Role of DNS in Next Generation Transition

TAPIO SOKURA
Helsinki University of Technology, Espoo

Without properly functioning Domain Name System (DNS) service a great deal of other Internet services would practically grind to a halt. This is why DNS plays a key role in smooth transition from IPv4 to IPv6.

This paper deals with ways of using DNS to facilitate a graceful transition to IPv6. DNS is discussed as an entity in IPv6 and a brief mail routing experiment in IPv4/IPv6 dual-stack environment is described. Also a few translation mechanisms between IPv4 and IPv6 worlds are introduced.

1. INTRODUCTION

The Domain Name System (DNS) is a critical part of today's Internet. Without DNS there would not be much web-surfing or emailing. This is as true with IPv6 as it is with IPv4, maybe even more so because of longer and thus harder to remember IP addresses.

So the basic function of DNS is to resolve names such as `www.example.org` to IP addresses such as `10.3.65.23` (IPv4) or `3ffe:26ff:7:abcd:35dg:dead:beef:6f9` (IPv6). Building on this the Resource Records (RRs) that make the use of IPv6 addresses possible in the DNS are presented first. The current situation of other DNS issues related to IPv6 will also be discussed.

Name resolution order is presented next; is IPv4 or IPv6 preferred over the other. Mail routing in mixed IPv4 and IPv6 environments is taken as a closer example and results of some simple practical experiments will be presented.

Next generation transition techniques enable limited internetworking between IPv4 and IPv6 worlds. Some clever ways of using DNS in conjunction with these techniques are introduced before closing remarks.

The reader is expected to be familiar with the basic properties of Internet Protocol versions four and six. Knowledge of the Domain Name System (DNS) is also required.

2. IPV6 DNS EXTENSIONS

When IPv6 was first defined in the middle of 1990's, it was immediately realized that changes need to be made to DNS. The most important additions and changes to DNS that relate to IPv6 are outlined in this section.

Name to IP mappings are discussed in section 2.1, new style name delegations in section 2.2 and IP to name mappings are handled in section 2.3. A few words of the current state of the root name servers in relation to IPv6 are said in section 2.4.

The reader of this section should be familiar with the basic concepts and resource record types used in IPv4 DNS.

2.1 Name to IP mappings

2.1.1 *AAAA.* The basic building block of IPv4 DNS is the `A` resource record that connects a name such as `machine.example.org` to an IPv4 address `10.83.239.5`, for example. The equivalent of an IPv4 `A` record in IPv6 world is the `AAAA` record that could map the same name to `123:df6:29ff:3e:2a0:45ff:fe26:953`, for example [10]. Disregarding address length, these two RRs are semantically identical.

2.1.2 *A6.* Later a kind of a competitor to `AAAA` resource record was defined: the `A6` RR [4]. The basic idea behind `A6` records is to make IPv6 addressing more flexible. `A6` allows chaining of several `A6` records to "build" the IPv6 address part by part. This can be useful when we have several IPv6 connections to upstream ISPs. That is, we are IPv6 multi-homed. `A6` records make it possible to give our IPv6 capable network devices DNS names that resolve to IPv6 addresses in all of our networks without defining each name several times as would have to be done with `AAAA` records.

As an example, we could define that `multihomed.example.org` resolves to IPv6 addresses `1234::aaaa`, `5678::aaaa` and `90ab:aaaa` using `A6` records. This could be done so that we first use a single `A6` record `multihomed.example.org` to point to addresses ending in `::aaaa` and to look for more address information in `nets.example.org`. Then we would have three

Author's contact information: tapio.sokura@iki.fi
I would like to thank my tutor Olli Knuuttila for his valuable comments and guidance on this work.

Published in *Internet Protocols for Mobile Computing - Seminar on Internetworking, Autumn 2002*, by Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory. The complete publication can be found at http://www.tml.hut.fi/Studies/T-110.551/2002/papers/December/index.html.
ISBN 951-22-6272-X - ISSN 1455-9749 - HUT - TML - TML-C9

`nets.example.org` `A6` records and each of them would point to a different front part of the address, namely `1234::`, `5678::` and `90ab::`. The more multi-homed hosts we have the less RRs we need compared to using `AAAA` records.

