"No Port 53, Who Dis?"
A Year of DNS over HTTPS over Tor

Alec Muffett (alec.muffett@gmail.com) — v3.1 final

An analysis of using home broadband with considerable freedom from DNS tracking, surveillance, and blocking, through exclusively using DNS over HTTPS over Tor.

Abstract

Users of DNS over cleartext UDP port 53 (Do53) — i.e. most users of the internet — are at risk from specified privacy and integrity threats, not all of which risks are mitigated by authoritative content signature schemes such as DNSSEC. DNS-over-TLS (DoT) by design does not address several of these risks. DNS-over-HTTPS (DoH) obviates many but not all of the risks, and its transport protocol (i.e. HTTPS) raises historical concerns of privacy due to (e.g.) "cookies." The Tor Network exists to provide TCP circuits with some freedom from tracking, surveillance, and blocking.

Thus: In combination with Tor, DoH, and the principle of "Don't Do That, Then" (DDTT) to mitigate request fingerprinting, I describe DNS over HTTPS over Tor (DoHoT).

Since February 2020, using off-the-shelf open-source software, I have provided DoHoT to my home network. A `dnscrypt-proxy` caching resolver presents locally as a Do53 resolver that is exclusively configured to make outbound resolution DoH calls over Tor. I have — aside from necessary heartbeats and bootstrap — blocked all outbound port 53 & 853 traffic at my firewall, in order to prevent leaks. I have not sought to prevent other forms of DoH traffic because I am less interested in the challenge of constraining name resolution than I am in enhancing its privacy and integrity.

After an initial five months of testing, tuning, selection of DoH servers, and being forgotten about in the light of world news, in the subsequent seven months (ending February 2021) the DoHoT system has issued more than 1.6 million DoH requests over Tor to a pool of 9 public DoH resolvers, and served an additional 773k responses to clients from cached results. I share performance statistics, a list of technical prejudices that I was told to expect, describe my failure (for the most part) to experience them, and a summary of the experiences of two people relying entirely upon this system for work and personal life during COVID-19 "lockdown".

Apology

This report was canvassed by the chairs of NDSS DNS Privacy and written in haste, therefore it eschews the niceties of LaTeX, faux-first-person "we" and two-column formatting. Proofreading is by the kind grace of friends and Twitter users.

Context

Prior to 2020, I was using a Raspberry Pi running *Pi-hole* software as a caching resolver (with use enforced by firewall blocks) to filter advertising and tracking websites; I found that Pi-hole was intruding upon the use of some streaming TV services (notably: those supported by advertising) and decided to decommission it; but rather than remove the infrastructure entirely, I decided to run a long-intended experiment of using DoH over Tor as my primary means of name resolution, presenting it to my home Do53 clients via a similar caching resolver.

Goal

The goal of this project was to determine *whether it is reasonable for a user at home to implement or deploy DoHoT for domestic use*, and *whether the resulting system is adequate for use*. From my 12 months of experience, the answer to both of these questions is "yes".

A non-goal for this project has been active pursuit of comparative performance metrics; although it would be reasonable to build a Selenium framework and measure the negative performance impact of DoHoT upon page-loads for popular sites — and there *must* be a negative performance impact, because additional infrastructure perforce will add cost — but to do so would not speak to the primary goals: we could easily fall into the trap of measuring *cost*, ignoring *benefit*, and never considering *value*.

Some people consider *"advertising and tracker blocking"* to be "value", and hence Pi-hole exists as a solution, amongst others. Some consider *"logging, instrumentation, and breach-detection"* to be "value". Others — including myself — desire *"minimal proliferation of linkable metadata"* (i.e. "privacy") and *"assured access to preferred sources of truth"* (i.e. "availability" and "authenticity") as value, and are willing to trade some performance in order to obtain it so long as the resultant system's performance is still tenable. DoHoT provides the latter two, and the former two can be layered on top of it quite easily.

Potential Confusion

There is much ongoing research in the space of using Tor for DNS resolution — and other privacy-enhanced DNS mechanisms — but DoHoT is particularly at risk of confusion with:

Using Tor's embedded Do53 server for DNS resolution

Every Tor daemon contains a UDP Do53 DNS resolver, controlled by the `DNSPort` configuration option, and which is disabled by default. The daemon is limited to handling A, AAAA and PTR requests, and works by parsing the request and using a node on the Tor network to perform the resolution, passing the result back to the client. The feature is not widely used, and the user has no active control over the privacy, integrity, authenticity, nor trustworthiness of the upstream resolution process.

DoHoT does not use this mechanism at all.

Using Tor {Onion, Hidden} Services for DNS resolution

In 2018, Cloudflare opened access to their "1.1.1.1" DoH service using a Tor Onion Service, using Tor's unique "onion networking" feature to provide Tor-capable DoH clients the privacy, authenticity, integrity and availability/unblockability "layer-3" guarantees which are offered by onion networking. Cloudflare subsequently published a "fun stuff" guide on how to configure clients or stub-proxies to use it.

