

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 11, 2020

J. Arkko  
Ericsson  
M. Thomson  
Mozilla  
T. Hardie  
Google  
March 10, 2020

Selecting Resolvers from a Set of Distributed DNS Resolvers  
draft-arkko-abcd-distributed-resolver-selection-01

## Abstract

This memo discusses the use of a set of different DNS resolvers to reduce privacy problems related to resolvers learning the Internet usage patterns of their clients.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 11, 2020.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">2</a>
<a href="#">2.</a>	<a href="#">Operational Context . . . . .</a>	<a href="#">4</a>
<a href="#">3.</a>	<a href="#">Goals and Constraints . . . . .</a>	<a href="#">4</a>
<a href="#">4.</a>	<a href="#">Query distribution strategies . . . . .</a>	<a href="#">6</a>
<a href="#">4.1.</a>	<a href="#">Client-based . . . . .</a>	<a href="#">6</a>
<a href="#">4.1.1.</a>	<a href="#">Analysis of client-based selection . . . . .</a>	<a href="#">6</a>
<a href="#">4.1.2.</a>	<a href="#">Enhancements to client-based selection . . . . .</a>	<a href="#">7</a>
<a href="#">4.2.</a>	<a href="#">Name-based . . . . .</a>	<a href="#">7</a>
<a href="#">4.2.1.</a>	<a href="#">Name reduction . . . . .</a>	<a href="#">8</a>
<a href="#">5.</a>	<a href="#">Early conclusions . . . . .</a>	<a href="#">9</a>
<a href="#">5.1.</a>	<a href="#">Analysis conclusions . . . . .</a>	<a href="#">9</a>
<a href="#">5.2.</a>	<a href="#">Recommendations . . . . .</a>	<a href="#">9</a>
<a href="#">5.3.</a>	<a href="#">Poor distribution strategies . . . . .</a>	<a href="#">9</a>
<a href="#">6.</a>	<a href="#">Effects of query distribution . . . . .</a>	<a href="#">10</a>
<a href="#">6.1.</a>	<a href="#">Caching considerations . . . . .</a>	<a href="#">10</a>
<a href="#">6.2.</a>	<a href="#">Consistency considerations . . . . .</a>	<a href="#">10</a>
<a href="#">6.3.</a>	<a href="#">Resolver load distribution and failover . . . . .</a>	<a href="#">11</a>
<a href="#">6.4.</a>	<a href="#">Query performance . . . . .</a>	<a href="#">11</a>
<a href="#">6.5.</a>	<a href="#">Debugging . . . . .</a>	<a href="#">11</a>
<a href="#">7.</a>	<a href="#">Further work . . . . .</a>	<a href="#">11</a>
<a href="#">8.</a>	<a href="#">References . . . . .</a>	<a href="#">12</a>
<a href="#">8.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">12</a>
<a href="#">8.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">12</a>
<a href="#">8.3.</a>	<a href="#">URIs . . . . .</a>	<a href="#">13</a>
<a href="#">Appendix A.</a>	<a href="#">Acknowledgements . . . . .</a>	<a href="#">13</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">13</a>

## [1.](#) Introduction

The DNS [[DNS](#)] is a complex system with many different security issues, challenges, deployment models and usage patterns. This document focuses on one narrow aspect within DNS and its security.

Traditionally, systems are configured with a single DNS recursive resolver, or a set of primary and alternate recursive resolvers. Recursive resolver services are offered by organisations such as enterprises, ISPs, and global providers. Even when clients use alternate recursive resolvers, they are typically all provided by the

same organisation.

The resolvers will learn the Internet usage patterns of their clients. A client might decide to trust a particular recursive resolver with information about DNS queries. However, it is

difficult or impossible to provide any guarantees about data handling practices in the general case. And even if a service can be trusted to respect privacy with respect to handling of query data, legal and commercial pressures or surveillance activity could result in misuse of data. Similarly, outside attacks may occur towards any DNS services. For a service with many clients, these risks are particularly undesirable.

This memo discusses whether DNS clients can improve their privacy through the potential use of a set of multiple recursive resolver services. The goal is indeed an improvement only. There is no expectation that it would be possible to have no part of the DNS infrastructure aware of what queries are being made, but perhaps there are mitigations that would make possible information collection from the DNS infrastructure harder.

