DNS Operations WG
Internet-Draft
Expires: May 1, 2004

A. Durand
SUN Microsystems, Inc.
J. Ihren
Autonomica
P. Savola
CSC/FUNET
Nov 2003

Operational Considerations and Issues with IPv6 DNS
draft-ietf-dnsop-ipv6-dns-issues-03.txt

Status of this Memo

Copyright Notice

Abstract

This memo presents operational considerations and issues with IPv6 Domain Name System (DNS), including a summary of special IPv6 addresses, documentation of known DNS implementation misbehaviour, recommendations and considerations on how to perform DNS naming for service provisioning and for DNS resolver IPv6 support, considerations for DNS updates for both the forward and reverse trees, and miscellaneous issues. This memo is aimed to include a summary of information about IPv6 DNS considerations for those who have experience with IPv4 DNS.

Table of Contents

1. Introduction

This memo presents operational considerations and issues with IPv6 DNS; it is meant to be an extensive summary and a list of pointers for more information about IPv6 DNS considerations for those with experience of IPv4 DNS.

The first section gives a brief overview of how IPv6 addresses and names are represented in the DNS, how transport protocols and resource records (don't) relate, and what IPv4/IPv6 name space fragmentation means and how to avoid it; all of these are described at more length in other documents.

The second section summarizes the special IPv6 address types and how they relate to DNS. The third section describes observed DNS implementation misbehaviour which have a varying effect on the use of IPv6 records with DNS. The fourth section lists recommendations and considerations for provisioning services with DNS. The fifth section in turn looks at recommendations and considerations about providing IPv6 support in the resolvers. The sixth and seveth sections describe considerations with forward and reverse DNS updates, respectively. The eighth section introduces several miscellaneous IPv6 issues relating to DNS for which no better place has been found in this memo. Appendix A looks briefly at the requirements for site-local addressing.

1.1 Representing IPv6 Addresses in DNS Records

In the forward zones, IPv6 addresses are represented using AAAA records. In the reverse zones, IPv6 address are represented using PTR records in the nibble format under the ip6.arpa. -tree. See [1] for more about IPv6 DNS usage, and [2] or [4] for background information.

In particular one should note that the use of A6 records, DNAME

records in the reverse tree, or Bitlabels in the reverse tree is not recommended [2].

1.2 Difference of DNS Transport and DNS Records

In DNS, the IP version used to transport the queries and responses is independent of the records being queried: AAAA records can be queried over IPv4, and A records over IPv6. The DNS servers must not make any assumptions about what data to return for Answer and Authority sections.

However, there is some debate whether the addresses in Additional section could be selected or filtered using hints obtained from which

transport was being used; this has some obvious problems because in many cases the transport protocol does not correlate with the requests, and because a "bad" answer is in a way worse than no answer at all (consider the case where the client is led to believe that a name received in the additional record does not have any AAAA records to begin with).

As stated in [1]:

> The IP protocol version used for querying resource records is independent of the protocol version of the resource records; e.g., IPv4 transport can be used to query IPv6 records and vice versa.

1.3 Avoiding IPv4/IPv6 Name Space Fragmentation

To avoid the DNS name space from fragmenting into parts where some parts of DNS are only visible using IPv4 (or IPv6) transport, the recommendation is to always keep at least one authoritative server IPv4-enabled, and to ensure that recursive DNS servers support IPv4. See DNS IPv6 transport guidelines [3] for more information.

2. DNS Considerations about Special IPv6 Addresses

There are a couple of IPv6 address types which are somewhat special; these are considered here.

2.1 Limited-scope Addresses

The IPv6 addressing architecture [5] includes two kinds of local-use addresses: link-local (fe80::/10) and site-local (fec0::/10). The site-local addresses are being deprecated [6], and are only discussed in Appendix A.

Link-local addresses should never be published in DNS, because they have only local (to the connected link) significance [7].

2.2 Privacy (RFC3041) Address

Privacy addresses (RFC3041 [8]) use a random number as the interface identifier. Publishing DNS records relating to such addresses would defeat the purpose of the mechanism and is not recommended. If absolutely necessary, a mapping could be made to some non-identifiable name, as described in [8].