DoHoT makes use of the Cloudflare Onion DoH server as part of its load-balanced pool of DoH servers; it also makes use of the non-onion Cloudflare 1.1.1.1 DoH service, as well as several other public DoH services. All DoH services are accessed in the same manner as any other HTTPS connection might be routed over the Tor network, the only distinction being that the Onion DoH service has a non-DNS, non-TCP/IP, onion address in its URL.

DoHoT uses Tor, but is not limited to onion networking.

Threat Model

Risks of Do53

In a "more complete" threat model, it is important to consider the risks of the Do53 protocol in both its *technical* and *political* (i.e. policy-based) aspects.

Considering Do53 as a *service*, users are at *technical risk* including:

- transport blocking: access to Do53 services may be entirely prohibited, or requests may be selectively dropped, impacting availability of the desired (or of any) service
- transport spoofing: a request to a specified Do53 server may be intercepted at layer-3, and a response may be forged and dispatched, by any server on the path
- transport fingerprinting: the layer-3 affinity for a client to a particular Do53 server, may be tracked and used as an identification signal
    - correlation attack: a subclass of transport fingerprinting, timestamps of (DoT?) requests between client & server may be recorded and used to retrospectively surveil and attribute requests, either with respect to log files or observation of the recursive resolver's upstream queries.
    - traffic analysis: another subclass of transport fingerprinting, characteristics of (DoT?) requests, such as packet length, may enable guesses at their content.
- stack fingerprinting: the layer-3 behaviour of a client may be profiled and used as an identification signal
- protocol & content fingerprinting: the layer-7 content of requests between a client and a particular Do53 server, may be profiled and used as an identification signal, e.g. offering of TLS cipher suites or other options.
- active upstream collection of signals & metadata, collusion, or compromise of privacy

These technical risks are the concrete issues which underlie further *risk to user-policy and user-choice*; users of Do53 are at risk of insults to their *policy* which may include:

- breach of utility: requests to a Do53 service of their choice can be responded to by a service other than their expectation, impacting metrics, trust, or logging pursuant to (e.g.) intrusion detection or indicators of compromise

- breach of function: requests to a Do53 service of their choice can be blocked or dropped by an intermediary, impacting local function (e.g. software development domain names that point to containers on localhost)
- breach of privacy: requests to a Do53 service of their choice to lookup custom or internal domains (etc) can be surveilled or logged, leading to leaks of metadata (e.g. internal development project names)
- impeding user choice: a user may have chosen to pay for use of a specific Do53 service. That the Do53 protocol historically permits — even encourages — third parties to block or tamper with that service is an insult to the customer relationship, user choice, the "end-to-end principle", and network neutrality

Observation: it is easy to overlook correlation attacks and traffic analysis when considering Do53 in a standalone manner, because Do53 is a cleartext protocol so these attacks are irrelevant; they become relevant when the payloads are encrypted (e.g. DoT).

Accepted Risks

DNS is a distributed database and relies upon mechanisms such as recursive resolvers, so that a query sent to one resolver may be passed upward through, surveilled, logged, or filtered by several other servers prior to a response being sent back to the user.

There is little or nothing that the user can do to mitigate the risks of this situation, other than to be selective regarding what queries are made, and to where they are sent. For this reason, we shall simply accept those risks and note in passing that the choice to use "Trusted Recursive Resolvers" (TRRs) can and will help provide "minimal proliferation of linkable metadata".

From this we can extrapolate that use of any transport protocol which does not assure both *authenticity* and *fail-closed availability* of communication with the TRR, is complicit in insulting user choice and breaching implicit policy.

Rejected Risks

Some literature goes to great lengths to explore the downsides of providing DNS Privacy, notably the Wikipedia entry on DoT and a similar Cloudflare article on DoT, both of which propose the importance of "*[giving] network administrators the ability to monitor and block DNS queries*" in pursuit of obtaining indicators of compromise (IOCs) from a network-observer vantage point, or leveraging Response Policy Zones (RPZs) and similar.

Ignoring the fact that DNS IOCs can also be obviated by use of technologies such as Tor, VPNs, or by the use of simple "hard coded" IP addresses, I believe that it is strategically unwise to prefer or promote "man in the middle" solutions in a world pursuing ever more end-to-end encryption and metadata-stripping through initiatives like TLS1.3 and Encrypted {Server Name Indication ESNI, Chat Hello ECH} in TLS.

I believe it would be wiser, instead, to pursue better instrumentation and monitoring of all devices within an enterprise trust perimeter, for instance via Mobile Device Management (MDM), by configuring per-client stub resolvers and filters in lieu of RPZs, or by configuring enterprise devices to use a dedicated or outsourced enterprise DNS service over DoH, enforcing that use via MDM. Individuals may pursue their choice of value-added DNS services, and states which wish to constrain what hosts their citizens should be able to access, are already part of a wider civil society discussion on this topic.

Risk mitigated by DoH

Use of HTTPS as a DNS resolution transport obviates risks of content-tampering, content-surveillance, response-spoofing, and some forms of content-based fingerprinting. Reciprocal to being proof against response-spoofing, DoH also assures the identity of your chosen DNS resolver.

