Persistent DNS connections for improved performance

Baptiste Jonglez, Sinan Birbalta, Martin Heusse

Poster: Persistent DNS connections for improved performance

Baptiste Jonglez*, Sinan Birbalta* and Martin Heusse*
*Univ. Grenoble Alpes, CNRS, Grenoble INP†, LIG, 38000 Grenoble, France
†Institute of Engineering Univ. Grenoble Alpes

***Abstract*—In the DNS resolution process, packet losses and ensuing retransmission timeouts induce marked latencies: the current UDP-based resolution process takes up to 5 seconds to detect a loss event. We find that persistent DNS connections based on TCP or TLS can provide an elegant solution to this problem. With controlled experiments on a testbed, we show that persistent DNS connections significantly reduces worst-case latency. We then leverage a large-scale platform to study the performance impact of TCP/TLS on recursive resolvers. We find that off-the-shelf software and reasonably powerful hardware can effectively provide recursive DNS service over TCP and TLS, with a manageable performance hit compared to UDP.**

***Index Terms*—DNS, TCP, TLS, DoT, persistent connection, latency, recursive resolver**

I. Introduction

The DNS (Domain Name System) needs to be reliable and to offer low latency in order for user-facing programs to provide a good experience. For a web browser, the latency of the initial DNS exchange cascades down to the page load time and may become noticeable by the user [2].

Today, DNS runs primarily on top of UDP, with a fallback to TCP for large messages. Since UDP does not provide any reliability mechanism, DNS must implement loss detection and retransmission at the application layer. This mechanism is sub-optimal and leads to very high end-to-end latency in the event of a packet loss. All stub resolvers implement a retransmission timeout, ranging from 1 second with an exponential backoff (Windows, OS X, IOS) to a fixed timeout of 5 seconds (Android, Linux). On Android, the most widely used operating system on smartphones and tablets, any application needs 5 seconds to recover from a single query loss!

In this work, we first show that using *persistent DNS connections* can improve worst-case query latency when a network experiences packet loss. Persistent DNS connections can be implemented using any of several standardized protocols: DNS-over-TCP (RFC 7766 & 7828), DNS-over-TLS (DoT, RFC 7858) or DNS-over-HTTPS (DoH, RFC 8484).

We then evaluate the performance impact of DNS-over-TCP and DNS-over-TLS on recursive resolvers, using a large-scale setup that involves tens of thousands of stub resolvers. We show that, while recursive resolvers can deliver less responses per second than with UDP, the performance of DNS-over-TCP and DNS-over-TLS remains acceptable for most use-cases: in our large-scale experiments, TCP throughput is 3 times lower than UDP, and TLS throughput is 5 times lower than UDP.

We argue that the improved worst-case latency and additional security benefits of DNS-over-TCP and DNS-over-TLS outweigh the performance impact on resolvers, especially since modern server hardware often has ample spare capacity. This is comforted by the fact that large providers such as Quad9 and Cloudflare have recently deployed public DNS resolvers with DNS-over-TLS support [3].

II. Persistent DNS connections for improved latency

Using *persistent DNS connections* means that, instead of UDP, a reliable transport protocol such as TCP is used. To amortize the cost of establishing the connection, the connection is kept open and can transport several successive DNS transactions. Figure 1 shows an example of using TCP to transport DNS messages between the stub resolver and the recursive resolver, assuming that the persistent connection had already been opened before. Notice how, thanks to the acknowledgements, the stub resolver can now measure the RTT towards the recursive resolver and adapt its retransmission timeout to recover more quickly from a packet loss. This acknowledgment system could be implemented in the DNS itself, but it is much more simple to use a widely-supported transport protocol such as TCP or SCTP.

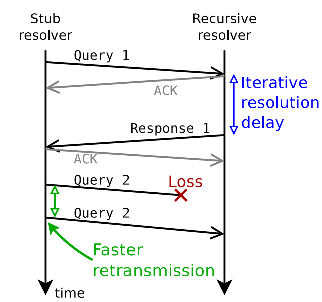

Fig. 1. DNS resolution over a persistent TCP connection: each message is acknowledged, allowing the use of a much shorter retransmission timeout.

To estimate the impact on client latency, we first run experiments in a controlled testbed with 3 machines. A client computer generates queries towards a server using either UDP or a persistent TCP connection. There is a router in the middle where we can apply packet loss and delays thanks to `netem`. For UDP, the retransmission timeout is set to 3 seconds.

