

Domain Name Auto-Registration for Plugged-in IPv6 Nodes

<draft-kitamura-ipv6-name-auto-reg-00.txt>

Hiroshi KITAMURA

NEC Corporation

kitamura@da.jp.nec.com

Background

- IPv6 addresses are **too long** to remember
- EUI64-based addresses are **too complicated** to remember

Strong requirements:

- to **register** *domain names* for plugged-in IPv6 nodes **automatically**
- to **use** *logical names* **that are easy to remember** instead of *IPv6 addresses* to specify IPv6 nodes

Objective

- Describes a **scheme** of “**Domain Name Auto-Registration** for Plugged-in IPv6 Nodes” mechanism
- Make it possible to *register* both regular and inverse domain name of plugged-in IPv6 nodes *automatically*.
- Propose a mechanism as **one of the IPv6 auto-configuration** (plug and play) functions.
- After the “*Address Autoconfiguration*” [RFC2462], it works as **a succeeding plug and play mechanism**.
- “*Dynamic Updates in the DNS*” [RFC2136] can help automatic domain name information registration.
- **Clarify and solve problems** in applying “*Dynamic Updates*” to “*Domain Name Auto-registration*”

Target Situation

- *Plug and play situations*
(NOT manually configured situations)
- Plug and play situations make **limitations**
and have **special characteristics**
- Mechanism must be **optimized to run efficiently**
for plugged-in IPv6 nodes
under plug and play situations.
- Administrative cost must be reduced
- Automated mechanism is necessary

Consideration for Plugged-in IPv6 Nodes

1. Plugged-in IPv6 nodes do *NOT have sufficient capability* to show their preferences.
It is difficult for plugged-in IPv6 nodes to show their preferences for their domain names.
2. It is *desirable* to achieve the mechanism *without installing new functions* into plugged-in IPv6 nodes.
3. **Having** (registering) **“SOME” domain name** *that is easy to remember* is essential.
Which actual name is assigned is NOT main concern for a plugged-in node.

“Default Domain Name”

An idea of “*default domain name*” is introduced

1. A new plugged-in IPv6 node appears
2. An appearance is automatically **detected**
3. A “*default domain name*” is **selected** for it
4. Both regular and inverse information of the “*default domain name*” are **registered** to DNS servers automatically

If a node is not satisfied with “default domain name”, it can rename it manually.
(under non-plug and play situations)

Problems in applying the Dynamic Updates mechanism 1/2

There are Problems in applying the Dynamic Updates [RFC2136] to automatic domain name registration.

For plugged-in nodes:

1. It is *difficult to become trusted nodes*.
 - Dynamic Updates messages from plugged-in nodes are usually not accepted
2. It is *difficult to know the locations* of the DNS servers to register their domain name information.
3. It is *difficult to prepare sufficient domain name information* to register.
 - Need to know their **DNS zone suffix information** to prepare FQDN information for registration
 - It is not easy for plugged-in nodes to acquire it.

