Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 2013
H. Kitamura
NEC Corporation
S. Ata
Osaka City University
October 16, 2012

Corresponding Auto Names for IPv6 Addresses

<draft-kitamura-ipv6-auto-name-03.txt>

Abstract

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses. With this mechanism, all IPv6 addresses (even if they are link-local scoped addresses) can obtain their own Names, and it will be able to use Names anywhere instead of IPv6 Addresses.

IPv6 address is too long and complicated to remember, and it is very nuisance to type a literal IPv6 address manually as an argument of applications. Also, it is very difficult for human beings to tell which IPv6 address is set to which actual IPv6 node. In this sense, literal IPv6 address information can be called meaningless information for human beings.

In order to solve above problems and to provide annotated meaningful information to IPv6 addresses, mechanisms called Corresponding Auto Names for IPv6 addresses is introduced. They will become human friendly information. By applying a simple naming rule to the Auto Names (e.g., use the same Auto Name Suffix for IPv6 addresses that are set to the same interface (node)), this will contribute to help people to understand which IPv6 address / Name indicates which actual IPv6 node and provide meaningful information.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses.

IPv6 address is too long and complicated to remember. For human being, values of EUI64-based or temporary addresses should be felt that they are random number series. So, it is very nuisance to type a literal IPv6 address manually as an argument of applications.

Furthermore, it is very normal and popular cases to set multiple IPv6 addresses to one node. One IPv6 node owns more than two IPv6 addresses (typically: one is link-local scoped address. the other is global scoped address) at least. Some IPv6 addresses (such as link-local scoped stateless auto-configuration addresses and temporary addresses) may become users' conscious-less address, because they are automatically set to the IPv6 node.

It is too difficult for human beings to tell which IPv6 address is set to which IPv6 node. In other words, when an IPv6 address is shown to a person, he almost can not tell that the shown IPv6 address indicates which IPv6 node. In this sense, literal IPv6 address information can be called useless or meaningless information for human beings.

Moreover, when more than two literal IPv6 addresses are shown to human beings, it is almost impossible to distinguish them whether they are same or not at a glance.

So, there are strong desires to use Name information (that is human friendly) instead of literal IPv6 Address information and to use meaningful and distinguishable information that can easily show which IPv6 address / Name indicates which actual IPv6 node.

The Corresponding Auto Names for IPv6 Addresses is introduced to solve above problems and to satisfy the above desires.

2. Goals (What can be achieved)

In this section, goals of the mechanisms of the Corresponding Auto Names for IPv6 Addresses and what can be achieved are shown by using examples.

2.1. Assumed typical IPv6 communication environment:

Two IPv6 nodes (Node A and Node B) are located on the same link. Their IPv6 Addresses are shown below.

```
    Node A:        Literal Address
                   ----------------------------------
     MAC Address:  00:0d:5e:b8:80:7b
                   ----------------------------------
     LL-Address:   fe80::20d:5eff:feb8:807b%fxp0
     ULA:          fd01:2345:6789::20d:5eff:feb8:807b
                   fd01:2345:6789::1234
     Global Addr:  2001:db8::20d:5eff:feb8:807b
                   2001:db8::1234

    Node B:        Literal Address
                   ----------------------------------
     MAC Address:  00:0c:76:d9:14:e3
                   ----------------------------------
     LL-Address:   fe80::20c:76ff:fed9:14e3%em0
     ULA:          fd01:2345:6789::20c:76ff:fed9:14e3
                   fd01:2345:6789::5678
     Global Addr:  2001:db8::20c:76ff:fed9:14e3
                   2001:db8::5678
```

They own altogether 5 IPv6 addresses respectively;
  One Link-Local scoped Address
  Two Unique Local Addresses (SLLAC and manual set address)
  Two Global scoped Addresses (SLLAC and manual set address)
They communicate each other.

