Network Working Group
Internet Draft
Expiration Date: November 1996

Robert Elz
University of Melbourne

May 1996

Identifying IPv6 Interfaces in Link-Local Addresses

draft-ietf-ipngwg-iid-02.txt

Status of this Memo

1. Abstract

This draft proposes a change to the way that IPv6 link local addresses are constructed, so that a node can guarantee that all of its link local addresses are unique within the node. The current definition of a link local address, a well known prefix, some number of zero bits, and a link specific unique token, ensures that it will be unique on the link, which is what is required for communications using those addresses over the link, but does not require uniqueness of the addresses within a node. In some cases all of a nodes interfaces may share the same link local address. Even the possibility of this means that link local addresses, which may in some situations be the only addresses that exist, cannot be used for internal definition of interfaces, or other purposes within the node. This draft suggests a method by which nodes may overcome this problem, by redefining the bits between the prefix and the token to be available to be used by the node to cause the addresses to be unique.

2. Introduction

IPv6 created link local addresses, by which all interfaces could always be numbered, regardless of any other addressing which may, or may not, be available. These addresses are only suitable for communicating within that link, and are only unique on the link. [Addrconf] defines link local addresses, the various IPv6 over foo specs define mechanisms for generating link local addresses such that they are highly likely to be unique, and [addrconf] defines methods for detecting most cases in which this procedure has failed to generate unique link local addresses.

However, nothing, so far, has defined any method for ensuring that link local addresses are unique within a node, nor for that matter, for justifying that any such uniqueness is useful. This draft attempts to achieve both of those purposes.

It is intended that this draft serve the purpose of encouraging debate and discussion on this issue, and then perhaps cause some modifications to published RFCs, or other drafts. It is not intended that this draft ever, itself, be more widely published. Because of this, several terms (eg: "addrconf") are not defined here, and assumed to be understood by the reader. Examination of IPv6 RFCs and drafts should provide explanations.

No apology is made for failing to misspell "neighbour" in this draft.

3. Identifying Interfaces

It is usual for a node with more than one interface to need, from time to time, a mechanism to identify a particular interface amongst the interfaces available. Currently, with IPv4, this has been done on an ad hoc basis, as IPv4 addresses could not be used. Not all interfaces necessarily posses an IPv4 address.

However, with IPv6, all (IPv6) interfaces will have a link local address. This address is intended to allow communications over the attached links and so is defined to be usable only on that link.

With a minor modification, the link local address could also serve the purpose of identifying interfaces within a node for IPv6. This relies upon all interfaces having a link local address, however this is already specified by [addrconf]. It also relies on the link local addresses being unique within the node, which is a property they do not currently have.

If this is adopted, implementations could, if they wanted, use link local addresses as their standard method for interface identification

for IPv6, eliminating the various methods used for IPv4, often not very satisfactory.

Many IPv4 implementations have used the concatenation of a variable length interface-type string, with a number indicating which interface of that type this is (eg: "le1", or "fta0"). This has multiple drawbacks, for coding it means the need to deal with a different kind of label than the addresses that are used to refer to interfaces on all other nodes, there is usually no stable number which can be expected to remain invariant over hardware reconfigurations (installing a new interface may change the number assigned to one that was otherwise untouched), and there are usually no tools available for mapping more user friendly (or stable) names to these identifiers.

Without this change, using link local addresses to identify interfaces is not possible, unless an implementation can guarantee, by some other means, that the tokens used by all interfaces will differ, always.

4. The method.

A link local address is created by taking a well known prefix (FE80::/10) [addrspec] and appending a link dependent unique token in the low order bits [addrconf]. The precise method, and means of generating the unique token is specified in the various "IP over foo" specifications for links of type "foo" [IPv6/Ether, IPv6/FDDI, ...].

For the purposes of this draft, the current [addrspec] link local address shall be considered to be comprised of three fields, the prefix, the intermediate-zeroes, and the token.

