Internet Engineering Task Force
INTERNET-DRAFT
Expires: November 27, 2001

Jun-ichiro itojun Hagino
IIJ Research Laboratory
May 27, 2001

Comparison of AAAA and A6 (do we really need A6?)
draft-ietf-dnsext-aaaa-a6-00.txt

Status of this Memo

The internet-draft will expire in 6 months. The date of expiration will be November 27, 2001.

Abstract

At this moment, there are two resource record types defined for holding IPv6 address in the DNS database; AAAA [Thomson, 1995] and A6 [Crawford, 2000] . AAAA has been used for IPv6 network operation since 1996. Questions arose whether we really need A6 or not, or whether it is really possible to migrate to A6 or not. Some says AAAA is enough and A6 is not necessary. Some says A6 is necessary and AAAA should get deprecated.

The draft tries to understand pros and cons between these two record types, and makes suggestions on deployment of IPv6 record type.

The draft does not cover the use of bit string label and DNAME resource record (reverse mapping), as it seems that nibble form is well accepted in the community, newer formats have too much deployment costs, thus we see few need/voice that calls for migration. Refer to IETF50 dnsext

working group minutes for more details.

1. A brief summary of the IPv6 resource record types

1.1. AAAA record

AAAA resource record is formatted as follows. DNS record type value for AAAA is 28 (assigned by IANA). Note that AAAA record is formatted as a fixed-length data.

```
     +------------+
     |IPv6 Address|
     | (16 octets)|
     +------------+
```

With AAAA, we can define DNS records for IPv6 address resolution as follows, just like A records for IPv4.

```
     $ORIGIN X.EXAMPLE.
     N            AAAA 2345:00C1:CA11:0001:1234:5678:9ABC:DEF0
     N            AAAA 2345:00D2:DA11:0001:1234:5678:9ABC:DEF0
     N            AAAA 2345:000E:EB22:0001:1234:5678:9ABC:DEF0
```

1.2. A6 record

A6 resource record is formatted as follows. DNS record type value for A6 is 38 (assigned by IANA). Note that A6 record is formatted as a variable-length data.

```
     +-----------+------------------+-------------------+
     |Prefix len.|  Address suffix  |    Prefix name    |
     | (1 octet) |  (0..16 octets)  |  (0..255 octets)  |
     +-----------+------------------+-------------------+
```

With A6, it is possible to define an IPv6 address by using multiple DNS records. Here is an example taken from RFC2874:

```
     $ORIGIN X.EXAMPLE.
     N            A6 64 ::1234:5678:9ABC:DEF0 SUBNET-1.IP6
     SUBNET-1.IP6 A6 48 0:0:0:1::  IP6
     IP6          A6 48 0::0       SUBSCRIBER-X.IP6.A.NET.
     IP6          A6 48 0::0       SUBSCRIBER-X.IP6.B.NET.

     SUBSCRIBER-X.IP6.A.NET. A6 40 0:0:0011:: A.NET.IP6.C.NET.
     SUBSCRIBER-X.IP6.A.NET. A6 40 0:0:0011:: A.NET.IP6.D.NET.

     SUBSCRIBER-X.IP6.B.NET. A6 40 0:0:0022:: B-NET.IP6.E.NET.

     A.NET.IP6.C.NET. A6 28 0:0001:CA00:: C.NET.ALPHA-TLA.ORG.

     A.NET.IP6.D.NET. A6 28 0:0002:DA00:: D.NET.ALPHA-TLA.ORG.

     B-NET.IP6.E.NET. A6 32 0:0:EB00::    E.NET.ALPHA-TLA.ORG.

     C.NET.ALPHA-TLA.ORG. A6 0 2345:00C0::
     D.NET.ALPHA-TLA.ORG. A6 0 2345:00D0::
     E.NET.ALPHA-TLA.ORG. A6 0 2345:000E::
```

If we translate the above into AAAA records, it will be as follows:

```
     $ORIGIN X.EXAMPLE.
     N            AAAA 2345:00C1:CA11:0001:1234:5678:9ABC:DEF0
     N            AAAA 2345:00D2:DA11:0001:1234:5678:9ABC:DEF0
     N            AAAA 2345:000E:EB22:0001:1234:5678:9ABC:DEF0
```

It is also possible to use A6 records in ``non-fragmented'' manner, like below.

```
     $ORIGIN X.EXAMPLE.
     N            A6 0 2345:00C1:CA11:0001:1234:5678:9ABC:DEF0
     N            A6 0 2345:00D2:DA11:0001:1234:5678:9ABC:DEF0
     N            A6 0 2345:000E:EB22:0001:1234:5678:9ABC:DEF0
```

There is a large design difference between A6 and AAAA. A6 imposes address resolutions tasks more to the resolver side, to reduce the amount of zone file maintenance cost. The complexity is in the resolver side. AAAA asks zone file maintainers to supply the full 128bit IPv6 address in one record, and the resolver side can be implemented very simple.

