Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 15, 2011

S. Woolf
Internet Systems Consortium, Inc.
X. Lee
J. Yao
CNNIC

March 14, 2011

Problem Statement: DNS Resolution of Aliased Names
draft-ietf-dnsext-aliasing-requirements-01.txt

Abstract

This document attempts to describe a set of issues that arises from the desire to treat a set or group of names as "aliases" of each other, "bundled," "variants," or "the same," which is problematic in terms of corresponding behavior for DNS labels and FQDNs.

With the emergence of internationalized domain names, among other potential use cases, two or more names that users will regard as having identical meaning may sometimes require corresponding behavior in the underlying infrastructure, possibly in the DNS itself. It's not clear how to accommodate this required behavior of such names in DNS resolution; in particular, it's not clear when they are best accommodated in registry practices for generating names for lookup in the DNS, existing DNS protocol elements and behavior, existing application-layer mechanisms and practices, or some set of protocol elements or behavior not yet defined. This document attempts to describe some of these cases and the behavior of some of the possible solutions discussed to date.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on September 15, 2011.

Copyright Notice

Table of Contents

1. Introduction

As the Internet and the DNS have evolved beyond their original realms of use, a set of needs and expectations has appeared about how DNS labels behave that is informed significantly by common human assumptions about how "names" or "words" work. One aspect of this is the notion or expectation that multiple sets of names may be similar to a human user, and expected to behave "the same" or as "aliases" of one another, across multiple services and interactions. The DNS was designed with the implicit expectation that names would be based on ASCII characters, and the "similarity" or "sameness" property doesn't seem to arise terribly often in the names people originally wanted to use in the DNS; thus the requirements of identical resolution of "aliased" or "bundled" names hasn't figured prominently as an attribute that needed to be accommodated in the generation or lookup of DNS names. However, with the standardization of internationalized domain names protocols (ref: IDNA and IDNAbis), more and more internationalized domain name labels [RFC3490] are appearing in DNS zones. In some cases, these labels [RFC3743] are accompanied by the expectation that they are "equivalent" or should behave "the same," often because these labels are derived from names or strings that users consider "the same" in some languages. Accordingly, Internet users hope for such labels to behave in DNS contexts as they expect the corresponding human constructs to behave, regardless of the specific service (smtp, http, etc.) involved..

The general issues of what "the same" means, or of defining "variants" in human scripts as codified in Unicode (or anywhere else) are well outside the scope of the DNS or the expertise of most of the people who work on it. They are matters for philosophers and linguists, and for applications developers, respectively. However, to the extent that these issues can be specified as involving the resolution of names in the DNS, it's reasonable to describe those expectations and attempt to accommodate them.

There is some existing technology defined in the DNS for behavior that can be described as one name behaving "the same" as another. For a single node in the DNS tree, CNAME can be used to map one name as an "alias" to another, "canonical" name. If there is a need to map a subtree of the DNS-- a zone, or a domain and its subdomains-- to another domain, DNAME has been defined to allow this behavior. However, there is no way currently defined to do both, as CNAME is required to be the sole record at its node in the tree. Behavior that combines the characteristics of CNAME and DNAME is not currently defined in the DNS.

If existing protocol does not meet the zone administrator's need to be able to treat one label, name, or zone as "the same" as another,

there are also administrative mechanisms available for manipulating databases underlying the generation and resolution of DNS names. Registry operators have many mechanisms for working around DNS protocol in order to get behavior they want for names in DNS zones, and management of "aliases" is no exception. However, it is not clear how much of the user and operator requirements for "aliases" can be met by mechanisms for provisioning DNS zones, at acceptable cost. Concerns have been raised about this approach particularly at large scales and there is a need both to provision possibly exponential numbers of domains and then to audit them for compliance with parent registry policy.

1.1. What this document does

Attempts to think about "aliases" or similar concepts as applied to the DNS have been difficult, both because use cases have been unclear and because terminology for describing and distinguishing them has not been readily available. This document attempts to provide both brief descriptions of identified use cases, and a rough organization for how to think about behavior in the DNS that might correspond to the requirements derived from them as a way of evaluating proposed solutions. This includes existing and additional possible solutions, from the perspective of both DNS (authoritative server, resolver, and client) and application needs.

As a departure point, we attempt to be rigorous about distinguishing DNS "labels" from "words" (a human construct) and "strings" (which we use here as machine-readable constructs that nonetheless may not conform to DNS label constraints, such as IDNA U-labels). The distinctions among what humans type or see, what applications use, and what DNS stores and resolves are sometimes subtle but particularly important.

A list of broad requirements is proposed for any DNS protocol changes that might be undertaken.

We also review existing technologies (CNAME, DNAME) and proposed new ones ("BNAME," "zone clones") against the proposed requirements.

