Designing for Tussle in (Encrypted) DNS

Austin Hounsel
Princeton University
ahounsel@cs.princeton.edu

Paul Schmitt
Princeton University
pschmitt@cs.princeton.edu

Kevin Borgolte
TU Delft
k.borgolte@tudelft.nl

Nick Feamster
University of Chicago
feamster@uchicago.edu

Abstract

Recent concerns over the privacy implications of the Domain Name System (DNS) have led to encrypting DNS queries and responses through protocols like DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT). While the trend towards encryption is a positive development, the resulting centralization of the DNS has fomented tussles involving ISPs, browser and device vendors, content delivery networks, and users. Current deployment trends, should they continue, result in dynamics that will increase barriers to entry to competition and threaten consumer protection. This development makes it necessary for us to re-think name resolution to allow tussles to play out within the context of the design of the Internet architecture. This paper articulates several current DNS tussles and offers principles to guide system design and implementation such that all stakeholders in the space could participate. We then explore how a refactored client DNS mechanism can open up new possibilities for de-centralized name resolution, preserving the benefits of encrypted DNS while satisfying other architectural desiderata, including performance, resilience, and privacy.

1 Introduction

DNS has long been insecure and vulnerable to eavesdropping, but that reality is changing, as protocols for encrypted DNS have recently been proposed and deployed, notably DNS-over-TLS (DoT) and DNS-over-HTTPS (DoH). DoH, in particular, has seen rapid adoption, as browser vendors have begun to move name resolution functionality into applications themselves, whereas in the past it was typically done at the OS level. DoH deployment depends on coordination between the stub resolver at the client side (e.g., in the browser) and the operator of the trusted recursive resolver (TRR). In some cases, that coordination is straightforward because the same organization operates both the browser and the TRR (e.g., Google offers both a browser and a public DNS service). In other cases, two organizations coordinate—as is the case where Mozilla has collaborated with Cloudflare to deploy an encrypted DNS service in Firefox, with Cloudflare serving as the primary trusted resolver.

DNS encryption is unquestionably a positive trend, but it is accompanied by a troubling consequence: the *increased centralization of a critical part of the Internet infrastructure*. This organizational centralization makes the DNS infrastructure itself less resilient to disruption from misconfiguration, attack, and outright manipulation. These threats are more than existential: An attack on DNS infrastructure in 2016 rendered many websites unreachable [35]. DNS queries are ripe for widespread manipulation, resulting in information control and censorship. DNS misconfiguration is also commonplace [4]. Centralization also has potentially adverse effects on competition, introducing new barriers to entry as organizations who operate recursive resolvers have access to DNS queries that can be used for a competitive advantage in other market sectors, from content delivery to advertising [3]. The increased centralization of DNS data into a handful of entities has also raised privacy concerns about tracking users' browsing patterns through their queries.

Every stakeholder in the Internet ecosystem has an interest in (encrypted) DNS queries: it is the quintessential *tussle space* [10]. Internet service providers (ISPs) and enterprise networks rely on observation of DNS queries to detect everything from compromised devices to botnets, or to offer services such as parental controls. Content delivery networks sometimes rely on DNS options to efficiently map clients to the nearest CDN replica. Users are concerned about ensuring that their Internet use (e.g., browsing patterns, device usage) remain private from eavesdroppers, which could include ISPs, enterprise networks, and CDNs.

These high stakes have resulted in heated arguments and political battles from mailing lists to standardization bodies, such as the Internet Engineering Task Force (IETF), whereby each of these stakeholders seeks to retain control over the DNS. Faced with the prospect of losing visibility into DNS queries, some ISPs have partnered with Mozilla to become trusted recursive resolvers. Users who have privacy concerns over their ISPs eavesdropping on their DNS traffic might be concerned by this development. Similarly, users who are more concerned with advertisers seeing their browsing patterns would be rightfully concerned that the IoT devices that they purchase from these same companies default to sending DNS queries to the TRR of the same company (e.g., many of Google's IoT products are hard-wired to use Google Public DNS as a TRR [39]). Left behind in all of these power struggles is the user, who is often left with no choice but to rely on a centralized DNS operator, either by coercion or through opt-out tactics, default configurations, and obscure menu configuration options.

In spite of the various tussles playing out between stakeholders, current designs for encrypted DNS resolution, which primarily couple DNS resolution to the browser or device, violate many principles for resolving tussles. In particular, Clark *et al.* outline several design principles for resolving tussles: (1) design for choice; (2) don't assume the answer; (3) make the consequence of choice visible; and (4) modularize along tussle boundaries [10].

The current designs for encrypted DNS violate all four of Clark's principles. First, current browser- and mobile-based approaches to encrypted DNS either default to sending all DNS queries

to a *single* centralized TRR operator (e.g., Cloudflare, Google), without giving the user an option for others. In some cases, even attempts to change the TRR will render the device inoperable. Second, existing configurations, which default to resolving encrypted DNS queries at a single TRR, assume that this is the correct "answer", precluding other designs that might offer users improved performance, privacy, or some tradeoff between these types of concerns. Third, the notion that one can choose a TRR is largely invisible to users; the consequences of these choices are even more obscure. Fourth, there is a clear modular boundary between application functionality (e.g., web browsing, the functions of some IoT device), and resolving DNS names; current architectures do not respect this boundary.

While the tussle over encrypted DNS continues in a tailspin, trends towards increasing centralization continue. In 2017, more than 40% of DNS traffic from Tor was resolved via Google Public DNS. More recent statistics have shown that more than 30% of DNS queries to ccTLDs come from five large cloud providers, two of whom offer their own centralized DNS service [32]. A small number of organizations who operate DNS resolvers are gaining increased market share. This centralization is occurring in spite of the fact that *anyone* can operate a trusted recursive resolver, and in fact hundreds of organizations do just that. The trends towards centralization of this critical part of the Internet are driven not by technical limitations, but rather by ongoing trends of Internet consolidation, coupled with the bundling of critical functionality like name resolution into applications themselves.

This paper makes the following contributions:

- We articulate the tussle spaces that exist in DNS, enumerate the stakeholders and their respective incentives, and explain why the current DNS infrastructure does not make it possible to resolve these tussles.
- We apply the principles of designing for tussles to explicitly guide design choices that would allow tussles to take place in DNS.
- We present the design of a custom DNS stub resolver and argue that this stub resolver is the appropriate place for tussles to take place.
- We explore how allowing tussles to be resolved in a stub resolver where every stakeholder can exert some level of control could allow for new possibilities in the evolution of DNS, exploring in particular how a custom stub resolver could offer the benefits of encrypted DNS while ensuring that the DNS itself remains decentralized.

The contribution of this paper is not to presume a solution or pick a winner in the encrypted DNS tussle. Rather, it is to design the infrastructure to allow tussles to play out such that all stakeholders can have a voice. We also empirically demonstrate that DNS functionality can be de-centralized while maintaining acceptable performance, offering interesting tradeoffs between privacy, performance, usability, and other concerns. There are *many* possible outcomes for the future of the DNS infrastructure. But, the DNS infrastructure must allow these tussles to take place without bias, so that the technology can evolve towards the best outcome for all.

2 How DNS Became a Tussle Space

In this section, we provide background on the development of encrypted DNS protocols and explain how these protocols have led to a centralization of DNS, which has created several tussles.

2.1 Encrypted DNS Protocols

DNS queries and responses have historically been unencrypted, which has garnered increasing concern in recent years, given research that has demonstrated that DNS traffic can be used to discover private information about users, ranging from the websites and webpages that they visit to the "smart" devices that they use (and how they operate them). Significant concern has been raised, for example, by the Federal Communications Commission (FCC) over an Internet service provider's ability to observe their subscribers' DNS traffic. A common threat scenario may be that of a user who associates to a wireless network for convenience (*e.g.*, in a coffee shop, airport, or any public space) and subsequently sends DNS queries to an associated DNS resolver, in cleartext [5].

