                 Tradeoffs in Domain Name System (DNS) Support
                    for Internet Protocol version 6 (IPv6)

Status of this Memo

Copyright Notice

Abstract

   The IETF has two different proposals on the table for how to do DNS
   support for IPv6, and has thus far failed to reach a clear consensus
   on which approach is better. This note attempts to examine the pros
   and cons of each approach, in the hope of clarifying the debate so
   that we can reach closure and move on.

Introduction

   RFC 1886 [RFC1886] specified straightforward mechanisms to support
   IPv6 addresses in the DNS. These mechanisms closely resemble the
   mechanisms used to support IPv4, with a minor improvement to the
   reverse mapping mechanism based on experience with CIDR. RFC 1886 is
   currently listed as a Proposed Standard.

   RFC 2874 [RFC2874] specified enhanced mechanisms to support IPv6
   addresses in the DNS. These mechanisms provide new features that
   make it possible for an IPv6 address stored in the DNS to be broken
   up into multiple DNS resource records in ways that can reflect the
   network topology underlying the address, thus making it possible for
   the data stored in the DNS to reflect certain kinds of network
   topology changes or routing architectures that are either impossible
   or more difficult to represent without these mechanisms. RFC 2874 is
   also currently listed as a Proposed Standard.

Both of these Proposed Standards were the output of the IPNG Working
Group. Both have been implemented, although implementation of
[RFC1886] is more widespread, both because it was specified earlier
and because it's simpler to implement.

There's little question that the mechanisms proposed in [RFC2874] are
more general than the mechanisms proposed in [RFC1886], and that
these enhanced mechanisms might be valuable if IPv6's evolution goes
in certain directions. The questions are whether we really need the
more general mechanism, what new usage problems might come along with
the enhanced mechanisms, and what effect all this will have on IPv6
deployment.

The one thing on which there does seem to be widespread agreement is
that we should make up our minds about all this Real Soon Now.

Main Advantages of Going with A6

While the A6 RR proposed in [RFC2874] is very general and provides a
superset of the functionality provided by the AAAA RR in [RFC1886],
many of the features of A6 can also be implemented with AAAA RRs via
preprocessing during zone file generation.

There is one specific area where A6 RRs provide something that cannot
be provided using AAAA RRs: A6 RRs can represent addresses in which a
prefix portion of the address can change without any action (or
perhaps even knowledge) by the parties controlling the DNS zone
containing the terminal portion (least significant bits) of the
address. This includes both so-called "rapid renumbering" scenarios
(where an entire network's prefix may change very quickly) and
routing architectures such as the former "GSE" proposal [GSE] (where
the "routing goop" portion of an address may be subject to change
without warning). A6 RRs do not completely remove the need to update
leaf zones during all renumbering events (for example, changing ISPs
would usually require a change to the upward delegation pointer), but
careful use of A6 RRs could keep the number of RRs that need to
change during such an event to a minimum.

Note that constructing AAAA RRs via preprocessing during zone file
generation requires exactly the sort of information that A6 RRs store
in the DNS. This begs the question of where the hypothetical
preprocessor obtains that information if it's not getting it from the
DNS.

Note also that the A6 RR, when restricted to its zero-length-prefix
form ("A6 0"), is semantically equivalent to an AAAA RR (with one
"wasted" octet in the wire representation), so anything that can be
done with an AAAA RR can also be done with an A6 RR.

Main Advantages of Going with AAAA

   The AAAA RR proposed in [RFC1886], while providing only a subset of
   the functionality provided by the A6 RR proposed in [RFC2874], has
   two main points to recommend it:

   - AAAA RRs are essentially identical (other than their length) to
     IPv4's A RRs, so we have more than 15 years of experience to help
     us predict the usage patterns, failure scenarios and so forth
     associated with AAAA RRs.

   - The AAAA RR is "optimized for read", in the sense that, by storing
     a complete address rather than making the resolver fetch the
     address in pieces, it minimizes the effort involved in fetching
     addresses from the DNS (at the expense of increasing the effort
     involved in injecting new data into the DNS).

Less Compelling Arguments in Favor of A6

   Since the A6 RR allows a zone administrator to write zone files whose
   description of addresses maps to the underlying network topology, A6
   RRs can be construed as a "better" way of representing addresses than
   AAAA. This may well be a useful capability, but in and of itself
   it's more of an argument for better tools for zone administrators to
   use when constructing zone files than a justification for changing
   the resolution protocol used on the wire.

Less Compelling Arguments in Favor of AAAA

   Some of the pressure to go with AAAA instead of A6 appears to be
   based on the wider deployment of AAAA. Since it is possible to
   construct transition tools (see discussion of AAAA synthesis, later
   in this note), this does not appear to be a compelling argument if A6
   provides features that we really need.

