         Happy Eyeballs Version 2: Better Connectivity Using Concurrency
                       draft-ietf-v6ops-rfc6555bis-04

Abstract

   Many communication protocols operated over the modern Internet use
   host names. These often resolve to multiple IP addresses, each of
   which may have different performance and connectivity
   characteristics. Since specific addresses or address families (IPv4
   or IPv6) may be blocked, broken, or sub-optimal on a network, clients
   that attempt multiple connections in parallel have a higher chance of
   establishing a connection sooner. This document specifies
   requirements for algorithms that reduce this user-visible delay and
   provides an example algorithm.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   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."

   This Internet-Draft will expire on February 18, 2018.

carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

   Many communication protocols operated over the modern Internet use
   host names. These often resolve to multiple IP addresses, each of
   which may have different performance and connectivity
   characteristics. Since specific addresses or address families (IPv4
   or IPv6) may be blocked, broken, or sub-optimal on a network, clients
   that attempt multiple connections in parallel have a higher chance of
   establishing a connection sooner. This document specifies
   requirements for algorithms that reduce this user-visible delay and
   provides an example algorithm.

   This document expands on "Happy Eyeballs" [RFC6555], a technique of
   reducing user-visible delays on dual-stack hosts. Now that this
   approach has been deployed at scale and measured for several years,
   the algorithm specification can be refined to improve its reliability
   and generalization. This document recommends an algorithm of racing
   resolved addresses that has several stages of ordering and racing to
   avoid delays to the user whenever possible, while preferring the use
   of IPv6. Specifically, it discusses how to handle DNS queries when
   starting a connection on a dual-stack client, how to create an
   ordered list of destination addresses to which to attempt
   connections, and how to race the connection attempts.

1.1. Requirements Language

   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
   "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

2. Overview

   This document defines a method of connection establishment, named
   "Happy Eyeballs Connection Setup". This approach has several
   distinct phases:

   1. Initiation of asynchronous DNS queries [Section 3]

   2. Sorting of resolved destination addresses [Section 4]

   3. Initiation of asynchronous connection attempts [Section 5]

   4. Establishment of one connection, which cancels all other attempts

Note that this document assumes that the host destination address
preference policy favors IPv6 over IPv4. If the host is configured
differently, the recommendations in this document can be easily
adapted.

3. **Hostname Resolution Query Handling**

When a client has both IPv4 and IPv6 connectivity, and is trying to
establish a connection with a named host, it needs to send out both
AAAA and A DNS queries. Both queries SHOULD be made as soon after
one another as possible, with the AAAA query made first, immediately
followed by the A query.

Implementations SHOULD NOT wait for both families of answers to
return before attempting connection establishment. If one query
fails to return, or takes significantly longer to return, waiting for
the second address family can significantly delay the connection
establishment of the first one. Therefore, the client SHOULD treat
DNS resolution as asynchronous. Note that if the platform does not
offer an asynchronous DNS API, this behavior can be simulated by
making two separate synchronous queries on different threads, one per
address family.

The RECOMMENDED algorithm proceeds as follows: if the AAAA query
returns first, the first IPv6 connection attempt is immediately
started. If the A query returns first due to reordering, the client
SHOULD wait for a short time for the AAAA response to ensure
preference is given to IPv6 (it is common for the AAAA response to
follow the A response by a few milliseconds). This delay will be
referred to as the "Resolution Delay". The RECOMMENDED value for the
Resolution Delay is 50 milliseconds. If the AAAA response is
received within the Resolution Delay period, the client immediately
starts the IPv6 connection attempt. If, at the end of the Resolution
Delay period, the AAAA response has not been received but the A
response has been received, the client SHOULD proceed to Sorting
Addresses [Section 4] and staggered connection attempts [Section 5]
using only the IPv4 addresses returned so far. If the AAAA response
arrives while these connection attempts are in progress, but before
any connection has been established, then the newly received IPv6
addresses are incorporated into the list of available candidate
addresses [Section 6] and the process of connection attempts will
continue with the IPv6 addresses added, until one connection is
established.

