ADDRCONF Working Group   Susan Thomson, Bellcore
INTERNET-DRAFT   Thomas Narten, IBM
<draft-ietf-addrconf-ipv6-auto-04.txt>   October 4, 1995

IPv6 Stateless Address Autoconfiguration

Status of this Memo

Abstract

This document specifies how hosts autoconfigure addresses for their interfaces in IP version 6. In particular, the document describes the steps a host takes in determining whether address autoconfiguration should be used, and if so, whether to use the stateful mechanism, the stateless mechanism or both. This document also specifies stateless address autoconfiguration, and how nodes verify the uniqueness of an address before assigning it to an interface. Stateful address autoconfiguration is specified elsewhere.

Contents

1. INTRODUCTION

This document specifies how hosts autoconfigure their interfaces in IP version 6. The autoconfiguration process includes determining what information should be autoconfigured (addresses, other information, or both), and in the case of addresses, whether they should be obtained through the stateless mechanism, the stateless mechanism, or both. This document also specifies stateless address autoconfiguration. Stateful address autoconfiguration is specified elsewhere.

IPv6 defines both a stateful and stateless address autoconfiguration mechanism. Stateless autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. The stateless mechanism allows a host to generate its own addresses using a combination of locally available information and information advertised by routers. Routers advertise prefixes that identify the subnet(s) associated with a link, while hosts generate an "interface token" that uniquely identifies an interface on a subnet. An address is formed by combining the two. In the absence of routers, a host can only generate link-local addresses. However, link-local addresses are sufficient for allowing communication among nodes attached to the same link.

In the stateful autoconfiguration model, hosts obtain interface addresses from a server. Servers maintain a database that keeps track of which addresses have been assigned to which hosts. In addition to addresses, stateful servers can also supply configuration information and parameters to a host. The stateful autoconfiguration mechanism allows hosts to obtain addresses, other configuration information or both from a server. Stateless and stateful autoconfiguration complement each other. For example, a host can use stateless autoconfiguration to configure its own addresses, but use stateful autoconfiguration to obtain other information. Stateful autoconfiguration is described in [DHCPv6].

The stateless approach is used when a site is not particularly concerned with the exact addresses hosts use, so long as they are unique and properly routable. The stateful approach is used when a site requires tighter control over exact address assignments. Both stateful and stateless address autoconfiguration may be used simultaneously. The site administrator specifies which type of autoconfiguration to use through the setting of appropriate fields in Router Advertisement messages [DISCOVERY].

IPv6 addresses are leased to an interface for a fixed (possibly infinite) length of time. Each address has an associated lifetime that indicates how long the address is bound to an interface. When a lifetime expires, the binding (and address) becomes invalid and the address may

be reassigned to another interface elsewhere in the Internet. To handle the expiration of address bindings gracefully, an address goes through two distinct phases while assigned to an interface. Initially, an address is "preferred", meaning that its use in arbitrary communication is unrestricted. Later, an address becomes "deprecated" in anticipation that its current interface binding will become invalid. While in a deprecated state, the use of address is discouraged, but not strictly forbidden. New communication (e.g., the opening of a new TCP connection) gives preference to using a non-deprecated address, with use of the deprecated address restricted to applications that have been using the deprecated address already and would have difficulty switching to another address without a service disruption.

Finally, this document describes the algorithm a node employs to verify that an address it is about to assign to an interface is unique on the link. The "duplicate address detection" algorithm is used before an address is actually used, independent of whether the address was obtained via stateless or stateful autoconfiguration.

The autoconfiguration process specified in this document applies only to hosts and not routers. Since host autoconfiguration uses information advertised by routers, routers will need to be configured by some other means. However, it is possible for routers to use the mechanism described in this document for generating a link-local address. All nodes (not only hosts) should use the duplicate address detection procedure.

Section 2 provides definitions for terminology used throughout this document. Section 3 describes the design goals that lead to the current autoconfiguration procedure. Section 4 provides an overview of the protocol, while Section 5 describes the protocol in detail.