Increasing concern over the privacy risks of DNS has led to the development and deployment of protocols that encrypt DNS queries and responses. Two prominent developments are DNS-over-TLS (DoT) [24] and DNS-over-HTTPS (DoH) [21]. Many public DNS providers, including Google, Cloudflare, Quad9, and others now provide services for both DoT and DoH. The challenge, naturally, concerns configuring clients to adopt these protocols. Recent proposals from Mozilla and Google involve sending DoH queries directly from the browser to a recursive resolver (sometimes simply referred to as a "resolver") as configured in the browser (perhaps even by default, although as of this writing the default settings have not yet changed). Similarly, the Android OS makes it possible to route all DNS queries via DoT to a Google-operated resolver [29].

2.2 The Centralization of DNS

Although the encryption of DNS is largely a positive development, an emergent side effect is the centralization of the protocol. Specifically, clients that are configured to use DoT or DoH operate using *centralized* architectures, whereby the client sends all DoT or DoH queries to a single recursive resolver. Conventional DNS would initially appear to share the same characteristics: a client typically sends all queries to its local resolver, typically one that is configured via DHCP. Yet, encrypted DNS creates the potential for additional centralization for several reasons. First, all clients may have the tendency to use the same encrypted DNS resolver (*e.g.*, Cloudflare, Google), *regardless* of their network attachment point. This scenario contrasts with the status quo, where different clients use different DNS resolvers corresponding to their local ISP. Second, because the selection of the resolver is bundled with the browser or device, users may have no easy or viable option to change this configuration.

These centralization trends have occurred rapidly, over a relatively short timespan. In June 2018, Mozilla announced a partnership with Cloudflare to deploy DoH to Firefox desktop users in the United States [31]. Mozilla implements DoH in the browser and Cloudflare operates a recursive resolver that supports DoH. Initially, this option was enabled in Firefox Nightly builds; over the course of 18 months, Mozilla transitioned to sending all DNS queries to

Cloudflare via DoH by default. In February 2020, Mozilla enabled DoH by default for all Firefox users in the United States—in many cases doing so with minimal information about the transition [11].

This partnership between Mozilla and Cloudflare was not seen by all as a net win for privacy. The Internet standards community expressed distrust and frustration over how DoH was rolled out to Firefox users. They argued that by only selecting a single DoH provider in their initial rollout, Mozilla was centralizing DNS data—which was previously distributed across numerous recursive resolvers—into a single organization. Others in the community believe that Mozilla should have waited until the standards community developed support for local resolvers to express their support for DoH, enabling ISPs and enterprises to remain the default DNS providers for their users. Critics have expressed their dismay on DoH-related mailing lists:

> i realize that the political tacticians who designed DoH are searching for a world in which network operators have no control plane choices. i think they're proceeding from the mistaken belief that all control is evil, and that all network operators are equally deserving of disintermediation. and other mistaken beliefs as well, which i won't enumerate.
>
> whether the situation turns out to be temporary or not is important to your final argument. probably you shouldn't go there so soon. spammers also believe that network operators should not be able to control their own networks, and malware authors, and botnet creators, and IoT innovators, and surveillance capitalists. none of those matters seem like they are, or will ever be, settled. so, none are "temporary".
>
> my network, my rules. anyone who acts otherwise will be treated by me as an adversary, even folks like mozilla who have been fellow travelers for decades now, if they continue to pursue unblockable endpoint technology.
>
> Paul Vixie <paul@redbarn.org>

In short, although encrypted DNS protocols do provide privacy benefits, the implementation of their deployment over the past three years has resulted in grave concern from the community about the potential for further centralization and consolidation of Internet infrastructure. Such centralization has had far-reaching effects: users can gain privacy from their ISPs but perhaps simply transfer their private information to another public resolver. Meanwhile, ISPs, who rely on the DNS for Internet security and network management purposes, have scrambled to deploy their own Trusted Recursive Resolvers [33]. Content delivery networks have also expressed concern about how the encryption of DNS queries below the recursive may affect their ability to localize clients. More recently, operators of the DNS root servers such as Verisign have expressed concerns about how these developments may affect their ability to localize clients [27].

All of these rapid developments in encrypted DNS have resulted in a plethora of tussle spaces—ones that are playing out in flame wars on mailing lists and heated discussions among advocacy groups and standards bodies. The current encrypted DNS architectures, which couple encrypted DNS with the browser or the device, make it difficult for these tussles to play out in the design of the architecture itself. The next section explores the resulting tussles in more detail.

3 Tussle Spaces in (Encrypted) DNS

Encrypted DNS has provoked tussles between various participants in the Internet ecosystem: users, who want good performance (and privacy); Internet service providers (ISPs), who often leverage DNS as a control plane for security and performance reasons, and may wish to infer customer behavior; content delivery networks, who use user DNS information to localize users to nearby content replicas; and public resolvers, who may use DNS queries to learn about user behavior. In this section, we explore the tussles concerning (encrypted) DNS in more detail, explain how the current architecture fails to resolve (and in some cases, exacerbates) them, and suggest various approaches for designing DNS architecture to account for these tussles.

3.1 Users vs. Public Resolvers & ISPs

Who Are The Stakeholders? The operators of many large commercial public resolvers—notably Google and Cloudflare—have an incentive to observe (and resolve) users' DNS queries, due to benefits garnered for content delivery, advertising, or both. Likewise, ISPs have an interest in observing and resolving DNS queries because they use the information for security purposes, as well as to implement various products like parental controls. Users not only desire a DNS resolver that performs well, but also one that doesn't track them or manipulates responses. The rest of this section explores this tussle.

What Is The Tussle? Public resolvers have seen increased usage in recent years, spurred on by their early adoption of encrypted DNS. Some users may not trust public resolvers with their DNS data, however. When Mozilla announced that Cloudflare would be the initial default recursive resolver for their DoH rollout to Firefox desktop users in the United States, users expressed concern over Cloudflare's potential motivations for participating in the program [25]. Some argued that the program would centralize DNS data at Cloudflare, raising concerns about robustness, competition, and privacy. On the other hand, researchers have found that Google's public resolver already sees a comparatively large portion of DNS queries over any other DNS resolver [32]. Thus, some may argue that the DNS is *already* centralized into a handful of resolvers. At the same time, as ISPs begin to deploy encrypted DNS, other users may not want their ISP to see all of their DNS queries, either.

Why Hasn't The Tussle Resolved? Most stub resolvers and applications that support encrypted DNS send all of a users' queries to a single recursive resolver that is configured on the operating system or by an application that embeds a stub resolver. Some users have expressed concern over coupling their queries with a single resolver because it may enable an operator to track their browsing histories over time. Although most users tend to not know about the DNS and its implications, we argue that for centralization to be truly addressed–whether at an ISP or a public recursive resolver–users need some way of meaningfully expressing preferences regarding privacy and availability. We recognize that many resolvers send queries to additional resolvers as a fallback mechanism, but this is not the proactive distribution of queries that we envision. Furthermore, this behavior is not for trust or privacy purposes, but rather for reliability.

3.2 Public Resolvers vs. Each Other

Who Are The Stakeholders? Many organizations operate publicly available recursive resolvers, which users can use to perform DNS resolution on their behalf. Although there are hundreds of public DoH recursive resolvers, only a few TRRs—notably, Google, Cloudflare, and Quad9—are optimized for large-scale deployment. In many cases, these organizations have an incentive to access more DNS queries and serve more users for various commercial purposes. First, they may use the brand recognition of these resolvers to lead customers to other products that they offer. Second, they may analyze the contents of DNS queries to determine which websites users are visiting. This data could then be used to track users, to deliver targeted advertising, or sold to third parties. Third, certain public resolvers are hosted by content delivery networks (CDNs), who may use DNS queries to localize users and direct them to nearby content caches.

