Initial Performance Measurements (Q1 2018)

Thanks to funding from the Open Technology Fund we have started work on some performance measurements for various DNS Privacy implementations. We report our initial results below.

Initial work

The goal of the first phase of work it to get a general understanding of the characteristics of the behaviour of several open source recursive implemetnations under TCP and TLS load. The first measurements look at 2 aspects:

- How does TCP and TLS compare to UDP for a very small number of client connections (1-16)
- For a small number of connections how does performance vary with the number of queries per connection

We concentrate on relative performance rather than absolute values.

Results and configuration files can be found in the Performance Testing git repo.

We also presented this work at RIPE 76: Slides, Video

Further follow up work can be found here: Follow-up Performance Measurements (Q4 2108)

Nameservers

We tested four nameservers:

- Bind 9.12.1 (does not support TLS)
- Unbound 1.7.0
- Knot Resolver 2.3.0
- dnsdist 1.3.0

Test setup

Software

We have used the well known dnsperf from Nominum (now owned by Akamai) for the basis of our testing.

- **dnsperf**: For UDP measurements to original tool from Nominum was used
- **dnsperf-tcp**: For TCP measurements we used a fork from dnsperf with added TCP support. The code is available on github. (This work was funded by a grant from NLnet Foundation).

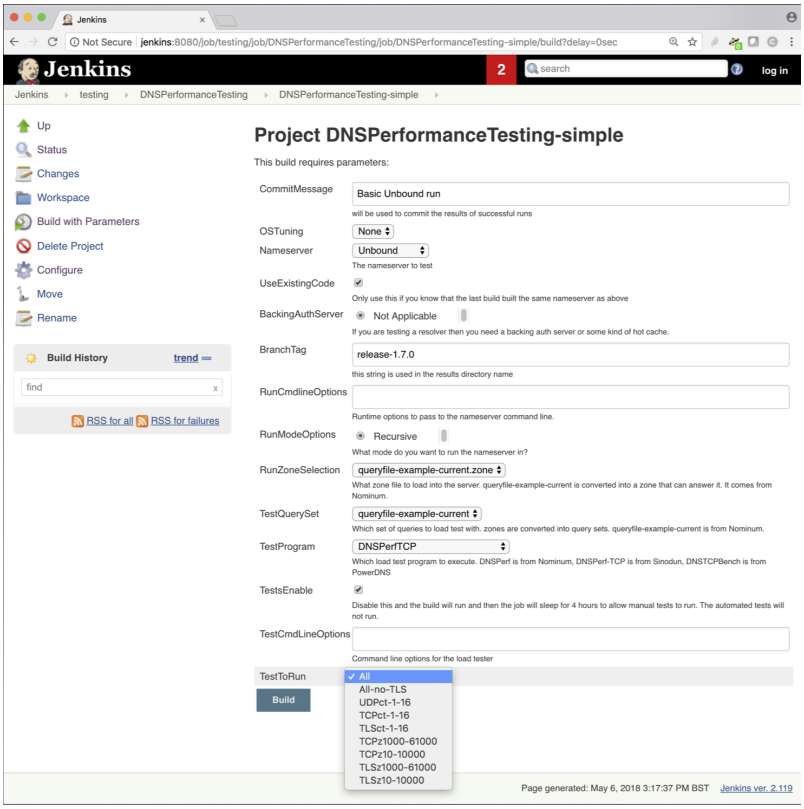
- **dnsperf-tls**: For TLS we had to re-factor dnsperf-tcp somewhat to accommodate TLS support. We implemented this with both OpenSSL and GnuTLS but saw no difference if performance of dnsperf and so we used the version with OpenSSL (TLS 1.2) for the tests below. This code is also available on github. We do note that we see some reduced performance for this version when using TCP and TLS at very low queries per connection (we believe due to limitations in the implementation because of how the dnsperf threading model interacts with TLS read /writes).

For future work we would like to create a bespoke testing tool designed to specifically handle TCP and TLS. We have started a review of existing testing tools here.

We control the test runs from Jenkins and the scripts use a mix of bash, awk and GNUplot to automatically generate plots of the results.

