DNS Operations T. Finch
Internet-Draft University of Cambridge
Intended status: Standards Track E. Hunt
Expires: January 9, 2020 ISC
P. van Dijk
PowerDNS
A. Eden
DNSimple
W. Mekking
ISC
July 8, 2019

Address-specific DNS aliases (ANAME)
draft-ietf-dnsop-aname-04

Abstract

This document defines the "ANAME" DNS RR type, to provide similar functionality to CNAME, but only for address queries. Unlike CNAME, an ANAME can coexist with other record types. The ANAME RR allows zone owners to make an apex domain name into an alias in a standards compliant manner.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

It can be desirable to provide web sites (and other services) at a bare domain name (such as "example.com") as well as a service-specific subdomain ("www.example.com").

If the web site is hosted by a third-party provider, the ideal way to provision its name in the DNS is using a CNAME record, so that the third party provider retains control over the mapping from names to IP address(es). It is now common for name-to-address mappings to be highly dynamic, dependent on client location, server load, etc.

However, CNAME records cannot coexist with other records with the same owner name. (The reason why is explored in Appendix B). This restriction means they cannot appear at a zone apex (such as "example.com") because of the SOA, NS, and other records that have to be present there. CNAME records can also conflict at subdomains, for example, if "department.example.edu" has separately hosted mail and web servers.

Redirecting website lookups to an alternate domain name via SRV or URI resource records would be an effective solution from the DNS point of view, but to date, browser vendors have not accepted this approach.

As a result, the only widely supported and standards-compliant way to publish a web site at a bare domain is to place address records (A and/or AAAA) at the zone apex. The flexibility afforded by CNAME is not available.

This document specifies a new RR type "ANAME", which provides similar functionality to CNAME, but only for address queries (i.e., for type A or AAAA). The basic idea is that the address records next to an ANAME record are automatically copied from and kept in sync with the ANAME target's address records. The ANAME record can be present at any DNS node, and can coexist with most other RR types, enabling it to be present at a zone apex, or any other name where the presence of other records prevents the use of a CNAME record.

Similar authoritative functionality has been implemented and deployed by a number of DNS software vendors and service providers, using names such as ALIAS, ANAME, apex CNAME, CNAME flattening, and top-level redirection. These mechanisms are proprietary, which hinders the ability of zone owners to have the same data served from multiple providers or to move from one provider to another. None of these proprietary implementations includes a mechanism for resolvers to follow the redirection chain themselves.

1.1. Overview

The core functionality of this mechanism allows zone administrators to start using ANAME records unilaterally, without requiring secondary servers or resolvers to be upgraded.

- The resource record definition in Section 2 is intended to provide zone data portability between standards-compliant DNS servers and the common core functionality of existing proprietary ANAME-like facilities.

- The zone maintenance mechanism described in Section 4 keeps the ANAME's sibling address records in sync with the ANAME target.

This definition is enough to be useful by itself. However, it can be less than optimal in certain situations: for instance, when the ANAME target uses clever tricks to provide different answers to different clients to improve latency or load balancing. The query processing rules in Section 6 require to include the ANAME record so that resolvers can use this information (as described in Section 5) to obtain answers that are tailored to the resolver rather than to the zone's primary master.

Resolver support for ANAME is not necessary, since ANAME-oblivious resolvers can get working answers from authoritative servers. It's just an optimization that can be rolled out incrementally, and that will help ANAME to work better the more widely it is deployed.

1.2. Terminology

An "address record" is a DNS resource record whose type is A or AAAA. These are referred to as "address types". "Address query" refers to a DNS query for any address type.

When talking about "address records" we mean the entire RRset, including owner name and TTL. We treat missing address records (i.e. NXDOMAIN or NODATA) the same successfully resolving as a set of zero address records, and distinct from "failure" which covers error responses such as SERVFAIL or REFUSED.

