Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

R. Shade
M. Warres
Google

July 8, 2016

HTTP/2 Semantics Using The QUIC Transport Protocol
draft-shade-quic-http2-mapping-00

Abstract

The QUIC transport protocol has several features that are desirable in a transport for HTTP/2, such as stream multiplexing, per-stream flow control, and low-latency connection establishment. This document describes a mapping of HTTP/2 semantics over QUIC. Specifically, this document identifies HTTP/2 features that are subsumed by QUIC, and describes how the other features can be implemented atop QUIC.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Table of Contents

1. Introduction

The QUIC transport protocol has several features that are desirable in a transport for HTTP/2, such as stream multiplexing, per-stream flow control, and low-latency connection establishment. This document describes a mapping of HTTP/2 semantics over QUIC. Specifically, this document identifies HTTP/2 features that are subsumed by QUIC, and describes how the other features can be implemented atop QUIC.

QUIC is described in [draft-hamilton-quic-transport-protocol]. For a full description of HTTP/2, see [RFC 7540].

2. QUIC advertisement

A server advertises that it can speak HTTP/2-over-QUIC via the Alt-Svc HTTP response header. It does so by including the header in any response sent over a non-QUIC (e.g. HTTP/2 over TLS) connection:

Alt-Svc: quic=":443"

In addition, the list of QUIC versions supported by the server can be specified by the v= parameter. For example, if a server supported both version 33 and 34 it would specify the following header:

```
Alt-Svc: quic=":443"; v="34,33"
```

On receipt of this header, a client may attempt to establish a QUIC connection on port 443 and, if successful, send HTTP/2 requests using the mapping described in this document.

Connectivity problems (e.g. firewall blocking UDP) may result in QUIC connection establishment failure, in which case the client should gracefully fallback to HTTP/2-over-TLS/TCP.

3. Connection establishment

HTTP/2-over-QUIC connections are established as described in [draft-hamilton-quic-transport-protocol]. The QUIC crypto handshake MUST use TLS [draft-thomson-quic-tls].

While connection-level options pertaining to the core QUIC protocol are set in the initial crypto handshake [Combined Crypto and Transport Handshake], HTTP/2-specific settings are conveyed in the HTTP/2 SETTINGS frame. After the QUIC connection is established, an HTTP/2 SETTINGS frame may be sent as the initial frame of the QUIC headers stream (StreamID 3, See [Stream Mapping]). As in HTTP/2, additional SETTINGS frames may be sent mid-connection by either endpoint.

TODO: decide whether to acknowledge receipt of SETTINGS through empty SETTINGS frames with ACK bit set, as in HTTP/2, or rely on transport-level acknowledgment.

Some transport-level options that HTTP/2-over-TCP specifies via the SETTINGS frame are superseded by QUIC transport parameters in HTTP/2-over-QUIC. Below is a listing of how each HTTP/2 SETTINGS parameter is mapped:

- SETTINGS_HEADER_TABLE_SIZE
  - Sent in HTTP/2 SETTINGS frame.
- SETTINGS_ENABLE_PUSH
  - Sent in HTTP/2 SETTINGS frame [TBD, currently set using QUIC "SPSH" connection option]
- SETTINGS_MAX_CONCURRENT_STREAMS

    * QUIC requires the maximum number of incoming streams per connection to be specified in the initial crypto handshake, using the "MSPC" tag. Specifying SETTINGS_MAX_CONCURRENT_STREAMS in the HTTP/2 SETTINGS frame is an error.

- SETTINGS_INITIAL_WINDOW_SIZE

    * QUIC requires both stream and connection flow control window sizes to be specified in the initial crypto handshake, using the "SFCW" and "CFCW" tags, respectively. Specifying SETTINGS_INITIAL_WINDOW_SIZE in the HTTP/2 SETTINGS frame is an error.

- SETTINGS_MAX_FRAME_SIZE

    * This setting has no equivalent in QUIC. Specifying it in the HTTP/2 SETTINGS frame is an error.

- SETTINGS_MAX_HEADER_LIST_SIZE

    * Sent in HTTP/2 SETTINGS frame.

As with HTTP/2-over-TCP, unknown SETTINGS parameters are tolerated but ignored. SETTINGS parameters are acknowledged by the receiving peer, by sending an empty SETTINGS frame in response with the ACK bit set.