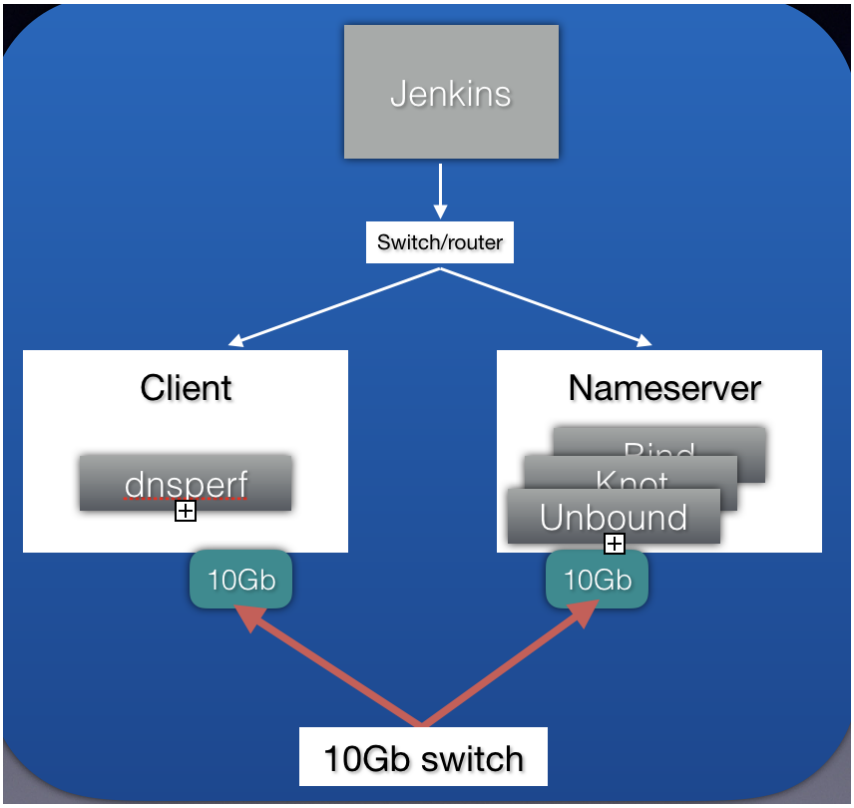
Test parameters

The results are the averages of 5 repeated runs each of 20 seconds with dnsperf set to allow a total of 5000 outstanding queries. We note that with a smaller number of outstanding queries we see quite different results and above this number we see no differences.



Hardware

Our current set up simply consists of one client machine and one nameserver machine connected via a 10Gb switch:



- The servers are both Supermicro Super Servers with a X10DRW-iT Motherboard running BIOS v2.0b. Each has 2 Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz with 8 cores (16 threads).
- They have an Intel Corporation 82574L Gigabit Network Connection used for management and a dual port Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 for load testing. The two 10Gb interfaces are connected to a Netgear M4300-8X8F 10Gb switch.
- The servers are running Ubuntu 18.04 LTS
- The 10Gb ethernet controller (ixgbe) drivers have been upgraded to the latest from Intel https://downloadcenter.intel.com/download/14687/?product=60020 version 5.3.3.

See below for sample results on 18.04 without the spectre/meltdown fixes and 16.04.

OS tuning

Only minimal OS tuning was performed in these first tests:

- The number of file descriptors was raised by adding the following to /etc/security/limits.conf

/etc/security/limits.conf

```
* soft nofile 100000
* soft nofile 100000
```

- The following TCP TIME_WAIT and ephemeral port range changes were made

```
sudo sysctl -w net.ipv4.tcp_fin_timeout=20
sudo sysctl -w net.ipv4.tcp_tw_reuse=1
sudo sysctl -w net.ipv4.ip_local_port_range="1025 65535"
```

Nameserver configuration

The nameserver test machine is also is set up with NSD acting as a root server that is able to answer all the questions in the queryfile-example-current from Nominum. At the start of each test run the recursive server cache was populated buy running the test queries against the nameserver which was configured to fetch the answers from the NSD instance.

The configurations used for each nameserver are available in the git repo.

In each case the nameserver was locked to four cores to ensure the client could saturate the nameserver. For Unbound, Knot resolver and BIND the nameserver was configured to use 4 threads. dnsdist was configured with 4 listeners and it should be noted that for TCP it uses a different threading model where it spawns a separate thread for each TCP connection it handles (creating them as required) so this most probably explains the different behaviour of dnsdist observed below.

Results

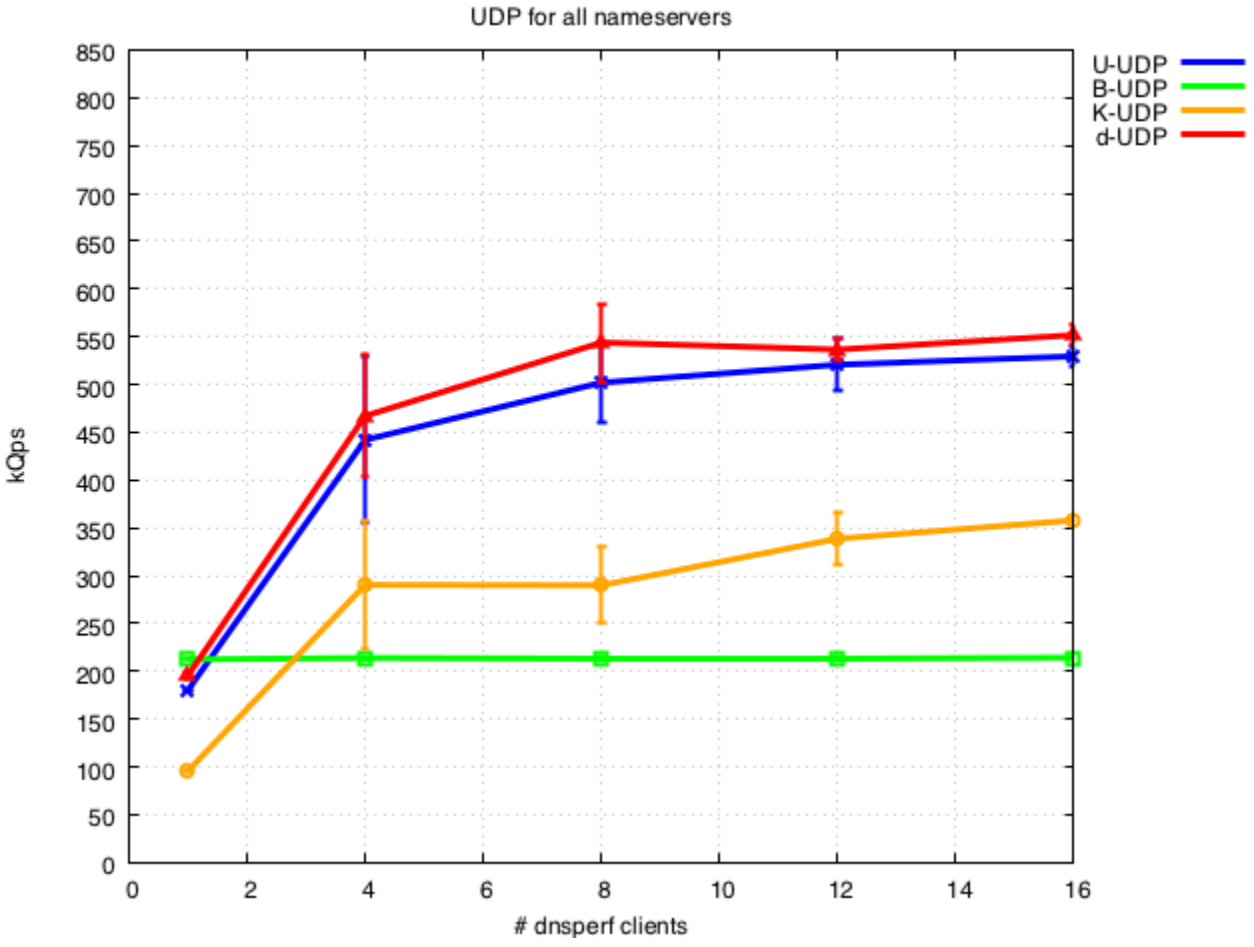
The test results and a full set of plots can be found in the git repo.

