An Empirical Study of the Cost of DNS-over-HTTPS

Timm Boettger, Felix Cuadrado, Gianni Antichi, Eder Leao Fernandes,
Gareth Tyson, Ignacio Castro and Steve Uhlig
Queen Mary University of London

ABSTRACT

DNS is a vital component for almost every networked application. Originally it was designed as an unencrypted protocol, making user security a concern. DNS-over-HTTPS (DoH) is the latest proposal to make name resolution more secure.

In this paper we study the current DNS-over-HTTPS ecosystem, especially the cost of the additional security. We start by surveying the current DoH landscape by assessing standard compliance and supported features of public DoH servers. We then compare different transports for secure DNS, to highlight the improvements DoH makes over its predecessor, DNS-over-TLS (DoT). These improvements explain in part the significantly larger take-up of DoH in comparison to DoT.

Finally, we quantify the overhead incurred by the additional layers of the DoH transport and their impact on web page load times. We find that these overheads only have limited impact on page load times, suggesting that it is possible to obtain the improved security of DoH with only marginal performance impact.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Network measurement**; **Network security**.

KEYWORDS

DNS-over-HTTPS, Transport, Performance

ACM Reference Format:
Timm Boettger, Felix Cuadrado, Gianni Antichi, Eder Leao Fernandes, Gareth Tyson, Ignacio Castro and Steve Uhlig. 2019. An Empirical Study of the Cost of DNS-over-HTTPS. In *IMC '19: ACM Internet Measurement Conference, October 21–23, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3355369.3355575

1 INTRODUCTION

Introduced in 1983, the Domain Name System (DNS) has become a critical component of the Internet. In addition to its original purpose of domain name resolution, DNS has also gained relevance due to its intensive use by Content Distribution Networks (CDNs) for traffic redirection [5, 8]. Most websites nowadays include content from third parties, hence requiring multiple DNS queries [7] to access a single page. To highlight this, Figure 1 shows the number of DNS queries required to fully retrieve each page in the Alexa global top 100k sites. Each website was retrieved through Firefox, logging the DNS requests at the stub resolver. Caches of both Firefox and the DNS stub resolver were emptied before requesting the next website. The figure illustrates that multiple DNS queries per page are the norm rather than the exception: about 50% of the sites require at least 20 DNS queries.

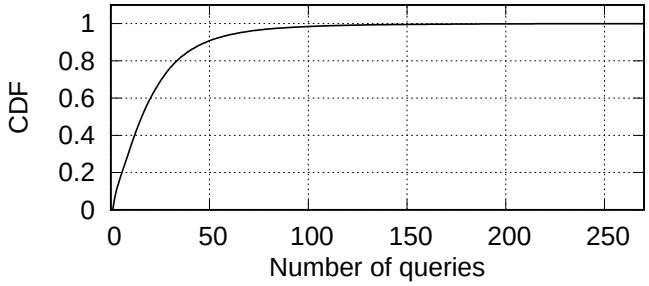

Figure 1: CDF of the number of DNS queries required to retrieve all embedded objects for each of the top 100k Alexa sites.

DNS impacts networked application performance [6] and can reveal information about the destination of a connection [4]. Addressing increasing concerns about security, DNS-over-TLS (DoT) [11] and more recently DNS-over-HTTPS (DoH) [10] have been proposed within the IETF. To increase security, these protocols rely on a TLS session between the client and the resolver. In the case of DoH, this TLS session also contains a HTTP connection. So far, DoT has only gained limited traction, whereas DoH has gathered substantial momentum already [13], with the help of notable players like Mozilla, Cloudflare and Google.

In this paper, we take a first look at the implications of securing DNS with DoH. We also compare DoT and DoH to shed some light on why the latter has recently gained so much interest. The following are the main contributions:

(1) We survey and characterize the current landscape for secure DNS via HTTP and TLS.
(2) We compare different transport protocols for securing DNS resolution, to understand the momentum behind DoH.
(3) We quantify the overheads incurred by the additional HTTP and TLS layers in DoH.
(4) We take a first look at the impact that switching to DoH has on web performance, more specifically DNS resolution times and page load times.

