Extensible delegations in DNS Recursive resolvers

Jesse van Zutphen, University of Amsterdam
jzutphen@os3.nl

Supervisors: Willem Toorop, Yorgos Thessalonikefs & Philip Homburg

July 8, 2024

Abstract

This paper explores implementing an extensible delegation mechanism in the Domain Name System (DNS) using only recursive resolver adaptations. The main question of this research is: "Is an approach of extensible delegation, that needs only resolver adaptation to bootstrap deployment, feasible to implement and deploy?". We introduce _deleg records for resolvers to dynamically discover delegation points without changes to authoritative name servers. Our approach allows incremental deployment and supports legacy systems. In our implementation, we found three query behaviors: no additional queries with upgraded name servers, one additional query with Deleg data but no upgrade, and three queries when Extensible Delegation support is unknown. Despite variations, query round trip times remain consistent due to parallelization. What makes our method unique in comparison to the other Extensible Delegation draft, is the mode where the name server is not upgraded to be able to support legacy systems. For this, one extra query is needed. Analysis of the .nl zone suggests similar additional query loads comparable to `DNSKEY` queries, approximately 0.50% of the total load. These results affirm the feasibility of our approach, justifying the additional workload by the improved delegation efficiency and ease of deployment it offers.

Keywords— Domain Name Server, Extensible delegations, Recursive resolver, DNSSEC, Unbound, Authoritative name server

1 Introduction

The design of the Domain Name System (DNS) is more than 40 years old [1], [2]. Its primary goal at the time was to facilitate the growth of the Internet in a scalable manner, and it has certainly succeeded in this regard. However, at the time, it did not and could not possibly have anticipated the current day Internet and how it is organized. The role of "DNS Operators" was not anticipated and does not have a place in the current DNS architecture.

As a result of recent discussions at the Internet Corporation for Assigned Names and Numbers (ICANN) DNS Symposium (IDS) 2023 [3], the 41st DNS Operations Analysis and Research Centre (DNS-OARC) workshop [4] and a DNS Hackathon session at the 118th Internet Engineering Task Force (IETF) conference [5], a new initiative has started to revisit the way delegations are accomplished in the DNS. This has resulted in the formation of a new IETF DNS delegation working group for which a charter [6] is currently being designed. Currently there is at least one draft proposal by Tim April[7] to be worked on in the new working group, which was initiated during the 118th IETF DNS Hackathon.

The "Extensible Delegation for DNS" draft proposal [7] introduces a new DNS Resource Record (RR) type at the delegation zone cut, maintaining the particularity in the DNS that the name of a delegation exists in both the parent and the child domain. The current proposal needs support in both the serving authoritative name servers, as well as in resolvers. We believe that an alternative approach where the delegation information is provided not at the zone cut, but authoritatively elsewhere in the parent zone, reduces the deployment requirements by needing support in the resolver only. Note that one of the objectives of the DNS delegation working group [6] is to "consider how well different solutions can be deployed, and should study possible consequences of deploying alternative delegation mechanisms".

Our proposal is to implement a new extensible delegation mechanism (designed in collaboration with staff at NLnet Labs), that needs implementation in the resolver only. The evaluation of the implementation with respect to efficiency and deployability could be valuable input for the first DNS delegation working group session that will have its first session at the IETF 120 in Vancouver in July this year.

1.1 Paper structure

This paper is structured into ten chapters. Chapter 2 will go into the defined main research question and supporting sub questions. Chapter 3 will describe the scientific contributions that have been made during this research. Chapter 4 provides context for the research by delving into the background and describing related works. Chapter 5 describes the experimental methodology by going over the experimental setup and reproducible steps for collecting the data. Chapter 6 represents the results of the experiments. Chapter 7 discusses the results of Chapter 6 and interprets them by adding additional context. Chapter 8 concludes the research and summarizes the findings by referring back to the research questions. Chapter 9 suggests areas for further research to build on this research findings. Chapter 10 outlines the ethical considerations of this research.

2 Research questions

This research focuses on an extensible delegation approach that strives for reduced deployment complexity by needing only resolver software adaptation. A proof of concept of the approach will be implemented in the Unbound resolver software.

Main Research Question

- Is an approach of extensible delegation, that needs only resolver adaptation to bootstrap deployment, feasible to implement and deploy?

Sub-Questions

1. What does such an approach look like and how would it work?
2. What resolver adaptations need to be made?
3. What is the performance impact of this approach with respect to resolver workload and traffic?
4. How does this approach compare to the other extensible delegation proposal?

3 Scientific contribution

During this research, I made an implementation of an Extensible Delegation mechanism that only needs implementation in the resolver. For as far as we know,

this is the very first Extensible Delegation implementation that is created. The other draft [8] has no supporting code yet and has as far as we know, only a specification in the form of an Internet Draft.

With this implementation, I have analyzed the performance of our Extensible Delegation Mechanism and compared it with the original draft.

In addition with this paper I am also working as co-author of the "Incrementally Deployable Extensible Delegation for DNS" Internet Draft [9], which is a proposal by Nlnet Labs to implement Extensible Delegations.The results from this research are used as a baseline for the Incrementally Deployable Extensible Delegation for DNS specification.

4 Background & Related works

4.1 DNS and delegations

The Domain Name System (DNS) is a critical component of the Internet's infrastructure, providing the necessary service of translating human-readable domain names (e.g., www.nlnetlabs.nl) into IP addresses that, for example, a web browser uses to identify their destination on the Internet. DNS operates in a hierarchical structure consisting of the root zone, top-level domains (TLDs), second-level domains, and so on. Each level of the hierarchy may be managed by different parties, and may be served by different authoritative servers. The delegation of authority from one level to the next is crucial for the distributed management of the DNS.