This draft proposes inserting a new field between the token and the prefix. That is, the intermediate-zeroes will be split into two fields, which we shall call the interface identifier, and the discretionary bits.

The interface identifier will be node defined with the sole purpose of disambiguating interfaces within a node if the token required on several links happens to be the same.

The discretionary bits can be used by the node for any purpose, and are no longer required to be zero.

4.1. Placement Option 1

The new interface identifier field will be placed in towards the left of the link local address. This is to guarantee that it can be in the same positions for all link types, regardless of the length of the token to be appended. This guarantees that if the interface identifier is unique within the node for all interfaces, then the generated link local addresses will also be unique within the node for all interfaces, regardless of the values of the discretionary bits or token. It also achieves maximal benefit for the "::" notational convention by keeping as many zeroes as possible in contiguous positions, though implementations are permitted to place any value they desire in the discretionary bits. In this option, the discretionary bits will be in lesser significant bit positions than the interface identifier.

It is suggested that the low order 16 bits of the high order 32 bits of the link local address be allocated to interface identifier. This allows another 6 bits between the current defined prefix, and the new field, which will be reserved for future use, potentially to define a different format for link local addresses at some future date. Using high order bits also has ramifications (or more precisely non-ramifications) with respect to multicast address selection for neighbour discovery, which will be expanded upon below. The various "IPv6 over foo" specifications will be altered to show this field.

4.2. Placement Option 2

The new interface identifier field will be placed in towards the right of the link local address. This places it just next to the token, which is assumed to have a defined maximum length. The interface identifier field will be at a fixed bit position regardless of token length, to the left of the longest possible token. This guarantees that if the interface identifier is unique within the node for all interfaces, then the generated link local addresses will also be unique within the node for all interfaces, regardless of the values of the discretionary bits or token. It also achieves maximal benefit for the "::" notational convention by keeping as many zeroes as possible in contiguous positions, though implementations are permitted to place any value they desire in the discretionary bits. In this option, the discretionary bits will be in more significant bit positions than the interface identifier.

It is suggested that the high order 16 bits of the low order 64 bits of the link local address be allocated to interface identifier. This assumes that the longest permitted token is 48 bits. These bits are beyond the range of bits used for multicast address selection for neighbour discovery, which will be expanded upon below. The various

"IPv6 over foo" specifications will be altered to show this field.

4.3. The New Fields

The "addrconf" specification will define the contents of the interface identifier field. This will specify that a node may insert any value in this field that it desires, but that it is intended that the field be used to cause all link local addresses assigned to the node to be unique. It will be recommended that nodes use the field to hold some interface hardware-specific value (or software-specific, for virtual interfaces) which is likely to remain constant over time, even if similar interfaces are added or removed. Thus, this field alone will be unique amongst all interfaces to the node, and so is sufficient to identify interfaces, regardless of whether or not the token varies from one interface to another.

The discretionary bits may be used by the implementation for any purpose.

5. Duplicate address detection

[Addrconf] specifies that any address generated by a node must be tested for uniqueness by being tested by the Duplicate Address Detection (DAD) algorithm.

For link local addresses as originally defined, this amounts to a test for uniqueness of the link specific token on the link, as all other bits in the address are the same for all nodes.

[Addrconf] uses this feature to permit DAD to be avoided for additional addresses created from the same token on the same link, when created in a standard manner - such as that specified for stateless autoconfiguration in [addrconf]. Such addresses are formed the same way in all nodes on the link, with the token inserted - known uniqueness of the token guarantees uniqueness within the same scope of the other generated address. Uniqueness in wider scopes is derived from the known properties of the prefix to which the token is appended.

As modified here, performing DAD on the link local address does not any longer amount to any uniqueness guarantee of the token, as two link local addresses (from different nodes) may have the same token, yet differ in interface identifier, or discretionary bits.

To avoid the extra burden of testing all autonomously configured addresses, this drafts specifies than when testing a link local address for uniqueness using DAD, the address tested shall be formed with the interface identifier and discretionary bits, that is, the

intermediate-zeroes, set to zero. This is the same address that was previously tested.