2 THE DOH LANDSCAPE

To better understand the current landscape of DoH resolvers, we take the list of DoH servers curated by the curl project,[1] and assess their supported feature set. We initially retrieved all information

[1] https://github.com/curl/curl/wiki/DNS-over-HTTPS

IMC '19, October 21–23, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6948-0/19/10.
https://doi.org/10.1145/3355369.3355575

Provider	DoH URL	MK
Google (i)	https://dns.google.com/resolve	G1
Google (ii)	https://dns.google.com/dns-query	G2
Cloudflare	https://cloudflare-dns.com/dns-query	CF
Quad9	https://dns.quad9.net/dns-query	Q9
CleanBrowsing	https://doh.cleanbrowsing.org/doh/family-filter	CB
PowerDNS	https://doh.powerdns.org/	PD
Blahdns	https://doh-ch.blahdns.com/dns-query	BD
	https://doh-jp.blahdns.com/dns-query	
	https://doh-de.blahdns.com/dns-query	
SecureDNS	https://doh.securedns.eu/dns-query	SD
Rubyfish	https://dns.rubyfish.cn/dns-query	RF
Commons Host	https://commons.host/	CH

Table 1: Compared DoH resolvers. Markers (MK) refer to column identifiers used in Table 2. Blahdns offers DoH services on three different URLs.

Feature	G1	G2	CF	Q9	CB	PD	BD	SD	RF	CH
`dns-message`	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
`dns-json`	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗
TLS 1.0	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗
TLS 1.1	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗
TLS 1.2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TLS 1.3	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓
CT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DNS CAA	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
OCSP MS	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
QUIC	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
DNS-over-TLS	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
Traf. Steering	DL*	DL*	AC+	AC+	AC+	UCÂś	UCÂś	UCÂś	UCÂś	AC+

* DNS Load Balancing + Anycast ÂśUnicast

Table 2: Comparison of DoH resolver features. Column names refer to markers in Table 1.

in this section on 10 October 2018. We then verified it and, where necessary, updated entries in both tables again on 10 September 2019.

As Table 1 shows, major players like Google, Cloudflare and IBM (Quad9), as well as some smaller players, support DoH. We observe diversity in their service configurations. While different base URLs for every service can be expected, it is surprising to see four different URL paths (`/`, `/resolve`, `/dns-query`, `/family-filter`) just among these nine providers. Google even uses different paths to two different services with the same base URL. While technically the DoH RFC [10] does not mandate a specific path to be used and leaves it up to the service operators, the majority of services still use the path `/dns-query`, which is the one used in all examples in the RFC. Given the huge efforts spent by operators to obtain easy-to-remember and thus easy-to-configure IP addresses for their UDP based DNS servers [12, 24], seeing such a potentially confusing variety and choices for the DoH service parameters is noteworthy. DoH operators indeed seem to have realised this, when we first collected this information in October 2018 we observed six different base paths for the same set of providers, while now we only observe four.

We now examine the features supported by the individual resolvers. HTTP supports the transmission of different content types. As per the DoH RFC, all DoH servers and clients must support the `application/dns-message` content type, which essentially is an encapsulation of the UDP DNS wireformat in HTTPS. Another widely supported type is `application/dns-json` which represents DNS messages in JSON format. While a draft RFC for the JSON DNS format [3] exists, its support is not mandatory for DoH servers. The `application/dns-message` content type is supported by all implementations except Google's. Google in fact operates two different services with two different paths (`/resolve` and `/dns-query`) on the same domain, with each service only supporting one content type. Curiously enough, the service supporting the RFC mandated format was initially named `/experimental` and has since then being renamed to `/dns-query`. This again highlights that operators have understood that too many different URLs are confusing and might lead to configuration errors. Of the remaining eight providers four also support the JSON format on the same path as the DNS wireformat.

DoH was designed as a secure service with transport encryption via TLS. TLS support is thus a strict requirement. There has been significant change on the TLS front recently, with TLS 1.3 becoming an official RFC and security vulnerabilities POODLE and BEAST rendering TLS 1.0 and lower standards insecure. All DoH servers support TLS 1.2, and seven of the nine providers also support TLS 1.3. This is a positive sign towards broader acceptance of DoH, since when we assessed these features in October 2018 only Cloudflare and SecureDNS supported TLS 1.3. On the other hand, we also see that some servers still support the deprecated TLS versions 1.0 and 1.1. We suspect the reason is that some operators are concerned about compatibility issues with older client libraries and thus also support older TLS versions. While the client can always insist on negotiating a connection with TLS 1.2 or higher, it would make sense that operators also use this opportunity to provide secure DNS and simultaneously put pressure on dropping TLS versions 1.1 or lower.

Four of the servers surveyed (Google, Cloudflare, IBM, and PowerDNS) also support DNS-over-TLS [11], the previous RFC for encrypting DNS requests using TLS. Despite having a three-year headstart over DoH, DoT has failed to gain significant traction compared to the previous specification. We will further explore both protocols in the next section.

