Effects of secured DNS transport on resolver performance

Etienne LE LOUËT*† Antoine BLIN* Julien SOPENA† Ahmed AMAMOU* Kamel HADDADOU*
GANDI, Paris, France* - Sorbonne Université, CNRS, LIP6, F-75005 Paris, France†

***Abstract*—Designed 40 years ago, DNS is still a core component of internet: billions of DNS queries are processed each day to resolve domain names to IP addresses. Originally designed for performances and scalability, its transport protocol is unencrypted, leading to security flaws. Recently, secure protocols have emerged, but the question of their scalability and sustainability remains open. In this paper we study the cost of switching from the legacy DNS transport to the newer ones, by first characterising the shape of the traffic between clients and secured public resolvers. Then, we replicate said traffic, to measure the added cost of each protocol. We found that, while connections usually stayed open, many closures and openings were made in some cases. Comparing these profiles over different DNS transports, we observe that switching from the legacy protocol to a more secure one can lead to an important performance penalty.**

I. Introduction

Introduced in 1983, the Domain Name System (DNS) is a core component of the Internet, as nearly every communication on it is preceded by at least one DNS query. It had originally been developed with a focus on performance and scalability by using UDP as transport to achieve both the lowest latencies and server load, but concerns regarding confidentiality and integrity have since emerged. New standards, DNS-over-TLS (DoT) [9] and DNS-over-HTTPS (DoH) [7], have been proposed within the IETF to secure DNS by encrypting queries and responses using TLS. While these new standards provide both confidentiality and integrity, the question of their cost remains opened, as there is no information on the engergetical or environmental sustainability of transitioning all DNS traffic from the old protocols to the new ones. In this paper, we propose an estimation of the additional server resources required to transition from a non-encrypted, non-connected, DNS protocol to a secure but costly protocol, by observing how existing secured DNS client use the service and measuring the added cost of these protocols in a controlled environment.

First, we conducted a characterisation of the behaviour of DoH clients and public resolvers, in order to gather the different patterns and settings applied by both entities in their use of the secured protocols (number of connection openings, connection duration, number of queries allowed per connection...). We noticed that, while they tend to try and keep a single connection alive, browsers can, in certain cases close and re-open them very frequently.

Then, we realised multiple benchmarks using two secured resolvers in order to, first, compute their performance baseline when using the unconnected legacy UDP protocol, then to measure the hardware resources consumption of each of the steps (connection establishment and upkeep, as well as message processing) of the new DNS protocols. We observe that while the additional memory consumption generated by the use of the new secured protocols is noticeable, the rate of increase is not important enough to be a problem in terms of scalability. In terms of CPU overhead, transitioning from UDP to DoH can lead to a 70% decrease in performances (for relatively long-lived TCP connections). The cost of encryption is in large part added by the TLS key exchange. For DoH (the most popular protocol) the cost of implementing the HTTP/2 layers lead to a performance penalty.

The rest of the paper is organised as follows: sections II and III describe the technical background and related work, section IV proposes a characterisation of client and resolver behaviour in order to understand the shape of the traffic near the resolvers, section V proposes a benchmark and an analysis of server side performances and in section VI, we conclude.

II. Technical background

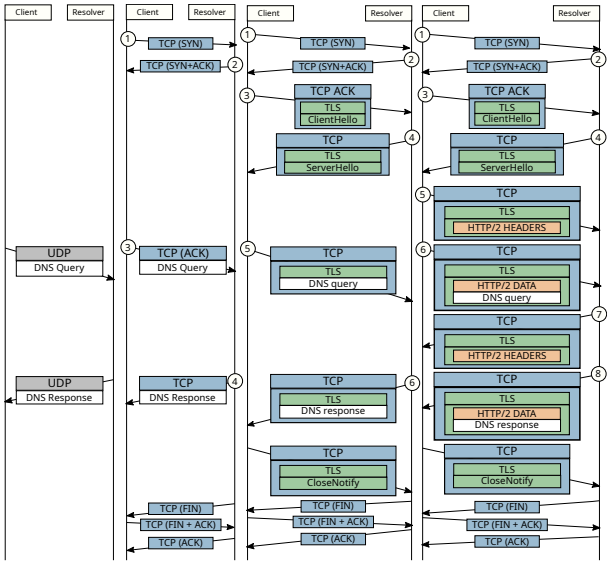


Figure 1: Comparison of DNS over UDP, TCP, TLS and HTTP/2

From a high-level point of view, DNS is a registry service queried by a client to resolve the IP address corresponding to a domain name. It has been implemented as a tree-like database, in which multiple servers hold only a fraction of the information : looking for information on it is therefore

a depth-first search starting from the root. However, if every client looking to translate a domain name to an IP address were to realise such a process, it would result in prohibitive latencies and an overload on the servers closest to the root; that's why iterating through the tree is delegated to resolvers, servers which, upon receiving a query from a client, can answer it either from their cache, that will have a higher *hit rate* since it likely received the same request from another client earlier, or by doing the resolution themselves.

