Happy Eyeballs, Version 3: *Better Connectivity Using Concurrency*

draft-pauly-happy-happyeyeballs-v3

Tommy Pauly, David Schinazi, Kenichi Ishibashi, Nidhi Jaju

HAPPY - IETF 122 Bangkok - March 2025

Agenda

Motivation & Scope

Brief Overview

Open Issues & Discussion Topics

Next steps

Why a new version?

Lots of changes to DNS and transports since RFC 8305!

- QUIC standardization
- More IPv6-only networks and mechanisms
- SVCB / HTTPS records (address hints, priorities, ALPN)
- Encrypted Client Hello

Goals of this document

1. Incorporate new technologies since RFC 8305 into new version of the Happy Eyeballs algorithm

2. Review and update configurable values in the algorithm based on realistic network scenarios and performance data measured in deployed networks

Scope of this document

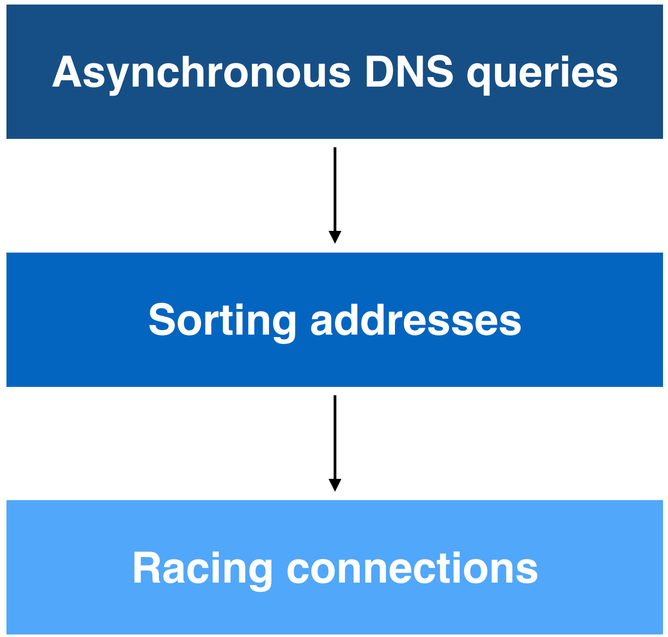
In scope

- Client-side algorithm for connection establishment
- Starts with a fully-qualified domain name (FQDN) and ends with a single established connection to a server

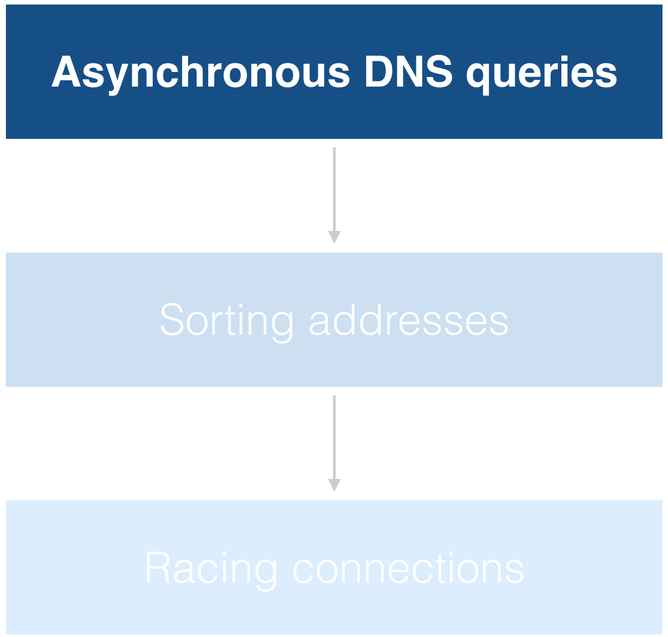
Out of scope

- Detecting and reporting misconfigured and broken networks (another document, please!)
- Specific mechanisms for "scenarios with multiple network links, or pools of multiple connections"

