Happy Eyeballs: Successful Introduction of New Technology to HTTP
draft-wing-http-new-tech-00

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 January 3, 2010.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

People like their computers to work quickly. During the transition to new technology, both old and new technologies have to peacefully
co-exist. However, if users experience connection delays attributed to the new technology the new technology will be shunned.

HTTP ("The Web") is one of the most visible and time-critical applications that is used by nearly every Internet user. It is critical that new technologies which improve HTTP not impair or delay the display of HTTP content. It is also important that users retain the ability to share URIs amongst friends and colleagues, even if the other users have not upgraded to the new technology.

This draft makes several recommendations to ensure user satisfaction and a smooth transition from HTTP’s pervasive IPv4 to IPv6 and from TCP to SCTP.

The audience for this draft is application developers and content providers. This draft is discussed on the Applications Discuss mailing list, https://www.ietf.org/mailman/listinfo/apps-discuss.
Table of Contents

1. Introduction ............................................. 4
2. Notational Conventions .................................. 4
3. Problem Statement ...................................... 4
   3.1. URIs and hostnames .................................. 5
   3.2. IPv6 .................................................. 5
   3.3. SCTP .................................................. 5
4. HTTP Client Recommendations ........................... 5
   4.1. IPv6 .................................................. 6
   4.2. SCTP .................................................. 7
5. Additional Considerations ............................... 9
   5.1. Additional Network and Host Traffic ............... 9
   5.2. Abandon Non-Winning Connections .................. 9
   5.3. Flush or Expire Cache .............................. 9
   5.4. Determining Address Type ........................... 9
   5.5. DNS Behavior ....................................... 10
   5.6. Thread safe DNS resolvers ........................ 10
   5.7. Middlebox Issues .................................. 10
   5.8. Multiple Interfaces ............................... 10
6. Content Provider Recommendations .................... 11
7. Security Considerations .................................. 11
8. Acknowledgements ........................................ 11
9. IANA Considerations ..................................... 11
10. References .............................................. 11
    10.1. Normative References .............................. 11
    10.2. Informational References ......................... 12
Authors’ Addresses ......................................... 13
1. Introduction

In order to use HTTP successfully over IPv6 or SCTP, it is necessary that the user enjoys nearly identical performance as compared to their old technology (IPv4 and TCP). A combination of today’s applications, IPv6 tunneling and IPv6 service providers, IPv4 NAT, 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, IPv4, SCTP, or TCP is the most optimal to connect to a server. The suggestions in this document provide a user experience which is superior to HTTP using TCP and IPv4, especially in IPv6/IPv4 transition environment with dual stack hosts (e.g., [RFC4213], DS-Lite [I-D.ietf-softwire-dual-stack-lite], 6rd [I-D.despres-6rd]).

The application recommendations in this document are primarily for HTTP clients ("web browsers") and may also be helpful for other 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, IPv6, SCTP, and TCP. 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, due to tunnel technologies might be slower than native IPv4 connectivity. However, due to port limitations inherent in stateful IPv6/IPv4 translators [BEHAVE], it is important that web browsers begin preferring IPv6 over IPv4 in order to avoid those port limitations.
As discussed in more detail in Section 3.3, there is no standard mechanism to indicate a host supports a non-TCP transport protocol, such as SCTP.

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, IPv6, TCP, or SCTP will only function if the other party also has 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 tunnel, and broken IPv6 peering. To prevent this delay an experiment with IPv6 connectivity, content providers use a separate namespace for their web server (e.g., ipv6.example.com), but doing that with production systems causes the problems described in Section 3.1.

3.3. SCTP

SCTP provides benefits over TCP [I-D.natarajan-http-over-sctp].

Unlike IPv6 which has an AAAA record, there is no DNS query that indicates a host supports SCTP [RFC4960], and HTTP URI scheme is not extensible to support an SRV query that could provide such support. Even if there was, it isn’t possible to determine if a middlebox, such as a firewall or a NAT, would block the SCTP association.

4. HTTP Client Recommendations

To provide fast connections for users, HTTP 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.

If an HTTP client supports IPv6 and SCTP (in addition to IPv4 and
TCP), the procedures described in Section 4.1 and Section 4.2 are performed together.

