Discovering WebSockets over HTTP/2 and HTTP/3 Design Team Report

IETF 117 – San Francisco – 2023-07

Lucas Pardue

Recap

- This topic has been rumbling along for ~2 years
- Presented at IETF 116 - outcome to form a design team
  - Representatives of client and server side
    - Bhavana Shankar
    - Dragana Damjanovic
    - Emile Stephan
    - Eric Kinnear
    - Felipe Gasper
    - Glenn Strauss
    - Ishiban Kenichi
    - Kershaw Jang
    - Momoka Yamamoto
    - Nidhi Jaju
    - Valentin Gosu
    - Victor Vasiliev

Key points

A client sees wss:// and has to make bet on connection to open a WebSocket

Using an HTTP/2 or HTTP/3 connection has advantages

But betting wrong means:

- latency impact on client side,
- wasted connection on server side

We want to increase the chances that client can bet on the correct horse

Picking between H1 or H2, is different from picking H3

We want to solve the case for WebSockets. MASQUE and WebTrans out of scope

Properties of a solution

Priority ⏬	Property	Must Have / Nice to have
1	Be able to establish websocket support without additional roundtrips	Must
2	The server needs to be able to roll back support for WebSockets without huge problems	Must
3	Support for WebSockets cannot be changed during an active connection	Must
4	Support needs to be Hop-by-Hop	Must
5	Logical consistency across HTTP/2 and HTTP/3	Nice
6	Determine status of support after a connection is established	Nice
7	Downgrade protection	Need more clarification
8	Fits well with existing negotiation patterns	Need more clarification

Potential solutions

1. DNS (SVCB / HTTPS)
    a. E.g. https://datatracker.ietf.org/doc/html/draft-damjanovic-websockets-https-rr
2. SETTINGs
    a. E.g. https://datatracker.ietf.org/doc/draft-momoka-httpbis-settings-enable-websockets/
3. ALPN
4. OPTIONS request

Scored, weighted, and ranked

Property	Must / Nice	DNS	SETTING	ALPN	OPTIONS
Be able to establish websocket support without additional roundtrips	Must	4	8	6	2
The server needs to be able to roll back support for the feature without huge problems	Must	2	6	4	8
Support for WebSockets cannot be changed during an active connection	Must	8	8	8	2
Support needs to be Hop-by-Hop	Must	8	8	8	8
Logical consistency across HTTP/2 and HTTP/3	Nice	3	4	2	4
Determine status of support after a connection is established	Nice	4	4	4	1

Total	38	29	32	25

Results analysis

OPTIONS is a clear negative outlier, discount it

DNS is on top

Not much delta between DNS, SETTINGS and ALPN

So, explore cons of remaining proposals

Constraints, caveats etc

- **DNS**
  - Not all clients can access SCVB / HTTPS *today*
    - "Tomorrow" that story could be different
- **SETTINGS**
  - Applies to a connection. Edge cases arise with connection coalescing. Especially origins that can be authoritatively served together but have different WebSocket configs.
  - Shifting goalposts on SETTINGS changes preconditions and expectations => endpoints deployed today that don't update could see a drop in H2 WebSockets rates
- **ALPN**
  - No concrete proposal
  - Unclear if helps/solves selection of H2 or H3

Many enter, one leaves (?)

There is no perfect solution

But given the common pain points in DT participants

A solution based on DNS SVCB/HTTPS seems like the most pragmatic path forward

DNS snags *today,* could be resolved in parallel to WebSocket work

While SETTINGS might ease pain *today*

- DNS & SETTINGS is not super convincingly better
- DNS & “I reject your bootstrap request” is close
- SETTING has edges we might regret later

Design Team Report Summary

Alignment on key issues, want to avoid costs … using interoperable solution

Ranked design possibilities against prioritized properties of any solution

Juxtapose possibilities against reality - can a solution be done in reasonable time with reasonable work

All things considered…

We think using a **DNS-based** solution is the most pragmatic solution to improving WebSocket discovery for clients and servers

Next steps

Accept our recommendation?

Wind down the design team and move discussion back to WG?

Backup slide on coalescing issue

- example.com and other.net
- Owned by the same entity and share a certificate that includes both in the SAN.
- DNS configured to present the same IP addresses for both.
- The client can satisfy its safety checks and coalesces

Now imagine that example.com has H2 WebSockets enabled and other.net does not. Fail and fallback.

SETTING is connection global. An H2 WebSocket SETTING cannot accommodate differential coalesced origin properties.

DNS **can** accommodate differential coalesced origin properties. Client could avoid fall and fallback latency.