Updated Algorithm



Updated Algorithm



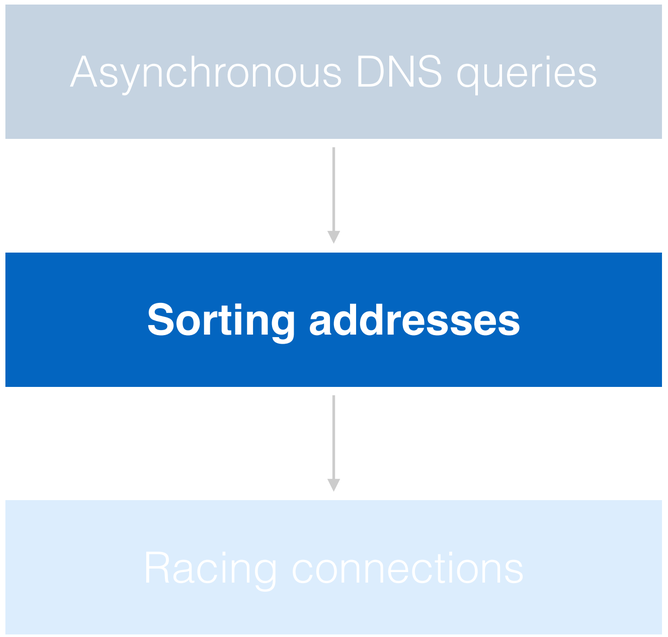
Prefer IPv6 DNS resolver addresses

Query **SVCB** / AAAA / A RRs in parallel (**SVCB** first)

Act on AAAA responses immediately, **if SVCB RRs not requested**

Wait up to 50ms for **SVCB and/or AAAA** if A comes back first

Updated Algorithm



Prefer ECH keys, if present

SVCB priorities, if present

Preferred ALPNs, if present

Discuss

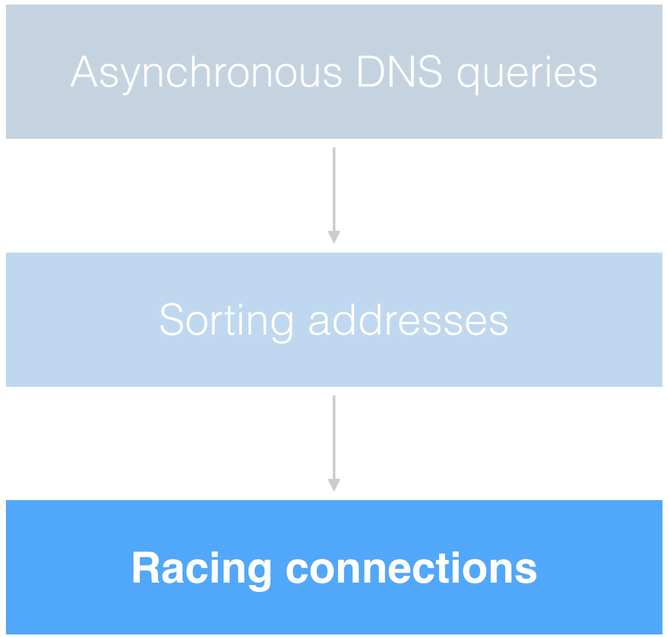
RFC 6724 address sorting

Historical round trip times

Place IPv6 at the start; two IPv6 addresses first if available

Use IPv4 first if historical data shows IPv6 brokenness

Updated Algorithm



Start attempts to addresses in parallel, staggered by a delay

Delay based on the TCP / **QUIC** retransmission timeout

Race ends when one connection completes

Racing through **full handshake (TCP / TLS / QUIC, etc.)**

Configurable Values

Resolution Delay
Time to wait for SVCB / AAAA response after receiving A response

Preferred Protocol Combination Count
Number of addresses of the preferred address family and protocol to be attempted before attempting next combination

Connection Attempt Delay, including Minimum and Maximum
Time to wait between connection attempts, in absence of RTT data

Last Resort Local Synthesis Delay
Time to wait after starting the last IPv6 attempt and before sending the A query

...and more?

Open Issues & Discussion Topics

Impacts of integrating SVCB

Requires sending new queries

- SVCB/HTTPS in initial flight, and extra service-specific A/AAAA queries

Incorporating IPv4/IPv6 hints

Sorting based on priorities (also applicable for SRV)

