Solution approaches for address-selection problems

draft-arifumi-v6ops-addr-select-sol-00.txt

(NTT PF Lab.) Arifumi Matsumoto
Tomohiro Fujisaki
(Intec NetCore, Inc.) Kenichi Kanayama
Ruri Hiromi
Possible Approaches for Address Selection Problems

- **Proactive Approach**
  - Deliver Everything At Once Approach
    - E.g. A host acquires RFC 3484 Policy Table
  - A Question and An Answer Approach
    - A host asks an Agent Server (e.g. a router) “which of my addresses is the best for a destination?”

- **Reactive Approach**
  - Try-and-Error Approach
    - An ICMP Error notifies the host of address false-selection and the host stores cache in case for the next try.
  - All by Oneself Approach
    - Shim6: A host performs failure detection and 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-03.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 change.
  - Frequent updates can cause a lot of traffic
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
## Requirement correspondence analysis summary

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Policy Dist</th>
<th>Router Assist</th>
<th>3484update</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>
So, the right method in the right place.
Next step

• Useful work?
• Become v6ops work item?
• Any questions and comments?