University of Amsterdam

Research Project 1

Extensible delegations in authoritative nameservers

February 9, 2025

Student:
Wouter Petri
1301944

Supervisors:
Willem Toorop; Philip Homburg

Abstract

This paper focusses on implementing optimisations for incremental deleg in the Domain Name System (DNS) in authoritative nameservers and designing an testbed for adaquate evaluation. The research questions are "What are the requirements of a testbed environment for fair and adequate evaluation of the proposal?" and "What behaviour optimisations can be done at the authoritative nameserver to achieve reduced (simultaneous) queries and how to implement them?". This paper shows the design of such a testbed and the requirements needed such as the zone file. The approach in this paper allows for reduced amount of simultaneous queries in resolvers when they are aware of incremental deleg. This is done by including the incremental delegation data in the response or by including the authenticated denial of existence in the response to signal to the resolver that the incremental delegation point does not exist.

Keywords— Domain Name Server, Extensible delegations, authoritative nameserver, nsd, simdzone, incremental deleg, incremental delegation

Contents

1 Introduction

In June 2024, "deleg" the IETF working group was founded with the purpose of developing and specifying a new way for delegations in the Domain Name System (DNS). The current method for realising delegations in the DNS has not been altered since its design in 1983 [1][2]. This new method aims to resolve some of the limitations that the current legacy method has with respect to capability signalling and securing referrals. Another goal is to align DNS delegation with modern operational practice and give authoritative DNS operators a role in design.
In July 2024, Jesse van Zutphen realised the resolver implementation for the NLnet Labs draft [3]. As of today, there is no implementation in the authoritative nameserver for optimisations for this proposed protocol.
In November 2024, the working group finished establishing the protocol's requirements and is now ready to commence protocol development. Currently, there are two proposed protocols. One NLnet Labs made [4] and one other [5]. During IETF 121 in Dublin, it was agreed that proof-of-concept implementations would be worked on and that testbeds for both protocols would be realised to enable the working group to experiment with and evaluate both protocols.

1.1 Paper structure

This paper is structured into 9 sections. Section 2 will define the research questions. Section 3 will describe the scientific contributions that have been made during this research. Section 4 will then go into the background of this research. Then section 5 will provide the context of the research using related work. Sections 6 and 7 cover the methods used in this research, as well as the implementations and tests performed during this research. Section 8 represent the results that will be discussed in section 9 where as well the suggestions for future work will be.

2 Research Questions

This project had as a starting point to realise the remaining matters for a testbed environment to evaluate the protocol made by NLnet Labs [4]. This will include a proof-of-concept implementation for the authoritative side of the protocol.

The project has the following research questions:

- What are the requirements of a testbed environment for fair and adequate evaluation of the proposal?
- What behaviour optimisations can be done at the authoritative nameserver to achieve reduced (simultaneous) queries and how to implement them?

3 Scientific Contribution

During this research, I implemented an Extensible Delegation mechanism in a resolver and implemented a new RR type in a zoneparser. To our knowledge, this will be the first authoritative nameserver implementation for a new extensible delegation method. The implementation will provide optimised behaviour with reduced (simultaneous) queries, to have fewer queries over the Internet. I also created a setup and zone file for a testbed environment so that the proposal from NLnet Labs can be tested.

4 Background

4.1 DNS

The Domain Name System (DNS) is a very important component of the Internet nowadays. DNS provides translation from human-readable network addresses (such as os3.nl) into IP addresses that computers can use to communicate and acquire resources. Email and visiting websites are one of the many examples that everyone uses every day. DNS uses a hierarchical structure that consists of the root zone, top-level domains (TLDs such as `.nl`), second-level domains, and so on. Companies and persons can acquire second-level domains by registering them with a registrar. Each level in this structure can be managed by different parties, for example `.nl` zone is managed by SIDN and the `os3.nl` zone is managed by os3. Therefore, the different zones are not all served by the same nameserver. To traverse trough these different levels, DNS delegation can be used.

4.2 DNS delegations

The current method DNS delegations work is using NS (nameserver) records [6] When referring to a nameserver with a record that is a subdomain of the requested domain, a glue record is used to prevent circular dependencies [7]. In figure 1 is a simplified example of delegation. These records only support a domain name. This means that it is impossible to know what kind of security features the target nameserver supports.

