S.Gopinath Rao
NRG, USM Malaysia

K. Ettikan
Intel ASG, Malaysia

Internet Draft
Document : <draft-gopinath-ipv6-hostname-auto-reg-02.txt>
Expires in six months

3 December 2002

IPv6 Hostname auto-registration Procedure
<draft-gopinath-ipv6-hostname-auto-reg-02.txt>

Status of this Memo

Abstract

This document describes a new method, which will automatically acquire the host name of active IPv6 nodes and register it to the local Domain Name System (DNS) server. Active IPv6 nodes are nodes that are connected to a network.

Our proposed new method [6] will automatically learn some useful information from all the IPv6 nodes, such as the host name and the IP address. The Agent using a new method will keep all the information that is send by the nodes. The Resolver will add the host name together with the IPv6 address to the DNS. This will be implemented without the need for any changes at the clients' site.

The objective of this document is to specify a new procedure and also the algorithm used in the procedure.

Table of Content

1. Introduction

Domain Name System has long been used for resolving host name to IP address and vice versa. The need of automatically sensing the host name and IP addresses on a network by DNS does not arise because IPv4 addresses are easy to remember. The improvement of DNS with the introduction of DNSv6 should have included this new feature so that it would be an intelligent application.

The reason for the proposed method and the solutions are as follows.

1. The main factor that drives us to introduce this procedure is the IPv6 addressing scheme. 128 bits IPv6 address is indeed very long to remember compared to IPv4. This hexadecimal representation makes unit of network difficult to be identified and remembered based on this IP address. Instead of remembering oneÆs IP address, it will be easier for us to just use the host name.

To solve this problem, a new registration procedure that will automatically sense all the IPv6 nodes on a specific link and add all the names of the nodes within that domain to its DNS. This will also give the users the freedom to choose their own host name for their node/s. An algorithm will be used to make sure that the host name is unique within the domain.

2. Maintenance of these addresses is another hassle for system administrator. In current DNS, system administrators manually enter the host name and the IP address associated with it into the host name file or the DNS. Whenever there is a change to a node in the domain, system administrator must manually edit the host name file to make sure that the new nodeÆs information is updated to the list.

   To solve the above-mentioned problem, the new procedure will periodically check and update the nodesÆ information into the host name file, if there are changes to the nodes. Even though we can use dynamic update [8] to send updated information to the DNS, this is restricted to certain operating system that implements the dynamic update.

By using this procedure, cost of handling and maintaining the host names can be reduced in huge amount. The amounts of time spend by system administrator on setting up the host names will also be reduced.

This document proposes a new procedure to dynamically handle the host names on an IPv6 LAN, which is named IPv6 Hostname DNS auto-registration procedure. This document also describes the algorithm used in the new procedure.

The implementation of this procedure involves few existing methods. For example, to get the nodes information, the use of Neighbor solicitation (Duplicate Address Detection (DAD) packet) and also the ICMPv6 packet will be implemented. This new procedure supports all types of IPv6 addresses available, both the stateless [1] and statefull IPv6 addresses. This includes link local addresses, site local addresses and global unicast addresses.

2. Study on existing and similar application [6]

The dynamic naming of host name was not implemented in IPv4 because IPv4 are easy to remember compared to IPv6 addresses. The only similar proposed method was discussed in a draft titled Domain Name Auto-Registration for plugged in IPv6 [3].

2.1 Domain Name Auto-Registration for plugged in IPv6 <draft-kitamura-ipv6-name-auto-reg-00.txt>

This draft has proposed to use a method for domain name auto-registration for plugged in IPv6 nodes. The method is divided into 2 functions, which are ôdetectorö function and ôregistrarö function. The ôdetectorö is used to detect the appearance of new IPv6 nodes and send it to a ôRegistrarö. The ôRegistrarö is used to prepare appropriate domain name information for registration and to register it by sending dynamic update messages to the corresponding request.

Two approaches are used in the draft to detect the nodes. One of the methods is by detecting the DAD packets. The other approach is by using the SNMP procedure. The setback of this method is that it needs to be implemented at both the client and the server site. A new Internet standard like ICMP needs to be set for this approach to work.