1.2. What this document does not do

This document makes no attempt to solve or even describe "translation" of one name into another in the DNS, which is likely to be impossible. "Translation" in general, or even the particular problem of determining when or why two DNS labels (or even FQDNs) should be considered "the same", is simply not in scope for the DNS protocol. We pre-suppose those decisions are made elsewhere and that the DNS needs to deliver behavior in conformance with that external

decision. In particular, we're talking about creating a property or association among a set of DNS names as "sameness" or "alias-of", but the correspondence between that set and any set of human- or application-visible strings is created outside of the DNS database and protocol. Language Variant Tables (RFC3743) serve as guides for registration policy, but the associations they create are basically expressed only as policy about what can be registered, not visible to DNS resolvers, applications, or users.

Accordingly, this document makes no comment on policy regarding when two names are "the same," what restrictions should be placed on their generation or use outside those imposed by the DNS protocol, or the ability of one approach over another to instantiate what a given user regards as "the same" for a language, script, culture, community, application, encoding, or purpose.

1.3. Terminology

All the basic terms used in this specification are defined in the documents [RFC1034], [RFC1035], [RFC2672] and [RFC3490].

We also note that there is a wide variety of terminology in use to describe the issues we attempt to treat in this document, and no consensus on which apply under what conditions. Terms for "a set of domain names that somehow need to be treated as similar" include "bundle," "variants," or "clones". As uniformity of terms is one of the goals of any work on this topic in the DNS, we try not to add to the confusion in the problem statement but can't claim to have finalized a recommendation in early versions of this document.

2. Problem Statement

From the point of view of the DNS, a number of attributes suggest themselves as important dimensions for evaluating what "the same" might mean.

One question is exactly what it is that's to be defined as "the same"? Are the end results to be identical, and if so from what perspective: that of the recursive resolver? The application? The human consumer of content? Is it enough that lookups on the FQDN portion of an email address result in the same A or AAAA records, or does some intermediate mapping need to be maintained between MX records in the resolution chain? What about the FQDN portion of a URL handed back to an application, or in resolution processes that include multiple lookups of records that may include FQDNs? Do there need to be general rules specified for the handling of FQDNs in RDATA of present and future RRtypes?

Another question is the behavior of multiple names with respect to one another: is it enough to define one as "canonical" or "preferred," with the others considered as "variants" that are transformed to the "preferred" form? Or is there a real need for multiple names to be "equivalent", interchangeable, with none considered "preferred" over the others? (We note here that no requirement for complete interchangeability or identity has been articulated, except anecdotally, and such equivalence would be extremely difficult to define in the DNS.)

In addition, the tree structure of the DNS requires that we consider the behavior of "identical" names across multiple zones in the hierarchy. Are mappings to be maintained in names more than a level, or two, deep? If so, with what characteristics, and what characteristics are required for scalability?

A further question arises with respect to how applications should interact with alias-specific DNS behavior. A basic requirement would seem to be "First, do no harm," or in other words, any extensions to DNS protocol in support of the desired "alias" behavior should not interfere with applications that expect to do such interpretation on their own. This concern is based in the expectation that DNS is simple and predictable, operating strictly as infrastructure under the process of creating "the user experience," not as part of it.

A key point in evaluating these questions is that DNS is a lookup-based protocol. A DNS name either resolves, or not. There’s no search function in the DNS, no fuzzy match. It provides lookup on a specified name. This is critically important because it means that any work to define any kind of "sameness" that can’t be expressed as a lookup, such as selection among a set of candidate names for which to return results, must be done elsewhere than in the DNS.

2.1. Registration of Domain Name Variants

To some degree, issues of "sameness" or creating an association among a set of names have existed around the use of domain names from the beginning. Points where the behavior of DNS labels have collided with expectations around the behavior of words have included DNS handling of case sensitivity, the kind of transformations a human expects to "just work" around "try-this.example" vs. "trythis.example", and continuing frustration that "confusingly similar" names can be delegated to different parties by DNS registries. However, the introduction of IDN has provided a forcing function in that it has added visibility for a wider variety of issues along these lines, and possibly the urgency of dealing with them for large numbers of users.

A need has been identified in connection with the introduction of IDN for defining how "variants" might behave as DNS names. Specifically defining "variant" is a matter for experts, but it's generally conceded that recognition and careful management of cases where multiple names are associated together as "variants" in the expectation or preference of users are important; without such management of grouped domain names, security risks may be increased and the quality of user experience may be compromised. [RFC3743] developed by JET (Joint Engineering Team) gives one possible solution of how to manage registration of a domain name, intended to be applied to the script and usage common across Chinese, Japanese, and Korean users. [RFC3743] proposed an algorithm which will allocate a group of names, consisting of a domain name and its variants, to the same domain holder. It means that the domain holder will get control of the domain name and its variants. [RFC4290] suggests the practice in [RFC3743] to be used in registrations of internationalized domain names. But [RFC3743] and [RFC4290] do not define how, exactly, these bundles of names are to be treated by the registry or the DNS in order to obtain the desired "identical" behavior. [RFC4690] said that the "variant" model introduced in [RFC3743] and [RFC4290] can be used by a registry to prevent the most negative consequences of possible confusion, by ensuring either that both names are registered to the same party in a given domain or that one of them is completely prohibited. The principles described in [RFC3743], [RFC4290] and [RFC4690] have been accepted by many registries. But the technical details of how to guarantee that a bundle of domain names are "identical" in the DNS remain unspecified.