The delegating nameserver (parent) and target nameserver (child) must both have the same NS record. However, this is not always the case; A study has shown that approximately 7.8% of the delegations in the `.com, .cnet` and `.org` zones are not consistent [8]

4.3 Challenges in DNS delegation

The created referral from the parent domain only includes a nameserver (NS) record to the child's domain. The parent nameserver is not authoritative over this referral data. Due to the fact that only authoritative data is signed in DNSSEC [9], NS records are not signed, and therefore it is relatively easy to perform an on-path substitution attack. During such an attack, a man in the middle that sees any query over the wire could then answer the query with the spoofed ip address. Any resolver will then follow the injected record and might retrieve invalid information.
There is also no information on what kind of security features the target nameserver supports, the only given fact is that the target nameserver should be available at the common known port 53 [10].

5 Related Work

5.1 The IDELEG RR type

The proposal from NLnet Labs describes a new RR type called IDELEG. This RR type is a variant of the Service Binding record (SVCB) [11]. For the implementation of this type, the internal code 65280 is used. This record will contain the following fields:

- Priority field
- Target name field
- Parameters

Just as in SVCB a priority of 0 indicates that the record is used in alias mode instead of in service mode. A lower priority value indicates a higher priority. This means that priority 1 will be preferred over priority value 2. Unlike in SVCB mixing of service mode and alias mode records is possible for the same name. The parameters are used to describe the ip hints and security features of the target nameserver.

IDELEG records will be placed with the "_deleg" label prefixed before the zone apex. So, referring to `webmail.os3.nl` the IDELEG record will be placed on `webmail._deleg.os3.nl`. Examples of a IDELEG record can be found in section 7.

5.2 Incremental deleg

The draft from NLnet Labs describes the mechanism for incrementally deployable extensible delegation [4], this becomes incremental since not all changes in the nameserver and resolver need to be implemented simultaneously for extensible delegation to work. Incremental delegation data is authoritative in the delegating parent; therefore, it will be signed when using DNSSEC. The draft also states that it is possible to outsource all delegation operational practise to another party using a DNAME record on the `_deleg` label on the apex of the delegating zone. Using multiple IDELEG records in alias mode is also possible to outsource delegation to different operators.

5.2.1 Incremental deleg in resolvers

Resolver support is crucial for extensible delegation to work, they will need to be able to work with the new IDELEG RR type and follow its delegations. For basic support of incremental deleg the resolvers would need to query the deleg delegation point at the `_deleg` label. These queries can be made simultaneously with the normal query. For example, the resolver should, in addition to `nlnetlabs.nl` also query `nlnetlabs._deleg.nl` to the `nl` nameserver. In this way, a resolver can see if incremental delegation exists. This behaviour should only occur if the triggering query is below the target zone apex. This prevents sending an additional query when not expecting a delegation. A resolver can keep track of whether a nameserver supports incremental deleg.

5.2.2 Incremental deleg in authoritative nameservers

Support in authoritative nameservers can help reduce the amount of queries sent over the network. Nameservers can include the response of

