v6ops
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2011

D. Wing
A. Yourtchenko
Cisco

October 25, 2010

Happy Eyeballs: Trending Towards Success with Dual-Stack Hosts
draft-wing-v6ops-happy-eyeballs-ipv6-01

Abstract

This document describes how a dual-stack client can determine the functioning path to a dual-stack server. This provides a seemless user experience during initial deployment of dual-stack networks and during outages of IPv4 or outages of IPv6.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and 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.

This Internet-Draft will expire on April 28, 2011.

Table of Contents

1. Introduction

In order to use HTTP successfully over IPv6, it is necessary that the user enjoys nearly identical performance as compared to IPv4. A combination of today's applications, IPv6 tunneling and IPv6 service providers, and some of today's content providers all cause the user experience to suffer (Section 3). For IPv6, Google ensures a positive user experience by using a DNS white list of IPv6 service providers who peer directly with Google [whitelist]. However, this is not scalable to all service providers worldwide, nor is it scalable for other content providers to operate their own DNS white list.

Instead, this document suggests a mechanism for applications to quickly determine if IPv6 or IPv4 is the most optimal to connect to a server. The suggestions in this document provide a user experience which is superior to connecting to ordered IP addresses which is helpful during the IPv6/IPv4 transition with dual stack hosts.

Following the procedures in this document, once a certain address family is successful, the application trends towards preferring that address family. Thus, repeated use of the application DOES NOT cause repeated probes over both address families.

While the application recommendations in this document are described in the context of HTTP clients ("web browsers"), but is useful and applicable to other time-sensitive applications.

2. Notational Conventions

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 [RFC2119].

3. Problem Statement

As discussed in more detail in Section 3.1, it is important that the same URI and hostname be used for IPv4 and IPv6. Using separate namespaces causes namespace fragmentation and reduces the ability for users to share URIs and hostnames, and complicates printed material that includes the URI or hostname.

As discussed in more detail in Section 3.2, IPv6 connectivity is sometimes broken entirely or slower than native IPv4 connectivity.

3.1. URIs and hostnames

URIs are often used between users to exchange pointers to content -- such as on Facebook, email, instant messaging, or other systems. Thus, production URIs and production hostnames containing references to IPv4 or IPv6 will only function if the other party is also using an application, OS, and a network that can access the URI or the hostname.

3.2. IPv6

When IPv6 connectivity is impaired, today's IPv6-capable web browsers incur many seconds of delay before falling back to IPv4. This harms the user's experience with IPv6, which will slow the acceptance of IPv6, because IPv6 is frequently disabled in its entirety on the end systems to improve the user experience.

Reasons for such failure include no connection to the IPv6 Internet, broken 6to4 or Teredo tunnels, and broken IPv6 peering.

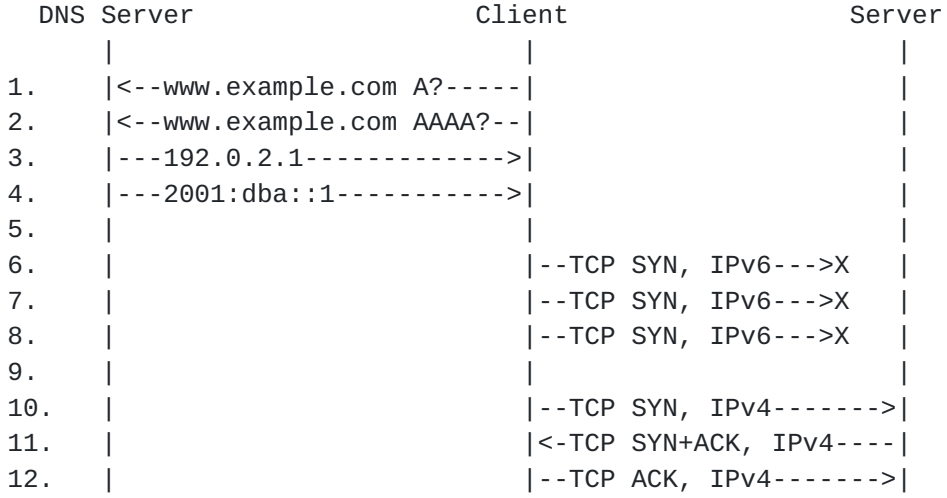

Figure 1: Existing behavior message flow

The client obtains the IPv4 and IPv6 records for the server (1-4). The client attempts to connect using IPv6 to the server, but the IPv6 path is broken (6-8), which consumes several seconds of time. Eventually, the client attempts to connect using IPv4 (10) which succeeds.