Our proposed protocol is similar to the above draft except that our protocol uses the DAD together with the ICMPv6 packet to get the host name.

Some of the idea discussed in this approach can be shared to produce a better method and algorithm.

3. Design of the new auto-registration procedure

This section describes the design of the new procedure, IPv6 Hostname auto-registration procedure [9]. The algorithm used in this procedure to solve the above mentioned problems would also be discussed in this section.

3.1 Design Issue

The new auto-registration procedure can be incorporated into the existing DNS. There are few issues involved in designing this procedure.

1. Assignment of multiple IPv6 addresses to a single interface makes names assignment a complex problem. Each IPv6 active nodes is capable of having multiple IP addresses that may have different reachability scope (link local, site local or global). It can get the IPv6 address using both stateless and statefull configuration method. The IPv6 nodes will have a default link local address, which is configured automatically using the autoconfiguration method [1]. Each node would also be able to construct itÆs own global address using the prefix advertised by a router on the same link. Beside that we can also manually

assign a global IPv6 address to that same interface. So, we

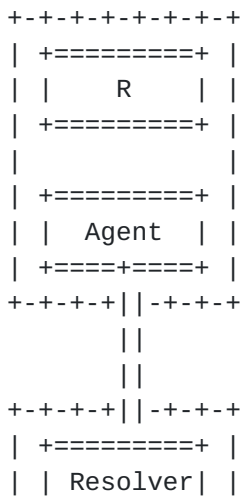
must make sure that all the addresses point to the same host name, unless the system administrator wants to configure each IP addresses with different host names. To select an address that will bind to the host name, default address selection can be used [10]. The default address selection is a method to select a source address or a destination address from multiple addresses that is assigned to a single interface.

2. The implementation of a new method usually requires a lot of changes at both the server and the client. To minimize the changes on nodes, the new proposed procedureÆs implementation required the development at only one site. This is implemented using the existing methods available, which supports all operating systems. Some of the methods developed uses the ICMPv6 and the neighbor discovery in IPv6. This will not only reduce the impact of introducing new protocol in the Internet but also make the implementation of new system much easier.

3. The locations of the entities on the network need to be decided. This is to make sure that optimum results can be achieved. If a DNS is shared between 2 LANs, the entities must be implemented so that it can detect all the nodes in both the LANs. This is because the DAD and the ICMPv6 packets used for detection of the nodes wonÆt be able to pass through the router that separates the LANs.

3.2 Design Method

Hostname auto-registration procedure is composed of 2 entities, the Agent, which detects all the IPv6 nodes on a link and the Resolver, which updates the DNS with the latest host name information received from the Agent/s.



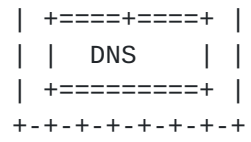


Fig. 1 The Overall architecture of the new procedure

Figure 1 shows the overall architecture of the new procedure and the communication between the two entities. The Agent resides on a router while the Resolver is placed together with the DNS. The Agent sends the information it captures to the Resolver using the communication pipe between the two entities.

3.2.1 The Agent

The AgentÆs function is to detect all the nodes on a link and keep the information in a specific file. It uses two detection methods for getting the information from a node. The methods are using the DAD packet and the ICMPv6 packet. The detection of DAD packets is restricted on a single LAN because the router wonÆt allow the DAD packets to pass through. So, the Agent must be placed on all the LANs in order to detect the DAD packets issued by the nodes. The best location for the Agent to be placed is with the router.

The Agent need to be manually configured first so that the information that it detects from the IPv6 nodes can be send to the Resolver to be updated into the DNS. It will also do periodic checking on the LAN to make sure that new changes to the nodes are updated to the DNS.

3.2.2 The Resolver

This entity sets an appropriate host name for registration into the corresponding DNS server. The host name is created from the information that is sent by the Agent. It later updates the host name into the DNS.

The Resolver must have a method to verify the uniqueness of the host name in the network. This is achieved by using the naming algorithm to name the nodes. This algorithm is also used to minimize the effects of misconfigured registration request by the nodes. All the names given by the Agent is kept in a specific file. This is to make sure that the information can be retrieved easily whenever necessary.

