Workgroup: Network Working Group
Internet-Draft:
draft-pauly-v6ops-happy-eyeballs-v3-00
Published: 23 October 2023
Intended Status: Informational
Expires: 25 April 2024
Authors: T. Pauly Apple Inc
D. Schinazi Google LLC
N. Jaju Google LLC
K. Ishibashi Google LLC

Happy Eyeballs Version 3: Better Connectivity Using Concurrency

Abstract

Many communication protocols operating over the modern Internet use hostnames. 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 chance of establishing a connection more quickly. This document specifies requirements for algorithms that reduce this user-visible delay and provides an example algorithm, referred to as "Happy Eyeballs". This document updates the algorithm description in RFC 8305.

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://tfpauly.github.io/draft-happy-eyeballs-v3/draft-pauly-v6ops-happy-eyeballs-v3.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-pauly-v6ops-happy-eyeballs-v3/.

Source for this draft and an issue tracker can be found at https://github.com/tfpauly/draft-happy-eyeballs-v3.

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 https://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 25 April 2024.

Copyright Notice

Table of Contents

1. Introduction

Many communication protocols operating over the modern Internet use hostnames. 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 chance of establishing a connection more quickly. This document specifies requirements for algorithms that reduce this user-visible delay and provides an example algorithm.

This document defines the algorithm for "Happy Eyeballs", a technique for reducing user-visible delays on dual-stack hosts. This definition updates the description in [HEV2], which itself obsoleted [RFC6555].

The Happy Eyeballs algorithm of racing connections to resolved addresses has several stages to avoid delays to the user whenever possible, while preferring the use of IPv6. This document 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.

As compared to [HEV2], this document adds support for incorporating SVCB / HTTPS resource records (RRs) [SVCB]. SVCB RRs provide alternative endpoints and associated information about protocol support, Encrypted ClientHello [ECH] keys, address hints, among other relevant hints which may help speed up connection establishment and improve user privacy. Discovering protocol support during resolution, such as for HTTP/3 over QUIC [RFC9114], allows upgrading between protocols on the current connection attempts, instead of waiting for subsequent attempts to use information from other discovery mechanisms such as HTTP Alternative Services [AltSvc]. These records can be queried along with A and AAAA records, and the updated algorithm defines how to handle SVCB responses to improve address and protocol selection.

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. Overview

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

1. Initiation of asynchronous DNS queries (Section 4)

2. Sorting of resolved destination addresses (Section 5)

3. Initiation of asynchronous connection attempts (Section 6)

4. Establishment of one connection, which cancels all other attempts (Section 6)

Note that this document assumes that the preference policy for the host destination address favors IPv6 over IPv4. IPv6 has many desirable properties designed to be improvements over IPv4 [IPV6].

This document also assumes that the preference policy favors QUIC over TCP. QUIC only requires one packet to establish a secure connection, making it quicker compared to TCP [QUIC].

If the host is configured to have different preferences, the recommendations in this document can be easily adapted.

4. 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 and immediately followed by the A query.

Additionally, if the client also wants to receive SVCB / HTTPS resource records (RRs) [SVCB], it **SHOULD** issue the SVCB query immediately before the AAAA and A queries (prioritizing the SVCB query since it can also include address hints). If the client has only one of IPv4 or IPv6 connectivity, it still issues the SVCB query prior to whichever AAAA or A query is appropriate. Note that upon receiving a SVCB answer, the client might need to issue futher AAAA and/or A queries to resolve the service name included in the RR.

Implementations **SHOULD NOT** wait for all answers to return before attempting connection establishment. If one query fails to return or takes significantly longer to return, waiting for the other answers 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 separate synchronous queries, each on a different thread.

The algorithm for acting upon received answers depends on whether the client sent out queries for SVCB RRs.

If the client did not request SVCB RRs:

- If a positive AAAA response (a response containing at least one valid AAAA RR) is received first, the first IPv6 connection attempt is immediately started.