2.3 6to4 Addresses

6to4 [9] specifies an automatic tunneling mechanism which maps a

public IPv4 address V4ADDR to an IPv6 prefix 2002:V4ADDR::/48. Providing reverse DNS delegation path for such addresses is a challenge. Note that similar difficulties don't surface with the other automatic tunneling mechanisms (in parcicular, providing reverse DNS information for Teredo hosts whose address includes the UDP port of the NAT binding does not seem reasonable).

If the reverse DNS population would be desirable (see Section 7.1 for applicability), there are a number of ways to tackle the delegation path problem [10], some more applicable than the others.

The main proposal [11] has been to allocate 2.0.0.2.ip6.arpa. to RIRs and let them do subdelegations in accordance to the delegations of the respective IPv4 address space. This has a major practical drawback: those ISPs and IPv4 address space holders where 6to4 is being used do not, in general, provide any IPv6 services -- as otherwise, most people would not use 6to4 to begin with -- and it is improbable that the reverse delegation chain would be completed either. In most cases, creating such delegation chains might just lead to latencies caused by lookups for (almost always) non-existant DNS records.

3. Observed DNS Implementation Misbehaviour

Several classes of misbehaviour in DNS servers, load-balancers and resolvers has been observed. Most of these are rather generic, not only applicable to IPv6 -- but in some cases, the consequences of this misbehaviour are extremely severe in IPv6 environments and deserve to be mentioned.

3.1 Misbehaviour of DNS Servers and Load-balancers

There are several classes of misbehaviour in certain DNS servers and load-balancers which have been noticed and documented [12]: some implementations silently drop queries for unimplemented DNS records types, or provide wrong answers to such queries (instead of a proper negative reply). While typically these issues are not limited to AAAA records, the problems are aggravated by the fact that AAAA records are being queried instead of (mainly) A records.

The problems are serious because when looking up a DNS name, typical getaddrinfo() implementations, with AF_UNSPEC hint given, first try to query the AAAA records of the name, and after receiving a response, query the A records. This is done in a serial fashion -- if the first query is never responded (instead of properly returning a negative answer), significant timeouts will occur.

In consequence, this is an enermous problem for IPv6 deployments, and

in some cases, IPv6 support in the software has even been disabled due to these problems.

The solution is to fix or retire those misbehaving implementations, but that is likely not going to be effective. There are some possible ways to mitigate the problem, e.g. by performing the lookups somewhat in parallel and reducing the timeout as long as at least one answer has been received; but such methods remain to be investigated; slightly more on this in Section 5.

3.2 Misbehaviour of DNS Resolvers

Several classes of misbehaviour have also been noticed in DNS resolvers [13]. However, these do not seem to directly impair IPv6

use, and are only referred to for completeness.

4. Recommendations for Service Provisioning using DNS

When names are added in the DNS to facilitate a service, there are several general guidelines to consider to be able to do it as smoothly as possible.

4.1 Use of Service Names instead of Node Names

When a node includes multiple services, one should keep them logically separate in the DNS. This can be done by the use of service names instead of node names (or, "hostnames").

For example, assume a node named "pobox.example.com" provides both SMTP and IMAP service. Instead of configuring the MX records to point at "pobox.example.com", and configuring the mail clients to look up the mail via IMAP from "pobox.example.com", one should use e.g. "smtp.example.com" for SMTP (for both message submission and mail relaying between SMTP servers) and "imap.example.com" for IMAP. Note that in the specific case of STMP relaying, the server itself must typically also be configured to know all its names to ensure loops do not occur. DNS can provide a layer of indirection between service names and where the service actually is, and using which addresses.

This is a good practice with IPv4 as well, because it provides more flexibility and enables easier migration of services from one host to another. A specific reason why this is relevant for IPv6 is that the different services may have a different level of IPv6 support -- that is, one node providing multiple services might want to enable just one service to be IPv6-visible while keeping some others as IPv4-only. Using service names enables more flexibility with different IP versions as well.

4.2 Separate vs the Same Service Names for IPv4 and IPv6

The service naming can be achieved in basically two ways: when a service is named "service.example.com" for IPv4, the IPv6-enabled service could be either added to "service.example.com", or added separately to a sub-domain, like, "service.ipv6.example.com".

Both methods have different characteristics. Using a sub-domain allows for easier service piloting, probably not disturbing the "regular" users of IPv4 service; however, the service would not be used without explicitly asking for it (or, within a restricted network, modifying the DNS search path) -- so it will not actually be used that much. Using the same service name is the "long-term" solution, but may degrade performance for those clients whose IPv6 performance is lower than IPv4, or does not work as well (see the next subsection for more).

