IPv6 Working Group
INTERNET-DRAFT
May 5, 2005
Expires November 2005

D. Thaler
M. Talwar
Microsoft
C. Patel
All Play, No Work

Bridge-like Neighbor Discovery Proxies (ND Proxy)
<draft-ietf-ipv6-ndproxy-02.txt>

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (C) The Internet Society (2005). All Rights Reserved.

Abstract

Bridging multiple links into a single entity has several operational advantages. A single subnet prefix is sufficient to support multiple physical links. There is no need to allocate subnet numbers to the different networks, simplifying management. Bridging some types of media requires network-layer support, however. This document describes these cases and specifies the IP-layer support that enables bridging under these circumstances.

The behavior of one common type of IPv4 ARP Proxy deployed today is documented herein for informational purposes, but this document concentrates on describing similar behavior for IPv6.

1. Introduction

In the IPv4 Internet today, it is common for Network Address Translators (NATs) [NAT] to be used to easily connect one or more leaf links to an existing network without requiring any coordination with the network service provider. Since NATs modify IP addresses in packets, they are problematic for many IP applications. As a result, it is desirable to address the problem (for both IPv4 and IPv6) without the need for NATs.

Another common solution is IEEE 802 bridging, as specified in [BRIDGE]. It is expected that whenever possible links will be bridged at the link layer using classic bridge technology [BRIDGE] as opposed to using the mechanisms herein. However, classic bridging at the data-link layer has the following limitations (among others):

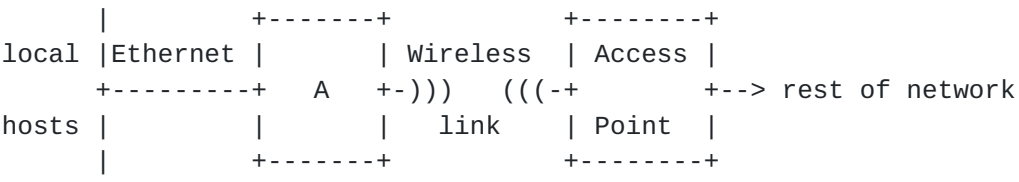
- It requires the ports to support promiscuous mode.

- It requires all ports to support the same type of link-layer addressing (in particular, IEEE 802 addressing).

As a result, two common scenarios, described below, are not solved, and it is these two scenarios we specifically target in this document. While the mechanism described herein may apply to other scenarios as well, we will concentrate our discussion on these two scenarios.

1.1. SCENARIO 1: Wireless upstream

The following figure illustrates a likely example:



In this scenario, the access point has assigned an IPv4 and/or an IPv6 subnet prefix to the wireless link, and uses link-layer encryption so that wireless clients may not see each other's data.

Classic bridging requires the bridge (node A in the above diagram) to be in promiscuous mode. In this wireless scenario, A cannot put its wireless interface into promiscuous mode, since one wireless node cannot see traffic to/from other wireless nodes. This document describes a solution for both IPv4 and IPv6 which does not involve NAT or require any change to the access point or router.

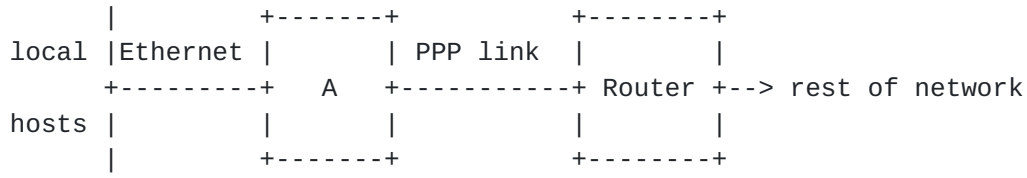
Multiple variants of IPv4 ARP proxying have been used for some years to solve this problem. ARP-based bridges were first described in [ARPPROXY], but that variant decrements the TTL, does not forward all-ones broadcasts, and requires proxies to keep per-packet state on recent subnet broadcasts. The first two characteristics can cause problems with applications and protocols which assume that nodes in the subnet prefix can be reached with TTL 1 (or with TTL 255, with TTL 255 verified on receipt) and/or with a subnet broadcast. The third characteristic results in scalability issues in proxy implementations. As a result, multiple variants have emerged in different implementations over time. In this document, we describe one such variant, and enable equivalent functionality for IPv6 to remove this incentive to deploy NATs in IPv6.