In traditional DNS delegation, the parent zone delegates authority to the child zone using NS (Name Server) records. These records are placed at the "zone cut", the boundary between the parent and child zones. An NS record specifies the authoritative name servers for the child zone. Additionally, if DNS Security Extensions (DNSSEC) are in use, a `DS` (Delegation Signer) record is included at the parent zone. The `DS` record contains a hash of the `DNSKEY` record from the child zone, establishing a chain of trust from the parent zone to the child zone.

For instance, the delegation of the zone child.example.com from example.com involves adding NS records for child.example.com in the example.com zone file. If DNSSEC is enabled, a `DS` record for child.example.com is also included in example.com to ensure cryptographic validation of the child zone's data.

4.2 Challenges in normal delegation

In a traditional DNS setup, when a DNS query is made for a subdomain, the parent domain's DNS server provides a referral to the authoritative DNS server of the child domain. However, a significant issue arises in this delegation process: the parent domain is not authoritative over the Name Server (NS) records in the referral. This also means that the parent does not sign these records and thus cannot be checked with DNSSEC. The consequence of this is that clients must trust this referral without a way to verify the authenticity of the child zone's `NS` record. This makes the name server prone to on-path substitution attacks[10].

Another problem is that both `NS` record at the parent and child should be consistent with each other. RFC 1034, Section 4.2.2[11] states the following about this: "As the last installation step, the delegation `NS` resource records and glue resource records necessary to make the delegation effective should be added to the parent zone. The administrators of both zones should insure that the `NS` and glue resource records which mark both sides of the cut are consistent and remain so". But in practice mistakes happen with keeping the delegations consistent. Research by Raffaele Sommese[12] delves into NS record inconsistency between the parent and child. This research shows that roughly 8% of the zones have an inconsistency between the parent and child `NS` records. The consequences of this are improper load balancing, increased latency, and unresponsive name servers.

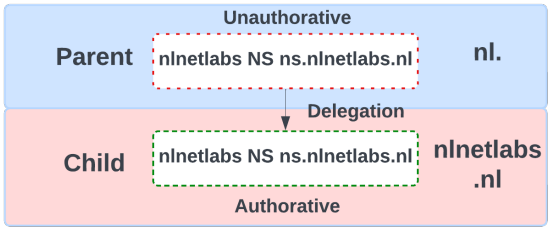

Figure 1: Authority of NS records

4.3 Extensible Delegations

To solve the problems of normal delegations, the `Deleg` draft[7] creates a new Resource Record at the parent side of the delegation. This Resource Record

can be used as a referral response and also adds the IP addresses in the response, making glue records unnecessary. It also has the capability to add transport protocol capabilities. This makes it possible to immediately set up an encrypted channel to the child server. An example of such a `Deleg` record can be found in figure 2. This resource record is based on the RFC 9460 `SVCB` record[13].

```
c1.example.org. 86400 IN DELEG 1 config3.
    example.net. ( alpn=dot ipv4hint
    =192.0.2.54 )
```

Figure 2: draft Deleg record [7]

What this resource record indicates is that c1.example.org is a delegation point in the current zone. This server can be reached at IPv4 address 192.0.2.54 and should be queried with DNS over TLS. Since unlike the traditional `NS` record, the parent is authoritative over this record. The record should also be DNSSEC signed and should also have the associated `RRSIG` records. Next to signing the delegation and being able to signal the transport protocol, there is also a third reason to implement extensible delegations. The service hosting industry makes extensive use of the `CNAME` records. `CNAME` records have the capability to give an individual name a different alias, which can be used to alias a name to a name within the zone of a service hosting entity, giving them administrative control over that name. The drawback with `CNAME` records is that according to RFC 2181 Section 10.3[14], `CNAME` records cannot be used to alias an `NS` record. `SVCB` records also have an alias mode, but this is allowed at the zone apex as well. This construct can be used to give administrative control of the entire authoritative name server to a DNS operator. Who can then make modifications to the authoritative name server without the need to notify the parent zone.

4.3.1 Alternatives for Deleg

There are also alternatives for Deleg. One of them is proposed in the draft for Delegation Revalidation [15]. This draft suggests sending an extra query in parallel to acquire the child domain `NS` resource record set. This way, the resolver will get the child's part of the delegation and be able to DNSSEC verify if the delegation that has been made is correct. Still, this does mean that a query needs to be made to an

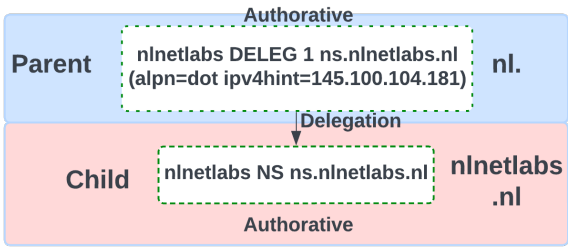

Figure 3: Authority of NS records with extensible delegations

"untrusted" authoritative name server. With Extensible Delegations, it is possible to proactively check whether the delegation is DNSSEC secure.

Another alternative was proposed by Fujiwara Kazunori in the draft for delegation information signer [16]. Here, a new Resource Record is added to the parent zone that contains a signed hash of the delegation so that Resolvers can validate whether the delegation is correct or not.

To sum up, there are other proposals to validate the delegation, but what Extensible Delegations makes unique is that it adds a way to signal the use of encrypted protocols like DNS over TLS, and it also adds the capability of aliasing delegations.