The Resolver must be configured so that it directly updates the

nodesÆ information to the DNS.

The main file in the Resolver consists of all the information send by the Agents. After updating the file, it will be copied by the Agent for their reference. This is important because a same node will move from one subnet to another subnet with same information. This scenario is further described in the next section.

In order to make the Resolver to work effectively, it must be placed together with the DNS. This will make sure that the information is updated as soon as the Resolver received it from the Agent.

3.2.3 Agent-Resolver Communication

All the Agents, which communicate to the same Resolver, will have a copy of the updated database from the Resolver. This database will be used to check if changes occur to a node as described below.

The link local address of each machine will be added in a new file. This address will be bind together with the global address and the host name of the machine in the file. The link local address can be used to check the uniqueness of the machine because the link local address is unique as it is generated using the Ethernet address and it never changes wherever the node moves. This information will be compared every time the Agent receives the updated information from a node before the host name is updated to the DNS via the Resolver.

In the development of the Agent and the Resolver, multi-link scenario needs to be taken into consideration. It is a situation where a node moves from one subnet to another subnet and uses a same DNS. It is possible for few scenarios to happen when a node moves from one subnet to another and when some changes happened to a node.

1. A node uses new IP address but same host name and NIC card.

   This is a complicated scenario. In this case, the link local address together with the new global address will be compared with the information in the AgentÆs database. Since the link local address is still in the database, the old global address associated with the link local address will be deleted from the Resolver. The new global address will be updated to the database. With this algorithm we can maintain the host name and at the same time change the global address.

2. A node uses new IP address using new NIC card but same host name

   This is same as two nodes asking the same host name. In this situation, the naming algorithm will give a new name for the

node. The old data will be overwritten after some time depends on system administrator (see section (g)). Until then the node has to use a new name generated using the naming algorithm or change the host name. This can be overcome if system administrator manually deletes the old information. The same host name cannot be used to avoid multiple updates by a node that uses the same name. This will be a hassle because every time a new node with same name activated, it will overwrite the old IP address. With this, some of the host name will be missing from the list.

3. A node changes both host name and IP address replacing old information

   The new information will be compared with the information in the file. First the algorithm will check the link local address in the file. Since the link local address is already in the file, the global and the host name will be compared. If both the information is new, then the old will data be over written with the new information. The Agent will send this updated information to the Resolver. The Resolver will then update the new information of the node to the DNS.

4. A node uses a new NIC card with old IP address and Host Name

   First the link local address will be compared with link local address in the file. If there is no entry, then the link local address will be entered and the host name will be compared with the list. If the host name is already exist, then a new name will be generated. If not, the host name will be entered together with the link local address and global IP address.

5. A node uses new NIC and Host Name but old IP address

   This is similar to scenario (4) above.

6. A node changes the host name but uses the same IP address

   The link local address of the specific node will be compared with the link local addresses in the file. If the IP address is already in the file, then the global IP address will be compared. If there is no change, then the host name will be compared. The new host name will overwrite the existing host name in the file before updating it in the DNS.

7. A node is inactive for certain duration

   This is a case when a node doesnÆt response to AgentÆs query. This scenario is possible when a node is crashed or not booted by the owner for some time. The Agent will check the node for 3 or 4 months (depends on the setting. If the node is still inactive after the duration, the information will be overwritten by new a node requesting for the same name. If the owner wants to keep the information after that time period, system administrator need to set a flag in the DNS indicating not to delete the nodeÆs information. This flag can be set as an additional field in the DNS or in the Resolver.

8. Two or more nodes request for the same host name

   Using the naming algorithm (section 3.4.1 & 3.4.2), the nodes will be given the name requested using first come first serve basis. When the nodes rebooted, the names will be maintained as given earlier.

3.3 Detection Method

This is the main method in the new procedure. In this method, the Agent will detect any new IPv6 nodes on a link, check the uniqueness of that node on that link and then update the information in the DNS.

There are 2 kinds of detection method used in the procedure. Both the detection methods enable a mechanism to work without the introduction of a new protocol and without any functions installed into the IPv6 nodes. This two methods need to be combined to make sure that the nodes are configured correctly with the host name in the DNS.