4. Client Recommendations

To provide fast connections for users, clients should make connections quickly over various technologies, automatically tune

itself to avoid flooding the network with unnecessary connections (i.e., for technologies that have not made successful connections), and occasionally flush its self-tuning.

4.1. IPv6

If a TCP client supports IPv6 and IPv4 and is connected to IPv4 and IPv6 networks, it can perform the procedures described in this section.

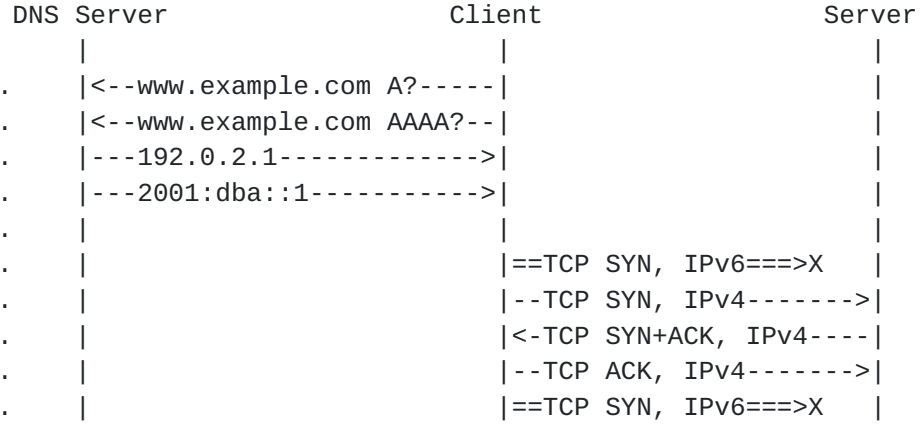

Figure 2: Happy Eyeballs flow 1, IPv6 broken

In diagram above, the client sends two TCP SYNs at the same time over IPv6 (6) and IPv4 (7). In the diagram, the IPv6 path is broken but has little impact to the user because there is no long delay before using IPv4. The IPv6 path is retried until the application gives up (10).

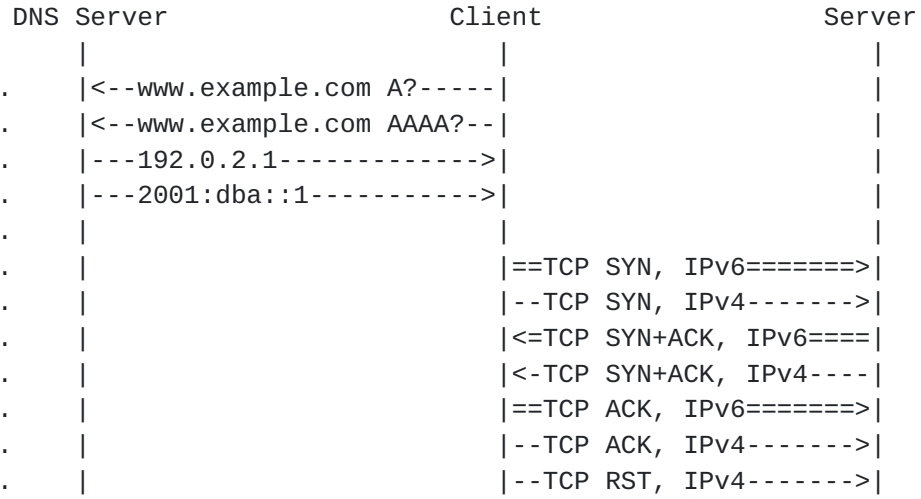

Figure 3: Happy Eyeballs flow 2, IPv6 working

The diagram above shows a case where both IPv6 and IPv4 are working, and IPv4 is abandoned (12).