4.3.2 Disadvantages other Extensible Delegations draft

The main disadvantages of the Deleg draft by Tim April are based on the fact that the `Deleg` record is located at the zone cut. This is similar to how the `DS` resource record has been implemented at the zone cut [17]. According to DNS pioneer Paul Vixie [18], putting the `DS` at the zone cut was a mistake. The reason for this is that it adds confusion and complexity. A delegation strongly implies that the child is authoritative for these records. The `DS` record is the exception for that, and with the current extensible delegation proposal, it seems like the same exception will be made for the `DELEG` record. The draft says the following about this [7]: "The `DELEG` resource record type is unusual in that it appears only on the parent zone's side of a zone cut. For example, the `DELEG` RRset for the delegation of 'foo.example' is part of the 'example' zone rather than in the 'foo.example' zone. This requires special processing rules for both name servers and resolvers because the name server for the child zone is authoritative for the name at the

zone cut by the normal DNS rules, but the child zone does not contain the DELEG RRset." The consequence of this can be that certain resolvers will return BOGUS. The reason for that is that it is unexpected to have records at the zone cut that are signed by the parent, as this is normally done by the child. Because of that, it is likely that legacy resolvers will not be able to DNSSEC verify these records.

Another disadvantage that comes with having the record at the zone cut is how to prove the existence of the Deleg record. The other Extensible Delegation draft does this by using a flag in the DNSKEY. When this flag is set, it means that the name server understands Extensible Delegations, and should return the NSEC records that prove that the Deleg record does not exist. This adds some complications since the DNSKEY was never designed to be used for signaling, so this is a hack. As a result of this, since the DNSKEY record is in the child zone, the child determines whether the parent uses Extensible Delegations. While normally the child has no input about what happens at the parent zone. This also means that before upgrading to Extensible Delegations, coordination is needed with the child name server. Another result of using the flag in the DNSKEY is that since this flag is binary, when this flag is on, all name servers that serve this zone need to be upgraded to support Deleg records at the zone cut. When there is a zone that is hosted by a lot of different name servers this can result in an administrative challenge. Lastly, because the draft expects the Deleg resource record to be added in the authority section of the query response, this needs additional logic inside the name server. The consequence of this is that the name server has to be upgraded in order to use Extensible Delegations, which is unfeasible for all name servers, since there will always be legacy software around.

5 Methodology

This section will focus on the proposed algorithm to implement extensible delegations in the recursive resolver and also on how the experiments are structured to test this implementation.

5.1 Custom extensible delegation algorithm

For this research, we defined our own extensible delegation algorithm that enhances the traditional delegation mechanism. The purpose and objective of this algorithm is to implement extensible delegations without the need to add records at the zone cut, requiring only logic changes in the recursive resolver and no changes in the authoritative name server.

5.1.1 Algorithm design

For our designed algorithm, we do not want the name server to add a record at the zone cut, instead, what we did was add an SVCB record in the parent zone of the delegation. The format of the name of the record is as follows: "{child zone delegation name}._deleg.{FQDN of the parent}" in figure 4 is an example of such record for the .nl zone that has a delegation to nlnetlabs.nl.

```
nlnetlabs._deleg.nl SVCB 1
 ns.nlnetlabs.nl ipv4hint=185.49.140.60
     ipv6hint=2a04:b900::8:0:0:60
```

Figure 4: _deleg record in the .nl zone

When the recursive resolver wants to resolve a query, it first checks its cache for information about the authoritative name server at the current delegation point. The resolver determines whether:

1. The authoritative name server has been upgraded to support the new delegation mechanism.
2. The authoritative name server has not been upgraded, but the zone file contains _deleg records.
3. The authoritative name server has not been upgraded, lacks _deleg records in its zone, or the resolver has no cached information about this server.

At this point, the resolver can be in one of three states regarding its knowledge about the authoritative name server:

1. Support in the name server is unknown: If the resolver has no information about support in the name server, nor about whether or not a _deleg label exists at the apex of the target zone, it sends three queries in parallel:

- An SVCB query for _deleg at the apex of the zone. This query helps determine if the zone may contain _deleg records. The result is cached