In contrast to the other, historically text-based, web protocols, the DNS message protocol has been implemented using binary message format in order to target the highest performances. Both UDP and TCP have been selected as transports. The first one, UDP, is connection-less (Figure 1 a), and provides high performances at the cost of reliability and message payload size. As such, it is the recommended protocol to transport standard DNS queries (which represents most of the DNS traffic). The second one, TCP, requires the exchange of three messages (arrows 1 through 3 on figures 1 b c d) to establish a connection, before sending any DNS messages. It has been mostly used to transport special DNS messages (zone transfers) that do not fit into an UDP datagram. As these legacy DNS transport protocols are unsecured, DNS is vulnerable to a variety of attacks detailed in section III. Several secured protocols have been proposed to deal with the aforementioned security flaws.

The first one, DoT [9], uses a TLS connection [14] to provide both integrity and confidentiality. It relies on a TCP connection to establish a TLS session between the client and the server. Two messages (arrows 3 and 4 in figure 1 c) are exchanged between both endpoints to derive, from their respective pairs of asymmetric keys, a symmetric key used to encrypt the DNS binary messages. During this process, the client also validates the identity of the resolver by using the latter's digital certificate.

DoH has been proposed as an alternative to offer a secure DNS transport. It relies on HTTP/2 [2] to carry the DNS messages, and may be seen as an additional layer built on top of TLS. Once a session is established (arrows 3 through 4 in fig 1), the multiple streams of the HTTP/2 protocol are used to transport DNS queries, either directly in their binary format as the body of an HTTP POST query (arrows 5, 6, 7 and 8 in figure 1 d), or as the base64-encoded URL parameter of a GET query (as it is less common it is not considered here).

Originally designed for performance, the legacy DNS protocol doesn't offer any security guarantees. To prevent data leaks and corruption, new protocols based on existing technologies used to guarantee security on the web have been pushed in order to secure DNS. Switching from a connection-less unencrypted protocol to connected ones making extensive use of cryptography seems to go against the original goals that drove to the development of DNS, therefore, the cost of this transition must be analysed.

III. Related works

We classify the works related to DNS security in three categories: works that focus on describing and proposing mitigations for different security flaws, works that aim to compare the client-side cost of secured DNS protocols, and finally, works that focus on evaluating their adoption.

a) Security issues and guarantees: When studying the security of an information system, there are three properties to consider : confidentiality, integrity and availability.

Confidentiality: DoUDP and DoTCP did not offer any kind of confidentiality for messages, meaning that any malicious actor could use captured DNS message to breach a user's privacy [3]. DoT and DoH circumvent this flaw by using the TLS protocol to carry their messages, therefore guaranteeing confidentiality. However, several studies have shown that some characteristics of DNS traffic can be exploited to, in some cases, de-anonymize encrypted DNS traffic. In [15], Siby et al show that, despite its use of encryption, it is still possible to determine the content of a DoH flow containing un-padded queries and responses, by using traffic analysis techniques. In [5], Bushart and Roshow show that even state-of-the-art padding strategy are weak against some traffic analysis attacks. However, it is worth noting that the models used in these attacks can only de-anonymize DNS flows they were previously trained on (usually popular websites), and that techniques such as arbitrarily delaying queries and responses, or the use of proxy networks such as TOR can be powerful mitigations against these attacks. Furthermore, these attacks require both a constant update of the model used to target websites in order to cope with their modification, and the knowledge of the source of the traffic, as clients have different behaviours regarding inter-query timings and message size.

Integrity: The DNS protocol did not initially offer mechanisms guaranteeing the integrity of data, meaning that an adversary could edit a DNS response, thus redirecting a client towards fraudulent services [11]. DNSSEC was later standardised, and guarantees the integrity of data exchanged between the resolver and name servers. On the other hand, the data exchanges between client and resolver still use the legacy protocol, leaving them vulnerable to the aforementioned attacks. As TLS guarantees the integrity of the messages it transports, using DoT or DoH in combination with a trusted resolver that validates the integrity of records by using DNSSEC, can protect against this category of attacks.