2. TERMINOLOGY

IP - Internet Protocol Version 6. The terms IPv4 and IPv6 are used only in contexts where necessary to avoid ambiguity.

node - a device that implements IP.

router - a node that forwards IP packets not explicitly addressed to itself.

host - any node that is not a router.

upper layer - a protocol layer immediately above IP. Examples are

```
                 transport protocols such as TCP and UDP, control
                 protocols such as ICMP, routing protocols such as OSPF,
                 and internet or lower-layer protocols being "tunneled"
                 over (i.e., encapsulated in) IP such as IPX, AppleTalk,
                 or IP itself.

   link        - a communication facility or medium over which nodes can
                 communicate at the link layer, i.e., the layer
                 immediately below IP.  Examples are Ethernets (simple
                 or bridged); PPP links; X.25, Frame Relay, or ATM
                 networks; and internet (or higher) layer "tunnels",
                 such as tunnels over IPv4 or IPv6 itself.

   interface   - a node's attachment to a link.

   autoconfigurable interface
               - an interface that has been configured by system
                 management to perform autoconfiguration.

   packet      - an IP header plus payload.

   address     - an IP-layer identifier for an interface or a set of
                 interfaces.

   unicast address
               - an identifier for a single interface. A packet sent to
                 a unicast address is delivered to the interface
                 identified by that address.

   multicast address
               - an identifier for a set of interfaces (typically
                 belonging to different nodes). A packet sent to a
                 multicast address is delivered to all interfaces
                 identified by that address.

   solicited-node multicast address
               - a multicast address to which Neighbor Solicitation
                 messages are sent. The algorithm for computing the
                 address is given in [DISCOVERY].

   link-layer address
               - a link-layer identifier for an interface.  Examples
                 include IEEE 802 addresses for Ethernet links and E.164
                 addresses for ISDN links.

   link-local address
               - an address having link-only scope that can be used to
                 reach neighboring nodes attached to the same link.  All
```

interfaces have a link-local unicast address.

site-local address
- an address having scope that is limited to the local site.

global address
- an address with unlimited scope.

communication
- any packet exchange between nodes that requires that the address of each node used in the exchange remain the same for the duration of the packet exchange. Examples are a TCP connection or a UDP request-response.

tentative address
- an address whose uniqueness on a link is being verified, prior to its assignment to an interface. A tentative address is not considered assigned to an interface in the usual sense. An interface discards received packets addressed to a tentative address, but accepts Neighbor Discover packets related to duplicate address detection for the tentative address.

preferred address
- an address assigned to an interface whose use by upper layer protocols is unrestricted. Preferred addresses may be used as the source or destination address of packets sent from or to the interface.

deprecated address
- An address assigned to an interface whose use is discouraged, but not forbidden. A deprecated address should no longer be used as a source address in new communications, but packets sent to deprecated address are delivered as expected. A deprecated address may continue to be used as a source address in communications where switching to a preferred address causes hardship to a specific upper-layer activity (e.g., an existing TCP connection).

valid address
- a preferred or deprecated address. A valid address may appear as the source or destination address of a packet, and the internet routing system is expected to be able to deliver packets sent to a valid address.

invalid address
- an address that is not assigned to any interface. A valid address becomes invalid when its deprecation lifetime expires. Invalid addresses should not appear as the   destination or source address of a packet. In the former case, the internet routing system will be unable to deliver the packet, in the later case the recipient of the packet will be unable to respond to it.

preferred lifetime
- the length of time that a valid address is preferred. When the preferred lifetime expires, the address becomes deprecated.

valid lifetime
- the length of time an address remains in the valid state. The valid lifetime must be greater then or equal to the preferred lifetime. When the valid lifetime expires, the address becomes invalid.

interface token
- a link-dependent identifier for an interface that is (at least) unique per link. Stateless address autoconfiguration combines an interface token with a prefix to form an address. An IEEE 802 hardware address is an example of a possible interface token. The manner in which an interface token is created and its length is link-specific, and is described in the specification for the particular link type (e.g., [IPv6-ETHER]).

2.1. Requirements

Throughout this document, the words that are used to define the significance of the particular requirements are capitalized. These words are:

MUST
This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.

MUST NOT
This phrase means the item is an absolute prohibition of this specification.