2.2. Identical DNS Resolution for Bundled DNS Names

To some extent, the desired behavior can be described: "identical DNS resolution" means that the process of resolving two domain names will end with the same result, in most cases the same IP address. In the history of DNS protocol development, there have been two attempts to specify such "identical resolution" behavior:CNAME[RFC1034] which maps or redirects itself, and DNAME[RFC2672] which maps or redirects its descendants. In the case of bundles or groups of names, however, some operators have asserted they need identical DNS resolution at all levels' domain names, including the domain name itself and its descendants. As alluded to above, registries are left with ad hoc provisioning and database management mechanisms for managing variant names, with some help from existing DNS protocol mechanisms for mapping labels or FQDNs to each other. However, some are finding the existing mechanisms to have unsatisfactory limitations; they are seeking more guidance on the use of existing mechanisms, and perhaps the addition of new ones in the DNS protocol.

2.3. Character Variants

Many defined scripts as used in many different languages have "character variants" included. There is no uniform definition of variants, and in fact their characteristics differ widely, but it's possible to define some. For example, the definition of variant characters in the JET Guidelines [RFC3743], intended for use with the CJK language/script communities, is roughly this: One conceptual character can be identified with several different code points in character sets for computer use. In UNICODE definitions of some scripts, including Han (chinese), some characters can be identified as "compatibility variants" of another character, which usually implies that the first can be remapped to the second without the loss of any meaning. In this document, variant characters are two or more characters that may be similar in appearance or identical in meaning (similarity in appearance is not required by the definition but often occurs).

With the introduction of IDNs in the DNS, perhaps most prominently in the root zone, decisions about how to deal with IDN variants is a significant challenge ahead of us. We describe here a couple of examples, Chinese and Greek; comparable situations exist in Arabic, Cyrillic, and others.

2.3.1. An example: Simplified and Traditional Chinese

For example, the IDN TLD "China"(U+4E2D U+56FD) and its variant (U+4E2D U+570B) are in the root today. The first one (U+4E2D U+56FD) can be considered the "original" IDN TLD and the second one (U+4E2D U+570B) can be considered the IDN TLD "variant". Ideally, it should be possible to treat the original IDN TLD and its IDN TLD variant as "identical" for purposes of DNS resolution, in a way similar to the case mapping most DNS users take for granted. However, this analogy is a bit perilous, and turns out to be hard to use as a guide to what behavior is actually desirable, not least because it's not fully consistent even within the DNS.

At this writing, four Han script IDN TLDs are in the root, including two pairs comprising a Traditional Chinese name and its Simplified counterpart. These operators will, in an ideal world, be able to share some operational experience around implementation of registry policy regarding managing multiple DNS trees as "the same"

2.3.2. An example: Greek

In Greek, almost every word has the "tonos" accent sign, but where it is placed (on which character) can vary. Further, some words end in a final sigma, which is represented differently to sigma appearing

elsewhere in the word. If a registry wishes to be able to enforce the association among all of the domain names that correspond to a "word" in Greek, with all its possible Unicode strings, some mechanism must be used to enumerate the "variant" names and tie them together. This makes sense from the human factors perspective, as depending on how the user types something, results may include a different domain to what was expected, although the user may have the firm belief that "the same word" was input in multiple cases.

As an example, the domain names "xn--0xadhj4a.gr" and "xn--0xaafjl.gr" appear to a native speaker/reader of Greek to represent "the same word," in a sense very much like the case insensitivity that native users of Latin script take for granted in the DNS.

2.3.3. An Example: Arabic

[STW: [to be added]

2.4. Use of Variants

It’s reasonable to pose the question at this point, without necessarily being able to answer it yet, of what is the ideal or intended impact of solving the issue identified so far for registries on applications and end users. Ultimately, simplifying the provisioning side may result in the same semantics as we have today for zones maintained in parallel but for less work. However, we later assert a proposed requirement that synthesizing the same record as a query would have obtained from an enumerated parallel tree isn’t enough-- that the property of association or "sameness" we’re creating with specific mechanisms needs to be useful in some specific way to the consumer of the data.

The trigger for raising the questions discussed here is based in user expectations that one name, in certain circumstances to be determined and somehow encoded by humans, can be treated as interchangeable with another with regards to a particular context or activity, with resolution of domain names as part of the context or activity. However, it’s useful to note here that satisfying that set of user expectations may or may not reasonably be done in the DNS, wholly or in part.

There are two arguments for placing functionality that links one name to another as "the same" in the DNS. Their validity is not yet determined, but they amount to:

1. The expectation that two or more names be "the same" is often expressed as a desire to register the associated names as a "bundle" or otherwise link them as domain names. This is because the domain name in a URL or email address is often presented to

the user as semantically meaningful, based on strings used to derive DNS labels-- proper names, "words," etc. This brings such concerns to the attention of providers of domain names, and suggests at least exploring how to answer the question near where it's asked-- i.e. the registry.