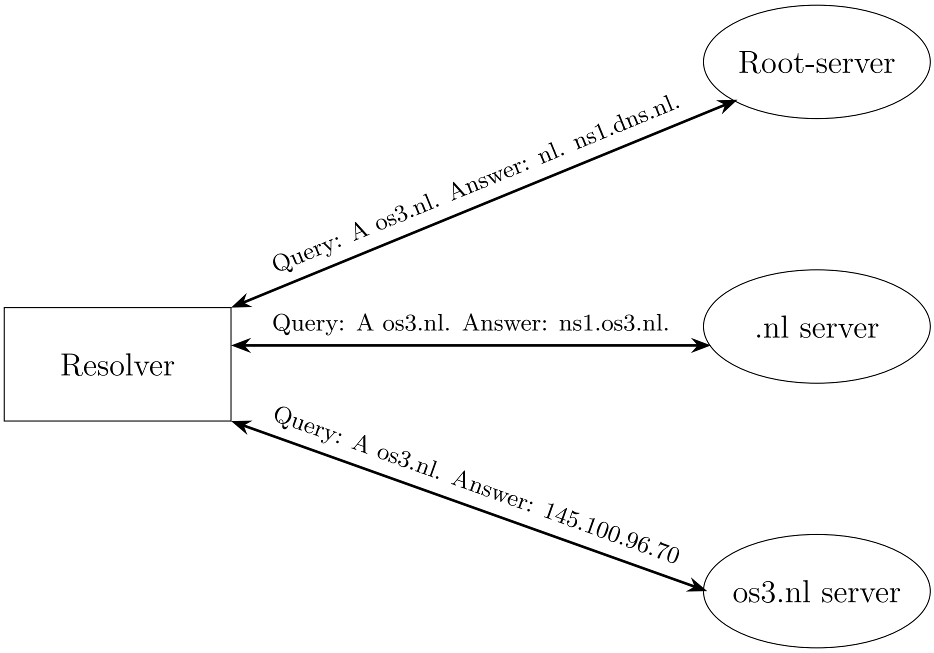

Figure 1: Simplified current delegation

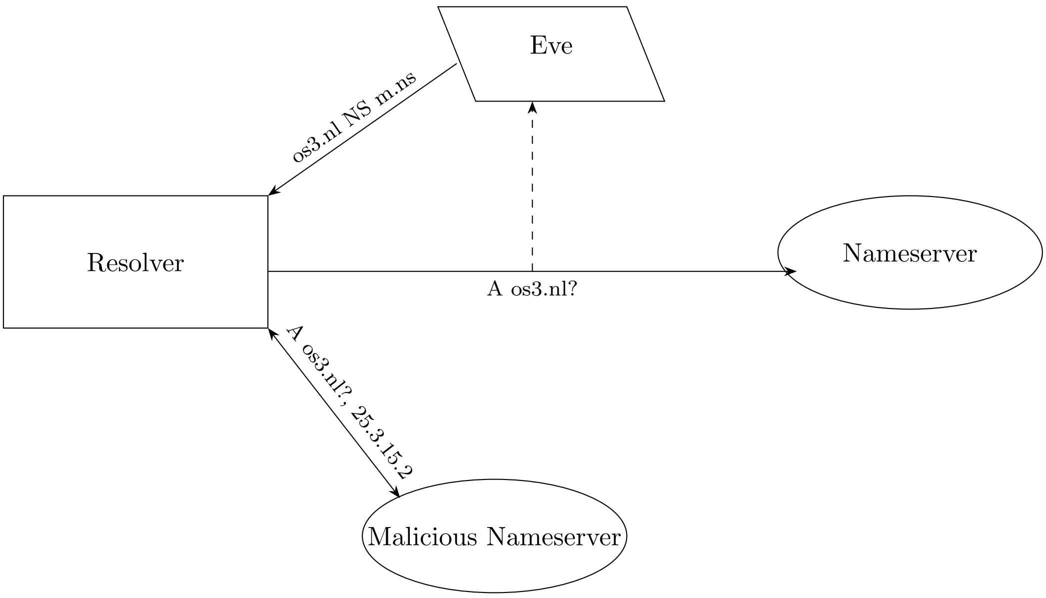

Figure 2: Simplified on-path substitution attack

incremental delegation together with the legacy referral. When using DNSSEC, the nameserver should also include the correct NSEC(3) proof with the answer so that the DNSSEC chain is correct. When a nameserver is looking for the incremental delegation, one out of the following 3 scenarios is true:

- IDELEG RRset(s) is found and should be added to the answer
- The incremental delegation point does not exist (NXDOMAIN)
- The incremental delegation point does exist but does not have any IDELEG RRsets

6 Methodology

This section will focus on the methods used to answer the proposed research questions. It is split to define the methods used per question.

6.1 Testbed requirements

Using the given proposal, we will look at the best method for a fair and adequate comparison of the proposal in an IETF testbed. More specifically, we will look at what setup would be required in such a testbed and what zone file(s) can be used.

6.2 Optimisations in authoritative nameservers

I will look at the current method of performing queries to authoritative nameservers using deleg and no deleg. Using the draft [4]. Then implement this in NSD and simdzone. One of the implementations will be to support the newly created IDLEG RR type and reduce the number of simultaneous queries required by implementing the behaviour specified in section 5.2.2

7 Implementation & Testing

This section will focus on the implementation and testing of the required behaviour of authoritative nameservers.

7.1 Implementation

To implement optimisations in the nameserver[1] there will first be the requirement that the nameserver can work with the new IDELEG RRset. There the implementation will exist of two parts, one in the zone parser and one in the nameserver daemon. Since all of these changes are experimental they will both have define guards and an configure option to enable them.

7.1.1 Zone parser

The full changes made in the code can be found on github [1]. To enable these changes in the parser the configuration flag `--enable-drafts` is used. Simdzone uses a hash to parse the zone files, since the hash matrix of types and classes differs when using new RRtypes the flag is not specified for IDELEG only. If the flag was more specific, an exponential amount of matrixes would be required when having support for multiple drafts. One draft would then require two matrixes, and two drafts would already require four. To prevent this, all drafts are under the same flag so that only two matrixes will be required. For the simdzone parser to work, there should be an implementation that binds the RRname used in a zonefile to the used number 65280. There should also be an implementation of a parser to parse the data and there should be information on how what fields the new record has. The fields of this type are mentioned in section 5.1. The parser used is a copy of the one used by the SVCB records, since there are no changes in the fields.