Preferences for ALPN / ECH

Impacts of integrating SVCB

Requires sending new queries

SVCB/HTTPS in initial flight, and extra service-specific A/AAAA queries

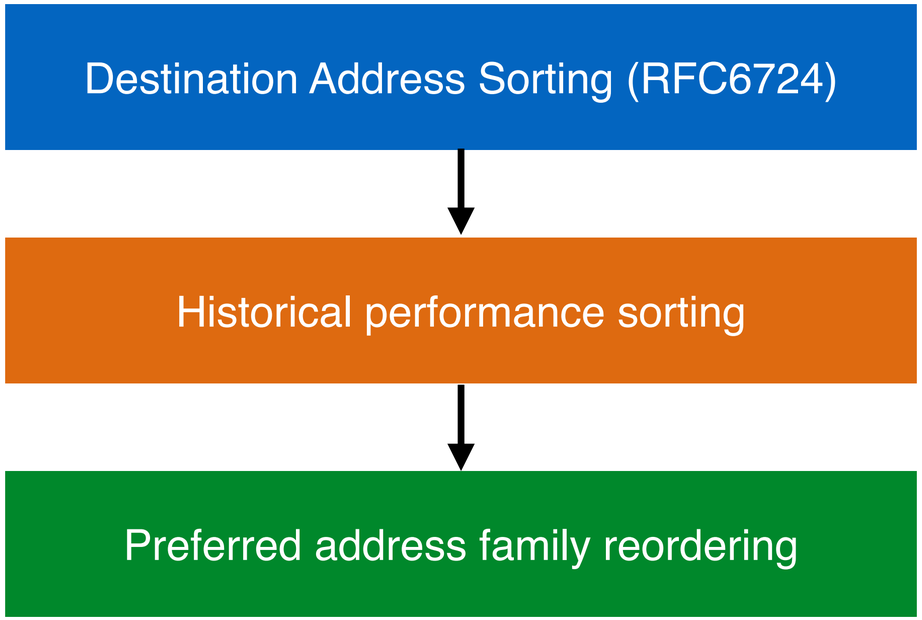
Incorporating IPv4/IPv6 hints

Sorting based on priorities (also applicable for SRV)

Preferences for ALPN / ECH

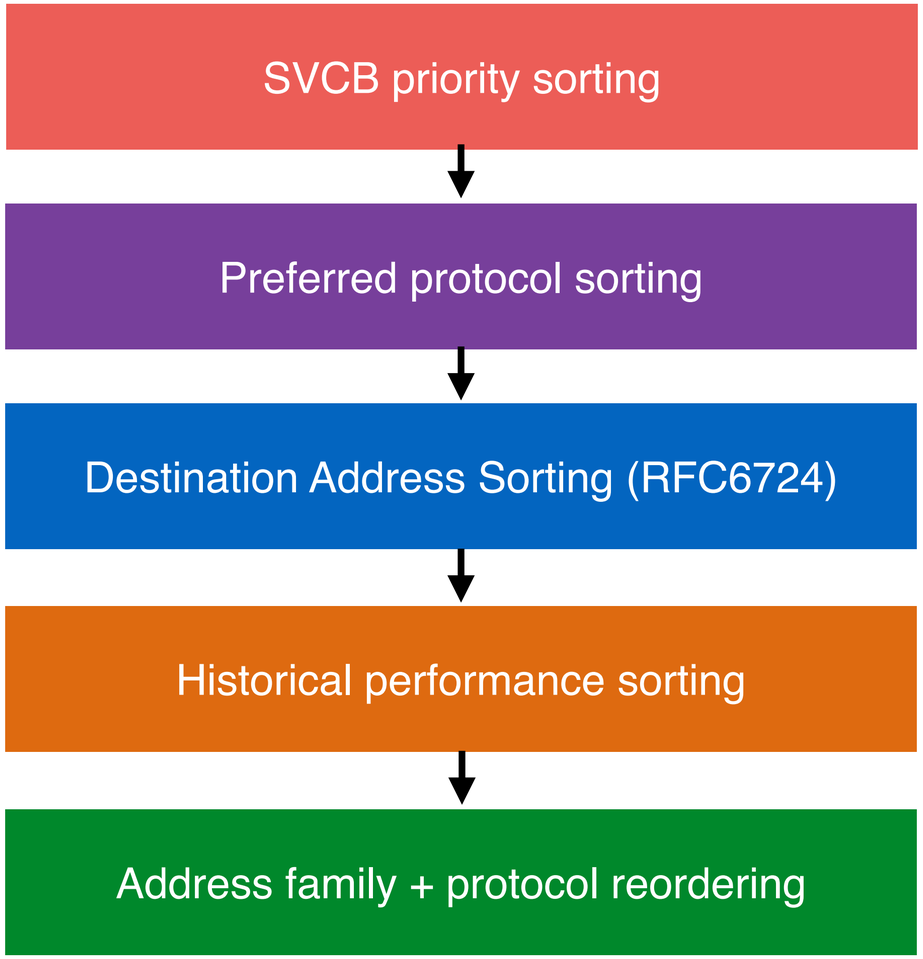
Impacts sorting algorithm

HEv2 sorting algorithm



(Put 2 IPv6 addresses at the top, followed by IPv4)

HEv3 sorting algorithm



Sorting order of operations

Should SVCB priority published by servers override local preferences around ALPN / ECH?