2. The desire for names on the Internet to act like words is often service-independent; users want to be able to use identical strings in the course of invoking multiple services that seem to be related, such as going to a webpage and then sending email to an address in "the same" domain (probably an FQDN). It's been noted that people are very comfortable with a certain amount of fuzziness about "alternative spellings" and assorted other variations within the notion of "sameness", but they nonetheless often want such an association to exist. In cases where a set of variant strings is parseable in the application, has corresponding A-labels that can be looked up in the DNS, etc. but only a subset can be typed on the user's input device or rendered on the user's screen, the association may be necessary to the successful completion of the activity the user is attempting. This argues in turn for some mechanism that is not dependent on the specific service, protocol, or application involved, since leaving it up to service-specific mechanisms, or conventions in the use of DNS records or other mechanisms not really intended for the purpose, leads to confusion and inconsistency.

3. Operational Considerations

Any change to a mature infrastructure protocol such as DNS needs to be informed by consideration of the tradeoffs among providing the associated service, using the service, and possibly conflicting means of offering comparable functionality. In the case of DNS, this requires that we look at provisioning (populating zones and the mechanics of authority servers responding to queries), service by intermediate and client resolvers, and related capabilities provided with existing facilities in the DNS and in applications.

3.1. Zone Provisioning and Authority Servers

The initial motivation for discussion of support for aliases in the DNS was provided by operators of top-level domain (TLD) registries. The problem facing them lies in scaling the provision of "bundles" of names that users expect to be treated "the same", as in the examples previously described.

3.1.1. Provisioning of ’aliases’ in the registry

The most obvious way to provision multiple names as "the same" is to delegate each separately, and then maintain the contents of the delegated zones together, from the same backend database or by some similar mechanism. This has the advantage of requiring no new technology; it can be done, and is done today, entirely with provisioning logic and registry policy.

However, it doubles the work and the number of records required. If provisioning isn’t done carefully, errors can arise, leaving inconsistencies. And provisioning multiple trees does nothing to link the resulting names directly; there is no property of "association" or isomorphism created in the DNS that corresponds to user or application expectations for "sameness". There is no way to tell, from resolving a name in one tree, that it’s part of a set or bundle of related names.

Separate provisioning also poses a limitation for some registry operators in that there is no way to verify that the trees are being maintained in parallel without exhaustively walking the zones, which may be large or nested to multiple levels. In the case, for example, of a zone A.B.C.example.com, in which each of A,B,C are derived from strings with a single character variant each, eight zones must be maintained in parallel and possibly available for audit by the authority over example.com, depending on its delegation policy.

3.1.2. Impact of special mechanisms

Once we begin to consider mechanisms for maintaining parallel zones or "aliases", we need to look at how the "alias" or association property is created and where the burden of maintaining it lies. In the case of proposed mechanisms, we attempt to describe them below.

Existing mechanisms besides the simple, straightforward provisioning of zones that are identical except for the ownernames of corresponding records include wildcards, CNAME, and DNAME. See below, but here we note that they require special processing by the authority server in order to synthesize responses that are supposed to be the equivalent of simply providing the parallel zones by one-to-one enumeration.

3.2. Recursive Resolvers

Another area where it’s necessary to review requirements and impacts of changes to the DNS is in resolver expectations and behavior, given that recursive resolvers do most of the work of getting the data out of DNS that provisioning activities put into it.

In practice, much of the work of special processing falls to resolvers. In particular, any scheme that can result in multiple queries and some need to chain the answers or disambiguate multiple answers is going to make more work for resolvers, and is going to need to specify that work in careful detail. Any ambiguity or lack of precision in specifying the use of "aliases" will propagate back to applications, and quite possibly leave applications writers and users worse off than they were without DNS mechanisms intended to "help".

Ideally any new RRtypes defined to support "aliases" would be provisioned on the authority server and require no special processing, which would make them transparent to intermediate resolvers. However, depending on how much such RRs and their processing need to be visible to the application to be effective, this may not be possible.

3.3. Applications

The most complex part of the analysis of costs and benefits of defining new technology for support of "aliases" by DNS is in determining what applications would do with such new mechanisms and how it would help to have them. In particular, it is critically important not to simply provide additional complexity, even in the name of making provisioning on the server side easier, unless there's some clear benefit to it for the ultimate client of the DNS as well--the user who is trying to "do something" on the Internet.

Such a clear benefit could come from the ability, alluded to above, to provide a facility that was anchored in the DNS and so did not have to be re-invented anew for each application or protocol that wished to have user-transparent access to the ability to reduce "aliases" to a canonical domain name without necessarily being aware, a priori, that the name was part of a set that could be deemed "the same".

An example used more than once in discussion is provided by SSL, as a protocol that uses domain names without necessarily using the DNS protocol per se. SSL certificates are tied to one domain name. It would be helpful to applications to have a non-protocol-specific way to identify securely cases where multiple domain names can be canonicalized to the domain name used for an SSL certificate. Currently HTTP has such an ability, but it's considered awkward to use and does not help writers or users of other application protocols.