It should be understood that this is a narrow aspect within a bigger set of topics even within privacy issues around DNS, let alone other security issues, deployment models, or the many protocol questions within DNS. Some of these other topics include detecting the tampering DNS query responses [[DNSSEC](#)], encrypting DNS queries [[DOT](#)] [[DOH](#)], application-specific DNS resolution mechanisms, or centralised deployment models. Those other topics are not covered in this memo and need to be dealt with elsewhere.

Specifically, the scope of this memo is not limited to DNS-over-TLS (DOT) or DNS-over-HTTPS (DOH) deployments nor does it take a stand on operating system vs. application or local vs. centralized DNS deployment models. This memo is intended to provide useful information for those that wish to consider trustworthiness of their recursive resolvers as a part of their privacy analysis.

Naturally, there are some interactions between different topics. For instance, privacy is affected both by what happens to data in transit and at the endpoints, so where privacy is a concern, one would expect to consider both aspects, and lack of consideration on one probably

leads to issues that dwarf the problems that this memo can address. Both issues are also important aspects when considering defense against pervasive monitoring efforts [[PMON](#)].

The rest of this memo is organized as follows. [Section 2](#) discusses the operational context that we imagine the multiple recursive resolver arrangement might be applied in. [Section 3](#) specifies security goals for a system that employs multiple recursive resolvers.

One key aspect of a system using multiple resolvers would be how to select which particular recursive resolver to use in a particular

situation. This is discussed in [Section 4](#). This section covers a number of possible strategies and further considerations for this selection, along with an analysis of the implications of choosing a particular strategy. There are technical issues in the use of multiple recursive resolvers, and there are both technical and non-technical questions in deciding on what recursive resolvers should even be in the set.

Some early recommendations are provided in [Section 5](#). [Section 6](#) discusses operational and other implications of the distributed approach. Finally, [Section 7](#) discusses potential further work in this area.

## [2](#). Operational Context

Our perspective is that of a client, choosing to either distribute or not distribute its queries to a set of different resolvers. And if the client decides to use distribution, it can choose exactly how it does that.

There are obviously additional operational aspect of this – such as central configuration mechanisms, resolver selection application choices, and so on. But these are not covered in this memo.

It should also be observed that the practices suggested in this memo are currently not widely used. Operational and other issues may be discovered, such as those outlined in [Section 6](#).

Many of these issues need further work, but this memo aims to discuss

the concept and analyse its impacts before dwelling into the technical arrangements for configuring and using this particular approach.

### 3. Goals and Constraints

This document aims to reduce the concentration of information about client activity by distributing DNS queries across different resolver services, for all DNS queries in the aggregate and for DNS queries made by individual clients. By distributing queries in this way, the goal is to reduce the amount of information that any given DNS resolver service can acquire about client activity. As such, creates a benefit for the client, but also makes these resolvers less valuable targets for attacks relating to that client's activity.

Any method for distributing queries from a single client needs to consider these benefits with regards to the following constraints:

- o A careful selection of the set of trusted resolvers must be the first priority. It does not make sense to add less trustworthy resolvers merely for the sake of distribution. For instance, there is no reason to mix resolvers with good reliability or high degree of privacy regulation with other resolvers: just include the best resolvers in the set.
- o As the goal is to reduce the amount of information given to any given resolver, a strategy that tells the same information to all resolvers is a poor one. A design that results in replicating the same query toward multiple services would thus be a net privacy loss. This happens quite easily over a long period of time, unless the distribution method is carefully designed.

More subtle leaks arise as a result of distributing queries for sub-domains and even domains that are superficially unrelated, because these could share a commonality that might be exploited to link them. For instance, some web sites use names that appear unrelated to their primary name for hosting some kinds of content, like static images or videos. If queries for these unrelated names were sent to different services, that effectively allows multiple resolvers to learn that the client accessed the web site.

A distribution scheme also needs to consider stability of query routing over time. A resolver can observe the absence of queries and infer things about the state of a client cache, which can reveal that queries were made to other resolvers. In general, different queries for the same resolution context, such as sub-resources for a web page load, which have some probability of being related or linked, should be sent to the same resolver. Failure to do so reveals information to more than one resolver.

In effect, there are two goals in tension:

- o split queries between as many different resolvers as possible; and
- o reduce the spread of linkable queries across multiple resolvers.