The "sibling address records" of an ANAME record are the address records at the same owner name as the ANAME, which are subject to ANAME substitution.

The "target address records" of an ANAME record are the address records obtained by resolving the ultimate target of the ANAME (see Section 3).

During the process of looking up the target address records, one or more CNAME or ANAME records may be encountered. These records are not the final target address records, and are referred in this document as "intermediate records". The target name must be replaced with the new name provided in the RDATA and the new target is resolved.

Other DNS-related terminology can be found in [RFC8499].

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [RFC2119].

2. The ANAME resource record

This document defines the "ANAME" DNS resource record type, with RR TYPE value [TBD].

2.1. Presentation and wire format

The ANAME presentation format is identical to that of CNAME [RFC1033]:

```
owner ttl class ANAME target
```

The wire format is also identical to CNAME [RFC1035], except that name compression is not permitted in ANAME RDATA, per [RFC3597].

2.2. Coexistence with other types

Only one ANAME <target> can be defined per <owner>. An ANAME RRset MUST NOT contain more than one resource record.

An ANAME's sibling address records are under the control of ANAME processing (see Section 4) and are not first-class records in their own right. They MAY exist in zone files, but they can subsequently be altered by ANAME processing.

An ANAME record MAY freely coexist at the same owner name with other RR types, except they MUST NOT coexist with CNAME or any other RR type that restricts the types with which it can itself coexist. That means An ANAME record can coexist at the same owner name with A and AAAA records. These are the sibling address records that are updated with the target addresses that are retrieved through the ANAME substitution process Section 3.

Like other types, An ANAME record can coexist with DNAME records at the same owner name; in fact, the two can be used cooperatively to redirect both the owner name address records (via ANAME) and everything under it (via DNAME).

3. Substituting ANAME sibling address records

This process is used by both primary masters (see Section 4) and resolvers (see Section 5), though they vary in how they apply the edit described in the final step. However, this process is not exclusively used by primary masters and resolvers: it may be executed as a bump in the wire, as part of the query lookup, or at any other point during query resolution.

The following steps MUST be performed for each address type:

1. Starting at the ANAME owner, follow the chain of ANAME and/or CNAME records as far as possible to find the ultimate target.

2. If a loop is detected, continue with an empty RRset, otherwise get the ultimate target's address records. (Ignore any sibling address records of intermediate ANAMEs.)

3. Stop if resolution failed. (Note that NXDOMAIN and NODATA count as successfully resolving an empty RRset.)

4. If one or more address records are found, replace the owner of the target address records with the owner of the ANAME record. Set the TTL to the minimum of the ANAME TTL, the TTL of each intermediate record, and the TTL of the target address records. Drop any RRSIG records.

5. Stop if this modified RRset is the same as the sibling RRset (ignoring any RRSIG records). The comparison MAY treat nearly-equal TTLs as the same.

6. Delete the sibling address RRset (if any) and replace it with the modified RRset.

At this point, the substituted RRset is not signed. A primary master will proceed to sign the substituted RRset, whereas resolvers can only use the substituted RRset when an unsigned answer is appropriate. This is explained in more detail in the following sections.

4. ANAME processing by primary masters

Each ANAME's sibling address records are kept up-to-date as if by the following process, for each address type:

- Perform ANAME sibling address record substitution as described in Section 3. Any edit performed in the final step is applied to the ANAME's zone. A primary server MAY use Dynamic Updates (DNS UPDATE) [RFC2136] to update the zone.

- If resolution failed, wait for a period before trying again. This retry time SHOULD be configurable.

- Otherwise, wait until the target address RRset TTL has expired or is close to expiring, then repeat.

It may be more efficient to manage the polling per ANAME target rather than per ANAME as specified (for example if the same ANAME target is used by multiple zones).

Sibling address records are committed to the zone and stored in nonvolatile storage. This allows a server to restart without delays due to ANAME processing, use offline DNSSEC signing, and not implement special ANAME processing logic when handling a DNS query.