SHOULD
This word or the adjective "RECOMMENDED" means that there may exist

valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

SHOULD NOT
This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighted before implementing any behavior described with this label.

MAY
This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example, another vendor may omit the same item.

3. DESIGN GOALS

Stateless autoconfiguration is designed with the following goals in mind:

- o Manual configuration of individual machines before connecting them to the network should not be required. Consequently, a mechanism is needed that allows a host to obtain or create unique addresses for each of its interfaces. Address autoconfiguration assumes that each interface can provide a unique identifier for that interface (e.g., an "interface token"). In the simplest case, an interface token consists of the link's hardware address. An interface token can be combined with a prefix to form an address.

- o Small sites consisting of a set of machines attached to a single link should not require the presence of a stateful server or router as a prerequisite for communicating. Plug-and-play communication is achieved through the use of link-local addresses. Link-local addresses have a well-known prefix that identifies the (single) shared link to which a set of nodes attach. A host forms a link-local address by concatenating the link-local prefix with its interface token.

- o A large site with multiple networks and routers should not require the presence of a stateful address configuration server. To enable hosts to generate site-local or global addresses, Router Advertisements, which are generated by routers, include options that list the set of active prefixes on a link.

- o Address configuration should facilitate the graceful renumbering of a site's machines. For example, a site may wish to renumber all of its nodes when it switches to a new network service provider. Renumbering is achieved through the leasing of addresses to interfaces and the assignment of multiple addresses to the same interface. Lease lifetimes provide the mechanism through which a site phases out old prefixes. The assignment of multiple addresses to an interface provides for a transition period during which both a new address and the one being phased out work simultaneously.

- o System administrators need the ability to specify whether stateless autoconfiguration, stateful autoconfiguration, or both should be used. Router Advertisements include flags specifying which mechanisms a host should use.

4. PROTOCOL OVERVIEW

This section describes the typical steps that take place when an interface autoconfigures itself. Autoconfiguration of a link-local address normally takes place when an interface is (re)initialized, e.g. at startup. Autoconfiguration of global and site-local addresses and other parameters is done periodically, starting as soon as possible after an interface has been initialised or enabled for autoconfiguration.

A node starts the autoconfiguration process by generating a link-local address for the interface. Before the address can be used, however, the node attempts to verify that the "tentative" address is not already in use by another node on the link. The node sends out a Neighbor Solicitation message containing the tentative address as the target. If another node is already using that address, it will return a Neighbor Advertisement saying so. If another node is also attempting to use the same address, it will send a Neighbor Solicitation for the target as well. If a node determines that its tentative link-local address is not unique, autoconfiguration stops and manual configuration of the machine is required.

Once a node ascertains that the tentative address is unique, it assigns it to the interface. At this point, the node has IP-level connectivity with neighboring nodes via its link-local address.

To generate site-local or global addresses, a host listens for Router Advertisements. To obtain an advertisement quickly, a host sends one or more Router Solicitations to the all-routers multicast group. If no Router Advertisement is received, the host assumes that stateful address autoconfiguration is desired and invokes an appropriate protocol.

Router Advertisements contain two flags indicating what type of stateful autoconfiguration (if any) should be performed. A "managed address configuration" flag indicates whether hosts should use stateful autoconfiguration to obtain addresses. An "other configuration" flag indicates whether hosts should use stateful autoconfiguration to obtain additional information (excluding addresses).

Router Advertisements also contain zero or more Prefix Information options that indicate what type of stateless address autoconfiguration should be done. It should be noted that the stateless and stateful address autoconfiguration fields in Router Advertisements are processed independently of one another, and a host may use both stateful and stateless address autoconfiguration simultaneously. One Prefix Information option field, the "autonomous address-configuration flag", indicates whether or not the option even applies to stateless autoconfiguration. If it does, additional option fields contain a subnet prefix together with lifetime values indicating how long addresses created from the prefix remain preferred and valid.

Routers advertise Router Advertisements periodically. Hosts process the information contained in each advertisement as described above.