- If a positive A response is received first (which might be due to reordering), the client **SHOULD** wait a short time for the AAAA response to ensure that preference is given to IPv6, since it is common for the AAAA response to follow the A response by a few milliseconds. This delay is referred to as the "Resolution Delay". If a negative AAAA response (no error, no data) is received within the Resolution Delay period or the AAAA response has not been received by the end of the Resolution Delay period, the client **SHOULD** proceed to sorting addresses (see Section 5) and staggered connection attempts (see Section 6) using any IPv4 addresses received so far.

If the client did request SVCB RRs:

- If the client receives any positive response back (containing a valid AAAA, A, or SVCB ServiceMode RR), it starts the Resolution Delay timer, which is run until both the AAAA and SVCB ServiceMode responses are received, or a SVCB response is received that also includes at least one address in the ipv6hint parameter. Once both records are received, or the timer expires, the client proceeds with the process of sorting addresses and staggered connection attempts.

For both variations of the algorithm, the **RECOMMENDED** value for the Resolution Delay is 50 milliseconds.

If new positive responses arrive while connection attempts are in progress, but before any connection has been established, then the newly received addresses are incorporated into the list of available candidate addresses (see Section 7) and the process of connection attempts will continue with the new addresses added, until one connection is established.

4.1. 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, the client **SHOULD** use at least one address from each address family from the available list.

5. 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 (see Section 6). It is important that the ordered list involve all addresses from both families and all protocols 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.

Note that the following sorting steps are an incremental sort, meaning that the client **SHOULD** sort within each sorted group for each incremental step.

If any of the answers were from SVCB RRs, they **SHOULD** be sorted ahead of any answers that were not associated with a SVCB record.

If the client supports TLS Encrypted Client Hello (ECH) discovery through SVCB records [SVCB-ECH], depending on the client's preference to handle ECH, the client **SHOULD** sort addresses with ECH keys taking priority to maintain privacy when attempting connection establishment.

The client then sorts the addresses received up to this point, within each group, by service priority if the set of addresses contain any answers from SVCB records. See Section 5.1 for details.

The client **SHOULD** also sort the addresses in protocol order, such that QUIC is prioritized over TCP, as it connects faster and

generally results in a better experience once connected. For example, QUIC provides improved delivery and congestion control, supports connection migration, and provides other benefits [QUIC].

Then, within each group at equal priority, the client **MUST** sort the addresses using Destination Address Selection ([RFC6724], Section 6).

If the client is stateful and has a history of expected round-trip times (RTTs) 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 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 Hypertext Transport Protocol (HTTP) cookies. This historical data **MUST NOT** be used across different network interfaces and **SHOULD** be flushed whenever a device changes the network to which it is attached.

Next, the client **SHOULD** modify the ordered list to interleave protocols and address families. Whichever combination of protocol and address family is first in the list should be followed by an address of the other protocol type and same address family, and then an endpoint from the same protocol and other address family. For example, if the first address in the sorted list is a QUIC IPv6 address, then the first TCP IPv6 address should be moved up in the list to be second in the list, and then the first QUIC IPv4 address should be moved up in the list to be third in the list. An implementation **MAY** choose to favor one protocol or address family more by allowing multiple addresses of that protocol or family to be attempted before trying the other combinations. The number of contiguous addresses of the first combination of properties will be referred to as the "Preferred Protocol Combination Count" and can be a configurable value. This avoids waiting through a long list of addresses from a given address family using a given protocol if connectivity over a protocol or an 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.

5.1. Sorting Based on Priority

SVCB [SVCB] RRs indicate a priority for each ServiceMode response. This prioirity applies to any IPv4 or IPv6 address hints in the RR itself, as well as any addresses received on A or AAAA queries for

the name in the Servicemode RR. The priority in a SVCB RR is always greater than 0.

SVCB answers with the lowest numerical value (such as 1) are sorted first, and answers with higher numerical values are sorted later.

Note that a SVCB RR with the TargetName "." applies to the owner name in the RR, and the priority of that SVCB RR applies to any A or AAAA RRs for the same owner name. These answers are sorted according to that SVCB RR's priority.

