

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

draft-ietf-happy-happyeyeballs-v3-03

Tommy Pauly, David Schinazi, Nidhi Jaju, Kenichi Ishibashi
HAPPY - IETF 125 Shenzhen - March 2026

Agenda

Updates in -03

Open Issues & Pull Requests

Updates in -03

Client SHOULD wait for the TLS handshake ([#109](#))

Include SVCB/HTTPS records in Resolution Delay definition ([#111](#))

Specify Connection Attempt Delay for session resumption ([#114](#))

Address sorting should include all addresses, including already attempted ([#118](#))

Update implementation guidance on delay values ([#119](#))

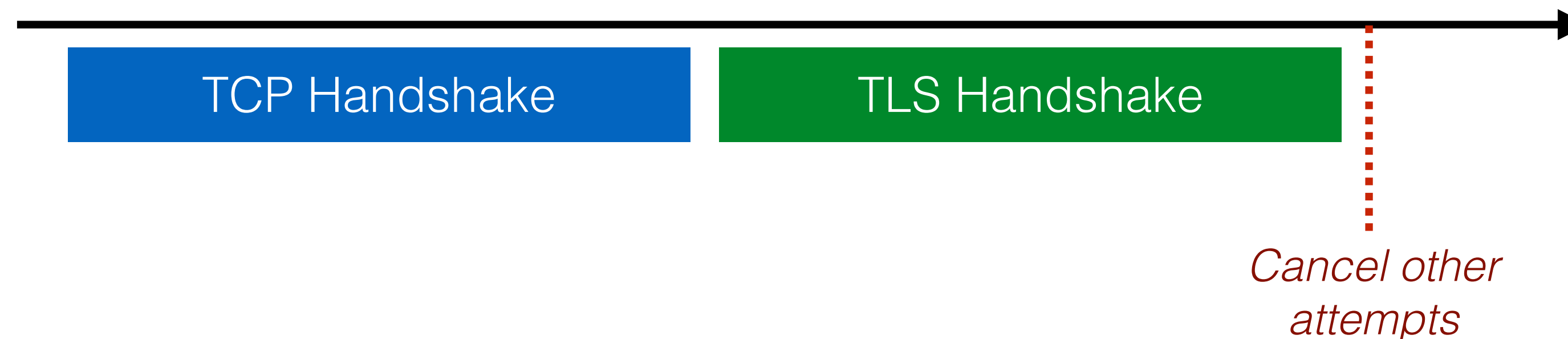
Update MTU issues ([#120](#))

Add text for returning to same network ([#121](#))

Client **SHOULD** wait for the TLS handshake (#109)

Discussed at IETF 124

clients ~~MAY~~ choose to **SHOULD** wait for the TLS handshake to successfully complete before cancelling other connection attempts



Specify Connection Attempt Delay for session resumption (#114)

Clarify that connection attempt delay is not changed for resumed sessions

Clients that support session resumption mechanisms (such as TLS session tickets or QUIC 0-RTT) SHOULD use the same Connection Attempt Delay for resumed connections as for new connections.

Update implementation guidance on delay values (#119)

Clarify that implementations are free to evolve delay values, but they **MUST** stay within the bounds established by the normative requirements in the document.

e.g. cannot go below the minimum delay value

Update MTU issues (#120)

Specified MTU considerations

For connections using TCP without TLS:

Issues where small handshake packets succeed, but large packets don't because of a small MTU being configured on the network

For connections using TLS:

Larger messages used in many TLS handshakes are often sufficient to ensure maximum-size TCP segments are deliverable

For QUIC connections:

Minimum MTU of 1200 bytes is guaranteed, but chance that larger values may not be available

Add text for returning to same network (#121)

Issue was raised that flushing historical data on network changes makes history virtually useless on mobile devices

Added text to mention that "if a client can reliably identify a previously attached network, it MAY retain and resume using historical data associated with that network upon reconnection."

Open Issues & Pull Requests

Open PRs

Rewrite Supporting IPv6-only networks section ([#94](#))

Open for a while, having a side conversation to make progress

Describe optimistic DNS ([#100](#))

No updates, leaning towards describing it separately

Refactor to separate requirements from algorithm ([#122](#))

New restructuring proposal to discuss

Rewrite Supporting IPv6-only networks section (#94)

Open PR from Ondřej — thanks!