For safety, all addresses obtained through autoconfiguration should be tested for uniqueness. In the case of addresses created through stateless autoconfig, however, the uniqueness of an address is determined primarily by the portion of the address formed from an interface token. Thus, if a node has already verified the uniqueness of a link-local address, additional addresses created from the same interface token need not be tested for uniqueness. In contrast, all addresses obtained via stateful address autoconfiguration should be tested for uniqueness individually. To accommodate sites that believe the overhead of performing duplicate address detection outweighs its benefits, the use of duplicate address detection can be disabled through the administrative setting of a per-interface configuration flag.

4.1. Site Renumbering

Address leasing facilitates site renumbering by providing a mechanism to time-out addresses in hosts. At present, the TCP/IP protocol suite provides no support for changing endpoint addresses while a TCP connection is open. If an end-point address changes, existing connections break and all communication to the old address fails. Even when applications use UDP as a transport protocol, addresses must generally remain the same during a packet exchange.

Distinguishing valid addresses into preferred and deprecated categories provides a way of indicating to upper layers that a valid address may

become invalid shortly, and future communication using the address will fail, should the address's deprecation lifetime expire before communication ends. To avoid this scenario, higher layers should use a preferred address (assuming one of sufficient scope exists) to increase the likelihood that an address will remain valid for the duration of the communication. It is up to system administrators to set appropriate prefix lifetimes in order to minimize the impact of failed communication when renumbering takes place. The deprecation period should be long enough that most, if not all, communications are using the new address at the time an address becomes invalid.

The IP layer is expected to provide a means for upper layers (including applications) to select the most appropriate source address given a particular destination and possibly other constraints. An application may choose to select the source address itself before starting a new communication or may leave the address unspecified, in which case the upper networking layers will use the mechanism provided by the IP layer to choose a suitable address on the application's behalf.

Detailed address selection rules are beyond the scope of this document.

5. PROTOCOL SPECIFICATION

Address autoconfiguration is performed on a per-interface basis. For multihomed hosts, address autoconfiguration is performed independently on each interface.

5.1. Host Configuration Variables

A host MUST allow the following variable to be configured for each multicast interface:

```
AutoConfig     If set, the host autoconfigures itself following the
               procedure described in this document.

               Default: TRUE


DuplAddrDetect If set, the node MUST use the duplicate detection
               procedure (Section 5.4) to verify addresses are
               unique before assigning them to an interface.

               Default: TRUE
```

5.2. Autoconfiguration-Related Variables

A host maintains a number of data structures and flags:

ManagedFlag — Copied from the Managed field of the most recently received Router Advertisement message. The flag starts out in a FALSE state.

OtherFlag — Copied from the Other field of the most recently received Router Advertisement message. The flag starts out in a FALSE state.

AddressList — List of addresses together with their associated lifetimes. Addresses on the list can be obtained through stateless or stateful address autoconfiguration, or some other external mechanism. AddressList initially contains no entries.

The values of these variables and flags changes over time as the lifetimes of prefixes (and addresses) expire, new prefixes are learned, etc.

If system management changes an interface's AutoConfig flag from TRUE to FALSE, the value of ManagedFlag and OtherFlag MUST be set to FALSE, with any in-progress autoconfiguration activities interrupted as described below in Section 5.5.3.

5.3. Creation of Link-Local Addresses

A host forms a link-local address whenever an interface is initialized and the AutoConfig flag is TRUE. (Note that the AutoConfig flag may be set independently of interface initialization. If the link-local address has not yet been created when the AutoConfig is changed from FALSE to TRUE, it is created at this time.) An interface is initialized after the following events:

- The interface is initialized at system startup time.

- The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management.

- The interface attaches to a link for the first time.

A link-local address is formed by prepending the well-known link-local prefix E8::0 [ADDR-ARCH] (of appropriate length) to the interface token.

If the interface token has a length of N bits, the interface token replaces the right-most N zero bits of the link-local prefix. If the interface token is more than 118 bits in length, autoconfiguration fails and manual configuration is required.

A link-local address has an infinite preferred and valid lifetime; it is never timed out.

5.4. Verifying The Uniqueness Of An Address

Duplicate address detection is performed on an interface only if the DuplAddrDetect configuration variable is set to TRUE.