Finally, DNS-over-HTTPS relies on the PKI-certificate system to ascertain the identity of the DNS resolver. To compensate known weaknesses and flaws of this system, techniques such as Certificate Transparency (CT)[2], Certification Authority Authorization (CAA) [9] records and Online Certificate Status Protocol (OCSP) [22] have been proposed. We check for support of CT, DNS CAA records and

[2]https://www.certificate-transparency.org

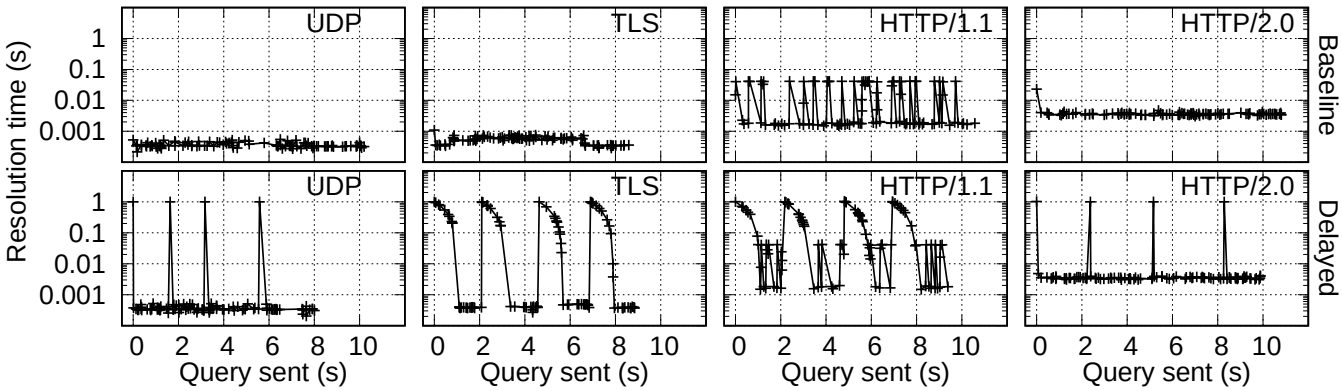

Figure 2: Impact of head-of-line-blocking on resolution times for DNS over different transport protocols. The upper charts depict the baseline and the lower ones the effect of a delay (1000ms for one in 25 queries).

OCSP in the Must-Staple (MS) configuration. While all certificates used for the DoH-servers are registered in the CT system, only Google offers DNS CAA records and no server demands OCSP MS. Again, we argue that the introduction of a new secure DNS protocol would be an ideal opportunity to establish and require support for all techniques that can further improve the security of DoH.

3 TRANSPORTS FOR SECURE DNS

In this section, we investigate the impact of different (secure) transport choices for DNS messages. We compare DNS-over-TLS with DoH using both HTTP/1.1 and HTTP/2.0.[3]

We compare the effect of these different choices of transport via a controlled experiment. We set up a local CoreDNS resolver, and use it to resolve 100 domain names via UDP, TLS, HTTP/1.1 and HTTP/2.0. As we are evaluating the impact of the transport protocol, we instruct our resolver to always return the same IP address independently of the domain name. Using unique domain names for each query rules out any impact of caching while still being able to attribute differences in resolution time to the transport protocol instead of the resolved domain name. We construct the queried domain names with a random prefix of constant length five followed by a fixed base domain. This construction ensures that effects of compression of query names are uniformly distributed across all queries, hence ensuring that differences in compressability of domain names do not impact our results. To introduce workload variability, we use non-deterministic query arrival time, where query arrival times follow a Poisson distribution with an average arrival rate of 10 queries per second. Experiments were carried out on a machine running CentOS 7 on a 4-core Intel Core i5-2500K CPU (3.30 GHz) and 8GB of RAM. Python 3.6 and CoreDNS 1.2.2 were used. Experiments were isolated with Docker containers. Python's standard packages for sockets, TLS and HTTP/1.1 were used, DNS handling was done with the dnspython package, for HTTP/2.0 support Facebook's doh-proxy package was used.

We carry out two measurement runs. In the first run, we obtain a baseline of the achievable performance by answering queries as fast as possible. We then instruct our resolver to delay one in every 25 queries by 1000ms, to observe whether delays in resolution time affect subsequent answers.

Figure 2 provides the results in both scenarios for each transport protocol. Resolution time is the time it takes the application to receive and fully parse a reply, not just the time it takes the network to transmit the message. The upper row shows baseline performance without delay. The second row shows per query resolution times with the introduced delay. The HTTP/1.1 scenario employs HTTP request pipelining, as we are assessing the resilience against slow or delayed queries of the individual transports (so HTTP/1.1 without pipelining would be an unfair comparison).