Problems in applying the Dynamic Updates mechanism 2/2

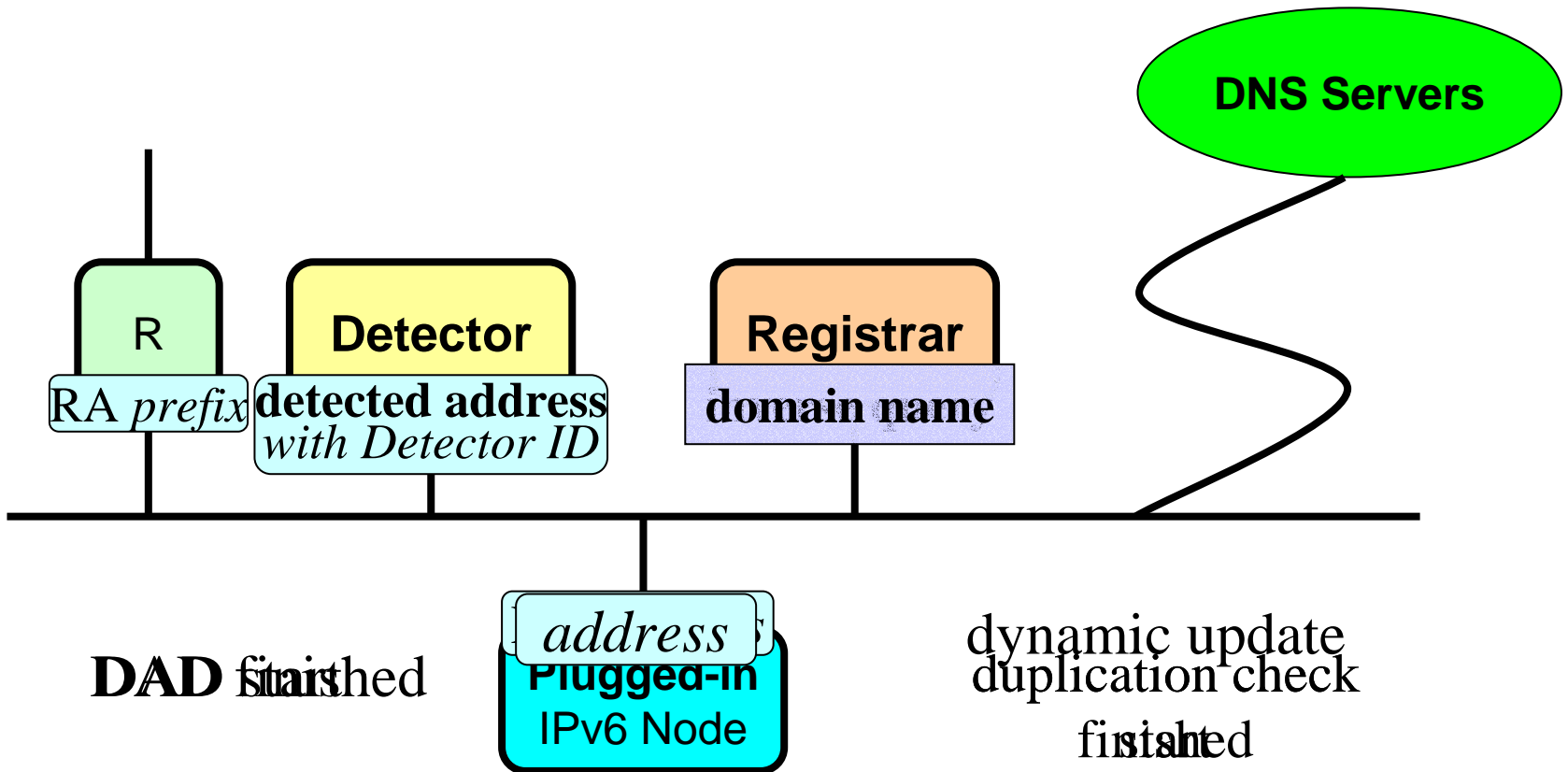
4. ***No explicit methods*** to avoid duplicated name registrations
 - DNS servers accepts Dynamic Updates messages *without doing duplication checks*, if the messages are correctly formatted and authenticated
 - (It is essentially difficult for DNS servers to distinguish overwrite (update) registrations from duplicate registrations)
5. ***No mechanisms*** to control (restrict) the number of issued Dynamic Updates messages for plugged-in nodes
 - It is necessary to control them to minimize the effects of malicious or misconfigured registration requests
6. ***Not clear timings*** to issue registration request messages
 - It is necessary to define trigger timings to start registrations

