Network Working Group
Internet-Draft
Expires: November 30, 2003

S. Roy
A. Durand
J. Paugh
Sun Microsystems, Inc.
June 2003

Dual Stack IPv6 on by Default
draft-ietf-v6ops-v6onbydefault-00.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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.

This Internet-Draft will expire on November 30, 2003.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This document discusses problems that can occur when dual stack nodes that have IPv6 enabled by default are deployed in IPv4 or mixed IPv4 and IPv6 environments. The problems include application connection delays, poor connectivity, and network security. Its purpose is to raise awareness of these problems so that they can be fixed or worked around.

Table of Contents

1. Introduction

This document specifically addresses operating system implementations that implement the dual stack IPv6 model, and would ship with IPv6 enabled by default. It addresses the case where such a system is installed and placed in an IPv4 only or mixed IPv4 and IPv6 environment, and documents potential problems that users on such systems could experience if the IPv6 connectivity is non-existent or sub-optimal.

It begins in Section 2 by examining problems within IPv6 implementations that defeat the destination address selection mechanism defined in [ADDRSEL] and contribute to poor IPv6 connectivity. Starting with Section 3 it then examines other issues that network software engineers and network and systems administrators should be aware of when deploying dual stack systems with IPv6 enabled.

2. No IPv6 Router

Consider a scenario in which a dual stack system has IPv6 enabled and placed on a link with no IPv6 routers. The system is using IPv6 Stateless Address Autoconfiguration [AUTOCONF], so it only has a link-local IPv6 address configured. It also has a single IPv4 address that happens to be a private address as defined in [PRIVADDR].

An application on this system is trying to communicate with a destination whose name resolves to public and global IPv4 and IPv6 addresses. The application uses an address resolution API that implements the destination address selection mechanism described in Default Address Selection for IPv6 [ADDRSEL]. The application will attempt to connect to each address returned in order until one succeeds. Since the system has no off-link IPv6 routes, the optimal scenario would be if the IPv4 addresses returned were ordered before the IPv6 addresses. The following sections describe where things can go wrong with this scenario.

2.1 Problems with Default Address Selection for IPv6

The Default Address Selection for IPv6 [ADDRSEL] destination address selection mechanism could save the application a few useless connection attempts by placing the IPv4 addresses in front of the IPv6 addresses. This would be desired since all IPv6 destinations in this scenario are unreachable (there's no route to them), and the system's only IPv6 source address is inadequate to communicate with

off-link destinations even if it did have an off-link route.

Let's examine how the destination address selection mechanism behaves in the face of this scenario when given one IPv4 destination and one IPv6 destination.

The first rule, "Avoid unusable destinations", could prefer the IPv4 destination over the IPv6 destination, but only if the IPv6 destination is determined to be unreachable. The unreachability determination for a destination as it pertains to this rule is an implementation detail. One implementable method is to do a simple forwarding table lookup on the destination, and to deem the destination as reachable if the lookup succeeds. The Neighbor Discovery on-link assumption mentioned in Section 2.2 makes this method somewhat irrelevant, however, as an implementation of the assumption could simply be to insert an IPv6 default on-link route into the system's forwarding table when the default router list is empty. The side-effect is that the rule would always determine that all IPv6 destinations are reachable. Therefore, this rule will not necessarily prefer one destination over the other.

The second rule, "Prefer matching scope", could prefer the IPv4 destination over the IPv6 destination, but only if the IPv4 destination's scope matches the scope of the system's IPv4 source address. Since [ADDRSEL] considers private addresses (as defined in [PRIVADDR]) of site-local scope, then this rule will not prefer either destination over the other. The link-local IPv6 source doesn't match the global IPv6 destination, and the site-local IPv4 source doesn't match the global IPv4 destination. The tie-breaking rule in this case is rule 6, "Prefer higher precedence". Since IPv6 destinations are of higher precedence than IPv4 destinations in the default policy table, the IPv6 destination will be preferred.

The solution in this case could be to add a new rule after rule 2 (rule 2.5) that avoids non-link-local IPv6 destinations whose source addresses are link-local. Of course, if the host is manually assigned a global IPv6 source address, then rule 2 will automatically prefer the IPv6 destination, and there is no fix other than to make sure rule 1 considers IPv6 destinations unreachable in this scenario.