For the baseline case without delay, we observe that both UDP and TLS deliver responses to queries in less than one millisecond. These values are expected for a controlled experiment setup running on the localhost.[4] HTTP/2.0 consistently delivers results in less than ten milliseconds. Only for HTTP/1.1, the baseline performance fluctuates significantly, which we attribute to issues in the pipelining support. Most major browsers tried to support HTTP/1.1 pipelining, but have ceased to support it due to too many interoperability issues negatively affecting performance [16, 21].

In the bottom row experiments, we observe that DNS via UDP is hardly affected by the delay. We clearly see four outliers for the four delayed queries, without any visible impact on subsequent queries. Indeed, DNS via UDP can utilize different connections via multiple port numbers to effectively multiplex queries and thus make them independent.

For TLS as transport protocol, we see that the delayed queries have a knock-on effect on subsequent queries; the serialization of the TLS connection implies that a reply to a subsequent query is only sent after the reply to the delayed query. Out-of-order delivery of replies via TLS is also possible since every request/reply pair can be identified by their unique ID. The DNS-over-TLS (DoT) RFC states that this feature should be supported, although it is

[3] We do not consider DNScrypt here since it takes an orthogonal approach. Whereas DoT and DoH encapsulate the original DNS UDP wireformat with TLS and HTTP headers respectively, DNScrypt uses a redesigned wireformat combining all these features into a single message.

[4] The first query is slower than the following ones due to additional packet round-trips by TCP and TLS handshakes during connection setup.

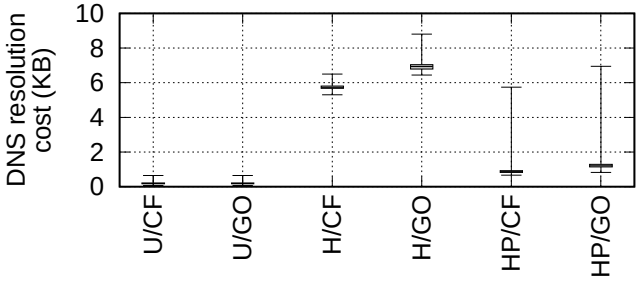

Figure 3: Total bytes per resolution. Domain names were resolved via UDP-DNS (U), DNS-over-HTTPS without persistent connection (H) and with a persistent connection (HP). The DNS servers of Cloudflare (CF) and Google (GO) were used. Whiskers span the full range of values.

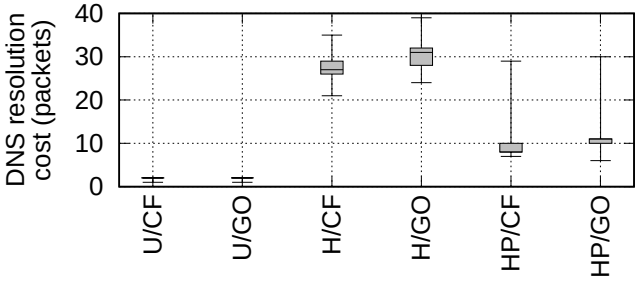

Figure 4: Total packets per resolution. Domain names were resolved via UDP-DNS (U), DNS-over-HTTPS without persistent connection (H) and with a persistent connection (HP). The DNS servers of Cloudflare (CF) and Google (GO) were used. Whiskers span the full range of values.

not mandatory. In practice, out-of-order delivery greatly complicates the server implementation compared to standard UDP, as it requires state management on the server side to handle these requests. From the only three existing DoT servers in the wild (as per Wikipedia at the time of writing this paper), we have verified that only Cloudflare supports out-of-order delivery. Implementing out-of-order delivery via TLS is akin to (re-)implementing the stream multiplexing part of SCTP, QUIC or HTTP/2.0. We believe that this is one of the main reasons why DoT failed to gain significant traction. Surprisingly, we are not aware of any other work explicitly demonstrating the potential performance impact of simple TLS transport on DNS.

For HTTP/1.1, we observe similar knock-on effects as for TLS. In the case of HTTP/1.1, there is no way to circumvent these as the in-order-delivery of requests is demanded by the RFC [18]. It is only when we turn to HTTP/2.0 that we observe a similar insensitivity to delayed queries as with UDP. Indeed, the DoH RFC states that HTTP/2.0 is the minimum recommended version of HTTP to be used.