6. 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 (Section 6.1), all other connections attempts that have not yet succeeded **SHOULD** be canceled. Any address that was not yet attempted as a connection **SHOULD** be ignored. At that time, the asynchronous DNS query **MAY** be canceled 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 to be 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 a default delay is 250 milliseconds. A more nuanced implementation's delay should correspond to the time when the previous attempt is retrying its handshake (such as sending a second TCP SYN or a second QUIC Initial), based on the retransmission timer ([RFC6298], [RFC9002]). If the client has historical RTT data gathered from other connections to the same host or prefix, it can use this information to influence its delay. Note that this algorithm should only try to approximate the time of the first handshake packet retransmission, and not any further retransmissions that may be influenced by exponential timer back off.

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.

6.1. Determining successful connection establishment

The determination of when a connection attempt has successfully completed (and other attempts can be cancelled) depends on the protocols being used to establish a connection. This can involve one or more protocol handshakes.

Client connections that use TCP only (without TLS or another protocol on top, such as for unencrypted HTTP connections) will determine successful establishment based on completing the TCP handshake only. When TLS is used on top of of TCP (such as for encrypted HTTP connections), clients **MAY** choose to wait for the TLS handshake to successfully complete before cancelling other connection attempts. This is particularly useful for networks in which a TCP-terminating proxy might be causing TCP handshakes to succeed quickly, even though end-to-end connectivity with the TLS-terminating server will fail. QUIC connections inherently include a secure handshake in their main handshakes, and thus only need to wait for a single handshake to complete.

While transport layer handshakes generally do not have restrictions on attempts to establish a connection, some cryptographic handshakes may be dependent on ServiceMode SVCB RRs and could impose limitations on establishing a connection. For instance, ECH-capable clients may become SVCB-reliant clients (Section 3 of [SVCB]) when SVCB RRs contain the "ech" SvcParamKey [ECH]. If the client is either an SVCB-reliant client or a SVCB-optional client that might switch to SVCB-reliant connection establishment during the process, the client **MUST** wait for SVCB RRs before proceeding with the cryptographic handshake.

6.2. Handling Application Layer Protocol Negotiation (ALPN)

The alpn and no-default-alpn SvcParamKeys in SVCB RRs indicate the "SVCB ALPN set," which specifies the underlying transport protocols supported by the associated service endpoint. When the client requests SVCB RRs, it **SHOULD** perform the procedure specified in Section 7.1.2 of [SVCB] to determine the underlying transport protocols that both the client and the service endpoint support. The client **SHOULD NOT** attempt to make a connection to a service endpoint whose SVCB ALPN set does not contain any protocols that the client supports. For example, suppose the client is an HTTP client that

only supports TCP-based versions such as HTTP/1.1 and HTTP/2, and it receives the following HTTPS RR:

```
example.com. 60 IN HTTPS 1 svc1.example.com. (
    alpn="h3" no-default-alpn ipv6hint=2001:db8::2 )
```

In this case, attempting a connection to 2001:db8::2 or any other address resolved for svc1.example.com would be incorrect because the RR indicates that svc1.example.com only supports HTTP/3, based on the ALPN value of "h3".

If the client is an HTTP client that supports both Alt-Svc [AltSvc] and SVCB (HTTPS) RRs, the client **SHOULD** ensure that connection attempts are consistent with both the Alt-Svc parameters and the SVCB ALPN set, as specified in Section 9.3 of [SVCB].

7. DNS Answer Changes During Happy Eyeballs Connection Setup

If, during the course of connection establishment, the DNS answers change by either adding resolved addresses (for example due to DNS push notifications [RFC8765]) 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 canceled, 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 (see Section 5).

8. 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.

8.1. 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 into the library instead of a hostname, the device queries the network for the NAT64 prefix using "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis" [RFC7050] and 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 (see Section 6).

Such translation also applies to any IPv4 address hints received in SVCB RRs.

8.2. Hostnames with Broken AAAA Records

At the time of writing, there exist a small but non-negligible number of hostnames 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:

- Mistyping of the IPv6 address in the DNS zone configuration
- Routing black holes
- Service outages

While an algorithm complying with the other sections of this document would correctly handle such hostnames 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 hostnames 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 and then perform local synthesis. Since these types of hostnames 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"; the delay is recommended to be 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 (see Section 8.1).