Appendix D describes how ANAME would fit in different DNS architectures that use online signing or tailored responses.

4.1. Zone transfers

ANAME is no more special than any other RRtype and does not introduce any special processing related to zone transfers.

A zone containing ANAME records that point to frequently-changing targets will itself change frequently, and may see an increased number of zone transfers. Or if a very large number of zones are sharing the same ANAME target, and that changes address, that may cause a great volume of zone transfers. Guidance on dealing with ANAME in large scale implementations is provided Appendix D.

Secondary servers rely on zone transfers to obtain sibling address records, just like the rest of the zone, and serve them in the usual way (see Section 6). A working DNS NOTIFY [RFC1996] setup is recommended to avoid extra delays propagating updated sibling address records when they change.

4.2. DNSSEC

A zone containing ANAME records that will update address records has to do so before signing the zone with DNSSEC [RFC4033] [RFC4034] [RFC4035]. This means that for traditional DNSSEC signing the substitution of sibling address records must be done before signing and loading the zone into the name server. For servers that support online signing, the substitution may happen as part of the name server process, after loading the zone.

DNSSEC signatures on sibling address records are generated in the same way as for normal (dynamic) updates.

4.3. TTLs

Sibling address records are served from authoritative servers with a fixed TTL. Normally this TTL is expected to be the same as the target address records' TTL; however the exact mechanism for obtaining the target is unspecified, so cache effects, following ANAME and CNAME chains, or deliberate policies might make the sibling TTL smaller.

This means that when adding address records into the zone as a result of ANAME processing, the TTL to use is at most that of the TTL of the address target records. If you use a higher value, this will stretch the TTL which is undesired.

TTL stretching is hard to avoid when implementing ANAME substitution at the primary: The target address records' TTL influences the update rate of the zone, while the sibling address records' TTL determine how long a resolver may cache the address records. Thus, the end-to-end TTL (from the authoritative servers for the target address records to end-user DNS caches) is nearing twice the target address record TTL. There is a more extended discussion of TTL handling in Appendix C.

5. ANAME processing by resolvers

When a resolver makes an address query in the usual way, it might receive a response containing ANAME information in the Answer section, as described in Section 6. This informs the resolver that it MAY resolve the ANAME target address records to get answers that are tailored to the resolver rather than the ANAME's primary master.

In order to provide tailored answers to clients that are ANAME-oblivious, the resolver MAY perform sibling address record substitution in the following situations:

- The resolver's client queries with DO=0. (As discussed in Section 8, if the resolver finds it would downgrade a secure answer to insecure, it MAY choose not to substitute the sibling address records.)

- The resolver's client queries with DO=1 and the ANAME and sibling address records are unsigned. (Note that this situation does not apply when the records are signed but insecure: the resolver might not be able to validate them because of a broken chain of trust, but its client could have an extra trust anchor that does allow it to validate them; if the resolver substitutes the sibling address records they will become bogus.)

In these first two cases, the resolver MAY perform ANAME sibling address record substitution as described in Section 3. Any edit performed in the final step is applied to the Answer section of the response.

If the resolver's client is querying using an API such as "getaddrinfo" [RFC3493] that does not support DNSSEC validation, the resolver MAY perform ANAME sibling address record substitution as described in Section 3. Any edits performed in the final step are applied to the addresses returned by the API. (This case is for validating stub resolvers that query an upstream recursive server with DO=1, so they cannot rely on the recursive server to do ANAME substitution for them.)

6. Query processing

6.1. Authoritative servers

6.1.1. Address queries

When a server receives an address query for a name that has an ANAME record, the response's Answer section MUST contain the ANAME record, in addition to the sibling address queries. The ANAME record indicates to a client that it might wish to resolve the target address records itself.

6.1.2. ANAME queries

When a server receives an query for type ANAME, regardless of whether the ANAME record exists on the queried domain, any sibling address records SHOULD be added to the Additional section. Note that the sibling address records may have been substituted already.