Risk added by DoH

HTTPS protocols have a richer set of fingerprinting- and tracking- and metadata-linkage opportunities than basic forms of HTTP requests; they include:

- HTTP Authentication: require the user to "log in" to use the service
- HTTP Cookies: creating trackable / linkable "sessions" between requests
- TLS Session Tickets: linking requests by TLS metadata
- TLS Fingerprinting: linking requests by TLS characteristics, such as cipher-suites
- HTTP Request Fingerprinting: existence of unique or deployment-distinct characteristics of HTTP client stacks

Many of these risks are addressable by the principle of "Don't Do That, Then" (DDTT) — so DoHoT deployments should avoid authentication, not send or accept cookies, avoid using TLS session tickets, standardise cipher suites in order to defeat TLS fingerprinting, and attempt to avoid distinctiveness. Fortunately, `dnscrypt-proxy` supports most or all of these features out-of-the-box.

Some say that an additional risk of DoH is that clients may use a multitude of DoH services and avoid enterprise control; I suggest this is better managed via MDM than by "blocking", not least because blacklists don't work and also don't reflect nomadic or home usage.

Risk Mitigated by Tor

The Tor Project exists to provide enhanced client (and, sometimes, server) communications integrity and privacy, by reducing the linkability of data communicated between client and server, and by providing network-block-circumvention technologies.

The mitigations provided by Tor complement those provided by DoH and address most of the remaining concerns in our threat model; layer-3 stack fingerprinting and tracking of IP address is obviated by Tor, and transport blocking is also obviated; even wholesale "blocking" of Tor is actively combatted (via "Bridges") by the Tor development team. Key benefits of Tor include resistance to traffic analysis and (to a greater extent) correlation attack — the latter further improved by DoHoT's distribution of requests via load-balancing.

Use of Tor adds potential for even greater assurance — Cloudflare's 1.1.1.1 DoH resolver is available as a Tor Onion Service, which is presented as a cryptographically assured 256-bit layer-3 network address within the Tor network.

Risk added by Tor

Tor's primary disadvantage in this space is one of configurability by people who are less technically capable; for long-term deployment, such as "at home", this is not a significant problem, but for dynamic captive-portal networks (aircraft, railways, hotels, ...) there may be additional complexity to deploying DoHoT. This could be improved on a per-platform basis with modest effort.

An additional risk of Tor in this space is one of population: that DoH requests which "come from Tor" must come from a small set of Tor users. This is currently true, but polling on Twitter has surfaced several other people who are experimenting in this space, not least because of the publication of the DoHoT configuration last year.

Risk mitigated by DoHoT

The DoHoT `dnscrypt-proxy` server load-balances requests amongst a pool of eight public DoH servers, plus an additional ninth "onion" address as described earlier. DoH load-balancing is configured with the default "p2" strategy. In this, requests are sent to a random choice of one of the two currently-fastest servers, plus *another* server selected at random from the set, so that speed rankings are continually refreshed.

Any request that exceeds 10 seconds is killed, and in a small nod towards performance-over-privacy, the default 30 second "keepalive" is retained to reduce renegotiation-loading upon the server pool.

The value of load-balancing follows from Tor's concept that *"anonymity loves company"*; having made a choice of several DoH servers, we can best address our threat model by having many users utilise all of those servers consistently and without linkable data, so that there is no reason for a service to respond to any given request in a "special" manner. Thereby we pursue equitable treatment and fair responses to our requests.

Compared to DoT

DoT addresses several core threats in our model, including: request surveillance, response forging and response tampering. DoT does not address:

- Transport Blocking: hardwired to port 853, DoT is strongly identifiable and trivially blockable, likely forcing a downgrade to Do53
- Correlation Attack: an observer who can observe the upstream and downstream of a DoT server will likely be able to surveil and then attribute requests to a user, defeating the intention of DoT
- Traffic Analysis: from several resources (Bortzmeyer, and links in that post) it is not simple to determine how resistant given DoT implementation is to traffic analysis — and not obvious to determine what will happen if/when there is a capability mismatch, e.g. when a desired padding option is not supported by the server

DoHoT leverages Tor to resist both blocking and correlation attacks, and resistance to traffic analysis is a leading feature of Tor's design; that these features can be improved separately and in a manner disjoint from "publishing an RFC and waiting for the world to adopt the new feature" is a positive benefit to operational privacy.

Compared to Oblivious DNS over HTTPS (ODNS, ODoH)

The original ODNS paper includes a section on page 3 marked "*Why not Tor?*", and the very first cited drawback of Tor-based solutions is "*Tor's fundamental design introduces substantial network latency to the end-to-end path*", continuing in that vein to discuss the *cost* of using Tor for DNS, rather than the *value* of using Tor for DNS.

To be fair to the authors: at the time or writing the features and opportunities of DoH were not entirely settled, and the associated presentation suggests that the authors considered DNS-over-Tor either in terms of "onion services" — which as this report shows, have much higher latency than ordinary DoH performed over Tor — or else in regard to Tor's embedded Do53 server (paper: "*Additionally, DNS in Tor is conducted by the exit node of the circuit...*") — neither of which are relevant to DoHoT; see "Potential Confusions", above.