Duplicate address detection is applied to an address once after an address is created, but before assigning it to an interface, regardless of whether the address is obtained through stateful, stateless or manual configuration. All addresses SHOULD be tested for uniqueness. However, when stateless address autoconfiguration is used, address uniqueness is determined solely by the interface token, assuming that subnet prefixes are assigned correctly (i.e., if all of an interface's addresses are generated from the same token, either all addresses or none of them will be duplicates). Thus, for a set of addresses formed from the same interface token, it is sufficient to check that one of the addresses is unique on the link. In such cases, one of those addresses MUST be verified before any of the addresses can be assigned to an interface. Normally, the link-local address would be tested, since it is the first address to be formed.

The procedure for detecting duplicate addresses makes use of Neighbor Solicitation and Advertisement messages as described below. If a duplicate address is discovered during the procedure, the interface will need to be manually configured with a new token, or all IP addresses for the interface will need to be manually configured. Note that the method for detecting duplicates is not completely reliable, and it is possible that duplicate addresses will still exist.

An address on which the duplicate address detection procedure is applied is said to be tentative until the procedure has been completed successfully. A tentative address is not considered "assigned to an interface" in the traditional sense. That is, the interface must accept Neighbor Solicitation and Advertisement messages containing the tentative address in the Target Address field, but processes such packets differently from those whose Target Address matches an address assigned to the interface. Other packets addressed to the tentative address should be silently discarded.

It should also be noted that duplicate address detection will nearly

always need to be performed before an address is assigned to an interface to avoid problems that directly result from multiple nodes using the same addresses. If address resolution is done in parallel with duplicate address detection, and the address is subsequently determined to be in use by another node, the node performing duplicate address detection may send packets containing the tentative address that interfere with the proper functioning of the other nodes, especially the one already using the address.

5.4.1. Message Validation

A node MUST silently discard any Neighbor Solicitation or Neighbor Advertisement that does not specify the validity checks as specified in [DISCOVERY]. A solicitation that passes these validity checks is called a valid solicitation or valid advertisement.

5.4.2. Sending Neighbor Solicitation Messages

Before sending a Neighbor Solicitation, an interface MUST join the all-nodes multicast address and the solicited-node multicast address of the tentative address. The former insures that the node receives Neighbor Advertisements from other nodes already using the address; the latter insures that two nodes attempting to use the same address simultaneously detect each other's presence.

To check an address, a node sends a Neighbor Solicitation with a Target Address set to the address being checked. The source of the solicitation is set to the unspecified address and the destination is set to the solicited-node multicast address of the target address.

If the Neighbor Solicitation is the first message to be sent from an interface after interface (re)initialization, the node should delay the message by a random amount of time between 0 and MAX_RTR_SOLICITATION_DELAY as specified in [DISCOVERY]. This serves to alleviate congestion when many nodes start up on the link at the same time, such as after a power failure, and may help to avoid race conditions when more than one node is trying to solicit for the same address at the same time.

There should be a way for a node to determine whether a sending interface loops back packets sent to a multicast address. Otherwise it will not be possible for a node to determine whether a solicitation received on an interface is from itself or from another node with a duplicate address. This issue is discussed in more detail below.

5.4.3. Receiving Neighbor Solicitation and Advertisement Messages

On receipt of a valid Neighbor Solicitation message on an interface, node behavior depends on whether the target address is tentative or not. If the target address is not tentative, the solicitation is processed in the normal way [DISCOVERY]. If the target address is tentative, processing takes place as follows. There are two cases to consider.

If the source address of the solicitation is not the unspecified address, a node is performing address resolution on the address. The node receiving the solicitation should silently discard the message and MUST NOT return a response. Responding to address resolution requests for a tentative address risks polluting the Neighbor Caches of other nodes should the address already be in use by another node.

If the source address of the Neighbor Solicitation is the unspecified address, the solicitation is from a node performing duplicate address detection. There are two cases to consider. First, the solicitation may have been sent by the receiving node (e.g., the packet was looped back). Alternatively, another node (with the same hardware address and/or interface token) is also attempting to use the address. In the first case, the solicitation should be ignored. In the second case, the tentative address is a duplicate and should not be used (by either node).