4. Sending a request on an HTTP/2-over-QUIC connection

A high level overview of sending an HTTP/2 request on an established QUIC connection is as follows, with further details in later sections of this document. A client should first encode any HTTP headers using HPACK [RFC7541] and frame them as HTTP/2 HEADERS frames. These are sent on StreamID 3 (see [Stream Mapping]). The exact layout of the HEADERS frame is described in Section 6.2 of [RFC7540]. No HTTP/2 padding is required: QUIC provides a PADDING frame for this purpose.

While HEADERS are sent on stream 3, the mandatory stream identifier in each HEADERS frame indicates the QUIC StreamID on which a corresponding request body may be sent. If there is no non-header data, the specified QUIC data stream will never be used.

4.1. Terminating a stream

A stream can be terminated in one of three ways:

- the request/response is headers only, in which case a HEADERS frame with the END_STREAM bit set ends the stream specified in the HEADERS frame

- the request/response has headers and body but no trailing headers, in which case the final QUIC STREAM frame will have the FIN bit set

- the request/response has headers, body, and trailing headers, in which case the final QUIC STREAM frame will not have the FIN bit set, and the trailing HEADERS frame will have the END_STREAM bit set

(TODO: Describe mapping of HTTP/2 stream state machine to QUIC stream state machine.)

5. Writing data to QUIC streams

A QUIC stream provides reliable in-order delivery of bytes, within that stream. On the wire, data is framed into QUIC STREAM frames, but this framing is invisible to the HTTP/2 layer. A QUIC receiver buffers and orders received STREAM frames, exposing the data contained within as a reliable byte stream to the application.

Bytes written to Stream 3 must be HTTP/2 HEADERS frames (or other HTTP/2 non-data frames), whereas bytes written to data streams should simply be request or response bodies. No further framing is required by HTTP/2 (i.e. no HTTP/2 DATA frames are used).

If data arrives on a data stream before the corresponding HEADERS have arrived on stream 3, then the data is buffered until the HEADERS arrive.

6. Stream Mapping

When HTTP/2 headers and data are sent over QUIC, the QUIC layer handles most of the stream management. HTTP/2 StreamIDs are replaced by QUIC StreamIDs. HTTP/2 does not need to do any explicit stream framing when using QUIC---data sent over a QUIC stream simply consists of HTTP/2 headers or body. Requests and responses are considered complete when the QUIC stream is closed in the corresponding direction.

Like HTTP/2, QUIC uses odd-numbered StreamIDs for client initiated streams, and even-numbered IDs for server initiated (i.e. server push) streams. Unlike HTTP/2 there are a couple of reserved (or dedicated) StreamIDs in QUIC.

6.1. Reserved Streams

StreamID 1 is reserved for crypto operations (the handshake, crypto config updates), and MUST NOT be used for HTTP/2 headers or body, see [core protocol doc]. StreamID 3 is reserved for sending and receiving HTTP/2 HEADERS frames. Therefore the first client initiated data stream has StreamID 5.

There are no reserved server initiated StreamIDs, so the first server initiated (i.e. server push) stream has an ID of 2, followed by 4, etc.

6.1.1. Stream 3: headers

HTTP/2-over-QUIC uses HPACK header compression as described in [RFC7541]. HPACK was designed for HTTP/2 with the assumption of in-order delivery such as that provided by TCP. A sequence of encoded header blocks must arrive (and be decoded) at an endpoint in the same order in which they were encoded. This ensures that the dynamic state at the two endpoints remains in sync.

QUIC streams provide in-order delivery of data sent on those streams, but there are no guarantees about order of delivery between streams. To achieve in-order delivery of HEADERS frames in QUIC, they are all sent on the reserved Stream 3. Data (request/response bodies) which arrive on other data streams are buffered until the corresponding HEADERS arrive and are read out of Stream 3.

This does introduce head-of-line blocking: if the packet containing HEADERS for stream N is lost or reordered then stream N+2 cannot be processed until they it has been retransmitted successfully, even though the HEADERS for stream N+2 may have arrived.