We also note that Prefix Delegation [PD] could also be used to solve this scenario. There are, however, two disadvantages to this. First, if an implementation already supports IPv4 ARP proxying (which is indeed supported in a number of implementations today), then IPv6 Prefix Delegation would result in separate IPv6 subnets on either side of the device, while a single IPv4 subnet would span both segments. This topological discrepancy can complicate applications and protocols which use the concept of a

local subnet. Secondly, the extent to which Prefix Delegation is supported, and supported without additional charge, is up to the service provider. Hence, there is no guarantee that Prefix Delegation will work without explicit configuration or additional charge. Bridging, on the other hand, allows the device to work with zero configuration, regardless of the service provider's policies, just as a NAT does. Hence bridging avoids the incentive to NAT IPv6 just to avoid paying for, or requiring configuration to get, another prefix.

1.2. SCENARIO 2: PPP upstream

The following figure illustrates another likely example:



In this scenario, the router believes that the other end of the PPP link (node A) has a single IPv4 address, as negotiated by PPP. For IPv6, it has assigned a /64 to the link and advertises it in an IPv6 Router Advertisement.

Classic bridging does not support non-802 media, and hence IPv4 connectivity is solved by making the proxy (node A in the above diagram) be a NAT. This document does not specify any other IPv4 solution for this scenario. However, this document does specify a solution for IPv6 which does not involve NAT or require any change to the router.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [KEYWORDS].

The term "proxy interface" will be used to refer to an interface (which could itself be a bridge interface) over which network layer proxying is done as defined herein.

In this document we make no distinction between a "link" (in the classic IPv6 sense) and a "subnet". We use the term "segment" to

apply to a bridged component of the link.

Finally, while it is possible that functionality equivalent to that described herein may be achieved by nodes which do not fulfill all the requirements in [NODEREQ], in the remainder of this document we will describe behavior in terms of an IPv6 node as defined in that document.

3. Requirements

Bridge-like proxy behavior is designed with the following requirements in mind:

- Support connecting multiple segments with a single subnet prefix.
- Support media which cannot be bridged at the link-layer. Note, this document does not support bridging of non-802 media for IPv4.
- Support both IPv6 and IPv4 for 802 media.
- Do not require any changes to existing routers. That is, any routers on the subnet should be unaware that the subnet is being bridged.
- Provide full connectivity between all nodes in the subnet. For example, if there are existing nodes (such as any routers on the subnet) which have addresses in the subnet prefix, adding a bridge-like proxy must allow bridged nodes to have full connectivity with existing nodes on the subnet.
- Prevent loops.
- Also work in the absence of any routers.
- Support nodes moving between segments. For example, a node should be able to keep its address without seeing its address as a duplicate due to any cache maintained at the proxy.
- Allow dynamic addition of a proxy without adversely disrupting the network.

- The proxy behavior should not break any existing classic bridges in use on a network segment.

3.1. Non-requirements

The following items are not considered requirements, as they are not met by classic bridges:

- Show up as a hop in a traceroute.
- Use the shortest path between two nodes on different segments.
- Be able to use all available interfaces simultaneously. Instead, bridging technology relies on disabling redundant interfaces to prevent loops.
- Support connecting media on which Neighbor Discovery is not possible. For example, some technologies such as [6TO4] use an algorithmic mapping from IPv6 address to the underlying link-layer (IPv4 in this case) address, and hence cannot support bridging arbitrary IP addresses.

The following additional items are not considered requirements for this document:

- Support network-layer protocols other than IPv4 and IPv6. We do not preclude such support, but it is not specified in this document.
- Support Neighbor Discovery, Router Discovery, or DHCPv4 packets using IPsec. We also note that the current methods for securing these protocols do not use IPsec so this is considered acceptable.
- Support Redirects for off-subnet destinations that point to a router on a different segment from the redirected host. While this scenario may be desirable, no solution is currently known which does not have undesirable side effects outside the subnet. As a result, this scenario is outside the scope of this document.

4. Bridge-Like Proxy Behavior

Network layer support for proxying between multiple interfaces SHOULD be used only when classic bridging is not possible.

When a proxy interface comes up, the node puts it in "all-multicast" mode so that it will receive all multicast packets. It is common for interfaces to not support full promiscuous mode (e.g., on a wireless client), but all-multicast mode is generally still supported.

As with all other interfaces, IPv4 and IPv6 maintain a neighbor cache (aka "ARP cache") for each proxy interface, which will be used as described below. For readability, we will describe the neighbor cache as if both IPv4 and IPv6 neighbors use the same state machine described in [ND].