Availability: The two aforementioned properties are necessary but not sufficient to fully protect a client. DNS is one of the most commonly filtered protocols (by governments or ISPs [10]). DoT, which uses port 853 by default can be easily blocked by port-based filters, while DoH is not, as it relies on a widely used protocol. It is still vulnerable to fingerprinting techniques, able to detect whether or not an encrypted flow contains DoH queries and response, like Vekshin et al prototyped in [16]. However, as we said earlier, these techniques requires models trained on a variety of clients,

resolver and traffic shape that require constant updating, so it is unrealistic to expect them to be used globally.

Despite the remaining security limitations, the benefits provided by DoT and DoH complete the efforts first undertaken with the introduction of DNSSEC.

b) Client-side performance: Various studies focus on the client-side cost of DoT or DoH: Hounsel et al. [8] compare the page load times using different combinations of DNS transports, network types and public resolvers. Boettger et al. [4] also compare the resolution times and protocol overhead of different secure DNS transports when using persistent or non-persistent connections. These studies find that connection reuse is beneficial for the client, and that secure DNS adds no noticeable cost to clients, except on some cellular networks.

c) Protocol Adoption: In [6] Garcìa et al analyse both the number of available DNS-over-encryption resolvers, as well as the use of DNS-over-encryption by various users. While the amount of DoH traffic had stayed stationary, representing about 1% of the current DNS traffic, the number of available DoH server is steadily growing, leaving the question of energetic sustainability, if their use is generalised, opened.

IV. Behaviour of clients and resolvers

As the newer DNS transports are connection based, new questions arise: while the protocol defines how to query the service, it doesn't specify how the underlying connections should be managed by both the client and resolver. The sequence of connections opening and requests sent is mostly controlled by the client, but to focus only on the client's behaviour is not enough, as the server has the choice to accept, reject, close or keep said connections opened.

The objective of this experiment is to characterise the shape of the traffic between already existing clients and publicly available resolvers, so we can generate similar traffic when measuring server-side performance. In section IV-A we describe the experimental setup used for the measurements while section IV-B contains an analysis of the different behaviours observed from the clients and the resolvers.

A. Experimental setup

We selected, as clients, two web browsers: Firefox v91.5 and Chromium v101.01, as it is possible to configure them to emit all their DNS traffic over HTTPS instead of relying on the host OS's services (which would, in the vast majority of cases result in unsecured DNS traffic). However, there are other software than web browsers that generate DNS traffic, and, in the majority of cases, said traffic is unsecured. A new category of software, called proxies, has emerged to resolve this issue. They capture all DNS queries emitted by software on the system, and transmit them to a configured resolver over a secured channel. This means that, by installing and configuring it, all software transparently benefits from a secured DNS channel to a trusted resolver, that is shared among all running applications, which saves both client and server resources when compared to a model in which each client individually implements secured DNS transports. As DoH has gained more traction than DoT, it is the only secured DNS transport available in web-browsers. Therefore, for the following experiments, all DNS traffic we generate is based on DoH. For the public resolvers, we chose three major players widely used by the public: Quad9, Google and Cloudflare. We gather the list of websites used for the resolution through a public list of domains names [18], filtered to keep the ones that still have an A record corresponding to a server accepting HTTP traffic. In order to generate the appropriate traffic we configure the browser with the selected DoH resolver, and then we load a JavaScript script making HTTP requests at various rate (one every 50 ms, 1000 ms and 60 000 ms) during a period of 30 minutes. To generate traffic using DNScrypt-proxy, we use a C program making UDP resolutions at the same rates as the ones configured for the browser, using the system's configured resolver, which, in this case, is DNSCrypt-proxy. We characterise the shape of the traffic by measuring, for each connection, its duration, the number of queries that were sent on it, as well as the origin (client or the server) and method (FIN or RST) of the its closure.