Whenever a node that implements this modification receives a Neighbour Solicitation packet containing a target address with prefix FE80::/10 it should consider only those 10 bits, and the final token bits, for which the size will be determined by the nature of the link over which the NS is received, when comparing the target address requested, and its own link local address for the link, all other bits of both the target address sought, and the local link local address should be considered to be zero.

Nodes that have been implemented according to previous versions of [Addrconf] and [Discovery], or which simply do not wish to make use of the interface identifier or discretionary bits, will have zeroes in all the intermediate-zeroes bits of their own link local addresses. When processing a ND packet sent for DAD purposes, simple comparison of the addresses will work, as that ND packet will also contain zeros in the intermediate-zeroes bits. Other ND packets may contain other values in those bits, but this node will only be concerned with those containing its own link local address, and that will be as this node defined it.

The Neighbour Advertisement returned in response to receiving a Neighbour Solicitation containing a target address with prefix FE80::/10, and sent from the unspecified address, that is, a DAD packet, should contain the responding node's link local address, modified to contain only zeroes in the intermediate-zeroes bits. A node that has not been modified to implement this specification will do this naturally, as the intermediate-zeroes bits in its link local address will all be zero anyway. Nodes implementing these modifications must check received NS packets for their source address being unspecified, which they must do anyway to reply correctly, and if so, and if a reply is to be sent, ensure that the intermediate-zeroes bits are cleared from the link local address.

When receiving Neighbour Advertisement packets during the DAD process, and the target address therein has a prefix of FE80::/10, a node implementing this specification must consider only the prefix, and token when comparing the returned target address and the tentative link local address for the node. The intermediate-zeroes bits must, for this purpose, and only this purpose, be treated as if they were set to zero the tentative link local address. Those bits will always be zero in the NA reply received.

This procedure then returns DAD of the link local address to being a uniqueness test of the token on the link, which then allows the token to be used to generate other unique addresses without testing those -

including the link local address as defined here.

Note that as defined here, an implementation that chooses not to make use of the interface identifier or discretionary bits fields, and always uses them as intermediate-zeroes as currently defined, need not alter its implementation from that pertaining to the previous definition of link local addresses. Received NS packets for DAD will contain zeroes in all relevant bits, and thus will appear as if this specification did not exist. The returned NA from the node will contain the interface link local address, just as now, which will be the address with the intermediate-zeroes field. Ordinary NS and NA packets directed to and from the node, will contain the node's link local address, as always, whatever bits it contains, as would any other packets to or from the node using its link local address. Nodes must thus never assume that link local addresses will always contain zeroes in the intermediate-zeroes bits. Nodes should not ever be tempted to make assumptions about the values of any addresses of any other nodes.

[Addrconf] and [Discovery] will be updated, as specified below, to include these procedures.

6. Multicast Address Generation

[Discovery] specifies the algorithm by which the solicited-node multicast address is generated. That uses only the low bits of the IPv6 address. By positioning the interface identifier in the upper bits of the link local address, the same solicited-node multicast address will be generated, whatever interface identifier is chosen. On some media, the token might be less than 32 bits wide. In such cases the node may choose to use some of the bits used for multicast address generation for other purposes. This draft strongly recommends against this practice, but does not prohibit it. However, nodes must be prepared to receive Neighbour Solicitation packets sent to either the node's link local address, or to the address formed from the prefix FE80::/16 and the interface token, with zeroes between (in the intermediate-zeroes field). This may mean accepting two different multicast addresses where one would ordinarily be sufficient.

7. Rationale

By guaranteeing that all interfaces have unique addresses within the node, the node can use those unique addresses to identify the interface, rather than having to invent some new name space for this purpose. Using addresses also allows all the standard tools to be used for interface identification, that are used for host identification. Interfaces can be named by Domain Name System names, which can be manipulated in the same way as other DNS names, and translated by standard routines into the addresses used as interface identifiers.