This section details how to provide robust dual stack service for both IPv6 and IPv4, so that the user perceives very fast application response.

The TCP client application is configured with one value, P. A positive value indicates a preference for IPv6 and a negative value indicates a preference for IPv4. A value of 0 indicates equal weight, which means the A and AAAA queries and associated connection attempts will be sent as quickly as possible. The absolute value of P is the measure of a delay before initiating a connection attempt on the other address family. There are two P values maintained: one is application-wide and the other is specific per each destination (hostname and port).

The algorithm attempts to delay the DNS query until it expects that address family will be necessary; that is, if the preference is towards IPv6, then AAAA will be queried immediately and the A query will be delayed.

The TCP client application starts two threads in order to minimize the user-noticeable delay ("dead time") during the connection attempts:

thread 1: (IPv6)

* If P<0, wait for absolute value of p*10 milliseconds
* send DNS query for AAAA
* wait until DNS response is received
* Attempt to connect over IPv6 using TCP

thread 2: (IPv4)

* if P>0, wait for p*10 milliseconds
* send DNS query for A
* wait until DNS response is received
* Attempt to connect over IPv4 using TCP

The first thread that succeeds returns the completed connection to the parent code and aborts the other thread (Section 4.2.2).

After a connection is successful, we want to adjust the application-wide preference and the per-destination preference. The value of P is incremented (decremented) each time an IPv6 (IPv4) connection is successfully made. When a connection using the less-preferred address family is successful, it indicates the wrong address family was used and the P is halved:

- If P>0 (indicating IPv6 is preferred over IPv4) and the first thread to finish was the IPv6 thread it indicates the IPv6 preference is correct and we need to re-enforce this by increasing the application-wide P value by 1. However, if the first thread to finish was the IPv4 thread it indicates an IPv6 connection problem occurred and we need to aggressively prefer IPv4 more by halving P and rounding towards 0.

- If P<0 (indicating IPv4 is preferred over IPv6) and the first thread to finish was the IPv4 thread it indicates the preference is correct and we need to re-enforce this gently by decreasing the application-wide P value by 1. However, if the first thread to finish was the IPv6 thread it indicates an IPv4 connection problem and we need to aggressively avoid IPv4 by halving P and rounding towards 0.

- If P=0 (indicating equal preference), P is incremented if the first thread to complete was the IPv6 thread, or decremented if the first thread to complete was the IPv4 thread.

After adjusting P, it should never be larger than 4 seconds -- which is similar to the value used by many IPv6-capable TCP client applications to switch to an alternate A or AAAA record.

Note: Proof of concept tests on fast networks show that even smaller value (around 0.5 seconds) is practical. More extensive testing would be useful to find the best upper boundary that still ensures a good user experience.

4.2. Additional Considerations

This section discusses considerations and requirements that are common to new technology deployment.

4.2.1. Additional Network and Host Traffic

Additional network traffic and additional server load is created due to these recommendations and mitigated by application-wide and per-destination timer adjustments. The procedures described in this document retain a quality user experience while transitioning from IPv4-only to dual stack. The quality user experience benefits the

user but to the detriment of the network and server that are serving the user.

4.2.2. Abandon Non-Winning Connections

It is RECOMMENDED that the non-winning connections be abandoned, even though they could be used to download content. This is because some web sites provide HTTP clients with cookies (after logging in) that incorporate the client's IP address, or use IP addresses to identify users. If some connections from the same HTTP client are arriving from different IP addresses, such HTTP applications will break.

4.2.3. Flush or Expire Cache

Because every network has different characteristics (e.g., working or broken IPv6 connectivity) the IPv6/IPv4 preference value (P) SHOULD be reset to its default whenever the host is connected to a new network ([cx-osx], [cx-win]). However, in some instances the application and the host are unaware the network connectivity has changed so it is RECOMMENDED that per-destination values expire after 10 minutes of inactivity.

4.2.4. Determining Address Type

[[[IS THIS SECTION NECESSARY ??