What Is The Tussle? Commercial organizations that operate public recursive resolver compete with one another and hence compete for users' DNS queries. Cloudflare and Google both operate content delivery networks; Comcast, who recently launched a TRR service, also delivers its own content (Comcast is owned by NBC Universal). To improve the performance of the delivery of content on their own CDNs, Cloudflare and Google may use DNS data to direct users to their local caches. ISPs who operate trusted recursive resolvers may offer additional services, such as malware protection or parental controls, that depend on seeing DNS traffic. Large operators of public resolvers are thus tussling over which party gets to see users' DNS queries and enjoy the benefits of improved performance to their own services.

Why Hasn't The Tussle Resolved? Despite the fact that there are hundreds of DoH resolvers deployed [12], only a few DoH resolvers are currently available as options in Firefox. Mozilla's trusted recursive resolver (TRR) program requires DoH resolvers to agree to specific requirements, to be listed in an (obscure) dropdown menu in the browser configuration. Approved TRRs must not retain DNS logs for more than 24 hours, and these logs cannot be sold or shared with other parties [16]. Although this program may be ultimately be beneficial to users' privacy, it affects competition between resolvers and effectively makes the browser vendor the gatekeeper for which organizations can participate in the DNS tussle space.

This arrangement favors some incumbents, while balkanizing the tussle space along various market segments. Notably absent from Mozilla's TRR program is Google's public DoH resolver (one can intuit from this fact that Google may be logging users' DNS queries). Thus, if users wish to use Google's DoH resolver in Firefox, they must manually configure it, even if Google is already configured as their unencrypted DNS resolver at the operating system level. No matter, of course: Google operates Chrome, the dominant Web browser, and has less of a need to join Mozilla's program in the first place. Such a dynamic further cements the centralization of companies that control large portions of the market. As of May 2020, Google Chrome assigns users to a DoH resolver that is consistent with the user's preferences and settings [2]. If Chrome finds that the DNS resolver configured on a user's operating system is included in a pre-defined table of DoH resolvers, then Chrome will assign that user to the corresponding DoH resolver. Such behavior can change at any time, of course—often with obscure or no notification to the user—as it did with Mozilla's rollout of a default DoH configuration last year.

3.3 Public Recursive Resolvers vs. ISPs

Who Are The Stakeholders? Despite the increasing popularity of public resolvers, users primarily use the resolvers that are advertised by their ISPs. ISPs may operate these resolvers for various reasons. For example, they might use these resolvers to provide parental controls for their customers or automatically block DNS-based malware. They may also operate resolvers to improve network performance for their customers by running a local DNS cache. Finally, they may use this data for commercial purposes, for example to re-direct customers to landing pages with advertisements when they mistype a URL [19].

What Is The Tussle? Given the historical role of ISPs in providing DNS resolution, ISPs are resistant to the increasing popularity of public resolvers. The debate between ISPs and public resolvers over who should perform DNS resolution has recently played out in web browsers. When Mozilla announced that they were enabling DoH by default in 2020, Cloudflare and NextDNS were the default recursive resolvers [34] Some have argued that by choosing a few resolvers as the default providers, Mozilla is enabling DNS data to be increasingly centralized into a handful of organizations, rather than the historical multitude of ISPs. Further, as of January 2021, Google Chrome only performs auto-upgrade for a few major ISP DoH resolvers, *e.g.*, Comcast and Spectrum [9]. ISPs contend that current deployment models of DoH favor large public resolvers over ISPs. In 2020, Comcast announced a collaboration with Mozilla to deploy their recursive resolver to Firefox users that use Comcast's networks [33]. Comcast would become a member of Mozilla's trusted recursive resolver program, agreeing to audits of their DNS data and various privacy requirements; however, as of January 2021 (Firefox 85.0), Comcast is still not listed as a TRR option.

Why Hasn't The Tussle Resolved? ISPs sometimes argue that Mozilla and Google should not have offered DoH resolution before a local DoH resolver discovery mechanism was standardized. As of January 2021, several mechanisms have been proposed within the Adaptive DNS Discovery IETF working group to support local DoH resolver discovery, but they have not yet been standardized [26]. As a result, customization remains cumbersome and obscure: users can only use an ISP's DoH resolver is if they know the information for the resolver in advance and configure the resolver manually. It remains unclear whether users that use ISPs that are a part of Mozilla's trusted recursive resolver program will exclusively be assigned to use the ISP-provided resolvers, or share queries across third party public recursive resolvers as well. It is also unclear which ISP resolver Firefox will use when users switch between networks whose DNS resolvers are all members of the trusted recursive resolver program.

4 Designing for Tussle

In this section, we outline Clark et al.'s principles for designing for tussle [10] and explain their implications for the DNS.

4.1 Design for Choice

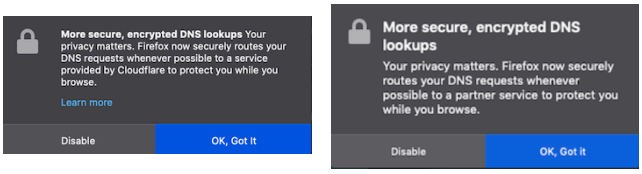
Definition. To address tussles over Internet protocols, Clark et al. argue that protocols should be designed such that users can choose who they communicate and exchange data with:

> Network protocols are designed so that different parties on the network can communicate with each other, consumers can make use of the resources of providers, and providers can interconnect with each other to provide service. It is important that protocols be designed in such a way that all the parties to an interaction have the ability to express preference about which other parties they interact with. Protocols must permit all the parties to express choice [10].

Implications for DNS. When addressing DNS tussles (*e.g.*, users not placing trust in a particular resolver), we not only interpret "design for choice" to mean that users should always be able to decide how DNS resolution is performed, but also that DNS configuration options should be presented in ways that are meaningful to users. Users should then be able to apply these configurations across any application on a device. Applications (or devices acting in the interests of their designers) should not be able to choose where DNS resolution is performed that violate users' wishes or in ways that users cannot override.

Many scenarios in today's Internet are not designed for choice. For example, Google Chromecast has reportedly used Google's public DNS resolver by default, and has lost function if network operators force another resolver to be used [39]. Users can technically set up a resolver on the local network to respond to queries destined for Google's resolver, but this requires significant expertise and is not a feasible choice for most users. The lack of choice in DNS resolution has significant implications for privacy and reliability. Users may not trust Google (or other resolvers) with their DNS queries, raising concerns about how users' data is handled against their wishes. Furthermore, if Google's own DNS resolver fails and clients have no feasible way of choosing other resolvers, then Google's devices that are hard-coded to use their resolver may lose function.

Users may be unaware of the DNS and the implications of choosing between DNS resolvers in the first place. However, users that desire to make these choices should at least be able to. This may be difficult to achieve on embedded devices or on the "Internet-of-Things" (IoT), but without visibility of what choices are made and the ability for users to make choices for themselves, the tussle between users and resolvers over trust will continue. Moreover, these choices should be presented in ways that are meaningful to users. For example, if users are presented with a dialog box by an application or stub resolver asking if they wish to enable encrypted DNS, the dialog box should attempt to inform users what the privacy implications are for enabling encrypted DNS with a particular resolver.