1. Duplicate Address Detection (DAD)

   The nodes can be detected by monitoring the duplicate address detection (DAD) packet issued by nodes that has been activated. The DAD packet is issued as part of neighbor solicitation in neighbor discovery [2]. The implementation of duplicate address detection is compulsory on all IPv6 nodes, regardless of whether they are obtained through stateful, stateless or manual configuration. DAD has the same functions as the ARP packet in IPv4 but DAD provides more information. It is issued to ensure that all configured addresses are likely to be unique on a given link [1].

By monitoring the DAD packets on a link, the Agent will capture it and the packets will be analyzed. Then using ICMPv6 packet, the Agent will issue a query to that node for more information (see section 2 of 3.3). All the information that has been captured will be kept in a specific file for reference and send to the Resolver. It should be noted that the DAD packets could only be detected on a link-local, thus the Agent must be positioned on a specific location to detect the packets (see section 3.4).

2. ICMPv6 û node information query and node information reply

This is the most important part in the Agent. ICMPv6 type that will be in the Agent are node information (NI) query and node information reply. The use of ICMPv6 NI query as type 139 and ICMPv6 NI reply type 140 is already defined in [4]. Currently the ICMPv6 packet type for querying nodes information is still in the process of upgrading. Node information query and node information reply are information-based messages. NI query packet is sent by the ôQuerierö node to ask some information from the ôResponderö node. The ôResponderö node, upon receiving the ICMPv6 NI query packet, replies to the querier with the information requested by the ôQuerierö node.

This will be an exact method that will be used by the Agent to query IPv6 nodes on a link. The ICMPv6 packet can be addressed to a unicast address, a link local multicast address or an anycast address depending on the scope of the information required. In this implementation we will skip the use of anycast and only use the unicast and the multicast address for querying information.

The ICMP source address should be that of the address where the Agent reside. This is to make sure that the reply is send back to the Agent. The destination address can be chosen from the 2 types of addresses according to the algorithm. There will be 2 scenarios for choosing the destination address:

i) If a node has been detected by capturing the DAD packet, then the destination address will be the unicast address of the detected node. After getting the IPv6 address from the DAD packet, the Agent will use it as a destination address in the ICMPv6 NI query requesting for host name (type 2 for qtype field in the ICMPv6 NI query).

ii) The Agent uses the predefined link local multicast address, which is FF02::1 as the destination address to do periodic

checking on a link to detect any changes or updates to a node. This query need to be sent from time to time so that

the changes can be detected earlier to make the new procedure efficient. The frequency of sending this type of packets will also need to be considered. Even though the ICMPv6 packets send are small, if it is send too frequent, it might flood the network. For it to work efficiently the internal time taken for sending the ICMPv6 packet need to be acceptable.

The content of qtype in the ICMPv6 packet that will be used by the Agent is either type 2 (DNS names) or 3 (Node Addresses) and the implementation of the Agent is done with an assumption that all the nodes are configured with the latest ICMPv6, which implements the ICMPv6 node information query. Upon receiving an ICMPv6 NI query, the nodes must check the queryÆs IPv6 destination address and discard it if it is not intended for the node. The destination address of the query should be either the IP address of the receiving node or the multicast address that the receiving node joined earlier. The use of these ICMPv6 type is described is detail in the IPv6 Node Information Queries draft [4].

3.3.1 Detecting the nodes

The Agents continuously monitor for DAD packets in the network. The DAD packets that it captures will be analyzed. It will compare IP address with the information in the database the Agent copied from the Resolver. If the IP address is already exists in the database, the Agent will further query the host name of the node. If there is no information in the database, the Agent will further query the node using the ICMPv6 packet. After getting the necessary information, the Agent will pass the information to the Resolver.

```
BEGIN
     1. AGENT MONITORS FOR DAD PACKETS AND CAPTURES IT
          CHECK IF IP ADDRESS EXISTS IN AGENTÆS FILE
               IF EXIST
                    THEN IGNORE THE DAD PACKET
               ELSE GOTO STEP 2
     2. USE ICMPV6 PACKET TO REQUEST THE INFORMATION FROM THE NODE
END
```