2.2. Auto Names examples

For all addresses, respective Corresponding Auto Names are prepared and registered to a some name resolving service DB (typically the DNS is used for this) automatically by the mechanism that detects these addresses (that is explained after in this document). Prepared Auto Names are shown below.

```
    Node A:        Literal Address                        Auto Name
                   ----------------------------------     ---------
     MAC Address:  00:0d:5e:b8:80:7b
                   ----------------------------------
     LL-Address:   fe80::20d:5eff:feb8:807b%fxp0       -> L0-7bz%fxp0
     ULA:          fd01:2345:6789::20d:5eff:feb8:807b  -> U0-7bz
                   fd01:2345:6789::1234                -> U1-7bz
     Global Addr:  2001:DB8::20d:5eff:feb8:807b        -> G0-7bz
                   2001:DB8::1234                      -> G1-7bz

    Node B:        Literal Address                        Auto Name
                   ----------------------------------     ---------
     MAC Address:  00:0c:76:d9:14:e3
                   ----------------------------------
     LL-Address:   fe80::20c:76ff:fed9:14e3%em0        -> L0-3ez%em0
     ULA:          fd01:2345:6789::20c:76ff:fed9:14e3  -> U0-3ez
                   fd01:2345:6789::5678                -> U1-3ez
     Global Addr:  2001:DB8::20c:76ff:fed9:14e3        -> G0-3ez
                   2001:DB8::5678                      -> G1-3ez
```

2.3. Auto Name Suffix for Grouped Addresses

In order to make Auto Names meaningful, IPv6 addresses are grouped and Auto Name Suffix is used to show grouped addresses.

For IPv6 addresses that are set to the same interface (node), the same Auto Name Suffix that stands for the Group ID is used for their Auto Names.

As shown above:

'-7bz' is used for Auto Name Suffix (Group ID) for Node A.
'-3ez' is used for Auto Name Suffix (Group ID) for Node B.

In order to make easier to identify and remember the Auto Name Suffixes, their naming rule is based on inheriting the last octet of the node's MAC address in this example.

2.4. Contribution in Regular Resolving (Name -> Address)

In order to communicate with the specific IPv6 address of the destination node, the following procedure to type literal IPv6 address is required in the current environment. They are very stressful and nuisance procedures for human beings.

When 'ping6' or 'telnet' to the specific IPv6 address of Node B from Node A is executed, the following commands are typed.

```
>ping6  fe80::20c:76ff:fed9:14e3%fxp0
>telnet fd01:2345:6789::20c:76ff:fed9:14e3
```

Especially for link-local scoped addresses or temporary addresses, there are no way to type Names instead of literal IPv6 addresses, because they are generally not registered to name resolving services.

By introducing the Corresponding Auto Names, above typed commands are changed and replaced with the following easy and rememberalbe name typing procedures.

```
>ping6  L0-3ez%fxp0
>telnet U0-3ez
```

2.5. Contribution in Reverse Resolving (Address -> Name)

Communication related status information is shown to human beings in literal IPv6 address format in the current environment.

'netstat -a' (on Node A) shows connection status as followed:

```
Local Address                 Foreign Address              (state)
fe80::20d:5eff:feb8:807b.8722 fe80:3::20c:76ff:fed9:14e3.23 ESTABLISH
fd01:2345:6789::1234.16258    fd01:2345:6789::5678.23       TIME_WAIT
```

'ndp -a' (on Node A) shows neighbor cache status as followed:

```
Neighbor                            Linklayer Addr.  Netif Expire   S
fe80::20d:5eff:feb8:807b%fxp0       0:0d:5e:b8:80:7b fxp0 permanent R
fd01:2345:6789::20d:5eff:feb8:807b  0:0d:5e:b8:80:7b fxp0 permanent R
fd01:2345:6789::1234                0:0d:5e:b8:80:7b fxp0 permanent R
2001:DB8::20d:5eff:feb8:807b        0:0d:5e:b8:80:7b fxp0 permanent R
2001:DB8::1234                      0:0d:5e:b8:80:7b fxp0 permanent R
fe80::221:85ff:fea7:82ff%fxp0       0:21:85:a7:82:ff fxp0 23h50m51s S
fe80::20c:76ff:fed9:14e3%fxp0       0:0c:76:d9:14:e3 fxp0 23h51m56s S
fd01:2345:6789::20c:76ff:fed9:14e3  0:0c:76:d9:14:e3 fxp0 23h52m50s S
fd01:2345:6789::5678                0:0c:76:d9:14:e3 fxp0 23h53m51s S
2001:DB8::20c:76ff:fed9:14e3        0:0c:76:d9:14:e3 fxp0 23h54m53s S
2001:DB8::5678                      0:0c:76:d9:14:e3 fxp0 23h55m54s S
```