3.1. Handling Multiple DNS Server Addresses

If multiple DNS server addresses are configured for the current
network, the client may have the option of sending its DNS queries
over IPv4 or IPv6. In keeping with the Happy Eyeballs approach,
queries SHOULD be sent over IPv6 first (note that this is not
referring to the sending of AAAA or A queries, but rather the address
of the DNS server itself and IP version used to transport DNS
messages). If DNS queries sent to the IPv6 address do not receive
responses, that address may be marked as penalized, and queries can
be sent to other DNS server addresses.

As native IPv6 deployments become more prevalent, and IPv4 addresses
are exhausted, it is expected that IPv6 connectivity will have
preferential treatment within networks. If a DNS server is
configured to be accessible over IPv6, IPv6 should be assumed to be
the preferred address family.

Client systems SHOULD NOT have an explicit limit to the number of DNS
servers that can be configured, either manually or by the network.
If such a limit is required by hardware limitations, it is
RECOMMENDED to use at least one address from each address family from
the available list.

4. Sorting Addresses

Before attempting to connect to any of the resolved destination
addresses, the client should define the order in which to start the
attempts. Once the order has been defined, the client can use a
simple algorithm for racing each option after a short delay
[Section 5]. It is important that the ordered list involves all
addresses from both families that have been received by this point,
as this allows the client to get the racing effect of Happy Eyeballs
for the entire list, not just the first IPv4 and first IPv6
addresses.

First, the client MUST sort the addresses received up to this point
using Destination Address Selection ([RFC6724], Section 6).

If the client is stateful and has history of expected round-trip
times (RTT) for the routes to access each address, it SHOULD add a
Destination Address Selection rule between rules 8 and 9 that prefers
addresses with lower RTTs. If the client keeps track of which
addresses it has used in the past, it SHOULD add another destination
address selection rule between the RTT rule and rule 9, which prefers
used addresses over unused ones. This helps servers that use the
client's IP address during authentication, as is the case for TCP
Fast Open [RFC7413] and some HTTP cookies. This historical data MUST
NOT be used across networks, and SHOULD be flushed on network
changes.

Next, the client SHOULD modify the ordered list to interleave address
families. Whichever address family is first in the list should be
followed by an address of the other address family; that is, if the
first address in the sorted list is IPv6, then the first IPv4 address
should be moved up in the list to be second in the list. An
implementation MAY want to favor one address family more by allowing
multiple addresses of that family to be attempted before trying the
other family. The number of contiguous addresses of the first
address family will be referred to as the "First Address Family
Count", and can be a configurable value. This is performed to avoid
waiting through a long list of addresses from a given address family
if connectivity over that address family is impaired.

Note that the address selection described in this section only
applies to destination addresses; Source Address Selection
([RFC6724], Section 5) is performed once per destination address and
is out of scope of this document.

5. Connection Attempts

Once the list of addresses received up to this point has been
constructed, the client will attempt to make connections. In order
to avoid unreasonable network load, connection attempts SHOULD NOT be
made simultaneously. Instead, one connection attempt to a single
address is started first, followed by the others in the list, one at
a time. Starting a new connection attempt does not affect previous
attempts, as multiple connection attempts may occur in parallel.
Once one of the connection attempts succeeds (generally when the TCP
handshake completes), all other connections attempts that have not
yet succeeded SHOULD be cancelled. Any address that was not yet
attempted as a connection SHOULD be ignored. At that time, the
asynchronous DNS query MAY be cancelled as new addresses will not be
used for this connection. However, the DNS client resolver SHOULD
still process DNS replies from the network for a short period of time
(RECOMMENDED at 1 second), as they will populate the DNS cache and
can be used for subsequent connections.