The authors continue to raise two other criticisms of Tor, that *censors attempt to block Tor* and that *exit node operators may be held accountable for acts of DNS resolution performed over Tor*, both of which sound curiously obvious to anyone who is enmeshed with the Tor community. The authors further conflate *resolution of DNS over Tor* with some research on how DNS can be used as a back-channel to deanonymise Tor users, and other research of how the ".onion" top-level domain sometimes leaks into DNS resolution.

From my perspective these latter points are not germane, thus I believe that ODNS was founded upon an assumption that 200-to-300ms median DNS resolution latency is unfit for client use. My lived experience contradicts this, especially given that I value the benefit that DoHoT provides. The qualitative performance experiment that I describe below, also lends weight to my perspective that higher latencies, well managed, are acceptable.

Yet upon this assumption two entire privacy-proxy protocols have been created — ODNS and ODoH — both of which fail to address the DoHoT threat model in several ways; they are both vulnerable to "watering-hole style" timing and correlation attacks, that a small tier of machines dedicated to offering privacy are an ideal candidate for backbone surveillance. Equally: any small and dedicated tier of "oblivious DNS proxies" are an obvious candidate for blocking by illiberal regimes or censorious ISPs. Further: both protocols are vulnerable to intentional or unintended ("shared log files") collusion between proxy- and resolver- services. ODoH in particular *may* risk a user configuring a colocated/owned proxy & resolver.

As such, I believe that the DNS Privacy community would benefit from broader reflection upon who needs or demands DNS privacy, what their threat models may be, and what performance tradeoffs those people might be willing to accept in order to obtain that value. Work by some ODNS-related researchers demonstrates thinking in this direction, but it is hard to shake the impression that following an ongoing attempt to bend DNS to reinvent Tor, there is now an additional effort to bend DNS to reinvent HTTPS load-balancing — irrespective that extant and original solutions already exist for both.

Why implement DoHoT rather than DoToT?

DoH and DoT are both TCP protocols that could be tunnelled over Tor, so why choose DoH for this experiment? Four reasons:

1. HTTPS is a mature transport technology that is evolved, improved, and bug-fixed independently, whereas DoT has only comparatively recently started to address issues such as "padding"; and software implementations are rare
2. The challenges of HTTPS load-balancing are well-understood and addressed
3. Wrapping a binary DNS query in HTTPS, or converting it into JSON, greatly increases observability and access to "debug" tooling, reducing scope for undetected small binary features that might aid tracking of the request
4. TCP port 443 for HTTPS is likely the protocol most frequently carried over Tor; TCP port 853 for DoT might experience blockages from cautious exit-nodes, hackers, state agencies, and over-enthusiastic security researchers

Prejudices versus Experiences

Others, hearing about this project for the first time, tend to make presumptions about the impact upon user experience that DoHoT will have; let's address some of them here:

- **DNS request latency will increase**: Yes, it does. The result is still okay. Sometimes clicking on a link may take a moment longer before the page starts to load, but that was much the same when I was using Pi-hole. Most of the time it's not noticeable. Some (approximately) 33% of requests are served from cache, so a cache is clearly essential in delivering performance. Cache size was set at 4096 entries which seemed reasonable and large enough to not be typically exhausted.
- **Page-loading times will increase**: except for initial hits on uncached domains I haven't observed this happening, which I attribute to clever browser behaviour that has sought to pipeline requests during page-render for many years now, so that DNS resolution lag is lost in the noise, below human perception, or simply not bad enough to be worth noticing.
- **Conferencing & Streaming services will stop working**: Netflix, Amazon Prime, YouTube, and BBC iPlayer all work well; Zoom, Skype, and Microsoft Teams also work well and have been extensively tested.
- **Gaming lag will increase**: I cannot test this one because I'm not a console or PC gamer, but I can't see that games will need to make indefinite numbers of distinct DNS queries in real time, although I am open to correction.
- **You will be directed to {wrong, distant} sites, impacting performance**: I have probably not (*see note*) observed this happening; the theory is that if I am in the UK but use a DoH server in Japan, then via the Japanese server's name-resolution my browser will be blindly directed to access a Japanese CDN with consequent "long-haul" impact upon performance. Perhaps this is happening a little and I am not noticing, but generally I believe that — as DoHoT is not a VPN and my web requests come from my authentic, geo-locatable IP address in the UK — most services are dynamically generating UK-geolocalized CDN URLs which globally resolve to a "nearby" address, leading to negligible loss in user experience.
  - **note**: there *was* one frustrating evening where the GoDaddy website was inexplicably convinced that I was in the Netherlands, but this may have been an upstream issue.
- **You will cause extra load to bearers**: Perhaps, but less than when using a VPN tunnel to another country. Also: one of the goals of DoHoT is to strip DNS metadata from traffic passing through my broadband connection, so that with the adoption of TLS1.3 and ESNI/ECH, casual observation of my broadband link will yield no

information other than destination IP addresses, packet sizes and flow rates. This would offer most of the benefits of using a VPN without the "everything visible to a single vendor" risk to privacy, and reduced "traffic-doubling" across backbones.