(a) *February 2020* **(b)** *September 2020*

Figure 1: *Firefox pop-up menus for opting out of encrypted DNS have changed over time. Initially, Cloudflare was explicitly mentioned. Over time, the consequences of this opt-out choice became more opaque to users*

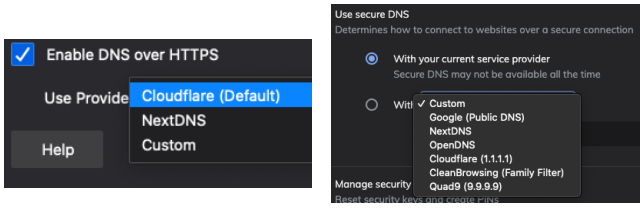
4.2 Don't Assume the Answer

Definition. Clark et al. also argue that when it comes to Internet architectures and protocols, "there is no such thing as value-neutral design":

> What choices designers include or exclude, what interfaces are defined or not, what protocols are open or proprietary, can have a profound influence on the shape of the Internet, the motivations of the players, and the potential for distortion of the architecture. Don't assume that you design the answer. You are designing a playing field, not the outcome [10].

Implications for DNS. DNS clients can select resolvers in a variety of ways, rather than designing a "playing field" As previously discussed, various applications and devices have made default choices for recursive resolvers that have been met with criticism. When Google configured its Chromecast devices to use their own recursive resolver, rather than the resolvers advertised by local networks, they assumed the answer, designing an outcome that not only restricted user choice, but also led to increased concerns over privacy and reliability. Similarly, when Mozilla launched its trusted recursive resolver program for DoH in the United States with Cloudflare as the default recursive resolver, users expressed concern over DNS centralization at an entity separate from their default DNS providers. Users were presented with a choice via a one-time, obscure pop-up menu, to opt out of using DoH (and Cloudflare's resolver) by default, DNS (and in turn, DoH), as shown in Figure 1; over the course of 2020 this option became more obscure, and in Firefox 85.0, the option was enabled by default with no opt-out. Disabling or changing these default settings is possible, but the options are buried multiple levels deep in configuration menus that users likely aren't even aware of. Figure 2 shows the configuration menus in Firefox and Brave, as well as the default configurations for each.

DNS clients can instead design for a playing field among resolvers instead of a particular outcome that may surprise users. If applications ultimately decide to choose resolvers that were not advertised by the local network, in addition to offering choice to users, in some cases it may also be appropriate to assign different default resolvers to different populations of users by default. For example, Mozilla's deployment of the trusted recursive resolver program with Cloudflare as the default provider for Europe was problematic, since Cloudflare is a United States-based company, and European users can have different expectations—and regulations—governing data protection practices. Although Mozilla has yet to roll out its trusted recursive resolver program to European users,



(a) *Firefox* **(b)** *Brave*

Figure 2: *Different browsers have different default settings for DNS resolution; most users are likely unaware of these options and wouldn't even understand them even if they could find them at all.*

there is nonetheless significant concern that a single resolver (or small set of resolvers) that was designed for United States users will be configured for European users. Other regions may also have unforeseen circumstances where assuming a particular answer may defy norms or catch users by surprise.

Applications and devices should not assume the answer, and they should especially not assume that one answer suffices for all users. They should enable users to express preferences about resolver selection when multiple choices are available. For example, when a local resolver supports DoH and the application/device also knows of multiple public resolvers that also support DoH, clients may want the local resolver to take precedence. Other clients may want public resolvers to take precedence, only using the local resolver when the configured public resolvers are unavailable. Some clients may even wish to split their queries across multiple recursive resolvers, preventing any single resolver from having access to all of their queries. In short, clients should be able to express preferences about how to select between multiple recursive resolvers—making fine-tuned decisions about how their queries are resolved—rather than only sending queries to a single resolver by default.

4.3 Modularize Along Tussle Boundaries

Definition. Finally, Clark et al. argue for breaking the Internet architecture into modular components when possible, such that tussles over one part of the architecture do not spill over into other parts:

> Functions that are within a tussle space should be logically separated from functions outside of that space, even if there is no compelling technical reason to do so. Doing this allows a tussle to be played out with minimal distortion of other aspects of the system's function [10].

Implications for DNS. Traditionally, operating systems have performed DNS resolution on behalf of applications running on a device. Today, applications themselves can make their own decisions about which DNS transports and recursive resolvers should be used, independent of what the operating system learns from the network or configures on its own. Entire devices have also reportedly ignored the DNS configurations learned from the network [39]. Such behavior prevents all stakeholders the potential to access DNS data. Users that wish to change how DNS resolution is performed on their devices may now need to make changes in multiple locations *e.g.*, in both the web browser and the operating system's stub resolver.

If each application on a device handles recursive resolver selection in a different way, then DNS tussles over privacy, trust, centralization, and reliability will continue. Applications may not trust local resolvers and may wish to always use encrypted DNS, but users wish to override the decisions made by the applications, then they will need to make changes in multiple locations instead of one and will continue to have little visibility into the decisions that each application makes.

5 A Place to Resolve Tussles

Decisions about DNS resolution should occur at a single place: a separate stub resolver that performs resolution for all applications and devices for which a user or users have a common set of preferences. Such modularization provides applications, ISPs, and users one place for tussles to resolve. We envision such an exchange taking place as follows: (1) a separate stub resolver could discover a collection of resolvers that support DoH, along with various characteristics of those resolvers (*e.g.*, geographic location); (2) a user might specify specific requirements or preferences about preferred resolvers or goals (*e.g.*, a preference to avoid a specific location, geography or ISP; or a preference of privacy over performance or vice versa); or, alternatively, an explicit selection; (3) the stub selects DoH resolvers by matching availability with user preference. This design bears some resemblance to the behavior of an operating system stub resolver, but bears the additional characteristics of configurability, and the option to place the stub resolver at a point in the network that is independent of any device but common to a set of user (or users) who share common preferences., such as a home network router.

We present a preliminary design of a stub resolver that adheres to these principles, with a specific use case of decentralizing DNS query resolution as an example goal. The rest of this section outlines this design and explains how the design satisfies the design principles we outlined in the previous section.

5.1 A Stub Resolver for DNS Tussles

A separate stub resolver that can resolve queries for *all* users and devices who share preferences about performance, privacy, security, and other considerations is an appropriate location to resolve the tussles in DNS. Such an architecture affords the following benefits, in adherence to Clark em et al.'s principles:

Users can choose how DNS queries are resolved. In contrast to the status quo, where users have lost control over DNS resolution—either because browsers have co-opted encrypted DNS resolution and made the options obscure, or because Internet-connected devices select their own DNS resolution mechanism without exposing these decisions to users—resolving *all* DNS queries in a separate stub resolver that a user can configure and customize puts the user back in control. (A separate, important question concerns whether users understand the consequences of these choices, and how to make those choices visible. This problem is an excellent avenue for future study in its own right.)

All stakeholders have access to this point of control. In contrast to the current architecture, where browser and device vendors hold control over which entities can be trusted recursive resolvers

and which resolvers are selected by default, a separate stub resolver can potentially be controlled by *any* of the stakeholders. Naturally, we expect that there will be push and pull, and even cooperation (or collusion) among these entities. But, modularizing the DNS resolution process in this fashion will make those actions *visible*: An anti-competitive maneuver such as restricting an API to configure the stub, or collusion between content providers and browser vendors would be plainly apparent in such an architecture—and likely reversible, if not through alternate implementations, then via regulatory mechanisms.

The stub resolver is configurable and does not presume an outcome. As described later in this section, we envision a stub resolver that affords many possible configuration options, and Section 6 explores one such customization option that involves decentralizing DNS queries across multiple trusted recursive resolvers, using one of many possible distribution strategies. Such a level of configurability required only modest modifications to existing DNS stub resolvers, as we describe in Section 5.2.

The architecture modularizes across tussle boundaries. The tussles that we have outlined can all take place within the constructs of the separate stub resolver—moving disputes from flame wars on mailing lists to a technical sphere where stakeholders can develop competing implementations, configurations, and resolution strategies.