In most cases, it makes sense to pilot or test a service using separate service names, and move to the use of the same name when confident enough that the service level will not degrade for the users unaware of IPv6.

4.3 Adding the Records Only when Fully IPv6-enabled

The recommendation is that AAAA records for a service should not be added to the DNS until all of following are true:

1. The address is assigned to the interface on the node.

2. The address is configured on the interface.

3. The interface is on a link which is connected to the IPv6 infrastructure.

In addition, if the AAAA record is added for the node, instead of service as recommended, all the services of the node should be IPv6-enabled prior to adding the AAAA record.

For example, if an IPv6 node is isolated from an IPv6 perspective (e.g., it is not connected to IPv6 Internet) constraint #3 would mean that it should not have an address in the DNS.

Consider the case of two dual-stack nodes, which both have IPv6 enabled, but the server does not have (global) IPv6 connectivity. As the client looks up the server's name, only A records are returned (if the recommendations above are followed), and no IPv6 communication, which would be unsuccessful, is even attempted.

The issues are not always so black-and-white. Usually it's important if the service offered using both protocols is of roughly equal quality, using the appropriate metrics for the service (e.g., latency, throughput, low packet loss, general reliability, etc.) -- this is typically very important especially for interactive or real-time services. In many cases, the quality of IPv6 connectivity is not yet equal to that of IPv4, at least globally -- this has to be taken into consideration when enabling services [14].

4.4 IPv6 Transport Guidelines for DNS Servers

As described in Section 1.3 and [3], there should continue to be at least one authorative IPv4 DNS server for every zone, even if the zone has only IPv6 records. (Note that obviously, having more servers with robust connectivity would be preferably, but this is the recommendation.)

5. Recommendations for DNS Resolver IPv6 Support

When IPv6 is enabled on a node, there are several things to consider to ensure that the process is as smooth as possible.

5.1 DNS Lookups May Query IPv6 Records Prematurely

The system library that implements the getaddrinfo() function for looking up names is a critical piece when considering the robustness of enabling IPv6; it may come in basically three flavours:

1. The system library does not know whether IPv6 has been enabled in the kernel of the operating system: it may start looking up AAAA records with getaddrinfo() and AF_UNSPEC hint when the system is upgraded to a system library version which supports IPv6.

2. The system library might start to perform IPv6 queries with getaddrinfo() only when IPv6 has been enabled in the kernel. However, this does not guarantee that there exists any useful IPv6 connectivity (e.g., the node could be isolated from the other IPv6 networks, only having link-local addresses).

3. The system library might implement a toggle which would apply some heuristics to the "IPv6-readiness" of the node before starting to perform queries; for example, it could check that a link-local IPv6 address exists, or a global IPv6 address exists.

First, let us consider generic implications of unnecessary queries for AAAA records: when looking up all the records in the DNS, AAAA records are typically tried first, and then A records. These are done in serial, and the A query is not performed until a response is

received to the AAAA query. Considering the misbehaviour of DNS servers and load-balancers, as described in Section 3.1, the look-up delay for AAAA may incur additional unnecessary latency, and introduce a component of unreliability.

One option here could be to do the queries partially in parallel; for example, if the final response to the AAAA query is not received in 0.5 seconds, start performing the A query while waiting for the result (immediate parallelism might be unoptimal without information sharing between the look-up threads, as that would probably lead to duplicate non-cached delegation chain lookups).

An additional concern is the address selection, which may, in some circumstances, prefer AAAA records over A records, even when the node does not have any IPv6 connectivity [15]. In some cases, the implementation may attempt to connect or send a datagram on a physical link [16], incurring very long protocol timeouts, instead of quickly failing back to IPv4.

Now, we can consider the issues specific to each of the three possibilities:

In the first case, the node performs a number of completely useless DNS lookups as it will not be able to use the returned AAAA records anyway. (The only exception is where the application desires to know what's in the DNS, but not use the result for communication.) One should be able to disable these unnecessary queries, for both latency and reliability reasons. However, as IPv6 has not been enabled, the connections to IPv6 addresses fail immediately, and if the application is programmed properly, the application can fall gracefully back to IPv4 [17].