4.1. Forwarding Packets

When a packet from any IP source address other than the unspecified address is received on a proxy interface, the neighbor cache of that interface SHOULD be consulted to find an entry for the source IP address. If no entry exists, one is created in the STALE state.

When any IP or ARP packet is received on a proxy interface, it must be parsed to see whether it is known to be of a type that negotiates link-layer addresses. This document covers the following types: ARP, IPv6 Neighbor Discovery, IPv6 Router Discovery, IPv6 Redirects, and DHCPv4. These packets are ones that can carry link-layer addresses, and hence must be proxied (as described below) so that packets between nodes on different segments can be received by the proxy and have the correct link-layer address type on each segment.

When any other IP broadcast or multicast packet (other than to the IPv6 Link-scoped RSTP Multicast Group) is received on a proxy interface, in addition to any normal IP behavior such as being delivered locally, it is forwarded unchanged (other than using a new link-layer header) out all other proxy interfaces on the same link. (As specified in [BRIDGE], the proxy may instead support multicast learning and filtering but this is OPTIONAL.) In particular, the IPv4 TTL or IPv6 Hop Limit is not updated, and no ICMP errors (except as noted in Section 4.1.1 below) are sent as a

result of attempting this forwarding.

When any other IP unicast packet is received on a proxy interface, if it is not locally destined then it is forwarded unchanged (other than using a new link-layer header) to the proxy interface for which the next hop address appears in the neighbor cache. Again the IPv4 TTL or IPv6 Hop Limit is not updated, and no ICMP errors (except as noted in Section 4.1.1 below) are sent as a result of attempting this forwarding. To choose a proxy interface to forward to, the neighbor cache is consulted, and the interface with the neighbor entry in the "best" state is used. In order of least to most preferred, the states (per [ND]) are INCOMPLETE, STALE, DELAY, PROBE, REACHABLE. A packet is never forwarded back out the same interface on which it arrived; such a packet is instead silently dropped.

If no cache entry exists (as may happen if the proxy has previously evicted the cache entry or if the proxy is restarted), the proxy SHOULD queue the packet and initiate Neighbor Discovery as if the packet were being locally generated. The proxy MAY instead silently drop the packet. In this case, the entry will eventually be recreated when the sender re-attempts neighbor discovery.

The link layer header, and the link-layer address within the payload for each forwarded packet will be modified as follows:

1) The source address will be the address of the outgoing interface.

2) The destination address will be the address in the neighbor entry corresponding to the destination IP address.

3) The link-layer address within the payload is substituted with the address of the outgoing interface.

4.1.1. Sending Packet Too Big Messages

Whenever any IPv4 packet is to be forwarded out an interface whose MTU is smaller than the size of the packet, and the Dont Fragment bit is set, the ARP proxy drops the packet and sends a Fragmentation Needed message back to the source.

Similarly, whenever any IPv6 packet is to be forwarded out an

interface whose MTU is smaller than the size of the packet, the ND proxy drops the packet and sends a Packet Too Big message back to the source, as described in [ICMPv6].

4.1.2. Proxying Packets With Link-Layer Addresses

Once it is determined that the packet is either multicast/broadcast or else is not locally destined (if unicast), the special types enumerated above (ARP, etc.) that carry link-layer addresses are handled by generating a proxy packet that contains the proxy's link-layer address on the outgoing interface instead. Section 7, "Guidelines to proxy developers", describes the scenarios in which the link-layer address substitution in the payload should be performed.

As with all forwarded packets, the link-layer header is also new. Note that any change to the length of a proxied packet, such as when the link-layer address length changes, will require corresponding changes to fields in the IP header, namely the IPv4 Total Length and Header Checksum fields, or the IPv6 Payload Length field.

4.1.3. IPv4 ARP Proxying

When any IPv4 or ARP packet is received on a proxy interface, it must be parsed to see whether it is known to be one of the following types: ARP, or DHCPv4.

4.1.3.1. ARP REQUEST Packets

If the received packet is an ARP REQUEST, the request is processed locally but no ARP REPLY is generated immediately. Instead, the ARP REQUEST is proxied (as described above) and the ARP REPLY will be proxied when it is received. This ensures that the proxy does not interfere with hosts moving from one segment to another since it never responds to an ARP REQUEST based on its own cache.

4.1.3.2. ARP REPLY Packets

If the received packet is an ARP REPLY, the neighbor cache on the receiving interface is first updated as if the ARP REPLY were

locally destined, and then the ARP REPLY is proxied as described above.