B. Results

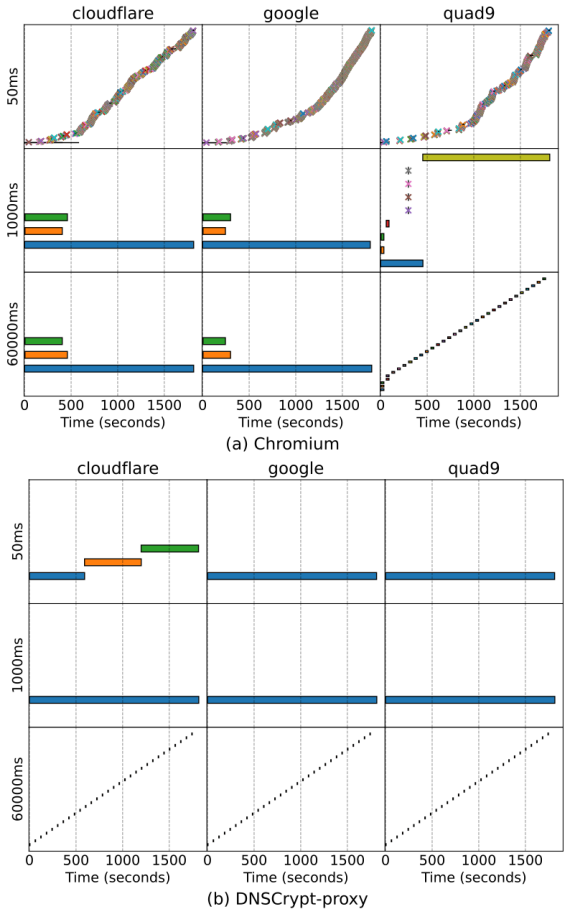


Figure 2: Connection use by Chromium and DNSCrypt-proxy for a set of resolver and query delays

Figure 2 presents, for each combination of software, inter-query delay and public resolver, the number and length of

connections to the resolver established by the client (for example, on fig 2(a), the top-right figure presents the number and length of TCP connections that chromium established towards the quad9 resolver. The horizontal axis of each sub-figure represents the time in the experiment, and a connection is presented as by a coloured rectangle, its leftmost and rightmost edges mark its start and end date respectively. Connections shorter than a second are represented by a cross. For example, we can observe that, when the inter-query delay is 50ms, DNScrypt-proxy established only one connection to the quad9 resolver (figure 2(b), top-right). On the other hand, we observe that, with the same inter-query delay of 50ms, firefox established a lot of short-lived (less than 1s) to the quad9 resolver (figure 2(a), top-right). We only elected to present chromium's profile, as Firefox's is similar.

a) DNScrypt-proxy: DNScrypt-proxy generates the least aggressive load towards the server. Indeed, its main behaviour is to open and keep opened a single TCP connection that it will use to perform all requests, regardless of intensity of the traffic generated. In addition, an internal timer is set to trigger the close of the TCP connection when it is unused, freeing servers resources (bottom row on figure 2b).

b) Web browsers: The browsers have a more aggressive usage of DNS resources: at the beginning of the sessions, they try to maximise the probability of having a successful connection to the DNS server by opening several connections in parallel to the same server, likely to speed up the early resolutions that browsers usually do, (figure 2a), leading to an increase in server resources usage. The following use of these opened connections depends on the intensity of the traffic. When the traffic has a low intensity, with a request frequency lower than 1 query per second (see the bottom two rows of figure 2a), a single connection is mainly used to handle the traffic, the remaining connections eventually being closed. We sometimes observe connection closures, forcing a re-opening (1000ms delay row on Figure 2a), or multiple connections at the same time (1000ms and 60 000ms delay rows on figure 2a), but these events are not numerous enough during the lifetime of an experiment to be significant. Under a DNS traffic with a high intensity (above 1 query per second) the connection pattern of the web browsers changes drastically. Not only does the browser fails to generate the traffic we ask for, we also observe connections being opened and closed in sequence, with every connection shutdown originating from the client (see top row on figure 2a) and each connection being used to carry few to no requests. From a server perspective, such behaviour represents the worst case, as, with each connection opening being costly, this leads to huge resource consumption.

c) Resolvers: Clients are not the sole responsible for the connection patterns. The resolvers have the choice to accept or deny the connections and the traffic issued from the clients. We have observed that Google has the most permissive resolver configuration of those we tested, as we didn't observe any limitation in terms of number of connections, their duration and the number of queries per second (qps) or per connection. Quad9 closes unused connection after around 20 seconds of inactivity (figure2a, bottom-right). Cloudflare does not impose any restriction on the connection duration, but limist the maximum number of requests per connection to 10 000 (figure 2b, top-left).

The intended behaviour of clients and resolvers seems to be to keep one TCP/TLS connections alive while they are used.