7.1.2 Nameserver daemon

The full list of code changes can be found on github [2]. To avoid accidentally enabling the optimised behaviour, the configuration flag `--enable-deleg` has been created. The changes made to support optimised behaviour have multiple steps;

1. Synthesising the ideleg delegation point

The correct delegation point consists of placing the _deleg label before the zone apex. To create this a function was created to add labels to a domain name (in nsd referenced as dname). To create this correct label, we used the delegation domain and the zone apex.

```
*ideleg_dname =
↪   labels_plus_dname(delegation_domain->dname,
↪   delegation_domain->dname->label_count -
↪   zone->apex->dname->label_count,
↪   label_plus_dname("_deleg",
↪   zone->apex->dname));
```

Code 1: Code used to create the incremental delegation domain name

After having acquired the correct delegation point, a lookup for the IDELEG records can be done.

2. Finding and adding the IDELEG records to the answer

With the correct delegation point, a lookup can be made to the domain database to see if it exists. If the records exist they will be added to the `AUTHORITY SECTION`. After this, there is support for incremental delegation without DNSSEC. The next steps will go into implementing the correct DNSSEC responses.

[1] https://github.com/NLnetLabs/simdzone/pull/248
[2] https://github.com/NLnetLabs/nsd/pull/422

3. Adding the correct NSEC responses When using DNSSEC, a zone will have NSEC or NSEC3 records, so we can prove that there are records or certain domain names that do not exist. These records will also show what records exist on that name. This step will go into the implementation of the NSEC responses, the next step will go into the NSEC3 responses. If the delegation point does exist but does not contain any IDELEG records, we will add the NSEC record of the delegation point to the `AUTHORITY SECTION`. If the delegation point does not exist, we must prove that it does indeed not exist. For authenticated denial of existence, the following NSEC records must be provided:

- Closest encloser
- NSEC record covering the next closer
- NSEC record covering the wildcard

The first will provide proof of which label under the delegation point does exist. This can be _deleg.<zone-apex> or the zone apex itself if the incremental deleg label does not exist. The second will provide proof that the requested domain name does not exist. The last one will prove that there is no wildcard. NSEC records will also provide the next canonical domain name. If such a name does not exist, it will provide the first. This makes it circular and allows the resolvers to see that the requested domain name does not exist. To find these records for our delegation point, a domain lookup is done that will return the closest match and the closest encloser as well. To find the proof for our delegation point, we then look up the NSEC records of these domains and include them in the `AUTHORITY SECTION`. If the zone is secure and no NSEC3 was used, these records exist. The nameserver daemon has already added the record covering the wildcard during one of the lookups.

4. Adding the correct NSEC3 responses Since NSEC allows domain walking, it is also possible to use NSEC3 which hashes all of the domain names. This prevents domain-walking, but also creates more difficulty to get the correct names. The same goes for when the delegation point exists; we will then add the NSEC3 record of that name. To provide authenticated denial of existance for nsec3, we will need to proof the same, but now with the hashed names. For the first and the last, we can find the NSEC3 proof with these records since the nsec3 cover is stored for existing domains; however, it is more difficult to find the NSEC3 record that covers the next closer. To find the next closer, we will have to create a label that is as long as the amount of labels of the encloser plus 1 and get those labels from the delgation point. Then we will have to find which NSEC3 record covers the hased name. That record will then be used as proof to show that the next closer does not exist.

7.2 Testing

To see if the implementation shows the correct behaviour we need to test it. This subsection will go into what tests are performed to validate the behaviour.

7.2.1 Infrastructure setup

To test adequately the NSD and simdzone software was used locally with a signed version of the zone file that can be found in appendix A. For testing, a signed version with NSEC is created, and a signed version with NSEC3. To perform the tests I will use Drill software from NLnet Labs [3].

7.2.2 Objectives

To see that the behaviour works, there are three main objectives for this test:

1. **Validation that the IDELEG records are returned**: When a delegating query is made the server should be able to find the incremental delegation point and add it to the right section. In addition, the correct owner is verified and the correct DNSSEC signatures are present. A query to `customer1.ideleg.net` will be made to check this.

2. **Validation that the correct NSEC records are returned**: When the incremental delegation point does not exist, the server should return 3 NSEC records to prove that this is indeed the case. The records will be verified to see if the NSEC records do indeed cover what is required. A query to `customer5.ideleg.net` will be made to test this.

[3] https://github.com/NLnetLabs/ldns/tree/features/ideleg

3. **Validation that the correct NSEC3 records are returned**: When the incremental delegation point does not exist, the server should return 3 NSEC3 records to prove that this is indeed the case. The records will be verified if all the covers are correct. A query to `customer5.ideleg.net` will be made to test this.