An important characteristic of such a solution for applications, however, is that the writer and user be able to tell when such a

mechanism was invoked in the DNS, to avoid interference among multiple possible ways to find "aliases" and compare them. This in turn implies a fair amount of complexity to be inflicted not only on DNS protocol but also on API/library writers seeking to use such new facilities, particularly given the caution above that DNS is a lookup protocol and must be given precise sequences of bits to look up.

Another characteristic of an "aliases" mechanism of interest to application writers is the difficulty, and therefore the likely speed and breadth of deployment, of such a DNS-based mechanism for canonicalizing aliased names. DNS is notorious, as an aging infrastructure protocol, for the long tail of deployment of significant protocol features. Again, a feature can be designed to be fairly easy to deploy, but without an incentive such as faster application development or more secure applications, it still may not see wide uptake even after it’s present in current code bases.

An additional assumption often made needs to be examined here, as well: that applications, and applications writers, have straightforward, well-defined ways of interacting with the DNS, into which new functionality can obviously and straightforwardly be added. This can be true, as in the case of a query for a specific RRtype at an unambiguously determined name by an application designed to find and take advantage of the data represented in records of that RRtype. However, it does not have to be, and often isn’t: there’s no standard DNS API, and no standard abstraction for applications to interact with the DNS. There is confusion about how to use DNS effectively, and over the actual behavior of existing "alias" mechanisms such as CNAME. Adding new technology is likely to be accompanied by challenges not only in getting it deployed to the installed base, but in getting its uses clearly documented for applications writers.

4. Proposed Requirements

These observations and examples, along with general discussion to date, lead to the following tentative set of actual requirements.

1. Any mechanism proposed in the DNS to support "aliases" or multiple names as "the same" MUST be workable for DNSSEC- signed zones.
2. Any mechanism proposed in the DNS to support "aliases" or multiple names as "the same" MUST be "backwards compatible," in that it MUST NOT change the established behavior of existing RRtypes and query processing.
3. Any mechanism proposed SHOULD NOT require more overhead of registries, authoritative servers, or clients than existing mechanisms for approximating the desired behavior, such as

provisioning of multiple parallel trees or CNAME processing. If a new solution is more work than existing mechanisms, imperfect as they may be, it's not clear where the incentives would lie to deploy it. This is particularly a concern for implementors and application developers.

4. Any mechanism proposed MAY require new RRtypes and special processing for them.
5. Any mechanism proposed MUST NOT only reduce costs of generating and providing authoritative service for DNS zones. It would be too easy to reduce costs on the authority server provider while adding costs elsewhere, particularly in terms of complexity. Given the central importance of DNS service to Internet operations, any change undertaken to lower the cost to providers may be useful, but should not simply shift costs to DNS users, whether applications or end users.

5. Possible Solutions

Currently, there are several possible mechanisms to support identical DNS resolution of "bundled" or "variant" names as "aliases" in the DNS. Existing mechanisms in the DNS include CNAME and DNAME. In addition, as described briefly above, registry operators have a great many techniques for applying policy to what names can be registered, and provisioning technology to how they are instantiated in the DNS, in support of keeping "variant" names behaving similarly to each other, or in preventing the use of such variants as might be considered confusing or dangerous.

In addition, there are new proposals for DNS protocol to support "aliases" in the DNS as part of the desired behavior of "variant" names: Names direction[BNAME], and "Zone clone".

All of the solutions have their advantages and disadvantages. In particular, there are a couple of limitations they all share. Every mechanism existing or proposed to support "aliases" in the DNS requires that one name be designated as the "canonical" name ("preferred" in the terminology of the JET variant mechanism) and any others bundled with it are to be considered "variants" or "aliases". The only known way to enforce a symmetrical or equivalent association is via careful registry provisioning within and across domains. In addition, the different "alias" mechanisms differ in subtle ways that have to be carefully reviewed against the desired behavior of the DNS in support of different types of "variants".

5.1. Mapping or Redirection of Domain Names

5.1.1. Mapping itself (CNAME)

It was recognized as part of the original specification of the DNS that a host can have many names; in fact this expectation predates the DNS, referring to the earlier specification of host names. In the simplest case for "aliases", Internet users need these multiple names to be resolved to the same IP address by a DNS server. The CNAME record [RFC1034], where "CNAME" is an abbreviation for "Canonical Name", is a way to designate aliases of the "real" or canonical name of a host. In some cases, CNAME can be used to produce the necessary association a bundle of variant domain names. But the CNAME only maps itself, not its descendants; in fact it is defined to not have descendants, as it is the only name at a node in the DNS tree and can’t exist at the same name as delegation. In the case of IDN variants, however, it is often desirable that the name map both itself and its descendants.

In terms, however, of deployment and availability, it’s useful to note that CNAME is already part of the installed base of DNS authority servers and intermediate mode resolvers. Using it for this purpose requires description of how to do it and how it behaves, but that already is available. There are no issues of uptake or backwards compatibility or new code or new documentation.