V. Server side performance

Moving from UDP, an unconnected protocol historically used for communication between clients and resolvers, to more complex connected ones can lead to an increase in consumption of hardware resources on the resolver side. Indeed, while the handling by the resolver, of DNS queries transported over in a UDP datagram simply requires receiving the datagram and then sending another one containing the answer once the resolution is completed, the use of session-based protocols is more complex, as they require the establishment of a session and the management of the state associated with it, in order to receive queries and emit responses.

Questions about scalability and resource consumption arise regarding the cost of these additional steps. In order to properly evaluate their cost, we realised a series of synthetic benchmarks, first using DNS over UDP (DoUDP) as a baseline, then DNS over TCP (DoTCP), DNS over TLS (DoT), and DNS over HTTPS (DoH). While it offers no privacy guarantees, measuring how DoTCP performs is still interesting because, as we have seen previously in section II, both secured protocols have been built on top of TCP. Thus, the comparison between DoUDP and DoTCP is a good performance indicator of the cost added by the TCP connection. Using the same approach, comparing DoTCP and DoT allow us to measure the performance cost of the TLS session establishment and traffic encryption, and comparing DoT with DoH gives us insights about the cost of the added HTTP/2 layers.

In section V-A, we describe the experimental environment of our benchmarks. Section V-B presents the results of a benchmark of the legacy UDP-based protocol, while Sections V-C, V-D and V-E describe the multiple synthetic benchmarks we realised to characterise the costs of the different steps of the connection-based protocols.

A. Experimental setup

We elected to run our benchmarks on a DNS architecture deployed on the Grid5000 [1] platform. Our test bed is composed of 22 Dell PowerEdge R640, each of them having an 18-core CPU with a base clock of 2.2 GHz and a turbo frequency of 3.9 GHz, 96 GiB of RAM and a 25 Gbps NIC, all connected together through the same switch. We run our resolver on one of those machines, use twenty of them as our clients and the remaining one as experiment monitor in charge of deploying and running the various actors and measurement tools on their respective machines.

We selected two resolvers implementations to test: Knot-Resolver, as it is used by important industry players (most notably Cloudflare) and dnsdist, that is not a resolver *per se*, but acts as a proxy and load balancer between a client and

another resolver, it can either answer from its cache, or forward the query to another resolver. Since it is compatible with both DoH and DoT, it allows to modernise an existing DNS infrastructure by adding support for these protocols without modifying the existing software. As we need a very high number of clients to reach 100% load on one core in our setup, we configure both software to only run on a single core of our resolver machine using Linux *cgroups*.

As we aim to focus on the resolver-side cost of transitioning from a legacy UDP based protocol to a session-based protocols for the client to resolver connections, we decided to exclude the cost of retrieving the records from the hierarchy of DNS name servers from the resolving process, to avoid measurements noise that could occur when querying external uncontrolled name servers. At the beginning of each experiment, we fill the cache of knot or dnsdist with the DNS records that will be queried, all subsequent clients queries resulting in a cache HIT. To reduce experimental variability as much as possible, all of the 2000 queried domain names have the same length and fake TLD.

Traffic is generated using Flamethrower [12], a DNS benchmarking utility compatible with all benchmarked protocols. We patched its code so it would be able to keep the underlying connections opened for a configurable duration, as the default behaviour was to close them once it sent a batch of queries. When benchmarking DoH, we send our queries in the body of a POST HTTP/2 query, as we detected that it was how our clients operated. A fork of Flamethrower including these changes is available on github [13].

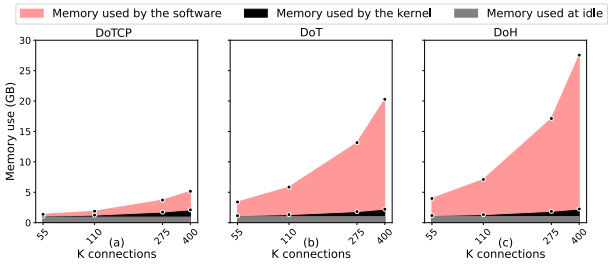
B. Baseline

As it is the legacy, most widely used and the most efficient transport for DNS, measuring how UDP performs gives us a baseline in terms of performances. Therefore, we benchmark our resolvers by sending as many requests as possible, stopping only when we started recording systematic losses. At best, knot answered 115 000 qps out of the 120 000 qps sent by our clients, while dnsdist answered 225 000 qps, out of the 240 000 sent. We investigated these losses and noticed that they were due to a saturation of the CPU, both resolvers being unable to process queries at such a rate, leading to the kernel-side UDP reception buffer filling up and packets having to be discarded. We explain the difference in performances of almost 50% between knot and dnsdist by the fact that dnsdist is a proxy and load balancer whose purpose is to pass queries to an upstream server as efficiently as possible, therefore having very few things to do when receiving a query other than answering it from its cache or forwarding it, while knot most likely has to do more processing, even in the case of a cache hit (query policy, response padding).