- Scenarios: native IPv6, native IPv4, synthesized IPv6, CLAT
- How to treating synthesized IPv6 addresses? Preferred over IPv4?
- Curious Corner Case of IPv4-only VPNs on IPv6-only Network
 - 198.51.100.0/24 is routed to VPN
 - ipv4only.example.net (198.51.100.1) resolved to 64:ff9b::198.51.100.1
 - traffic sent to IPv6-only network (default) instead of VPN

Supporting IPv6-only networks section: Proposal

"PvD": a set of interfaces the connection might end up sending packets on

If the PvD has a global IPv6 address: MUST ask for AAAA

IPvD has a non-loopback IPv4 OR a NAT64 prefix discovered: MUST ask for A

- IPv4 answer: added to the candidate list
- IPv6 answer:
 - synthesized (contained PREF64): "un synthesize", add IPv4 to the candidate list (might need to synthesize again, Sec. 8.1, when connecting)
 - Non-synthesized: add IPv6 to the candidate list

Supporting IPv6-only networks: Example

DNS Responses: [2001:db8::1, 198.51.100.1, 64:ff9b::198.51.100.1]

NAT64 prefix detected: 64:ff9b::/96

- detection mechanism is out of scope
- the NAT64 prefix is specific to the interface used for the name resolution

Resulting candidate list: [2001:db8::1, 198.51.100.1]

Refactor to separate requirements from algorithm (#122)

Open PR from Ben — thanks!

- Large proposal for restructuring the document, based in part on a line from the abstract that is carried over from RFC 8305:

*This document specifies **requirements for algorithms** that reduce this user-visible delay and provides an **example algorithm**, referred to as "Happy Eyeballs".*

- What is the line between requirements, recommendations, and examples? And how should that impact structure?

Current

~~~

## 4. Hostname Resolution

- 4.1. Sending DNS Queries
- 4.2. Handling DNS Answers Asynchronously
- 4.3. Handling New Answers
- 4.4. Handling Multiple DNS Server Addresses

## 5. Grouping and Sorting Addresses

- 5.1. Grouping By Application Protocols and Security Requirements
- 5.2. Grouping By Service Priority
- 5.3. Sorting Destination Addresses Within Groups

## 6. Connection Attempts

- 6.1. Determining successful connection establishment
- 6.2. Handling Application Layer Protocol Negotiation
- 6.3. Dropping or Pending Connection Attempts

~~~

9. Summary of Configurable Values

PR 122

~~~

## 4. Hostname Resolution

- 4.1. Sending DNS Queries
- 4.2. Handling DNS Answers Asynchronously
- 4.3. Handling New Answers
- 4.4. Handling Multiple DNS Server Addresses

## 5. Filtering and Prioritizing Addresses

- 5.1. History-Driven Prioritization
- 5.2. When to Apply Application Preferences

## 6. Connection Establishment

- 6.1. Determining successful connection establishment
- 6.2. Connection attempts waiting for SVCB

~~~

8. Example: The Happy Eyeballs Algorithm

8.1. Configuration Parameters

- 8.1.1. System Parameters
- 8.2. Algorithm Parameters

8.2. Resolution

8.3. Grouping and Sorting Addresses

- 8.3.1. Grouping By Application Protocols and Security Requirements
- 8.3.2. Grouping By Service Priority
- 8.3.3. Sorting Destination Addresses Within Groups

8.4. Connection Attempts

- 8.4.1. Handling Application Layer Protocol Negotiation

Refactor to separate requirements from algorithm (#122)

Some subjective comments:

- "Example algorithm" is only ever mentioned in the abstract/intro, but nothing else in the document treats it as an example. This is largely a "leftover".
- Very little of Happy Eyeballs is truly normative — it is a best practice to get good results, but doesn't impact interoperability.
- Splitting the sections could make the three-step structure of the algorithm more repetitive
 - ➔ If the goal is to highlight which parts of the algorithm are **normative**, we can summarize **requirements** like we do for tunable values
 - ➔ Alternatively, we could segment the normative requirements within each step from the implementation recommendations, but leave their flow intact

Thank you!