4.1.3.3. DHCPv4 Packets

If the received packet is a DHCPv4 DISCOVER or REQUEST message, then instead of changing the client's hardware address in the payload, the BROADCAST (B) flag is set in the proxied packet. This ensures that the proxy will be able to receive and proxy the response since the response will be broadcast rather than unicast to that hardware address. The hardware address is thus used only as a unique identifier and hence need not be a link-layer address on the same segment.

One limitation of this rule is that if the authentication protocol for DHCPv4 described in [DHCPAUTH] is used, only clients that set the BROADCAST flag themselves will be able to use DHCPv4 through the proxy. If [DHCPAUTH] is not used, a DHCPv4 client might still detect, with previously undefined behavior, that the broadcast bit has been changed from the setting in the message originally set by the client. However, the point of this rule is not to solve this problem, but rather to document existing practice.

4.1.4. IPv6 ND Proxying

When any IPv6 packet is received on a proxy interface, it must be parsed to see whether it is known to be one of the following types: IPv6 Neighbor Discovery, IPv6 Router Discovery, or IPv6 Redirects.

4.1.4.1. ICMPv6 Neighbor Solicitations

If the received packet is an ICMPv6 Neighbor Solicitation, the NS is processed locally as described in section 7.2.3 of [ND] but no NA is generated immediately. Instead the NS is proxied as described above and the NA will be proxied when it is received. This ensures that the proxy does not interfere with hosts moving from one segment to another since it never responds to an NS based on its own cache.

4.1.4.2. ICMPv6 Neighbor Advertisements

If the received packet is an ICMPv6 Neighbor Advertisement, the neighbor cache on the receiving interface is first updated as if the NA were locally destined, and then the NA is proxied as described above.

4.1.4.3. ICMPv6 Router Advertisements

Unless RSTP is implemented as described in section 6, the following special processing is done for IPv6 Router Advertisements (RAs).

A new "Proxy" bit is defined in the existing Router Advertisement flags field as follows:

```
+-+-+-+-+-+-+-+-+
|M|O|H|Prf|P|Rsv|
+-+-+-+-+-+-+-+-+
```

where "P" indicates the location of the Proxy bit, and "Rsv indicates the remaining reserved bits.

The proxy determines an "upstream" proxy interface, typically through a physical choice dictated by the scenario (see Scenarios 1 **and 2 above), or through manual configuration.**

When an RA with the P bit clear arrives on the upstream interface, the P bit is set when the RA is proxied out all other ("downstream") proxy interfaces (see secion 6).

If an RA with the P bit set has arrived on a given interface (including the upstream interface) within the last 60 minutes, that interface MUST NOT be used as a proxy interface; i.e., proxy functionality is disabled on that interface.

Furthermore, if any RA (regardless of the value of the P bit) has arrived on a "downstream" proxy interface within the last 60 minutes, that interface MUST NOT be used as a proxy interface.

If, on the other hand, RSTP is implemented as described in section 6, no special processing is done for RAs. That is, the P bit is ignored, and the P bit is unmodified in RAs proxied to other interfaces.

4.1.4.4. ICMPv6 Redirects

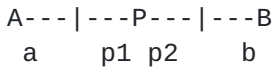
If the received packet is an ICMPv6 Redirect message, then the proxied packet should be modified as follows. If the proxy has a valid (i.e., not INCOMPLETE) neighbor entry for the target address on the same interface as the redirected host, then the TLLA option in the proxied Redirect simply contains the link-layer address of the target as found in the proxy's neighbor entry, since the redirected host may reach the target address directly. Otherwise, if the proxy has a valid neighbor entry for the target address on some other interface, then the TLLA option in the proxied packet contains the link-layer address of the proxy on the sending interface, since the redirected host must reach the target address through the proxy. Otherwise, the proxy has no valid neighbor entry for the target address, and the proxied packet contains no TLLA option, which will cause the redirected host to perform neighbor discovery for the target address.

4.2. Originating Packets

Locally originated packets that are sent on a proxy interface also follow the same rules as packets received on a proxy interface. If no neighbor entry exists when a unicast packet is to be locally originated, an interface can be chosen in any implementation-specific fashion. Once the neighbor is resolved, the actual interface will be discovered and the packet will be sent on that interface. When a multicast or broadcast packet is to be locally originated, an interface can be chosen in any implementation-specific fashion, and the packet will then be forwarded out other proxy interfaces on the same link as described in Section 4.1 above.