A simple implementation can have a fixed delay for how long to wait
before starting the next connection attempt. This delay is referred
to as the "Connection Attempt Delay". One recommended value for this
delay is 250 milliseconds. If the client has historical RTT data, it
can also use the expected RTT to choose a more nuanced delay value.
The recommended formula for calculating the delay after starting a
connection attempt is: MAX(1.25 * RTT_MEAN + 4 * RTT_VARIANCE, 2 *
RTT_MEAN), where the RTT values are based on the statistics for
previous address used. If the TCP implementation leverages
historical RTT data to compute SYN timeout, these algorithms should
match so that a new attempt will be started at the same time as the
previous is sending its second TCP SYN. While TCP implementations
often leverage an exponential backoff when they detect packet loss,
the "Connection Attempt Delay" SHOULD NOT perform such an aggressive
backoff, as it would harm user experience.

The Connection Attempt Delay MUST have a lower bound, especially if
it is computed using historical data. More specifically, a
subsequent connection MUST NOT be started within 10 milliseconds of
the previous attempt. The recommended minimum value is 100
milliseconds, which is referred to as the "Minimum Connection Attempt
Delay". This minimum value is required to avoid congestion collapse
in the presence of high packet loss rates. The Connection Attempt
Delay SHOULD have an upper bound, referred to as the "Maximum
Connection Attempt Delay". The current recommended value is 2
seconds.

6. DNS Answer Changes during Happy Eyeballs Connection Setup

If, during the course of connection establishment, the DNS answers
change either by adding resolved addresses (for example, due to DNS
push notifications [DNS-PUSH]), or removing previously resolved
addresses (for example, due to expiry of the TTL on that DNS record),
the client should react based on its current progress.

If an address is removed from the list that already had a connection
attempt started, the connection attempt SHOULD NOT be cancelled, but
rather be allowed to continue. If the removed address had not yet
had a connection attempt started, it SHOULD be removed from the list
of addresses to try.

If an address is added to the list, it should be sorted into the list
of addresses not yet attempted according to the rules above
(Section 4).

7. Supporting IPv6-only Networks with NAT64 and DNS64

   While many IPv6 transition protocols have been standardized and
   deployed, most are transparent to client devices. The combined use
   of NAT64 [RFC6146] and DNS64 [RFC6147] is a popular solution that is
   being deployed and requires changes in client devices. One possible
   way to handle these networks is for the client device networking
   stack to implement 464XLAT [RFC6877]. 464XLAT has the advantage of
   not requiring changes to user space software, however it requires
   per-packet translation if the application is using IPv4 literals and
   does not encourage client application software to support native
   IPv6. On platforms that do not support 464XLAT, the Happy Eyeballs
   engine SHOULD follow the recommendations in this section to properly
   support IPv6-only networks with NAT64 and DNS64.

   The features described in this section SHOULD only be enabled when
   the host detects one of these networks. A simple heuristic to
   achieve that is to check if the network offers routable IPv6
   addressing, does not offer routable IPv4 addressing, and offers a DNS
   resolver address.

7.1. IPv4 Address Literals

   If client applications or users wish to connect to IPv4 address
   literals, the Happy Eyeballs engine will need to perform NAT64
   address synthesis for them. The solution is similar to "Bump-in-the-
   Host" [RFC6535] but is implemented inside the Happy Eyeballs library.

   When an IPv4 address is passed in to the library instead of a host
   name, the device queries the network for the NAT64 prefix using
   "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis"
   [RFC7050] then synthesizes an appropriate IPv6 address (or several)
   using the encoding described in "IPv6 Addressing of IPv4/IPv6
   Translators" [RFC6052]. The synthesized addresses are then inserted
   into the list of addresses as if they were results from DNS queries;
   connection attempts follow the algorithm described above (Section 5).

7.2. Host Names with Broken AAAA Records

   At the time of writing, there exist a small but non negligible number
   of host names that resolve to valid A records and broken AAAA
   records, which we define as AAAA records that contain seemingly valid
   IPv6 addresses but those addresses never reply when contacted on the
   usual ports. These can be for example caused by:

   o Mistyping of the IPv6 address in the DNS zone configuration

   o Routing black holes

   o Service outages

   While an algorithm complying with the other sections of this document
   would correctly handle such host names on a dual-stack network, they
   will not necessarily function correctly on IPv6-only networks with
   NAT64 and DNS64. Since DNS64 recursive resolvers rely on the
   authoritative name servers sending negative ("no error no answer")
   responses for AAAA records in order to synthesize, they will not
   synthesize records for these particular host names, and will instead
   pass through the broken AAAA record.