3.3.2 Algorithm using unicast address as destination in the ICMPv6 packet

The Agent sends unicast ICMPv6 query to a specific node requesting more information about the node. If there is no reply from the node, the Agent will request again. The Agent will kept requesting for few

times if there is no reply from the node. Once the node replied, the

Agent will extract the information and send it to the Resolver if the information is new or some changes occurred the node.

```
BEGIN
     1. AGENT SENDS ICMPV6 UNICAST QUERY
     2. AGENT CAPTURES THE REPLY
          IF NO REPLY FROM THE NODE
             GOTO STEP 1
          ELSE
               EXTRACT INFORMATION AND UPDATE THE FILE
END
```

3.3.3 Algorithm using multicast address as destination in the ICMPv6 packet

This algorithm is similar to unicast query. The different is the target node/s is a multicast group. Once the Agent received the reply from the node, it will queue the information and check in the database copied from the Resolver. If there are changes, then the Agent will send the updated data to the Resolver.

```
BEGIN
  1. AGENT SEND MULTICAST QUERY REQUESTING INFORMATION
  2. AGENT READS THE QUEUE (REPLY FROM NODES)
       IF IT IS NOT EMPTY
            GOTO STEP 3
       ELSE GOTO END
  3. CHECK THE INFORMATION CAPTURED
       IF INFORMATION IS NEW
            GOTO STEP 4
       ELSE IF INFORMATION EXISTS BUT PART OF IT IS DIFFERENT
            GOTO STEP 5
       ELSE IF INFORMATION IS SAME
            GOTO STEP 2
  4. UPDATE THE FILE
       GOTO STEP 2
  5. CHANGE INFORMATION IN THE FILE WITH THE NEW INFORMATION
       GOTO STEP 2
END
```

3.4 Updating the DNS by the Resolver

The ResolverÆs function is to update the DNS with the information received from the Agent/s. It will receive the information from the Agent/s and will activate the naming algorithm before updating it to the DNS. The Resolver must be configured to communicate with the DNS server so that the information received will be automatically

updated. The naming algorithm is used to make sure that the names are unique in link.

3.4.1 Naming

The unique feature of the auto-registration procedure is the naming of the IPv6 nodes. The procedure allows the users to set their preferable name for their IPv6 node/s. Even though we donÆt have the control over the naming, the procedure still decides whether to use the specific name given by each user to their nodes. System administrator also has a choice to use this algorithm or manually configure the names. He also can use both manual and the approach described in this document.

The Resolver uses first come first serve basis in the naming algorithm. If there is a conflict of names between 2 IPv6 nodes, the node that registered first will be given the name. For the second node, a new name will be generated using the name given by the node. For example if the conflict name is ôipv6-dnsö, then the algorithm will assign a new name for the second node based on that conflict name such as ôipv6-dns-2ö. In this way the each node will still have a unique name.

The technique used in this draft is different from the approach documented in ôDomain Name Auto-Registration for Plugged-in IPv6 Nodesö [3]. In that approach, the naming is still handled by system administrator and no freedom given to each user. The second approach, in that draft, uses predefined name, which the registrar randomly generates. He also uses a method to detect the names automatically but in the method described, all the nodes must define a special MIB. This is difficult because it involve both the server and the client (nodes).

To make the process simpler, we only use 2 ways for naming a node, which is either configured by system administrator for fix host name for an IPv6 node or use the name given by the node.

Another important issue is the format of the naming. Before passing the information captured to the Resolver, the Agent should check if host name is in a valid format. If it is not, then the information given by a node will be dropped. This is to ensure that weird names are not registered into DNS.