5.1.2. Mapping its descendants

In order to maintain the address-to-name mappings in a context of network renumbering, a DNAME record or Delegation Name record defined by [RFC2672] was invented to create an alias for all its subdomains. In contrast, the CNAME record creates an alias only of a single name (and not of its subdomains). As with the CNAME record, the DNS lookup will continue by retrying the lookup with the new name. If a DNS resolver sends a query without EDNS[EDNS0], or with EDNS version 0, then a name server synthesizes a CNAME record to simulate the semantics of the DNAME record. A DNAME record is very much like the CNAME record, but while the CNAME record only applies for one name, with a DNAME record one can create aliases for all the records for its subdomain.

DNAME is can be considered slightly less widely deployed than CNAME for the EDNS0 compatibility reason described above, but it’s been defined in the DNS for quite some time, and includes a backwards compatibility mechanism in the CNAME synthesis just described, so use of DNAME does not rely on deployment of resolver code capable of special processing for DNAME; it relies entirely on authority server capability.

5.1.3. Mapping itself and its descendants

Bundling of "variant" strings or names as domain names, possibly along with other use cases not yet identified, require the ability to map a whole tree of the domain space to another domain. The current DNS protocols do not support this function. A new DNS resource record [BNAME] has been proposed to deal with this problem.

The advantage of BNAME is that it would enable a class of "aliasing" behavior that some operators find desirable, particularly in preference to some of the provisioning overhead they describe having to deploy to support potentially large numbers of "bundles" of variants at multiple levels of the DNS tree. The disadvantage is that it may not provide the behavior people really want while requiring the time and resources to code and deploy any new DNS facility.

Alternatively, a proposal has been made that would leave CNAME as already specified, but eliminating the constraint that a CNAME must be alone at a node in the DNS tree. This would avoid any coding and deployment overhead associated with new RRtypes, while obtaining the desired behavior. Concerns expressed about it, however, include the possible (but not yet specified) effort required for backwards compatibility to avoid harm to implementations that expect, and use, the old behavior.

Both of these mechanisms would require both authority server and resolver changes to enable the new capability.

5.2. Zone Clone

The proposal of "zone clone" or "dns shadow", is an alternative solution for a higher level of support than the DNS currently provides for "alias" behavior across zones. In this scheme, a new RRtype, SHADOW, is specified; it can exist at a zone apex and can be used to define "clones" or "shadows" of the zone content so that records in the zone are reachable via lookups from multiple delegations. This mechanism varies fundamentally from CNAME/DNAME/BNAME in that it creates a local copy on each cooperating authoritative server that has the original zone, reachable by the names specified in the SHADOW RR. Its scope, then, is the zone as maintained by an authoritative server rather than a single RRset (even one corresponding to a delegation).

This scheme has the advantage that it allows a SHADOW zone to be used in all the same contexts as the canonical or underlying zone, including contexts where a CNAME or DNAME (or, presumably, a BNAME) cannot appear, such as in the RDATA of certain RRtypes. Of the

proposed DNS protocol mechanisms, it probably comes closest to the behavior some have requested as "equivalence," where none of the bundled or SHADOW names is canonical or preferred over the others. It does implicate an unknown level of effort to implement and support.

6. IANA Considerations

There are no obvious IANA considerations in this memo; we reiterate that the determination of which names are to be considered "the same" is explicitly out of scope.

7. Security Considerations

[STW: Looking for examples for this section.]

Unsolved issues that will have to be considered in the definition of what "the same" means for the DNS include the implications for DNSSEC, and whether "identical" resolution includes DNSSEC validation in the expected "identical" behavior.

Another area of possible peril includes SSL certificates, "Host" headers as seen by web servers, and other security-relevant data often associated with domain names. It will have to be considered whether, and how, the "sameness" property maps into the expected behavior of security-related protocols that use domain names, particularly given that it's unlikely that all operators will ever use the same set of constructs (whether in the DNS or elsewhere) to signal whether different "names" are "the same" for purposes of the function of a particular application or protocol.

In addition, there is a large cluster of security risks at the user and application levels that motivate significant portions of the interest in what it means to treat a set of names as "aliases" of each other. One set of issues is around the expectation that two strings are seen as "different" by the user in some obvious way (such as visually) but need to be treated as "the same". The potential for user confusion and subversion is not hard to imagine in cases where two visually distinct strings are nonetheless likely to be expected by the user to behave "the same" in some functional way. This is the case we have attempted to address here.

There is a separate but complementary set of issues that arise around cases where strings that look "the same" should nonetheless be treated as different-- the so-called "confusing visual similarity" problem. The easy example is substituting the Unicode codepoint for

a character in one script, or a string of them, for the Unicode codepoints for similar-looking characters in an altogether different script. This has a different set of potential risks to users, and has not been discussed here. It's often closely related to the "alias" issue we have attempted to deal with, however, which poses risks of its own to analysis of the either subject.