In this section we have seen that DNS-over-TLS and DNS-over-HTTPS/1 suffer from head-of-line-blocking. Only with HTTP/2.0 DoH manages to provide similar results to UDP DNS with respect to head-of-line blocking. This difference in behavior might (at least in part) explain why it was easier for DNS-over-HTTPS/2.0 to gain traction than for DNS-over-TLS.

4 OVERHEAD

In the previous section, we have seen that DNS-over-HTTPS/2 offers significant advantages over DNS-over-TLS and DNS-over-HTTPS/1. However, the requirement for HTTP/2 introduces additional layers and thus more headers and overhead. In this section, we compare the overhead incurred by DNS-over-HTTPS/2 and regular UDP-based DNS.

To obtain a set of domain names that is representative of the real-world, we fetch the top 100,000 webpages as per global Alexa ranking and gather all domains that were resolved during these fetches. We instruct the local stub resolver to log all queries. The Alexa list was retrieved on 15 September 2018. In contrast to browser-generated HTTP Archive (HAR) files, this allows us to obtain the domains that are not part of the actual webpage but are contacted by common web browsers during page load, e.g., OCSP records for secure TLS connection establishment. While fetching these 100,000 webpages, 2,178,235 DNS queries were sent. As domain names can be embedded in more than one page, these 100,000 page fetches resolved 281,414 unique domain names. Notably, almost 25% of all DNS queries can be attributed to the fifteen most frequently queried domain names. We then resolve these domain names from a university vantage point via regular UDP-based DNS and DNS-over-HTTPS, using the respective resolvers of both Google and Cloudflare.

Figure 3 shows the distribution of request sizes for all six scenarios. Figure 4 depicts the number of packets. When comparing UDP-based DNS with DoH, we see that the UDP transport systematically leads to fewer bytes and fewer packets exchanged, with the median DNS exchange consuming only 182 bytes bytes and 2 packets. A single DoH resolution in the median case on the other hand requires 5737 bytes and 27 packets to be sent for Cloudflare and 6941 bytes and 31 packets for Google. A single DoH exchange thus consumes more than 30 times as many bytes and roughly 15 times as many packets than in the UDP case. Persistent connections allow to amortize one-off overheads over many requests sent. In this case, the median Cloudflare resolution consumes 864 bytes in 8 packets, the median Google resolution 1203 bytes in 11 packets. While this is significantly smaller compared to the case of a non-persistent connection, DoH resolution still consumes roughly more than four times as many bytes and packets than UDP-based DNS does.

While in the legacy case there is no significant difference, in the DoH case Google's server leads to larger transactions than Cloudflare's. This is caused by Google needing more bytes to establish and maintain the TLS connection than Cloudflare. The reason is Google's usage of a certificate larger than Cloudflare's: in our specific setup, Cloudflare transmits two certificates worth 1,960 bytes and Google transmits two certificates worth 3,101 bytes. When using a persistent TLS connection, the overheads get amortized over the many requests made.

We now break down the overhead for DoH. As a by-product, we showcase some of the new features of HTTP/2 in comparison to HTTP/1. Next to header compression using HPACK [20],

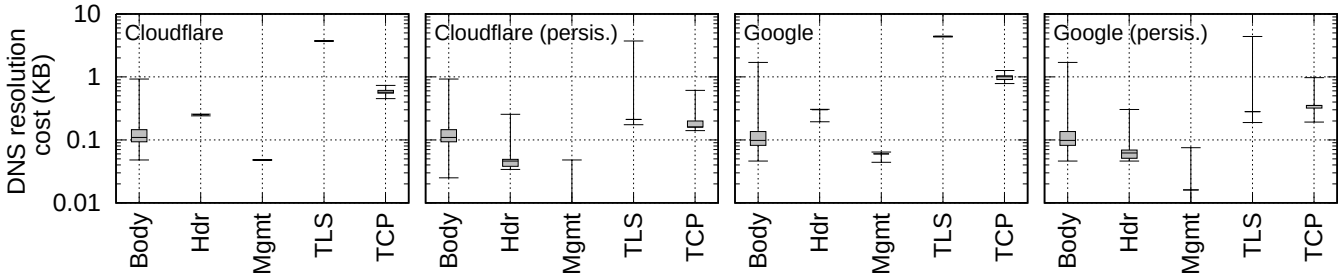

Figure 5: Overheads per DNS resolution for DNS-over-HTTPS/2. First two columns show sizes for (HTTP) bodies and headers exchanged. Mgmt refers to messages being exchanged to maintain the HTTP/2 connection like settings and windows updates. TLS and TCP refer to sizes of the respective layers.