2.1.3 *AAAA or A6?*. So we've established that `A6` records are more versatile than `AAAA` records. But which ones are really used? The short answer at the moment is `AAAA`, which is — for the time being — also recommended by RFC 3363 [2] over `A6` records. Besides being complicated and error-prone to use, `A6` records also cause considerable overhead to the name resolution process. The overhead comes from the fact that you often have to traverse a path of several `A6` records to obtain the whole IPv6 address. `AAAA` queries instead always return a full IPv6 address.

No one can tell what happens in the future, but the current trend is `AAAA`.

2.2 DNAME

The idea behind `DNAME` is to alias an entire subtree of the DNS hierarchy to another subtree. `DNAME` can be seen as a relative of `CNAME`, which aliases a single DNS name to another name. `DNAME` is also a quite recent addition to DNS and not much used. [3]

An example: We have a `DNAME` record that aliases `example.org` to `anotherexample.com`. Now when looking up the name `a.b.example.org` our DNS resolver stumbles on the above mentioned `DNAME` record and goes on looking for `a.b.anotherexample.com`.

As can be seen from the definition, `DNAME` is not an IPv6 specific extension to DNS. But `DNAME`s play a role in IPv6 reverse lookup process when used in conjunction with `A6` records. Because the use of `A6` records is currently deprecated, so is the corresponding reverse lookup usage of `DNAME`s in IPv6.

2.3 Reverse lookup

Reverse name resolution is used to find a DNS name for a given IP address. This is performed in IPv6 using the `PTR` record in much the same way as is done in IPv4.

One difference is found the DNS name space used for resolution: IPv4 uses the `IN-ADDR.ARPA` tree while IPv6 currently uses `IP6.ARPA` and `IP6.INT` trees.

Two separate reverse lookup trees are in use because the early RFCs, namely 1886 [10], specified `IP6.INT`, but newer ones (RFC 3152) [1] have moved to recommending `IP6.ARPA`. The change is being done because `ARPA` tree is nowadays considered the place for DNS-related technical information.

This division causes problems in reverse name lookups because all IPv6 address spaces currently in use aren't found in both trees. Address delegations for the worldwide IPv6 testbed 6bone[1], for example, are currently only found in the `IP6.INT` tree. The result is that if the DNS resolver of an operating system checks only the `IP6.ARPA` tree for an IPv6 address starting with `3ffe:`, nothing is found. A common approach in current DNS resolver libraries is to first try the `IP6.ARPA` tree and if that doesn't produce an RR, then try `IP6.INT`.

There is also an early proposal (work in progress) to use ICMPv6 node information queries and replies for reverse DNS lookups [7].

2.4 Root name servers

The DNS servers that sit on top of the Internet name space — at the 'dot' that is — are called root name servers. The IP addresses of root name servers is the only thing a resolving DNS server needs to know; everything else it can find out by asking directions from the root name servers.

Currently there are no official IPv6 root name servers in operation. This is not a big problem for the time being, because a resolving DNS server will need IPv4 connectivity also to reach those DNS zones that are served by servers having IPv4 connectivity only.

Nevertheless IPv6-reachable root name servers would in my opinion be an important symbolic gesture towards accepting IPv6 into official production use. At the moment there exists a closed testbed of IPv6 root and certain top-level domain name servers [8].

3. NAME RESOLUTION ORDER CONSIDERATIONS

As a rough rule of thumb, it can be said that most programs and operating system resolvers capable of both IPv4 and IPv6 tend to prefer IPv6. This selection of course only needs to be done when both transports are available. It is also quite common to have built-in functionality that tries all IP addresses and/or protocol versions transparently without asking the user.