2. Deployment status

2.1. Name servers/resolvers

As of writing, AAAA is deployed pretty widely. BIND4 (since 4.9.4), BIND8, BIND9 and other implementations support AAAA, as both DNS servers and as resolver libraries. On the contrary, the author knows of only one DNS server/resolver implementation that supports A6; BIND9.

Almost all of the IPv6-ready operating systems ship with BIND4 or BIND8 resolver library. Even for the implementations that use resolver library other than BIND, they only support AAAA. Therefore, they cannot query A6 records (unless aplications gets linked with BIND9 libraries explicitly).

2.2. IPv6 network

IPv6 network has been deployed widely since 1996. Though many of the participants consider it to be experimental, commercial IPv6 services has been deployed since around 1999, especially in asian countries.

2.3. DNS database

There are no IPv6-reachable root DNS servers, partly because we have both AAAA and A6, and we are not decided about which is the one we would like to really deploy (so we cannot put IPv6 root NS records). The lack of IPv6-reachable root DNS servers is now preventing IPv6-only or IPv4/v6 dual stack network operations.

At this moment, very small number of ccTLD registries accept registeration requests for IPv6 glue records. Many of the ccTLDs and gTLDs do not take IPv6 glue records, partly because of the lack of consensus between AAAA and A6. Again, the lack of IPv6 glue records is causing pain in IPv6-ready network operations. For example, JP ccTLD accepts IPv6 glue records and registers them as AAAA records. IPv6 NS records (with AAAA) works flawlessly from our experiences. For example, try the following commands to see how JP ccTLD registers IPv6 glue records (``/e'' is for English-language output):

```
% whois -h whois.nic.ad.jp wide.ad.jp/e
% whois -h whois.nic.ad.jp ns1.v6.wide.ad.jp/e
```

3. Deploying DNS records

At this moment, the following four strategies are proposed for the deployment of IPv6 DNS resource record; AAAA, fragmented A6 records, non-fragmented A6 records, and AAAA synthesis.

3.1. AAAA records

AAAA records have been used on IPv6 network (also known as 6bone) since it has started in 1996 and has been working just fine ever since. AAAA record is a straight extension of A records; it needs a single query-response roundtrip to resolve a name into an IPv6 address.

A6 was proposed to add network renumbering friendliness to AAAA. With AAAA, a full 128bit IPv6 address needs to be supplied in a DNS resource record. Therefore, in the event of network renumber, administrators need to update the whole DNS zone file with the new IPv6 address prefixes. We will discuss the issues with renumbering in a dedicated section.

3.2. Fragmented A6 records

If we are to use fragmented A6s (128bit splitted into multiple A6s), we have a lot of issues/worries.

If we are to resolve IPv6 addresses using fragmented A6 records, we need to query DNS multiple times to resolve a single DNS name into an IPv6 address. Since there has been no DNS record types that require multiple queries for resolution of the address itself, we have very little experience on such resource records.

There will be more delays in name resolution compared to queries to A/AAAA records. If we define a record with more number of fragments, there will be more query roundtrips. There are only few possibilities to query fragments in parallel. In the above example, we can resolve A.NET.IP6.C.NET and A.NET.IP6.D.NET in parallel, but not others.

If the DNS record TTL is smaller than the communication delays between the querier and the DNS servers, the querier will not be able to reassemble a complete IPv6 record. With the following records, you cannot reassemble an IPv6 address if the delay is larger than 1 second. By the time you get the topmost 64 bits in the address, the lowermost 64 bits get invalidated. If the querier tries to query lowermost 64 bits again, the querier can go into an infinite loop (see the section on security considration). [BIND 9.2.0snap: resolver does not go into infinite loop, meaning that BIND 9.2.0snap resolver does not really honor DNS record TTL during A6 reassembly]

```
     ; resolver can go into an infinite loop if RTT > 1 second
     $ORIGIN X.EXAMPLE.
     $TTL 1
     N            A6 64 ::1234:5678:9ABC:DEF0 SUBNET-1.IP6
     SUBNET-1.IP6 A6 0  2345:00C1:CA11:1::
```

Some says that caches will avoid querying fragmented A6s again and again. However, most of the library resolver implementations do not cache anything. The traffic between library resolver and the first-hop nameserver will not be decreased by the cached records. The TTL problem (see above) is unavoidable for the library resolver without cache. [XXX will they interpret TTL field? BIND8 resolver does not]

If some of the fragments are DNSSEC-signed and some are not, how should we treat that address? RFC2874 section 6 briefly talks about it, not sure if complete answer is given.