Increasing load

In the first test runs UDP, TCP and TLS were used to load the nameserver as the number of dnsperf clients were increased from 1 to 16. For TCP /TLS 20,000 queries per connection were sent so this is simulating persistent connections to measure the throughput of TCP/TLS connections.

UDP only

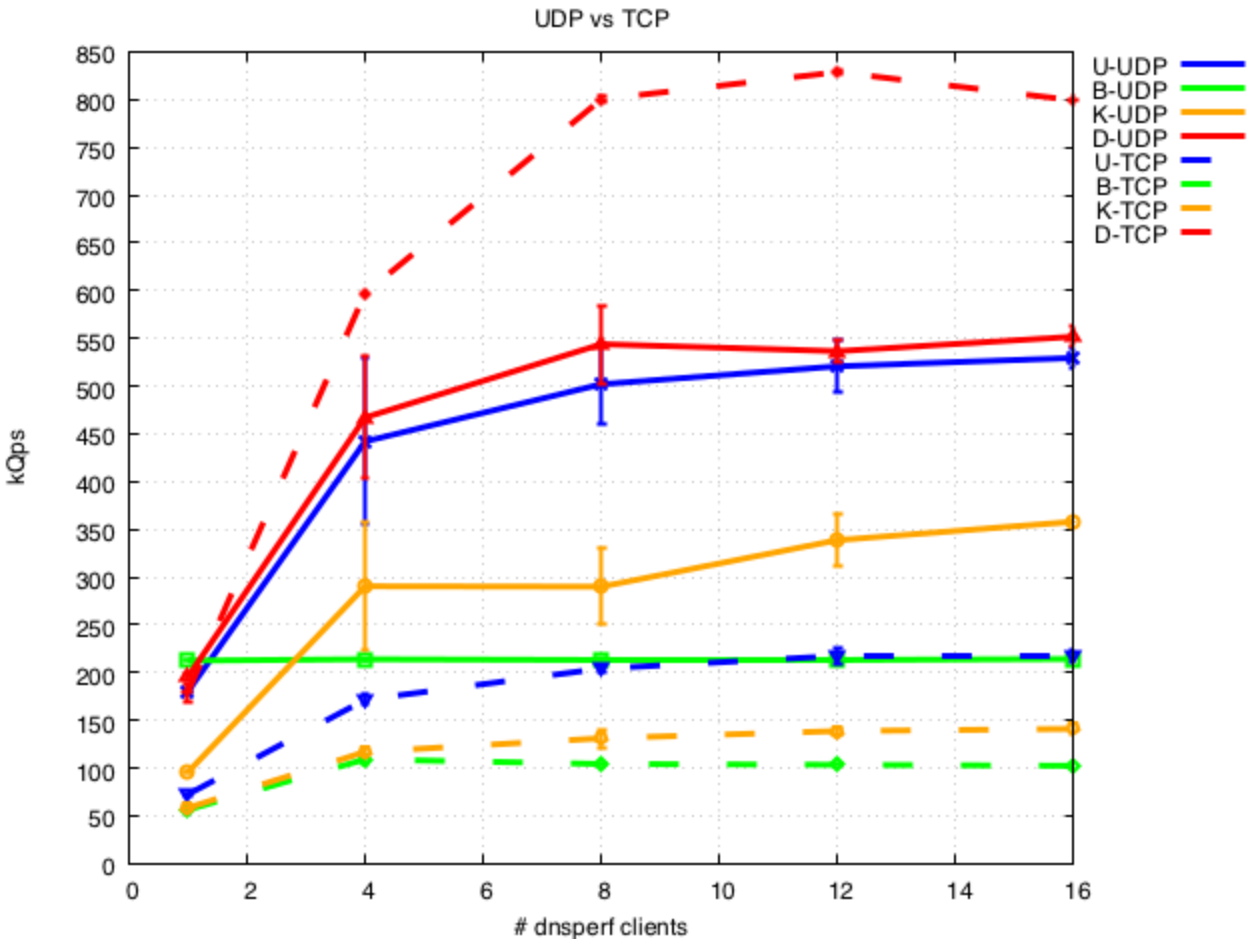
We note that Unbound and dnsdist have the best performance here and BIND and Knot Resolver have lower UPD performance. It is interesting to note how flat the performance of BIND is compared to the other nameservers.