8. Acknowledgements

Most of the ideas here and much of the text is taken from discussions on the DNSEXT and DNSOP WG mailing lists. Particular help is acknowledged from the authors of the proposed solutions drafts, and from the many contributors to the IDNAbis work and its underpinnings. Special thanks at the intersection of DNS and IDNAbis is owed to Patrik Faltstrom, Cary Karp, John Klensin, Vaggelis Segredakis, and Andrew Sullivan for their patient explanations.

9. Change History

[[anchor28: RFC Editor: Please remove this section.]]

9.1. draft-yao-dnsext-identical-resolution: Version 00

- Domain Name Identical Resolution Problem Statement (initial attempt)

9.2. draft-yao-dnsext-identical-resolution: Version 01

- Expanded introduction
- Added Greek example
- Added some detail to descriptions of proposed solutions

10. References

10.1. Normative References

[ASCII] American National Standards Institute (formerly United States of America Standards Institute), "USA Code for Information Interchange", ANSI X3.4-1968, 1968.

[EDNS0] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2672] Crawford, M., "Non-Terminal DNS Name Redirection", RFC 2672, August 1999.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, September 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

[RFC3743] Konishi, K., Huang, K., Qian, H., and Y. Ko, "Joint Engineering Team (JET) Guidelines for Internationalized Domain Names (IDN) Registration and Administration for Chinese, Japanese, and Korean", RFC 3743, April 2004.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4290] Klensin, J., "Suggested Practices for Registration of Internationalized Domain Names (IDN)", RFC 4290, December 2005.

[RFC4690] Klensin, J., Faltstrom, P., Karp, C., and IAB, "Review and Recommendations for Internationalized Domain Names (IDNs)", RFC 4690, September 2006.

10.2. Informative References

[BNAME] Yao, J., Lee, X., and P. Vixie, "Bundle DNS Name Redirection", draft-yao-dnsext-bname-01.txt (work in progress), 12 2009.

[CNAME-DNAME] Sury, O., "CNAME+DNAME Name Redirection", draft-sury-dnsext-cname-dname-00.txt (work in progress), 4 2010.

[IDN-TLD-Variants] Yao, J. and X. Lee, "IDN TLD Variants Implementation Guideline", draft-yao-dnsop-idntld-implementation-01.txt (work in progress), 11 2009.

[RFC2672bis] Rose, S. and W. Wijngaards, "Update to DNAME Redirection in the DNS", Internet-Draft ietf-dnsext-rfc2672bis-dname-17.txt, 6 2009.

[SHADOW] Vixie, P., "Use of DNS to Carry Configuration Metadata Concerning Automatic Replication of Zones", draft-vixie-dnsext-dnsshadow-00.txt (work in progress), 2 2010.

Authors' Addresses

Suzanne Woolf
Internet Systems Consortium, Inc.
950 Charter St.
Redwood City, CA 94063

Phone: +1 650 423 1333
Email: woolf@isc.org

Xiaodong LEE
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing

Phone: +86 10 58813020
Email: lee@cnnic.cn

Jiankang YAO
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing

Phone: +86 10 58813007
Email: yaojk@cnnic.cn

DNSEXT Working Group P. Vixie, ISC
INTERNET-DRAFT R. Joffe, Centergate
<draft-vixie-dnsext-resimprove-00.txt> F. Neves, Registro
Intended Status: For Your Information June 22, 2010

Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness

Status of this Memo

This Internet-Draft will expire on December 31, 2010.

Abstract

This document describes several mechanisms which can be employed by iterative caching DNS resolvers to improve resiliency, robustness, and responsiveness. These improvements are optional and they require no changes to the protocol, or to authority servers, or to DNS stub resolver clients.

1. Introduction

1.1. Iterative caching DNS resolvers can be both compliant and interoperable without also being optimal. Indeed a wide range of optimizations, mechanisms, and outright "tricks" can be employed without affecting correctness or interopability. Such optimizations could be recommended but never required.

1.2. This document describes several practices which can improve the resilience, robustness, and responsiveness of iterative caching DNS resolvers. These practices are not required for correctness or interoperability, and no other DNS protocol agent need be modified to gain the prospective benefits of implementing these practices.

1.3. Three practices are described here:

A. Revalidating a delegation when a parent NS RRset TTL expires.
B. Stopping a downward cache search when an NXDOMAIN is encountered.
C. Upgrading the credibility of NS RRsets upon delegation events.

These practices are described in detail in later sections of this document. While they are described together this single document for editorial convenience, and are known to work well together, they are in no way interdependent.

2. Delegation Revalidation Upon NS RRSet Expiry

2.1. Because the delegating NS RRset at the bottom of the parent zone and the apex NS RRset in the child zone are unsynchronized, the TTL of the parent’s delegating NS RRset is meaningless. A child zone’s apex NS RRset is authoritative and thus has a higher cache credibility than the parent’s delegating NS RRset, so, the NS RRset "below the cut" immediately replaces the parent’s delegating NS RRset in cache when an iterative caching DNS resolver crosses a zone cut.

2.2. The lowest TTL found in a parent zone’s delegating NS RRset should be stored in the cache and used to trigger delegation

revalidation as follows. Whenever a cached RRset is being considered for use in a response, the cache should be walked upward toward the root, looking for expired delegations. At the first expired delegation encountered while walking upward toward the root, revalidation should be triggered, putting the processing of dependent queries on hold until validation is complete.