Because a link local address must exist for all interfaces [addrconf] and must always be generated in a standard way, the address is effectively available for internal uses instantly the interface is made known to the rest of the system - even before DAD is performed. This means that no other interface identification is required, the unique link local address can be, if the implementation desires, the only interface identification provided. If DAD fails, [addrconf] specifies that the interface be shut down. Even then the link local address will still be unique within the node, and can still be used to refer to the inactive interface.

With all interfaces identified by unique intra-node addresses, implementations can, if they desire, make use of address fields in the regular API for interface identification. For example, an application might make use of source routing via a local interface in order to direct packets to be transmitted over a specific link - which would not imply that a routing header would be present in the packet transmitted if the local address was all it contained, or that that address would ever be present in a transmitted routing header. Directing packets through specific interfaces, especially if unnumbered, has not been easy, or even possible, in many current IP implementations.

Implementations with multiple tokens available to create link local addresses have the opportunity to connect multiple interfaces to one link if that will produce a useful configuration - perhaps to achieve higher bandwidth through a switch that permits multiple parallel conversations. An implementation with only a single token would need to invent a token, which is certainly not to be recommended, survive with a shared link local address, or modify some other part of the link local address, presumably using the technique described here. With unique link local addresses, stateful autoconfiguration could be used to obtain other addressing, where stateless autoconfiguration would generate the same addresses for all such interfaces. Without that, there would be no unique token for DHCP to use to assign different addresses,

Where the node concerned is participating in routing protocols, such as OSPF for IPv6, it also seems that having multiple interfaces on one link sharing the same link local address might be a problem. (Precise details unavailable...)

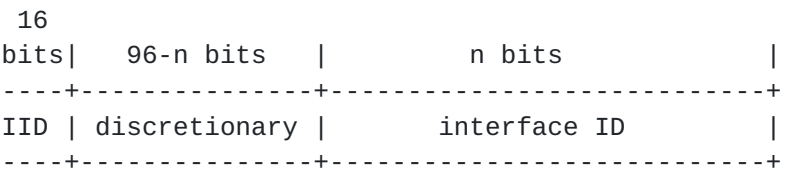
8. Specific Document Changes Suggested

In order to implement this change, the documents listed in this section must be modified as indicated, or in such a way as to lead to the same effect. Some of the documents concerned are, at the date of this memo, available only as Internet Drafts. Because the changes below relate to a specific version of each document, the normal practice of never referencing an Internet Draft other than as "work in progress" will be suspended here, and specific drafts will be referenced.

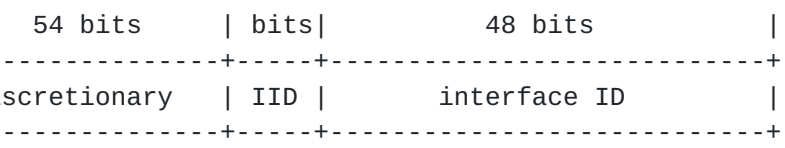
Note that in these changes, the reference annotations have been changed so as to be compatible with those used in this document, rather than those used in the original. This is for clarity here only, clearly the references in each case should be in the format required by the specific document.

8.1. Changes to RFC1884

In section 2.4.8, on page 11, the diagram showing the format of a link local address must be changed to either:



or:



Which is chosen will depend upon which placement of the interface identifier (IID) is decided to be better. See section 4.

No other changes are required.

8.2. Changes to draft-ietf-addrconf-ipv6-auto-07.txt

8.2.1. Changes to Terminology

In section 2, somewhere on pages 5 to 8, the following definitions should be added:

interface identifier

- An integer identifier for each interface attached to a node that is unique within the node, and relatively stable. That is, the identifier should not normally change across resets of the IPv6 layer, nor reboots of the node, and should generally be unchanged by the addition of deletion of other interfaces.

discretionary bits

- A value that the implementation or node can set to any value for any purpose when defining the enclosing structure or value. Once set, the bits must not be altered as long as the particular structure remains defined, or the value required.