Fixing the destination address selection mechanism by adding such a rule is only a mitigating factor if applications use standard name

resolution API's that implement this mechanism, and these applications try addresses in the order returned. This may not be an acceptable assumption in some cases, as there are applications that use hard coded addresses and address search orders (DNS resolver is one example), and/or literal addresses passed in from the user. Such applications will obviously be subject to whatever connection delays are associated with attempting a connection to an unreachable destination. This is discussed in more detail in the next few sections.

2.2 Neighbor Discovery's On-Link Assumption Considered Harmful

Let's assume that the application described in Section 2 is attempting a connection to an IPv6 address first, either because the destination address selection mechanism described in Section 2.1 returned the addresses in that order, or because the application isn't trying the addresses in the order returned. Regardless, the user expects that the application will quickly connect to the destination. It is therefore important that the system quickly determine that the IPv6 destination is unreachable so that the application can try the IPv4 destination.

Neighbor Discovery's [ND] conceptual sending algorithm dictates that when sending a packet to a destination, if a host's default router list is empty, then the host assumes that the destination is on-link.

For an IPv6 enabled host deployed on a network that has no IPv6

routers as is the case in this scenario, the result is that link-layer address resolution must be performed on all IPv6 addresses to which the host sends packets. The Application will not receive acknowledgment of the unreachability of destinations that are not on-link until at least address resolution has failed, which is no less than three seconds (MAX_MULTICAST_SOLICIT * RETRANS_TIMER), plus any transport layer connection timeouts which could be minutes in the case of TCP. The delay with respect to TCP is discussed later in Section 2.3.

On a network that has no IPv6 routing and no IPv6 neighbors, making the assumption that every IPv6 destination is on-link will be a costly and incorrect assumption. If an application has a list of addresses associated with a destination and the first 15 are IPv6

addresses, then the application won't be able to successfully send a packet to the destination until the attempts to resolve each IPv6 address have failed (45 seconds), which could be compounded by any transport timeouts associated with each connection attempt.

If IPv6 hosts don't assume that destinations are on-link as described above, then communication with destinations that are not on-link and unreachable will immediately fail. The IPv6 implementation should be able to immediately notify applications that it has no route to such IPv6 destinations, and applications won't waste time waiting for address resolution to fail.

If hosts need to communicate with on-link destinations, then then they need to be explicitly configured to have on-link routes for those destinations.

2.2.1 Other Problems with the On-Link Assumption

The on-link assumption is problematic in ways not directly related to the scenario described, but they should still be briefly mentioned here.

One problem is that default routes are not special. The on-link assumption as stated in [ND] would have a host assume that all destinations are on-link when its default router list is empty. Clearly it shouldn't make this assumption if it has an off-link route that covers the destination and that route isn't a default route. The absence of a default route does not mean that destinations are not reachable through more specific routes.

Another problem is that the on-link assumption behavior is undefined on multi-homed hosts. When such a host has no default routers and it is trying to reach a destination to which it has no route, should it try NUD on all interfaces at once? Should it simply realize that the

destination is unreachable? The latter is the simplest solution. Determining that a destination is unreachable when there is no route to that destination is the simple solution for all cases, not just the multi-homed case.

2.3 Transport Protocol Robustness

Making the same set of assumptions as Section 2.2, regardless of how long the network layer takes to determine that the IPv6 destination is unreachable, the delay associated with a connection attempt to an unreachable destination can be compounded by the transport layer. In order for our application to quickly fail from an unreachable address to a potentially reachable one, the transport layer should notify the application by failing a connection attempt, or passing ICMPv6 errors up to the application, etc...

In the case of a socket application attempting a connection via TCP, it would be unreasonable for the connect() function to block even after the system has received notification that the address is unreachable via an ICMPv6 Destination Unreachable message. Therefore, TCP should abort connections in SYN-SENT or SYN-RECEIVED state when it receives an ICMPv6 Destination Unreachable message. This helps the cases mentioned in Section 2 in which a node has no default routers and NUD fails for destinations assumed to be on-link, and when firewalls or other systems that enforce scope boundaries send such ICMPv6 errors as described in Section 3.1 and Section 3.3.