5. Example

Consider the following topology, where A and B are nodes on separate segments which are connected by a bridge-like proxy P:



A and B have link-layer addresses a and b, respectively. P has link-layer addresses p1 and p2 on the two segments. We now walk through the actions that happen when A attempts to send an initial

IPv6 packet to B.

A first does a route lookup on the destination address B. This matches the on-link subnet prefix, and a destination cache entry is created as well as a neighbor cache entry in the INCOMPLETE state. Before the packet can be sent, A needs to resolve B's link-layer address and sends a Neighbor Solicitation (NS) to the solicited-node multicast address for B. The SLLA option in the solicitation contains A's link-layer address.

P receives the solicitation (since it is receiving all link-layer multicast packets) and processes it as it would any multicast packet by forwarding it out to other segments on the link. However, before actually sending the packet, it determines if the packet being sent is one which requires proxying. Since it is an NS, it creates a neighbor entry for A on interface 1 and records its link-layer address. It also creates a neighbor entry for B (on an arbitrary proxy interface) in the INCOMPLETE state. Since the packet is multicast, P then needs to proxy the NS out all other proxy interfaces on the subnet. Before sending the packet out interface 2, it replaces the link-layer address in the SLLA option with its own link-layer address, p2.

B receives this NS, processing it as usual. Hence it creates a neighbor entry for A mapping it to the link-layer address p2. It responds with a Neighbor Advertisement (NA) sent to A containing B's link-layer address b. The NA is sent using A's neighbor entry, i.e. to the link-layer address p2.

The NA is received by P, which then processes it as it would any unicast packet; i.e., it forwards this out interface 1, based on the neighbor cache. However, before actually sending the packet out, it inspects it to determine if the packet being sent is one which requires proxying. Since it is an NA, it updates its neighbor entry for B to be REACHABLE and records the link-layer address b. P then replaces the link-layer address in the TLLA option with its own link-layer address on the outgoing interface, p1. The packet is then sent out interface 1.

A receives this NA, processing it as usual. Hence it creates a neighbor entry for B on interface 2 in the REACHABLE state and records the link-layer address p1.

6. Loop Prevention

An implementation MUST ensure that loops are prevented, via either:

a) by using the P bit in RA's as described below, or

b) by running the Rapid Spanning Tree Protocol (RSTP) defined in [BRIDGE] on all proxy interfaces as described below.

Loop prevention using RSTP would typically be done by devices that already implement this protocol as a result of supporting normal briding functionality, and is done here as follows. IEEE 802 interfaces use the protocol exactly as specified in [BRIDGE]. Operation of RSTP over other types of link layers is done by encapsulating the RSTP frame in an IPv6 header. The Next Header field is set to [TBA by IANA], indicating that an RSTP header follows. The Destination Address field is set to the Link-scoped RSTP Multicast Group [TBA by IANA]. All proxies operating on non-IEEE 802 media join this group so they will receive RSTP packets. RSTP packets are never forwarded or proxied.

Loop avoidance using the P bit in RAs is done as follows. The proxy determines an "upstream" proxy interface, typically through a physical choice dictated by the scenario (see Scenarios 1 and 2 above), or through manual configuration. As described in Section 4.1.3.3, only the upstream interface is allowed to receive RAs, and never from other proxies. Proxy functionality is disabled on an interface otherwise. Finally, a proxy MUST wait until it has sent two P bit RAs on a given "downstream" interface before it enables forwarding on that interface.

Note that it is possible that a site has a mixture of P bit proxies and RSTP-capable proxies. To understand how this works, consider the case where only P bit proxies are present. However, within each link, there may be classic IEEE 802 bridges. Those classic bridges are responsible for ensuring that that link is loop-free. To P bit proxies, proxies which run RSTP are the same in this respect as classic IEEE 802 bridges.

7. Guidelines to proxy developers

Proxy developers will have to accomodate protocols or protocol options (for example, new ICMP messages) that are developed in the

future, or protocols that are not mentioned in this document (for example, proprietary protocols). This section prescribes guidelines that can be used by proxy developers to accomodate protocols that are not mentioned herein.

1) If a link-layer address carried in the payload of the protocol can be used in the link-layer header of future messages, then the proxy should substitute it with its own address. For example the link-layer address in NA messages is used in the link-layer header for future messages, and, hence, the proxy substitutes it with its own address.

   For broadcast/multicast packets, the link-layer address substituted within the payload will be different for each outgoing interface.