8.2.2. Changes to link local address formation

In section 5.3, in the paragraph beginning "A link-local address is formed ..." on page 15, will be replaced by the following paragraphs.

> A link local address is formed from five component parts. The most significant bits are always the well known prefix FE80::/10 [Addrspec]. The least significant bits are the interface token, chosen in an interface specific way. Between those are placed fields containing zeroes (padding), an interface identifier field, and a field containing any value that the implementation desires to place there.
>
> The interface identifier field is a 16 bit field that can contain either zeroes, or a value chosed by the implementation that is unique among all interfaces connected to the node. Such a value should be chosen in a way that long term stability of the value chosen is likely. That is, neither a random number, nor a simple count of interfaces as they are configured is appropriate. Suitable choices would relate to the particular hardware configuration, such as the slot number used on the bus, or the address in I/O space of the interface adaptor, or similar. It is highly desirable that the interface identifier not alter merely because some other interface is added or removed.

Following those paragraphs, one of the following paragraphs will be added, depending on which choice is made for the positioning of the interface identifier field.

Either:

> The fields of the link local address will be formed in the order, from most to least significant bits, the well known prefix (FE80::/10) first, then a six bit zero field, then the 16 bit interface identifier field, then an M bit discretionary field, followed by an N bit interface token. M shall be 96 - N. If the interface token is more than 96 bits in length, autoconfiguration fails, and manual configuration is required.

Or:

> The fields of the link local address will be formed in the order, from most to least significant bits, the well known prefix (FE80::/10) first, then 54 discretionary bits, then the 16 bit interface identifier field, followed by an optional variable width field of zeroes, followed by an N bit interface token. If the interface token is less than 48 bits, the zero filled field is included, and is 48 - N bits wide. If the interface token is exactly 48 bits, no field of zeroes is included. If the interface token is more than 48 bits in length, autoconfiguration fails, and manual configuration is required.

8.2.3. Changes to Duplicate Address Detection

In section 5.4.2, on page 16, the paragraph which begins "To check an address, a node sends ..." shall be replaced by:

> To check an address, a node sends DupAddrDetectTransmits Neighbor Solicitations, each separated by RetransTimer milliseconds. If the tentative address is not a link local address, the solicitation's Target Address is set to the address being checked. If the tentative address is a link local address, the solicitation's Target Address is set to the address being checked, modified such that only the prefix and interface token fields are non-zero, all other bits of the link local tentative address must be zero. The IP source is set to the unspecified address and the IP destination is set to the solicited-node multicast address of the Target Address.

In section 5.4.3, on page 17, the following paragraph should precede the paragraphs that currently exist:

> While performing Duplicate Address Detection for a link local address, the node must consider a received Neighbor Solicitation message sent with a Target Address that is also a link local address, and which has the same interface token value as the node's tentative link local address, to be a Neighbor Solicitation sent to that tentative link local address.

In section 5.4.4 on page 18, the following paragraph should precede the paragraph that currently exists:

> While performing Duplicate Address Detection for a link local address, the node must consider a received Neighbor Advertisement message sent with a Target Address that is also a link local address, and which has the same interface token value as the node's tentative link local address, to be a Neighbor Advertisement sent to that tentative link local address.

8.3. Changes to draft-ietf-ipngwg-discovery-06.txt

In section 4.3, on page 24, the Target Address field of the Neighbour Solicitation packet is changed to be:

> The IP address of the target of the solicitation. It MUST NOT be a multicast address. If Duplicate Addres Detection is in progress [Addrconf] and the address being tested ia a link local address, then only the prefix (FE80::/10) and the interface token are non-zero, all other bits MUST be zero.

In section 4.4, on page 26, the paragraph describing the Target Address field of a Neighbour Advertisment shall be augmented as follows.

