Corresponding Auto Names for IPv6 Addresses
<draft-kitamura-ipv6-auto-name-02.txt>

Hiroshi KITAMURA
NEC Corporation
kitamura@da.jp.nec.com
Introduction

IPv6 address is too long and complicated to remember for human. It is very nuisance or almost impossible to ‘type’ a literal IPv6 address manually.

Also, literal IPv6 address information can be called meaningless. Because it is very difficult for human to tell which IPv6 address is set to which actual IPv6 node at a glance.

Strong desires:
• Use human-friendly “Name” instead of literal IPv6 Address.
• Annotate literal IPv6 address and Change information from almost meaningless to meaningful.

An idea “Corresponding Auto Names” is introduced to solve above problems and to satisfy the above desires.
Assumed Typical IPv6 Communication Environment

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

Node B (MAC: 00:0c:76:d9:14:e3)
Literal Address
- fe80::20c:76ff:fed9:14e3%em0
- fd01:2345:6789::20c:76ff:fed9:14e3
- fd01:2345:6789::5678
- 2001:db8::20c:76ff:fed9:14e3
- 2001:db8::5678

Router

ULA RA fd01:2345:6789::/64
Global RA 2001:db8::/64
Auto Names Examples

Node A (MAC: 00:0d:5e:b8:80:7b)

- Literal Address
  - fe80::20d:5eff:feb8:807b%fxp0
  - fd01:2345:6789::20d:5eff:feb8:807b
  - fd01:2345:6789::1234
  - 2001:db8::20d:5eff:feb8:807b
  - 2001:db8::1234

- Auto Name
  - L0-7bz%fxp0
  - U0-7bz
  - U1-7bz
  - G0-7bz
  - G1-7bz

Node B (MAC: 00:0c:76:d9:14:e3)

- Literal Address
  - fe80::20c:76ff:fed9:14e3%em0
  - fd01:2345:6789::20c:76ff:fed9:14e3
  - fd01:2345:6789::5678
  - 2001:db8::20c:76ff:fed9:14e3
  - 2001:db8::5678

- Auto Name
  - L0-e3z%em0
  - U0-e3z
  - U1-e3z
  - G0-e3z
  - G1-e3z
Auto Name Suffix for Grouped Addresses

In order to make Auto Names meaningful,
– IPv6 addresses are grouped.
– 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 is used for their Auto Names.

As shown above example:
• ‘-7bz’ is used for Auto Name Suffix for Node A (00:0d:5e:b8:80:7b)
• ‘-e3z’ is used for Auto Name Suffix for Node B (00:0c:76:d9:14:e3)

Naming rule of Auto Name Suffixes is based on
inheriting the last octet of the node's MAC address.
Contribution in **Regular** Resolving (Name -> Address) (1/3) at command lines

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

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

Almost **Impossible** to ‘type’ commands for human

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

Become **Possible** to ‘type’ commands for human
Contributions in **Regular Resolving** (Name -> Address) (2/3) in URLs

When we access URLs that include a literal IPv6 address by ‘**web browser**’, the following strings are must be typed.

```
http://[fe80::20c:76ff:fed9:14e3%fxp0]/....
http://[fd01:2345:6789::20c:76ff:fed9:14e3]/....
```

---

**AlmostImpossible to ‘type’ such URLs for human**

It is nuisance to use ‘[’ and ’]’ in URLs.

```
http://L0-3ez%fxp0/....
http://U0-3ez/....
```

---

**Become Possible to ‘type’ such URLs for human**

We are released from using ‘[’ and ’]’ in URLs !!
Contribution in Regular Resolving (Name -> Address) (3/3) at filter configurations

Configure access filter (e.g., /etc/hosts.allow) as follows:

sshd : [fe80::20c:76ff:fed9:14e3%fxp0]      : allow

Almost impossible to ‘type’ entries for human. ‘copy and paste’ is required to make entries. Impossible to understand meanings at a glance

sshd : L0-3ez%fxp0  : allow
sshd : U0-3ez        : allow

Become Possible to ‘type’ entries for human
Easy to understand the meanings at a glance
Contribution in Reverse Resolving (Address -> Name) (1/2) session status

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

<table>
<thead>
<tr>
<th>Local Address</th>
<th>Foreign Address</th>
<th>(state)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fe80::20d:5eff:feb8:807b.8722</td>
<td>fe80::20c:76ff:fed9:14e3.23</td>
<td>ESTABLISH</td>
</tr>
<tr>
<td>fd01:2345:6789::1234.16258</td>
<td>fd01:2345:6789::5678.23</td>
<td>TIME_WAIT</td>
</tr>
</tbody>
</table>