HTTP/2 also supports a differential transmission mechanism that only transmits the changed headers during the subsequent exchanges. HTTP/2 also defines new message frames to manage the connection. Figure 5 shows a breakdown of overheads into individual layers and protocols. Across all four cases, the distribution of body sizes is similar, albeit Google tends to send slightly larger bodies in the extreme case.

Every additional layer of complexity adds overhead that is at least the same size as the original DNS payload. Notably, even the overhead incurred by TLS encryption and TCP headers and additional messages is already of the size of the complete DNS payload. Regarding the HTTP/2 overhead (headers and mgmt), we see that using a persistent connection leads to less data being exchanged. For the headers, this is caused by HTTP/2's differential headers feature, which in sequential requests and replies only transmits those headers that have changed. The management messages are required to manage the HTTP/2 connection and multiplexing of different streams. They do not need to be sent for every single client-server-interaction. Therefore, when using a persistent and thus re-usable connection, the amount of management bytes sent per request-response-cycle is smaller in comparison to non-persistent connections. For the overhead incurred by TLS, for the non-persistent connections, the overhead is dominated by the server certificate as discussed above. In the case of persistent connections, the upper whiskers in Figure 5 are caused by the (at least once) necessary certificate exchange. The median values however are significantly lower as an established connection is re-used many times. This variability in the TLS overhead also causes different overheads at the TCP and outer layers, as the higher number of bytes transmitted for the TLS layer also leads to more packets.

In summary, many of the one-time overheads required for TCP, TLS and HTTP connection setup and management can be amortized if a persistent connection is used. However, even in this case, the median overhead caused by the TLS and TCP layer are each already of the size of the actual DNS message. For DNS resolution over HTTP, this effect is pronounced because of the comparably small size of the DNS message. When considering transmitting web pages via HTTPS, this effect will be less pronounced in comparison to DNS messages, given the larger size of websites.

5 DOH PERFORMANCE

In the previous sections we have quantified the potential impact of head-of-line-blocking as well as the additional overheads of DoH. In this section, we assess whether DoH impacts performance, more specifically we look at a web browsing scenario and investigate how a change to DoH affects page loading times.

We use the Firefox web browser to measure webpage load times for the 1,000 highest ranked webpages in the global Alexa ranking. The Alexa list was retrieved on 18 April 2019. We choose Firefox because as of the time of writing this paper it was the only browser with documented support for DoH. We use Firefox 66.0.3 for the experiments. We rely on the Browsertime framework from the sitespeed.io project[5] to instruct Firefox for the measurements and collect HAR files with the performance statistics.

We measure performance using the locally configured resolver, and also using the public resolvers from Google and Cloudflare over legacy DNS as well as DoH. This way, the performance obtained with the local resolver provides a baseline, allowing us to assess how a change to a cloud-provided DNS service affects performance. For the cloud provided DNS services, we also assess the performance difference between using the traditional UDP-based DNS protocol and DNS-over-HTTPS. In this setup, each website was loaded three times with the browser cache purged before each measurement iteration. This was done from a university-local server.

The left plot in Figure 6 shows the CDFs of the cumulative DNS resolution times per webpage in milliseconds. By cumulative DNS resolution times, we mean the time it would take to perform all DNS queries serially, whereas in reality they can be parallelised. We crop the CDF plot at 20,000ms, since the results have a very long tail.

We first observe that the cloud-based name resolution via UDP leads to faster resolution times than using the local resolver. From the cloud-based ones, Cloudflare leads to faster resolution times than Google. When comparing DoH-services from Cloudflare and Google, we observe that using DoH leads to longer DNS resolution times than when using the traditional DNS resolution. This is to be expected from the added overhead for encryption and transport. Also, we observe that the DoH resolution provides comparable resolution times to the local resolver, with again Cloudflare slightly faster than Google.