It should be noted that the Requirements for Internet Hosts [HOSTREQS] document, in section 4.2.3.9., states that TCP MUST NOT abort connections when receiving ICMP Destination Unreachable messages that indicate "soft errors", where soft errors are defined as ICMP codes 0 (network unreachable), 1 (host unreachable), and 5 (source route failed), and SHOULD abort connections upon receiving the other codes (which are considered "hard errors"). ICMPv6 didn't exist when that document was written, but one could extrapolate the concept of soft errors to ICMPv6 Type 1 codes 0 (no route to destination) and 3 (address unreachable), and hard errors to the other codes.

When [HOSTREQS] was written, hosts were usually assigned a single address, and applications would mostly only try one address when establishing communication with a destination. Not aborting a connection was a sane thing to do if re-trying a single address was a better alternative over quitting the application altogether. With IPv6, and especially on dual stack systems, destinations are often assigned multiple addresses (at least one IPv4 and one IPv6 address), and applications iterate through destination addresses when attempting connections. Since soft errors conditions are those that

would entice an application to continue iterating to another address, TCP shouldn't make the distinction between ICMPv6 soft errors and hard errors when in SYN-SENT or SYN-RECEIVED state. It should abort a connection in those states when receiving any ICMPv6 Destination Unreachable message. It should make this distinction when a connection is in any other state.

Many TCP implementations already behave this way, but others do not. This should be noted as a best current practice in this case.

3. Other Problematic Scenarios

This section describes problems that could arise for a dual stack system with IPv6 enabled when placed on a network with IPv6 connectivity.

3.1 IPv6 Network of Smaller Scope

A network that has a smaller scope of connectivity for IPv6 as it does for IPv4 could be a problem in some cases. If applications have access to name to address mapping information that is of greater scope than the connectivity to those addresses, there is obvious potential for suboptimal network performance. Hosts will attempt to communicate with IPv6 destinations that are outside the scope of the IPv6 routing, and depending on how the scope boundaries are enforced, applications may not be notified that packets are being dropped at the scope boundary.

If applications aren't immediately notified of the lack of reachability to IPv6 destinations, then they aren't able to efficiently fall back to IPv4. They then have to rely on transport layer timeouts which can be minutes in the case of TCP.

An example of such a network is an enterprise network that has both IPv4 and IPv6 routing within the enterprise and has a firewall configured to allow some IPv4 communication, but no IPv6 communication.

3.1.1 Alleviating the Scope Problem

To allow applications to correctly fall back to IPv4 when IPv6 packets are destined beyond their allowed scope, the devices enforcing the scope boundary should send ICMPv6 Destination Unreachable messages back to senders of such packets. The sender's transport layer should act on these errors as described in Section 2.3.

3.2 Poor IPv6 Network Performance

By default in dual stack nodes, applications will try IPv6 destinations first. If the IPv6 connectivity to those destinations is poor while the IPv4 connectivity is better (i.e., the IPv6 traffic experiences higher latency, lower throughput, or more lost packets

than IPv4 traffic), applications will still communicate over IPv6 at the expense of network performance. There is no information available to applications in this case to advise them to try another destination address.

An example of such a situation is a node which obtains IPv4 connectivity natively through an ISP, but whose IPv6 connectivity is obtained through a configured tunnel whose other endpoint is topologically such that most IPv6 communication is done through triangular routes. Operational experience on the 6bone shows that IPv6 RTT's are poor in such situations.

3.2.1 Dealing with Poor IPv6 Network Performance

There are few options from the end node's perspective. One is to configure each node to prefer IPv4 destinations over IPv6. If hosts implement the Default Address Selection for IPv6 [ADDRSEL] policy table, IPv4 mapped addresses could be assigned higher precedence, resulting in applications trying IPv4 for communication first. This has the negative side-effect that these nodes will almost never use IPv6 unless the only address published in the DNS for a given name is IPv6, presumably because of this phenomenon.