Trailing headers (trailers) can also be sent on stream 3. These are sent as HTTP/2 HEADERS frames, but MUST have the END_STREAM bit set, and MUST include a ":final-offset" pseudo-header. Since QUIC supports out of order delivery, receipt of a HEADERS frame with the END_STREAM bit set does not guarantee that the entire request/response body has been fully received. Therefore, the extra ":final-offset" pseudo-header is included in trailing HEADERS frames to indicate the total number of body bytes sent on the corresponding data stream. This is used by the QUIC layer to determine when the full request has been received and therefore when it is safe to tear

down local stream state. The ":final-offset" pseudo header is stripped from the HEADERS before passing to the HTTP/2 layer.

6.1.2. Stream states

The mapping of HTTP/2-over-QUIC with potential out of order delivery of HEADERS frames results in some changes to the HTTP/2 stream state transition diagram [https://tools.ietf.org/html/rfc7540#section-5.1]. Specifically the transition from "open" to "half closed (remote)", and the transition from "half closed (local)" to "closed" takes place only when:

- the peer has explicitly ended the stream via either
  - an HTTP/2 HEADERS frame with END_STREAM bit set and, in the case of trailing headers, the :final-offset pseudo-header
  - or a QUIC stream frame with the FIN bit set.
- and the full request or response body has been received.

7. Stream Priorities

HTTP/2-over-QUIC uses the HTTP/2 priority scheme described in [RFC7540 Section 5.3]. In the HTTP/2 priority scheme, a given stream can be designated as dependent upon another stream, which expresses the preference that the latter stream (the "parent" stream) be allocated resources before the former stream (the "dependent" stream). Taken together, the dependencies across all streams in a connection form a dependency tree. The structure of the dependency tree changes as HTTP/2 HEADERS and PRIORITY frames add, remove, or change the dependency links between streams.

Implicit in this scheme is the notion of in-order delivery of priority changes (i.e., dependency tree mutations): since operations on the dependency tree such as reparenting a subtree are not commutative, both sender and receiver must apply them in the same order to ensure that both sides have a consistent view of the stream dependency tree. HTTP/2 specifies priority assignments in PRIORITY frames and (optionally) in HEADERS frames. To achieve in-order delivery of HTTP/2 priority changes in HTTP/2-over-QUIC, HTTP/2 PRIORITY frames, in addition to HEADERS frames, are also sent on reserved stream 3. The semantics of the Stream Dependency, Weight, E flag, and (for HEADERS frames) PRIORITY flag are the same as in HTTP/2-over-TCP.

Since HEADERS and PRIORITY frames are sent on a different stream than the STREAM frames for the streams they reference, they may be

delivered out-of-order with respect to the STREAM frames. There is no special handling for this--the receiver should simply assign resources according to the most recent stream priority information that it has received.

ALTERNATIVE DESIGN: if the core QUIC protocol implements priorities, then this document should map the HTTP/2 priorities scheme to that provided by the core protocol. This would likely involve prohibiting the sending of HTTP/2 PRIORITY frames and setting of the PRIORITY flag in HTTP/2 HEADERS frames, to avoid conflicting directives.

8. Flow Control

QUIC provides stream and connection level flow control, similar in principle to HTTP/2's flow control but with some implementation differences. As flow control is handled by QUIC, the HTTP/2 mapping need not concern itself with maintaining flow control state, or how/when to send flow control frames to the peer. The HTTP/2 mapping must not send HTTP/2 WINDOW_UPDATE frames.

The initial flow control window sizes (stream and connection) are communicated during the crypto handshake (see [Connection establishment]). Setting these values to the maximum size (2^31 - 1) effectively disables flow control.

Relatively small initial windows can be used, as QUIC will attempt to auto-tune the flow control windows based on usage. See [draft-hamilton-quic-transport-protocol] for more details.

9. Server Push

HTTP/2-over-QUIC supports HTTP/2 server push. During connection establishment, the client indicates whether or it is willing to receive server pushes via the SETTINGS_ENABLE_PUSH setting in the HTTP/2 SETTINGS frame (see [Connection Establishment]), which defaults to 1 (true).

As with server push for HTTP/2-over-TCP, the server initiates a server push by sending an HTTP/2 PUSH_PROMISE frame containing the StreamID of the stream to be pushed, as well as request header fields attributed to the request. The PUSH_PROMISE frame is sent on stream 3, to ensure proper ordering with respect to other HEADERS and non-data frames. Within the PUSH_PROMISE frame, the StreamID in the common HTTP/2 frame header indicates the associated (client-initiated) stream for the new push stream, while the Promised Stream ID field specifies the StreamID of the new push stream.