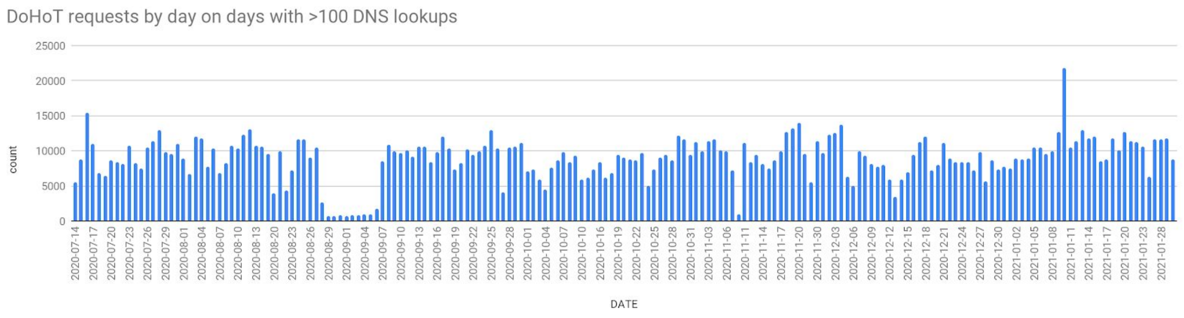
- **Other DoH clients on your local network might bypass your blocking**: Fine, at least they won't be observed by my ISP.
- **IoT will get confused**: Chromecast and Portal have been observed to be making Do53 requests to 8.8.8.8 and 1.1.1.1 (respectively) in defiance of the DHCP DNS resolver advertisement which provides DoHoT; Chromecast in particular makes such a request every 20 to 30 seconds. Nonetheless, both devices still function. No other issues have been observed.
- **How can you blindly trust a DoH resolver to do what you want?**: Frankly, I cannot, but I don't mind because I am using a pool, and (to a reasonable extent) they don't know who or where I am, and therefore should treat my requests equitably; and if any given one ceases to treat my requests equitably then it will be dropped from the pool; but at the very least I am strongly assured that I am talking to that pool of DoH resolvers rather than some spoofing intermediary via untrusted Do53. The members of the pool are chosen to promise to use DNSSEC upstream, and to promise a lack of filtering. The proxy software enables auto-selection of these characteristics from an even larger pool.
- **The Recursive DoH Resolver can track your {IP Address, TCP Fingerprint}**: this is dealt with by Tor and the cloud of exit nodes.
- **The Recursive DoH Resolver can track your {Authentication, Cookies, TLS Tickets}**: this is dealt with by DDTT.
- **The Recursive DoH Resolver can track your Requests**: Hopefully I've given them nothing to link that to, other than perhaps a few seconds' worth of concurrent requests in a single TCP session — some requests of which might be to "nonce" domain-names that would enable some forms of fingerprinting but require collusion with the site I am accessing. I am content with that, given that the threat model is mostly directed at privacy from third parties that exist outside of my trust model.
- **Why don't you measure a cloud of non-Tor DoH?** my goal with this project is to address my threat model, and — although `dnscrypt-proxy` would bring some benefits such as DDTT to raw DoH — my threat model would not be addressed by a plain DoH solution, even if it were somewhat faster.
- **What about IPv6 lookup?** I don't have IPv6 at home so have not been able to test this; therefore I disabled AAAA-resolution in the configuration for simplicity.
- **What about record types like SRV, NAPTR & DNSKEY?** They work as normal.
- **This is just your experience!** Yes, absolutely. The test above is *"whether DoHoT is reasonable to use?"* and my experience of TCP/IP networking dates back (just) to the days of ARPANET, so perhaps I am more forgiving than the modern consumer. I believe the most accurate statement that I can make is "DNS via DoHoT on a good broadband connection does not feel particularly worse than (say) DNS on a good-quality 4G network, and the experience overall is wholly acceptable."