People almost can not tell which shown literal IPv6 address indicates which IPv6 node. In this sense, shown information is meaningless and useless.

By introducing the Corresponding Auto Names, above complicated information is converted into simple and meaningful information and shown as followed.

'netstat -a' (on Node A) shows connection status as followed:

```
Local Address              Foreign Address              (state)
L0-7bz.8722                L0-e3z.23                   ESTABLISH
U0-7bz.16258               U0-e3z.23                   TIME_WAIT
```

'ndp -a' (on Node A) shows neighbor cache status as followed:

```
Neighbor                            Linklayer Addr.  Netif Expire   S
L0-7bz%fxp0                         0:0d:5e:b8:80:7b fxp0 permanent R
U0-7bz                              0:0d:5e:b8:80:7b fxp0 permanent R
U1-7bz                              0:0d:5e:b8:80:7b fxp0 permanent R
G0-7bz                              0:0d:5e:b8:80:7b fxp0 permanent R
G1-7bz                              0:0d:5e:b8:80:7b fxp0 permanent R
L0-ffz%fxp0                         0:21:85:a7:82:ff fxp0 23h50m51s S
L0-3ez%fxp0                         0:0c:76:d9:14:e3 fxp0 23h51m56s S
U0-3ez                              0:0c:76:d9:14:e3 fxp0 23h52m50s S
U1-3ez                              0:0c:76:d9:14:e3 fxp0 23h53m51s S
G0-3ez                              0:0c:76:d9:14:e3 fxp0 23h54m53s S
G1-3ez                              0:0c:76:d9:14:e3 fxp0 23h55m54s S
```

Other examples where the Auto Name technique can contributes:

In log files of a server application, accesses from clients are recoded into them in literal IPv6 address format. It is almost impossible to read and understand the log files effectively without this Auto Name technique.

Also, in packet dumping applications, address information is shown in literal IPv6 address format. This Auto Name technique can significantly help for human beings to analyze and understand dumped packets.

Shown communication related status information in Auto Name format is simple and easy enough for human beings to understand. As shown above, troublesome IPv6 literal Address information can be converted into meaningful and distinguishable information by using the Corresponding Auto Names technique, and we can achieve our goals.

3. Deployed Notions and Functions that are used in Auto Names

3.1. Stateless Name

We know that we can categorize Addresses into two types. One is "stateful" address type, and the other is 'stateless' address type.

On the other, we have not been applied the same categorization to domain Names or host Names clearly. It has been assumed that existing all Names are categorized into stateful type and there is no stateless name type. Authors think that it is a time to change this preconception.

We can grasp that the introduced Corresponding Auto Name is realization of "stateless" name type, and we have deployed a notion Stateless Name clearly here.

Table 1 Stateless Name

	Stateful	Stateless
Address	DHCPv6	SLAAC
Nmae	existing Domain Names	Auto Names

3.2. Scoped Name

We also know that a notion called "scope" (such as link-local scope, global-scope) is introduced when we deal with addresses. Every address has its own scope.

In domain names or host names cases, the "scope" notion have NOT clearly introduced now. It is generally assumed that all names are global information and "scope" notion does not exist.

The Corresponding Auto Name is achieved by introducing Scoped Name obviously.

Scope of Auto Name for IPv6 address is the same to the scope of its IPv6 address. For example, scope of the Auto Name for the link-local IPv6 address is link-local. They are only effective within the link-local scope.