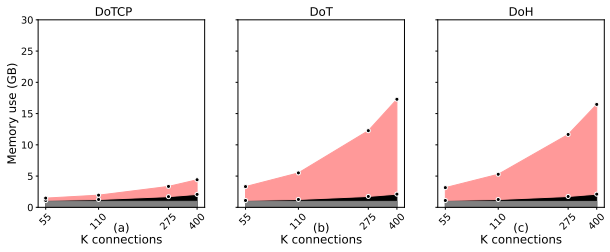
C. Memory usage of keeping connections alive

Transitioning from UDP, a connection-less protocol, to connection-based ones raises the question of the maximum number of connections a resolver can handle. Therefore, we have devised an experiment aiming to measure the limits (in terms of memory) of the number of connections that can be handled by a server.

For each protocol we tested (DoTCP, DoT and DoH), we used Flamethrower to generate as many connections towards our resolver as possible, spread across our 20 machines. We configured both clients so that they would not close established connections, and increased the kernel-side limits on the number of outgoing or incoming connections. We measure two separate values: the Resident Set Size (amount of memory used by the process present in physical RAM) of the resolver, and the total amount of memory used on the machine. By calculating the difference between these two values, we are able to estimate the amount of memory used by the kernel. There was an option in dnsdist allowing for the release of memory linked idle connections, which we chose to deactivate as our interest lied in estimating how much memory an active connection would consume.



(a) Knot-resolver



(b) dnsdist

Figure 3: Memory usage of the resolvers relative to the number of connections

Figure 3 shows, for each protocol, the total physical memory used relative to the number of connections. At the top (in red) is the resident set size of the resolver, in the middle (in black) is memory used by the kernel, and, at the bottom (in grey) is the amount of memory when the server is idle, presented for reference. The memory used by the kernel is the same in every experiment, which is consistent with the fact that the kernel only handles TCP connections, the common part between the three protocols. For both resolvers, we observe an increase in memory consumption when switching from DoTCP to DoT, as the handling of TLS sessions requires additional state, handled by the resolver. When switching from DoT to DoH, we can notice a difference in behaviour between both resolvers. When considering knot-resolver, we see a clear increase in memory consumption between DoT and DoH, due to the fact that the DoH stack of knot-resolver is built upon its TLS stack. Therefore the memory consumed by HTTP2's protocol layers are added to the memory consumed by the TLS

layer. For Dnsdist, we observe that DoH has a lower memory consumption than DoT. While this seems counter-intuitive, it is consistent with the memory consumption per connection and protocol announced in its documentation. As the number of simultaneous connections never reaches 400 000 in the following experiments, we conclude that memory consumption won't be the limiting factor in our setup.

D. Cost of handling queries

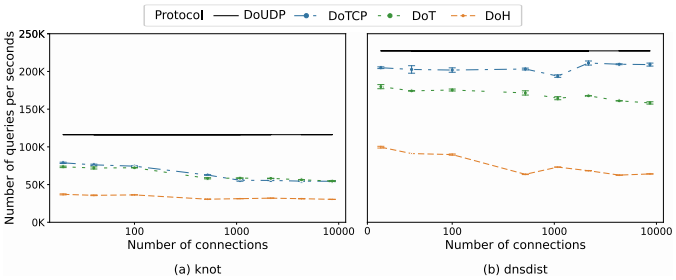


Figure 4: Queries per second handled by both resolvers when connections last all experiment

While the use of these new connected protocols seems to cause no issue regarding memory consumption, it can induce a CPU overhead due to the additional steps required when handling messages. These can be broken down into two parts : First, the connection establishment, and then, the handling of individual messages (see section II). The experiment described here aimed to estimate the cost of handling individual queries.