IPv4/6 protocol independency can basically be achieved in a C language program by using the system call `getaddrinfo()` for name resolution. If it is used with a loose enough wildcard in the protocol restriction field, the underlying IP version of the resulting address structure can be either four or six. In addition to this a program naturally needs to be aware of IPv6 in other levels, for example when parsing configuration files and user input.

Although it is possible to define one DNS name to have both `A` and `AAAA` resource records, one should be careful in their usage. A combination of IPv4-only connectivity and DNS returning both `A` and `AAAA` records for one DNS name can result in a failed connection depending on the quirks of the program used. This could happen if the client program tried only one protocol or one address and then gave up, for example.

Therefore it can be considered good practice to separate IPv6 names in the DNS to a subtree of their own, like `ipv6.example.org`. This also makes it possible for the user to immediately see which protocol he is currently using; a useful aid in debugging. On the other hand in many non-mission-critical environments it's not a problem to have both IPv4 and IPv6 connectivity behind the same name. This can even be desirable for example in SMTP servers. In these cases it is also useful to have a separate tree resolving only to IPv6 addresses to be able to force the use of IPv6.

[1] IPv6 address prefix `3ffe::/16`

4. MAIL ROUTING IN MIXED IPV4 AND IPV6 ENVIRONMENTS

To present a real-life example of address selection between IPv4 and IPv6 I did some experiments with SMTP. The idea here is to demonstrate how address selection works in practice and what should be considered when dealing with SMTP in mixed/dual-stack environments.

4.1 Software and testing methodology

The operating system used in these experiments was Red Hat Linux 8.0. The originating mail transfer agent (MTA) was Sendmail 8.12.5 that comes as standard with Red Hat and is also IPv6 ready without source code patches or recompilation. Network traffic was monitored with tcpdump 3.6.3 and domain name service was provided by BIND 9.2.1.

Testing was done by sending email to `user@domain`, where domain was varied from test1...4. Only one email was active in the mail queue at any given time to make sure the observed behavior was triggered by the particular piece of email under investigation. To make sure the behavior of MTA was consistent tests were run several times. Also the command `sendmail -q` was actively used to attempt immediate delivery of queued mail. After observing the generated traffic the MTA was shut down and its mail queue emptied before next test run.

An isolated test network was used for the measurements. The network had one DNS server (IPv4), one sending MTA (IPv4 and IPv6) and one receiving "MTA" (IPv4 and IPv6). The receiving MTA was simply an idle computer with working IPv4 and IPv6 addresses, there was no MTA running. Testing was done this way because observing the failing connection attempts is enough to see which IP version and destination addresses were used and in which order. Tcpdump was run on the sending MTA to catch DNS and SMTP traffic and on the receiving MTA to confirm the observed SMTP traffic and it's destination address. A simple illustration of the setup is shown in Fig. 1.

Below are the DNS RRs that were used in BIND style zone file format. The IP addresses have been changed from the real values used in testing to make reading easier and less error-prone.

```
v4-mx     IN A    10.11.11.11
v6-mx     IN AAAA 3ffe::1111
v4-v6-mx  IN A    10.22.22.22
          IN AAAA 3ffe::2222

test1     IN MX   10 v4-v6-mx
test2     IN MX   10 v4-mx
          IN MX   10 v6-mx
test3     IN MX   10 v4-mx
          IN MX   20 v6-mx
test4     IN MX   10 v6-mx
          IN MX   20 v4-mx
```

To recap on `MX` records: the smaller the integer value right after `MX` is, the more preferred the host is as a mail exchanger.

4.2 Results

4.2.1 *user@test1.* The first test case can be thought of as a basic dual-stack mail server configuration. The mail server has both IPv4 and IPv6 addresses under the same name in DNS, `v4-v6-mx`. This name is then listed as the most preferred mail exchanger for the domain `test1`.

In this situation Sendmail behaves very predictably: first the IPv6 address is tried and if that fails, then IPv4. If both attempts fail — that is no IP address is found, no TCP connection is formed or a 4xx SMTP reply code is received — the MTA takes a break and tries again after a certain time period. The retries are also always done in the same order because `AAAA` records are preferred over `A` records for the same DNS name.