It is much harder to implement A6 fragment reassemble code, than to implement AAAA record resolver. AAAA record resolver can be implemented as a straight extension of A record resolver.

- It is much harder to design timeout handling for the A6 reassembly. There would be multiple timeout parameters, including (1) communcation timeout for a single A6 fragment, (2) communcation timeout for the IPv6 address itself (total time needed for reassembly) and (3) TTL timeout for A6 fragment records.

- In the case of library resolver implementation, it is harder to deal with exceptions (signals in UNIX case) for the large code fragment for resolvers.

- When A6 prefix length field is not multiple of 8, address suffix portion needs to be shifted bitwise while A6 fragments are reassembled. Also, resolver implementations must be careful about overwraps of the bits. From our implementatation experiences, the logic gets very complex and we (unfortunately) expect to see a lot of security-critical bugs in the future.

In RFC2874, a suggestion is made to use limited number of fragments per an IPv6 address. However, there is no protocol limitation defined. The lack makes it easier for malicious parties to impose DoS attacks using lots of A6 fragments (see the section on security consideration). [BIND 9.2.0snap: The implementation limits the number of fragments to be smaller than 16; It is not a protocol limitation but an implementation choice. Not sure if it is the right choice or not]

With fragmented A6 records, in multi-prefix network configuration, it is not possible for us to limit the address on the DNS database to the specific set of records, like for load distribution purposes. Consider the following example. Even if we would like to advertise only 2345:00D2:DA11:1:1234:5678:9ABC:DEF0 for N.X.EXAMPLE, it is not possible to do so. It becomes mandatory for us to define the whole IPv6 address by using ``A6 0'' for N.X.EXAMPLE, and in effect, the benefit of A6 (renumber friendliness) goes away.

```
     ; with the following record we would advertise both records
     $ORIGIN X.EXAMPLE.
     N             A6 64 ::1234:5678:9ABC:DEF0 SUBNET-1.IP6
     M             A6 64 ::2345:2345:2345:2345 SUBNET-1.IP6
     SUBNET-1.IP6 A6 0  2345:00C1:CA11:1::
                   A6 0  2345:00D2:DA11:1::

     ; we need to do the following, jeopardizing renumbering
     ; friendliness for N.X.EXAMPLE
     $ORIGIN X.EXAMPLE.
     N             A6 0 2345:00C1:CA11:1:1234:5678:9ABC:DEF0
     M             A6 64 ::2345:2345:2345:2345 SUBNET-1.IP6
     SUBNET-1.IP6 A6 0  2345:00C1:CA11:1::
                   A6 0  2345:00D2:DA11:1::
```

A6 resource record type and A6 fragment/reassembly were introduced to help administrators on network renumber. When network gets renumbered, the administrator needs to update A6 fragment for the higher address bits (prefixes) only. Again, we will discuss the issues with renumbering in a dedicated section.

3.3. Non-fragmented A6 records

There are proposals to use non-fragmented A6 records in most of the places, like ``A6 0 <128bit>'', so that we would be able to switch to fragmented A6 records when we find a need for A6.

>From the packet format point of view, the approach has no benefit against AAAA. Rather, there is a one-byte overhead to every (unfragmented) A6 record compared to an AAAA record.

If the nameserver/resolver programs hardcode A6 processing to handle no fragments, there will be no future possibility for us to introduce fragmented A6 records. When there is no need for A6 reassembly, there will be no code deployment, and even if the reassembly code gets deployed they will not be tested enough. The author believes that the ``prepare for the future, use non-fragmented A6'' argument is not worthwhile.

In the event of renumbering, non-fragmented A6 record has the same property as AAAA (the whole zone file has to be updated).

3.4. AAAA synthesis (A6 and AAAA hybrid approach)

At this moment, end hosts support AAAA records only. Some people would like to see A6 deployment in DNS databases even with the lack of end hosts support. To workaround the deployment issues of A6, the following approach is proposed in IETF50 dnsext working group slot. It is called ``AAAA synthesis'':

o Deploy A6 DNS records worldwide. The proposal was not specific about whether we would deploy fragmented A6 records, or non-fragmented A6 records (``A6 0'').

o When a host queries AAAA record to a DNS server, the DNS server queries A6 fragments, reassemble it, and respond with an AAAA record.