   In order to support these scenarios, the client device needs to query
   the DNS for the A record then perform local synthesis. Since these
   types of host names are rare, and in order to minimize load on DNS
   servers, this A query should only be performed when the client has
   given up on the AAAA records it initially received. This can be
   achieved by using a longer timeout, referred to as the "Last Resort
   Local Synthesis Delay" and RECOMMENDED at 2 seconds. The timer is
   started when the last connection attempt is fired. If no connection
   attempt has succeeded when this timer fires, the device queries the
   DNS for the IPv4 address and on reception of a valid A record, treats
   it as if it were provided by the application (Section 7.1).

7.3. Virtual Private Networks

   Some Virtual Private Networks (VPN) may be configured to handle DNS
   queries from the device. The configuration could encompass all
   queries, or a subset such as "*.internal.example.com". These VPNs
   can also be configured to only route part of the IPv4 address space,
   such as 192.0.2.0/24. However, if an internal hostname resolves to
   an external IPv4 address, these can cause issues if the underlying
   network is IPv6-only. As an example, let's assume that
   "www.internal.example.com" has exactly one A record, 198.51.100.42,
   and no AAAA records. The client will send the DNS query to the
   company's recursive resolver and that resolver will reply with these
   records. The device now only has an IPv4 address to connect to, and
   no route to that address. Since the company's resolver does not know
   the NAT64 prefix of the underlying network, it cannot synthesize the
   address. Similarly, the underlying network's DNS64 recursive
   resolver does not know the company's internal addresses, so it cannot
   resolve the hostname. Because of this, the client device needs to
   resolve the A record using the company's resolver then locally
   synthesize an IPv6 address, as if the resolved IPv4 address were
   provided by the application (Section 7.1).

8. Summary of Configurable Values

   The values that may be configured as defaults on a client for use in
   Happy Eyeballs are as follows:

   o Resolution Delay (Section 3): The time to wait for a AAAA response
      after receiving an A response. RECOMMENDED at 50 milliseconds.

   o First Address Family Count (Section 4): The number of addresses
      belonging to the first address family (such as IPv6) that should
      be attempted before attempting another address family.
      RECOMMENDED as 1, or 2 to more aggressively favor one address
      family.

   o Connection Attempt Delay (Section 5): The time to wait between
      connection attempts in the absence of RTT data. RECOMMENDED at
      250 milliseconds.

   o Minimum Connection Attempt Delay (Section 5): The minimum time to
      wait between connection attempts. RECOMMENDED at 100
      milliseconds. MUST NOT be less than 10 milliseconds.

   o Maximum Connection Attempt Delay (Section 5): The maximum time to
      wait between connection attempts. RECOMMENDED at 2 seconds.

   o Last Resort Local Synthesis Delay (Section 7.2): The time to wait
      after starting the last IPv6 attempt and before sending the A
      query. RECOMMENDED at 2 seconds.

   The delay values described in this section were determined
   empirically by measuring the timing of connections on a very wide set
   of production devices. They were picked to reduce wait times noticed
   by users while minimizing load on the network. As time passes, it is
   expected that the properties of networks will evolve. For that
   reason, it is expected that these values will change over time.
   Implementors should feel welcome to use different values without
   changing this specification. Since IPv6 issues are expected to be
   less common, the delays SHOULD be increased with time as client
   software is updated.

9. Limitations

   Happy Eyeballs will handle initial connection failures at the TCP/IP
   layer, however other failures or performance issues may still affect
   the chosen connection.

9.1. Path Maximum Transmission Unit Discovery

Since Happy Eyeballs is only active during the initial handshake and
TCP does not pass the initial handshake, issues related to MTU can be
masked and go unnoticed during Happy Eyeballs. Solving this issue is
out of scope of this document. One solution is to use Packetization
Layer Path MTU Discovery [RFC4821].