[5] https://www.sitespeed.io

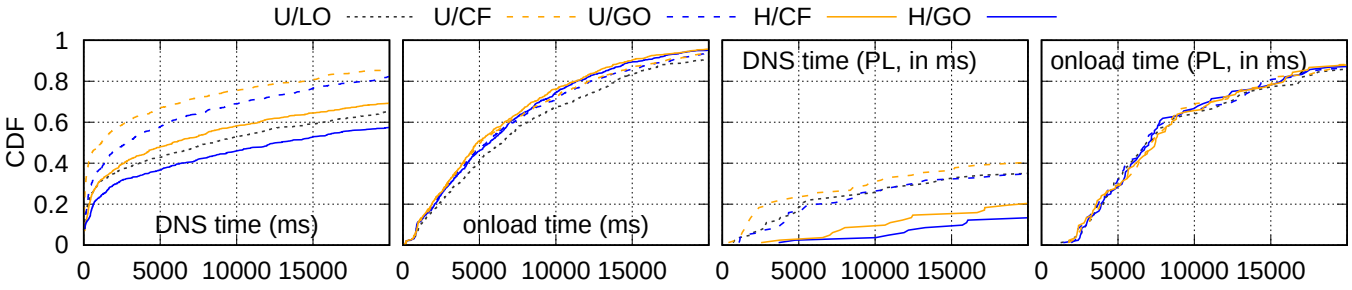

Figure 6: CDF of DNS resolution and page load times (time of onload event): U/ indicates legacy resolver, H/ indicates resolution via DoH, /LO indicates local resolver, /GO indicates Google and /CF indicates Cloudflare.

Even though these results show that changing to DNS resolution via DoH leads to longer DNS resolution times, this does not necessarily translate into longer page load times. The second plot in Figure 6 shows CDFs of the complete page load time, measured as the time when the onload event was triggered. The onload event is triggered when the whole page including all dependent resources like stylesheets and images has been loaded [17]. Note that overall page load times are faster than DNS resolution times as the browser sends requests in parallel, whereas DNS plots shows cumulative DNS resolution times without parallelism. The figure shows that page load times are comparable for all resolution approaches. As for the previous DNS resolution times, using a cloud-based DNS service offers slightly faster page load times. There is however little difference between page load time via legacy DNS or DNS-over-HTTPS: both resolution mechanisms achieve similar page load times.

Note that we also attempted to run the same experiments from PlanetLab. Unfortunately, at the time of writing this paper, only 39 nodes were able to run these experiments, as most of them were unreachable, and among those that were reachable, many were running an OS that was too old to support a recent enough version of Firefox that supports DoH. The limited results (plots on the right in Figure 6) we obtained from PlanetLab however are consistent with those we have obtained locally: DNS resolution via DoH takes longer, but page load times overall change only little when changing the resolution method.

Overall, the results of this section show that a switch to DNS-over-HTTPS does not seem to incur significantly longer page loading times. This means it is feasible to benefit from the better privacy guarantees of DoH without sacrificing user-perceived page loading times.

6 RELATED WORK

DNS-over-HTTPS still is a relatively new protocol. To the best of our knowledge, this paper is the first to look into the differences between DNS-over-HTTPS, DNS-over-TLS and UDP-based DNS. Mozilla has published a blog post [15] briefly describing their experience with a DoH trial in Firefox. This blog post however focuses more on reporting experiences of using a third-party resolver than on implications that stem directly from using DoH, especially the transport aspect. In a blog post [13], Geoff Huston also asks for the advantage of DoH over DoT. This post discusses application features like HTTP push and namespaces, but does not discuss insensibility against slow queries as we do.

Since the inception of DNS, the Internet has evolved and changed, exposing the DNS protocol to new threats and challenges. The unencrypted transport of DNS leads to security and censorship issues [4, 14], whereas using UDP makes DNS usable for distributed denial-of-service attacks [2]. Other works have proposed protocol changes to use persistent connections and encryption [26]. These works list and discuss issues with the traditional UDP-based transport for DNS, of which most can be addressed by using DNS-over-HTTPS instead. In that sense, they provide good arguments to change to DoH, but do not discuss details of DoH directly.

Content delivery networks often use DNS to perform their traffic redirection. It is an active research area, with works aiming at better understanding these redirection strategies [5, 8, 19]. Other works study DNS resolver behavior in the wild with respect to latency and traffic redirection [1], look at the impact of DNS on overall application delays in the Internet [6, 25] or look at DNS infrastructure provisioning at the client side [23]. While all these works also target DNS, they have a stronger focus on the actual applications of DNS than the protocol itself.

7 CONCLUSION

DNS is one of the most important protocols for many networked applications today and was originally designed as an unencrypted protocol. Growing concerns about user privacy have led to propose more secure approaches. In this paper, we have surveyed the current DoH landscape. We have exposed the diversity in the supported content types, in the support for DNS-over-TLS, and in the supported TLS versions. We have seen, that while most DoH servers support a good set of security parameters, many of them still do support deprecated legacy settings. We have then studied the behavior of DoT and DoH against delayed queries, showing that HTTP/2 offers advantages over HTTP/1 and DNS-over-TLS. In the process, we have exposed the likely reason why DoT has not gained traction compared to DoH, despite having had a head start of a few years before DoH. We have then quantified the overheads incurred by the HTTP and TLS layers of HTTP/2. Finally, we have measured how DoH impacts page load times. This has shown that it is possible to obtain the additional security of DoH with only marginal performance penalties.