   Another argument in favor of AAAA RRs over A6 RRs appears to be that
   the A6 RR's advanced capabilities increase the number of ways in
   which a zone administrator could build a non-working configuration.
   While operational issues are certainly important, this is more of
   argument that we need better tools for zone administrators than it is
   a justification for turning away from A6 if A6 provides features that
   we really need.

Potential Problems with A6

   The enhanced capabilities of the A6 RR, while interesting, are not in
   themselves justification for choosing A6 if we don't really need
   those capabilities. The A6 RR is "optimized for write", in the sense
   that, by making it possible to store fragmented IPv6 addresses in the
   DNS, it makes it possible to reduce the effort that it takes to
   inject new data into the DNS (at the expense of increasing the effort
   involved in fetching data from the DNS). This may be justified if we
   expect the effort involved in maintaining AAAA-style DNS entries to
   be prohibitive, but in general, we expect the DNS data to be read
   more frequently than it is written, so we need to evaluate this
   particular tradeoff very carefully.

   There are also several potential issues with A6 RRs that stem
   directly from the feature that makes them different from AAAA RRs:
   the ability to build up address via chaining.

   Resolving a chain of A6 RRs involves resolving a series of what are
   almost independent queries, but not quite. Each of these sub-queries
   takes some non-zero amount of time, unless the answer happens to be
   in the resolver's local cache already. Assuming that resolving an
   AAAA RR takes time T as a baseline, we can guess that, on the
   average, it will take something approaching time N*T to resolve an
   N-link chain of A6 RRs, although we would expect to see a fairly good
   caching factor for the A6 fragments representing the more significant
   bits of an address. This leaves us with two choices, neither of
   which is very good: we can decrease the amount of time that the
   resolver is willing to wait for each fragment, or we can increase the
   amount of time that a resolver is willing to wait before returning
   failure to a client. What little data we have on this subject
   suggests that users are already impatient with the length of time it
   takes to resolve A RRs in the IPv4 Internet, which suggests that they
   are not likely to be patient with significantly longer delays in the
   IPv6 Internet. At the same time, terminating queries prematurely is
   both a waste of resources and another source of user frustration.
   Thus, we are forced to conclude that indiscriminate use of long A6
   chains is likely to lead to problems.

   To make matters worse, the places where A6 RRs are likely to be most
   critical for rapid renumbering or GSE-like routing are situations
   where the prefix name field in the A6 RR points to a target that is
   not only outside the DNS zone containing the A6 RR, but is
   administered by a different organization (for example, in the case of
   an end user's site, the prefix name will most likely point to a name
   belonging to an ISP that provides connectivity for the site). While
   pointers out of zone are not a problem per se, pointers to other
   organizations are somewhat more difficult to maintain and less

susceptible to automation than pointers within a single organization
would be. Experience both with glue RRs and with PTR RRs in the IN-
ADDR.ARPA tree suggests that many zone administrators do not really
understand how to set up and maintain these pointers properly, and we
have no particular reason to believe that these zone administrators
will do a better job with A6 chains than they do today. To be fair,
however, the alternative case of building AAAA RRs via preprocessing
before loading zones has many of the same problems; at best, one can
claim that using AAAA RRs for this purpose would allow DNS clients to
get the wrong answer somewhat more efficiently than with A6 RRs.

Finally, assuming near total ignorance of how likely a query is to
fail, the probability of failure with an N-link A6 chain would appear
to be roughly proportional to N, since each of the queries involved
in resolving an A6 chain would have the same probability of failure
as a single AAAA query. Note again that this comment applies to
failures in the the process of resolving a query, not to the data
obtained via that process. Arguably, in an ideal world, A6 RRs would
increase the probability of the answer a client (finally) gets being
right, assuming that nothing goes wrong in the query process, but we
have no real idea how to quantify that assumption at this point even
to the hand-wavey extent used elsewhere in this note.

One potential problem that has been raised in the past regarding A6
RRs turns out not to be a serious issue. The A6 design includes the
possibility of there being more than one A6 RR matching the prefix
name portion of a leaf A6 RR. That is, an A6 chain may not be a
simple linked list, it may in fact be a tree, where each branch
represents a possible prefix. Some critics of A6 have been concerned
that this will lead to a wild expansion of queries, but this turns
out not to be a problem if a resolver simply follows the "bounded
work per query" rule described in RFC 1034 (page 35). That rule
applies to all work resulting from attempts to process a query,
regardless of whether it's a simple query, a CNAME chain, an A6 tree,
or an infinite loop. The client may not get back a useful answer in
cases where the zone has been configured badly, but a proper
implementation should not produce a query explosion as a result of
processing even the most perverse A6 tree, chain, or loop.