The need to limit replication of private information about queries eliminates naive distribution schemes, such as those discussed in [Section 5.3](#). The designs described in [Section 4](#) all attempt to balance these different goals using different properties from the context of a query ([Section 4.1](#)) or the query name itself ([Section 4.2](#)).

## [4.](#) Query distribution strategies

This section introduces and analyzes several potential strategies for distributing queries to different resolvers. Each strategy is formulated as an algorithm for choosing a resolver  $R_i$  from a set of  $n$  resolvers  $\{R_1, R_2, \dots, R_n\}$ .

The designs presented in [Section 4](#) assume that the stub resolver performing distribution of queries has varying degrees of contextual information. In general, more contextual information allows for finer-grained distribution of information between resolvers.

### [4.1.](#) Client-based

The simplest strategy is to distribute each different client to a

different resolver. This reduces the number of users any particular service will know about. However, this does little to protect an individual user from the aggregation of information about queries at the selected resolver.

In this design clients select and consistently use the same resolver. This might be achieved by randomly selecting and remembering a resolver. Alternatively, a resolver might be selected using consistent hashing that takes some conception of client identity as input:

$$i = h(\text{client identity}) \% n$$

For the purposes of this determination, a client might be an entire device, with the selection being made at the operating system level, or it could be a selection made by individual applications. In the extreme, an individual application might be able to partition its activities in a way that allows it to direct queries to multiple resolvers.

#### [4.1.1.](#) Analysis of client-based selection

This is a simple and effective strategy, but while it provides distribution of DNS queries in the aggregate, it does little to divide information about a particular client between resolvers. It effectively only reduces the number of clients that each resolver can acquire information about. This provides systemic benefit, but does not provide individual clients with any significant advantage as there is still some resolver service that has a complete view of the user's DNS usage patterns.

In addition, there are specific issues where this selection method is used in particular deployment modes. Where different applications

make independent resolver selections, activities that involve multiple applications can result in information about those activities being exposed to multiple resolvers. For instance, an application could open another application for the purposes of handling a specific file type or to load a URL. This could expose queries related to the activity as a whole to multiple resolvers.

Making different selections at the level of a device resolves this

issue. But of course it is still possible that an individual who uses multiple devices might perform similar activities on those devices, but have DNS queries distributed to different resolvers, resulting in replicating that individual's information to multiple resolvers. The individual may or may not be identifiable through fingerprinting of the specific set of queries being made from the devices.

#### [4.1.2.](#) Enhancements to client-based selection

Clients can break continuity of records by occasionally resetting state so that a different resolver is selected. A client might choose to do this when it moves to a new network location, and may otherwise appear as a new client its current resolver. But it is unclear if there's a sufficient advantage to breaking continuity, as the potential benefits are offset by the client's information being disclosed to several resolvers as part of performing a series of resets. And it is possible that a particular individual's usage patterns can be identified across network locations and periods of using other resolvers.

Breaking continuity is less effective if any state, in particular cached results, is retained across the change. If activities that depend on DNS querying are continued across the change then it might be possible for the old resolver to make inferences about the activity on the new resolver, or the new resolver to make similar guesses about past activity. As many modern applications provide session continuity features across shutdowns and crashes, this can mean that finding an appropriate point in time to perform a switch can be difficult.

#### [4.2.](#) Name-based

Clients might also additionally attempt to distribute queries based on the name being queried. This results in different names going to different resolvers.

A naive algorithm for name distribution uses the target name as input to a fixed hash:

However, this simplistic approach fails to prevent related queries from being distributed to different resolvers in several ways. For instance, queries that are executed after receiving a CNAME record in a response will leak the same information as the original query that resulted in the CNAME record. Services that use related domain names – such as where "example.com" uses "static.example.com" or "cdn.example.net" – might reveal the use of the combined service to a resolver that receives a query for any associated name. In both cases, sensitive information is effectively replicated across multiple resolvers.

#### [4.2.1](#). Name reduction

In order to reduce the effect of distributing similar names to different servers, a grouping mechanism might be used. Leading labels in names might be erased before being input to the hashing algorithm. This requires that the part of the suffix that is shared between different services can be identified. For the purposes of ensuring that queries are consistently routed to the same resolver, a weak signal is likely sufficient.