5.2 Prototype Implementation

To prototype the stub resolver described in Section 5.1, we forked the open-source `dnscrypt-proxy` stub resolver [13]. The `dnscrypt-proxy` stub resolver supports DoT, DoH, and DNSCrypt, acting as a DNS proxy that can run on users' machines and routers. We extended `dnscrypt-proxy` to support new strategies and policies for resolving DNS queries across sets of different TRRs (Section 6 details a variety of possible policies for decentralizing DNS queries, for example). We extended the `getOne()` function within `serversInfo.go`, which is responsible for indexing into the array of upstream resolvers for each query based on which load-balancing strategy is specified in the configuration file [14]. We implemented new values for the `lb_strategy` option in the configuration file to enable the selection of the round-robin and hash-based query distribution strategies. For the random strategy, we use the code provided by the stub resolver. We also modified `dnscrypt-proxy` to perform the initial sorting of resolvers by name instead of latency, which allows for implementation of a wider variety of policies for selecting resolvers, as opposed to simply load balancing based on latency.

We modified `dnscrypt-proxy` to allow the client to specify various options in the configuration file concerning strategies for distributing queries over multiple recursive resolvers. The configuration file allows the user to customize choice of DNS protocol (*e.g.*, DNS, DoT, DoH), the sets of resolvers to use, and how to distribute queries across multiple resolvers, if the user specifies multiple resolvers. Our particular modifications concern the case study that we detail in Section 6. The most important aspect of the prototype implementation is that it allows for such modification in the first place. The modifications we present in Section 6 are only one example of many possible avenues.

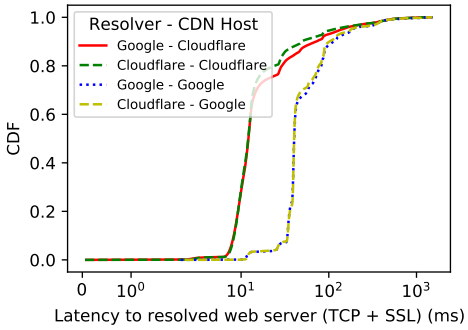


Figure 3: *TCP and SSL setup times to CDN servers operated by Cloudflare and Google. Each line shows setup times when a particular DNS resolver is used for either Cloudflare or Google hosted content.*

6 Case Study: Re-Decentralizing the DNS

In this section, we evaluate how the architecture we presented can facilitate various tussles concerning the centralization of encrypted DNS. We explore how the architecture can enable decentralization of queries, evaluate the effect of distributing queries on the accuracy CDN localization, and use a public dataset of DNS queries captured for a fiber-to-the-home (FTTH) network of approximately 100 homes in Cleveland, OH to evaluate how different strategies might have implications for performance and privacy [1].

6.1 Query Distribution Strategies

To examine the potential benefits and drawbacks of decentralizing user DNS traffic across multiple resolvers, we define several query distribution strategies.

Hash-Based Distribution In a hash-based distribution, second-level domain names (SLDs) are hashed in order to select a resolver, meaning that queries for the same SLD will always be sent to the same resolver. For example, all queries issued by a client for `google.com` and `images.google.com` will be sent to the same resolver. Furthermore, if the same client later queries `images.google.com`, the query will be forwarded to the same resolver as before.

Random Distribution In the random distribution strategy, queries are randomly sent to a set of defined resolvers R, resulting in each resolver handling $\frac{1}{R}$ of the client's queries.

Round-Robin Distribution Using this strategy, queries are sequentially striped across a set of resolvers R. The round-robin strategy results in each resolver would be assigned $\frac{1}{R}$ of the client's queries.

6.2 How Does Query Distribution Affect CDN Localization?

To begin our analysis of query distribution, we study CDN localization. In particular, we seek to understand whether distributing queries across multiple recursive resolvers could negatively affect CDN localization, causing users to connect to CDN servers that are suboptimal. To perform this analysis, we fetch each HTTP request from the *requests_desktop* table in the HTTP Archive for October

2020 [23]. We also use information provided by the HTTP Archive to determine which CDN each domain name hosts its content on, if applicable. For comparison purposes, we study content that is hosted by Cloudflare or Google, as these providers also both operate DNS resolvers. We resolve the domain names twice, once using Cloudflare's DNS and once using Google's, for each request that was hosted by either Cloudflare or Google's CDN networks, and measure the latency for TCP and SSL connection setup to the resolved IPs from a 500 Mbps residential fiber connection. This helps us understand whether content hosted by one CDN performs better when resolved using a particular DNS resolver.

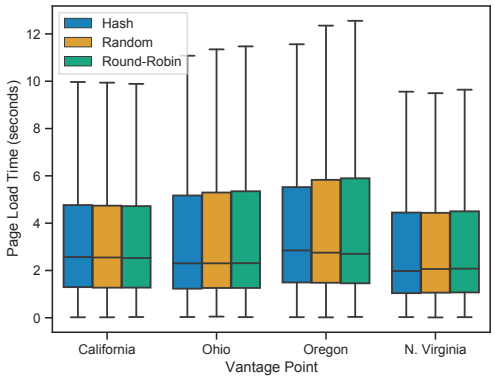
Figure 3 on the preceding page shows the results of our analysis. Each line in the figure shows a cumulative distribution function (CDF) for combined TCP and TLS setup times for a given resolver and CDN. For example, the line that corresponds to "Google - Cloudflare" in the legend shows combined TCP and TLS setup times when Google's resolver is used to resolve hosted on Cloudflare's CDN. We find that concerns over whether distributing queries over multiple resolvers will affect CDN localization are not significant in our experiment. When either Google's resolver or Cloudflare's resolver is used to resolver Google-hosted content, TCP and TLS setup times follow the same distribution. The distributions for each resolver are slightly different when Cloudflare content is resolved, but for the most part, the distributions are very similar. We note that Google and Cloudflare host two of the most popular public recursive resolvers, but we expect similar results with any resolver that is widely distributed.

6.3 What is the Effect of Query Distribution on Page Load Times?

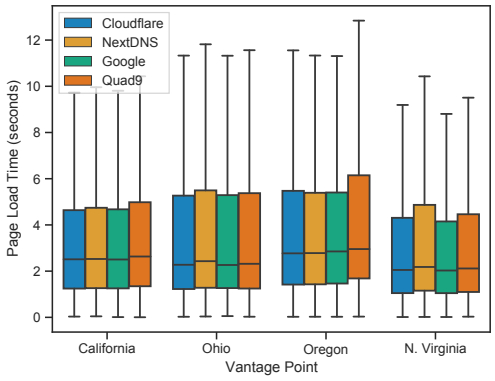
To measure the performance impact of distributing DNS queries across multiple resolvers, we performed page loads from several vantage points for 20 days. We created an Amazon EC2 instance in four vantage points–Ohio, North Virginia, California, and Oregon–that each ran Debian Linux. To perform our measurements, we extended a Docker image created by Hounsel et al. that performs page loads using a headless version of Mozilla Firefox 84.0.1 controlled by Selenium [22]. Each page load is performed within a separate Docker container. Once we launch a container, we fist run our fork of `dnscrypt-proxy` within the container with a configuration file that corresponds to the strategy that we intend to measure. We then modify `/etc/resolv.conf` to use our stub resolver, and we initiate a page load. Once the page load completes, an HTTP Archive Object (HAR) corresponding to the page load is extracted from the container, and we close the container. We read the timing for the `onLoad` event in each HAR to measure page load times. [1] With the authors' permission, we have published our open-source fork of their image.

We used the Tranco top-list to perform measurements for popular websites that users are likely to visit. The Tranco top-list takes multiple top-lists as input (such as the Alexa top-list) and aggregates their rankings over time [36]. We downloaded the Top 1,000,000 list available on December 12th, 2020 and extracted the Top 1,000 websites from the list. We then performed continuously page loads for each of these websites from each vantage point using each strategy.