Interactions with DNSSEC

   One of the areas where AAAA and A6 RRs differ is in the precise
   details of how they interact with DNSSEC. The following comments
   apply only to non-zero-prefix A6 RRs (A6 0 RRs, once again, are
   semantically equivalent to AAAA RRs).

Other things being equal, the time it takes to re-sign all of the
addresses in a zone after a renumbering event is longer with AAAA RRs
than with A6 RRs (because each address record has to be re-signed
rather than just signing a common prefix A6 RR and a few A6 0 RRs
associated with the zone's name servers). Note, however, that in
general this does not present a serious scaling problem, because the
re-signing is performed in the leaf zones.

Other things being equal, there's more work involved in verifying the
signatures received back for A6 RRs, because each address fragment
has a separate associated signature. Similarly, a DNS message
containing a set of A6 address fragments and their associated
signatures will be larger than the equivalent packet with a single
AAAA (or A6 0) and a single associated signature.

Since AAAA RRs cannot really represent rapid renumbering or GSE-style
routing scenarios very well, it should not be surprising that DNSSEC
signatures of AAAA RRs are also somewhat problematic. In cases where
the AAAA RRs would have to be changing very quickly to keep up with
prefix changes, the time required to re-sign the AAAA RRs may be
prohibitive.

Empirical testing by Bill Sommerfeld [Sommerfeld] suggests that
333MHz Celeron laptop with 128KB L2 cache and 64MB RAM running the
BIND-9 dnssec-signzone program under NetBSD can generate roughly 40
1024-bit RSA signatures per second. Extrapolating from this,
assuming one A RR, one AAAA RR, and one NXT RR per host, this
suggests that it would take this laptop a few hours to sign a zone
listing 10^{5} hosts, or about a day to sign a zone listing 10^{6}
hosts using AAAA RRs.

This suggests that the additional effort of re-signing a large zone
full of AAAA RRs during a re-numbering event, while noticeable, is
only likely to be prohibitive in the rapid renumbering case where
AAAA RRs don't work well anyway.

Interactions with Dynamic Update

DNS dynamic update appears to work equally well for AAAA or A6 RRs,
with one minor exception: with A6 RRs, the dynamic update client
needs to know the prefix length and prefix name. At present, no
mechanism exists to inform a dynamic update client of these values,
but presumably such a mechanism could be provided via an extension to
DHCP, or some other equivalent could be devised.

Transition from AAAA to A6 Via AAAA Synthesis

   While AAAA is at present more widely deployed than A6, it is possible
   to transition from AAAA-aware DNS software to A6-aware DNS software.
   A rough plan for this was presented at IETF-50 in Minneapolis and has
   been discussed on the ipng mailing list. So if the IETF concludes
   that A6's enhanced capabilities are necessary, it should be possible
   to transition from AAAA to A6.

   The details of this transition have been left to a separate document,
   but the general idea is that the resolver that is performing
   iterative resolution on behalf of a DNS client program could
   synthesize AAAA RRs representing the result of performing the
   equivalent A6 queries. Note that in this case it is not possible to
   generate an equivalent DNSSEC signature for the AAAA RR, so clients
   that care about performing DNSSEC validation for themselves would
   have to issue A6 queries directly rather than relying on AAAA
   synthesis.

Bitlabels

   While the differences between AAAA and A6 RRs have generated most of
   the discussion to date, there are also two proposed mechanisms for
   building the reverse mapping tree (the IPv6 equivalent of IPv4's IN-
   ADDR.ARPA tree).

   [RFC1886] proposes a mechanism very similar to the IN-ADDR.ARPA
   mechanism used for IPv4 addresses: the RR name is the hexadecimal
   representation of the IPv6 address, reversed and concatenated with a
   well-known suffix, broken up with a dot between each hexadecimal
   digit. The resulting DNS names are somewhat tedious for humans to
   type, but are very easy for programs to generate. Making each
   hexadecimal digit a separate label means that delegation on arbitrary
   bit boundaries will result in a maximum of 16 NS RRsets per label
   level; again, the mechanism is somewhat tedious for humans, but is
   very easy to program. As with IPv4's IN-ADDR.ARPA tree, the one
   place where this scheme is weak is in handling delegations in the
   least significant label; however, since there appears to be no real
   need to delegate the least significant four bits of an IPv6 address,
   this does not appear to be a serious restriction.

   [RFC2874] proposed a radically different way of naming entries in the
   reverse mapping tree: rather than using textual representations of
   addresses, it proposes to use a new kind of DNS label (a "bit label")
   to represent binary addresses directly in the DNS. This has the
   advantage of being significantly more compact than the textual
   representation, and arguably might have been a better solution for
   DNS to use for this purpose if it had been designed into the protocol

from the outset. Unfortunately, experience to date suggests that
deploying a new DNS label type is very hard: all of the DNS name
servers that are authoritative for any portion of the name in
question must be upgraded before the new label type can be used, as
must any resolvers involved in the resolution process. Any name
server that has not been upgraded to understand the new label type
will reject the query as being malformed.