The second case is similar to the first, except it happens to a smaller set of nodes when IPv6 has been enabled but connectivity has not been provided yet; similar considerations apply, with the exception that IPv6 records, when returned, will be actually tried first which may typically lead to long timeouts.

The third case is a bit more complex: optimizing away the DNS lookups with only link-locals is probably safe (but may be desirable with different lookup services which getaddrinfo() may support), as the link-locals are typically automatically generated when IPv6 is enabled, and do not indicate any form of IPv6 connectivity. That

is, performing DNS lookups only when a non-link-local address has been configured on any interface could be beneficial -- this would be an indication that either the address has been configured either from a router advertisement, DHCPv6, or manually. Each would indicate at least some form of IPv6 connectivity, even though there would not be

guarantees of it.

XXXX: are there any actual recommendations in here?!? :-)

5.2 Recursive DNS Server Discovery

Recursive IPv6 DNS server discovery is a subject of active debate at the moment: the main proposed mechanisms include the use of well-known addresses [18], the use of Router Advertisements to convey the information [19], and using DHCPv6 (or the stateless subset of it [20]) for DNS server configuration. No consensus has been reached yet.

Note that IPv6 DNS server discovery, while an important topic, is not required for dual-stack nodes with dual-stack networks: IPv6 DNS records can very well be queried over IPv4.

5.3 IPv6 Transport Guidelines for Resolvers

As described in Section 1.3 and [3], the recursive resolvers should be IPv4-only or dual-stack to be able to reach any IPv4-only DNS server. Note that this requirement is also fulfilled by an IPv6-only stub resolver pointing to a dual-stack recursive DNS resolver.

6. Considerations about Forward DNS Updating

While the topic how to enable updating the forward DNS, i.e., the mapping from names to the correct new addresses, is not specific to IPv6, it bears thinking about especially due to adding Stateless Address Autoconfiguration [21] to the mix.

Typically forward DNS updates are more manageable than doing them in the reverse DNS, because the updater can, typically, be assumed to "own" a certain DNS name -- and we can create a form of security association with the DNS name and the node allowed to update it to point to a new address.

A more complex form of DNS updates -- adding a whole new name to a DNS zone, instead of updating an existing one -- is considered out-of-scope (XXX: at least for now, send text/feedback!).

6.1 Manual or Custom DNS Updates

The DNS mappings can be maintained by hand, in a semi-automatic fashion or by running non-standardized protocols. These are not considered at more length in this memo.

6.2 Dynamic DNS

Dynamic DNS updates (DDNS) [22][23] is a standardized mechanism for dynamically updating the DNS. It works equally well with stateless address autoconfiguration (SLAAC), DHCPv6 or manual address configuration. The only (minor) twist that with SLAAC, the DNS server cannot tie the authentication of the user to the IP address, and stronger mechanisms must be used. Actually, relying on IP addresses for Dynamic DNS is rather insecure at best, so this is probably not a significant problem (but requires that the authorization keying will be explicitly configured).

Note that the nodes must somehow be configured with the information about the servers where they will attempt to update their addresses, sufficient security material for authenticating themselves to the server, and the hostname they will be updating. Unless otherwise configured, the first could be obtained by looking up the authorative name servers for the hostname; the second must be configured explicitly unless one chooses to trust the IP address -based authentication (not a good idea); and lastly, the nodename is typically pre-configured somehow on the node, e.g. at install time.

Care should be observed when updating the addresses not to use longer TTLs for addresses than are preferred lifetimes for the autoconfigured addresses, so that if the node is renumberedin a managed fashion, the amount of stale DNS information is kept to the minimum.

7. Considerations about Reverse DNS Updating

Forward DNS updating was rather straightforward; reverse DNS is significantly trickier especially with certain mechanisms. However, first it makes sense to look at the applicability of reverse DNS in the first place.

7.1 Applicability of Reverse DNS

Today, some applications use reverse DNS to either look up some hints about the topological information associated with an address (e.g. resolving web server access logs), or as a weak form of a security check, to get a feel whether the user's network administrator has "authorized" the use of the address (on the premises that adding a reverse record for an address would signal some form of authorization).

One additional, maybe slightly more useful applicability is ensuring the reverse and forward DNS contents match and correspond to a configured name or domain. As a security check, it is typically

accompanied by other mechanisms, such as a user/password login; the main purpose of the DNS check is to weed out the majority of unauthorized users, and if someone managed to bypass the checks, he would still need to authenticate "properly".