8.3. Virtual Private Networks

Some Virtual Private Networks (VPNs) 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 and then locally synthesize an IPv6 address, as if the resolved IPv4 address were provided by the application (Section 8.1).

9. 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 4): The time to wait for a AAAA response after receiving an A response. Recommended to be 50 milliseconds.

* Preferred Protocol Combination Count (Section 5): The number of addresses belonging to the preferred address family (such as IPv6) using the preferred protocol (such as QUIC) that should be attempted before attempting the next combination of address family and protocol. Recommended to be 1; 2 may be used to more

aggressively favor a particular combination of address family and protocol.

* Connection Attempt Delay (Section 6): The time to wait between connection attempts in the absence of RTT data. Recommended to be 250 milliseconds.

* Minimum Connection Attempt Delay (Section 6): The minimum time to wait between connection attempts. Recommended to be 100 milliseconds. **MUST NOT** be less than 10 milliseconds.

* Maximum Connection Attempt Delay (Section 6): The maximum time to wait between connection attempts. Recommended to be 2 seconds.

* Last Resort Local Synthesis Delay (Section 8.2): The time to wait after starting the last IPv6 attempt and before sending the A query. Recommended to be 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.

10. Limitations

Happy Eyeballs will handle initial connection failures at the transport layer (such as TCP or QUIC); however, other failures or performance issues may still affect the chosen connection.

10.1. Path Maximum Transmission Unit Discovery

For TCP connections, 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. For QUIC connections, a minimum MTU of at least 1200 bytes [RFC9000], Section 8.1-5 is guaranteed, but there is a chance that larger values may not be available. Solving this issue is out of scope of this document. One solution is to use "Packetization Layer Path MTU Discovery" [RFC4821].

10.2. 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 HTTP.

10.3. 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.

11. Security Considerations

Note that applications should not rely upon a stable hostname-to-address mapping to ensure any security properties, since DNS results may change between queries. Happy Eyeballs may make it more likely that subsequent connections to a single hostname use different IP addresses.

12. IANA Considerations

This document does not require any IANA actions.

13. References

13.1. Normative References

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-17, 9 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-17>.

[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>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/rfc/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/rfc/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/rfc/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/rfc/rfc6147>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/rfc/rfc6298>.

[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/rfc/rfc6535>.

[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>.

[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/rfc/rfc7050>.

[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>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[SVCB] Schwartz, B. M., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-12, 11 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-svcb-https-12>.

[SVCB-ECH] Schwartz, B. M., Bishop, M., and E. Nygren, "Bootstrapping TLS Encrypted ClientHello with DNS Service Bindings", Work in Progress, Internet-Draft, draft-sbn-tls-svcb-ech-00, 11 March 2023, <https://datatracker.ietf.org/doc/html/draft-sbn-tls-svcb-ech-00>.

13.2. Informative References

[AltSvc] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://www.rfc-editor.org/rfc/rfc7838>.

[HEV2] 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>.

[IPV6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/rfc/rfc8200>.

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[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/rfc/rfc6555>.

[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/rfc/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/rfc/rfc7413>.

[RFC8765] Pusateri, T. and S. Cheshire, "DNS Push Notifications", RFC 8765, DOI 10.17487/RFC8765, June 2020, <https://www.rfc-editor.org/rfc/rfc8765>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

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, Warren Kumari, Erik Nygren, Jordi Palet Martinez, Rui Paulo, Stephen Strowes, Jinmei Tatuya, Dave Thaler, Joe Touch, and James Woodyatt for their input and contributions.

Authors' Addresses

Tommy Pauly
Apple Inc

Email: tpauly@apple.com

David Schinazi
Google LLC

Email: dschinazi.ietf@gmail.com

Nidhi Jaju
Google LLC

Email: nidhijaju@google.com

Kenichi Ishibashi
Google LLC

Email: bashi@google.com