UDP vs TCP

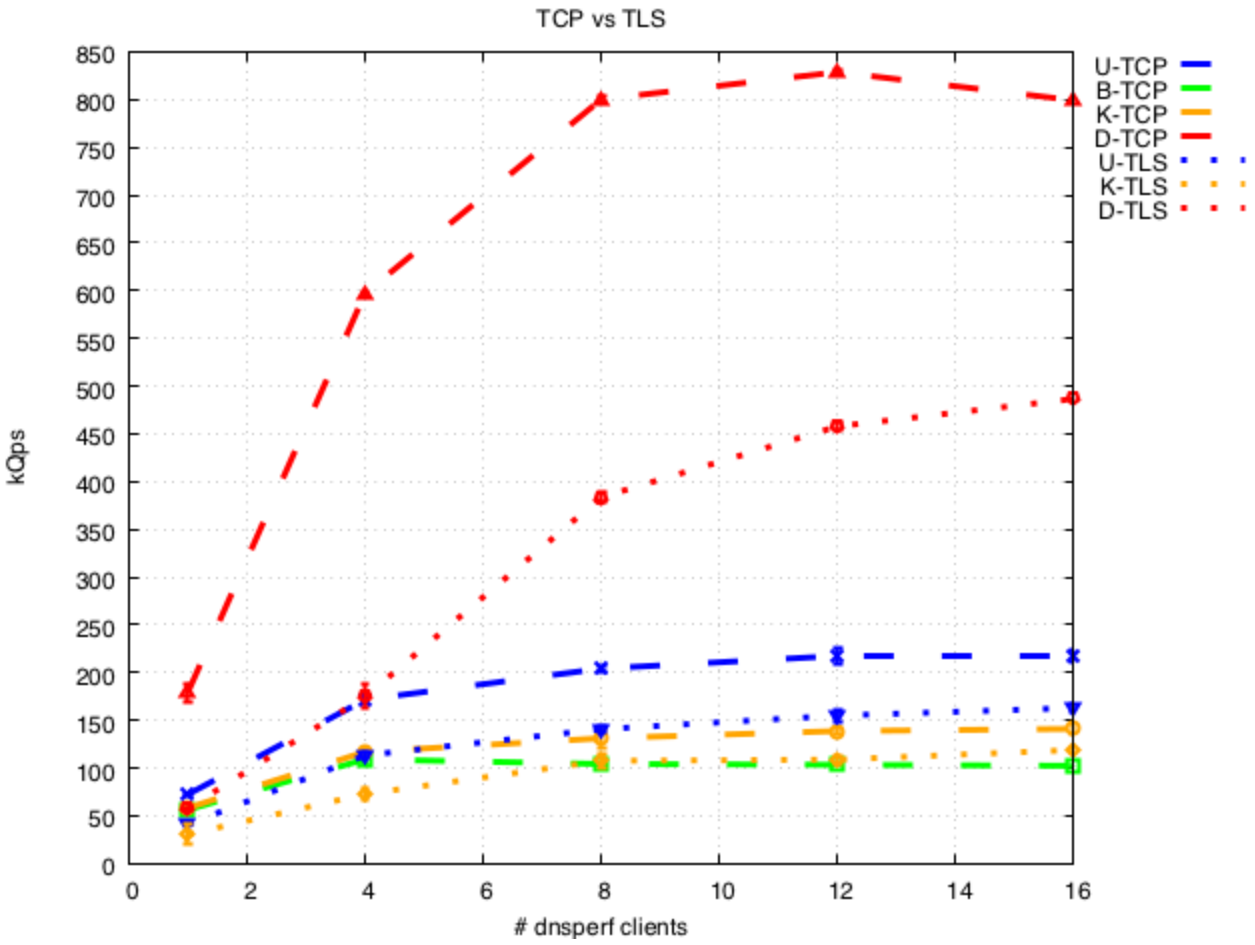
The surprising finding here is that with these specific test conditions dnsdist shows much better TCP performance the UDP performance. But as noted above it uses a very different threading model to the other nameservers. The other nameservers all show a roughly similar decrease in throughput over TCP of around 50%.

Note that Unbound serially processes the TCP queries on a connection (i.e. it processes and sends a response to a TCP query before processing the next query on the connection) instead of processing queries concurrently (as specified in RFC7766). Work is in progress to change this behaviour.



TCP vs TLS

(BIND does not support TLS yet). As expected there is a drop in performance for TLS compared to TCP, of roughly 20-30%.