Basic Design

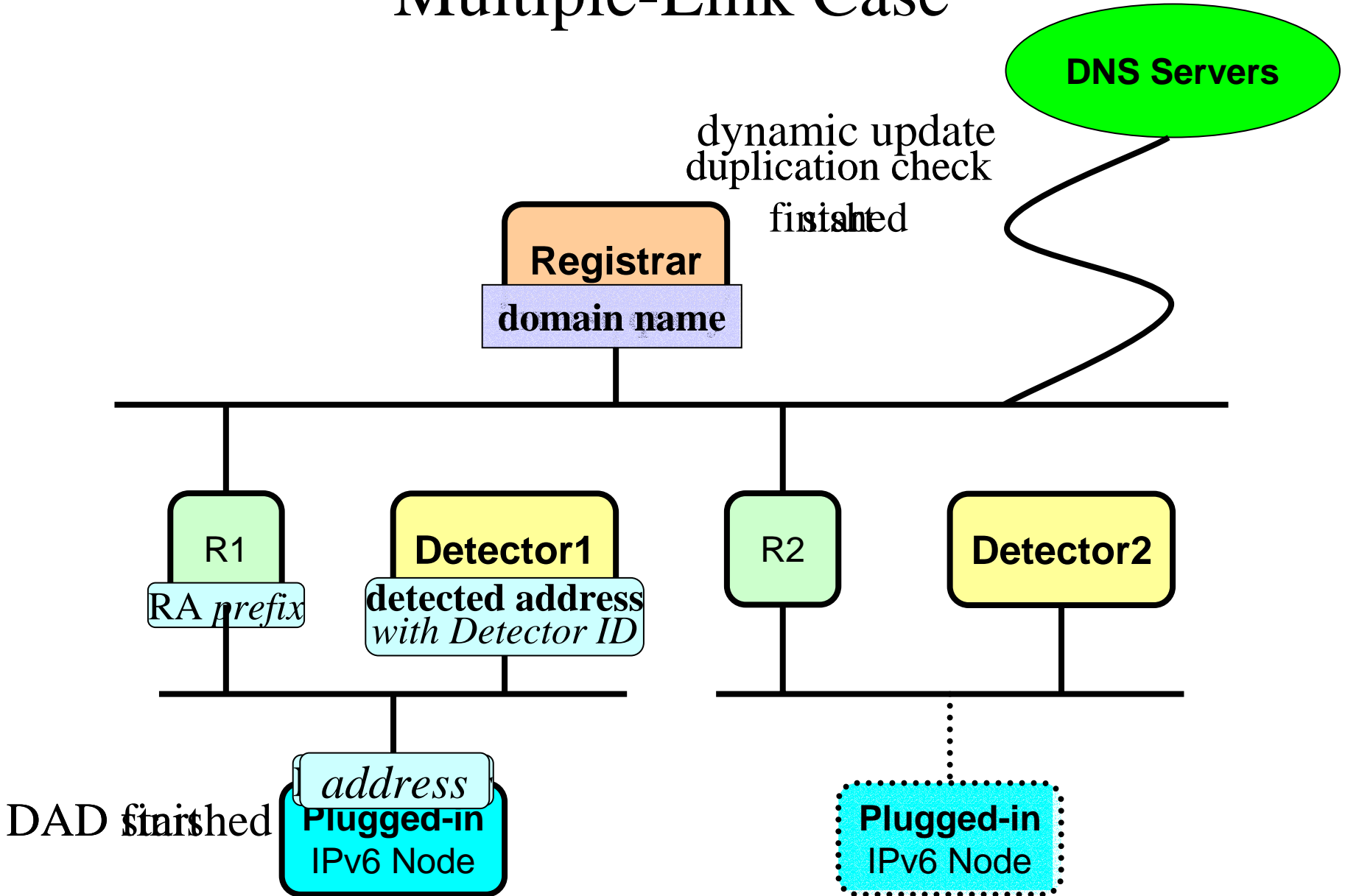
“Domain Name Auto-Registration” mechanism is composed of **two new functions**.

- **“Detector”** function,
 - detects appearances of new plugged-in IPv6 nodes
 - sends “*detected address*” to **“Registrar”**
- **“Registrar”** function
 - checks the received “*detected address*”
 - prepares “*default domain name*” for it
 - registers the “*domain name*” info to DNS servers.

Single-Link Case



Multiple-Link Case



Design Policies

1. To make **the mechanism easy to control**

- Mechanism is maintained by administering only Registrars
- The number of DNS servers' trusted nodes is reduced.
- Administrative costs are reduced.
- Registration information is aggregated at Registrars
- It becomes easy to control registrations and minimize the effects of malicious or misconfigured registration requests.

2. To make **Detector easy to implement**

- To watch for DAD packets on each end link, Detectors must be simple enough to install on any types of IPv6 nodes
- Intelligent operations are not placed on Detector on each end link
- Intelligent operations are concentrated on Registrars

3. To make **the mechanism flexible** and to **apply to various environments** (office networks, home networks, etc.)

- e.g., at home networks, Registrars can be placed at the Provider side, and Detectors can be placed at the User side.

Methods of Detecting Appearances of New Plugged-in IPv6 Nodes

- For “*standard*” plugged-in nodes that **do not issue** special packets to show their appearance
 - Initial procedure is to autoconfigure address and to do DAD
 - DAD packets have sufficient characteristics (issued only when IPv6 nodes are plugged in, and address information is included)
 - **Watch for DAD packets** to detect appearances
- For “*active*” plugged-in nodes that **issue special packets** to show their appearance
(will be discussed further in other documents)

Methods of Controlling Registration

- All candidate information (detected addresses) is checked by using inverse DNS resolving queries of them
- **Only when NO FQDN information for it is found** and it is verified that the detected information is based on first appearance of the plugged-in node, “**default domain name**” for registration is selected and both regular and inverse domain name information are registered to the DNS servers by the Dynamic Update messages.
- Registrar does *not have to have a local database* to maintain the status of the detected information and no DNS registration inconsistency problems are caused
- By restricting the number of Dynamic Update messages from the Registrar per unit of time, the effects of malicious or misconfigured registration requests are minimized.