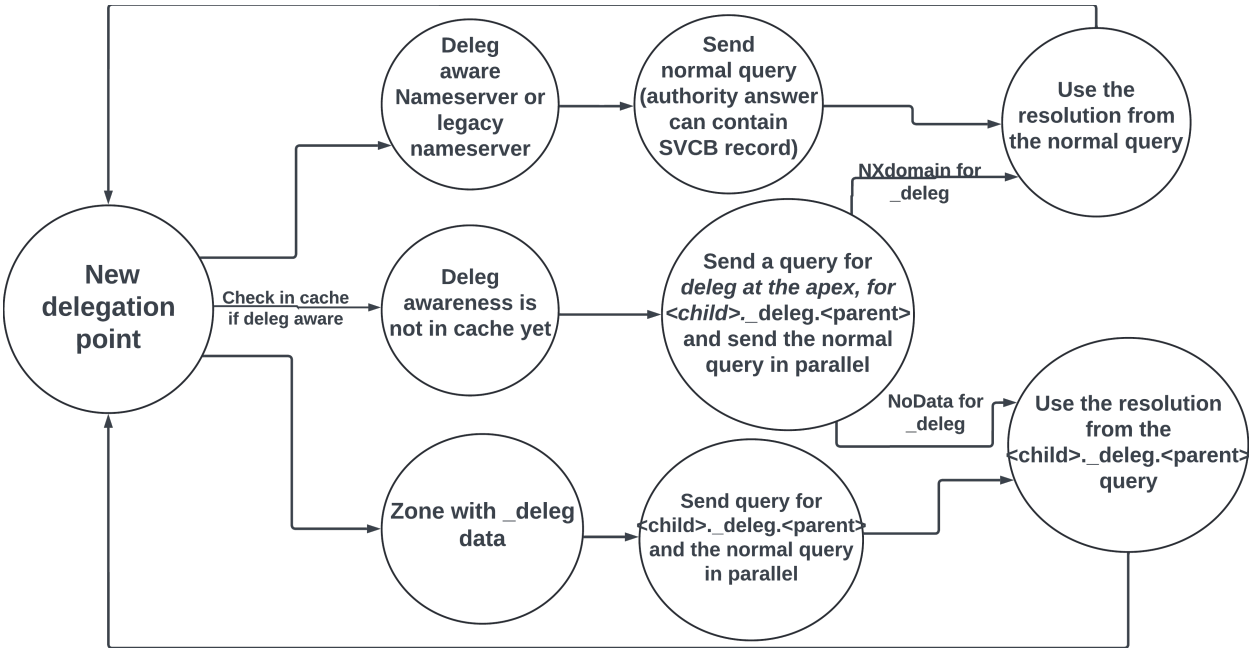

Figure 5: State machine extensible delegation algorithm

for future use. If the response is NXDOMAIN, it indicates no _deleg records in the zone. If the response is NODATA, it signifies Extensible Delegation data exists within the zone.

- A query for <child>._deleg.<parent>, checking if there are specific _deleg records for the child zone.
- A standard query to obtain an NS or SVCB RRset. This query is used to fallback on NS records when Extensible Delegations are not supported, or to receive an SVCB RRset in the authority section if the authoritative server has been upgraded. If a SVCB RRset containing referral information is in the authority section of the referral response, the resolver caches this information, indicating that the name server supports extensible delegations and thus no additional queries are needed when this server gets queried again.

Queries are sent in parallel because the DNS protocol restricts sending multiple queries in a single packet (QDCOUNT must be set to one) [19]. Parallel querying ensures the resolver maintains an equal number of round trips.

2. Legacy name server with _deleg Data in the zone: If the resolver knows that the zone contains _deleg records but the name server is not upgraded, it sends two parallel queries:

- A query for '<child>._deleg.<parent>'.
- A standard query to obtain the legacy referral information. The result of the _deleg query takes priority. The NS RRset is used if this specific delegation has not been made (yet) for Extensible Delegations. Additionally, the standard query is necessary for DNSSEC to retrieve the DNSKEY record.

3. Name server with support for extensible delegations: If the resolver has cached information indicating that the name server has support for extensible delegations, it sends only the standard query. The authoritative section of the response will contain the SVCB RRset (or proof of non-existence) for the extensible delegation.

The algorithm has been implemented in Unbound version 1.20.1 and the code can be found at my Github [20]. The state machine for this algorithm can be found in figure 5.

5.1.2 Advantages/disadvantages implementation

An interesting aspect of this algorithm is that Query Name Minimisation [21] is inherently part of it. The reason is that for every query to get the delegation, only the delegation name of the child is sent to the parent instead of the entire query. This results in the same behavior as Query Name Minimisation. Recent research by Jonathan Magnusson et al. [22] has shown that Query Name Minimisation adoption is at 64%. Implementing our proposed Extensible Delegation method could increase this adoption. However, requiring Query Name Minimisation could also be a disadvantage. A 2015 study by Shumon Huque showed that 12.8% of Query Name Minimisation queries failed [23].

The main disadvantage of our Extensible Delegation method is that extra queries need to be made in order to find out if the name server supports Extensible Delegations and whether the target zone contains Extensible Delegations. The result of that is that the load of name servers will increase.

5.2 Experiments & Testing

5.2.1 Experimental objectives

To evaluate the impact of the new delegation mechanism on the workload of recursive DNS resolvers, a controlled experimental environment has been established. This environment consists of custom root and TLD servers, and a multi-level delegation hierarchy. The goal of this experiment is to first test the correctness of the implemented algorithm. And secondly to analyze the performance of this new algorithm. We do this by analysing the amount of queries that is needed to perform in order to resolve a query and the amount of round trips that are needed. These metrics are chosen because the amount of queries signify the added load that Authoritative Name Servers need to handle and we know that our solution will add extra queries. For the amount of round trips is chosen because this signifies the time it will take for a query to resolve and thus the performance of the algorithm. Here we compare the following cases: normal delegations without Query Name Minimisation, normal delegations with Query Name, our Extensible Delegation implementation (best and worst case) and the draft for Extensible Delegations by Tim April [8]. We have chosen normal delegations without Query Name minimisation to act as a base case. We use normal delegations with query name because, as mentioned earlier, our implementations behaves in a similar way as Query Name Minimisation and its interesting to see if we also get a similar amount of queries that have to be done. We also have a test case for the other draft of Tim April to compare with. This is interesting to see how to different implementations measure up with each other.

We specifically chose for the amount of queries as metric for the experiments because this is the main disadvantage of our implementation as described in Section 5.1.2.

5.2.2 Infrastructure setup

For the experiments the following setup has been created:

- A root server
- A Top Level Domain server (serving the .nl. zone)
- Nlnetlabs server (serving the nlnetlabs.nl. zone)
- Zagreb server (serving the zagreb.nlnetlabs.nl. zone)

In this setup there is also a recursive resolver. This recursive resolver is running Unbound version 1.20.1 with the additions to the code to use the implemented Extensible Delegation algorithm.

in figure 6 the experimental setup is visualized. To reproduce the experiments, the used zone files can be found in appendix a.1.

5.2.3 Experimental procedure

For the experiments, the recursive resolvers will query for '`TXT` test.zagreb.nlnetlabs.nl'. This ensures that all the delegations of the test environment are followed. By querying a `TXT` record instead of an A record, we can observe the full behavior of the QNAME minimisation aspect of the algorithm. The algorithm will send A queries to higher-level domains to anonymize the type of record being requested.