Several options for grouping domain names into equivalence sets might be used:

- o The public suffix list [[1](#)] provides a manually curated list of shared domain suffixes. Names can be reduced to include one label more than the list allows, referred to as effective top-level domain plus one (eTLD+1). This reduces the number of cases where queries for domains under the same administrative control are sent to different resolvers.
- o Services often relies on multiple domain names across different eTLD+1 domains. Developing equivalence sets might be needed to avoid broadcasting queries to servers. Mozilla maintains a manually curated equivalence list [[2](#)] for web sites that aims to maps the complete set of unrelated names used by services to a single service name.
- o Other technologies, such as the proposed first party sets [[3](#)] or the abandoned DBOUND [[DBOUND](#)] provide domain owners a means to declare some form of equivalence for different names.

Each of these techniques are imperfect in different ways. They may also skew the distribution of queries in ways that might concentrate information on particular resolvers. Moreover, resolver choice based solely on the name to be resolved rather than per-client information

reduces the anonymity set of queries sent to each resolver. In contrast to a client-based strategy, attackers can predict the target resolver for a given name using a name-based strategy. This may have implications for on-path attacker attempts to identify otherwise encrypted queries. Of course, even a name-based mechanism might use some non-public information as well for its choice, which might reduce these issues.

## [5.](#) Early conclusions

### [5.1.](#) Analysis conclusions

Both the client-based and more advanced name-based strategies may provide benefits. The former may provide primarily a systemic benefit, while the latter may provide also some privacy benefits to each individual client. However, neither strategy is perfect, and can leak the same information to multiple resolvers in some cases.

### [5.2.](#) Recommendations

Both strategies are, however, likely generally beneficial in the common cases, and can improve the overall privacy situation. And they are certainly a considerable privacy improvement over a situation where a large number of clients use a single resolver.

Their use may also reduce any pressures against specific resolvers, as information available in these specific resolvers does not constitute all information about all clients. As such, the use of one of these distribution strategies is tentatively recommended, subject to further testing, discussion, and resolving any remaining operational issues.

The naive name-based strategy is, however, not recommended, and neither are other, even simpler strategies listed in [Section 5.3](#). It should be noted that no technique presented in this memo can defend against a situation where an actor such as a surveillance agency has access to information from all resolvers.

### [5.3.](#) Poor distribution strategies

Random allocation to a resolver might be implemented:

```
i = rand() % n
```

Similar drawbacks can be seen where clients iterate over available resolvers:

i = counter++ % n

Whether this choice is made on a per-query basis, these two methods eventually provide information about all queries to all resolvers over time. Domain names are often queried many times over long periods, so queries for the same domain name will eventually be distributed to all resolvers. Only one-off queries will avoid being distributed.

Implementing either method at a much slower cadence might be effective, subject to the constraints in [Section 4.1.2](#). This only slows the distribution of information about repeated queries to all resolvers.

## [6.](#) Effects of query distribution

Choosing to use more than one DNS resolver has broader implications than just the effect on privacy. Using multiple resolvers is a significant change from the assumed model where stub resolvers send all queries to a single resolver.

### [6.1.](#) Caching considerations

Using a common cache for multiple resolvers introduces the possibility that a resolver could learn about queries that were originally directed to another resolvers by observing the absence of queries. Though this can reduce caching performance, clients can address this by having a per-resolver cache and only using the cache for the selected resolver.

### [6.2.](#) Consistency considerations

Making the same query to multiple resolvers can result in different answers. For instance, DNS-based load balancing can lead to different answers being produced over time or for different query origins. Or, different resolvers might have different policies with respect to blocking or filtering of queries that lead to clients receiving inconsistent answers.

In the extreme, an application might encounter errors as a result of receiving incompatible answers, particularly if a server operator

(incorrectly) assumes that different DNS queries for the same client always originate from the same source address. This is most likely to occur if name-based selection is used, as queries could be related based on information that the client does not consider.

### [6.3.](#) Resolver load distribution and failover

Any selection of resolvers that is based on random inputs will need to account for available capacity on resolvers. Otherwise, resolvers with less available query-processing capacity will receive too high a proportion of all queries. Clients only need to be informed of relative available capacity in order to make an appropriate selection. How relative capacities of resolvers are determined is not in scope for this document.

The choice of different resolvers would also need to work well with whatever mechanisms exist for failover to alternate resolvers when one is not responsive. The same is true of IPv4/IPv6 connectivity, the availability of communications to specific ports, etc. And the dynamic situation should obviously not lead to extensive leakage to different resolvers, either.