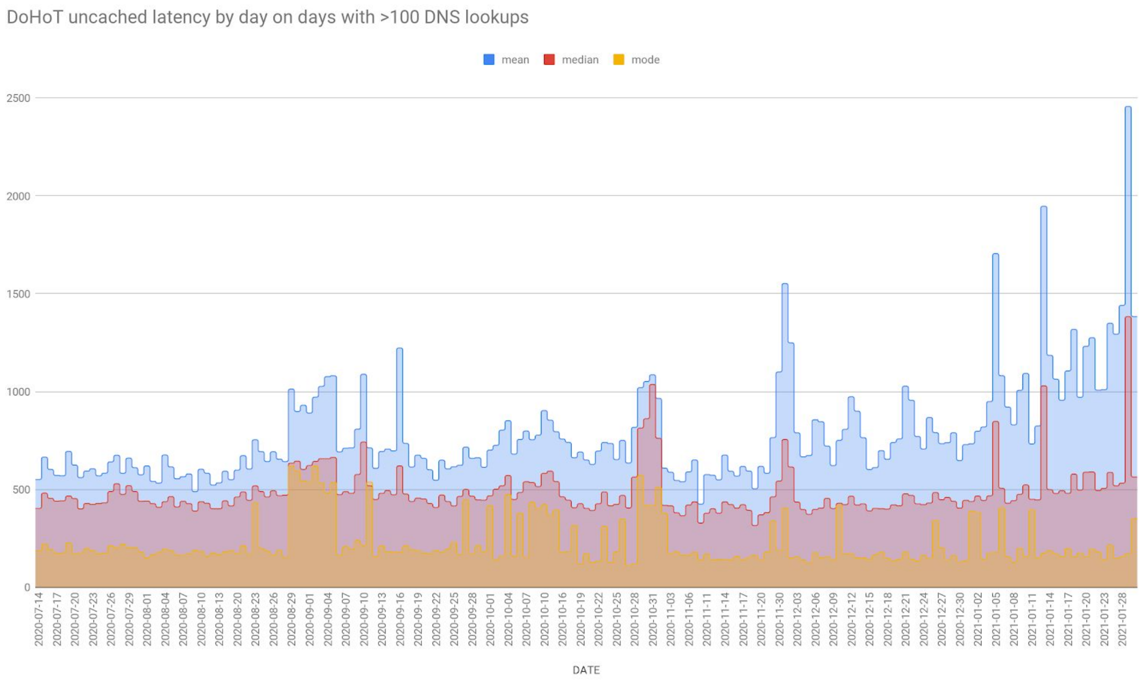
Measuring Technical Performance

My home network is a robust 70-down/20-up megabit DSL connection, and the DoHoT server is a dedicated RaspberryPi model 3B attached via Wi-Fi. All numbers come from the `dnscrypt-proxy` log files, which are archived weekly and are reduced to unique data points, then analysed with custom awk scripts and the python `statistics` package.

No attempt has been made to compensate for systemic outages, occasional data loss, traffic delays due to household remodelling, power outages, upstream network outages, local network outages, nor acts of Roomba. Therefore six days have been removed from the "daily" stats, on the grounds that the availability of 100 data points is a reasonable minimum requirement to perform a "daily" percentile analysis.