2) If the link-layer address in the payload of the protocol will never be used in any link-layer header, then the proxy should not substitute it with its own address. No special actions are required for supporting these protocols. For example, [DHCPv6] is in this category.

8. IANA Considerations

To support loop prevention over non-802 media, IANA should assign:

1) a Protocol Number for RSTP, and

2) an IPv6 Link-Local Scope multicast address for All-RSTP-Speakers.

9. Security Considerations

Proxies are susceptible to the same kind of security issues that plague hosts using unsecured Neighbor Discovery or ARP. Even if these protocols are secured, the proxies may process unsecured messages, and update the neighbor cache. Malicious nodes within the subnet can take advantage of this property, and hijack traffic. The threats are discussed in detail in [PSREQ].

As a result, securing Neighbor Discovery or ARP must take into account the ability to proxy messages. This document does not

introduce any new requirements in this regard.

From an IPv6 perspective, RFC 2461 [ND] already defines the ability to proxy Neighbor Advertisements. Since the ND packets must be modified whenever the link-layer address formats are different (as with PPP) or promiscuous reception is not possible (as with 802.11), securing any solution in this space requires that hosts have a secure relationship with the proxy.

It is reasonable for nodes on the leaf subnet to have a secure relationship with the proxy, and accept ND packets from either the owner of a specific address (normal SEND), or which it can verify are from a trusted proxy (see below).

For nodes on the external subnet, there is a tradeoff between security (where all nodes have a secure relationship with the proxy) and privacy (where no nodes are aware that the proxy is a proxy). In the case of a point-to-point external link (Scenario 2) however, SEND may not be a requirement on that link.

Verifying that ND packets come from a trusted proxy requires an extension to the SEND protocol and is left for future work, but is similar to the problem of securing Router Advertisements which is supported today.

10. Appendix A: Comparison with Naive RA Proxy

It has been suggested that a simple Router Advertisement (RA) proxy would be sufficient, where the subnet prefix in an RA is "stolen" by the proxy and applied to a downstream link instead of an upstream link. Other ND messages are not proxied.

There are many problems with this approach. First, it requires cooperation from all nodes on the upstream link. No node (including the router sending the RA) can have an address in the subnet or it will not have connectivity with nodes on the downstream link. This is because when a node on a downstream link tries to do Neighbor Discovery, and the proxy does not send the NS on the upstream link, it will never discover the neighbor on the upstream link. Similarly, if messages are not proxied during DAD, conflicts can occur.

Second, if the proxy assumes that no nodes on the upstream link have addresses in the prefix, such a proxy could not be safely

deployed without cooperation from the network administrator since it introduces a requirement that the router itself not have an address in the prefix. This rules out use in situations where bridges and Network Address Translators (NATs) are used today, which is the problem this document is directly addressing. Instead, where a prefix is desired for use on one or more downstream links in cooperation with the network administrator, Prefix Delegation [PD] should be used instead.

11. Authors' Addresses

Dave Thaler
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
Phone: +1 425 703 8835
EMail: dthaler@microsoft.com

Mohit Talwar
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
Phone: +1 425 705 3131
EMail: mohitt@microsoft.com

Chirayu Patel
All Play, No Work
Bangalore, Karnataka 560038
Phone: +91-98452-88078
EMail: chirayu@chirayu.org

12. Normative References

[ARP]
D. Plummer, "An Ethernet Address Resolution Protocol", STD 37, RFC 826, November 1982.

[ARPPROXY]
J. Postel, "Multi-LAN address resolution", RFC 925, October 1984.

[BRIDGE]
T. Jeffree, editor, "Media Access Control (MAC) Bridges", ANSI/IEEE Std 802.1D, 2004, http://standards.ieee.org/getieee802/download/802.1D-2004.pdf.

[DHCPv4]
R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[ICMPv6]
Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

[KEYWORDS]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March, 1997.

[ND] Narten, T., Nordmark, E. and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[NODEREQ]
J. Loughney, "IPv6 Node Requirements", Work in progress, draft-ietf-ipv6-node-requirements-11.txt, August 2004.

13. Informative References

[6TO4]
Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[DHCPAUTH]
Droms, R. and W. Arbaugh, Eds., "Autentication for DHCP Messages", RFC 3118, June 2001.

[DHCPv6]
Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C.

and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[NAT]
Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[PD] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

[PSREQ]
Nikander, P., Kempf, J. and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

14. Full Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

15. Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights

in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.