Almost Meaningless information for human

<table>
<thead>
<tr>
<th>Local Address</th>
<th>Foreign Address</th>
<th>(state)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0-7bz.8722</td>
<td>L0-e3z.23</td>
<td>ESTABLISH</td>
</tr>
<tr>
<td>U1-7bz.16258</td>
<td>U1-e3z.23</td>
<td>TIME_WAIT</td>
</tr>
</tbody>
</table>

Become Meaningful information for human

Also, Beautified display by fixed length character of Auto Name
Contribution in **Reverse Resolving** (Address -> Name) (2/2) packet dump etc.

**Other examples** where the Auto Names can contributes:

- In **access log files** of a server application:
  
  Accessed clients are can be recoded as meaningful Auto Names instead of (almost meaningless) literal IPv6 address.

- In **packet dumping** applications:
  
  Address information can be shown as meaningful Auto Names

The Auto Name technique can significantly help for human to analyze and understand above information.

Auto Name format is simple and easy enough for human to understand.

By using the Auto Names technique,
  
  literal IPv6 addresses are annotated and these information are converted from almost meaningful to meaningful.
Deployed Notions and Functions used in Auto Names

• **Stateless Name**

<table>
<thead>
<tr>
<th></th>
<th>Stateful</th>
<th>Stateless</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Address</strong></td>
<td>DHCPv6</td>
<td>SLAAC</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Existing Domain Names</td>
<td>Auto Names</td>
</tr>
</tbody>
</table>

• **Scoped Name**

<table>
<thead>
<tr>
<th></th>
<th>Global</th>
<th>Site-Local (ULA)</th>
<th>Link-Local</th>
<th>Node-Local</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Address</strong></td>
<td>e.g., 2001:db8::/64</td>
<td>e.g., fd01:2345:6789::/64</td>
<td>fe80::/64</td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Existing Domain Names</td>
<td>Existing Domain Names / Auto Names</td>
<td>Auto Names</td>
<td>Auto Names</td>
</tr>
</tbody>
</table>

Scope is dependent on how Auto Names data is dealt and which “name services” are used.
Design of Auto Names  
(Conceptual Design on Naming Rules)

**Auto Name** is *fixed 6 characters strings* and composed of "<P><I>-<NGI>" format.

- **<P>:** stands for Prefix part of IPv6 Address  
  1 character: (e.g., 'L', 'U', 'G')
- **<I>:** stands for Interface ID part of IPv6 Address  
  1 character: (e.g., '0', '1', '2', , , '9', 'a', , , 'z')
- **<NGI>:** stands for Node (Interface) Group ID  
  3 characters: (e.g., '7bz', '3ez') *inherited* from the last octet (2 characters) of the node's MAC address

Above Auto Name examples satisfy <P><I>-<NGI> format.
- Node A: L0-7bz, U0-7bz, U1-7bz-u2, G0-7bz, G1-7bz
- Node B: L0-3ez, U0-3ez, U1-3ez-u2, G0-3ez, G1-3ez
Site-dependent Mapping tables (for collision avoidance)

Mapping tables are used only when Auto Names are generated
(These tables are not used for usual name resolving operations)

- MAC address – <NGI> value mapping table

<table>
<thead>
<tr>
<th>MAC Address</th>
<th>&lt;NGI&gt; value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:0d:5e:b8:80:7b</td>
<td>–7bz</td>
</tr>
<tr>
<td>00:0c:76:d9:14:e3</td>
<td>–e3z</td>
</tr>
</tbody>
</table>

- Prefix – <P> value mapping table

<table>
<thead>
<tr>
<th>Prefix</th>
<th>&lt;P&gt; value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fe80::/64</td>
<td>Link-Local</td>
</tr>
<tr>
<td>fd01:2345:6789::/64</td>
<td>Site-Local (ULA)</td>
</tr>
<tr>
<td>2001:db8::/64</td>
<td>Global</td>
</tr>
</tbody>
</table>
**Auto Names techniques in short**

- Under certain **scoped name** environment, All IPv6 addresses (formed as Prefix + I/F ID) are shown in only **fixed 6 characters** ("<P><I>-<NGI>“) strings format.  
  [kind of address **compression techniques** are used.]