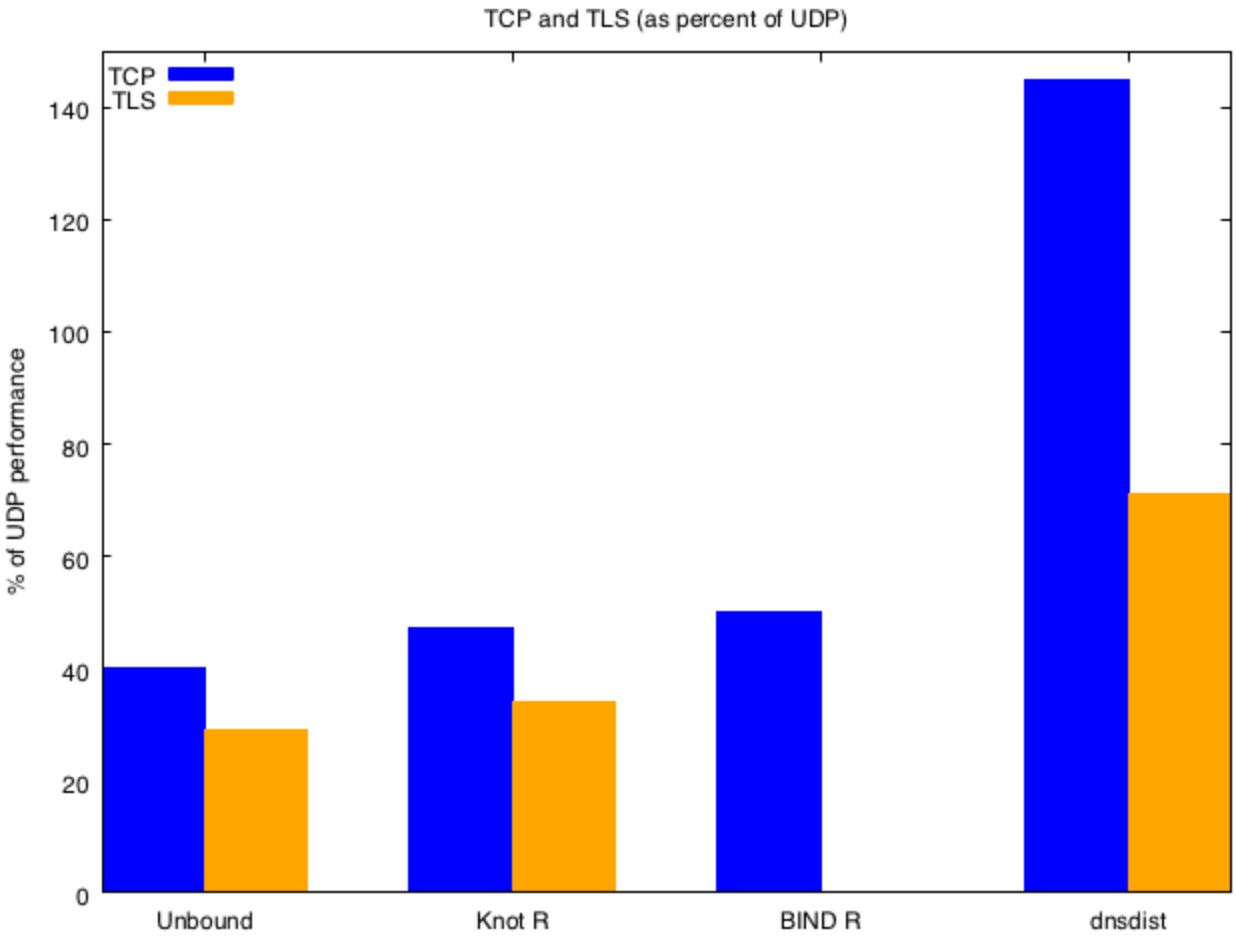


TCP and TLS as a percentage of UDP

The plot below show the TCP/TLS performance as a percentage of the UDP performance of the nameserver for loading with 8 clients (20,000 queries per connection for TCP/TLS).

Whilst dnsdist show the best TCP performance, the relative drop in performance for TLS compared to TCP is significant.