DoHoT requests per day, and latency, on days with >100 requests



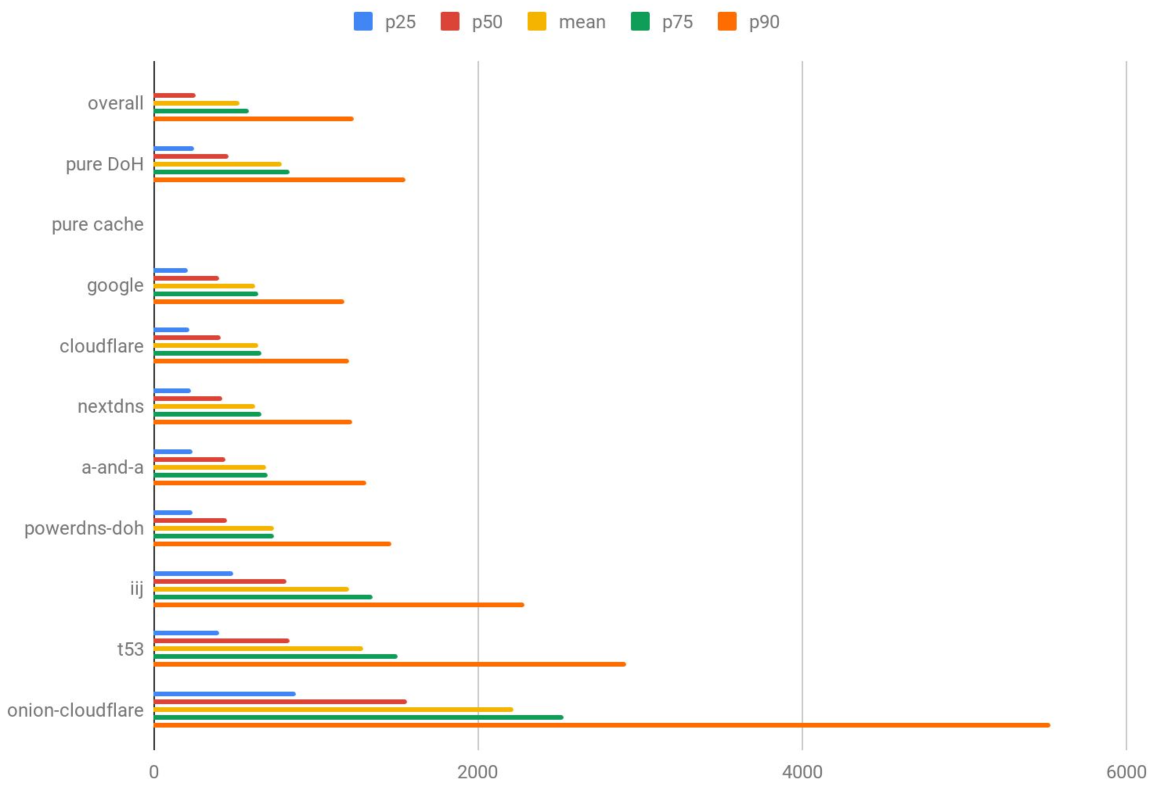


Aside: clearly visible in the statistics is our 10-day vacation in Aug/Sep 2020, during which time the home network was hardly used; it's interesting to see a precisely corresponding uptick in latency, presumably due to each request needing to be served by a fresh network connection over Tor. The mean-daily-median figure for resolution over the central 8 days of that period is 640ms, compared to overall medians of 262ms (including cache) or 464ms (excluding cache). The mean-daily-mean over that period is 986ms. I believe that it is reasonable to consider this as the worst case network behaviour, certainly demonstrating the benefits of both connection "keep-alive", and caching.

Whole-timeframe overall (blue) and per-DoH-server (white) latency

server	**count**	**min**	**p25**	**p50**	**p75**	**p90**	**p95**	**p99**	**max**	**mean**	**mode**
overall	2406088	0	0	262	592	1238	1873	5555	10726	536	0
only DoH	1633002	30	251	464	838	1558	2492	6483	10726	789	170
only cache	773086	0	0	0	0	0	0	1	2176	0	0
google	321821	37	213	403	644	1174	1716	5544	10007	632	164
cloudflare	317427	42	221	417	669	1207	1775	5595	10726	649	160
nextdns	278311	30	234	423	669	1227	1731	4379	10003	631	170
a-and-a	270325	45	242	446	708	1317	1905	5568	10216	691	171
powerdns	180415	37	238	455	745	1467	2330	6249	10001	748	180
iij	106336	50	494	825	1356	2290	3619	7438	10021	1204	428
t53	105285	57	404	844	1508	2917	4352	7785	10037	1291	204
onion-cf	53082	194	876	1563	2534	5535	6988	9102	10218	2224	1484

Excluding "count", all other numeric columns are in milliseconds; a 10-second timeout is configured and enforced by `dnscrypt-proxy`, which is reflected in the "max" column.



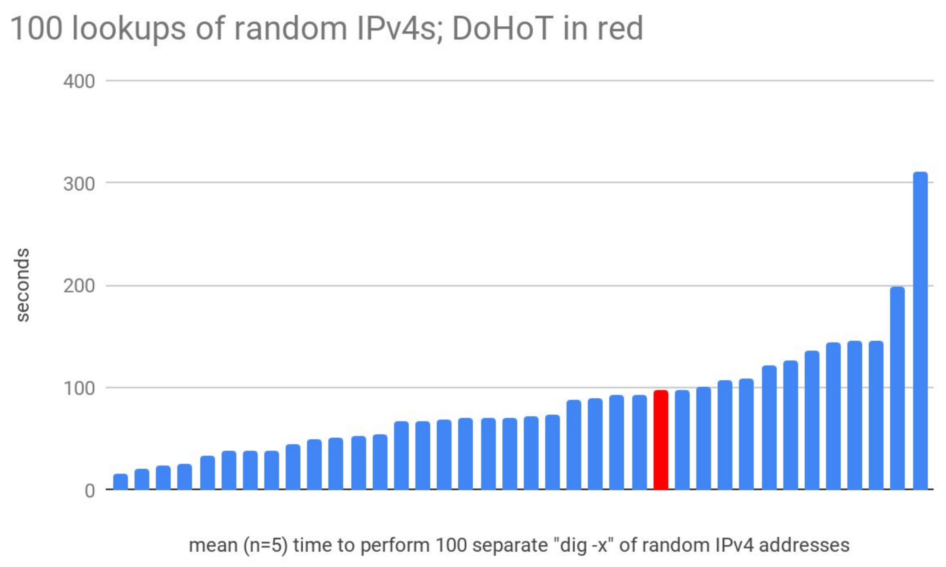
Some, possibly all, of the increase in mean resolution time during January 2021 may be attributable to qualitative testing for the purposes of this paper, caused by several thousand distinct reverse-resolutions during test development.

Summary for clarity: the median request latency for my DoHoT deployment, measured over this period is 262ms, and if locally cached results are ignored the median is 464ms. I have found this to be an acceptable user experience.

Measuring Qualitative Performance

Feedback for an early version of this report noted that, as a community, we suffer from the inability to "*capture 'user experience', as opposed to purely benchmarking latency or page loads*" — and I wholeheartedly agree.

In order to address this lack, I ran an experiment with a small set of volunteers who are distributed around the globe. They ran a script — sometimes in different DNS configurations or on different systems — which performed 5 passes of 100 separate invocations of "dig -x" of randomly generated and possibly nonexistent or illegal IPv4 addresses. The goal was to avoid any benefits of caching and to force an end-to-end resolution to be performed.



The results are fascinating, and for clarity I have picked out the DoHoT system in red (at the 68th percentile) in the comparison graph.

