Persistent DNS connections for improved performance

Baptiste Jonglez, Sinan Birbalta, Martin Heusse

Persistent DNS connections for improved performance

Baptiste JONGLEZ, Sinan BIRBALTA, Martin HEUSSE
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

THE CASE FOR DNS-OVER-TLS

DNS runs primarily over **UDP** today, but this raises **security and privacy issues**:

1. No source address validation, enabling **massive DDoS reflection attacks**;
2. **Unreliable** IP fragmentation, which is required when sending large DNS answers;
3. **No encryption**, allowing pervasive surveillance and censorship through DNS traffic manipulation.

DNS-over-TLS and **DNS-over-HTTPS** [2], as recently specified, both solve all these problems. We focus on DNS-over-TLS between stub resolver and recursive resolver, and study its performance at scale.

CHALLENGES

Deploying DNS-over-TLS at scale comes with a number of challenges. The cost of opening a new TLS connection is significant, both in terms of latency and CPU load. **Persistent connections** thus become necessary to amortize this cost. However, it means that servers need to manage tens of thousands of TLS connections, consuming resources.
We focus on the following questions:

1. **What is the impact of DNS-over-TLS on query latency?**
2. **What is the performance impact of large-scale DNS-over-TLS on recursive resolvers?**

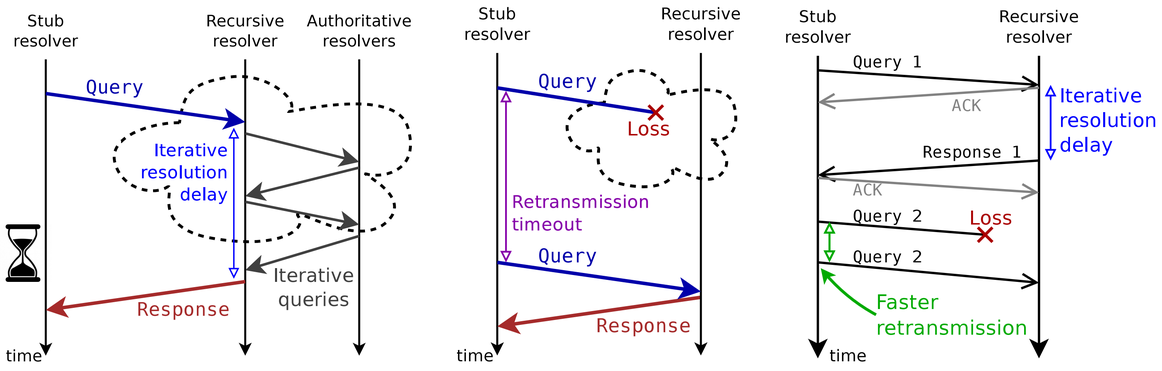
This poster presents experiments which answer these two questions.

PERSISTENT CONNECTIONS CAN IMPROVE LATENCY

- **Without losses**: once established, a persistent connection provides the same latency as UDP, 1 RTT [3].
- **With losses**: UDP performs very poorly and exhibits **large latency**. A stub resolver has to guess whether the iterative resolution process is taking longer than expected (left), or if a message was lost and a retransmission is required (middle). Most implementation use a simple timeout:

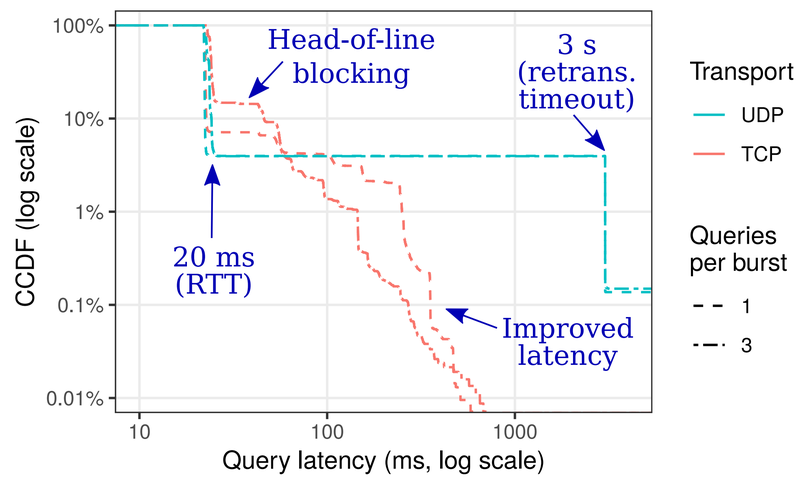
Stub resolver	First retransmission timeout	Retransmission strategy	Time before application failure
Glibc 2.24 (Linux)	5 seconds	Constant interval	40 seconds
Bionic (android 7.1.2)	5 seconds	Constant interval	30 seconds
Windows 10	1 second	Exponential backoff	12 seconds
OS X 10.13.6	1 second	Exponential backoff	30 seconds
IOS 11.4	1 second	Exponential backoff	30 seconds