It is not clear whether it makes sense to require or recommend that reverse DNS records be updated. In many cases, it would just make more sense to use proper mechanisms for security (or topological information lookup) in the first place. At minimum, the applications which use it as a generic authorization (in the sense that a record exists at all) should be modified as soon as possible to avoid such lookups completely.

7.2 Manual or Custom DNS Updates

Reverse DNS can be updated using manual or custom methods, naturally. These are not further described here, except for one special case.

One way to deploy reverse DNS would be to use wildcard records, for example, by configuring one name for a subnet (/64) or a site (/48). Naturally, such a name could not be verified from the forward DNS, but would at least provide some form of "topological information" or

"weak authorization" if that is really considered to be useful. Note that this is not actually updating the DNS as such, as the whole point is to avoid DNS updates completely by manual configuration of a generic name.

7.3 DDNS with Stateless Address Autoconfiguration

Dynamic DNS with SLAAC is a bit complicated, but manageable with a rather low form of security with some implementation.

Every node on a link must then be allowed to insert its own reverse DNS record in the reverse zone. However, in the typical case, there can be no stronger form of authentication between the nodes and the server than the source IP address (the user may roam to other administrative domains as well, requiring updates to foreign DNS servers), which might make attacks more lucrative.

Moreover, the reverse zones must be cleaned up by some janitorial process: the node does not typically know a priori that it will be disconnected, and cannot send a DNS update using the correct source address to remove a record.

7.4 DDNS With DHCP

With DHCP, the reverse DNS name is typically already inserted to the DNS that reflects to the name (e.g., "dhcp-67.example.com").

If a more explicit control is required, similar considerations as with SLAAC apply, except for the fact that typically one must update a reverse DNS record instead of inserting one -- due to a denser address assignment policy -- and updating a record seems like a slightly more difficult thing to secure.

7.5 DDNS with Dynamic Prefix Delegation

In cases where more than one address is being used and updated, one should consider where the updated server resides. That is, whether the prefixes have been delegated to a node in the local site, or whether they reside elsewhere, e.g., at the ISP. The reverse DNS updates are typically easier to manage if they can be done within a single administrative entity -- and therefore, if a reverse DNS delegation has been made, it may be easier to enable reverse DNS at

the site, e.g. by a wildcard record, or by some DNS update mechanism.

8. Miscellaneous DNS Considerations

This section describes miscellaneous considerations about DNS which seem related to IPv6, for which no better place has been found in this document.

8.1 NAT-PT with DNS-ALG

NAT-PT [24] DNS-ALG is a critical component (unless something replacing that functionality is specified) which mangles A records to look like AAAA records to the IPv6-only nodes. Numerous problems have been identified with DNS-ALG [25].

8.2 Renumbering Procedures and Applications' Use of DNS

One of the most difficult problems of renumbering procedures [26] is that an application which gets a DNS name disregards information such as TTL, and uses the result obtained from DNS as long as it happens to be stored in the memory of the application. For applications which run for a long time, this could be days, weeks or even months; some applications may be clever enough to organize the data structures and functions in such a manner that look-ups get refreshed now and then. This is an issue with no clear solution.

9. Acknowledgements

Some recommendations (Section 4.3, Section 5.1) about IPv6 service provisioning were moved here from [27] by Erik Nordmark and Bob Gilligan. Havard Eidnes provided useful feedback and improvements.

10. Security Considerations

This document reviews the operational procedures for IPv6 DNS operations and does not have security considerations in itself.

However, it is worth nothing that in particular with Dynamic DNS Updates, security models based on the source address validation are very weak and cannot be recommended. On the other hand, it should be

noted that setting up an authorization mechanism (e.g., a shared secret, or public-private keys) between a node and the DNS server has to be done manually, and may require quite a bit of time and expertise.

To re-emphasize which was already stated, reverse DNS checks provide very weak security at best, and the only (questionable) security-related use for them may be in conjunction with other mechanisms when authenticating a user.

Normative References

[1] Thomson, S., Huitema, C., Ksinant, V. and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.

[2] Bush, R., Durand, A., Fink, B., Gudmundsson, O. and T. Hain, "Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)", RFC 3363, August 2002.

