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
    - David Benjamin
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

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

1. DNS (SVCB / HTTPS)

2. SETTINGs

3. ALPN

4. OPTIONS request
# Scored, weighted, and ranked

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

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
  ○ Server-side selection making isn’t free
    ■ Building required flexibility to ALPN probably requires TLS library API re-thinking
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.