- IPv6 Address information is annotated and changed almost meaningless ➡️ meaningful

- Human can remember, understand and ‘type’ **Auto Names** (instead of literal IPv6 addresses).
Discussions

Please let us know you comments.

Goal of this I-D is to be published as “Informational RFC”. 
Reserved slides are started from here.
<P> Value

<P> value stands for Prefix (Scope) part of IPv6 Address as 1 character format.

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

Typically, following characters are used:

"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 circumstances.
<I> Value

<i> value stands for Interface ID part of IPv6 Address as 1 character format.
<i> value assignment is based on three address type categorization

<table>
<thead>
<tr>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;0&quot;</td>
<td>used for <strong>EUI64-based</strong> address</td>
</tr>
<tr>
<td>&quot;1&quot; - &quot;9&quot;</td>
<td>used for <strong>manually</strong> set addresses</td>
</tr>
<tr>
<td></td>
<td>(stateful addresses will be categorized here)</td>
</tr>
<tr>
<td>&quot;a&quot; - &quot;z&quot;</td>
<td>used for <strong>automatically</strong> generated and set addresses</td>
</tr>
<tr>
<td></td>
<td>except EUI64-based</td>
</tr>
<tr>
<td></td>
<td>(Temporary addresses are categorized here)</td>
</tr>
</tbody>
</table>
Address Type Distinction

• EUI64-based Address Identification
  – When *IPv6 and MAC* addresses are found simultaneously, it is easy to identify.

• Manual or Automatic Distinction
  – *Human bias* is checked by using "*Zero Contain Rate*" in IPv6 Address.
<NGI> Value

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

<NGI> value is shown as 'XYZ' format:
'XY': (1st, 2nd chars) are **inherited** from
the **last octet** (2 characters) 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" ,,,

**Collision Probability** of 256 states (1 octet):
By using the *birthday paradox theorem*, probability is calculated.
If there are **19 nodes** (interfaces) on the same scope,
collision is happened **with 50%** probability.
Collision check procedure for 'XY' is necessary.
Question: Who will became happy with this Auto Name technique?

Answer: People who face literal IPv6 address problems.

If you feel frustration to handle literal IPv6 addresses, you can become happy with this Auto Name technique.
Index

• Introduction

• Goals: What can be achieved
  – Assumed typical IPv6 communication environment
  – Auto Names examples
  – Auto Name Suffix for Grouped Addresses
  – Contributions in Regular Resolving (Name -> Address)
  – Contributions in Reverse Resolving (Address -> Name)

• Deployed Notions and Functions
  – Stateless Name
  – Scoped Name

• Design of Auto Names
  – Conceptual Design on Naming Rules

• Discussions
Contribution in Reverse Resolving
(Address -> Name)  (2/3)  neighbor cache

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

<table>
<thead>
<tr>
<th>Neighbor</th>
<th>Linklayer Addr.</th>
<th>Netif Expire</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>fe80::20d:5eff:feb8:807b%fxp0</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>fd01:2345:6789::20d:5eff:feb8:807b</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>fd01:2345:6789::1234</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>2001:db8::20d:5eff:feb8:807b</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>2001:db8::1234</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>fe80::221:85ff:fea7:82ff%fxp0</td>
<td>0:21:85:a7:82:ff</td>
<td>fxp0 23h50m51s</td>
<td>S</td>
</tr>
<tr>
<td>fe80::20c:76ff:fed9:14e3%fxp0</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h51m56s</td>
<td>S</td>
</tr>
<tr>
<td>fd01:2345:6789::20c:76ff:fed9:14e3</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h52m50s</td>
<td>S</td>
</tr>
<tr>
<td>fd01:2345:6789::5678</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h53m51s</td>
<td>S</td>
</tr>
<tr>
<td>2001:db8::20c:76ff:fed9:14e3</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h54m53s</td>
<td>S</td>
</tr>
<tr>
<td>2001:db8::5678</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h55m54s</td>
<td>S</td>
</tr>
<tr>
<td>L0-7bz%fxp0</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>U0-7bz</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>U1-7bz</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>G0-7bz</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>G1-7bz</td>
<td>0:0d:5e:b8:80:7b</td>
<td>fxp0 permanent</td>
<td>R</td>
</tr>
<tr>
<td>L0-ffz%fxp0</td>
<td>0:21:85:a7:82:ff</td>
<td>fxp0 23h50m51s</td>
<td>S</td>
</tr>
<tr>
<td>L0-3ez%fxp0</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h51m56s</td>
<td>S</td>
</tr>
<tr>
<td>U0-3ez</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h52m50s</td>
<td>S</td>
</tr>
<tr>
<td>U1-3ez</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h53m51s</td>
<td>S</td>
</tr>
<tr>
<td>G0-3ez</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h54m53s</td>
<td>S</td>
</tr>
<tr>
<td>G1-3ez</td>
<td>0:0c:76:d9:14:e3</td>
<td>fxp0 23h55m54s</td>
<td>S</td>
</tr>
</tbody>
</table>
Name Services