2.3. To revalidate a delegation, the iterative caching DNS resolver will forward the query that triggered revalidation to the nameservers at the closest enclosing zone cut above the revalidation point. While searching for these nameservers, additional revalidations may occur, perhaps placing an entire chain of dependent queries on hold, unwinding in downward order as revalidations closer to the root must be complete before revalidations further from the root can begin.

2.4. If a delegation can be revalidated at the same node, then the old apex NS RRset should be deleted from cache and then the new delegating NS RRset should be stored in cache. The minimum TTL from the new delegating NS RRset should also be stored in cache to facilitate future revalidations. This order of operations ensures that the RRset credibility rules do not prevent the new delegating NS RRset from entering the cache. It is expected that the child’s apex NS RRset will rapidly replace the parent’s delegating NS RRset as soon as iteration restarts after the revalidation event.

2.5. If the new delegating NS RRset cannot be found (RCODE=NXDOMAIN) or if there is a new zone cut at some different level of the hierarchy (insertion or deletion of a delegation point above the revalidation point) or if the new RRset shares no nameserver names in common with the old one (indicating some kind of redelegation, which is rare) then the cache should be purged of all names and RRsets at or below the revalidation point. This facilitates redelegation or revocation of a zone by a parent zone administrator, and also conserves cache storage by deleting unreachable data.

2.6. To make the timing of a revalidation event unpredictable from the point of view of a potential cache-spoof attacker, the parent’s delegating NS RRset TTL should be reduced by a random fraction of its value before being stored for use in revalidation activities.

3. Stopping Downward Cache Search on NXDOMAIN

3.1. In virtually all existing resolvers, a cached NXDOMAIN is not considered "proof" that there can be no child domains underneath. This is due to an ambiguity in RFC 1034 that failed to distinguish

empty nonterminal domain names from nonexistent names. For DNSSEC, the IETF had to distinguish this case, but the implication on non-DNSSEC resolvers wasn't fully realized.

3.2. When searching downward in its cache, an iterative caching DNS resolver should stop searching if it encounters a cached NXDOMAIN. The response to the triggering query should be NXDOMAIN.

3.3. When an iterative caching DNS resolver stores an NXDOMAIN in its cache, all names and RRsets at or below that node should be deleted since they will have become unreachable.

3.4. By implication, a stream of queries FOO.TLD, BAR.FOO.TLD where FOO.TLD does not exist would normally cause both queries to be forwarded to TLD's nameservers. Following this recommended practice, the second query and indeed any other query for names at or below FOO.TLD would not be forwarded.

4. Upgrading NS RRset Credibility Upon Delegaton Events

4.1. Noting that a parent's delegating NS RRset is nonauthoritative "glue" whereas a child's apex NS RRset is authoritative, the latter will replace the former in cache whenever the latter is encountered. However, it is not mandatory for the child zone's nameservers to include the apex NS RRset in responses, thus it is possible for an iterative caching DNS resolver to never learn the authoritative NS RRset for a zone.

4.2. When a delegation response is received during iteration, a validation query should be sent in parallel with the forwarding of the triggering query to the delegated nameservers for the newly discovered zone cut. The response to the triggering query should be delayed until both the forwarded query and the validation query have been answered.

4.3. A validation query consists of a query for the child's apex NS RRset, sent to the newly discovered delegation's nameservers. Normal iterative logic applies to the processing of responses to validation queries, including storing the results in cache, propagating NXDOMAIN back to the triggering query, trying the next server on SERVFAIL or timeout, and so on.

4.4. If there are no nameserver names in common between the child's apex NS RRset and the parent's delegation NS RRset, then the responses received from forwarding the triggering query to the

parent's delegated nameservers should be discarded after validation, and this query should be forwarded again to the child's apex nameservers.

5. Security Considerations

5.1. A successful cache exploit which inserted a fake NXDOMAIN can deny more service from an iterative caching DNS resolver that implements the recommendation to stop downward searches when a cached NXDOMAIN is encountered.

5.2. A perfectly timed cache exploit which inserted a fake NS RRset during a delegation validation event could cause one fewer good response to be heard (per TTL expiry interval) from an iterative caching DNS resolver that implements the recommendation to validate delegations.

IANA Considerations

None.

Normative References

[RFC1035] P. Mockapetris, "Domain Names - Implementation and Specification," RFC 1035, USC/Information Sciences Institute, November 1987.

Authors' Addresses

Paul Vixie

Internet Systems Consortium
950 Charter Street
Redwood City, CA, USA
EMail: vixie@isc.org

Rodney Joffe

Centergate Research Group, LLC
420 S Smith Rd
Tempe, AZ 85281 USA
EMail: rjoffe@centergate.com

Frederico A. C. Neves

NIC.br / Registro.br
Av. das Nacoes Unidas, 11541, 7
Sao Paulo, SP 04578-000 BR
EMail: fneves@registro.br