- Currently, the draft has the client rank or eliminate services based on its preferences; this conflicts with ECH-SVCB draft
- Since these values are only known due to SVCB, this seems reasonable to take the server preference

Sorting order of operations

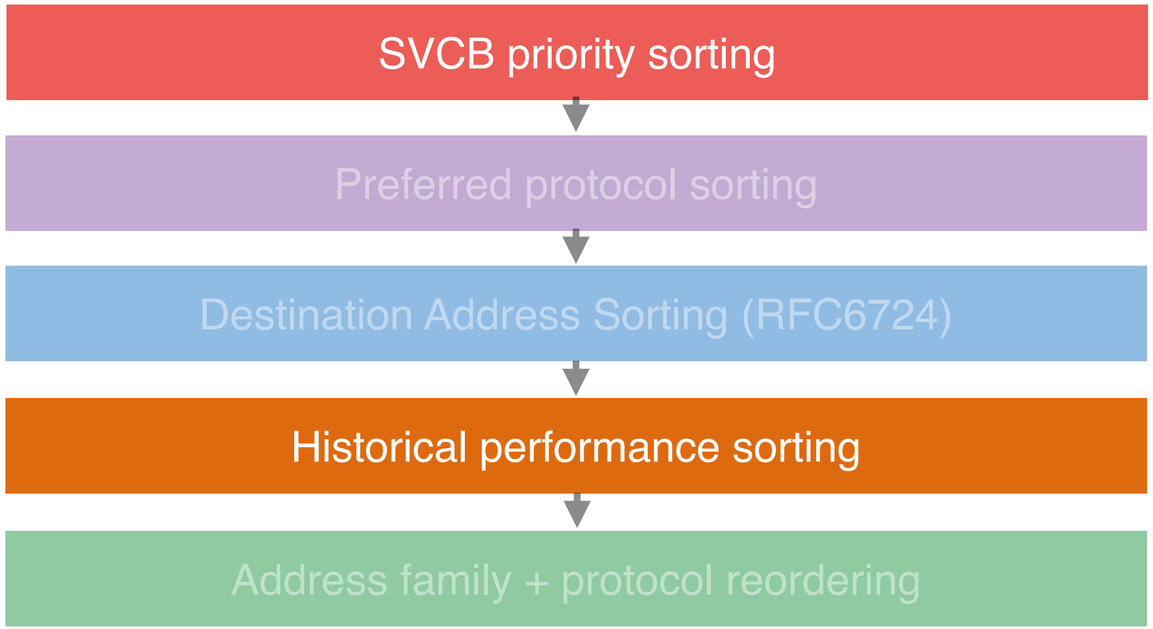
Should SVCB priority be able to override preferred address family sorting (original Happy Eyeballs)?

- Currently, the draft keeps the reordering from original HEv2
- How often will a particular service include only IPv4, while others in the same set include IPv6?

Sorting order of operations

How should SVCB priority interact with historical performance information?

Is a later sorting step able to reorder across SVCB priorities?



SVCB, ECH, and Happy Eyeballs

ECH-SVCB says that support for ECH SHOULD be consistent across all services in an RRSET — a mix of support and non-support can have downgrade attacks

If there *is* a mix, should a client using happy eyeballs reject the services, or sort the ECH-supporting service to the top?

Or should it let it be sorted as usual, potentially lower?

This is a matter of which party is responsible for the security properties of the connection

IPv6-only and IPv6-mostly networks

When NAT64 is present on IPv6-only networks, should we use the fact that an address is synthesized from IPv4 to rank it lower?

Handling DNS64 vs PREF64 vs both

- Since NAT64 might be present without DNS64, run an A query in addition to AAAA if there is a NAT64 prefix known (PREF64)

Optimistic DNS

"Optimistic DNS" is the practice of using expired DNS answers while waiting for new answers

Should we define this here as part of Happy Eyeballs?

Should we reference some other document instead, or is this only relevant within the scope of a Happy Eyeballs stack?

Next Steps

Should we adopt draft-pauly-happy-happyeyeballs in the HAPPY WG?

Are there any topics that need special breakouts?

Interims, design teams, data gathering

Should we arrange testing / hackathon setups for next meeting?