In these experiments, the cache of the recursive resolver will be empty, except for the cached root zone after root priming. Root priming is an essential preliminary step that typically occurs before querying. By ensuring that other zones are not pre-cached, we can accurately analyze the resolver's behavior as

it queries all the name servers in our test environment. This setup allows for an assessment of the algorithm's performance and impact when the name servers have not been upgraded. Also, by having the cache empty, we can assess our algorithm in the worst case, since it has no data in its cache about whether the name servers have Extensible Delegation support or whether there is Extensible Delegation data in the zone. With this, we can assess the query overhead of our algorithm in the least favorable situation.

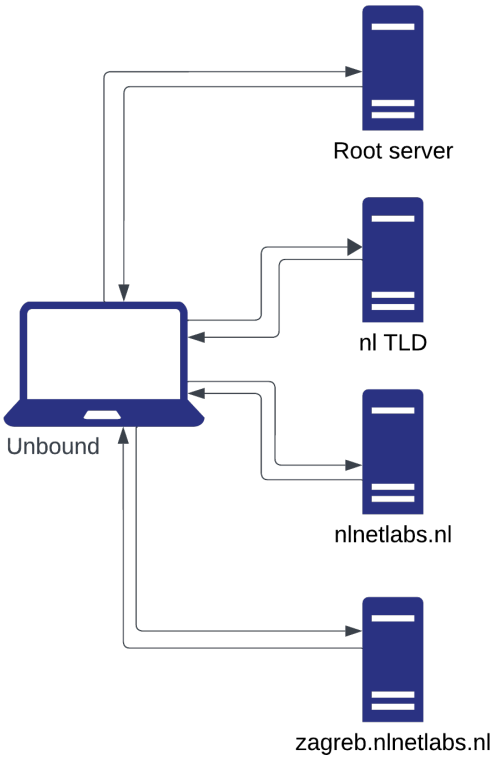

Figure 6: Experimental setup

6 Results

6.1 Resolver behaviour

After analysing the behavior of incremental deployable Deleg within the experimental setup we are able to define that the amount of queries is dependent on three boolean variables:

- (_) _deleg is present in the apex of the parent zone, or it is not present.
- (A) The authoritative name server that serves the parent supports Extensible Delegations.
- (S) The parent zone is secure, or it is insecure.

The resolver will always have to perform one additional query for _deleg per zone when it is not cached yet, regardless of whether this name server supports Extensible Delegations. Once the resolver determines if the name server supports Extensible Delegations, it will follow the behavior shown In table 1. The query behavior is dependent on the three different boolean values.

6.2 Performance

To evaluate the performance, we compared our algorithm to the base case of Query name minimisation within the test environment. The comparison for a resolver with an empty cache and no information about support at the authoritative name servers is shown in Appendix a.2. In our test scenario, we observed that our solution requires additional queries. Incremental deployable Deleg needs 12 queries, while the base case of QNAME minimization needs 5 queries. The round trip time remains the same at 5. The increased number of queries is a result of the extra steps needed to handle when the name

Table 1: Table of query behavior

_	A	S	Query behavior
			No additional <child>._deleg.<parent> queries
		X	No additional <child>._deleg.<parent> queries
	X		No additional <child>._deleg.<parent> queries
	X	X	No additional <child>._deleg.<parent> queries
X			Parallel legacy and <child>._deleg.<parent> queries (for every query)
X		X	Parallel legacy and <child>._deleg.<parent> queries (for every query)
X	X		-Assume no _deleg awareness initially, and act as if the auth is not _deleg aware (parallel legacy and <child>._deleg.<parent> queries (for every query). -Assume _deleg awareness when <child>._deleg.<parent> SVCB for the delegation is seen in the authority section in the referral response. Assume awareness for the duration of the TTL of the _deleg SVCB RRset. -When _deleg awareness is assumed, only do legacy queries and expect the <child>._deleg.<parent> SVCB RRset in the authority section (if it exists)
X	X	X	Assume no _deleg awareness initially, and act as if the name server is not _deleg aware (parallel legacy and <child>._deleg.<parent> queries (for every query). -Assume _deleg awareness when a signed <child>._deleg.<parent> SVCB for the delegation is seen in the authority section in the referral response, or a DNSSEC proof of non-existence of such RRset. -Assume awareness for the duration of the TTL of the _deleg SVCB RRset or the TTL of the NSEC(3) proofing the non-existence. -When _deleg awareness is assumed, only do legacy queries and expect a signed <child>._deleg.<parent> SVCB RRset in the authority section, or a proof of non-existence for that RRset. If neither are present, do a follow up query for <child>._deleg.<parent>.

server does not have support for Extensible Delegations. The conventional query returning the legacy conventional referral information needs to remain for fallback (when there is no extensible delegation) and also for the DS RRset for DNSSEC. Also because the cache is empty and thus the resolver does not know that there is Extensible Delegation data in the zone, there is another extra query needed to _deleg. But because all the extra queries that are needed can be sent in parallel, this will not affect the round trip time.

7 Discussion

7.1 Performance impact of additional queries

One of the critical findings from our experiment is that incremental deployable Deleg requires more queries than QNAME minimization. Specifically, our implementation needs 12 queries (in the worst case) compared to the 5 required by QNAME minimization. However, both approaches maintain the same number of round trips, indicating that the overall resolution time is not negatively impacted. The acceptability of additional DNS queries has been a subject of study in various contexts. For instance, DNS over QUIC (DoQ) introduces additional queries for encryption and security purposes, yet according to a study by Mike Kosek et al. [24] it maintains performance comparable to traditional DNS methods. This suggests that modern DNS infrastructure can handle extra queries in parallel efficiently without significantly degrading performance.