In order to measure the additional cost per request, we must take into consideration the number of simultaneously opened connections over which requests are sent. To do this, we sent a fixed amount of traffic over a variable number of already opened connections. The total fixed amount of queries sent, as well as the minimum number of connections was chosen to ensure that the CPU utilisation of the resolver process and the frequency of the core it ran on, were as high as possible. We ran into an issue when using Flamethrower, meaning we could not exceed a certain number of queries per second with DoH, so the number of queries per second sent are not the same between DoH and DoT / DoTCP. However, all tested configurations allowed us to reach 100% CPU use, which means that this issue does not affect the validity of our results.

Each point on Figure 4 represents the average number of queries per second that were successfully answered by the tested resolver, with bars presenting the minimum and maximum value reached, for a specific protocol and a specific number of connections. We also represented the max traffic handled with UDP for comparison purposes. For every connected protocol there is a performance drop when compared to UDP, due to the management of the state associated with the connections. We observe a noticeable performance between DoTCP any DoT, for dnsdist only while for knot-resolver performances are the same. Since knot-resolver has to execute more tasks than dnsdist upon receiving a DNS query (as discussed in section V-B), the additional cost of symmetric encryption is absorbed by the cost of handling DNS queries, and as dnsdist has less work to do upon receiving a query, the cost per message added by symmetric encryption has a bigger impact. When comparing DoH to other protocols, we notice, for both resolvers, a huge drop in performances (by a factor of two), explained by the added cost of the HTTP/2 protocol layers. For every protocol, we observe that performances tend to drop when the number of connection increases (up to a 40% decrease when considering dnsdist over DoH), except for dnsdist over TCP.

E. Overhead of establishing connections

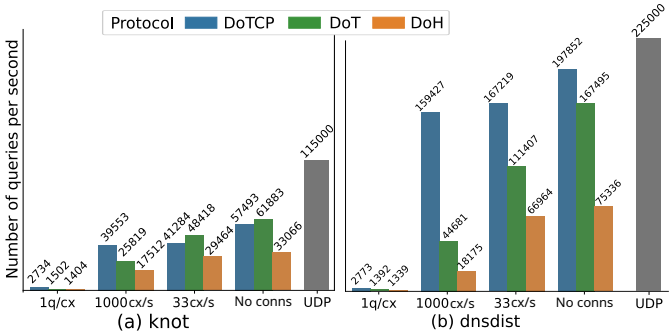


Figure 5: Queries per second handled per resolver and protocol according to number of connections per second

In the following experiment, we measure the cost of opening and closing connections for each protocol. We use the same experimental parameters as the one used in the previous experiment (figure 4) to plot the point corresponding to 1000 connections, taking into account connection establishment and tear down. Thus, by comparing this experiment and the previous one, we can infer the overhead of connection establishment (TCP connection establishment, TLS key exchange). We run two sets of experiments, one in which connections last 30s, the other in which connections last 1s (to match the long and short-lived connections we observe in IV). This means that, in the first experiment, 33 connection establishments occur every second, while in the second experiment 1000 connection establishments occur every second. We run an additional batch of experiments to reproduce the very short-lived connections also observed in section IV.

The results of the experiment are presented in Figure 5 with the number of queries per second handled by the resolver for each combination of resolver, protocol, and connection duration. We also plotted the maximum queries per second we reached in the previous experiment (figure 4) as a baseline for comparison. In general for knot and dnsdist resolvers we observe the same performance variations between protocols but with different orders of magnitude : When few connection establishments occur (connections last for 30s), we observe a 20% decrease in performances for TCP and DoT, relative to when no connections establishment occur. for TCP and DOT lose around 20% performances. However, we do not observe this performance loss for DoH, as the CPU cost of query management still is the bottleneck.

When the frequency of connection establishments increases (connections last for 1 second), the performances of TCP do not change, while both encrypted protocols see a decrease in performances due to the added cost of the TLS key exchange.

When connections are used for one query only, the cost of establishing TCP connections induces a collapse of performances for all protocols.

	KNOT			DNSDIST		
Protocol	1 q/cx	1000 cx/s	33 cx/s	1 q/cx	1000 cx/s	33 cx/s
UDP	0.12 kWh			0.05 kWh		
DoTCP	3.11	0.34	0.33	3.08	0.085	0.08
DoT	8.98	0.52	0.28	9.46	0.30	0.12
DoH	9.675	0.775	0.46	9.85	0.74	0.2

Table I: Estimation, **in kWh** of the energy consumed by running 10k qps for one day using the measured profiles

For all of the experiments we measured the energy consumption of the resolver, first at idle, then during the experiment, when one core is fully loaded, and the difference gives us the lower bound of the energy consumed by these protocols. In order to compare these protocols we compute the cost for one request and use this cost to obtain the energy consumed, in kWh, of handling 10k qps for a day (results in table I).