To verify the last objective, a list is created using ldns tools to see the NSEC3 hash and their corresponding domain name:

```
0dg6p6q668m6o06btohv5ln2v1kr0hhu. customer1.ideleg.net.
14kmp3a4dkn1akla5d8keadvck83984q. customer2.ideleg.net.
2bav9365ohr6hpfo8j47jaqfqpsnn985. supporting.ideleg.net.
5bt67l45m6kb3i8ec8qvdm63gtm3ju0h. customer5.ideleg.net.
68gih9vrb5437viu53uda68r96s6eelo. *._deleg.ideleg.net
a4f7b74ogt93gbguku097mjhc31kkb7t. ns.customer2.ideleg.net.
d0fnjlpq8avci25hdci4o7j83db3tknr. customer3._deleg.ideleg.net.
d9g85i5lbqmn1bt81sf0b4i943uv2o69. customer2._deleg.ideleg.net.
dli2ie0heh58o8tuld5bch1ffrj8g7ht. legacy-auth.ideleg.net.
gqjf4c525qhuv7p5m7s3a66d7r978pue. ideleg.net.
iuedbq94mkgvdesdrqts2krv3tjuc4b7. customer4._deleg.ideleg.net.
jr2r5b2u14b9om4s9i468pjoojfq1j5h. customer4.ideleg.net.
o4t84dns9aleliu8p73ro28b7rr47m2s. customer5._deleg.ideleg.net
qogoicks6ih63cpgpnq4ik5f6h2aeoj9. customer3.ideleg.net.
sp8jp025sf88j2ehjg5j5snjk0dmemfq. customer6._deleg.ideleg.net
uiud4j47bn2l4m3b1k7gf31rsp20r29o. _deleg.ideleg.net.
urdvtsdhi1olrabbu2fcjjpqmp13qrvn. customer1._deleg.ideleg.net.
```

8 Results

8.1 Requirements for the testbed

For fair and adequate testing, it is important that a testbed is created consisting of servers that support and not support extensible delegation. The domain `ideleg.net` was acquired by NLnet Labs for testing. The ideleg zone is served with two supporting nameservers The zone file can be found in appendix A. The two main servers will be reachable on `ideleg.net` and `ideleg.nlnetlabs.nl`. Then two more servers are created `supporting.ideleg.net` and `legacy-auth.ideleg.net` These servers have support and no support for extensible delegations, respectively. The setup can be seen in figure 3. All servers are reachable over IPv4 and IPv6.

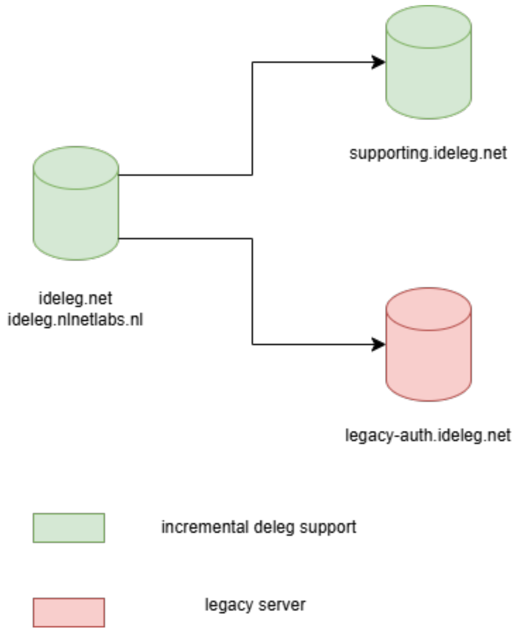

Figure 3: The `ideleg.net` setup

8.2 Authoritative nameserver

The results of the queries can be found in appendices B to D. This section will go briefly over the results linked to the tests given in section 7.2.2. Table 1 gives a overview of the results.

Test number	Result
1	pass
2	pass
3	pass

Table 1: Test results overview

8.2.1 Validation that the IDELEG records are returned

In appendix B the first query can be found, it has the correct IDELEG record with the correct owner. The DNSSEC signature for the records is also present. Therefore, this test passed.

8.2.2 Validation that the correct NSEC records are returned

In appendix C the second query can be seen. We see the following NSEC records that prove the following:

- Next closer, this NSEC record proves that `customer5._deleg.ideleg.net` does not exist, since it would have been before `customer1.ideleg.net` in canonical order.
- Closest encloser, this NSEC records shows that `_deleg.ideleg.net` exists.
- Wildcard cover, if the wildcard existed, it would have been between these names.

Since all the required proof was there, this test passes.