For our query distribution strategies, we used the DoH resolvers provided by Cloudflare, Google, Quad9, and NextDNS. We chose these resolvers due to their popularity and their support in major browsers. For example, as of January 13th, 2021, Cloudflare and NextDNS are the two default DoH providers that are listed in Mozilla Firefox 84.0.1. Similarly, Google Chrome automatically



(a) *Query distribution strategies.*



(b) *Individual resolvers.*

Figure 4: *Page load times from each vantage point using query distribution models and individual resolvers.*

Location	Hash	Random	Round-robin
California	2.57	2.55	2.52
Ohio	2.30	2.30	2.31
Oregon	2.85	2.75	2.70
N. Virginia	1.97	2.06	2.08

Table 1: *Median page load times (in seconds) from each vantage point using each query distribution strategy.*

[1] We disabled the DNS cache for dnscrypt-proxy. Firefox maintains its own in-memory DNS cache, but because each page load was performed within a separate Docker container and because Firefox clears its cache upon exit each page load uses clean DNS and HTTP caches .

Location	Cloudflare	NextDNS	Google	Quad9
California	2.51	2.53	2.50	2.63
Ohio	2.28	2.43	2.27	2.32
Oregon	2.77	2.78	2.85	2.96
N. Virginia	2.05	2.18	2.03	2.12

Table 2: *Median page load times (in seconds) from each vantage point using a single resolver for all DNS queries.*

upgrades users of Cloudflare, NextDNS, Google, and Quad9's resolvers to DoH [2, 9]. In addition to performing pageloads with the query distribution strategies using these resolvers, we also measure page load times when using each of these resolvers on their own for all DNS queries.

Figure 4a on the preceding page shows page load times for each query distribution strategy, and Figure 4b on the facing page shows page load times for each resolver. Each boxplot shows data all pageloads for a given vantage point and query distribution strategy. The bottom of each box shows the 25th percentile, the horizontal line in each box shows the 50th percentile (median), and the top of each box shows the 75th percentile. Table 1 on the preceding page and Table 2 show median page load times for each strategy and resolver.

Our performance measurements illuminate several trends. First, we see that for most vantage points, each strategy performs similarly in terms of median page load times. The largest gap in performance for these strategies was in Oregon between the hash strategy and round-robin strategy, with the hash model performing 185 ms slower. However, the largest difference between two strategies was lower in other vantage points, with 50 ms in California, 10 ms in Ohio, and 110 ms in N. Virginia. Second, we see that page load times are similar with each resolver, although Quad9 does show comparatively high page load times in Oregon. Finally we see that users are able to obtain comparable performance when using our query distribution strategies to using a single, popular cloud DNS resolver. The difference in median page load times between the slowest query distribution strategy and the fastest cloud resolver was 70 ms in California, and 40 ms in Ohio. Furthermore, the difference between the fastest strategy and the fastest cloud resolver was 20 ms in California, and 30 ms in Ohio.

6.4 How Does Query Distribution Affect Domain Names Seen By Resolvers?

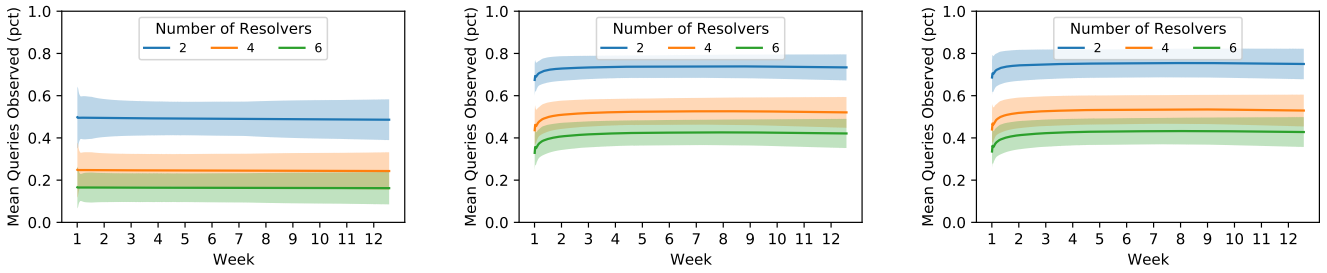
To get some initial insights into how our strategies affect the distribution of domain names seen by various resolvers, we apply our strategies to a real-world, anonymized dataset of DNS queries. The dataset consists of anonymized DNS queries for approximately 100 homes connected to a fiber-to-the-home (FTTH) network in a residential neighborhood in Cleveland, OH [1]. Each home is connected by a gateway device that proves a single public IP address for each home through NAT. This dataset consists of queries issued over a seven-day period each month during 2018 (*i.e.*, 12 weeks). We utilize this dataset to study what percentage of unique domain names queried by each home (*i.e.*, each unique source IP address) are seen by a varying number of DNS resolvers over time if our query distribution strategies were used (random, round-robin, and hash). Full details about the dataset and collection methodology can be found in Allman et al [1].

To perform this analysis, we first group the queries that each source IP address (*i.e.*, each home) issued together. We then extract the timestamp for each query, ordering each address' queries by the time in which they were issued. Finally, we simulate each of our query distribution strategies "after the fact" on each address' ordered list of queries by applying a function for each query distribution strategy to determine which resolver each query would go to if a particular strategy was used.

Figure 5 on the next page visualizes the simulations for each query distribution strategy on our dataset. Each subfigure corresponds to a particular query distribution strategy. At the top of each hour that passed in the dataset, we map each unique domain name a source IP address queried during the previous hour to a recursive resolver, based on a given strategy and number of resolvers. Then, for each recursive resolver and source IP address, we compute the percent of unique domain names seen by the resolver based on the total number of unique domain names the IP address queried in the dataset. Finally, we compute the mean and standard deviation of the distribution of percentages for each combination of an address and a resolver.

We observe a couple of trends from our simulations. First, when the hash strategy is used, each resolver sees fewer unique domain names for each address than when the other strategies are used. When four resolvers are used for the hash strategy, the strategy stabilizes with an average of ≈25% of unique domain names for each address seen by each resolver over time. On the other hand, when four resolvers are used for the random strategy and the round-robin strategy, the strategies stabilize with an average of ≈50% of queries seen by each resolver. This intuitively makes sense, as the hash strategy sends all queries for each second-level domain name to the same resolver each time, whereas the random strategy and the round-robin strategy can send successive queries for the same second-level domain name to different resolvers.

We also observe that each strategy stabilizes relatively quickly in terms of how many unique domain names are seen by each resolver for each source IP address. The dataset provided by Allman et al. represents 12 weeks of DNS queries for approximately 100 homes. However, after just one week, the random and round-robin strategies stabilize with a mean of ≈45% of unique domain names seen by four resolvers across each home. We attribute these behaviors in part to users' browsing patterns within each home. Most users likely frequently visited a small number of popular websites (e.g., `facebook.com` or `google.com`, and visited a long-tail of websites less frequently. Thus, each resolver likely saw queries for the most frequently visited websites, and the long-tail of queries for less popular websites were not distributed often among multiple resolvers. The hash strategy also shows an almost constant percentage of unique domain names that are seen by each resolver over time, due to its deterministic nature in sending queries for particular domain names to the same resolvers.