When adding address records to the Additional section, if not all address types are present and the zone is signed, the server SHOULD include a DNSSEC proof of nonexistence for the missing address types.

6.2. Resolvers

6.2.1. Address queries

When a server receives an address query for a name that has an ANAME record, the response's Answer section MUST contain the ANAME record, in addition to the sibling address queries.

The Additional section MAY contain the target address records that match the query type (or the corresponding proof of nonexistence), if they are available in the cache and the target address RDATA fields differ from the sibling address RRset.

An ANAME target MAY resolve to address records via a chain of CNAME and/or ANAME records; any CNAME/ANAME chain MUST be included when adding target address records to a response's Additional section.

6.2.2. ANAME queries

When a resolver receives an query for type ANAME, any sibling address records SHOULD be added to the Additional section. Just like with an authoritative server, when adding address records to the Additional section, if not all address types are present and the zone is signed, the resolver SHOULD include a DNSSEC proof of nonexistence for the missing address types.

7. IANA considerations

IANA is requested to assign a DNS RR TYPE value for ANAME resource records under the "Resource Record (RR) TYPEs" subregistry under the "Domain Name System (DNS) Parameters" registry.

IANA might wish to consider the creation of a registry of address types; addition of new types to such a registry would then implicitly update this specification.

8. Security considerations

When a primary master updates an ANAME's sibling address records to match its target address records, it uses its own best information as to the correct answer. The primary master might sign the updated records, but that is not a guarantee of the actual correctness of the answer. This signing can have the effect of promoting an insecure response from the ANAME <target> to a signed response from the <owner>, which can then appear to clients to be more trustworthy than it should. DNSSEC validation SHOULD be used when resolving the ANAME <target> to mitigate this possible harm. Primary masters MAY refuse to substitute ANAME sibling address records unless the <target> node is both signed and validated.

When a resolver substitutes an ANAME's sibling address records, it can find that the sibling address records are secure but the target address records are insecure. Going ahead with the substitution will downgrade a secure answer to an insecure one. However this is likely to be the counterpart of the situation described in the previous paragraph, so the resolver is downgrading an answer that the ANAME's primary master upgraded. A resolver will only downgrade an answer in this way when its client is security-oblivious; however the client's path to the resolver is likely to be practically safer than the resolver's path to the ANAME target's servers. Resolvers MAY choose not to substitute sibling address records when they are more secure than the target address records.

9. Acknowledgments

Thanks to Mark Andrews, Ray Bellis, Stefan Buehler, Paul Ebersman, Richard Gibson, Tatuya JINMEI, Hakan Lindqvist, Mattijs Mekking, Stephen Morris, Bjorn Mott, Richard Salts, Mukund Sivaraman, Job Snijders, Jan Vcelak, Paul Vixie, Duane Wessels, and Paul Wouters, Olli Vanhoja, Brian Dickson for discussion and feedback.

10. Changes since the last revision

[This section is to be removed before publication as an RFC.]

The full history of this draft and its issue tracker can be found at https://github.com/each/draft-aname [1]

10.1. Version -04

o Split up section about Additional Section processing.

o Update Additional Section processing requirements.

o Clarify when ANAME resolution may happen [#43].

o Revisit TTL considerations [#30, #34].

o ANAME goes into the Answer section when QTYPE=A|AAAA [#62].

o Update alternative setups section with concerns (Brian Dickson) [#68].

o Add section on ANAME loops (open issue [#45]).

10.2. Version -03

o Grammar improvements (Olli Vanhoja)

o Split up Implications section, clarify text on zone transfers and dynamic updates [#39].

o Rewrite Alternative setup section and move to Appendix, add text on zone transfer scalibility concerns and GeoIP.

10.3. Version -02

Major revamp, so authoritative servers (other than primary masters) now do not do any special ANAME processing, just Additional section processing.