• It is not clarified: which actual ‘name services’ is used for Auto Names.
  – DNS is first strong candidate for it.
    • All OS have DNS resolver implementations.
    • By using the DNS user authenticate implementation, it is easy to achieve the ‘Scoped Name’ features.
Target **IPv6 Addresses** of Auto Names

- Target of Auto Names:
  - **All** unicast IPv6 addresses
    (include link-local scoped addresses) are target

- Exception (non-target):
  - “**Well-managed**” IPv6 addresses are basically **non-target**

Definition of “**Well-managed**” addresses:

Their “Domain Names” are manually (or statefully) registered into name services (such as the DNS) already.

**Reverse** mapping Auto Name entries are needed for **All** addresses.

**Regular** mapping Auto Name entries will **not be mandated** for “**Well-managed**” addresses.

(It is OK to register **Regular** mapping entries, because one-multiple entries are possible and they will not cause problems.)
Regular (Name -> Address) and Reverse (Name <- Address) mapping

Well-managed manual mapping
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.
Single-Link Case

Name Server or Some DB

Detector
- detected address with Detector ID

Registrar
- domain name

R
- RA prefix

DAD finished

address
- Plugged-in IPv6 Node

dynamic update duplication check finished
Single-Link Case

(1) Plug-in IPv6 Node

(2) Detect

(3) Request

(4) Check & Name

(5) Register

Name Server or Some DB
Multiple-Link Case

DAD finished

Plugged-in IPv6 Node

R1
RA prefix

Detector1
detected address with Detector ID

R2

Detector2

Name Server or Some DB

dynamic update duplication check finished

Registrar
domain name

DAD finished

Plugged-in IPv6 Node

address

Plugged-in IPv6 Node

address

Plugged-in IPv6 Node

address

Plugged-in IPv6 Node

address

Plugged-in IPv6 Node

address

Plugged-in IPv6 Node

address

Plugged-in IPv6 Node
Multiple-Link Case

(1) Plug-in
(2) Detect
(3) Request
(4) Check & Name
(5) Register

Plugged-in IPv6 Node
Registrar

Name Server or Some DB

R1
Detector1

Plugged-in IPv6 Node

R2
Detector2
# Roles/Characteristics Comparison

<table>
<thead>
<tr>
<th></th>
<th>Detector</th>
<th>Registrar</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Roles</strong></td>
<td>Detect appearance</td>
<td>Check received data</td>
</tr>
<tr>
<td></td>
<td>Send detected data</td>
<td>Prepare “Auto Name”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Register to name service</td>
</tr>
<tr>
<td><strong>Intelligence</strong></td>
<td>NOT required</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td>Simple</td>
<td>Complex</td>
</tr>
<tr>
<td><strong>Maintenance</strong></td>
<td>Almost Free</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Located place</strong></td>
<td>Limited</td>
<td>NOT limited</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Easy</td>
<td>Not easy</td>
</tr>
</tbody>
</table>
Typical Procedures

IPv6 Node → Router → Detector → Registrar → Name Server (or Some DB)

**DAD link-local**
1. **Plug-in**
2. **Detect**
3. **Request** detected address
4. **Check & Name**

**DAD global**
1. **Detect**
2. **Detect**
3. **Request** detected address
4. **Check & Name**

Check by Reverse and Regular Resolving
Register both Regular and Reverse Entries

Links:
- (a) NS
- (b) [no NA]
- (c) RA
- (d) [no NA]
- (e) RA
- (f) [no NA]
- (g) RA
- (h) [no NA]
- (i) RA
- (j) [no NA]
- (k) RA
- (l) RA
- (m) [no NA]
- (n) RA
- (o) [no NA]
- (p) RA
- (q) [no NA]
- (r) RA
- (s) [no NA]
- (t) RA
- (u) [no NA]
- (v) RA
- (w) [no NA]
- (x) RA