4.1. IPv6

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 HTTP client 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 HTTP client 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 5.2).

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 HTTP clients 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. SCTP

Due to the proliferation of NATs on the IPv4 Internet the best success for SCTP can be achieved by attempting both native SCTP connections and SCTP-over-UDP [I-D.tuexen-sctp-udp-encaps] connections.

For SCTP the following parameters are used:
SWAIT: Application-wide wait time for an SCTP association attempt to complete. Default value of 50ms is RECOMMENDED.

PREF: This denotes per-destination transport preference. Possible values are "TCP", "SCTP", and "BOTH". Default value of "BOTH" is RECOMMENDED.

The HTTP client starts several threads in order to minimize the user-noticeable delay ("dead time") during the connection attempts. The client starts one or more threads based on the following logic:

If ((PREF == BOTH) or (PREF == SCTP)) start thread 1. If making a connection using IPv4 start thread 2.

If ((PREF == BOTH) or (PREF == TCP)) start thread 3.

thread 1 (SCTP):
* Attempt to connect using SCTP (i.e., send SCTP INIT)

thread 2 (SCTP over UDP):
* Attempt to connect using SCTP over UDP (i.e., send SCTP INIT over UDP)

thread 3 (TCP):
* Attempt to connect using TCP

If an SCTP association attempt was made by a thread, the HTTP client waits for at least K ms; K = max(SWAIT, time taken for the TCP connection to complete). If the TCP connection finishes during this wait period, the HTTP client MAY choose TCP for the current HTTP transfer but MUST wait until K ms to figure if the SCTP association can be completed.

If the HTTP client did not choose TCP during the wait period and the SCTP association completes successfully, the HTTP client prefers SCTP over TCP connections and abandons the TCP connection.

After a connection is successful, we want to adjust the per-destination preference for this destination. It is not recommended to dynamically adjust the application-wide default value for SWAIT. If the SCTP association was successful, set destination’s PREF="SCTP", else set PREF="TCP".
5. Additional Considerations

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

5.1. 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 intent of this document is to show how good user experience can be maintained while the transitioning from IPv4 to IPv6, and transitioning from TCP to SCTP. The good user experience is to the benefit of the user but to the detriment of the network and server that are serving the user.

5.2. 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.

Editor’s note: If we can provide guidance to IPv6 and SCTP developers that connections from the same client could arrive on IPv4, IPv6, TCP, and SCTP we could eliminate the above paragraph. But could we be sure all web sites would follow such guidance?

5.3. Flush or Expire Cache

Because every network has different characteristics (working or broken IPv6 connectivity, middlebox that permits or blocks SCTP, etc.) the IPv6/IPv4 preference value (P) and the SCTP parameters (SWAIT and PREF) SHOULD be reset to their default whenever the host is connected to a new network. However, in some instances the application and the host are unaware the network connectivity has changed (e.g., when behind a NAT) so it is RECOMMENDED that per-destination values expire after 10 minutes of inactivity.

5.4. Determining Address Type

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.

5.5. 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.]]

5.6. 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.

5.7. 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.

5.8. Multiple Interfaces

Interaction of the suggestions in this document with multiple interfaces is for further study.
6. Content Provider Recommendations

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

7. Security Considerations

[[Placeholder.]]

See Section 5.2.

8. 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 [I-D.ietf-mmusic-ice]), 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.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[I-D.tuexen-sctp-udp-encaps]


10.2. 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>.

[I-D.despres-6rd]

[I-D.ietf-mmusic-ice]

[I-D.ietf-softwire-dual-stack-lite]
Durand, A., Droms, R., Haberman, B., and J. Woodyatt,
"Dual-stack lite broadband deployments post IPv4 exhaustion", draft-ietf-softwire-dual-stack-lite-00 (work in progress), March 2009.

[I-D.natarajan-http-over-sctp]
Natarajan, P., Amer, P., Leighton, J., and F. Baker,


[whitelist]
Google, "Google IPv6 DNS Whitelist", March 2008,
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

Preethi Natarajan
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA  95134
USA

Email:  prenatar@cisco.com