IPv6 Neighbor Cache Update
<draft-kitamura-ipv6-neighbor-cache-update-00.txt>
Index

• Introduction / Background

• Problems
  – on (Not-Used) Long Remained NC entries.

• Proposed Solutions  (Neighbor Cache Update (Delete))
  – Heuristic Type:  (w/o any ND message extensions)
  – Explicit Type:   (w/ small extension (NA flags))
  – Explicit + Heuristic Combined Type

• Implementation
Introduction / Background

IP address’s “Using Status” is frequently changed
“Used” <=> “Not Used”

• Disconnecting / Connecting nodes
  from/to networks at mobile environments

• Suspending / Hibernating / Resuming nodes
  – Turn Off / On PCs
  – Release / Discover IP address by DHCP

• Utilize Changeable-type Addresses:
  Temporary Address / Ephemeral Address*

  * <draft-kitamura-ipv6-ephemeral-address-01>
Problems on (Not-Used) Remained Neighbor Cache Entries

• What’s happens when (IP address is gone) IP address’s Using Status is changed form “Used” to “Not Used”? 

• Related Neighbor Cache Entries (that are created for the “Gone IP addresses”) are not deleted and still remained for a long time (typically 24 hours).
Example:
(Not-Used) Long Remained NC entries 1/2
Example:
(Not-Used) Long Remained NC entries 2/2

<table>
<thead>
<tr>
<th>IP</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP_1</td>
<td>L2_A</td>
</tr>
<tr>
<td>IP_2</td>
<td>L2_B</td>
</tr>
<tr>
<td>IP_3</td>
<td>L2_C</td>
</tr>
<tr>
<td>IP_4</td>
<td>L2_D</td>
</tr>
<tr>
<td>IP_5</td>
<td>L2_E</td>
</tr>
</tbody>
</table>

but

(Not-Used) NC entries Still Remained for a long time
Why Not-Used NC entries are remained?

• NC state procedures are showed in right figure that is defined in ND specification [RFC4861].

• Not-Used NC entries are remained at STALE state for a long time and finally they are deleted by the “garbage collections”. 
Characteristics on
(Not-Used) Long Remained NC entries

It is clear:
– from efficient resource management viewpoint: NOT Good.
– from security enhancement viewpoint: NOT Good.
What should we do?

- We have to follow the manner:
  “Leave everything neat and tidy when you go behind you”

- When using status of an IP address is changed from “Used” to “Not-Used”, its related cache entry should be deleted cooperatively.

- We have to provide quick and clear neighbor cache update (delete) functions.
Proposed Solutions: Neighbor Cache Update (Delete) Methods

Three types of Neighbor Cache Update (delete) methods are proposed.

1. **Heuristic** Type:
   Does NOT require any ND message extensions

2. **Explicit** Type:
   Requires small extensions (NA message Flags)

3. **Explicit + Heuristic** Combined Type:
   Any types of nodes are supported effectively
Heuristic Type
Neighbor Cache Update

- **Stimulate** the remaining **STALE** (inactivated) NC entry by sending the special NS message (source = Gone IP address) from client node.

- (The target NC entry is **activated** by issuing NA.) Its state is proceeded to next state **DELAY** and finally the target **NC entry is deleted**.

- Takes short time periods for **DELAY** and **PROBE** states.

- No ND message extensions are required.
Explicit Type: Neighbor Cache Update

- Issue an **Extended NA message** (+extended flags) to **delete** target NC entry from client node.

- If a receiver node understands **the extended flags**, the target NC entry is quickly deleted.

- If the node does **not** understand, the message is simply **ignored**. (the NC entry is not deleted and errors are not reported.)
Explicit Type:
NA Message Flags Extensions

D: Delete flag (Delete entry except its state is REACHABLE)
F: Force Delete flag (Force to delete entry at any states)
Explicit + Heuristic Combined Type Neighbor Cache Update

• Support both types of nodes that do and do not understand the NA extensions effectively.
  – Nodes do understand extensions: the entry is deleted quickly by the 1st Explicit operation.
  – Nodes do not understand extensions: the entry is deleted shortly by the 2nd Heuristic operation.

• In any node cases, the target NC entry is surely deleted.
Implementations

• Proposed all “Neighbor Cache Update” specification has been implemented and verified.

• Delete Responder (Edge Router) type:
  – Explicit Type:
    • FreeBSD
  – Heuristic Type:
    • IOS, Linux, FreeBSD, MacOS X, Windows, etc.

• Delete Initiator (Client) type:
  – Explicit / Heuristic Type: (Verified)
    • FreeBSD
  – Explicit / Heuristic Type: (Under Developing)
    • Linux, MacOS X, Windows, etc.
Consensus Verification to Proposed Methods

Which methods do you prefer?

1. **Heuristic** Type:
   Does **NOT** require any ND message extensions

2. **Explicit** Type:
   Requires small extensions (NA message Flags)

3. **Explicit + Heuristic** Combined Type:
   Any types of nodes are supported effectively
   [Authors recommend this type method]
Related Issues

• Same types of problems can be found in IPv4 ARP table entries.

• How do we have to deal with it?