4.2.2 *user@test2.* The second test case is similar to the first one, but there is a slight difference in practice. The difference comes from the fact that there are two equally preferred mail exchanger records, `v4-mx` and `v6-mx`. The equal preference values causes some randomization to occur: Sendmail picks either name as the one to try first and does an `AAAA` lookup. If the lookup fails Sendmail searches for an `A` record for the same name. If there is no `A` or `AAAA` record found, Sendmail moves on to the other name and goes through the same lookup process again. If the delivery attempt fails on all listed mail exchangers, Sendmail takes a break and tries again later.

So the practical difference to test1 is that you never know which IP version is tried first. This is also true for delivery retries, which can happen in a different order compared to the earlier attempts. Predictability is usually considered a good thing in computer world. Therefore I see little reason to prefer this organization of mail exchangers compared to test1.

4.2.3 *test3 and test4.* These two cases are sort of opposites of each other and behave very consistently. In test3 mail is always primarily delivered via IPv4 and secondarily via IPv6. Organizing DNS records the way shown in test4 prefers IPv6 over IPv4.

The different mail exchanger preference values used in test3 and test4 make the preference order explicit. In practice test4 is equal to test1 in IP version selection order because most implementations always prefer IPv6 over IPv4, if both transports are available for a given name. But this practice is just a recommendation, it is not wrong or "against the rules" to prefer IPv4 over IPv6.

4.3 Further thoughts and observations

No `A6` queries were observed during the measurements. Whether `A6` records existed in the DNS or not didn't seem to have any effect on Sendmail's (or Linux' DNS resolver's) behavior[2]. `A6` records included in the additional section of DNS reply packets did not seem to be used either. This was expected and follows the current recommendation of preferring `AAAA` records instead of `A6` [2]. Nothing prevents one from using both record types at the same time, but there is not much to gain currently.

[2]Most of the tests were performed with and without `A6` records' presence. They were left out from the zone file listing for clarity.

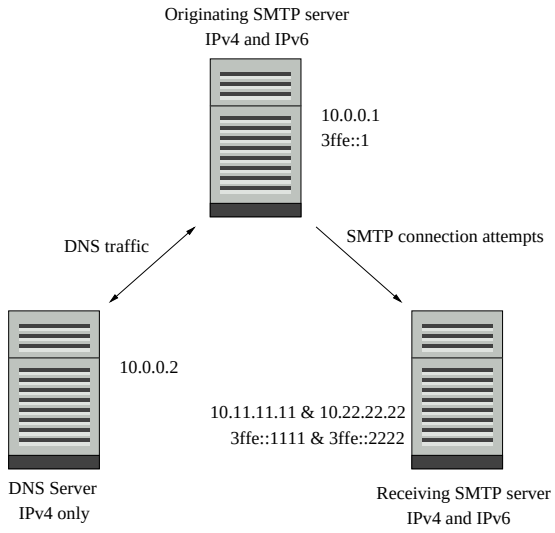

Fig. 1. Mail routing experiment

The DNS traffic captures showed many seemingly unnecessary but repeating DNS queries. For example the mail exchanger records were always queried twice in a row as were the `A` records. But `AAAA` records were always queried only once. Sendmail also triggered a lot of queries that were not directly related to mail delivery. Envelope sender's domain was queried for an `A` record before every delivery attempt, for example.

When planning on using IPv6 SMTP servers for incoming mail it is important to check that all mail exchangers listed in DNS have a way of delivering mail to the most preferred exchanger. Consider this: Your most preferred mail exchanger is capable of IPv4 only and your next preferred server operates in IPv6 only. So now you can receive mail both via IPv4 and IPv6. But what happens when a mail is delivered over IPv6 to the next preferred server? Your mail gets stuck there, because IPv4-only and IPv6-only servers can't talk to each other without some sort of a translation mechanism in between. So make sure email has a way of getting from all secondary mail exchangers to the primary exchanger [9].