11. References

11.1. Normative References

[RFC1033] Lottor, M., "Domain Administrators Operations Guide", RFC 1033, DOI 10.17487/RFC1033, November 1987, <https://www.rfc-editor.org/info/rfc1033>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <https://www.rfc-editor.org/info/rfc3597>.

[RFC4033] 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>.

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

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

11.2. Informative References

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <https://www.rfc-editor.org/info/rfc882>.

[RFC0973] Mockapetris, P., "Domain system changes and observations", RFC 973, DOI 10.17487/RFC0973, January 1986, <https://www.rfc-editor.org/info/rfc973>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/info/rfc1996>.

[RFC2065] Eastlake 3rd, D. and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, DOI 10.17487/RFC2065, January 1997, <https://www.rfc-editor.org/info/rfc2065>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W.

Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <https://www.rfc-editor.org/info/rfc3493>.

11.3. URIs

[1] https://github.com/each/draft-aname

[2] https://github.com/each/draft-aname/issues/45

Appendix A. Implementation status

PowerDNS currently implements a similar authoritative-only feature using "ALIAS" records, which are expanded by the primary server and transfered as address records to secondaries.

[TODO: Add discussion of DNSimple, DNS Made Easy, EasyDNS, Cloudflare, Amazon, Dyn, and Akamai.]

Appendix B. Historical note

In the early DNS [RFC0882], CNAME records were allowed to coexist with other records. However this led to coherency problems: if a resolver had no cache entries for a given name, it would resolve queries for un-cached records at that name in the usual way; once it had cached a CNAME record for a name, it would resolve queries for un-cached records using CNAME target instead.

For example, given the zone contents below, the original CNAME behaviour meant that if you asked for "alias.example.com TXT" first, you would get the answer "owner", but if you asked for "alias.example.com A" then "alias.example.com TXT" you would get the answer "target".

```
alias.example.com.      TXT    "owner"
alias.example.com.      CNAME  canonical.example.com.
canonical.example.com.  TXT    "target"
canonical.example.com.  A      192.0.2.1
```

This coherency problem was fixed in [RFC0973] which introduced the inconvenient rule that a CNAME acts as an alias for all other RR types at a name, which prevents the coexistence of CNAME with other records.

A better fix might have been to improve the cache's awareness of which records do and do not coexist with a CNAME record. However that would have required a negative cache mechanism which was not added to the DNS until later [RFC1034] [RFC2308].

While [RFC2065] relaxed the restriction by allowing coexistence of CNAME with DNSSEC records, this exception is still not applicable to other resource records. RRSIG and NSEC exist to prove the integrity of the CNAME record; they are not intended to associate arbitrary data with the domain name. DNSSEC records avoid interoperability problems by being largely invisible to security-oblivious resolvers.

Now that the DNS has negative caching, it is tempting to amend the algorithm for resolving with CNAME records to allow them to coexist with other types. Although an amended resolver will be compatible with the rest of the DNS, it will not be of much practical use because authoritative servers which rely on coexisting CNAMEs will not interoperate well with older resolvers. Practical experiments show that the problems are particularly acute when CNAME and MX try to coexist.

Appendix C. On preserving TTLs

An ANAME's sibling address records are in an unusual situation: they are authoritative data in the owner's zone, so from that point of view the owner has the last say over what their TTL should be; on the other hand, ANAMEs are supposed to act as aliases, in which case the target should control the address record TTLs.

However there are some technical constraints that make it difficult to preserve the target address record TTLs.

The following subsections conclude that the end-to-end TTL (from the authoritative servers for the target address records to end-user DNS caches) is nearing twice the target address record TTL.

C.1. Query bunching

If the times of end-user queries for a domain name are well distributed, then (typically) queries received by the authoritative servers for that domain are also well distributed. If the domain is popular, a recursive server will re-query for it once every TTL seconds, but the periodic queries from all the various recursive servers will not be aligned, so the queries remain well distributed.