The fastest, lowest-latency user "jg1" is based in the US, has fibre to the premises, a "business"-grade connection to their ISP, and is located close to core internet connectivity; however other users, both above and below DoHoT in the latency rankings, are using "Pi-hole" proxies and other stubs and filters to provide added value and control to their DNS experience.

Experiment: 5x100 "dig -x" timings, mean, and spread; up/down in megabits

source	up	down	dns	time1	time2	time3	time4	time5	min	max	mean	spread
jg1a	40	60	google	16	16	15	22	14	14	22	16.6	8
am2a			hetzner	14	24	19	18	28	14	28	20.6	14
am2b	10	100	virginmedia	17	31	25	29	16	16	31	23.6	15
jr1e	100	1000	google	22	27	30	23	28	22	30	26	8
ya1b	10	100	google	21	29	50	31	40	21	50	34.2	29
pr1a	12	350	xfinity	53	30	30	41	36	30	53	38	23
pb1c	20	200	virgin media	33	46	36	44	33	33	46	38.4	13
jr1a	100	100	isp	48	37	49	30	29	29	49	38.6	20
pb1d	20	200	virgin media	51	45	38	53	38	38	53	45	15
le1g				51	53	52	48	44	44	53	49.6	9
pb1b	20	200	virgin media	59	54	40	47	54	40	59	50.8	19
pb1a	20	200	virgin media	61	45	75	45	41	41	75	53.4	34
ah1a	12	50	pihole-google-tpg	45	48	56	72	52	45	72	54.6	27
le1k	100	100	isp	45	107	64	59	58	45	107	66.6	62
pb2a	50	350	google	53	58	85	66	77	53	85	67.8	32
ib1a	35	350	virgin	69	67	67	86	54	54	86	68.6	32
jr1g	100	1000	isp	53	32	62	87	118	32	118	70.4	86
le1f				76	56	87	63	71	56	87	70.6	31
ws1a	500	500	isp	117	68	59	60	50	50	117	70.8	67
le1b				52	89	69	92	61	52	92	72.6	40
le1i				75	77	75	75	70	70	77	74.4	7
le1a				71	114	90	81	82	71	114	87.6	43
dr1a		30	myself	125	104	51	84	82	51	125	89.2	74
le1e				115	94	83	89	81	81	115	92.4	34
le1l				93	93	106	80	90	80	106	92.4	26
am1a	20	70	dohot	122	88	103	106	66	66	122	97	56
db1a	20	200	pihole-cloudflare	89	84	108	80	125	80	125	97.2	45
jr1d	100	1000	cloudflare	115	94	101	82	111	82	115	100.6	33
le1h				102	158	109	110	57	57	158	107.2	101
jr1f	100	1000	cloudflare	92	91	125	113	123	91	125	108.8	34
jr1b	100	1000	quad9_unfiltered	117	155	137	125	71	71	155	121	84
db1b	20	200	pihole-quad9	212	87	101	178	52	52	212	126	160
ya1a	10	100	cloudflare	132	133	151	129	135	129	151	136	22
le1c				152	104	170	132	160	104	170	143.6	66
jr1c	100	1000	quad9_filtered	149	122	208	152	94	94	208	145	114
le1j				143	151	166	128	139	128	166	145.4	38
le1d				198	216	189	235	155	155	235	198.6	80
le1m				309	299	293	374	283	283	374	311.6	91

Many users in the slower 50% of results are in countries such as Ireland, Greece and Russia, even if they are directly using Cloudflare or Quad9 for resolution. This suggests that excessive consideration of latency as a fundamental metric of usability is in part a privilege of technologists who reside in countries which are well served by internet backbone and high-speed "local loop" infrastructure.

Conclusions

DNS over HTTPS over Tor is viable for end-user use. It works. It can be deployed today by technically capable users, anywhere in the world, using "off the shelf" open source software. It provides strong privacy, availability, authenticity, anonymity, and censorship-resistance — not guaranteed, but likely greater than that of any other extant solution. The latency performance cost that it imposes is of a scale which some in the world would consider "normal DNS behaviour", and others would consider an acceptable tradeoff in exchange for blocking adverts (e.g. "Pi-hole").

There are opportunities for research and improvement: pre-warming DNS caches, tuning selection of Tor's first-hop "guard nodes" to assure good service to nomadic clients, but given that deployments of DoHoT are meant as per-client or per-premises "stub resolvers" then it's worth considering how much of any performance tuning effort might simply be wasted and/or better left to other technologies in the user stack, such as the web browser.

Acknowledgements

- Frank Denis, for dnscrypt-proxy
- The Tor Project, for Tor
- The Pi-hole community
- John Todd (Quad9) for technical feedback on the impact of keepalive
- multiple, generous, DNS-performance experimenters
- multiple, kind, paper reviewers from NDSS, Tor, and the Twitterverse

Links

All code, configuration, and previous results are available at the following repository; the pages will be updated with this paper after publication.

- https://github.com/alecmuffett/dohot
    - project home page
- https://github.com/alecmuffett/dohot/blob/master/TECH.md
    - installation guide