Table 1 Scoped Name

	Global	Site-Local(ULA)	Link-Local	Node-Local
Address	2001:db8::/64	fd01:2345:6789::/64	fe80::/64	
Nmae	Domain Names	Domain Names / Auto Names	Auto Names	Auto Names

At some special situation (that it is enough that Auto Name information is shared within a node), Node-Local scoped Auto Name is possible. At such a situation, we can use /etc/hosts file as a name service.

Deployment of Scoped Name:

Scoped Name notion can be easily achieved with current technologies.

As shown above, at a special case when we adopt /etc/hosts file as a name service for Auto Names, scope of Auto Names naturally becomes Node-Local.

At general cases when we adopt the DNS as a name service for Auto Names, scope of Auto Names is easily managed by the DNS query access permission control on DNS servers.

3.3. Target IPv6 Addresses

One of the goals of the Auto Name technique is to provide and set Names to all IPv6 addresses (include Link-local addresses).

All IPv6 Addresses are targets of Auto Names.

Some IPv6 Addresses have their own names (let's call them Proper Names) that are assigned manually and are registered into name resolving services (such as the DNS). It may be thought that it is not necessary to assign Auto Name to such IPv6 addresses which have Proper Names. However, we strongly recommend assigning Auto Names to them, too.

In order to provide meaningful information to IPv6 addresses, uniformed Name information for all IPv6 addresses is necessary. This is very natural behavior for 'Stateless' type technology.

Since one-to-multiple mapping is allowed in name resolving services, it will not cause problems to assign both Proper Name and Auto Name for one IPv6 address.

We may need some function that controls name display priority (which name is first Proper Name or Auto Name). This function also achieved easily by using existing current technologies.

If Auto Name and Proper Name are implemented as different name resolving services (e.g., one is /etc/hosts, the other is the DNS), name display priority is can be easily controlled by nsswitch.conf function.

If Auto Name and Proper Name are implemented as same name resolving service, name display priority is can be controlled by their registration order to the name resolving service DB.

4. Design of Auto Names

4.1. Conceptual Design on Naming Rules

Auto Names are composed of "<P><I>-<NGI>" format:

<P>: stands for Prefix of Address

    1 character: (e.g., 'L', 'U', 'G')

<I>: stands for Interface ID of Address

    1 character: (e.g., '0', '1')

<NGI>: stands for Node (Interface) Group ID

    3 characters:
      (e.g., '7bz', '3ez')

Above discussed Auto Name examples satisfy <P><I>-<NGI> format.

```
on Node A: L0-7bz, U0-7bz, U1-7bz, G0-7bz, G1-7bz
on Node B: L0-3ez, U0-3ez, U1-3ez, G0-3ez, G1-3ez
```

4.1.1. <P> Value:

<P> value stands for Prefix (Scope) (upper 64 bit) of Address as 1 character format.

Auto Names of IPv6 addresses whose prefixes are same use the same <P> value.

Typically, following characters are used for <P> value:
 "L": used for Link-local scoped addresses.
 "U": used for ULA
 "G": used for Global scoped address

If multiple prefixes for the same scope are used, other character (such as "H", "I",,,) can be used depending on the circumstances.

 "Prefix - <P> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "Prefix - <P> value" mapping table) is used to avoid collision and manage mappings of them.

4.1.2. <I> Value:

<I> value stands for Interface ID (lower 64 bit) of Address as 1 character format.

Following characters are used for <I> value:
<I> value assignment is based on three address type categorization.

```
"0":       used for EUI64-based address
"1",,"9":  used for manually set addresses
           (stateful addresses will be categorized here)

"a",,"z":  used for automatically generated and set addresses
           except EUI64-based
           (Temporary addresses are categorized here)
```

4.1.3. <NGI> Value:

<NGI> value is also called Auto Name-Suffix.

In order to make IPv6 addresses meaningful, IPv6 addresses are grouped. It is very natural to group IPv6 addresses by which node (interface) they are set. So, IPv6 addresses that are set to the same node (interface) are grouped into the same group.

