Network Working Group
Internet-Draft
Expires: May 13, 2002

J-F. Boudreault
Marc Blanchet
Viagenie inc.
November 13, 2001

Reference ID for NTPv6
draft-boudreault-ipv6-ntp-refid-00.txt

Status of this Memo

This Internet-Draft will expire on May 13, 2002.

Copyright Notice

Abstract

This document proposes a solution to solve the reference ID issue for IPv6 NTP secondary server.

Table of Contents

1. Definition of reference ID in NTP protocol

According to RFC 1305 (Network Time Protocol Version 3), Reference Clock Identifier "is a 32-bit code identifying the particular reference clock. In the case of stratum 0 (unspecified) of stratum 1 (primary reference), this is a four-octet, left-justified, zero-padded ASCII string. [...] In the case of stratum 2 and greater (secondary reference) this is the four-octet Internet address of the primary reference host." Reference ID is included as a 32-bits field in every NTP packet.

This reference ID is essential to avoid 'loopback' during the selection of synchronization source for NTP server. A 'loopback' happens when a NTP server choose a reference source that is a secondary NTP server who use itself as a source. This result in false synchronization. Reference ID being the IPv4 address of the reference source, one can avoid 'loopback' by comparing this field (from NTP server reply packet) with our IPv4 address, and reject the potential source if they are equal.

2. Problem Statement

The Reference ID is a 32 bits field in the NTP packet definition. It is enough for IPv4 32 bits addresses but not enough for IPv6 128 bits addresses. There is a problem if a secondary server is using a IPv6-only NTP server as reference source. We need a way to avoid 'loopback' for synchronization of secondary NTP server with a IPv6-only NTP server.

In this document we present our understanding of three possible solutions. The first solution is to change the size of reference ID field in NTP packet definition from 32 bits to 128 bits. The second solution is to use timestamps informations to avoid 'loopback'. The third solution is to apply an algorithm the IPv6 address to compress it to 32 bits and use these IPv6-compressed address in reference ID.

3. Protocol Change

3.1 Description

A solution is to change the reference ID field from 32 bits to 128 bits. This way one could put IPv6 source as reference ID and IPv6 embedded IPv4 for IPv4 source. We need to change NTP packet definition to modify reference ID field from 32 bits to 128 bits.

3.2 Impacts

By using a new protocol definition for ntp packet, this breaks the support for old version of ntp. To support them, one needs to implement both versions of ntp packet, one with 32 bits reference id and one with 128 bits reference id. When a server receives a packet, it first puts it in a buffer structure to look at the version field, and decide which structure to use according to ntp version. When server will reply to request, it will need to make sure it respond with the same version and use the correct packet definition that the client support. If a secondary server using IPv6 reference source receive a request from an old version client, it will need to ignore it because it will not be able to put his reference id in the old ntp packet definition. This will need an important redefinition of ntp specifications.

3.3 Implementation

To change the size of reference ID, one needs to modify the definition of NTP packet structure. This will have big impact on compatiblity with older versions of NTP. To maintain this compatiblity, one will need to support old packet format. It means that it needs to implement both NTP packet structure definitions, one with a 32 bits reference ID field and one with a 128 bits reference ID field.

In reception procedure, the server will need to first put the packet in a NTP structure buffer, analyze the version of the packet and put it in the correct NTP packet structure.

In transmission procedure, the server will need to first know the NTP version of the destination, and put informations in correct packet format. In every procedure that use reference ID, one will need to support both versions and test every time to know which version is used.

4. Using Timestamps

4.1 Description

This was first proposed by David L. Mills in RFC 2030 (Simple Network Time Protocol Version 4) to resolve reference id problem for IPv6 source. By using other informations from ntp packet, one could avoid loopback by comparing timestamps of the last transmitted packet from reference source and originate timestamp of the reply from ntp server. When a reference source is selected, each time ntp server receive a packet from this source, it will update his reference id to the low 32 bits of transmit timestamp field from this ntp packet.