> Note that when responding to a Neighbour Solicitation sent from the unspecified address (Source Address of the Neighbour Solicitation) seeking a link local address, the Target Address MUST be the link local address modified by including only the prefix (FE80::/10) and interface token fields. All other bits MUST be zero. This should be equivalent to the address in the Target Address field of the Neighbour Solicitation packet, but need not be identical to the node's link local address on the interface.

In section 7.2.4, on page 61, the paragraph that begins "If the source of the solicitation is the unspecified address, ..." will be altered to be:

> If the source of the solicitation is the unspecified address, the node MUST set the Solicited flag to zero and multicast the

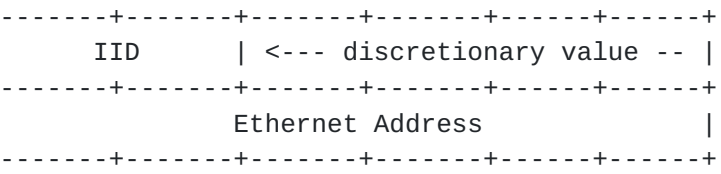
advertisement to the all-nodes address. Further, if the Target Address is a link local address, then only the prefix and interface token fields may have non-zero values, all other bits of the Target Address MUST be zero. Otherwise, the node MUST set the Solicited flag to one and unicast the advertisement to the Source Address of the solicitation.

8.4. Changes to draft-ietf-ipngwg-ethernet-ntwrks-02.txt

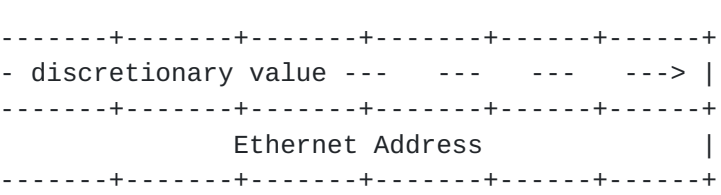
On page 2, in the section entitled "Stateless Autoconfiguration and Link-Local Addresses", the paragraph which begins: "The IPv6 Link-local address ..." will be changed to be:

> The IPv6 Link-local address [Addrspec] for an Ethernet interface is formed by appending the interface's IEEE 802 address to the 80-bit prefix created by appending the interface's host specific interface identifier [Addrconf] and other host or implementation defined bits to the well known ten bit prefix FE80::/10.

The diagram that immediately follows will be replaced by either:



or:



Which is chosen will depend upon which placement of the interface identifier (IID) is decided to be better.

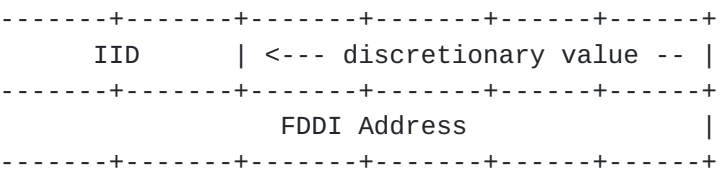
8.5. Changes to draft-ietf-ipngwg-fddi-ntwrks-03.txt

On page 4, in the section entitled "Stateless Autoconfiguration and Link-Local Addresses", the paragraph which begins: "The IPv6 Link-local address ..." will be changed to be:

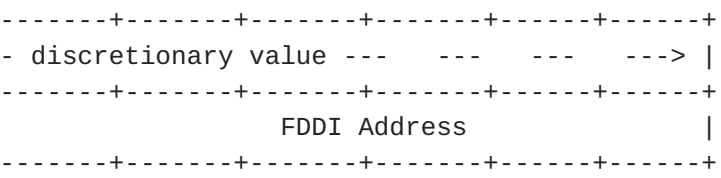
> The IPv6 Link-local address [Addrspec] for an FDDI interface is formed by appending the interface's IEEE 802 address to the 80-bit prefix created by appending the interface's host specific interface identifier [Addrconf] and other host or implementation

defined bits to the well known ten bit prefix FE80::/10.