<NGI> value is shown as 'XYZ' format:

```
'XY': (1st, 2nd chars) are inherited from
       the last octet (2 charters) of the node's MAC address
'Z' : (3rd char) suffix char to avoid a collision of 'XY'
       starting from "z"
       if 'XY' is collided, 'Z' is changed into "y", "x" ,,,
```

By using the birthday paradox theorem, collision probability of 256 states (1 octet) is calculated. If 19 nodes (interfaces) exist on the same link, collision is happened with 50% probability. Collision check procedure of the last octet of MAC addresses is necessary.

"MAC address - <NGI> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "MAC address - <NGI> value" mapping table) is needed to avoid collision and manage mappings of them.

Detailed methods how to distinguish and categorize IPv6 addresses are described at the following section.

We can found one remarkable thing with the naming rules:

"L0-XYz" is a special and very standard Auto Name. "L0-XYz" is an Auto Name that assigned for Link-local scoped EUI64-based address. Since Almost all the IPv6 nodes have Link-local scoped EUI64-based address, with "L0-XYz" name information we can reach or communicate the node whose last octet MAC address is known as "XY".

4.2. Address Type Distinction

If we can obtain address type information from Auto Name, convenient environment will be provided. So, there are strong desires to understand type of detected IPv6 addresses. In this section, methods how to distinguish and categorize IPv6 addresses are described.

4.2.1. EUI64-based Address Identification

Only with IPv6 address information, it is impossible to identify that the detected IPv6 address is EUI64-based address or not.

In proposed IPv6 address detection method which is described in the following section, IPv6 address and MAC address that are set to the same node (interface) is detected simultaneously.

So, with this detection method, it is very easy to identify that the detected IPv6 address is EUI64-based address or not.

4.2.2. "Zero Contain Rate" and Manual or Automatic Distinction

It is generally difficult to distinguish whether the detected IPv6 address is manually or automatically set address.

In order to distinguish IPv6 address types, "Zero Contain Rate" technique is introduced.

For a human being, "Zero" is a special value. When a human being omits a part of information, "Zero" is used for the omitted part of information implicitly.

For a machine "Zero" is NOT a special value. "Zero" is treated as almost equal to other values.

We can reach a fact that manually set IPv6 address contains many "Zero", because it is assigned by a human being. 64bit is long for

a human being and there must be too many omitted part filled with "Zero". Especially, a human being is apt to omit and fill with "Zero" upper part of 64bit Interface ID.

In other words, we can see human relation bias from "Zero Contain Rate" of IPv6 address.

Bias Check by using Mathematical Technique:

By using mathematical probability technique, we can distinguish whether value of 64bit Interface ID is biased (manual) or non-biased (automatic).

When we see 64bit value in 8bit (1 octet) unit, it follows the binomial distribution (n=8 and p=1/256).

Under this distribution, the probability to meet "Zero" octet two times is 0.042%. It means that to meet "Zero" octet two times is too rare case if the value is non-biased.

So, we can adopt the following method:

If the value of 64bit Interface ID contains two and above "Zero" octets,
  the value is bias and
  the IPv6 address is identified as a manually set address.

Of cause, this method is not perfect because this is based on mathematical probability technique and heuristic human behavior, but this is very effective.
Even if wrong identification is done, no big problems are found.

5. Name Services

It is not clearly defined which Name Services is utilized to achieve Auto Names specifications. In other words, any types of Name Services can be utilized. Which Name Services is utilized is tightly dependent on which Scoped Name is adopted for the Auto Names.

If wide Scoped Name is adopted, it is very natural to utilize wide Name Servce (i.e. the DNS) as the Name Services for Auto Names. If narrow scope (e.g., node-local scoped name) is adopted for Auto Names, it becomes possible to utilize node-local scoped Name Service (e.g., /etc/hosts) for Auto Name.

6. Security Considerations

Auto Names are generated and registered to the name service in this document. In order to register correct Auto Names information, communication between Detector and Registrar and communication between Registrar and Name Server should be protected and be secured.