Since the main benefit of the bit label approach appears to be an
ability that we don't really need (delegation in the least
significant four bits of an IPv6 address), and since the upgrade
problem is likely to render bit labels unusable until a significant
portion of the DNS code base has been upgraded, it is difficult to
escape the conclusion that the textual solution is good enough.

DNAME RRs

   [RFC2874] also proposes using DNAME RRs as a way of providing the
   equivalent of A6's fragmented addresses in the reverse mapping tree.
   That is, by using DNAME RRs, one can write zone files for the reverse
   mapping tree that have the same ability to cope with rapid
   renumbering or GSE-style routing that the A6 RR offers in the main
   portion of the DNS tree. Consequently, the need to use DNAME in the
   reverse mapping tree appears to be closely tied to the need to use
   fragmented A6 in the main tree: if one is necessary, so is the other,
   and if one isn't necessary, the other isn't either.

   Other uses have also been proposed for the DNAME RR, but since they
   are outside the scope of the IPv6 address discussion, they will not
   be addressed here.

Recommendation

   Distilling the above feature comparisons down to their key elements,
   the important questions appear to be:

   (a) Is IPv6 going to do rapid renumbering or GSE-like routing?

   (b) Is the reverse mapping tree for IPv6 going to require delegation
       in the least significant four bits of the address?

   Question (a) appears to be the key to the debate. This is really a
   decision for the IPv6 community to make, not the DNS community.

   Question (b) is also for the IPv6 community to make, but it seems
   fairly obvious that the answer is "no".

Recommendations based on these questions:

(1) If the IPv6 working groups seriously intend to specify and deploy
    rapid renumbering or GSE-like routing, we should transition to
    using the A6 RR in the main tree and to using DNAME RRs as
    necessary in the reverse tree.

(2) Otherwise, we should keep the simpler AAAA solution in the main
    tree and should not use DNAME RRs in the reverse tree.

(3) In either case, the reverse tree should use the textual
    representation described in [RFC1886] rather than the bit label
    representation described in [RFC2874].

(4) If we do go to using A6 RRs in the main tree and to using DNAME
    RRs in the reverse tree, we should write applicability statements
    and implementation guidelines designed to discourage excessively
    complex uses of these features; in general, any network that can
    be described adequately using A6 0 RRs and without using DNAME
    RRs should be described that way, and the enhanced features
    should be used only when absolutely necessary, at least until we
    have much more experience with them and have a better
    understanding of their failure modes.

Security Considerations

   This note compares two mechanisms with similar security
   characteristics, but there are a few security implications to the
   choice between these two mechanisms:

(1) The two mechanisms have similar but not identical interactions
    with DNSSEC. Please see the section entitled "Interactions with
    DNSSEC" (above) for a discussion of these issues.

(2) To the extent that operational complexity is the enemy of
    security, the tradeoffs in operational complexity discussed
    throughout this note have an impact on security.

(3) To the extent that protocol complexity is the enemy of security,
    the additional protocol complexity of [RFC2874] as compared to
    [RFC1886] has some impact on security.

IANA Considerations

   None, since all of these RR types have already been allocated.

Acknowledgments

   This note is based on a number of discussions both public and private
   over a period of (at least) eight years, but particular thanks go to
   Alain Durand, Bill Sommerfeld, Christian Huitema, Jun-ichiro itojun
   Hagino, Mark Andrews, Matt Crawford, Olafur Gudmundsson, Randy Bush,
   and Sue Thomson, none of whom are responsible for what the author did
   with their ideas.

References

   [RFC1886]   Thomson, S. and C. Huitema, "DNS Extensions to support
                  IP version 6", RFC 1886, December 1995.

   [RFC2874]   Crawford, M. and C. Huitema, "DNS Extensions to Support
                  IPv6 Address Aggregation and Renumbering", RFC 2874,
                  July 2000.

   [Sommerfeld]   Private message to the author from Bill Sommerfeld dated
                  21 March 2001, summarizing the result of experiments he
                  performed on a copy of the MIT.EDU zone.

   [GSE]   "GSE" was an evolution of the so-called "8+8" proposal
                  discussed by the IPng working group in 1996 and 1997.
                  The GSE proposal itself was written up as an Internet-
                  Draft, which has long since expired. Readers interested
                  in the details and history of GSE should review the IPng
                  working group's mailing list archives and minutes from
                  that period.

Author's Address

   Rob Austein

   EMail: sra@hactrn.net

Full Copyright Statement

Acknowledgement