The diagram that immediately follows will be replaced by either:



or:



Which is chosen will depend upon which placement of the interface identifier (IID) is decided to be better.

8.6. Changes to draft-ietf-ipngwg-pppext-ipv6cp-02.txt

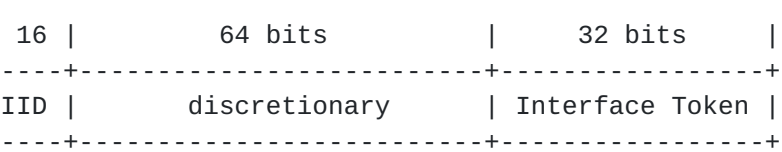
In section 5, the paragraph which begins "The most significant 10 bits of the address" on page 10 will be replaced by either:

> The most significant 10 bits of the address is the Link-Local prefix FE80::. This is followed by six more bits of zero, then the 16 interface identifier bits, then 64 discretionary bits, followed by the Interface Token field.

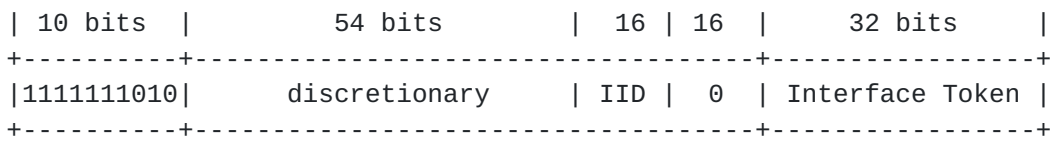
or:

> The most significant 10 bits of the address is the Link-Local prefix FE80::. This is followed by 54 discretionary bits, then the 16 interface identifier bits, then 16 bits of zero, followed by the Interface Token field.

The choice of replacement text will depend upon which placement of the interface identifier field is considered best. The immediately preceding diagram in the draft will be replaced by either:



or:



Which is chosen will depend upon which placement of the interface identifier (IID) is decided to be better. Note that in the second option, the token field remains 48 bits wide though the PPP token is defined as only being 32 bits. The remaining 16 bits are zero padded. This ensures that the interface identifier field will remain in the same bit positions in all link local addresses.

8.7. Changes to other link specific documents

Other documents specifying IPv6 implementation details on other link types will be changed in ways similar to those indicated above.

9. Security Considerations

Addressing and security are unrelated concepts. Attempts to pretend otherwise are misguided.

10. References

[IPv6] "Internet Protocol, Version 6 (IPv6) Specification", S. Deering, R. Hinden, RFC1883, January 4, 1996.

[Addrspec] "IP Version 6 Addressing Architecture", R. Hinden, S. Deering, RFC1884, January 4, 1996.

[IPv6/Ether] "A Method for the Transmission of IPv6 Packets over Ethernet Networks" Matt Crawford, Work In Progress (soon to be an RFC).

[IPv6/FDDI] "A Method for the Transmission of IPv6 Packets over FDDI Networks" Matt Crawford, Work In Progress (soon to be an RFC).

[Discovery] "Neighbor Discovery for IP Version 6 (IPv6)" Thomas Narten, Erik Nordmark, W A Simpson, Work In Progress (soon to be an RFC).

[Addrconf] "IPv6 Stateless Address Autoconfiguration" Susan Thomson, Thomas Narten, Work In Progress (soon to be an RFC).

[DHCPv6] "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)"
J. Bound, C. Perkins,
Work In Progress (soon to be an RFC).

11. Acknowledgements

My thanks to Matt Crawford for assisting getting this draft into a semi-presentable state, which is not to imply that he agrees with the proposal. Also to Dennis Ferguson for pointing out additional justification for use of a method like this. The IPNGWG working group also devoted some valuable meeting time to discussion of this issue, all who contributed there also contributed here.

12. Author's Address

Robert Elz
University of Melbourne
Parkville
Victoria 2052
Australia

EMail: kre@munnari.OZ.AU