The approach needs to be diagnosed/specified in far more detail. For example, the following questions need to be answered:

o What is the DNS error code against AAAA querier, if the A6 reassembly fails?

o What TTL should the synthesized AAAA record have? [BIND 9.2.0snap uses TTL=0]

o Which nameserver should synthesize the AAAA record, in the DNS recursize query chain? Is the synthesis mandatory for every DNS server implementation?

o What should we do if the A6 reassembly takes too much time?

o What should we do about DNSSEC signatures?

o What if the resolver wants no synthesis? Do we want to have a flag bit in DNS packet, to enable/disable AAAA synthesis?

o Relationships between A6 TTL, AAAA TTL, A6 query timeouts, AAAA query timeouts, and other timeout parameters?

The approach seems to be vulnerable against DoS attacks, because the nameserver reassembles A6 fragments on behalf of the AAAA querier. See security consideration section for more details.

3.5. Issues in keeping both AAAA and A6

If we are to keep both AAAA and A6 records onto the worldwide DNS database, it would impose more query delays to the client resolvers. Suppose we have a dual-stack host implementation. If they need to resolve a name into addresses, the node would need to query in the following order (in the order which RFC2874 suggests):

o Query A6 records, and get full IPv6 addresses by chasing and reassembling A6 fragment chain.

o Query AAAA records.

o Query A records.

o Sort the result based on destination address ordering rule. An example of the ordering rule is presented as a draft [Draves, 2000] .

o Contact the destination addresses in sequence.

The ordering imposes additional delays to the resolvers. The above ordering would be necessary for all approaches that use A6, as there are existing AAAA records in the world.

4. Network renumbering

Some says that there will be more frequent renumbers in IPv6 network operation, and A6 is necessary to reduce the zone modification cost as well as zone signing costs on renumber operation.

It is not clear if we really want to renumber that frequently. With IPv6, it should be easier for ISPs to assign addresses statically to the downstream customers, rather than dynamically like we do in IPv4 dialup connectivity today. If ISPs do assign static IPv6 address block to the customers, there is no need to renumber customer network that frequently

(unless the customer decides to switch the upstream ISPs that often).
NOTE: Roaming dialup users, like those who carry laptop computers

worldwide, seems to have a different issue from stationary dialup users. See [Hagino, 2000] for more discussions.

It is questionable if it is possible to renumber IPv6 networks more frequently than with IPv4. Router renumbering protocol [Crawford, 2000] , IPv6 host autoconfiguration and IPv6 address lifetime [Thomson, 1998] can help us renumber the IPv6 network, however, network renumbering itself is not an easy task. If you would like to maintain reachability from the outside world, a site administrator needs to carefully coordinate site renumber. The minimal interval between renumber is restricted by DNS record timeouts, as DNS records will be cached around the world. If the TTL of DNS records are X, the interval between renumber must be longer than 2 * X. If we consider clients/servers that tries to validate addresses using reverse lookups, we also need to care about the relationship between IPv6 address lifetime [Thomson, 1998] and the interval between renumber. At IETF50 ipngwg session, there was a presentation by JINMEI Tatsuya regarding to site renumbering experiment. It is recommend to read through the IETF49 minutes and slides. [XXX Fred Baker had a draft on this - where?] For the network renumbering to be successful, no configuration files should have hardcoded (numeric) IP addresses. It is a very hard requrement to meet. We fail to satisfy this in many of the network renumbering events, and the failure causes a lot of troubles.

At this moment there is no mechanism defined for ISPs to renumber downstream customers at will. Router renumbering protocol [Crawford, 2000] is defined only for a single administrative domain (customers will not want to share IPsec authentication key for routers, with the upstream ISP). Even though it may sound interesting for ISPs, it would cause a lot of (social and political) issues in doing so, so the author would say it is rather unrealistic to pursue this route.

A6 was designed to help administators update zone files during network renumbering events. Even with AAAA, zone file modification itself is easy; you can just query-replace the addresses in the zone files. The difficulty of network renumber comes from elsewhere.

With AAAA, we need to sign the whole zone again with DNSSEC keys, after renumbering the network. With A6, we need to sign upper bits only with DNSSEC keys. Therefore, A6 will impose less zone signing cost on the event of network renumbering. As seen above, it is questionable if we renumber network that often, so it is questionable if A6 brings us an overall benefit.

5. Security consideration

There are a couple of security worries mentioned in the above. To give a brief summary:

- There will be a higher delay imposed by query/reply roundtrips for fragmented A6 records. This could affect every services that relies upon DNS records.