Determining whether a multicast solicitation was looped back to the sender or actually came from another node is implementation-dependent. If two interfaces happen to have the same hardware link address, one cannot distinguish the two cases by comparing the packet contents. Instead, the implementation must have a good understanding of the interface's multicast loopback semantics. In particular:

- If a Neighbor Solicitation for a tentative address is received prior to having sent a Neighbor Solicitation, the tentative address is a duplicate.

- If a Neighbor Solicitation has been sent, and an identical one is received, the tentative address is a duplicate if the interface does not loopback multicast packets.

- In all cases, if more Neighbor Solicitation for the tentative address are received than have been sent, the tentative address is a duplicate.

If a Neighbor Advertisement containing the tentative address is received while performing duplicate address detection, the node MUST disable that interface and log a system management error. If no such advertisement is received within the time specified, the address is no longer

considered to be tentative and can be assigned to the interface.

If a duplicate address is detected, the node does not respond to the solicitation. Instead, it disables the interface and logs a system management error.

5.5. Creation of Global- and Site-Local Addresses

5.5.1. Sending Router Solicitations

Router Advertisements are sent periodically to the all-nodes multicast address. To obtain an advertisement quickly, a host sends out Router Solicitations as described in [DISCOVERY].

5.5.2. Absence of Router Advertisements

If a link has no routers, a host MUST use stateful autoconfiguration to obtain addresses and other configuration information. From the perspective of autoconfiguration, a link has no routers if no Router Advertisements are being received. Router Advertisements can be absent in two scenarios:

- From the time autoconfiguration was last initiated, no Router Advertisements have been received at all, after having sent Router Solicitations as described in [DISCOVERY].

- At least one Router Advertisement was received, but enough time has elapsed since receipt of the last advertisement that a new one should have been received. Autoconfiguration does not attempt to detect this situation.

When a host determines that no routers are present on a link, it sets the value of ManagedFlag and OtherFlag to TRUE, initiating stateful autoconfiguration as described in Section 5.5.3 (if necessary). If a router subsequently begins sending Router Advertisements, the rules in Section 5.5.3 insure that hosts process them in the proper way.

5.5.3. Router Advertisement Processing

Autoconfiguration silently ignores Router Advertisement messages received on interfaces in which the AutoConfig flag is set to FALSE.

On receipt of a valid Router Advertisement (as defined in [DISCOVERY]), a host copies the value of the advertisement's Managed bit into

ManagedFlag. If the value of ManagedFlag changes from FALSE to TRUE, the host should invoke the stateful address autoconfiguration protocol. If the value of the ManagedFlag changes from TRUE to FALSE, any activity related to stateful address autoconfiguration should be halted. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful address configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

An advertisement's Other bit is processed in an analogous manner. A host copies the value of the Other bit into OtherFlag. If the value of OtherFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol. If the value of the OtherFlag changes from TRUE to FALSE, any activity related to stateful autoconfiguration for parameters other than addresses should be halted. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

For each Prefix-Information option in the Router Advertisement:

a) If the Autonomous flag is not set, silently ignore the Prefix.

b) If the prefix is the link-local prefix, silently ignore the Prefix Information Option.

c) If the preferred lifetime is greater than the valid lifetime, silently ignore the Prefix Information Option. A node MAY wish to log a system management error in this case.

d) If the prefix advertised matches the prefix of an autoconfigured address already in the list, then set the preferred timer to that of the option's preferred lifetime, and set the valid lifetime to that of the option's valid lifetime.

e) If the prefix advertised does not match the prefix of an address already in the list, then form an address by appending the interface token to the prefix as follows:

```
|            128 - N bits              |       N bits          |
+--------------------------------------+-----------------------+
|            link prefix               |   interface token     |
+--------------------------------------------------------------+
```

If the sum of the prefix length and interface token length does not

equal 128 bits, the Prefix Information option MUST be ignored. An implementation MAY wish to log a system management error in this case. It is the responsibility of the system administrator to insure that the lengths of prefixes contained in Router Advertisements are consistent with the length of interface tokens for that link type.

In those cases where a site requires the use of longer prefixes than can be accommodated by the interface token, stateful autoconfiguration can be used.