8.2.3 Validation that the correct NSEC3 records are returned

In appendix D the final query can be found. We used the table in section 7.2.2 to convert the names. The following NSEC3 records are present in order;

- The next closer (`customer5._deleg.ideleg.net` = `o4t84dns9aleliu8p73ro28b7rr47m2s`) should be between these two names if it existed; therefore, it shows that this name does not exist.
- The closest encloser (`_deleg.ideleg.net` = `uiud4j47bn2l4m3b1k7gf31rsp20r29o`), this records shows that the closest encloser exists.
- The wildcard (`*._deleg.ideleg.net` = `68gih9vrb5437viu53uda68r96s6eelo`), should be between these two names, therefore there is no wildcard.

Since all the required proofs were there, this test also passes.

9 Conclusion and Future Work

9.1 Discussion

Although the nameserver supports extensible delegation, not all aspects are fully functional yet, these can be found in section 9.3. The implementation made at this point works as it is almost performing an additional query alongside the one by the resolver. The implemented changes are almost all implementations that were already there. The only difference is that they are now working outside of normal behaviour. It might be easier to see if it is possible to take the extensible delegation as a starting point and work from there; this in contrary to adding a lot of functions to the already existing delegation function. Including of the incremental delegation in the answers reduces the amount of extra queries required by resolvers. Resolvers would need to start with one parallel query to determine whether the nameserver supports extensible delegation. Adding incremental delegation records to the answer would reduce the impact on performance as written in the research of Jesse van Zutphen [3].

The current setup for the testbed allows testing of all features. Since there are mixed supporting and nonsupporting servers, it will show that the proprosal is indeed incrementally deployable. However, to fully test this, the resolver side of the proposal should be made to work with the received data. Without that implementation, the testbed would be incomplete. The `ideleg.net` zone also serves legacy data so that legacy resolvers are still able to find the correct answer.

9.2 Conclusion

This research set out to investigate what is needed for an adequate evaluation of the proposal for incrementally deployable extensible delegation. This research was also created to realise an implementation in the authoritative nameserver software. This section will reflect on the research questions set in section 2. To start with the first question *What are the requirements of a testbed environment for fair and adequate evaluation of the proposal?*. To fairly and adequately evaluate the proposal, there would need to be at least an implementation in a resolver that can work with the newly proposed IDELEG RR type. In order to test the proposal, a testbed would be required consisting of mixed supporting and nonsupporting nameservers. This testbed infrastructure and the testbed domain `ideleg.net` zonefile are made. This will allow for a fair evaluation.

For the second question: *What behaviour optimisations can be done at the authoritative* *nameserver to achieve reduced (simultaneous) queries and how to implement them?*. The foremost optimisations that can be made is to include the incremental delegation data whenever a referral is queried. In order to show that the incremental delegation point does not exist, it should also send proof of this with the legacy answer to signal to the resolver that this point was searched and did not exist to prevent them from sending additional queries to the incremental delegation point.

9.3 Future Work

This research had the main focus on the authoritative nameserver side of the proposal from NLnet Labs . Although most of the cases are implemented, there are a few that still need to be done. This consists of expanding wildcards and generating NS record from incremental delegations if they do not exist. To make the proposal work, the resolver from NLnet Labs should also handle the new RR type, an implementation of the previous draft was made, but it should be updated to work with the IDELEG RR type. Now that the testbed architecture is created, they should be configured to serve data so that an resolver can follow the delegations given from the main nameservers.

It can also be useful if this draft is chosen and released to gather statistics to see how the adoption is going. Similar research was done on IPv6 [12].

References

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

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

[3] Jesse van Zutphen. *Extensible Delegations in DNS Recursive Resolvers.* July 2024. URL: https://nlnetlabs.nl/downloads/publications/extensible-deleg-in-resolvers_2024-07-08.pdf.

[4] Philip Homburg et al. *Incrementally Deployable Extensible Delegation for DNS.* Internet-Draft draft-homburg-deleg-incremental-deleg-01. Work in Progress. Internet Engineering Task Force, Jan. 2025. 26 pp. URL: https://datatracker.ietf.org/doc/draft-homburg-deleg-incremental-deleg/01/.

[5] Tim April et al. *Extensible Delegation for DNS.* Internet-Draft draft-wesplaap-deleg-01. Work in Progress. Internet Engineering Task Force, Oct. 2024. 19 pp. URL: https://datatracker.ietf.org/doc/draft-wesplaap-deleg/01/.

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

[7] Mark P. Andrews et al. *DNS Glue Requirements in Referral Responses.* RFC 9471. Sept. 2023. DOI: 10.17487/RFC9471. URL: https://www.rfc-editor.org/info/rfc9471.

[8] Raffaele Sommese and Moura et all. "When parents and children disagree: Diving into DNS delegation inconsistency." In: *Proceedings of the Passive and Active Measurement Workshop.* Eugene, OR, USA, 2020.