5. IPV4 AND IPV6 INTERWORKING IN RELATION TO DNS

IPv4 and IPv6 interworking mechanisms are one way for SMTP servers speaking different versions of IP to exchange mail, for example. IPv4 will probably continue to live beside IPv6 for years after most of the Internet has transitioned to the new protocol version. This also holds true the other way around. Many applications don't need global connectivity anyway so they'll be right at home talking to the ancient server lurking in the far corner of the server room.

But some applications will need to contact hosts using different IP version and this is where the interworking mechanisms come to play. Two IPv4/6 translation mechanisms are presented next. Several other interworking mechanisms have been specified, these two were selected because of the unusual ways they make use of DNS.

5.1 IPv6-to-IPv4 Transport Relay Translator

The idea behind an IPv6-to-IPv4 Transport Relay Translator (TRT) is to pass TCP connections and UDP streams originating in IPv6 world and destined to an IPv4 host. This is an application level system that doesn't require complicated rewriting of IP headers or difficult MTU considerations, for example. Instead there is a TRT machine in the network that breaks the connection into two parts: one from the IPv6 host to the TRT and another from the TRT to the IPv4 host.

So how does an IPv6-only host know what IPv6 address to connect to to reach a certain IPv4 host? This is where tricks with DNS comes into play. In order for this system to work transparently to the IPv6 client we need a special DNS server. The DNS server receives the requests from the client and answers them in IPv6 compatible records only. When a certain resource record exists originally only in the IPv4 domain, the DNS server embeds this IPv4 address into an IPv6 resource record that is returned to the client. This v6-to-v4 record is formed by pasting a dummy network prefix hosted by the TRT and the destination IPv4 address together. For example if the dummy prefix held by TRT was fec0:0:0:1::/64 and the IPv4 address of the target was 10.1.1.1, the resulting IPv6 address would be fec0:0:0:1::10.1.1.1. (aka fec0:0:0:1::a01:101). Because the destination IPv4 address is embedded in the IPv6 address, the TRT has no trouble passing the traffic to the right direction.

A considerable limiting factor of this approach lies in it's one-way nature. Also protocols that transfer endpoint IP addresses inside the data stream (e.g. FTP) can be broken by a TRT. Because the IPv4 address space is considerably smaller than the IPv6 space, it is easy to embed IPv4 addresses to IPv6 addresses but impossible the other way around. This means that all communication between hosts separated by a TRT has to be initiated

from the IPv6 side. TRT is also a stateful mechanism and as such it necessarily doesn't scale as well as stateless translation mechanisms [6].

5.2 NAT-PT

Network Address Translation - Protocol Translation (NAT-PT) represents a slightly different way of translating protocols compared to TRT. The main difference is that a NAT-PT device translates single packets from IPv4 to IPv6 and vice versa. This includes shifting IP addresses back and forth, recalculating packet checksums and general header rearranging. Because NAT-PT operates in a stateless way on single packets, it probably scales a bit better than TRTs.

NAT-PT makes it possible for IPv4 hosts to initiate connections to IPv6 hosts by using a pool of globally unique IPv4 addresses reserved for this translation use. A DNS Application Level Gateway (ALG) is used to handle the mapping of scarce IPv4 addresses to their IPv6 equivalents. The creation of a mapping is triggered by a DNS query from an IPv4 client asking for a resource that is only reachable via IPv6 — an `AAAA` or `A6` record. This causes the DNS-ALG to check for an existing mapping; if none is found a new one is created. The allocated IPv4 address is then passed back to the client as an `A` record. The client can now use the mapped address to communicate with an IPv6 host behind the NAT-PT. After a period of inactivity the mapping is removed and the IPv4 address is returned into the pool for future use.