However, imagine that the TTLs of an ANAME's sibling address records are decremented in the same way as cache entries in recursive servers. Then all the recursive servers querying for the name would try to refresh their caches at the same time when the TTL reaches zero. They would become synchronized, and all the queries for the domain would be bunched into periodic spikes.

This specification says that ANAME sibling address records have a normal fixed TTL derived from (e.g. equal or nearly equal to) the target address records' original TTL. There is no cache-like decrementing TTL, so there is no bunching of queries.

C.2. Upstream caches

There are two straightforward ways to get an RRset's original TTL:

o by directly querying an authoritative server;

o using the original TTL field from the RRset's RRGIG record(s).

However, not all zones are signed, and a primary master might not be able to query other authoritative servers directly (e.g. if it is a hidden primary behind a strict firewall). Instead it might have to obtain an ANAME's target address records via some other recursive server.

Querying via a separate recursive server means the primary master cannot trivially obtain the target address records' original TTLs. Fortunately this is likely to be a self-correcting problem for similar reasons to the query-bunching discussed in the previous subsection. The primary master can inspect the target address records just after the TTL expires when its upstream cache has just refreshed them, so the TTL will be nearly equal to the original TTL.

A related consideration is that the primary master cannot in general refresh its copies of an ANAME's target address records more frequently than their TTL, without privileged control over its resolver cache.

Combined with the requirement that sibling address records are served with a fixed TTL, this means that the end-to-end TTL will be the target address record TTL (which determines when the sibling address records are updated) plus the sibling address record TTL (which determines when end-user caches are updated). Since the sibling address record TTL is derived from the target address records' original TTL, the end-to-end TTL will be nearing twice the target address record TTL.

C.3. ANAME chains

ANAME sibling address record substitution is made slightly more complicated by the requirement to follow chains of ANAME and/or CNAME records. The TTL of the substituted address records is the minimum of TTLs of the ANAME, all the intermediate records, and target records. This stops the end-to-end TTL from being inflated by each ANAME in the chain.

With CNAME records, repeat queries for "cname.example. CNAME target.example." must not be fully answered from cache after its TTL expires, but must instead be sent to name servers authoritative for "cname.example" in case the CNAME has been updated or removed. Similarly, an ANAME at "aname.example" means that repeat queries for "aname.example" must not be fully answered from cache after its TTL expire, but must instead be sent to name servers authoritative for aname.example in case the ANAME has been updated or removed.

C.4. ANAME substitution inside the name server

When ANAME substitution is performed inside the authoritative name server (as described in #alternatives) or in the resolver (as described in #resolver) the end-to-end TTL will actually be just the target address record TTL.

An authoritative server that has control over its resolver can use a cached target address RRset and decremented TTL in the response to the client rather than using the original target address records' TTL. It SHOULD however not use TTLs in the response that are nearing zero to avoid query bunching Appendix C.1.

A resolver that performs ANAME substitution is able to get the original TTL from the authoritative name server and use its own cache to store the substituted address records with the appropriate TTL, thereby honoring the TTL of target address records.

C.5. TTLs and zone transfers

When things are working properly (with secondary name servers responding to NOTIFY messages promptly) the authoritative servers will follow changes to ANAME target address records according to their TTLs. As a result the end-to-end TTL is unchanged from the previous subsection.

If NOTIFY doesn't work, the TTLs can be stretched by the zone's SOA refresh timer. More serious breakage can stretch them up to the zone expiry time.

Appendix D. Alternative setups

If you are a large scale DNS provider, ANAME may introduce some operational concerns.

D.1. Reducing query volume

When doing ANAME target lookups, an authoritative server might want to use longer TTLs to reduce query volume, for ANAME values that do not change frequently. This is the same concern a recursive resolver may be exposed to when receiving answers with short TTLs. An authoritative server doing ANAME target lookups therefor could use the same mitigation as a recursive nameserver, that is set a configured minimum TTL usage. This may however contribute to TTL stretching as described in Section 4.3 so the configured minimum should not be too low.