9.2. Application Layer

If the DNS returns multiple addresses for different application
servers, the application itself may not be operational and functional
on all of them. Common examples include Transport Layer Security
(TLS) and the Hypertext Transport Protocol (HTTP).

9.3. Hiding Operational Issues

It has been observed in practice that Happy Eyeballs can hide issues
in networks. For example, if a misconfiguration causes IPv6 to
consistently fail on a given network while IPv4 is still functional,
Happy Eyeballs may impair the operator's ability to notice the issue.
It is recommended that network operators deploy external means of
monitoring to ensure functionality of all address families.

10. Security Considerations

This memo has no direct security considerations.

11. IANA Considerations

This memo includes no request to IANA.

12. Acknowledgments

The authors thank Dan Wing, Andrew Yourtchenko, and everyone else who
worked on the original Happy Eyeballs design [RFC6555], Josh
Graessley, Stuart Cheshire, and the rest of team at Apple that helped
implement and instrument this algorithm, and Jason Fesler and Paul
Saab who helped measure and refine this algorithm. The authors would
also like to thank Fred Baker, Nick Chettle, Lorenzo Colitti, Igor
Gashinsky, Geoff Huston, Jen Linkova, Paul Hoffman, Philip Homburg,
Erik Nygren, Jordi Palet Martinez, Rui Paulo, Stephen Strowes, Jinmei
Tatuya, Dave Thaler, Joe Touch and James Woodyatt for their input and
contributions.

13. References

13.1. 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,
           <http://www.rfc-editor.org/info/rfc2119>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU
           Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007,
           <https://www.rfc-editor.org/info/rfc4821>.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X.
           Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052,
           DOI 10.17487/RFC6052, October 2010, <https://www.rfc-
           editor.org/info/rfc6052>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful
           NAT64: Network Address and Protocol Translation from IPv6
           Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146,
           April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van
           Beijnum, "DNS64: DNS Extensions for Network Address
           Translation from IPv6 Clients to IPv4 Servers", RFC 6147,
           DOI 10.17487/RFC6147, April 2011, <https://www.rfc-
           editor.org/info/rfc6147>.

[RFC6535] Huang, B., Deng, H., and T. Savolainen, "Dual-Stack Hosts
           Using "Bump-in-the-Host" (BIH)", RFC 6535,
           DOI 10.17487/RFC6535, February 2012, <https://www.rfc-
           editor.org/info/rfc6535>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with
           Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April
           2012, <https://www.rfc-editor.org/info/rfc6555>.

[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/info/rfc6724>.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of
           the IPv6 Prefix Used for IPv6 Address Synthesis",
           RFC 7050, DOI 10.17487/RFC7050, November 2013,
           <https://www.rfc-editor.org/info/rfc7050>.

13.2. Informative References

[DNS-PUSH]
           Pusateri, T. and S. Cheshire, "DNS Push Notifications",
           Work in Progress, draft-ietf-dnssd-push, March 2017.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT:
           Combination of Stateful and Stateless Translation",
           RFC 6877, DOI 10.17487/RFC6877, April 2013,
           <https://www.rfc-editor.org/info/rfc6877>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP
           Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014,
           <https://www.rfc-editor.org/info/rfc7413>.

Appendix A. Differences from RFC6555

"Happy Eyeballs: Success with Dual-Stack Hosts" [RFC6555] mostly
concentrates on how to stagger connections to a hostname that has an
AAAA and an A record. This document additionally discusses:

o how to perform DNS queries to obtain these addresses

o how to handle multiple addresses from each address family

o how to handle DNS updates while connections are being raced

o how to leverage historical information

o how to support IPv6-only networks with NAT64 and DNS64

Authors' Addresses

    David Schinazi
    Apple Inc.
    1 Infinite Loop
    Cupertino, California 95014
    US

    Email: dschinazi@apple.com

    Tommy Pauly
    Apple Inc.
    1 Infinite Loop
    Cupertino, California 95014
    US

    Email: tpauly@apple.com