

HTTP/3 Open Issues



Old and Moldy

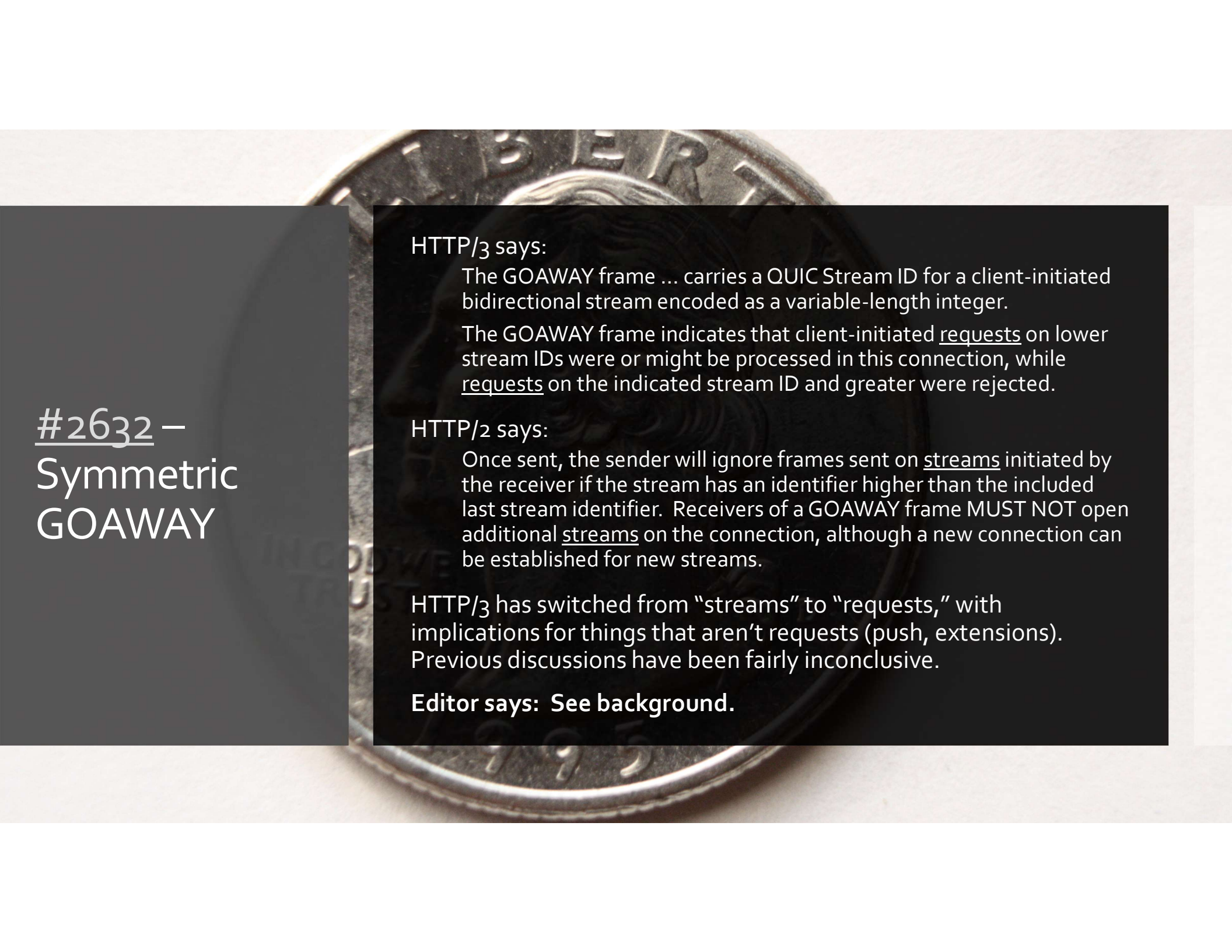
#253 & #2223 – Authority, Coalescing, and Alt-Svc

Spec says:

Once a connection exists to a server endpoint, this connection MAY be reused for requests with multiple different URI authority components. The client MAY send any requests for which the client considers the server authoritative.

An authoritative HTTP/3 endpoint is typically discovered because the client has received an Alt-Svc record from the request's origin which nominates the endpoint as a valid HTTP Alternative Service for that origin. As required by [RFC7838], clients **MUST** check that the nominated server can present a valid certificate for the origin before considering it authoritative. **Clients MUST NOT assume that an HTTP/3 endpoint is authoritative for other origins without an explicit signal.**

Editor says: Still waiting on http-core discussion of authority, which is now tracked by http-core/#237.



#2632 – Symmetric GOAWAY

HTTP/3 says:

The GOAWAY frame ... carries a QUIC Stream ID for a client-initiated bidirectional stream encoded as a variable-length integer.

The GOAWAY frame indicates that client-initiated requests on lower stream IDs were or might be processed in this connection, while requests on the indicated stream ID and greater were rejected.

