Network
Internet-Draft
Obsoletes: 6555 (if approved)
Intended status: Standards Track
Expires: October 10, 2017

T. Pauly
D. Schinazi
Apple Inc.

April 8, 2017

Happy Eyeballs Version 2: Better Connectivity Using Concurrency
draft-ietf-v6ops-rfc6555bis-00

Abstract

Many communication protocols operated over the modern Internet uses 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.

Table of Contents

1. Introduction

Many communication protocols operated over the modern Internet uses host names. These often correspond to multiple IP addresses, whose performance can vary. 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 algorithm.

This documents 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 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" RFC 2119 [RFC2119].

2. Overview

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

1. Initiation of asynchronous DNS queries [Section 3]

2. Sorting of resolved 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 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 MUST 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 MUST treat DNS resolution as asynchronous. Note that if the platforms 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. If the AAAA query returns first, the first IPv6

connection attempt MUST be immediately started. If the A query returns first, the client SHOULD wait for a short time for the AAAA response. 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 MUST immediately start 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). 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 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, 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 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 for 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.

5. Connection Attempts

Once the list of addresses 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.

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 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 congestive 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. Summary of Configurable Values

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

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

- 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.

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

- 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.

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

As time advances, 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. In particular, IPv6 issues are expected to be less common, therefore the Resolution Delay SHOULD be increased with time as client software is updated.

8. 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.

8.1. Path Maximum Transmission Unit Discovery

Since Happy Eyeballs is only active during the initial handshake and TCP does not pas 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.

8.2. Application Layer

If the DNS returns multiple application servers for a given service, 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).

8.3. 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.

9. Security Considerations

This memo has no direct security considerations.

10. IANA Considerations

This memo includes no request to IANA.

11. 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 Nick Chettle, Paul Hoffman, Philip Homburg, Joe Touch and James Woodyatt for their input and contributions.

12. References

12.1. 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>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://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, <http://www.rfc-editor.org/info/rfc6724>.

12.2. Informative References

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

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://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:

- how to perform DNS queries to obtain these addresses
- how to handle multiple addresses from each address family
- how to handle DNS updates while connections are being raced
- how to leverage historical information

Authors' Addresses

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

Email: tpauly@apple.com

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

Email: dschinazi@apple.com