When a ntp server is looking for synchronization source, it will compare the low 32 bits from originate timestamp field of ntp reply packet (this will correspond to his transmit timestamp from request) with reference id field. If they are equal, that is because this server updated his reference id after received the request packet, and that means that it is using this server as synchronization source. In this case it needs to be discarded.

4.2 Impacts

It could append that a ntp request packet is send to a ntp server at the same time than the reference source of this server send it a packet. By this way, the ntp server will adjust his reference id to transmit time from his ntp source. If this value is the same than the transmit timestamp of ntp request packet, the requester will reject this server even if it is not using itself as a source. This way could reject a good potential source of synchronization.

4.3 Implementation

In clock update procedure, when the stratum of system is more than 1 (secondary server), the server will put the low 32 bits from transmit timestamp of the last received NTP packet from reference source.

In clock selection procedure, the server needs to modifiy loopback detection test to compare reference ID field with originate timestamp field.

In reception procedure, every time the server receive a packet from our reference source, it will need to update reference ID with the low 32 bits of the transmit timestamp field.

Depending on the exact implementation of NTP, one need to be sure we can obtain the informations we need to use this solution. That could be a problem in some implementations.

5. IPv6 Address Compression

5.1 Description

By applying a compression algorithm, one could compress

IPv6 addresses to 32 bits and put the compressed address as the Reference Id. The proposed way is to ignore some bits because of some assumptions in the use of these bits in the IPv6 address.

In order to avoid to overlap the IPv4 address space in the reference ID field, the leftmost 3 bits of the Reference ID field is '111'b when an IPv6 address is compressed in the reference ID. '111' in the first 3 bits correspond to the Class D and E IPv4 addresses. Since this address space is not used for Reference ID field, this gives no overlap between the two spaces.

The IPv6 address architecture [RFC2373] defines specific boundaries in the address as shown below.

```
| 3 |  13 |          32          |   16   |    64 bits      |
+---+-----+---------------------+---------+-----------------+
|001| TLA |       NLA ID        | SLA ID | Interface ID    |
+---+-----+---------------------+---------+-----------------+
```

The last 64 bits is the interface ID. If the NTP server uses autoconfig with a unique layer2 identifier, then this is identified by a "uniqueness" bit in the EUI-64 format used in the Interface ID. This identifier is used for the Reference ID. If the identifier is more than 32 bits (for example: Ethernet 48 bits), then the last rightmost 28 bits (32 - 3 - 1 bits) are used for the Reference ID. and one bit is 1 for identifying a unique address.

If the NTP server is not having a unique interface identifier in its address, then the following assumptions are made:
- we expect that most NTP server sites that are interconnected will only have one or a few NTP servers. This means that the differentiator is mostly based on the identification of the site instead of the host. The algorithm will try to use most of the site identification (the first 48 bits) instead of the subnet id (the next 16 bits) or interface id (the last 64 bits).
- because of the compression of 0 in writing IPv6 addresses makes easier to put most zeros in a manually assigned address, then most of the leftmost bits in the rightmost 64 bits of the address will be all zeros. So this compression algorithm will only use the rightmost 4 bits of the rightmost 64 bits of the address.
- for the same reason of zero compression for writing, the SLA ID will usually have many zeros at the left. This algorithm only uses the rightmost 4 bits of this field.
- since the current addressing architecture is only defined for the 001 as the first three bits, these bits are redundant and not used.
- this leaves us with 45 bits of TLA/NLA bits and only 32-3-4-4=21

bits left. For no real reasons except the basis of the current allocation policies, this algorithm uses the rightmost 4 bits of the TLA field and the rightmost 17 bits of the NLA field.