Disabling IPv6 on the end nodes is another solution. The idea would be that enabling IPv6 on dual stack nodes is a manual process that would be done when the administrator knows that IPv6 connectivity is adequate.

3.3 Security

Enabling IPv6 on a host implies that the services on the host may be open to IPv6 communication. If the service itself is insecure and depends on security policy enforced somewhere else on the network (such as from a firewall), then there is potential for new attacks against the service.

A firewall, for example, may not be enforcing the same policy for IPv4 as for IPv6 traffic. One possibility is that the firewall could have more relaxed policy for IPv6, perhaps by letting all IPv6 packets pass through, or by letting all IPv4 protocol 41 packets pass through. In this scenario, the dual stack hosts within the protected

network could be subject to different attacks than for IPv4.

Even if a firewall has a stricter policy or identical policy for IPv6 traffic than for IPv4 (the extreme case being that it drops all IPv6 traffic), IPv6 packets could go through the network untouched if tunneled over a transport layer. This could open the host to direct IPv6 attacks.

A similar problem could exist for VPN software. A VPN could protect all IPv4 packets but drop all others onto the local subnet unprotected. At least one widely used VPN behaves this way. This is problematic on a dual stack host that has IPv6 enabled on its local

network. It establishes its VPN link and attempts to communicate with destinations that resolve to both IPv4 and IPv6 addresses. The destination address selection mechanism prefers the IPv6 destination so the application sends packets to an IPv6 address. The VPN doesn't know about IPv6, so instead of protecting the packets and sending them to the remote end of the VPN, it passes such packets in the clear to the local network.

3.3.1 Mitigating Security Risks

Establishing a security policy that is the same for IPv4 and IPv6 would help mitigate this risk.

There is still a risk that IPv6 packets could be tunneled over a transport layer such as UDP, implicitly bypassing security policy. Some more complex mechanism could be implemented to apply the correct policy to such packets. This could be easy to do if tunnel endpoints are co-located with a firewall, but more difficult if internal nodes do their own IPv6 tunneling.

4. Application Robustness

Enabling IPv6 on a dual stack node is only useful if applications that support IPv6 on that node properly cycle through addresses returned from name lookups and fall back to IPv4 when IPv6 communication fails. Simply cycling through the list of addresses returned from a name lookup when attempting connections works in most cases for most applications, but there are still cases where that's not enough. Applications also need to be aware that the fact that a dual stack destination's IPv6 address is published in the DNS does not necessarily imply that all services on that destination function over IPv6.

One example is an application that resolves a destination name to a list of addresses with the intent to contact multiple services at that destination (i.e., http and IMAP). The application tries to contact the first service via the first IPv6 address in the list and succeeds. The success of the connection results in the application throwing away the list of addresses it was given by the name lookup and remembering this single IPv6 address as the usable address for the destination. The second service it tries, however, is only implemented for IPv4. The application tries to communicate to the

second service using the same IPv6 address, the connection fails, and no fall back to IPv4 occurs because the application is outside of the context of cycling through the list of addresses returned from the name lookup.

Applications should not assume that because a service is reachable at an IPv6 address, that other services at that destination are also reachable via that IPv6 address.

5. Security Considerations

This document raises security concerns in Section 3.3.

Normative References

[ADDRSEL] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

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

Informative References

[AUTOCONF]
Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[HOSTREQS]
Braden, R., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989.

[PRIVADDR]
Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

Authors' Addresses

Sebastien Roy
Sun Microsystems, Inc.
1 Network Drive
UBUR02-212
Burlington, MA 01801

EMail: sebastien.roy@sun.com

Alain Durand
Sun Microsystems, Inc.
17 Network Circle
UMPK17-202
Menlo Park, CA 94025

EMail: alain.durand@sun.com

James Paugh
Sun Microsystems, Inc.
17 Network Circle
UMPK17-202
Menlo Park, CA 94025

EMail: james.paugh@sun.com

Appendix A. Acknowledgments

The authors gratefully acknowledge the contributions of Jim Bound, Mika Liljeberg, Erik Nordmark, Pekka Savola, and Ronald van der Pol.

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than

English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.