Figure 2 shows the CCDF of latency for 2% of packet loss in each direction and a RTT of 200 ms. For UDP, 96.0% of queries are immediately successful (200 ms), while 3.86% of UDP queries need a single retransmission after 3 s and end up with a latency as high as 3200 ms. For TCP, only 77.3 % of queries have a latency equal to the RTT: this is caused by head-of-line blocking. The issue is particularly severe in this experiment because the RTT is large compared to the average inter-query interval (162 ms), so a single lost message causes several subsequent messages to be blocked at the receiver. Still, the worst-case latency is much favorable than UDP: 92.1 % of TCP queries completed under 500 ms, the 99th percentile is reduced from 3200 ms to 1006 ms, and the 99.9th percentile is reduced from 6200 ms to just 1157 ms.

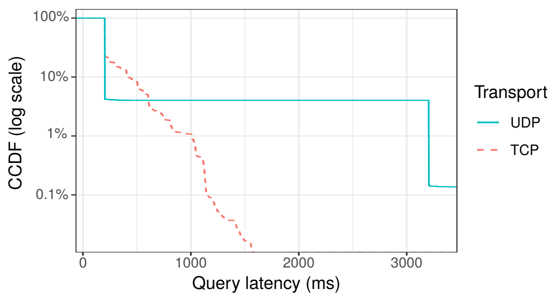

Fig. 2. Latency comparison of DNS-over-UDP and DNS-over-TCP with 2% of packet loss in each direction and 200 ms of RTT.

Previous work [4] has highlighted that other factors can significantly impact query latency, most notably when a recursive resolver cannot reply to queries out-of-order. Nevertheless, with no packet loss, a warm cache and a persistent connection, the authors found that TCP and TLS provide the same latency as UDP (1 RTT).

III. Recursive resolver performance

To assess how recursive resolvers can cope with persistent DNS connections, we built a large-scale setup that loads two recursive resolvers software (`unbound` and `bind9`) with queries from tens of thousands of stub resolvers. These clients run in virtual machines and connect over TCP or TLS to a single recursive resolver, running on a dedicated server. All servers are part of the Grid'5000 [1] research platform. Our custom DNS client establishes persistent connections to the resolver, and then generates queries on these connections according to a Poisson process.

Figure 3 shows the peak performance of `unbound` running on a single CPU core, as a function of the number of connections. With few clients, performance of DNS-over-TCP is close to that of DNS-over-UDP. When the number of clients increases, performance of DNS-over-TCP drops, stabilizing around a slowdown of 75%. For DNS-over-TLS, the performance profile is similar to TCP, but with a 80% to 85% slowdown compared to UDP.

We expect our performance results for TLS to be optimistic, because we chose to focus on the steady-state: all persistent connections are opened before starting to send queries. This hides the CPU cost of establishing a new TLS session, which

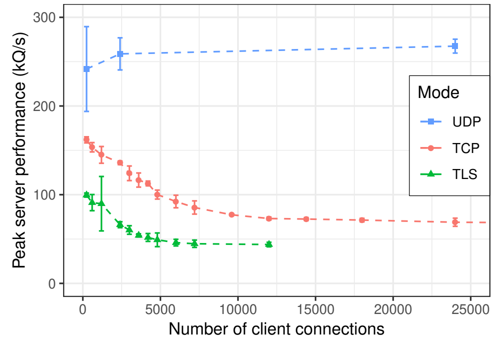

Fig. 3. Performance of `unbound` when the number of clients increases. Each point shows the average peak performance for this number of clients over a few experiments, with 95% confidence intervals.

is possibly large in a real DNS setup where customers will come and go.

We tested the performance of `bind9`, which is lower than `unbound` but shows a similar profile otherwise. We also looked at `unbound`'s multi-core performance with TCP and found that performance scales linearly with the number of threads, up to around 12 threads.

IV. Conclusion

While UDP is undeniably lightweight and suitable for DNS, a fraction of DNS queries suffer critically long delays even with mild packet loss. This worst-case latency can be substantially reduced if a persistent transport such as TCP or TLS is used instead of UDP.

Nevertheless, switching to TCP or TLS has an impact on the load of the recursive resolver. We have evaluated the drop in maximum query rate for two popular implementations and we find that it is significant, especially with a large number of concurrent connections. Still, the hardware can relatively easily be scaled up to compensate this additional load.

Overall, the switch from UDP to TCP or TLS is multifaceted. The performance impact needs to be taken into account by recursive resolvers operators, but other factors such as improved privacy and better protection from DDoS attacks make this switch attractive. In any case, we have shown that using persistent connections on a large scale is feasible with modern hardware and software.

References

[1] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, et al. Adding virtualization capabilities to the Grid'5000 testbed. In I. Ivanov, M. Sinderen, F. Leymann, and T. Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.

[2] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I. J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl. Reducing Internet Latency: A Survey of Techniques and Their Merits. *IEEE Communications Surveys Tutorials*, 18(3):2149–2196, 2016.

[3] Quad9. Quad9 DNS: Internet Security and Privacy in a Few Easy Steps. https://www.quad9.net/, 2018. Online; accessed on 14 December 2018.

[4] 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.