### [6.4.](#) Query performance

Distribution of queries between resolvers also means that clients are exposed to greater variations in performance [[HBSHF19](#)]. In contrast, using a single resolver, as would result from the client-based method in [Section 4.1](#), promotes use of a persistent connection.

### [6.5.](#) Debugging

The use of multiple resolvers may complicate debugging.

## [7.](#) Further work

Should there be interest in the deployment of ideas laid out in this memo, further work is needed. There would have to be ways to

configure systems to use multiple resolvers, including for instance:

- o Central configuration mechanisms to enable the use of multiple resolvers, perhaps through usual network configuration mechanisms or choices made by applications using resolver services directly. It may also be necessary to employ discovery mechanisms, such as, e.g., [[I-D.schinazi-httpbis-doh-preference-hints](#)] or [[I-D.pauly-dprive-adaptive-dns-privacy](#)] (but see [Section 3](#))
- o Mechanisms to allow both failover to working resolvers when a resolver is unreachable,
- o Additional testing for potential operational issues discussed in [Section 2](#) would be beneficial.

Finally, more work is needed to determine factors other than privacy that could motivate having queries routed to the same resolver. The choice between different approaches is often a combination of several factors, and privacy is only one of those factors.

## [8.](#) References

### [8.1.](#) Normative References

- [DNS] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [DNSSEC] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", [RFC 4033](#), DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [DOH] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", [RFC 8484](#), DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [DOT] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", [RFC 7858](#), DOI 10.17487/RFC7858, May

2016, <<https://www.rfc-editor.org/info/rfc7858>>.

- [PMON] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", [BCP 188](#), [RFC 7258](#), DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.

## 8.2. Informative References

- [DBOUND] Levine, J., "Publishing Organization Boundaries in the DNS", [draft-levine-dbound-dns-03](#) (work in progress), April 2019.
- [HBSHF19] Austin Hounsell, ., Kevin Borgolte, ., Paul Schmidt, ., Jordan Holland, ., and . Nick Feamster, "Analyzing the Costs (and Benefits) of DNS, DoT, and DoH for the Modern Web", ANRW '19, July 22, 2019, Montreal, QC, Canada, n.d., <<https://dl.acm.org/doi/10.1145/3340301.3341129>>.
- [I-D.pauly-dprive-adaptive-dns-privacy] Kinnear, E., Pauly, T., Wood, C., and P. McManus, "Adaptive DNS: Improving Privacy of Name Resolution", [draft-pauly-dprive-adaptive-dns-privacy-01](#) (work in progress), November 2019.

Arkko, et al.

Expires September 11, 2020

[Page 12]

---

Internet-Draft

Distributed Resolver Selection

March 2020

- [I-D.schinazi-httpbis-doh-preference-hints] Schinazi, D., Sullivan, N., and J. Kipp, "DoH Preference Hints for HTTP", [draft-schinazi-httpbis-doh-preference-hints-01](#) (work in progress), January 2020.
- [MSCVUS] Wikipedia, ., "Microsoft Corp. v. United States", [https://en.wikipedia.org/wiki/Microsoft\\_Corp.\\_v.\\_United\\_States](https://en.wikipedia.org/wiki/Microsoft_Corp._v._United_States), n.d..

## 8.3. URIs

- [1] <https://publicsuffix.org/>
- [2] <https://github.com/mozilla-services/shavar-prod-lists/blob/master/disconnect-entitylist.json>
- [3] <https://github.com/krgovind/first-party-sets>

## [Appendix A](#). Acknowledgements

The authors would like to thank Christian Huitema, Ari Keraenen, Mark Nottingham, Stephen Farrell, Gonzalo Camarillo, Mirja Kuehlewind, David Allan, Daniel Migault, Goran AP Eriksson, Christopher Wood, and many others for interesting discussions in this problem space.

### Authors' Addresses

Jari Arkko  
Ericsson

Email: [jari.arkko@piuha.net](mailto:jari.arkko@piuha.net)

Martin Thomson  
Mozilla

Email: [martin.thomson@gmail.com](mailto:martin.thomson@gmail.com)

Ted Hardie  
Google

Email: [ted.ietf@gmail.com](mailto:ted.ietf@gmail.com)