NAT-PT doesn't solve all IPv4-to-IPv6 and IPv6-to-IPv4 translation problems. A fundamental problem is that the number of available IPv4 addresses for translation pool use is finite. This poses a strict upper limit for the number of cross-version mappings that can exist on any given time [11].

6. SUMMARY

IP version six has been coming just-around-the-corner for several years now. Things look better for IPv6 adoption now than a few years ago when Network Address Translations became fashionable. NATs pushed the IPv4 address shortage problem further into the future, but they don't provide a long term solution.

IPv6:ification will not be a quick and effortless process. Luckily there are ways to smoothen the transition, smart usage of DNS being one of them. Looking at the state of IPv6 DNS it seems that some detours have been taken, like the currently deprecated `A6` RR and the switch of the reverse lookup tree.

Mail routing in mixed IPv4/IPv6 environments is not that different from the IPv4-only world, but there are a few issues that should be kept in mind. Especially important is making sure that all secondary mail servers can deliver incoming mail to a more preferred server.

Internetworking between different versions of IP can be enhanced by using DNS in new dynamic ways. Still, there is no catch-all solution for the problem of cross-version communication in IP.

GLOSSARY

ALG. Application Level Gateway

DNS. Domain Name System

MTA. Mail Transfer Agent

MTU. Maximum Transfer Unit

NAT. Network Address Translation

RR. Resource Record

SMTP. Simple Mail Transfer Protocol

TRT. Transport Relay Translator

REFERENCES

[1] Bush R. RFC 3152: Delegation of IP6.ARPA. Online. Updated August 2001, referred to 3 November 2002. URL: http://www.ietf.org/rfc/rfc3152.txt.

[2] Bush R., Durand A., Fink B., Gudmundsson O. and Hain T. RFC 3363: Represention Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS). Online. Updated August 2002, referred to 3 November 2002. URL: http://www.ietf.org/rfc/rfc3363.txt.

[3] Crawford M. RFC 2672: Non-Terminal DNS Name Redirection. Online. Updated August 1999, referred to 11 October 2002. URL: http://www.ietf.org/rfc/rfc2672.txt.

[4] Crawford M. and Huitema C. RFC 2874: DNS Extensions to Support IPv6 Address Aggregation and Renumbering. Online. Updated July 2000, referred to 11 October 2002. URL: http://www.ietf.org/rfc/rfc2874.txt.

[5] Gilligan R. and Nordmark E. RFC 1933: Transition Mechanisms for IPv6 Hosts and Routers. Online. Updated April 1996, referred to 11 October 2002. URL: http://www.ietf.org/rfc/rfc1933.txt.

[6] Hagino J. and Yamamoto K. RFC 3142: An IPv6-to-IPv4 Transport Relay Translator. Online. Updated June 2001, referred to 11 October 2002. URL: http://www.ietf.org/rfc/rfc3142.txt.

[7] Hagino J. Use of ICMPv6 node information query for reverse DNS lookup. Online. Updated 20 June 2002, referred to 11 October 2002. URL: http://www.ietf.org/internet-drafts/draft-itojun-ipv6-nodeinfo-revlookup-00.txt.

[8] Manning B. Message on the 6bone mailing list, 2002-10-28. Online. Updated 28 October 2002, referred to 3 November 2002. URL: http://mailman.isi.edu/pipermail/6bone/2002-October/006573.html.

[9] Nakamura M. and Hagino J. SMTP operational experience in mixed IPv4/IPv6 environements. Online. Updated 28 June 2002, referred to 11 October 2002. URL: http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-ipv6-smtp-requirement-06.txt.

[10] Thomson S. and Huitema C. RFC 1886: DNS Extensions to support IP version 6. Online. Updated December 1995, referred to 11 October 2002. URL: http://www.ietf.org/rfc/rfc1886.txt.

[11] Tsirtsis G. and Srisuresh P. RFC 2766: Network Address Translation — Protocol Translation (NAT-PT). Online. Updated February 2000, referred to 11 October 2002. URL: http://www.ietf.org/rfc/rfc2766.txt.