[9] Scott Rose et al. *DNS Security Introduction and Requirements.* RFC 4033. Mar. 2005. DOI: 10.17487/RFC4033. URL: https://www.rfc-editor.org/info/rfc4033.

[10] Internet Assigned Numbers Authority. *Service Name and Transport Protocol Port Number Registry.* https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=2. Accessed: 2025-01-23.

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

[12] Jakub Czyz et al. "Measuring IPv6 adoption". In: *SIGCOMM Comput. Commun. Rev.* 44.4 (Aug. 2014), pp. 87–98. ISSN: 0146-4833. DOI: 10.1145/2740070.2626295. URL: https://doi.org/10.1145/2740070.2626295.

A ideleg.net zone file

The zone file has been altered to fit the width of the page.

```
$ORIGIN ideleg.net.
$TTL 60
@                 SOA ideleg.net. wouter.petri.os3.nl. (
                                2025012270 ; serial
                                1800       ; refresh
                                900        ; retry
                                604800     ; expire
                                3600       ; minimum
                                )

                  NS     ideleg.net.
                  NS     ideleg.nlnetlabs.nl.
                  A      78.47.51.169
                  AAAA   2a01:4f8:c0c:92cd::1

_deleg            IDELEG 1 @ ipv4hint=78.47.51.169 ipv6hint=2a01:4f8:c0c:92cd::1
_deleg            IDELEG 2 ideleg.nlnetlabs.nl. ipv4hint=94.130.183.253
                         ipv6hint=2a01:4f8:1c0c:4864::1

supporting        A      188.245.247.219
                  AAAA   2a01:4f8:c2c:99d7::1
legacy-auth       A      94.130.76.72
                  AAAA   2a01:4f8:c2c:b1ed::1

;; IDELEG delegations
customer1._deleg IDELEG 10 supporting   ipv4hint=188.245.247.219 ipv6hint=2a01:4f8:c2c:99d7::1
customer2._deleg IDELEG 10 ns.customer2 ipv4hint=188.245.247.219 ipv6hint=2a01:4f8:c2c:99d7::1
customer3._deleg IDELEG 10 supporting   ipv4hint=188.245.247.219 ipv6hint=2a01:4f8:c2c:99d7::1
                 IDELEG 20 legacy-auth  ipv4hint=94.130.76.72    ipv6hint=2a01:4f8:c2c:b1ed::1
customer4._deleg IDELEG 0 ideleg.customer2

;; Legacy delegations
customer1         NS     supporting
customer2         NS     ns.customer2
ns.customer2      A 188.245.247.219
                  AAAA 2a01:4f8:c2c:99d7::1
customer3         NS     supporting
                  NS     legacy-auth
customer5         NS     legacy-auth

; Assuming ideleg.customer2.ideleg.net. IDELEG 10 ns.customer2.ideleg.net.
                         ipv4hint=188.245.247.219 ipv6hint=2a01:4f8:c2c:99d7::1
customer4         NS ns.customer2
```

B Query to `customer1.ideleg.net`

The results are trimmed and spaced to fit within the page

```
$ ./drill customer1.ideleg.net @localhost -p 8080 -D
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 51534
;; flags: qr rd ; QUERY: 1, ANSWER: 0, AUTHORITY: 5, ADDITIONAL: 4
;; QUESTION SECTION:
;; customer1.ideleg.net.        IN      A

;; ANSWER SECTION:

;; AUTHORITY SECTION:
customer1.ideleg.net.   60      IN      NS      supporting.ideleg.net.
customer1._deleg.ideleg.net.    60      IN      IDELEG  10 supporting.ideleg.net.
        ipv4hint=188.245.247.219 ipv6hint=2a01:4f8:c2c:99d7::1
customer1._deleg.ideleg.net.    60      IN      RRSIG   IDELEG 13 4 60 20250304144845 2025...
0dg6p6q668m6o06btohv5ln2v1kr0hhu.ideleg.net.    60      IN      NSEC3
        1 0 1 -  14kmp3a4dkn1akla5d8keadvck83984q NS
0dg6p6q668m6o06btohv5ln2v1kr0hhu.ideleg.net.    60      IN      RRSIG   NSEC3 13 3 60 2025...

;; ADDITIONAL SECTION:
supporting.ideleg.net.  60      IN      A       188.245.247.219
supporting.ideleg.net.  60      IN      RRSIG   A 13 3 60 20250304144845 20250204144845 63...
supporting.ideleg.net.  60      IN      AAAA    2a01:4f8:c2c:99d7::1
supporting.ideleg.net.  60      IN      RRSIG   AAAA 13 3 60 20250304144845 20250204144845 ...

;; Query time: 0 msec
;; EDNS: version 0; flags: do ; udp: 1232
;; SERVER: 127.0.0.1
;; WHEN: Tue Feb  4 22:56:45 2025
;; MSG SIZE  rcvd: 698
```