In general usage, scope of Auto Names will be local (not global). Auto Names are usually local scoped names. So, we do not have to be too sensitive on the correctness of Auto Names.

7. IANA Considerations

This document does not require any resource assignments to IANA.

Appendix A. IPv6 Address Appearance Detection and Auto Name Registration

In order to generate and register Auto Names automatically, two types of mechanisms are needed. One is a mechanism that detects IPv6 address appearance. The other is a mechanism that checks the detected addresses and generates Auto Names and registers them to name service.

Two functions ("Detector" and "Registrar") are introduced. Detector" function takes in charge of the former mechanism, and "Registrar" takes in charge the latter mechanism.

A.1. IPv6 Address Appearance Detection mechanism

In order to detect newly appeared IPv6 address, DAD message (NS for DAD) is effectively used.

DAD message has the following good capabilities:

- issued only when node would like to set new IPv6 address

- issued for All types (link-local, global, temporary,,,)

- L2 broadcast and easy to capture (without using mirror port)

- distinguishable from other NS messages, because source address of the message is unspecified ("::") and different from others

- Captured DAD message includes all necessary information (such as, IPv6 address and MAC address)

Detector captures DAD messages and detects newly appeared IPv6 addresses. Detected information is sent to Registrar.

A.2. Auto Names Generation and Registration mechanism

At first, Registrar checks the Detected address information that is sent from Detector(s). By using the reverse resolving (Address -> Name), it is checked whether the Detected address information is first appearance or not. If an entry for the address does NOT exist, it is confirmed that the address is first appearance and it should be registered to the name server.

After Name for the address is prepared, duplication of the Name can be checked by using the regular resolving (Name -> Address). If an entry for the Name exist, it is confirmed that Name is duplicated (collided). Another Name is prepared and checked again until the Name

is not duplicated.

Finally, Registrar registers both Regular and Reverse resolving entries for the address and prepared Auto Name are registered to the name server.

A.3. Placement of Detector and Registrar

Placement of Detector and Registrar is designed to make the mechanisms flexible and to make it to be applied to various environments (office networks, home networks, etc.)

Figure 1 and 2 show typical examples that indicate locations where Detector and Registrar functions are placed on the IPv6 network. Figure 1 shows a case for a single link, and Figure 2 shows a case for multiple links.

```
               |                                 +------------+
               |                                 | Name Server|
             +-+-+  %%%%%%%%%%%%  #############  +------+-----+
             | R |  % Detector %  # Registrar #        /
             +-+-+  %%%%%%%%%%%%  #############       +---+
               |         |              |                /
           ----+---------+-------+------+---------------+-----
                                 |
                           +===========+
                           | IPv6 Node |
                           +===========+
```

Fig. 1 Single-Link Case Example

```
                                         +------------+
                                         | Name Server|
                     #############       +------+-----+
                     # Registrar #            /
                     #############           +---+
                           |                   /
           ----+-----------+------------+-------+------
               |                        |
             +-+-+   %%%%%%%%%%%%%    +-+-+   %%%%%%%%%%%%%
             | R1|   % Detector1 %    | R2|   % Detector2 %
             +-+-+   %%%%%%%%%%%%%    +-+-+   %%%%%%%%%%%%%
               |           |            |           |
           ----+-----+-----+-----   ----+-----+-----+-----
                     |                        |
               +===========+            +===========+
               | IPv6 Node |            | IPv6 Node |
               +===========+            +===========+
```