Most of these extra queries were needed because we were using a cold cache. After the query for _deleg at the apex was cached, our solution would have only needed an extra query per delegation (and if there was no _deleg, data no extra queries would be needed). This query behavior is similar to the `DNSKEY` query behavior. For instance, on July 1st, 2024, 0.50% of queries in the .nl zone were for the `DNSKEY` resource record [25]. In the case for the `DNSKEY` the extra query is generally accepted by the DNS community due to the security benefits [26]. Our solution's additional queries can be similarly justified by the improved delegation efficiency and ease of deployment it offers.

7.2 Comparison with other Extensible Delegation draft

Our solution requires more queries than the other Extensible Delegation draft, particularly during the transition period when name servers do not yet support Extensible Delegations. However, this comparison is not entirely fair, as the original draft does not accommodate this transitional mode. Once name server support is established, our method requires no additional queries. Only when the resolver has to find out if there is Extensible Delegation support from the name server or zone, are there two additional queries needed to gather this information for the cache.

8 Conclusion

This research set out to investigate the feasibility of an extensible delegation approach that requires only resolver adaptation to bootstrap deployment. In this Section we will reflect on the research questions defined in Section 3. To start with the first sub-question: "*What Does Such an Approach Look Like and How Would It Work?*". The approach that we defined involves querying for _deleg records as the first step in the delegation process. If these records are found, the resolver uses the provided `SVCB` record hints for IPv4 or IPv6 addresses to determine the new delegation point. If the _deleg records are not found, the resolver falls back on standard DNS resolution procedures. This dual-path mechanism ensures backward compatibility and allows gradual deployment without requiring changes to authoritative name servers.

For the second sub-question: "*What Resolver Adaptations Need to Be Made?*". The main adaptation needed in recursive resolvers is the ability to query and interpret _deleg queries and handle `SVCB` records to extract delegation hints. This involves modifying the resolver's query logic to include an additional step for _deleg queries and parsing the RRsets accordingly to change the delegation point based on the fully qualified target name and SvcParams. The implementation was implemented into the Unbound resolver version 1.20.1, which can be found on GitHub [20]. In addition to the code, a test bed has been created to verify the behavior of the resolver and reproduce the results.

The third subquestion was: "*What Is the Performance Impact of This Approach with Respect to Resolver Workload and Traffic?*". Our custom implementation increases the number of DNS queries per resolution process but maintains the same number of round trips as QNAME minimization. Specifically, with name server support, only one additional query per zone, is required, which is comparable to the `DNSKEY` queries used in DNSSEC. For the .nl zone, this resulted in an increase in query load (0.50%). When there is no name server support but Deleg data in the zone, two queries are needed per zone, and when there is no Deleg data and no name server support, three queries are needed.

Previous research by Mike Kosesk et al. has shown that modern DNS can handle additional queries in parallel without significantly reducing the performance. This demonstrates that the additional workload introduced by our solution is manageable within modern DNS infrastructures.

The last subquestion was: "*How Does This Approach Compare to the other Extensible Delegation Proposal?*". The only fair comparison to be made between our implementation and the original draft is when there is name server support, because that is the only mode that the original draft supports. In this case, our implementation needs one more query because of the trade-off of not using the `DNSKEY` as a signaling mechanism. But the round trip time stays the same regardless.

Finally, to conclude the main question: "*Is an approach of extensible delegation, that needs only resolver adaptation to bootstrap deployment, feasible to implement and deploy?*". The results of our experiments and performance evaluations affirm that this approach is indeed feasible.

Our implementation of the extensible delegation mechanism using _deleg in the parent zone demonstrated that necessary adaptations to recursive resolvers can be integrated by adding additional logic. Performance evaluations showed that while the new mechanism increases the number of queries, it does not significantly impact resolution times due to efficient query handling and parallel processing capabilities. This ensures that the overall impact on resolver workload is manageable. Additionally, the ease of deployment is a significant advantage. By requiring changes only at the resolver level, our solution can be gradually rolled out without disrupting existing DNS operations, facilitating incremental adoption. This incremental deployment capability reduces the risk associated with widespread changes and makes it possible to adopt Extensible Delegations even when us-

ing legacy systems.

9 Future works

This research has mainly been focused on adapting recursive resolvers to implement the extensible delegation mechanism using _deleg records. While this approach has proven feasible and effective, this research was only focused on making Extensible Delegations work by making modifications to the recursive resolvers. To further expand our testbed, research can be done on how to upgrade the name servers to implement Extensible Delegations. This would reduce the number of queries needed and improve the overall efficiency of the DNS resolution process.

Another future work could be to monitor the deployment rate of Extensible Delegations by querying TLD's in a cron job to gather statistics about the current deployment. Similar research has been done for the deployment of DNSSEC [27] and DANE [28].

10 Ethical paragraph

For this research, we have delved into a new way of implementing DNS Extensible delegations into Unbound. DNS is a critical component of the internet architecture. Advancements into DNS can affect millions of people around the world. Because of this the implementation, needs to be well documented because transparency is important in order for my work to be reviewed and possible be reused later on.

Lastly, my research has not involved any user data and all my tests have been done on isolated domains allocated for the purpose of testing the implementation. No critical infrastructure has been handled during my research and the impact on real world operations has been minimal.

References

[1] P. Mockapetris, *Domain names: Concepts and facilities*, RFC 882, Nov. 1983. DOI: 10.17487/RFC0882. [Online]. Available: https://www.rfc-editor.org/info/rfc882.

[2] P. Mockapetris, *Domain names: Implementation specification*, RFC 883, Nov. 1983. DOI: 10.17487/RFC0883. [Online]. Available: https://www.rfc-editor.org/info/rfc883.

[3] ICANN, *ICANN DNS Symposium*. [Online]. Available: https://www.icann.org/ids.

[4] OARC, *OARC 41*. [Online]. Available: https://indico.dns-oarc.net/event/47/.