Naming Rules for Default Domain Names

- FQDN = (Node's original Prefix) + (DNS zone Suffix)
 - **DNS zone Suffix:** given to the Registrar manually
 - **Node's original Prefix:** **discuss here**
- It is *not necessary* to define naming rules *explicitly*
- Each site can define its *own naming rules* (algorithms) per link according to site policy.
- Which naming rule is applied to “*detected address*” is dependent on the **Detector ID**

Naming Rule **Examples** for a Node's Prefix

1. Prefix Letter(s) + Number

–Simplest rule

2. Predefined Names

–Prepare predefined names, and select from them

3. Use Preferences of Plugged-in Nodes.

–Inquire the preference or property, and use the obtained information as a **hint** to define a prefix

- “Passive” indication

use parts of MIB to indicate their preferences

- “Active” indication

define and use special protocol

(will be discussed further in other documents)

Typical Procedures

Plugged-in

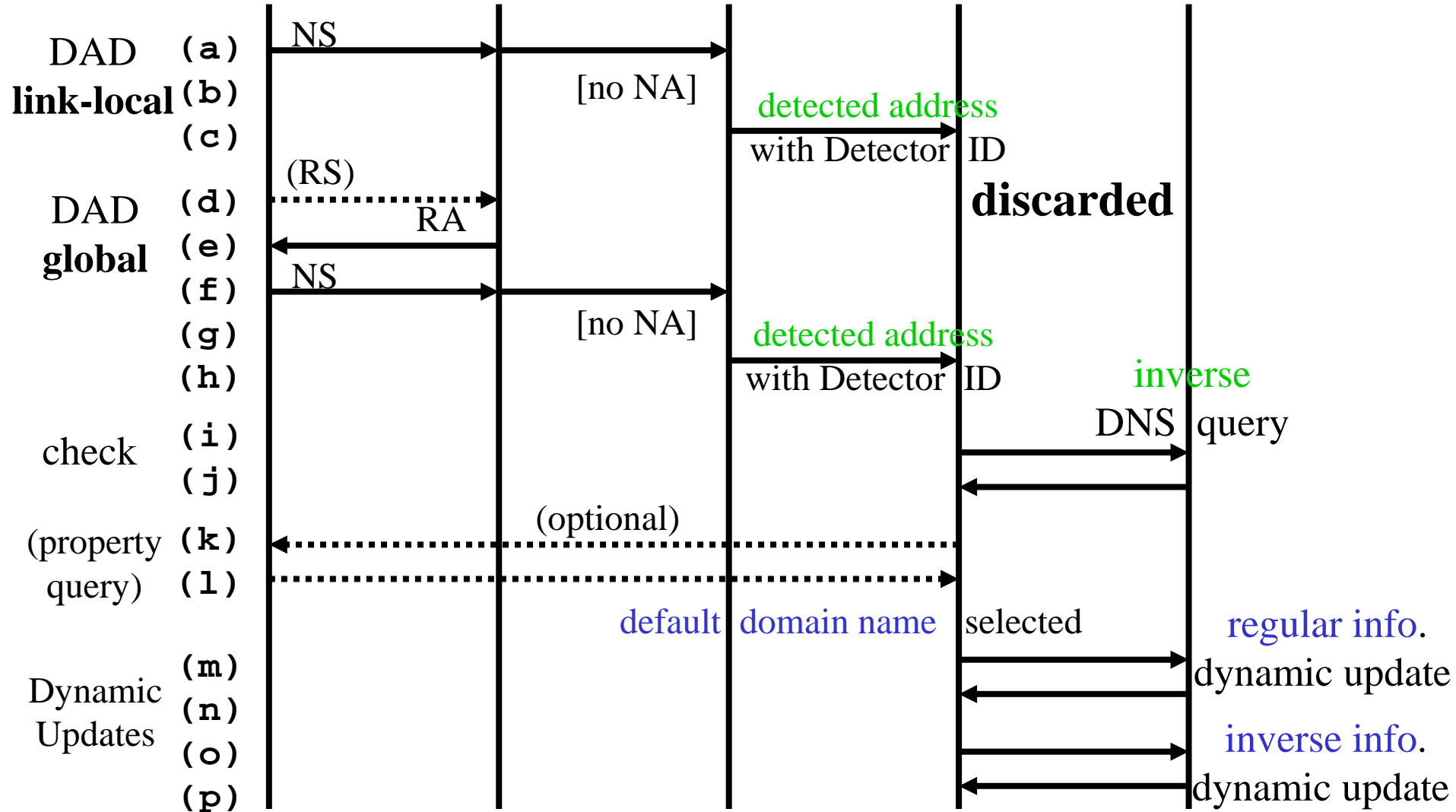
IPv6 Node

Router

Detector

Registrar

DNS servers



Additional Benefits 1/2

- Plugged-in nodes can obtain their assigned “*default domain name*” by simply executing inverse DNS name query with their IPv6 address argument.
- Since all IPv6 nodes have DNS name resolving functions, it can be achieved without installing new functions
- Obtained FQDN information includes **DNS zone suffix** for plugged-in nodes

Plugged-in nodes can obtain **DNS zone suffix** easily

(It cannot obtain suffix list, but one suffix will be enough for plugged-in IPv6 nodes)

Additional Benefits 2/2

- Even under manual (administrative) DNS configuration situation, it is not easy to prepare address information to register to the DNS
- The mechanism can help to collect such address information
- Only by recording received “*detected addresses*” on the **Registrar**, address information is collected

Consideration 1

DAD

- “Domain Name Auto-Registration” mechanism is based on the DAD mechanism.
- Behaviors of the DAD mechanism affect the “Domain Name Auto-Registration” mechanism
- Potential problems on the DAD mechanism have discussed on its definition “Address Autoconfiguration [RFC2462].
- It is not necessary to discuss on this issue here

Consideration 2

Under Extraordinary Situations where DAD packets are not issued

- Without alternative triggers,
appearance detection functions do not work.
(look for alternative triggers?)
- Define special protocol and packets to show
appearances (for “active” plugged-in nodes cases)

Maybe “Domain Name Auto-Registration”
mechanism is not necessary under such situations

Consideration 3

DNS Server Discovery for Plugged-in Nodes

- Plugged-in nodes need to know the location of a DNS server (to query their assigned “default domain name”)