3.4.2 Naming algorithm

The Resolver first checks the IP address given by the Agent with the IP address in its database. If the IP address is new, the Resolver

checks if the name given by the node exists in the database. If not,

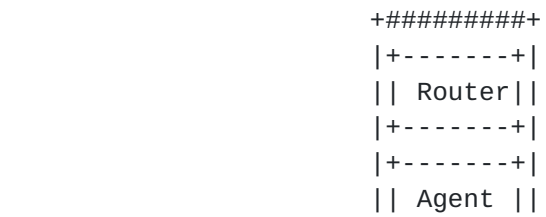
the IP address together with the host name will be registered into DNS. If the name is already exists, the Resolver will generate a similar name for the node as mentioned in the previous section.

```
BEGIN
  1. RESOLVER COMPARES THE HOST NAME FILE GIVEN BY THE  AGENT WITH
  THE ITÆS FILE
       IF IP ADDRESS IS NEW
            GOTO STEP 2
       ELSE GOTO STEP 3
  2. CHECK IF NAME IS NEW
       IF NAME IS NEW
            GOTO STEP 4
       ELSE GOTO 5
  3. COMPARE THE NAME IN THE FILE THAT IS ATTACHED TO THE IP ADDRESS
       IF IT IS NOT SAME
            GOTO STEP 5
       ELSE END
  4. UPDATE IP AND NAME TO THE DNS
       END
  5. GENERATE A NEW SIMILAR NAME
       GOTO STEP 4
END
```

3.5 Location of the Agent and the Resolver in a network

Since the Agent must be placed to detect the appearance of IPv6 nodes, the location of the Agent is very much restricted.

The best location for the Agent is to be placed together with the router while the Resolver need to be placed with the DNS. Using this method, each link will have an Agent, which is manually configured to directly communicate with the Resolver. This kind of configuration is needed for the Agent to capture the DAD packets issued by the nodes. If the Agent is not installed in every link, the DAD packets cannot be detected because the router will drop all the DAD packets from going out of it.




||Resolver||
|+-------+|

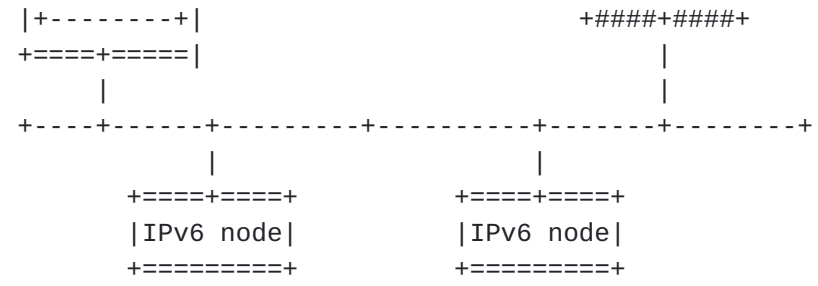


Fig. 2 The configuration on a single LAN with DNS located in the same LAN

Figure 2 shows an example configuration of the entities on a single LAN, with the DNS in the same subnet. In this case, the Agent will directly communicate with the Resolver to update the host name file. The Agent will dynamically get all the names and update the Resolver from time to time. The Resolver will then update the DNS with this latest information.