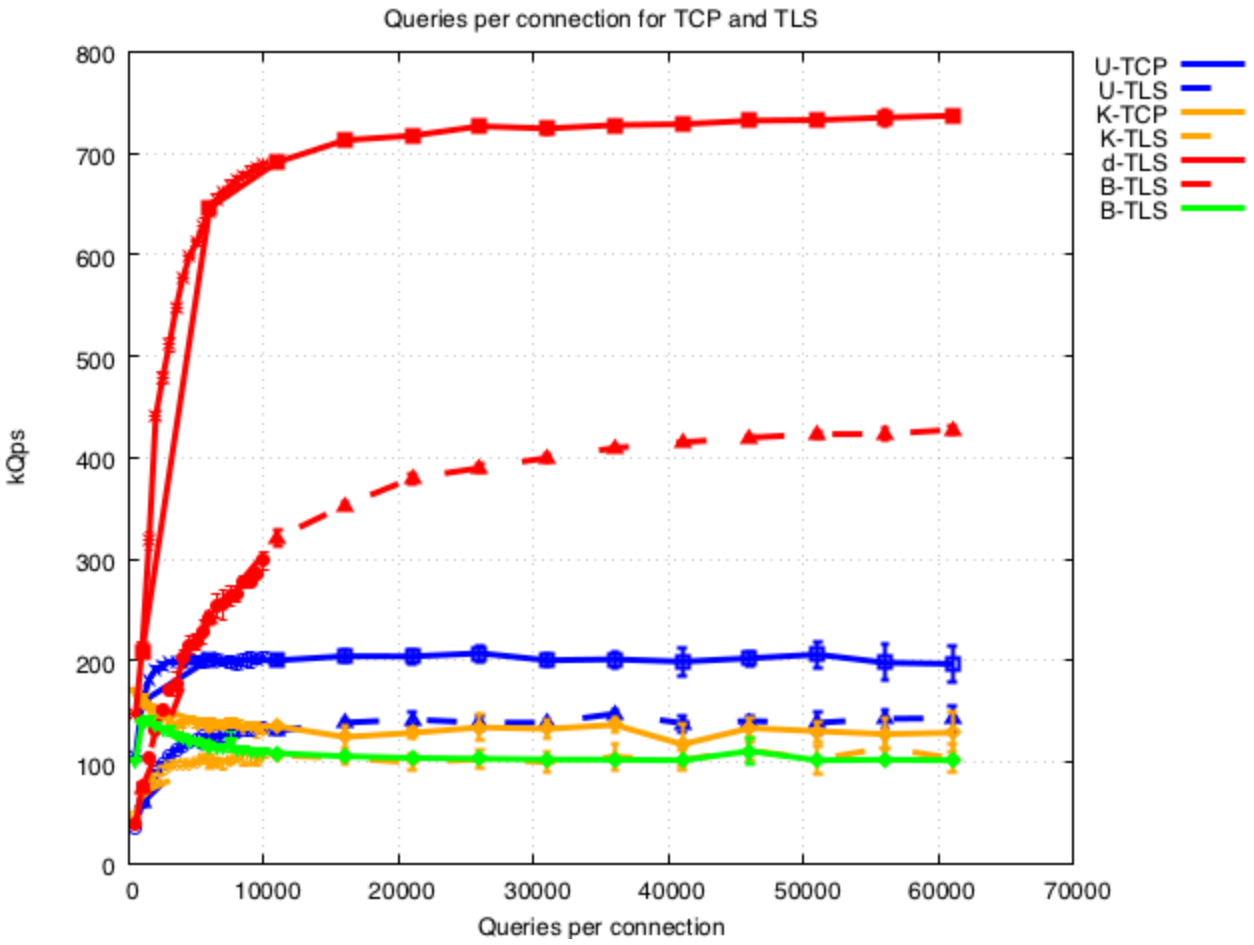
Unbound has the lowest TCP/TLS performance due to the lack of concurrent processing, whereas BIND and Knot have comparable UDP vs TCP performance.