- <draft-ietf-ipngwg-dns-discovery-02.txt> and <draft-ietf-ipngwg-dns-discovery-analysis-00.txt> will show solutions.
- It is not necessary to discuss on this issue here

Consideration 4

Multi-homing (Detector ID selection)

- Detector ID selection procedure on Detector may have something to do with Multi-homing issues
- Prefix of “*detected address*” is compared with prefixes of Detector’s addresses, and prefix-matched address is selected as **Detector ID**
- There is no ambiguity and *no multi-homing problems*
- If prefix of “*detected address*” does not match any of prefixes of Detector’s addresses, it is discarded
- This procedure helps to avoid illegal registrations

Consideration 5

Perfectness of Duplicated Registration Check and (negative) DNS cache

- Only with checking by the inverse query on Registrar, duplicated registration check dose not work perfectly under some critical situations.
- Dynamic Updates [RFC2136] defines transaction atomicity issues. By setting “Prerequisite” for Dynamic Updates messages appropriately, registration check is complemented.
- If a DNS cache function runs on Registrar, it is recommended to disable DNS negative cache [RFC2308] function or shorten TTL of it.

(Duplicated domain name registration does not cause serious problems, but it is recommended to avoid it)

Consideration 6

Site-local or Link-local scope addresses

- The “*Domain Name Auto-Registration*” mechanism is basically designed for domain name registrations for **global unicast** addresses.
- By setting the query scope of the target DNS server appropriately, the mechanism will be able to be applied to domain name registrations for *site-local* and *link-local* scope unicast addresses.

Depend on DNS registration policies

Consideration 7

Issued Packets

- DAD packets are not issues frequently on normal networks
- DNS query packets for duplication check depend on number of issued DAD packets.
(It can be reduced by putting DNS cache function on the Registrar)
- Number of issued Dynamic Update messages are not large (it is in proportion to number of nodes)
- Only on renumbering situations, many DAD packets are issued.

Sample Implementation

- Basic functions *have been implemented* and *verified*
 - on KAME platforms with BIND9
 - on Ethernet
- Detector watches for DAD packets in *promiscuous mode* in this implementation. (because destination addresses of DAD packets are solicited-node multicast addresses)

Next Steps

- Update the I-D
 - Add consideration issues
 - Multi-homing issues
 - DNS cache issues
 - ...
- Only a *scheme* of “Domain Name Auto-Registration” mechanism is described
 - *If necessary*, I can define simple protocols
 - between Detector and Registrar
 - between Registrars (relay protocol)
 - ...