ACKNOWLEDGMENTS

We thank our shepherd Taejoong Chung and the anonymous reviewers for their reviews and constructive feedback.

This research is supported by the UK's Engineering and Physical Sciences Research Council (EPSRC) under the EARL: sdn EnAbled MeasuRement for alL project (Project Reference EP/P025374/1).

REFERENCES

[1] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. 2010. Comparing DNS resolvers in the wild. In *Proceedings of IMC*.

[2] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. 2013. DNS Amplification Attack Revisited. *Computers & Security* (2013).

[3] StÃĺphane Bortzmeyer. 2013. *JSON format to represent DNS data.* Internet-Draft draft-bortzmeyer-dns-json-01. https://datatracker.ietf.org/doc/html/draft-bortzmeyer-dns-json-01 Work in progress.

[4] StÃĺphane Bortzmeyer. 2015. DNS Privacy Considerations. RFC 7626. https://doi.org/10.17487/RFC7626

[5] Timm Böttger, Felix Cuadrado, Gareth Tyson, Ignacio Castro, and Steve Uhlig. 2018. Open Connect Everywhere: A Glimpse at the Internet ecosystem through the Lens of the Netflix CDN. *SIGCOMM CCR* (2018).

[6] Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, P Brighten Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. 2017. Why is the Internet so slow?!. In *Proceedings of PAM*.

[7] Michael Butkiewicz, Harsha V. Madhyastha, and Vyas Sekar. 2011. Understanding Website Complexity: Measurements, Metrics, and Implications. In *Proceedings of IMC*.

[8] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. 2013. Mapping the Expansion of Google's serving Infrastructure. In *Proceedings of IMC*.

[9] Phillip Hallam-Baker and Rob Stradling. 2013. DNS Certification Authority Authorization (CAA) Resource Record. RFC 6844. https://rfc-editor.org/rfc/rfc6844.txt

[10] Paul E. Hoffman and Patrick McManus. 2018. DNS Queries over HTTPS (DoH). RFC 8484. https://doi.org/10.17487/RFC8484

[11] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. 2016. Specification for DNS over Transport Layer Security (TLS). RFC 7858. https://rfc-editor.org/rfc/rfc7858.txt

[12] Geoff Huston. [n.d.]. APNIC Labs enters into a Research Agreement with Cloudflare. https://labs.apnic.net/?p=1127.

[13] Geoff Huston. [n.d.]. DOH! DNS over HTTPS explained. https://blog.apnic.net/2018/10/12/doh-dns-over-https-explained.

[14] Philip Levis. 2012. The Collateral Damage of Internet Censorship by DNS Injection. *SIGCOMM CCR* (2012).

[15] Patrick McManus. [n.d.]. Firefox Nightly Secure DNS Experimental Results. https://blog.nightly.mozilla.org/2018/08/28/firefox-nightly-secure-dns-experimental-results.

[16] Mozilla. [n.d.]. Bug 264354 - Enable HTTP pipelining by default. https://bugzilla.mozilla.org/show_bug.cgi?id=264354.

[17] Mozilla. [n.d.]. Window: load event. https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event.

[18] Henrik Frystyk Nielsen, Jeffrey Mogul, Larry M Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. 1999. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616. https://rfc-editor.org/rfc/rfc2616.txt

[19] John S Otto, Mario A Sánchez, John P Rula, and Fabián E Bustamante. 2012. Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions. In *Proceedings of IMC*.

[20] Roberto Peon and Herve Ruellan. 2015. HPACK: Header Compression for HTTP/2. RFC 7541. https://rfc-editor.org/rfc/rfc7541.txt

[21] The Chromium Projects. [n.d.]. HTTP Pipelining. https://www.chromium.org/developers/design-documents/network-stack/http-pipelining.

[22] Stefan Santesson, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Dr. Carlisle Adams. 2013. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960. https://doi.org/10.17487/RFC6960

[23] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. 2013. On measuring the client-side DNS infrastructure. In *Proceedings of IMC*.

[24] Marty Strong. [n.d.]. Fixing reachability to 1.1.1.1, GLOBALLY! https://blog.cloudflare.com/fixing-reachability-to-1-1-1-1-globally.

[25] Srikanth Sundaresan, Nazanin Magharei, Nick Feamster, Renata Teixeira, and Sam Crawford. 2013. Web performance bottlenecks in broadband access networks. In *SIGMETRICS Performance Evaluation Review*.

[26] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. 2015. Connection-oriented DNS to improve privacy and security. In *IEEE Symposium on Security and Privacy (SP)*.