- There is no upper limit defined for the number of A6 fragments for defining an IPv6 address. Malicious parties may try to put a very complex A6 chains and confuse nameservers worldwide.

- A6 resolver/nameserver is much harder to implement correctly than AAAA resolver/nameserver. A6 fragment reassembly code needs to take care of bitwise data reassembly, bitwise overwrap checks, and others. From our implementatation experiences, we expect to see a lot of security-issue bugs in the future.

- Interaction between DNS record TTL and the DNS query delays leads to non-trivial timeout problem.

We would like to go into more details for some of these.

5.1. DoS attacks against AAAA synthesis

When a DNS server is configured for AAAA synthesis, malicious parties can impose DoS attacks using the interaction between DNS TTL and query delays. The attack can chew CPU time and/or memory, as well as some network bandwidth on a victim nameserver, by the following steps:

- Bad guy configures a record with very complex A6 chain, onto some nameservers. (bad guy has to have controls over the servers). The nameservers can be located anywhere in the world. The A6 chain should have a very low TTL (like 1 or 0 seconds). The attack works better if we have higher delays between the victim nameservers and the nameservers that serve A6 fragments.

- Bad guy queries the record using AAAA request, to the victim nameserver.

- The victim nameserver will try to reassemble A6 fragments. During the reassembly process, the victim nameserver puts A6 fragments into the local cache. The cached records will expire during the reassembly process. The nameserver will need to query a lot of A6 fragments (more traffic). The server can go into an infinite loop, if it tries to query the expired A6 fragments again.

To remedy this problem, we have a couple of solutions:

(1) Deprecate A6 and deploy AAAA worldwide. If we do not have A6, the problem goes away.

(2) Even if we use A6, do not configure nameservers for AAAA synthesis. Deployment issues with existing IPv6 hosts get much harder.

(3) Impose a protocol limitation to the number of A6 fragments.

(4) Do not query the expired records in A6 chain again. In other words, implement resolvers that ignore TTL on DNS records. Not sure if it is the right thing to do.

6. Conclusion

NOTE: the section expresses the impressions of the author.

A6/AAAA discussion has been an obstacle for IPv6 deployment, as the deployment of IPv6 NS recodrs have been deferred because of the discussion. The author do not see benefit in keeping both AAAA and A6 records, as it imposes more query delays to the clients. So the author believes that we need to pick one of them.

Given the unlikeliness of frequent network renumbering, the author believes that the A6's benefit in lower zone signing cost is not significant. The benefit of A6 (in zone signing cost) is much less than the expected complication that will be imposed by A6 operations.

>From the above discussions, the author suggests to keep AAAA and deprecate A6. The author believes that A6 can cause a lot of problem than the benefits it may have. A6 will make IPv6 DNS operation more complicated and vulnerable to attacks. AAAA is proven to work right in our IPv6 network operation since 1996. AAAA has been working just fine in existing IPv6 networks, and the author believes that it will in the coming days.

References

Thomson, 1995.
S. Thomson and C. Huitema, "DNS Extensions to support IP version 6" in RFC1886 (December 1995). ftp://ftp.isi.edu/in-notes/rfc1886.txt.

Crawford, 2000.
M. Crawford, C. Huitema, and S. Thomson, "DNS Extensions to Support IPv6 Address Aggregation and Renumbering" in RFC2874 (July 2000). ftp://ftp.isi.edu/in-notes/rfc2874.txt.

Draves, 2000.
Richard Draves, "Default Address Selection for IPv6," Internet Draft (November 2000). work in progress material.

Hagino, 2000.
Jun-ichiro Hagino and Kazu Yamamoto, "Requirements for IPv6 dialup PPP operation" in draft-itojun-ipv6-dialup-requirement-00.txt (July 2000). work in progress material.

Crawford, 2000.
Matt Crawford, "Router Renumbering for IPv6" in RFC2894 (August 2000).

ftp://ftp.isi.edu/in-notes/rfc2894.txt.

Thomson, 1998.
S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration" in
RFC2462 (December 1998). ftp://ftp.isi.edu/in-notes/rfc2462.txt.

Change history

none.

Acknowledgements

The draft was written based on discussions in IETF IPv6 and dnsext working groups, and help from WIDE research group.

Author's address

Jun-ichiro itojun HAGINO
Research Laboratory, Internet Initiative Japan Inc.
Takebashi Yasuda Bldg.,
3-13 Kanda Nishiki-cho,
Chiyoda-ku,Tokyo 101-0054, JAPAN
Tel: +81-3-5259-6350
Fax: +81-3-5259-6351
Email: itojun@iijlab.net