[3] Durand, A. and J. Ihren, "DNS IPv6 transport operational guidelines", draft-ietf-dnsop-ipv6-transport-guidelines-01 (work in progress), October 2003.

Informative References

[4] Bush, R., "Delegation of IP6.ARPA", BCP 49, RFC 3152, August 2001.

[5] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[6] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", draft-ietf-ipv6-deprecate-site-local-02 (work in progress), November 2003.

[7] Hazel, P., "IP Addresses that should never appear in the public DNS", draft-ietf-dnsop-dontpublish-unreachable-03 (work in progress), February 2002.

[8] Narten, T. and R. Draves, "Privacy Extensions for Stateless

Address Autoconfiguration in IPv6", RFC 3041, January 2001.

[9] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[10] Moore, K., "6to4 and DNS", draft-moore-6to4-dns-03 (work in progress), October 2002.

[11] Bush, R. and J. Damas, "Delegation of 2.0.0.2.ip6.arpa", draft-ymbk-6to4-arpa-delegation-00 (work in progress), February 2003.

[12] Morishita, Y. and T. Jinmei, "Common Misbehavior against DNS Queries for IPv6 Addresses", draft-morishita-dnsop-misbehavior-against-aaaa-00 (work in progress), June 2003.

[13] Larson, M. and P. Barber, "Observed DNS Resolution Misbehavior", draft-ietf-dnsop-bad-dns-res-01 (work in progress), June 2003.

[14] Savola, P., "Moving from 6bone to IPv6 Internet", draft-savola-v6ops-6bone-mess-01 (work in progress), November 2002.

[15] Roy, S., "Dual Stack IPv6 on by Default", draft-ietf-v6ops-v6onbydefault-00 (work in progress), October 2003.

[16] Roy, S., "IPv6 Neighbor Discovery On-Link Assumption Considered Harmful", draft-ietf-v6ops-onlinkassumption-00 (work in progress), October 2003.

[17] Shin, M., "Application Aspects of IPv6 Transition", draft-shin-v6ops-application-transition-02 (work in progress), October 2003.

[18] Ohta, M., "Preconfigured DNS Server Addresses", draft-ohta-preconfigured-dns-00 (work in progress), July 2003.

[19] Jeong, J., "IPv6 DNS Discovery based on Router Advertisement", draft-jeong-dnsop-ipv6-dns-discovery-00 (work in progress), July 2003.

[20] Droms, R., "A Guide to Implementing Stateless DHCPv6 Service", draft-ietf-dhc-dhcpv6-stateless-01 (work in progress), October 2003.

[21] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

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

[23] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

[24] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[25] Durand, A., "Issues with NAT-PT DNS ALG in RFC2766", draft-durand-v6ops-natpt-dns-alg-issues-00 (work in progress), February 2003.

[26] Baker, F., "Procedures for Renumbering an IPv6 Network without a Flag Day", draft-baker-ipv6-renumber-procedure-01 (work in progress), October 2003.

[27] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", draft-ietf-v6ops-mech-v2-01 (work in progress), October 2003.

Authors' Addresses

Alain Durand
SUN Microsystems, Inc.
17 Network circle UMPL17-202
Menlo Park, CA 94025
USA

EMail: Alain.Durand@sun.com

Johan Ihren
Autonomica
Bellmansgatan 30
SE-118 47 Stockholm
Sweden

EMail: johani@autonomica.se

Pekka Savola
CSC/FUNET

Espoo
Finland

EMail: psavola@funet.fi

Appendix A. Site-local Addressing Considerations for DNS

As site-local addressing is being deprecated, and it is not yet clear whether an addressing-based replacement (and which kind) is devised, the considerations for site-local addressing are introduced here.

The interactions with DNS come in two flavors: forward and reverse DNS.

To actually use site-local addresses within a site, this implies the deployment of a "split-faced" or a fragmented DNS name space, for the zones internal to the site, and the outsiders' view to it. The procedures to achieve this are not elaborated here. The implication is that site-local addresses must not be published in the public DNS.

To faciliate reverse DNS (if desired) with site-local addresses, the stub resolvers must look for DNS information from the local DNS servers, not e.g. starting from the root servers, so that the site-local information may be provided locally. Note that the experience private addresses in IPv4 has shown that the root servers get loaded for requests for private address lookups in any case.

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.