With a persistent connection, the stub resolver can measure the RTT and retransmit much more efficiently (right):



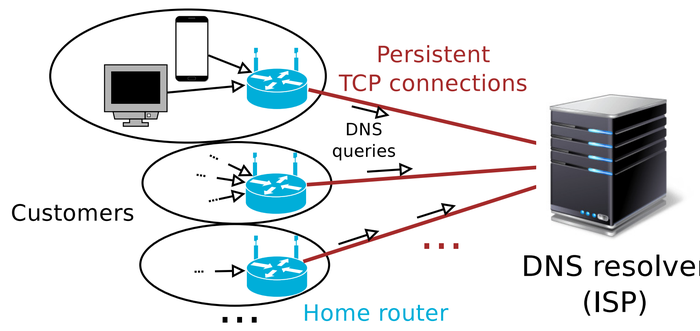
LATENCY MEASUREMENTS: UDP VS. TCP

We use a small testbed to measure DNS latency with various RTT and loss values, and compare UDP with TCP. The figure below is taken for 20 ms RTT and 2% of packet loss in each direction. TCP significantly **improves the worst-case latency** (the 99th percentile is reduced from 3020 ms to 145 ms), at the cost of a small amount of **head-of-line blocking**.



LARGE-SCALE DNS PERFORMANCE EVALUATION

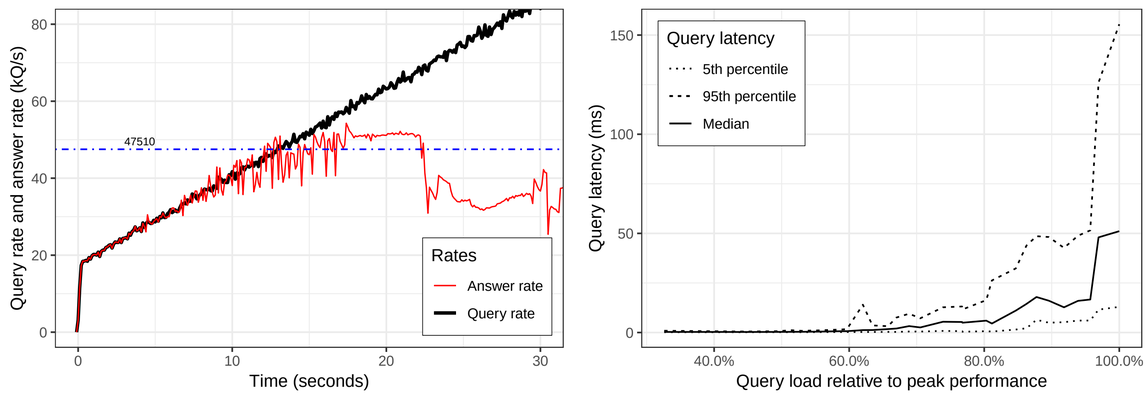
We assess the performance of large-scale DNS-over-TCP/TLS using the following model:



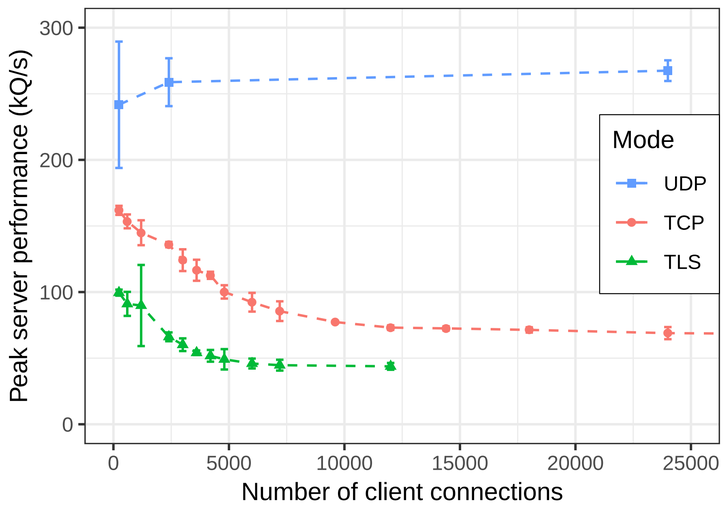
We setup a large-scale testbed on Grid'5000 [1] to implement this deployment model. Each physical machine hosts several Virtual Machines (VM) acting as clients. A dedicated machine hosts the DNS resolver.