(a) *Hash Model* **(b)** *Random Model* **(c)** *Round-robin Model*

Figure 5: *Simulations of the number of unique domain names seen by a varying number of resolvers for each query distribution model, based on an anonymized DNS dataset of approximately 100 homes in a single fiber-to-the-home (FTTH) network in Cleveland, OH.*

7 Related Work

Internet Tussle Spaces. Clark et al. first introduced tussle spaces in 2002 to describe parts of the Internet ecosystem where stakeholders have opposing interests and vie to favor those interests [10]; the initial paper explores various tussle spaces involving provider lock-in on IP addressing, competition in residential broadeand access, and choice in route selection, as well as tussles around trust and denial of service. Today, the tussle around encrypted DNS carries perhaps the highest stakes. Although Clark et al. did not foresee the current tussle space two decades ago, Walfish et al. did highlight another tussle in DNS: control over the namespace hierarchy [40]; that tussle concerned which parties controlled the use of certain Internet names and advocated for an alternative flat namespace. Others have observed that Software Defined Networking (SDN) was in fact the answer to a tussle in Internet routing between operators and vendors, largely resulting from vertical integration of hardware and software and the bundling of programmable function with hardware routers in ways that restricted choice and control [17]. We view the current trends in encrypted DNS, where browser and device vendors are vertically integrating DNS functions with applications, as a somewhat analogous situation.

DNS Security and Privacy. DNS is experiencing widespread revolution as it is extended and modified to support various security and privacy properties. DNSSEC provides integrity to DNS [15]. T-DNS[41] address security issues with DNS, such as lack of confidentiality and amplified denial-of-service attacks. T-DNS has not been widely adopted, but it served as the primary inspiration for DNS over TLS (DoT) [24]. DNS over HTTPS (DoH) [21] aims to solve the same problems as DoT, but uses HTTP as a transport protocol. Other work investigated the adoption of secure DNS and their real-world benefits. Lu et al. [30] have found that the adoption of encrypted DNS improved, but it remains low compared to unencrypted DNS, and it currently suffers from deployment issues, like the use of invalid TLS certificates. Bushart et al. [6] and Siby et al. [38] studied the privacy benefits of DoT and DoH in a web browsing scenario. Oblivious DNS (ODNS) [37] hides the queried domain names from a user's recursor by leveraging encryption and an additional resolver that learns the queried name but cannot associate it with a user, effectively denying linkability of queried names to individual users. ODNS has been adapted by Kinnear et al. and extended to Oblivious DoH (ODoH) [28], supported by Apple and Cloudflare. Encrypted DNS protocols require encryption and stateful network connections. One of the interesting questions surrounding the architecture that we pose in this paper is whether a user might suffer significant performance penalties depending on how DNS resolution is configured (*e.g.*, decentralized). Fortunately, many possible options for distributing DNS queries yield acceptable performance, as shown by Hounsel *et al.* [7, 22].

Centralization of DNS. In this paper, we explored one possible application of modularizing along tussle boundaries, specifically the decentralization of DNS. Such an architecture could ultimately leverage much previous work, which has studied various aspects of centralization and proposed modifications to DNS that could help mitigate these trends. Foremski et al. find that the top 10% of DNS recursors serve approximately 50% of DNS traffic [18]. Moura et al. [32] also encounter centralization in their study of DNS requests to two country code top-level domains (ccTLD), with five large cloud providers being responsible for over 30% of all queries for the ccTLDs of the Netherlands and New Zealand. In their study, Moura et al. identify instances of advantages and disadvantages of centralization: the adoption of QNAME minimization by Google benefited a substantial amount of users instantaneously upon deployment, but breakage and downtime also affect users equally. Hoang et al. [20] propose and evaluate K-resolver, which distributes queries over multiple DoH recursors, so that no single resolver can build a complete profile of the user and each recursor only learns a subset of domains the user resolved. Callahan et al. [8] analyzed through passive measurements how modern DNS is being operated, and discovered previously unknown behavior of DNS recursors, such as the use of prefetching to keep cached names fresh.

8 Summary and Future Directions

Recent trends in encrypted DNS architectures and deployments over the past two years have introduced new tussles in DNS between users, ISPs, CDNs, and device and browser vendors. The architectures that are currently being deployed do not conform

to Clark et al.'s recommendations for designing for tussle, thus igniting heated disputes on mailing lists, as well as practices that threaten competition, user privacy, and the security an resilience of the Internet as a whole. In this paper, we have argued that the current situation largely results from the bundling of DNS resolution with browsers and devices, in ways that are opaque to users. We argue for a reversal of this trend, modularizing DNS resolution in a separate stub resolver that can be configured and customized by all stakeholders, thereby allowing tussles to play out.

This paper has argued for a change in direction of the encrypted DNS architecture to facilitate tussle—we explicitly do *not* advocate for a single best architecture but rather posit that with the correct underlying substrate, many alternatives are possible, and that all stakeholders should be able to implement, deploy, evaluate, and customize these alternatives. In this vein, we believe that this paper lays the groundwork for much future work in both research and industry, as we explore various alternative strategies for resolving encrypted DNS queries. This paper provides one such starting point as a proof-of-concept, but we believe that many others are possible and can now be explored.

Finally, this paper largely neglects another principle for designing in tussle spaces: *making the consequences of choice visible*. It is clear that current encrypted DNS architectures violate this principle (*e.g.*, in current browser configurations, who could possibly understand the ramifications of picking NextDNS vs. Cloudflare as a trusted recursive?) Yet, the solutions here are unclear and deserve concerted study. Often exposing too many details and options to users only serves to confuse them further. Thus, given the ability to choose, important questions lie ahead about which types of encrypted DNS configuration are most valuable to users, and how various options can best be presented to them.

References

[1] Mark Allman. 2020. Putting DNS in Context. In *Proceedings of the 2020 Internet Measurement Conference (IMC)* (Virtual Event, 2020-10), Fabián Bustamante and Nick Feamster (Eds.). Association for Computing Machinery (ACM). https://dl.acm.org/doi/10.1145/3419394.3423659

[2] Kenji Baheux. 2020. *A safer and more private browsing experience with Secure DNS.* https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html

[3] Kevin Borgolte, Tithi Chattopadhyay, Nick Feamster, Mihir Kshirsagar, Jordan Holland, Austin Hounsel, and Paul Schmitt. 2019. How DNS over HTTPS is Reshaping Privacy, Performance, and Policy in the Internet Ecosystem. In *Proceedings of the Research Conference on Communications, Information and Internet Policy* (48 ed.) (2019-09). SSRN, Washington DC, USA, 1–9. https://doi.org/10.2139/ssrn.3427563

[4] Kevin Borgolte, Tobias Fiebig, Shuang Hao, Christopher Kruegel, and Giovanni Vigna. 2018. Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)* (25 ed.) (San Diego, CA, USA, 2018-02), Patrick Traynor and Alina Oprea (Eds.). Internet Society (ISOC). https://doi.org/10.14722/ndss.2018.23327

[5] Stephane Bortzmeyer. 2015. *DNS Privacy Considerations.* RFC 7626. RFC Editor. https://www.ietf.org/rfc/rfc7626.txt (Informational).

[6] Jonas Bushart and Christian Rossow. 2019. Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS. arXiv:1907.01317

[7] Timm Böttger, 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 *Proceedings of the 2019 Internet Measurement Conference* (19 ed.) (2019-10), Anna Sperotto, Roland van Rijswijk-Deij, and Cristian Hesselman (Eds.). Association for Computing Machinery (ACM), Amsterdam, Netherlands, 15–21. https://doi.org/10.1145/3355369.3355575

[8] Thomas Callahan, Mark Allman, and Michael Rabinovich. 2013. On Modern DNS Behavior and Properties. *ACM SIGCOMM Computer Communication Review* 43, 3 (2013), 7–15.

[9] Chromium. 2021. *doh_provider_entry.cc: Chromium Code Search.* https://source.chromium.org/chromium/chromium/src/+/master:net/dns/public/doh_provider_entry.cc

[10] David D Clark, John Wroclawski, Karen R Sollins, and Robert Braden. 2002. Tussle in cyberspace: defining tomorrow's internet. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications.* 347–356.

