IPv6 operations
Internet-Draft
Intended status: Informational
Expires: 7 January 2026

O. Caletka
RIPE NCC

6 July 2025

A recommendation for filtering address records in stub resolvers
draft-caletka-aaaa-filtering-01

Abstract

Since IPv4 and IPv6 addresses are represented by different resource records in the Domain Name System, operating systems capable of running both IPv4 and IPv6 need to execute two queries when resolving a host name. This document discusses the conditions under which a stub resolver can optimize the process by not sending one of the queries if the host is connected to a single-stack network.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://oskar456.github.io/ietf-aaaa-filtering/draft-caletka-aaaa-filtering.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-caletka-aaaa-filtering/.

Discussion of this document takes place on the v6ops Working Group mailing list (mailto:v6ops@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/v6ops/. Subscribe at https://www.ietf.org/mailman/listinfo/v6ops/.

Source for this draft and an issue tracker can be found at https://github.com/oskar456/ietf-aaaa-filtering.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Most operating systems support both the IPv6 and the IPv4 networking stack. When such a host is connected to a dual-stack network, whenever a process requests resolution of a DNS name, two DNS queries need to be issued - one for an A record representing an IPv4 address, and one for a AAAA record representing an IPv6 address. The results of such queries are then merged and ordered based on [RFC6724] or

used as input for the Happy Eyeballs algorithm [RFC8305].

When such a host is connected to a single-stack network, only one DNS query needs to be performed: there is no reason for querying for a AAAA record if the host has no IPv6 connectivity, the same way there is no reason to look for an A record if the host has no IPv4 connectivity. Such an optimization however has to consider any possible means of obtaining connectivity for a particular address family, including but not limited to IPv6 Transition Mechanisms or VPNs.

Please note that Multicast DNS [RFC6762] or similar link-local name resolution protocols are not considered in scope of this document.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Connectivity detection algorithms

Whenever an application asks the stub resolver to resolve a domain name without specifying the address family, the stub resolver follows one of the algorithms specified below:

3.1. Routing table based algorithm

This algorithm assumes that the host has connectivity of particular address family if there is at least one route to a destination that is not in Link-Local address space. If there are only routes for destinations in the Link-Local address space, the host is not able to send any packets towards any destination address that could be possibly obtained from the DNS. Therefore, sending an address query for that particular address family is unnecessary.

For each address family supported by the operating system:

1. Read the routing table of the address family;

2. Remove all the routes towards Link-Local destinations from the routing table, ie.:

    * remove routes towards addresses from Section 2.5.6 of [RFC4291] from the IPv6 routing table

   * remove routes towards addresses from [RFC3927] from the IPv4 routing table

3. If the routing table is not empty, send the corresponding name query to the DNS:

   * AAAA query for IPv6

   * A query for IPv4

3.2. IP address based algorithm

As an alternative to analyzing the routing tables, the stub resolver might choose to determine the connectivity by looking at the addresses configured on all network interfaces. This is similar to an application using the flag AI_ADDRCONFIG when interacting with the stub resolver using getaddrinfo() function [GAI].

For each address family supported by the operating system:

1. Collect all addresses configured on all interfaces

2. Remove all Link-Local addresses from the list, ie.:

   * remove addresses from Section 2.5.6 of [RFC4291] from the list of IPv6 addresses

   * remove addresses from [RFC3927] from the list of IPv4 addresses

3. If the list of addresses is not empty, send the corresponding name query to the DNS:

   * AAAA query for IPv6

   * A query for IPv4

4. Connectivity detection considerations

When detecting the connectivity presence, it is necessary to consider ANY routes towards non Link-Local address space and/or IP addresses on ALL interfaces and not just the default route and/or just the default network interface. The implementations SHOULD NOT try to determine connectivity by hardcoding a particular publicly reachable IP address [CHROME].

Improper detection can cause issues for:

* private networks without reachability to the Internet

* VPN tunnels using different address family than the native address family of the host, providing possibly only a limited subset of routes (split-mode VPN)

5. Filtering DNS results

If the host does not have full connectivity for both address families (there are no default gateways for both IPv4 and IPv6), it is possible that the IP(v6) address obtained from the DNS falls into the address space not covered by a route. This should not be problem for a properly written application, since [RFC6724] requires applications to try connecting to all addresses received from the stub resolver.

However, in order to minimize the impact on poorly designed applications, the stub resolver MAY remove addresses not covered by an entry in the routing table from the list of DNS query results sent to the application.

5.1. Filtering IPv4-mapped addresses

As an extension to the filtering of DNS results, the stub resolver MAY also remove IPv4-mapped IPv6 addresses (Section 2.5.5.2 of [RFC4291]) from the list of DNS query results sent to the application.

IPv4-mapped IPv6 addresses are not valid destination addresses [IANA], therefore they should never appear in AAAA records. Sending IPv4-mapped IPv6 address to the application might cause address family confusion for applications using IPv4 compatibility of IPv6 sockets [RFC3493].

6. Effects of not doing address record filtering

The optimization described above is OPTIONAL. A stub resolver of a dual-stack capable host can always issue both A and AAAA queries to the DNS, merge and order the results and send them to the application even if it has only single-stack connectivity. Sending packets to a destination not covered by an entry in the routing table will be immediately refused, so a properly written application will quickly iterate through the list of addresses and finally select the one using the same address family as the connectivity of the host.

However, it should be noted that such behavior increases load on the DNS system. If such an optimization is removed (for instance by a software update) on a large single-stack network, this might overload parts of the DNS infrastructure, since the number of queries will double.

7. Security Considerations

Reducing the number of queries allows an attacker observing the DNS traffic to figure out which address families the host uses.

Suddendly disabling the optimization can overload parts of the DNS infrastructure due to doubling the number of queries.

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <https://www.rfc-editor.org/rfc/rfc3493>.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <https://www.rfc-editor.org/rfc/rfc3927>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/rfc/rfc4291>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <https://www.rfc-editor.org/rfc/rfc6724>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/rfc/rfc6762>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/rfc/rfc8305>.

9.2. Informative References

[CHROME] "Chrome Host Resolution - IPv6 and connectivity", n.d., <https://chromium.googlesource.com/chromium/src/+/0de3ceea881115dd18e79e1d9ea4e090c655996b/net/dns/README.md#IPv6-and-connectivity>.

[GAI] "freeaddrinfo, getaddrinfo - The Open Group Base Specifications Issue 8, IEEE Std 1003.1-2024", n.d., <https://pubs.opengroup.org/onlinepubs/9799919799/functions/freeaddrinfo.html>.

[IANA] "IANA IPv6 Special-Purpose Address Registry", n.d., <https://www.iana.org/assignments/iana-ipv6-special-registry>.

Acknowledgments

TODO acknowledge.

Author’s Address

Ondナ册j Caletka
RIPE NCC
Email: ondrej@caletka.cz