For some transitional technologies such as a dual-stack host, it is easy for the application to recognize the native IPv6 address (learned via a AAAA query) and the native IPv4 address (learned via an A query). For other transitional technologies [RFC2766] it is impossible for the host to differentiate a transitional technology IPv6 address from a native IPv6 address (see Section 4.1 of [RFC4966]). Replacement transitional technologies are attempting to bridge this gap. It is necessary for applications to distinguish between native and transitional addresses in order to provide the most seamless user experience.

]]]

4.2.5. Debugging and Troubleshooting

This mechanism is aimed to help the user experience in case of connectivity problems. However, this precise reason also makes it tougher to use these applications as a means of the verification that the problems are fixed. To assist in that regard, the applications implementing the proposal in this document SHOULD also provide a mechanism to temporarily use only one address family.

4.2.6. DNS Behavior

Unique to DNS AAAA queries are the problems described in [RFC4074] which, if they still persist, require applications to perform an A query before the AAAA query.

[[Editor's Note: It is believed these defective DNS servers have long since been upgraded. If so, we can remove this section.]]

4.2.7. Thread safe DNS resolvers

Some applications and some OSs do not have thread safe DNS resolvers, which complicates implementation of simultaneous A and AAAA queries for IPv4/IPv6.

4.2.8. Middlebox Issues

Some devices are known to exhibit what amounts to a bug, when the A and AAAA requests are sent back-to-back over the same 4-tuple, and drop one of the requests or replies [DNS-middlebox]. However, in some cases fixing this behaviour may not be possible either due to the architectural limitations or due to the administrative constraints (location of the faulty device is unknown to the end hosts or not controlled by the end hosts). The algorithm described in this draft, in the case of this erroneous behaviour will eventually pace the queries such that this issue is will be avoided. The algorithm described in this draft also avoids calling the operating system's getaddrinfo() with "any", which should prevent the operating system from sending the A and AAAA queries on the same port.

4.2.9. Multiple Interfaces

Interaction of the suggestions in this document with multiple interfaces, and interaction with the MIF working group, is for further study.

4.3. Content Provider Recommendations

Content providers SHOULD provide both AAAA and A records for servers using the same DNS name for both IPv4 and IPv6.

4.4. Security Considerations

[[Placeholder.]]

See Section 4.2.2.

4.5. Acknowledgements

The mechanism described in this paper was inspired by Stuart Cheshire's discussion at the IAB Plenary at IETF72, the author's understanding of Safari's operation with SRV records, Interactive Connectivity Establishment (ICE [RFC5245]), and the current IPv4/IPv6 behavior of SMTP mail transfer agents.

Thanks to Fred Baker, Jeff Kinzli, Christian Kuhtz, and Iljitsch van Beijnum for fostering the creation of this document.

Thanks to Scott Brim and Stig Venaas for providing feedback on the document.

4.6. IANA Considerations

This document has no IANA actions.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

5.2. Informational References

[DNS-middlebox] Various, "DNS middlebox behavior with multiple queries over same source port", June 2009, <https://bugzilla.redhat.com/show_bug.cgi?id=505105>.

[RFC2766] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[RFC4074] Morishita, Y. and T. Jinmei, "Common Misbehavior Against DNS Queries for IPv6 Addresses", RFC 4074, May 2005.

[RFC4966] Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[cx-osx] Adium, "AIHostReachabilityMonitor", June 2009, <https://bugzilla.redhat.com/show_bug.cgi?id=505105>.

[cx-win] Microsoft, "NetworkChange.NetworkAvailabilityChanged Event", June 2009, <http://msdn.microsoft.com/en-us/library/system.net.networkinformation.networkchange.networkavailabilitychanged.aspx>.

[whitelist] Google, "Google IPv6 DNS Whitelist", March 2008, <http://www.google.com/intl/en/ipv6>.

Authors' Addresses

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA

Email: dwing@cisco.com

Andrew Yourtchenko
Cisco Systems, Inc.
De Kleetlaan, 7
San Jose, Diegem B-1831
Belgium

Email: ayourtch@cisco.com