[11] Selena Deckelmann. 2020. *Firefox continues push to bring DNS over HTTPS by default for US users .* https://blog.mozilla.org/blog/2020/02/25/firefox-continues-push-to-bring-dns-over-https-by-default-for-us-users/

[12] Frank Denis. 2021. *DNSCrypt: public-resolvers.* https://download.dnscrypt.info/resolvers-list/v3/public-resolvers.md

[13] DNSCrypt. 2021. *dnscrypt-proxy 2: A flexible DNS proxy, with support for encrypted DNS protocols.* https://github.com/DNSCrypt/dnscrypt-proxy

[14] DNSCrypt. 2021. *dnscrypt-proxy/serversInfo.go.* https://github.com/DNSCrypt/dnscrypt-proxy/blob/master/dnscrypt-proxy/serversInfo.go

[15] Donald E. Eastlake and Charles W. Kaufman. 1997. *Domain Name System Security Extensions.* RFC 2065. RFC Editor. http://www.ietf.org/rfc/rfc2065.txt (Internet Standard).

[16] Marshall Erwin. 2019. *Trusted Recursive Resolvers - Protecting Your Privacy With Policy and Technology.* https://blog.mozilla.org/netpolicy/2019/12/09/trusted-recursive-resolvers-protecting-your-privacy-with-policy-technology/

[17] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2014. The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review* 44, 2 (2014), 87–98.

[18] Pawel Foremski, Oliver Gasser, and Giovane C. M. Moura. 2019. DNS Observatory: The Big Picture of the DNS. In *Proceedings of the 19th Internet Measurement Conference (IMC)* (Amsterdam, The Netherlands, 2019-10), Phillipa Gill and Robert Beverly (Eds.). Association for Computing Machinery (ACM). https://doi.org/10.1145/3355369.3355566

[19] Chris Griffiths. 2012. *Comcast Domain Helper Shuts Down.* https://corporate.comcast.com/comcast-voices/comcast-domain-helper-shuts-down

[20] Nguyen Phong Hoang, Ivan Lin, Seyedhamed Ghavamnia, and Michalis Polychronakis. 2020. K-resolver: Towards Decentralizing Encrypted DNS Resolution. https://arxiv.org/pdf/2001.08901.pdf arXiv preprint arXiv:2001.08901.

[21] Paul Hoffman and Patrick McManus. 2018. *DNS Queries over HTTPS (DoH).* RFC 8484. RFC Editor. https://www.ietf.org/rfc/rfc8484.txt (Proposed Standard).

[22] Austin Hounsel, Kevin Borgolte, Paul Schmitt, Jordan Holland, and Nick Feamster. 2020. Comparing the Effects of DNS, DoT, and DoH on Web Performance. In *Proceedings of The Web Conference 2020.* 562–572.

[23] HTTP Archive. 2021. *HTTP Archive.* https://httparchive.org/

[24] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessel, and Paul Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS).* RFC 7858. RFC Editor. https://www.ietf.org/rfc/rfc7858.txt (Proposed Standard).

[25] Bert Hubert. 2019. *Centralised DoH is Bad for Privacy, in 2019 and Beyond.* https://blog.powerdns.com/2019/09/25/centralised-doh-is-bad-for-privacy-in-2019-and-beyond/

[26] Internet Engineering Task Force. 2021. *Adaptive DNS Discovery (add).* https://datatracker.ietf.org/wg/add/documents/

[27] Burt Kaliski. 2020. *A Balanced DNS Information Protection Strategy: Minimize at Root and TLD, Encrypt When Needed Elsewhere.* https://blog.verisign.com/security/a-balanced-dns-information-protection-strategy-minimize-at-root-and-tld-encrypt-when-needed-elsewhere/

[28] Eric Kinnear, Patrick McManus, Tommy Pauly, and Christopher A. Wood. 2021. *Oblivious DNS Over HTTPS.* Internet Draft draft-pauly-dprive-oblivious-doh-04. RFC Editor. https://tools.ietf.org/id/draft-pauly-dprive-oblivious-doh-04.txt

[29] Erik Kline and Ben Schwartz. 2018. *DNS-over-TLS Support in Android P.* https://android-developers.googleblog.com/2018/04/dns-over-tls-support-in-android-p.html

[30] Chaoyi Lu, Baojun Liu, Zhou Li, Shuang Hao, Haixin Duan, Mingming Zhang, Chunying Leng, Ying Liu, Zaifeng Zhang, and Jianpeng Wu. 2019. An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come?. In *Proceedings of the 19th Internet Measurement Conference (IMC)* (Amsterdam, The Netherlands, 2019-10), Phillipa Gill and Robert Beverly (Eds.). Association for Computing Machinery (ACM). https://doi.org/10.1145/3355369.3355580

[31] Patrick McManus. 2018. *Improving DNS Privacy in Firefox.* https://blog.nightly.mozilla.org/2018/06/01/improving-dns-privacy-in-firefox/

[32] Giovane CM Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. 2020. Clouding up the Internet: how centralized is DNS traffic becoming?. In *Proceedings of the 2020 Internet Measurement Conference (IMC)* (Virtual Event, 2020-10), Fabián Bustamante and Nick Feamster (Eds.). Association for Computing Machinery (ACM). https://doi.org/10.1145/3419394.3423625

[33] Mozilla. 2020. *Comcast's Xfinity Internet Service Joins Firefox's Trusted Recursive Resolver Program.* https://blog.mozilla.org/blog/2020/06/25/comcasts-xfinity-internet-service-joins-firefoxs-trusted-recursive-resolver-program/

[34] Mozilla. 2020. *The Facts: Mozilla's DNS over HTTPs (DoH).* https://blog.mozilla.org/netpolicy/2020/02/25/the-facts-mozillas-dns-over-https-doh/

[35] Nicole Perlroth. 2016. *Hackers Used New Weapons to Disrupt Major Websites Across U.S.* https://www.nytimes.com/2016/10/22/business/internet-problems-attack.html

[36] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. *Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation.* https://tranco-list.eu/

[37] Paul Schmitt, Anne Edmundson, Allison Mankin, and Nick Feamster. 2019. Oblivious DNS: Practical Privacy for DNS Queries. In *Proceedings of the 19th Privacy Enhancing Technologies* (19 ed.) (Stockholm, Sweden, 2019-07), Carmela Troncoso and Kostas Chatzikokolakis (Eds.). Sciendo, 228–244. https://doi.org/10.2478/popets-2019-0028

[38] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. 2020. Encrypted DNS -> Privacy? A Traffic Analysis Perspective. In *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)* (27 ed.) (San Diego, CA, USA, 2020-02), Xu Dongyan and Ahmad-Reza Sadeghi (Eds.). Internet Society (ISOC). https://doi.org/10.14722/ndss.2020.24301

[39] Paul Vixie. 2019. *My Chromecast Ultra Would Not Start Until I Began Answering 8.8.8.8.* https://mailarchive.ietf.org/arch/msg/dnsop/WCVv57IizUSjNb2RQNP84fBclI0/

[40] Michael Walfish, Hari Balakrishnan, and Scott Shenker. 2004. Untangling the Web from DNS.. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (1 ed.) (San Francisco, CA, USA, 2004-03), Robert Morris and Stefan Savage (Eds.). USENIX Association. https://dl.acm.org/doi/10.5555/1251175.1251192

[41] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. 2015. Connection-oriented DNS to Improve Privacy and Security. In *Proceedings of the 36th IEEE Symposium on Security & Privacy (S&P)* (36 ed.) (San Jose, CA, USA, 2015-05), Vitaly Shmatikov and Lujo Bauer (Eds.). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/sp.2015.18