HTTP/2 says:

Once sent, the sender will ignore frames sent on streams initiated by the receiver if the stream has an identifier higher than the included last stream identifier. Receivers of a GOAWAY frame **MUST NOT** open additional streams on the connection, although a new connection can be established for new streams.

HTTP/3 has switched from “streams” to “requests,” with implications for things that aren’t requests (push, extensions). Previous discussions have been fairly inconclusive.

Editor says: See background.



SETTINGSeS

Current Landscape

Spec requires that SETTINGS:

- MUST be sent as the first frame of the control stream
- MUST be sent as soon as the transport is ready to send data
- MUST NOT depend on having received anything from the peer

Until SETTINGS is received:

- In 1-RTT, clients and servers assume defaults
- In 0-RTT:
 - Clients SHOULD remember values *or* MAY assume defaults
 - Servers MUST NOT accept 0-RTT if old settings can't be recovered

Settings as not-a-frame?

SETTINGS is an unusual frame:

- Sent as the first frame as control stream
- Can never be sent again

Starts to sound more like a required blob than a frame type.

Parallels:

- Push ID (prefix at beginning of push streams)
- Transport parameters (blob in handshake)

Control stream header? (#2783)

Advantages:

- Functionally equivalent to what we have now in all the good ways
- Some error conditions become impossible
- Removes the oddball frame, makes push less oddball
 - Beginning of unidirectional streams is already special

Disadvantages:

- Functionally equivalent to what we have now in all the bad ways
- Existing stream prefix is not variable-length

Settings in TPs?

Putting settings in Transport Parameters is appealing for several reasons:

- Eliminates a special-case frame type
- Guarantees receipt before any 1-RTT data
 - Makes extensibility story much stronger
 - Removes race conditions around coalesced packets
- Consistent with QUIC's "ethos" of combining handshakes / crossing layers

Ugly for several reasons, too:

- Client TPs are unencrypted
- Clients can offer many ALPN tokens; each needs settings
- No other TP requires knowledge of the peer's transport parameters or handshake to send

How, then, shall we
send settings?

HTTP/2 Comparison

#2787 – H2 vs. H3 Concurrency

HTTP/2 limits “concurrent requests” (100 recommended). HTTP/3 relies on QUIC’s `initial_max_bidi` (100 recommended) / `MAX_STREAMS`. Are these equivalent?

- When is a stream no longer active?
 - In HTTP/2, when server sends `END_STREAM`
 - In QUIC, when data + FIN is ACK’d
- When does a client know it can send a new request?
 - In HTTP/2, when it receives the `END_STREAM`
 - In QUIC, when it receives a `MAX_STREAMS`

This implies at least 1 extra RTT before a concurrency “slot” becomes available again.

Should HTTP/3 recommend a value greater than 100 to keep parity? Should the guidance be more loosely framed than max streams?

See also #2956 – is this “recommended” or “RECOMMENDED”?

#2817 – H2-to-H3 Error Translation

An intermediary that converts between HTTP/2 and HTTP/3 is likely to end up needing to propagate errors between the protocols. We don't have much guidance for this.

Some points from the mailing list discussion on this:

- Suspect an intermediary that receives error codes from the upstream would convert many of these to 502 errors
- The REJECTED error code is a possible exception to the above behaviour
- It may be preferable to propagate "temporary" errors (REFUSED_STREAM) and CONNECT_ERROR (the use case related to websocket usage, although there is no formal HTTP/3 websockets yet).

Structural Changes

#2911 – Separate-er error spaces

In HTTP/2, any stream error could be “promoted” to a connection error by an implementation for any reason.

With recent changes, almost all errors are either mandatory connection errors, or errors that (probably) don’t make sense to promote to connection errors.

Should we divide the stream-closure and connection-closure error code spaces and remove this freedom to promote?

Editor says: I don’t see the benefit, personally.

Unstated Assumptions

#2963 – Early response without EARLY_RESPONSE

Spec says:

A server can send a complete response prior to the client sending an entire request if the response does not depend on any portion of the request that has not been sent and received. When this is true, a server MAY abort reading the request stream with error code HTTP_EARLY_RESPONSE, send a complete response, and cleanly close the sending part of the stream.

Do we need to explicitly state that if the server does *not* abort reading, the client MUST continue sending the request body even if it sees a response begin?

What about if the response is “complete”?

#3061 – Missing QUIC Version Hints

In one draft, called draft-00 by some, an Alt-Svc parameter arose....

- Version hints save a round-trip by suggesting what versions the server might support.
- If you don't get a hint, you just guess.
- If the hint or your guess is wrong, you just VN. No big deal.

Except that the Wheel of Consensus has turned, eliminating Version Negotiation as a thing.

- Is this parameter still useful, or should we save it until there are multiple QUIC versions?
- Do we need to explicitly discuss what version number to select in its absence?
- Are the quotes necessary when only a single version is supported? (#3063)