The server push response is conveyed in the same way as a non-server-push response, with response headers and (if present) trailers carried by HTTP/2 HEADERS frames sent on reserved stream 3, and response body (if any) sent via QUIC stream frames on the stream specified in the corresponding PUSH_PROMISE frame.

10. Error Codes

The HTTP/2 error codes defined in [RFC7540 Section 7] map to QUIC error codes as follows:

- NO_ERROR (0x0)
  - Maps to QUIC_NO_ERROR
- PROTOCOL_ERROR (0x1)
  - No single mapping?
- INTERNAL_ERROR (0x2)
  - QUIC_INTERNAL_ERROR? (not currently defined in core protocol spec)
- FLOW_CONTROL_ERROR (0x3)
  - QUIC_FLOW_CONTROL_RECEIVED_TOO_MUCH_DATA? (not currently defined in core protocol spec)
- SETTINGS_TIMEOUT (0x4)
  - ? (depends on whether we support SETTINGS acks)
- STREAM_CLOSED (0x5)
  - QUIC_STREAM_DATA_AFTER_TERMINATION
- FRAME_SIZE_ERROR (0x6)
  - QUIC_INVALID_FRAME_DATA
- REFUSED_STREAM (0x7)
  - ?
- CANCEL (0x8)
  - ?

- COMPRESSION_ERROR (0x9)
  - QUIC_DECOMPRESSION_FAILURE (not currently defined in core spec)
- CONNECT_ERROR (0xa)
  - ? (depends whether we decide to support CONNECT)
- ENHANCE_YOUR_CALM (0xb)
  - ?
- INADEQUATE_SECURITY (0xc)
  - QUIC_HANDSHAKE_FAILED, QUIC_CRYPTO_NO_SUPPORT
- HTTP_1_1_REQUIRED (0xd)

TODO: fill in missing error code mappings.

11. Other HTTP/2 frames

QUIC includes some features (e.g. flow control) which are also present in HTTP/2. In these cases the HTTP/2 mapping need not re-implement them. As a result some HTTP/2 frame types are not required when using QUIC, as they either are directly implemented in the QUIC layer, or their functionality is provided via other means. This section of the document describes these cases.

11.1. GOAWAY frame

QUIC has its own GOAWAY frame, and QUIC implementations may to expose the sending of a GOAWAY to the application. The semantics of sending a GOAWAY in QUIC are identical to HTTP/2: an endpoint sending a GOAWAY will continue processing open streams, but will not accept newly created streams.

QUIC's GOAWAY frame is described in detail in the [draft-hamilton-quic-transport-protocol].

11.2. PING frame

QUIC has its own PING frame, which is currently exposed to the application. QUIC clients send periodic PINGs to servers if there are no currently active data streams on the connection.

QUIC's PING frame is described in detail in the [draft-hamilton-quic-transport-protocol].

11.3. PADDING frame

There is no HTTP/2 padding in this mapping; padding is instead provided at the QUIC layer by including QUIC PADDING frames in a packet payload. An HTTP/2 over QUIC mapping should treat any HTTP/2 level padding as an error, to avoid any possibility of inconsistent flow control states between endpoints (e.g. client sends HTTP/2 padding, counts it against flow control, server ignores).

12. Normative References

[RFC2119] Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels", March 1997.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", May 2015.

[RFC7541] Peon, R. and H. Ruellan, "HPACK: Header Compression for HTTP/2", May 2015.

[draft-hamilton-quic-transport-protocol] Hamilton, R., Iyengar, J., Swett, I., and A. Wilk, "QUIC: A UDP-Based Multiplexed and Secure Transport", July 2016.

[draft-thomson-quic-tls] Thomson, M. and R. Hamilton, "Porting QUIC to TLS", March 2016.

[draft-iyengar-quic-loss-recovery] Iyengar, J. and I. Swett, "QUIC Loss Recovery and Congestion Control", July 2016.

Authors' Addresses

Robbie Shade
Google

Email: rjshade@google.com

Mike Warres
Google

Email: mpw@google.com