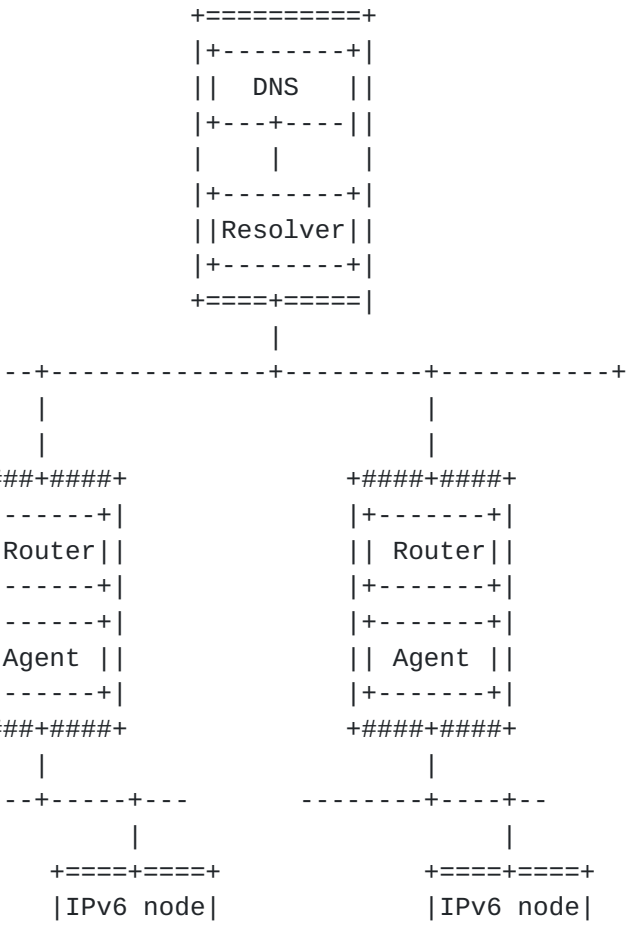


Fig.3 Configuration on 2 different LANs sharing one DNS

Figure 3 shows the implementation where one DNS is shared among nodes from different LANs. In this scenario, each LAN will have an Agent that will communicate directly with the Resolver that resides on a DNS. To avoid the conflict of names from different LAN, each Agent will make a copy of host name files from the Resolver. In the case of multiple Agents accessing the Resolver at the same time, the procedure will allow the first Agent to update the changes to the Resolver. It will be first come first server basis. In this case the naming of nodes will be organized without the names conflicting with other nodes from different LAN/s.

4. Conclusion

The implementation of the auto-registration procedure is aimed to solve two major problems discussed above. In the future the capability of this new procedure can be extended by adding more functions and make it more secure in transmitting the data.

5. Security Consideration

Security is one of the main issues in developing this new procedure. Since the Agent directly sends the information to the Resolver, the information must be send securely. The other transmission that needs to be handled securely is between the Resolver and the DNS.

It also must make sure that the message send by the Agent are from the Agent that is registered with the Resolver. If not the data should be ignored. This is to make sure that no malicious data being send to the Resolver.

The security issue for this new technique needs to be discussed further.

6. References

[1] S. Thompson, T. Narten, ôIPv6 Stateless Address Autoconfigurationö, RFC 2462, December 1998.

[2] T. Narten, E. Nordmark, ôNeighbor Discovery for IP Version 6 (IPv6)ö, RFC 2462, December 1998.

[3] H. Kitamura, ôDomain Name Auto-Registration for Plugged-in IPv6 Nodesö, <draft-kitamura-ipv6-name-auto-reg-00.txt>, 13 July 2001.

[4] M. Crawford, ôIPv6 Node Information Queriesö, <draft-ietf-ipngwg-icmp-name-lookups-07.txt>, 28 August 2000.

[5] IEEE, ôGuidelines for 64-bit Global Identifier (EUI-64) Registration Authorityö, http://standards.ieee.org/regauth/oui/tutorials/EUI64.html, March 1997

[6] Gopinath Rao Sinniah, Ettikan Kandasamy Karuppiah and R. Sureswaran, ôHost Name Resolution Protocol (HNRP) û The existing implementation and the need for a new protocolö, Proceedings IEEE MICC2001, Kuala Lumpur, October 21-24.

[7] R. Hinden, S. Deering, ôIP Version 6 Addressing Architectureö, RFC 2373, July 1998.

[8] P. Vixie, S. Thompson, Y. Rekhter and J. Bound, ôDynamic Updates in the Domain Name Systemö, RFC 2136, April 1997.

[9] Gopinath Rao Sinniah, Ettikan Kandasamy Karuppiah and R. Sureswaran, ôHost Name Resolution Protocol (HNRP) û The implementation methodö, Proceedings APAN Conference 2001, Penang, Aug. 20-22.

[10] R. Draves, "Default Address Selection for IPv6", <draft-ietf-ipng-default-addr-select-05.txt>, 4 June 2001.

7. Authors' Addresses

Gopinath Rao Sinniah
Faculty of Engineering and Computer Technology,
Asian Institute of Medicine, Science and Technology (AIMST),
2, Persiaran Cempaka, Sungai Petani,
Kedah, Malaysia.
Tel: +604-4422884
Fax: +604-4422881
Email: gopi@nrg.cs.usm.my

Ettikan Kandasamy Karupiah
ASG Penang & Shannon Operations,
Intel Microelectronis (M) Sdn. Bhd.,
Bayan Lepas Free Trade Zone III,
Penang, Malaysia.
Tel: +60-4-859-2591
Fax: +60-4-859-7899
Email: ettikan.kandasamy.karuppiah@intel.com