The chosen bits are identified with the number of bits below the following figure.

```
| 3 |  13 |          32          |   16   |    64 bits      |
+---+-----+---------------------+--------+-----------------+
|001| TLA |       NLA ID        | SLA ID | Interface ID    |
+---+-----+---------------------+--------+-----------------+
        |4|         |    16     |      |4|               |4|
```

5.2 Algorithm

5.2.1 Unique Interface ID of the IPv6 address based on EUI-64

If the IPv6 address has the "uniqueness" bit on, then the reference ID is composed of the following fields:

```
| 4  |  4 |       24                |
+----+----+-------------------------+
|1111| A  |        B                |
+----+----+-------------------------+
|                 32                |
```

The compression algorithm is the following:
- the first leftmost 4 bits are '1111'b.
- the next 4 bits, identified as A in the previous figure, are the 16-19 bits of the Interface ID of the IPv6 address, counting with 0 as the leftmost bit of the Interface ID field (of 64 bits).
- the next 24 bits, identified as B in the previous figure, are the rightmost 24 bits of the IPv6 address.

TBD: discussion on 64 bits EUI-64 identifiers

5.2.2 Non unique Interface ID

The reference ID is then composed of the following fields:

```
| 4  |  4 |       16        | 4  |  4 |
+----+----+-----------------+----+----+
|1110| A  |        B        | C  | D  |
+----+----+-----------------+----+----+
|                32                   |
```

The compression algorithm is the following:
- the rightmost 3 bits are '111'b
- the next field, identified as A, is of 4 bits and the bits are the 12-15th bits of the IPv6 address, starting with zero as the leftmost bit.

- the next field, identified as B, is of 16 bits and the bits are the 42-47th bits of the IPv6 address, starting with zero as the leftmost bit.
- the next field, identified as C, is of 4 bits and the bits are the 60-63th bits of the IPv6 address, starting with zero as the leftmost bit.
- the last field, identified as D, is of 4 bits and the bits are the rightmost 4 bits of the IPv6 address.

5.2 Impacts

This compression algorithm is obviously going to produce some false hits. Our current understanding of the previous proposal based on Timestamps tells us that that proposal also exhibits some false hits. Which one is more probable than the other is difficult to guess. We consider for now essentially equal unless the contrary is demonstrated.

By compressing IPv6 address, it could append that two IPv6 addresses give the same 32-bits compressed address. If this append, a ntp server could reject a source that use a reference server with the same IPv6-compressed address. So we could reject a good potential source of synchronization due to IPv6-compressed addresses conflict.

TBD: discussion on other addresses: link-local, site-local,
        ipv4-compatible, 6to4, ...

5.3 Implementation

Each time one uses IPv6 address to create or compare reference id, we just need to apply the algorithm to obtain 32 bits IPv6-compressed address that will fit the field. By this way we will be able to avoid 'loopback' by comparing reference source IPv6-compressed address with our IPv6-compressed address. They will be the same if we are the reference source and we apply the same algorithm to compress IPv6 address.

Every time we set reference ID from IP address or compare IP address with reference ID, we will apply a algorithm to the IP address. This algorithm will return a 32 bits address corresponding to the complete IPv4 address or compressed IPv6 address.

The key advantage of this proposal is the fact that it is very easy to compute the reference ID. In fact, it has been implemented with a small C macro on the current IPv6 branch of the ntpd code.

It also brings backward compatibility with IPv4 servers already in place.

6. Discussion on Solutions

There is no 'perfect' solution for this problem. Every solutions has some consequences. In both case of using timestamps or IPv6 compressed address, we could have conflicts and reject good potential source of reference. And changing of the protocol definition without braking compatibility with older version has important implications for deployment and version negociation.

We believe that using IPv6-compressed address is the better solution. Being easy to implement, we could minimise the possiblity of conflict with a good compression algorithm.

7. Acknowledgements

TBD

8. References

TBD

9. Authors' Addresses

Jean-Francois Boudreault
Viagenie inc.
2875 boul. Laurier, bureau 300
Sainte-Foy, QC G1V 2M2
Canada

Phone: +1 418 656 9254
EMail: Jean-Francois.Boudreault@viagenie.qc.ca
URI: http://www.viagenie.qc.ca/

Marc Blanchet
Viagenie inc.
2875 boul. Laurier, bureau 300
Sainte-Foy, QC G1V 2M2
Canada

Phone: +1 418 656 9254
EMail: Marc.Blanchet@viagenie.qc.ca
URI: http://www.viagenie.qc.ca/

Acknowledgement

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