[5] IETF, *IETF 118 Hackathon*. [Online]. Available: https://www.ietf.org/meeting/hackathons/118-hackathon/.

[6] IETF, *Dns delegation*, 2024. [Online]. Available: https://datatracker.ietf.org/doc/charter-ietf-deleg/.

[7] IETF, *Extensible delegation for dns*, 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-dnsop-deleg/.

[8] R. Weber, *Draft extensible delegation for dns*. [Online]. Available: https://github.com/fl1ger/deleg.

[9] P. Homburg, J. van Zutphen, and W. Toorop, *Incrementally deployable extensible delegation for dns*, 2024. [Online]. Available: https://nlnetlabs.github.io/incremental-deleg/draft-homburg-deleg-incremental-deleg.html.

[10] J. Abley, "REFER: A New Referral Mechanism for the DNS," Internet Engineering Task Force, Internet-Draft draft-jabley-dnsop-refer-00, Feb. 2021, Work in Progress, 14 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-jabley-dnsop-refer/00/.

[11] *Domain names - concepts and facilities*, RFC 1034, Nov. 1987. DOI: 10.17487/RFC1034. [Online]. Available: https://www.rfc-editor.org/info/rfc1034.

[12] R. Sommese, *When parents and children disagree: Diving into dns delegation inconsistency*. [Online]. Available: https://ris.utwente.nl/ws/portalfiles/portal/237707586/Sommese2020when.pdf.

[13] B. M. Schwartz, M. Bishop, and E. Nygren, *Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)*, RFC 9460, Nov. 2023. DOI: 10.17487/RFC9460. [Online]. Available: https://www.rfc-editor.org/info/rfc9460.

[14] R. Elz and R. Bush, *Clarifications to the DNS Specification*, RFC 2181, Jul. 1997. DOI: 10.17487/RFC2181. [Online]. Available: https://www.rfc-editor.org/info/rfc2181.

[15] S. Huque, P. A. Vixie, and W. Toorop, "Delegation Revalidation by DNS Resolvers," Internet Engineering Task Force, Internet-Draft draft-ietf-dnsop-ns-revalidation-06, Mar. 2024, Work in Progress, 10 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dnsop-ns-revalidation/06/.

[16] K. Fujiwara, "Delegation Information (Referrals) Signer for DNSSEC," Internet Engineering Task Force, Internet-Draft draft-fujiwara-dnsop-delegation-information-signer-00, Nov. 2020, Work in Progress, 6 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-fujiwara-dnsop-delegation-information-signer/00/.

[17] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends, *Resource Records for the DNS Security Extensions*, RFC 4034, Mar. 2005. DOI: 10.17487/RFC4034. [Online]. Available: https://www.rfc-editor.org/info/rfc4034.

[18] *Encrypted DNS Policy and Technical Call*. [Online]. Available: https://419.consulting/encrypted-dns/f/deleg-the-hairy-dns-camel.

[19] R. Bellis and J. Abley, *In the dns, qdcount is (usually) one*, Feb. 2023. [Online]. Available: https://www.ietf.org/archive/id/draft-bellis-dnsop-qdcount-is-one-00.html.

[20] J. van Zutphen, *Implementation extensible delegations in unbound*, https://github.com/jessevz/unbound, 2024.

[21] S. Bortzmeyer, R. Dolmans, and P. E. Hoffman, *DNS Query Name Minimisation to Improve Privacy*, RFC 9156, Nov. 2021. DOI: 10.17487/RFC9156. [Online]. Available: https://www.rfc-editor.org/info/rfc9156.

[22] J. Magnusson, M. Muller, A. Brunstrom, and T. Pulls, "A second look at dns qname minimization," in *Passive and Active Measurement*, A. Brunstrom, M. Flores, and M. Fiore, Eds., Cham: Springer Nature Switzerland, 2023, pp. 496–521, ISBN: 978-3-031-28486-1.

[23] S. Huque, *Query name minimization and authoritative dns server behavior*. [Online]. Available: https://indico.dns-oarc.net/event/21/contributions/298/attachments/267/487/qname-min.pdf.

[24] M. Kosek, L. Schumann, R. Marx, T. V. Doan, and V. Bajpai, *Dns privacy with speed? evaluating dns over quic and its impact on web performance*, May 2023. [Online]. Available: https://www.marilia.unesp.br/Home/Instituicao/Docentes/RosangelaCaldas/topicosdesenvolvimentodoensaiocientifico/book---scientific-writing.pdf.

[25] S. labs, *Dns statistics*, Jul. 2024. [Online]. Available: https://stats.sidnlabs.nl/en/dns.html.

[26] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends, *DNS Security Introduction and Requirements*, RFC 4033, Mar. 2005. DOI: 10.17487/RFC4033. [Online]. Available: https://www.rfc-editor.org/info/rfc4033.

[27] ICANN, *Dnssec deployment*. [Online]. Available: https://ithi.research.icann.org/graph-m7.html.

[28] G. M. University, *Global dnssec deployment tracking*. [Online]. Available: https://secspider.net/growth.html.

