 8832
memory: used(2686) / allocated(60000)
RTT: min/avg/max/stddev:  0.723/25.543/646.593/104.274 ms
```

Numbers

- TCP DNS: 30k conn/s, 25k qr/s
- UDP (but similar tool organization) DNS: between 90k and 145k qr/s
- Ixia: to be done

Where to find it

- to be announced