Vary queries per connection

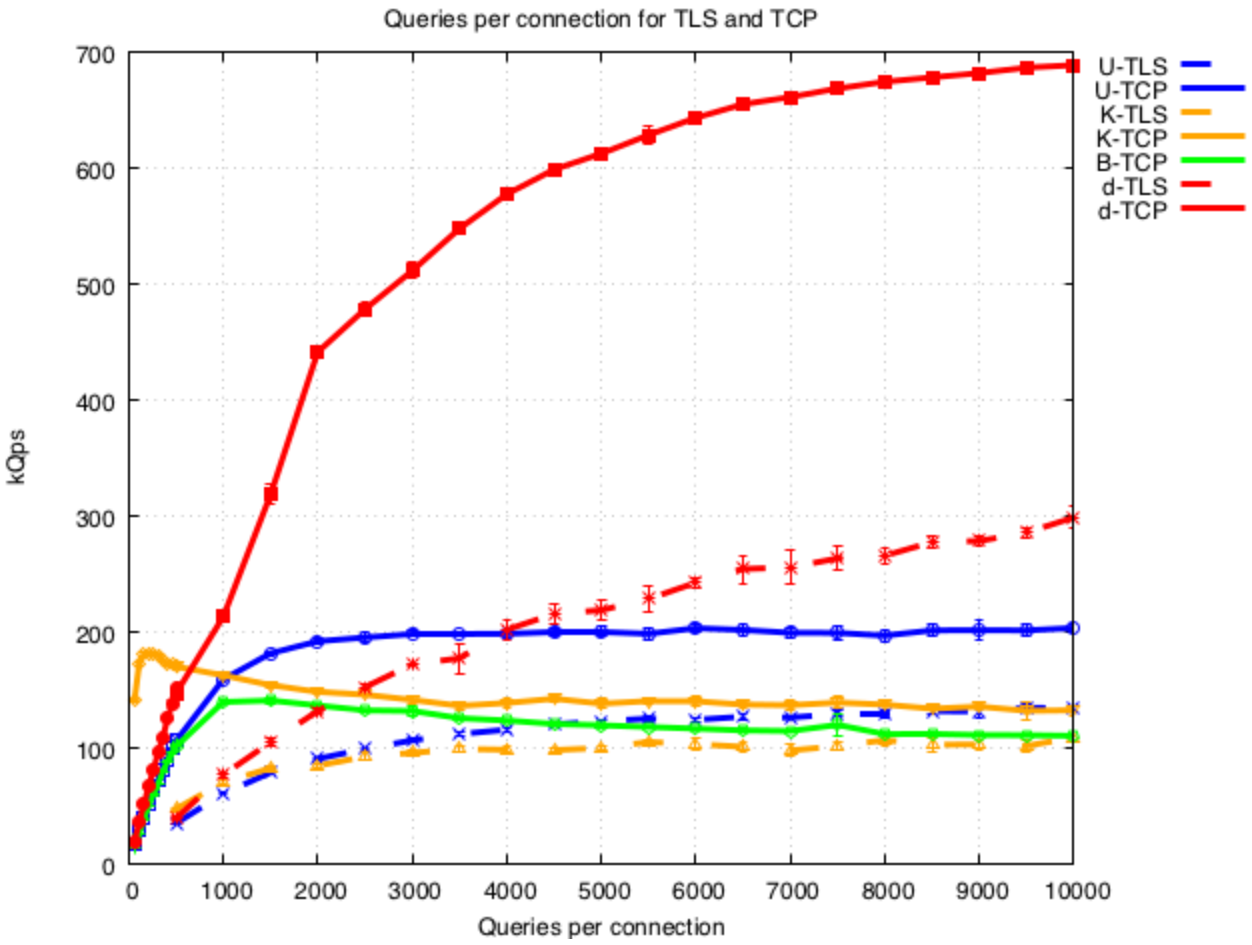
Less than 60,000 queries per connection

Most of the nameservers show steady performance above 10,000 q/conn as would be expected.



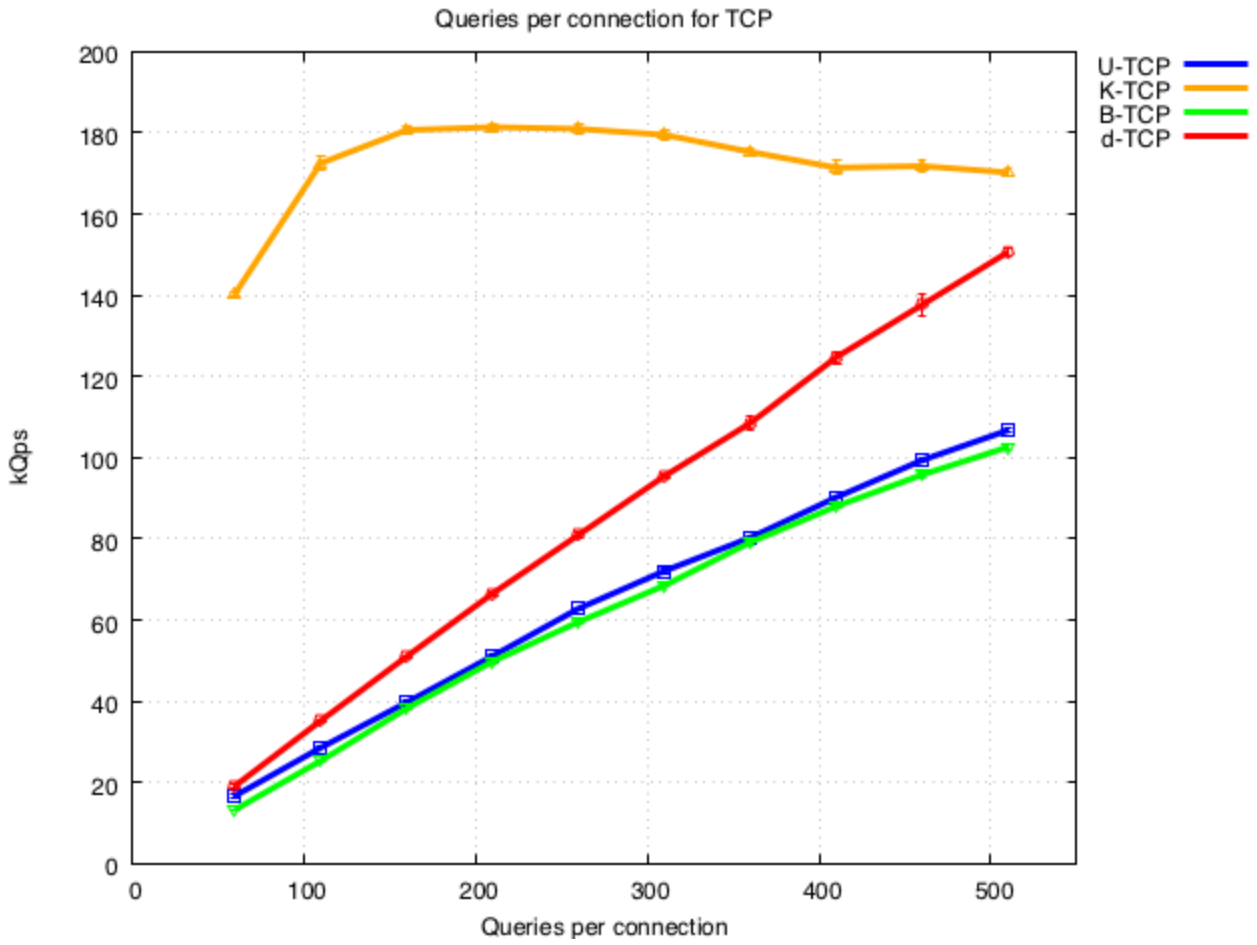
Less than 10,000 queries per connection

dnsdist has a relatively smooth decline in performance in this range, Unbound and BIND are much flatter and decline only below ~1500 q/con. It is noted that Knot appears very flat for TCP until much lower q/conn (but not for TLS - we think this is due to limitations in our test tool and are investigating).

