Solution approaches for address-selection problems

draft-arifumi-6man-addr-select-sol-00.txt

NTT PF Lab.  Arifumi Matsumoto
Tomohiro Fujisaki
Intec NetCore, Inc.  Ruri Hiromi
Kenichi Kanayama
About our series of drafts

- At v6ops
  - PS (Problem statement draft) is at AD review
    - lists up address selection related problems.
  - REQ (Requirements draft) is at AD review
    - lists up requirements for solutions.
  - SOL (Solution analysis draft) was at v6ops
    - outlines and evaluates 4 kinds of possible approaches
- SOL moves from v6ops to 6man
  - Mainly because this entails protocol work.
  - And 6man is there now.
Motivation for address selection

- Detailed in PS, but very shortly …
- Detailed control over unmanaged hosts’ address selection behavior:
  - Put less/higher priority on 6to4, Teredo and ULA,…
    - 6to4 comes before IPv4 by default.
  - Smooth IPv4 to IPv6 transition
    - v4-only -> v4 then v6 -> v6 then v4 -> v6-only
  - Smooth address renumbering
    - More quick and definitive renum. process
Motivation for address selection

Cont.

- To replace a NAT box:
  - NAT lies everywhere in IPv4 network
  - How do we deploy IPv6 in these sites?

Beautiful!

But, we cannot always merge NW1 and 2

We decided not to NAT, so we need an alternative way
Possible Approaches for Address Selection Problems

• Proactive Approach
  – Deliver Everything At Once Approach
    • E.g. A host acquires RFC 3484 Policy Table
    • E.g. K. Fujikawa’s address selection proposal
  – A Question and An Answer Approach
    • A host asks an Agent Server(router) about addresses.

• Reactive Approach
  – Try-and-Error Approach
    • Host stores addr-select cache based on ICMP error
  – All by Oneself Approach
    • Shim6: A host performs failure detection, address cycling
The Most Proactive Approach
“Deliver Everything At Once Approach”

- E.g. “RFC 3484 Policy Table Delivery by DHCPv6”
  - draft-fujisaki-dhc-addr-select-opt-04.txt
- Requirement correspondence analysis
  - Dynamicness depends on the transport mechanism.
  - Policy collision can happen when belongs to multiple admin domain simultaneously.
- Other Issue
  - OS with Policy Table needs no
Proactive Approach
“A Question and An Answer Approach”

- E.g. “Routing system assistance for address selection”
- Requirement correspondence analysis
  - Dynamically changing network status is easily reflected.
  - Policy can collide in multiple admin domain and with multiple servers.

Other Issues
- Host implementation needs a big change.
- Application also has to be modified.
Reactive Approach
“Try-and-Error Approach”

- E.g. RFC3484-update by M. Bagnulo
  - An ICMP Error notifies address mal-selection.
  - Hosts store cache of address-pair reachability

- Requirement correspondence analysis
  - Dynamically changing network status is easily reflected.
  - The usability can degrade badly dependent on application behavior.

- Other Issues
  - Per destination host cache can be so big.
The Most Reactive Approach
“All by Oneself Approach”

- E.g. Shim6
  - A host can perform failure detection and address cycling without a help from outside.
- Requirement correspondence analysis
  - A User may have to wait before finding working address pair.
  - Central control can only be implemented by packet filtering
- Other Issues
  - No router modification needed.
  - The host implementation has to be changed
Applicability Domain

- **Policy Dist.**
- **Routing System Assist.**
- **3484-update**
- **Shim6**

The right method in the right place.
# Requirement correspondence analysis summary

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Policy Dist</th>
<th>Router Assist</th>
<th>3484-update</th>
<th>Shim6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Timing</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Dynamic Update</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Node-Specific</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Appl-Specific</td>
<td>Fair</td>
<td>Fair</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Multi-Interface</td>
<td>Fair</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Central Control</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Route Selection</td>
<td>Fair</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Other Issue</td>
<td>Freq. updates cause traffic</td>
<td>Big Impact on a host’s stack</td>
<td>Big Impact on a host’s stack</td>
<td>Big impact on a host’s stack</td>
</tr>
</tbody>
</table>
Discussion@Chicago and ML

- About multi-prefix way,
  - It isn’t simple and should be avoided.
  - It’s necessary in today’s complex network.
    - >> The discussion ends up undecided.

- About requirement,
  - “compatibility with RFC3493” is important
    - >> Then, was included in the req. list in -04.

- About “policy table distribution method”,
  - Manybody likes it.
    - “looks like the only implementable approach”
  - Zone-index should not be distributed
    - >> Then, zone-index was made optional in -04.
Next step

- Is this work useful?
  - as 6man wg item.

- Have we decided one direction?
  - Policy Table Distribution
  - Q and A approach
  - Try and Error approach
  - All by oneself approach