Fig. 2 Multiple-Link Case Example

A.4. Detection and Registration Procedures

Figure 3 shows an example of typical detection and registration procedures at IPv6 links where DAD packets are issued. DAD message packets are used for the appearance detection.

```
   IPv6 Node    Router     Detector     Registrar    Name Server

     | link local |          |          |            |
  (a)|---DAD NS--->--------->|          |            |
  (b)|    no NA   |          |(detect)  |            |
  (c)|            |          |=========>|            |
  (d)|            |          |          |----------->|(Reverse Check)
  (e)|            |          |          |<-----------|
     |            |          |          |            |
  (f)|            |          |          |----------->|(Regular Check)
  (g)|            |          |          |<-----------|
     |            |          |          |            |
  (h)|            |          |          |===========>|(Reg. Register)
  (i)|            |          |          |<-----------|
  (j)|            |          |          |===========>|(Rev. Register)
  (k)|            |          |          |<-----------|
     |   global   |          |          |            |
  (l)|(----RS--->)|          |          |            |
  (m)|<----RA-----|          |          |            |
  (n)|---DAD NS--->--------->|          |            |
  (o)|    no NA   |          |(detect)  |            |
  (p)|            |          |=========>|            |
     |            |          |          |            |
  (q)|            |          |          |----------->|(Reverse Check)
  (r)|            |          |          |<-----------|
     |            |          |          |            |
  (s)|            |          |          |----------->|(Regular Check)
  (t)|            |          |          |<-----------|
     |            |          |          |            |
  (u)|            |          |          |===========>|(Reg. Register)
  (v)|            |          |          |<-----------|
  (w)|            |          |          |===========>|(Rev. Register)
  (x)|            |          |          |<-----------|
     |            |          |          |            |
```

Appendix B. Implementation

Auto Name functions have been implemented at the following environments. It has been verified that designed functions work well.

Used functions:
  Packet capture on Detector: libpcap
  Name Server: DNS (BIND9)
  Name Registration: nsupdate (BIND9 bundled)

OS:
  FreeBSD 6.2R
  (Since FreeBSD OS specific funtions are not used to implement, codes will run on UNIX type OS that has libpcap (such as Linux).)

Acknowledgment

A part of this work is supported by the program: SCOPE (Strategic Information and Communications R&D Promotion Programme) operated by Ministry of Internal Affairs and Communications of JAPAN.

References

Normative References

[RFC4291] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006

[RFC4861] T. Narten, E. Nordmark, W. Simpson and H. Soliman, "Neighbor Discovery for IP Version 6 (IPv6)," RFC 4861, September 2007

[RFC4862] S. Thomson, T. Narten and T. Jinmei "IPv6 Stateless Address Autoconfiguration," RFC4862, September 2007

[RFC4941] T. Narten, R. Draves and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," RFC4941, September 2007

[RFC1034] P. Mockapetris, "Domain names - concepts and facilities ", RFC 1034, November 1987

[RFC1035] P. Mockapetris, "Domain names - implementation and specification", RFC 1035, November 1987

[RFC2136] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System," RFC 2136, April 1997

[RFC4795] B. Aboba, D. Thaler, and L. Esibov, "Link-Local Multicast Name Resolution (LLMNR)," RFC4795, January 2007

Informative References

[RFC4620] M. Crawford and B. Haberman, "IPv6 Node Information Queries," RFC4620, August 2006

[mDNS] S. Cheshire and M. Krochmal, "Multicast DNS" <draft-cheshire-dnsext-multicastdns-14.txt> work in progress, February 2011

[RFC3849] G. Huston, A. Lord and P. Smith, "IPv6 Address Prefix Reserved for Documentation," RFC3849, July 2004

Authors' Addresses

Hiroshi Kitamura
Knowledge Discovery Research Laboratories, NEC Corporation
(SC building 12F)1753, Shimonumabe, Nakahara-Ku, Kawasaki,
Kanagawa 211-8666, JAPAN
Graduate School of Information Systems,
University of Electro-Communications
5-1 Chofugaoka 1-Chome, Chofu-shi, Tokyo 182-8585, JAPAN
Phone: +81 44 431 7686
Fax: +81 44 431 7680
Email: kitamura@da.jp.nec.com

Shingo Ata
Graduate School of Engineering, Osaka City University
3-3-138, Sugimoto, Sumiyoshi-Ku, Osaka 558-8585, JAPAN
Phone: +81 6 6605 2191
Fax: +81 6 6605 2191
Email: ata@info.eng.osaka-cu.ac.jp