If connection use is fair (30s) the use of secured protocols is worth considering, even though it increases energy consumption by a factor of two for DoT and four for DoH, but when we consider a higher load, in which connection openings occur more frequently the use of such protocols become more costly (up to 15 times for DoH at 1s), or un-sustainable in the case of aggressive connection opening (0.05 kWh versus 9.85 kWh).

VI. Conclusion

DNS is still at the core of today's internet. Originally designed for performances, using an un-encrypted connectionless protocol, growing concerns about security have led to the standardisation of secured protocols. In this article, we studied the resolver-side cost of transitioning to such protocols.

We benchmarked public resolvers using various DoH clients to gather profiles. We found that all entities tried to maintain connections alive but that, when faced with a high load, browsers became unstable and we observed of a very high number of small connections. Then, we measured the additional cost of each step of the connected protocols (DNS over TCP, DoT, DoH). We observed that transitioning from the legacy protocol to a secured one lead to, at minimum, a division of the performances by two, due to TCP connection establishment, TLS key exchange and message encryption, and that the performance loss was even more noticeable when considering the switch to DoH because of the protocol layers added by HTTP/2, which, when considering the fact that the use of this protocol seems to be pushed by the industry, is contradictory with DNS's initial objectives of efficiency. Furthermore, in the case in which connection are short, performances take an even bigger hit, which can lead to a large increase in the number of resolvers as well as their energy consumption, leading in turn to a higher operating and environmental cost. As it is, switching 100% of the DNS traffic to DoH is not sustainable. To realise this transition, it will be necessary to ensure that clients can keep their connections alive as much as possible, and to use less costly protocols than HTTP/2, that still retain its ability to go through firewalls. Therefore, it could be interesting in the future to look at the still in development, QUIC-based HTTP/3.

VII. Acknowledgments

This work is supported by the ENE5AI project (DOS0185314/DOS0185315/DOS0185931/DOS0185932), the FOG SLA project (DOS0168403/00-DOS0168405/00) and the ANRT (CIFRE n°2022/0178)

References

[1] D. Balouek et al. "Adding Virtualization Capabilities to the Grid'5000 Testbed". In: *Cloud Computing and Services Science*. 2013.

[2] M. Belshe, R. Peon, and M. Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540.

[3] S. Bortzmeyer. *DNS Privacy Considerations*. RFC7626.

[4] Timm Böttger et al. "An Empirical Study of the Cost of DNS-over-HTTPS". In: *Internet Measurement Conference (IMC '19)*. 2019.

[5] Bushart and Rossow. "Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS". In: *10th USENIX Workshop on Free and Open Communications on the Internet, FOCI 2020*.

[6] S. Garcia et al. "Large scale measurement on the adoption of encrypted DNS". In: *arXiv preprint arXiv:2107.04436* (2021).

[7] P. Hoffman and P. McManus. *DNS Queries over HTTPS (DoH)*. RFC 8484.

[8] Austin Hounsel et al. "Comparing the Effects of DNS, DoT, and DoH on Web Performance". In: *Web Conference 2020 (WWW '20)*. 2020.

[9] Z. Hu et al. *Specification for DNS over Transport Layer Security (TLS)*. RFC 7858.

[10] G. Lowe, P. Winters, and M. Marcus. "The great DNS wall of China". In: *MS, New York University* (2007).

[11] I. M. M. Dissanayake. "DNS Cache Poisoning: A Review on its Technique and Countermeasures". In: *National Information Technology Conference (NITC '18)*.

[12] ns1labs. *flamethrower*. URL: github.com/DNS-OARC/flamethrower.

[13] Etienne Le Louët ns1labs. *flamethrower*. URL: github.com/etienne-lelouet/flamethrower.

[14] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446.

[15] S. Siby et al. "Encrypted DNS ==> Privacy? A Traffic Analysis Perspective". In: *The Network and Distributed System Security Symposium (NDSS '20)*.

[16] D. Vekshin, K. Hynek, and T. Cejka. "DoH Insight: Detecting DNS over HTTPS by Machine Learning". In: *15th International Conference on Availability, Reliability and Security (ARES '20)*.