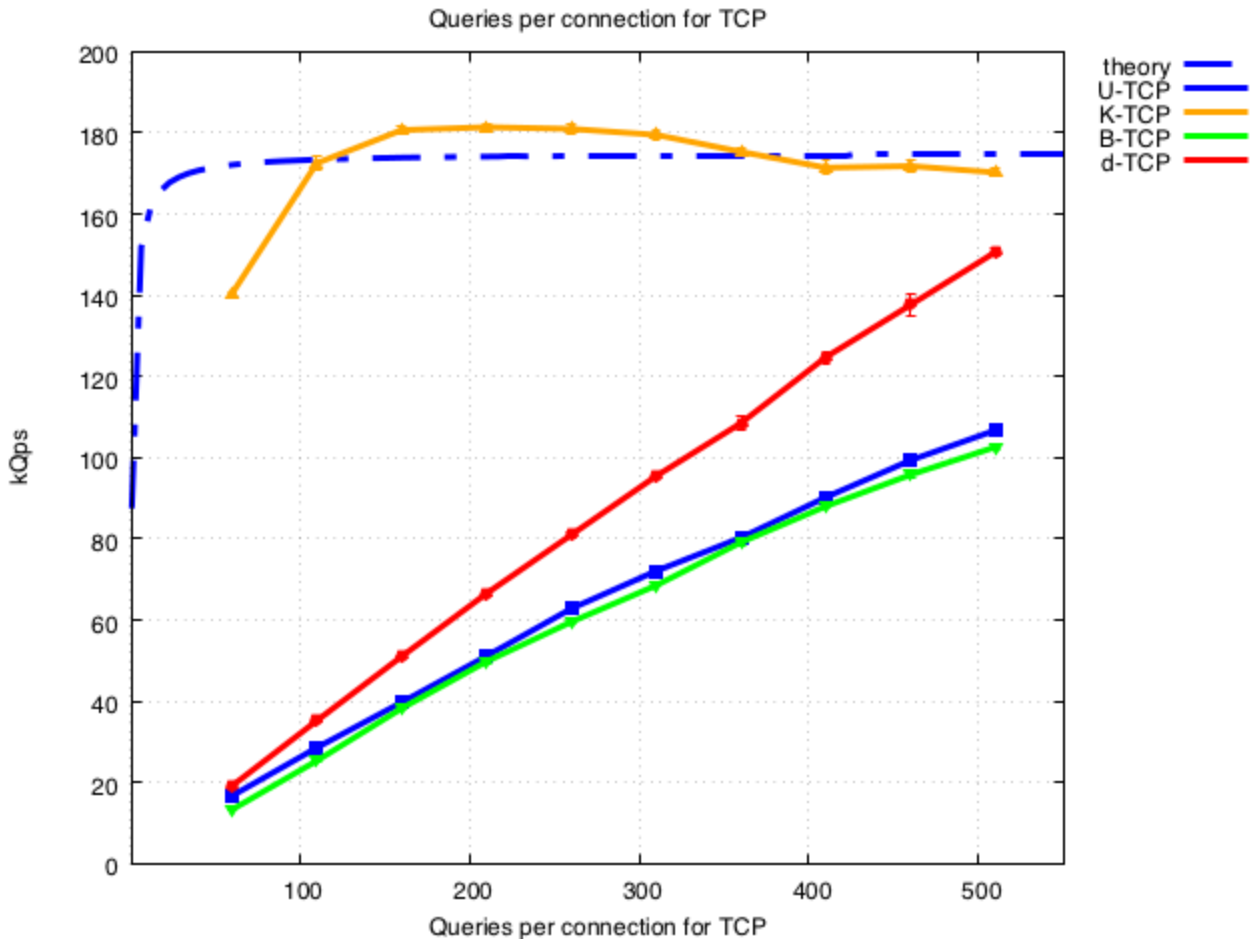


Less than 500 queries per connection

By examining the region below 500 in detail we see a marked difference for Knot Resolver compared to the other nameservers. Unfortunately our current test configuration encounters issues at low numbers of q/con so we don't have numbers for below 60q/con. We are actively working to resolve this and extend these plots.



The plot below includes a purely theoretical extrapolation of Knot resolver performance if amortization of the TCP handshake is the only factor that degrades performance (handshakes are amortized at (N+1)/N where N is the number of queries per connection).



Key conclusions

- Typically saw TCP was 40-50% of UDP throughput (except dnsdist, which uses a different threading model and performed better)
- TLS was 30-40% of UDP throughput
- Varying the number of queries per connection (including low numbers) shows amortisation of setup occurs between 100 or ~2000 q/conn depending on the nameserver

TODO list

We clearly have more work to do to further investigate the behaviour we seen This includes:

- Drill to lower q/conn for TCP and TLS
- Dig into implementations….
- Experiment with OS and nameserver tuning
- Add more options: TFO, TLS Session Resumption, TLS 1.3,…
- Compare to TLS proxy e.g. nginx, haproxy
- Add concurrent processing to Unbound
- Scale to MANY clients
- Use new/different test tool?

Comments on test stability

We do note that we see variations of the order of 10-15% when performing identical test runs on the same setup at different times (e.g. on following days with not change to the setup).

So each of the the data sets for a given setup is from sequential runs of the test suite for each nameserver for better comparison.