If an address is formed successfully, the host adds it to AddressList, initializing its preferred and valid lifetime values from the Prefix Information option.

5.5.4. Address Lifetime Expiry

A preferred address becomes deprecated when its preferred lifetime expires. A deprecated address SHOULD continue to be used as a source address in existing communications, but SHOULD NOT be used in new communications if a current (non-deprecated) address is available and it has sufficient scope. The IP layer MUST continue to accept datagrams destined to a deprecated address since a deprecated address is still a valid address for the interface.

An address becomes invalid when its valid lifetime expires. An invalid address MUST NOT be used as a source address in outgoing communications and MUST NOT be recognized as a valid destination address for the interface in incoming communications.

Note that if a Prefix Information option is received with a preferred lifetime of zero, the address with that prefix is immediately deprecated. Similarly, if the advertised valid lifetime is zero, the address with that prefix immediately becomes invalid.

5.6. Configuration Consistency

It is possible for hosts to obtain address information using both stateless and stateful protocols since both may be enabled at the same time. It is also possible that the values of other configuration parameters such as MTU size and hop limit are advertised both by a router[DISCOVERY] and the stateful protocol. If the same configuration information is provided using multiple sources, then the value of this information should be consistent. However, it is not an error if the information is detected to be inconsistent: hosts accept the union of all information received using the stateless and stateful protocols. If

different sources configure the same information, then the parameters are updated with the most recently advertised values.

6. OPEN ISSUES/TODO

o Is duplicate address detection strong enough (we only send one NS). Constants OK?

o is configurability of DuplAddrDetect good enough? Note that:

- One can't assume that all nodes are on the net at any one time, so performing DAD just once or twice does not guarantee that there won't be collisions later.

- Turning DuplAddrDetect on/off is difficult in practice. It is a per-host (interface) flag, which means it must be turned off in each machine. If this is don't by setting a kernel flag and then having everyone boot the same kernel, DAD will be turned off for all nodes, not just a few.

- it might be nice to turn DuplAddrDetect on/off via RAs, but that means nodes will delay creating link-local addresses until they've received an RA or concluded that no routers are present. This is likely to delay the process longer than performing DAD. (Ouch.)

- perhaps allow RSs to be sent out with unspecified source address, in order to solicited RAs with at "do/don't perform DAD"?

o Possible Neighbor Discovery Changes

-Should we allow RSs to be sent out with unspecified source address to allow DAD and the RSs to be sent in parallel, rather than sequentially. This would reduce the impact of DAD delay.

Need to specify that a Router Solicitation is sent out when AutoConfig flag changes from FALSE to TRUE.

o what is the correct language to use in talking about an "MAC address" used as an interface token. Should we use "hardware address"? "MAC" address? Something else?

o ensure use of "node" vs. "host" is right; autoconfig really applies

to only hosts, but duplicate address detection wants to be more general. Also, link-local address can apply to all nodes, not only hosts.

7. SECURITY CONSIDERATIONS

To be completed.

8. REFERENCES

[IPv6-ETHER]
M. Crawford. "A Method for the Transmission of IPv6 Packets over Ethernet Networks", Internet Draft.

[ADDR-ARCH]
R. Hinden and S. Deering, "Internet Protocol Version (IPv6) Addressing Architecture", Internet Draft, May 1995, draft-ietf-ipngwg-addr-arch-03.txt

[DISCOVERY]
T. Narten, E. Nordmark and W. A. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", Internet Draft, September 1995, <draft-ietf-ipngwg-discovery-02.txt>

Acknowledgements

The authors would like to thank the members of both the IPNG and ADDRCONF working groups for their input. In particular, thanks to Jim Bound, Steve Deering, and Erik Nordmark.

AUTHORS' ADDRESSES

Susan Thomson
Bellcore
445 **South Street**
Morristown, NJ 07960
USA

phone: +1 201-829-4514
email: set@thumper.bellcore.com

Thomas Narten
IBM Corporation
P.O. Box 12195
Research Triangle Park, NC 27709-2195
USA

phone: +1 919 254 7798
email: narten@vnet.ibm.com