MEASURING RECURSIVE RESOLVER PERFORMANCE

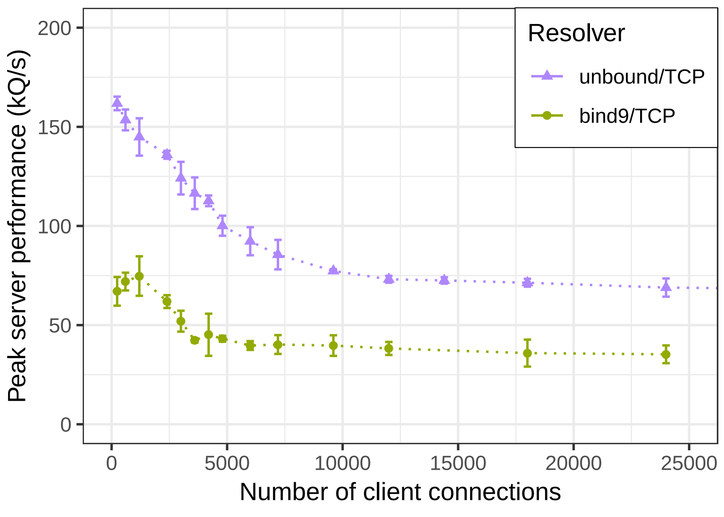
To measure the peak performance of a recursive resolver, we increase the query load linearly, until the resolver cannot cope with the load permanently (left, peak performance in blue, `bind` 9.13.3 with 1 thread, 3000 TCP clients). Latency remains acceptable even when approaching the saturation point (right, same experiment). The 95th percentile stays below 20 ms even at 80% load.



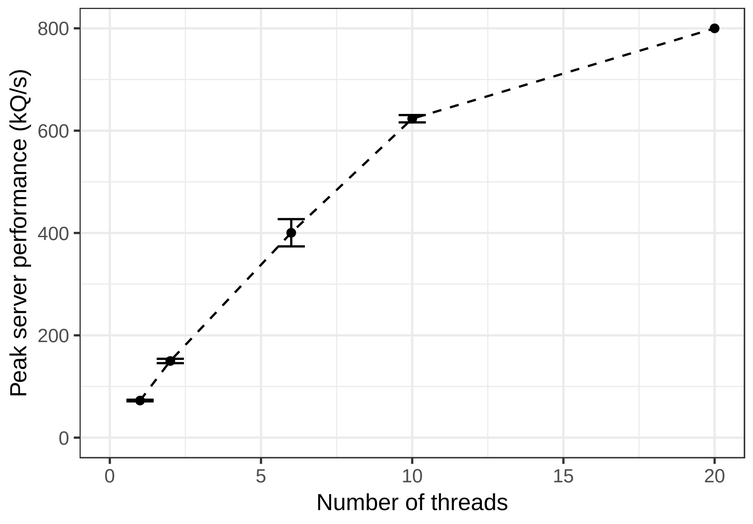
RESULT 1: UDP VS. TCP VS. TLS



RESULT 2: BIND VS. UNBOUND



RESULT 3: MULTICORE SCALING



CONCLUSION

- DNS-over-TLS runs at about 20% of the performance of DNS-over-UDP;
- Performance scales linearly with the number of CPU cores, up to a hardware limit;
- With lots of CPU cores, performance gets close to UDP (common bottleneck);
- DNS-over-TLS improves worst-case latency when messages are lost;

Further work: study the impact of client churn, alleviate the head-of-line blocking problem.

REFERENCES

[1] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, et al. Adding virtualization capabilities to the Grid'5000 testbed. In IvanI. Ivanov, Marten Sinderen, Frank Leymann, and Tony Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.

[2] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). RFC 8484 (Proposed Standard), October 2018.

[3] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya. Connection-Oriented DNS to Improve Privacy and Security. In *2015 IEEE Symposium on Security and Privacy*, pages 171–186, May 2015.