A Appendix

A.1 Zone files

```
$ORIGIN .
$TTL 3600
@             SOA    root-server jesse.zagreb.nlnetlabs.nl (
                            2024061300 ; serial
                            1800      ; refresh (30 minutes)
                            900       ; retry (15 minutes)
                            604800    ; expire (1 week)
                            3600      ; minimum (1 hour)
)
@             NS     root-server
root-server   A      152.42.143.251
root-server   AAAA   2a03:b0c0:2:d0::1630:7001
_deleg        SVCB   1 root-server ipv4hint=152.42.143.251 ipv6hint=2a03:b0c0:2:d0::1630:7001

;; nl delegation
nl._deleg     SVCB   1 ns.dns.nl. ipv4hint=178.62.197.215 ipv6hint=2a03:b0c0:2:d0::1605:a001

;; Legacy nl delegation
nl            NS     ns.dns.nl
;; Glue
ns.dns.nl.    A      178.62.197.215
ns.dns.nl.    AAAA   2a03:b0c0:2:d0::1605:a001
```

Listing 1: root zone

```
$ORIGIN nl.
$TTL 3600
@             SOA    ns.dns jesse.zagreb.nlnetlabs (
                            2024061300 ; serial
                            1800      ; refresh (30 minutes)
                            900       ; retry (15 minutes)
                            604800    ; expire (1 week)
                            3600      ; minimum (1 hour)
)
;; Legacy NS entry.
@             NS     ns.dns
ns.dns        A      178.62.197.215
ns.dns        AAAA   2a03:b0c0:2:d0::1605:a001

_deleg SVCB   1 ns.dns.nl. ipv4hint=178.62.197.215 ipv6hint=2a03:b0c0:2:d0::1605:a001

;; nlnetlabs.nl delegation
nlnetlabs._deleg     SVCB   1 ns.nlnetlabs ipv4hint=185.49.140.60 ipv6hint=2a04:b900::8:0:0:60

;; Legacy nlnetlabs.nl delegation
nlnetlabs            NS     ns.nlnetlabs
;; Glue
ns.nlnetlabs         A      185.49.140.60
ns.nlnetlabs         AAAA   2a04:b9
```

Listing 2: nl. zone

```
$TTL 10200                          ; 3 hours
$TTL 240                            ; 4 minutes
$ORIGIN nlnetlabs.nl.

@      IN     SOA    ns hostmaster (
                     2024061600    ; Serial
                        28800      ; Refresh 8 hours
                         7200      ; Retry  2 hours
                       604800      ; Expire 7 days
                          240      ; Minimum 1 hours
              )

nameservers   IN     SVCB   1 ns ipv4hint=185.49.140.60 ipv6hint=2a04:b900::8:0:0:60
              IN     SVCB   1 ns.nlnetlabs.org. ipv4hint=185.49.141.53 ipv6hint=2a04:b900:0:100::53

zagreb._deleg IN     SVCB   1 ns.zagreb ipv4hint=145.100.104.181 ipv6hint
    =2001:610:158:1046:145:100:104:181

yorgos._deleg IN     SVCB   0 nameservers
jesse._deleg  IN     SVCB   0 config.zagreb
willem._deleg IN     SVCB   0 zagreb._deleg
philip.homburg._deleg IN    SVCB   0 config.zagreb

legacy-deleg  IN     NS     ns.legacy-deleg
ns.legacy-deleg IN   A      145.100.104.181
ns.legacy-deleg IN   AAAA   2001:610:158:1046:145:100:104:181
```

Listing 3: nlnetlabs zone

```
$ORIGIN zagreb.nlnetlabs.nl.
$TTL 86400

@ IN SOA ns.zagreb.nlnetlabs.nl. jessevz@nlnetlabs.nl (2024060601
                                                       3600
                                                       1800
                                                       1209600
                                                       86400
                                                      )
  IN NS ns.zagreb.nlnetlabs.nl.
  IN NS ns.legacy-deleg.nlnetlabs.nl.
config.zagreb.nlnetlabs.nl 86400 IN SVCB 1 . (ipv4hint=145.100.104.181 ipv6hint
    =2001:610:158:1046:145:100:104:181)
config.zagreb.nlnetlabs.nl. IN A 145.100.104.186
zagreb.nlnetlabs.nl. IN A 145.100.104.199
test IN TXT "text record for experiments"
```

Listing 4: zagreb zone

A.2 Query performance

Table 2: Table of performance incremental Deleg

	Query → / ← Response		
rt	**Incremental deployable Deleg**	**Qname minimization**	**name server**
1	nl. A → ← nl.NS nl._deleg . SVCB → ← nl._deleg.SVCB	nl. A → ← nl. NS	root
2	nlnetlabs.nl. A → ← nlnetlabs.nl. NS _deleg.nl. SVCB → ← _deleg.nl. (NODATA) nlnetlabs._deleg.nl. SVCB → ← nlnetlabs._deleg.nl. SVCB	nlnetlabs.nl. A → ← nlnetlabs.nl. NS	.nl.
3	zagreb.nlnetlabs.nl. A → ← zagreb.nlnetlabs.nl. NS _deleg.nlnetlabs.nl. SVCB → ← _deleg.nlnetlabs.nl. (NODATA) zagreb._deleg.nlnetlabs.nl. SVCB → ← nlnetlabs._deleg.nlnetlabs.nl. SVCB	zagreb.nlnetlabs.nl A → ← zagreb.nlnetlabs.nl. NS	nlnetlabs.nl.
4	test.zagreb.nlnetlabs.nl. A → ← test.zagreb.nlnetlabs.nl. (NODATA) _deleg.zagreb.nlnetlabs.nl. SVCB → ← _deleg.zagreb.nlnetlabs.nl. (NXDOMAIN) test._deleg.zagreb.nlnetlabs.nl. SVCB → ← test._deleg.zagreb.nlnetlabs.nl. (NXDOMAIN)	test.zagreb.nlnetlabs.nl. A → ← test.zagreb.nlnetlabs.nl. (NODATA)	zagreb.nlnetlabs.nl.
5	test.zagreb.nlnetlabs.nl. TXT → ← test.zagreb.nlnetlabs.nl. TXT	test.zagreb.nlnetlabs.nl. TXT → ← test.zagreb.nlnetlabs.nl. TXT	zagreb.nlnetlabs.nl.
	Round-trips: 5, queries: 12	Round-trips: 5, queries 5	