C Query to `customer5.ideleg.net` with NSEC records

The results are trimmed and spaced to fit within the page

```
$ ./drill customer5.ideleg.net @localhost -p 8080 -D
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 26382
;; flags: qr rd ; QUERY: 1, ANSWER: 0, AUTHORITY: 7, ADDITIONAL: 4
;; QUESTION SECTION:
;; customer5.ideleg.net.        IN      A

;; ANSWER SECTION:

;; AUTHORITY SECTION:
customer5.ideleg.net.   60      IN      NS      legacy-auth.ideleg.net.
customer4._deleg.ideleg.net.    60      IN      NSEC    customer1.ideleg.net. RRSIG NSEC IDELEG
customer4._deleg.ideleg.net.    60      IN      RRSIG   NSEC 13 4 60 20250304144839 2025020...
_deleg.ideleg.net.      60      IN      NSEC    customer1._deleg.ideleg.net. RRSIG NSEC IDELEG
_deleg.ideleg.net.      60      IN      RRSIG   NSEC 13 3 60 20250304144839 20250204144839 ...
customer5.ideleg.net.   60      IN      NSEC    legacy-auth.ideleg.net. NS RRSIG NSEC
customer5.ideleg.net.   60      IN      RRSIG   NSEC 13 3 60 20250304144839 20250204144839 ...

;; ADDITIONAL SECTION:
legacy-auth.ideleg.net. 60      IN      A       94.130.76.72
```

```
legacy-auth.ideleg.net. 60      IN      RRSIG   A 13 3 60 20250304144839 20250204144839 635...
legacy-auth.ideleg.net. 60      IN      AAAA    2a01:4f8:c2c:b1ed::1
legacy-auth.ideleg.net. 60      IN      RRSIG   AAAA 13 3 60 20250304144839 20250204144839 ...

;; Query time: 0 msec
;; EDNS: version 0; flags: do ; udp: 1232
;; SERVER: 127.0.0.1
;; WHEN: Tue Feb  4 15:48:54 2025
;; MSG SIZE  rcvd: 807
```

D Query to customer5.ideleg.net with NSEC3 records

The results are trimmed and spaced to fit within the page

```
$ ./drill customer5.ideleg.net @localhost -p 8080 -D
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 8690
;; flags: qr rd ; QUERY: 1, ANSWER: 0, AUTHORITY: 7, ADDITIONAL: 4
;; QUESTION SECTION:
;; customer5.ideleg.net.        IN      A

;; ANSWER SECTION:

;; AUTHORITY SECTION:
customer5.ideleg.net.   60      IN      NS      legacy-auth.ideleg.net.
jr2r5b2u14b9om4s9i468pjoojfq1j5h.ideleg.net.    60      IN      NSEC3
        1 0 1 -  qogoicks6ih63cpgpnq4ik5f6h2aeoj9 NS
jr2r5b2u14b9om4s9i468pjoojfq1j5h.ideleg.net.    60      IN      RRSIG   NSEC3 13 3 60 2025...
uiud4j47bn2l4m3b1k7gf31rsp20r29o.ideleg.net.    60      IN      NSEC3
        1 0 1 -  urdvtsdhi1olrabbu2fcjjpqmp13qrvn RRSIG IDELEG
uiud4j47bn2l4m3b1k7gf31rsp20r29o.ideleg.net.    60      IN      RRSIG   NSEC3 13 3 60 2025...
5bt67l45m6kb3i8ec8qvdm63gtm3ju0h.ideleg.net.    60      IN      NSEC3
        1 0 1 -  d0fnjlpq8avci25hdci4o7j83db3tknr NS
5bt67l45m6kb3i8ec8qvdm63gtm3ju0h.ideleg.net.    60      IN      RRSIG   NSEC3 13 3 60 2025...

;; ADDITIONAL SECTION:
legacy-auth.ideleg.net. 60      IN      A       94.130.76.72
legacy-auth.ideleg.net. 60      IN      RRSIG   A 13 3 60 20250304144845 20250204144845 63...
legacy-auth.ideleg.net. 60      IN      AAAA    2a01:4f8:c2c:b1ed::1
legacy-auth.ideleg.net. 60      IN      RRSIG   AAAA 13 3 60 20250304144845 20250204144845 ...

;; Query time: 0 msec
;; EDNS: version 0; flags: do ; udp: 1232
;; SERVER: 127.0.0.1
;; WHEN: Tue Feb  4 15:49:40 2025
;; MSG SIZE  rcvd: 879
```