D.2. Zone transfer scalability

A frequently changing ANAME target, or a ANAME target that changes its address and is used for many zones, can lead to an increased number of zone transfers. Such DNS architectures may want to consider a zone transfer mechanism outside the DNS.

Another way to deal with zone transfer scalability is to move the ANAME processing (Section 3) inside the name server daemon. This is not a requirement for ANAME to work, but may be a better solution in large scale implementations. These implementations usually already rely on online DNSSEC signing for similar reasons. If ANAME processing occurs inside the name server daemon, it MUST be done before any DNSSEC online signing happens.

For example, some existing ANAME-like implementations are based on a DNS server architecture, in which a zone's published authoritative servers all perform the duties of a primary master in a distributed manner: provisioning records from a non-DNS back-end store, refreshing DNSSEC signatures, and so forth. They don't use standard standard zone transfers, and already implement their ANAME-like processing inside the name server daemon, substituting ANAME sibling address records on demand.

D.3. Tailored responses

Some DNS providers will tailor responses based on information in the client request. Such implementations will use the source IP address or EDNS Client Subnet [RFC7871] information and use geographical data (GeoIP) or network latency measurements to decide what the best answer is for a given query. Such setups won't work with traditional DNSSEC and provide DNSSEC support usually through online signing. Similar such setups should provide ANAME support through substituting ANAME sibling records on demand.

Also, an authoritative server that uses the client address to tailor the response should obviously not use its own address when looking up ANAME targets, or it could direct clients to a suboptimal server (e.g. a wrong language, or regional restricted content). Instead the authoritative server should look up the ANAME targets on behalf of the client address. It could use for example EDNS Client Subnet for this.

In short, the exact mechanism for obtaining the target address records in such setups is unspecified; typically they will be resolved in the DNS in the usual way, but if an ANAME implementation has special knowledge of the target it can short-cut the substitution process, or it can use clever tricks such as client-dependant answers to make the answer more optimal.

Appendix E. ANAME loops

The ANAME sibling address substitution algorithm in Section 3 poses a challenge of detecting a loop between two or more ANAME records. Imagine this setup: two authoritative servers X and Y performing ANAME sibling address substition on the fly (i.e. they attempt to resolve the ANAME target when the client query arrives). If server X gets a query for FOO.TEST which is an ANAME to BAR.TEST, it will send a query to server Y for BAR.TEST which is an ANAME to FOO.TEST. Server Y will then start a new query to server X, which has no way to know that it is regarding the original FOO.TEST lookup.

The only indicator of the presence of the loop in the described setup is the network timeout. Ideally we would recognize the loop explicitly based on the exchanged DNS messages.

On-the-fly ANAME substitution is allowed and it's just the most obvious scenario where the problem can be demonstrated, but this loop can also be encountered in other situations. The root cause is that when the server gets a query it doesn't know why and that the server always attempts to fully resolve the ANAME target before sending the response.

TODO: Solve this issue [https://github.com/each/draft-aname/issues/45 [2]]

Authors' Addresses

Tony Finch
University of Cambridge
University Information Services
Roger Needham Building
7 JJ Thomson Avenue
Cambridge CB3 0RB
England

Email: dot@dotat.at

Evan Hunt
ISC
950 Charter St
Redwood City, CA 94063
USA

Email: each@isc.org

Peter van Dijk
PowerDNS.COM B.V.
Den Haag
The Netherlands

Email: peter.van.dijk@powerdns.com

Anthony Eden
DNSimple
Boston, MA USA

Email: anthony.eden@dnsimple.com
URI: https://dnsimple.com/

Matthijs Mekking
ISC
950 Charter St
Redwood City, CA 94063
USA

Email: matthijs@isc.org