Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

A. Frindell
Facebook
E. Kinnear
Apple Inc.
V. Vasiliev
Google

4 March 2024

WebTransport over HTTP/3
draft-ietf-webtrans-http3-09

Abstract

WebTransport [OVERVIEW] is a protocol framework that enables clients constrained by the Web security model to communicate with a remote server using a secure multiplexed transport. This document describes a WebTransport protocol that is based on HTTP/3 [HTTP3] and provides support for unidirectional streams, bidirectional streams and datagrams, all multiplexed within the same HTTP/3 connection.

Note to Readers

Discussion of this draft takes place on the WebTransport mailing list (webtransport@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/search/?email_list=webtransport>.

The repository tracking the issues for this draft can be found at <https://github.com/ietf-wg-webtrans/draft-ietf-webtrans-http3/issues>. The web API draft corresponding to this document can be found at <https://w3c.github.io/webtransport/>.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

HTTP/3 [HTTP3] is a protocol defined on top of QUIC [RFC9000] that can multiplex HTTP requests over a QUIC connection. This document defines a mechanism for multiplexing non-HTTP data with HTTP/3 in a manner that conforms with the WebTransport protocol requirements and semantics[OVERVIEW]. Using the mechanism described here, multiple WebTransport instances can be multiplexed simultaneously with regular HTTP traffic on the same HTTP/3 connection.

1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document follows terminology defined in Section 1.2 of [OVERVIEW]. Note that this document distinguishes between a WebTransport server and an HTTP/3 server. An HTTP/3 server is the server that terminates HTTP/3 connections; a WebTransport server is an application that accepts WebTransport sessions, which can be accessed via an HTTP/3 server.

2. Protocol Overview

WebTransport servers in general are identified by a pair of authority value and path value (defined in [RFC3986] Sections 3.2 and 3.3 correspondingly).

When an HTTP/3 connection is established, the server sends a SETTINGS_WEBTRANSPORT_MAX_SESSIONS setting in order to indicate support for WebTransport over HTTP/3. This process also negotiates the use of additional HTTP/3 extensions.

WebTransport sessions are initiated inside a given HTTP/3 connection by the client, who sends an extended CONNECT request [RFC8441]. If the server accepts the request, a WebTransport session is established. The resulting stream will be further referred to as a _CONNECT stream_, and its stream ID is used to uniquely identify a given WebTransport session within the connection. The ID of the CONNECT stream that established a given WebTransport session will be further referred to as a _Session ID_.

After the session is established, the peers can exchange data using the following mechanisms:

* A client can create a bidirectional stream and transfer its ownership to WebTransport by providing a special signal in the first bytes.

* A server can create a bidirectional stream and transfer its ownership to WebTransport by providing a special signal in the first bytes..

* Both client and server can create a unidirectional stream using a special stream type.

* A datagram can be sent using HTTP Datagrams [HTTP-DATAGRAM].

A WebTransport session is terminated when the CONNECT stream that created it is closed.

3. Session Establishment

3.1. Establishing a Transport-Capable HTTP/3 Connection

In order to indicate support for WebTransport, the server MUST send a SETTINGS_WEBTRANSPORT_MAX_SESSIONS value greater than "0" in its SETTINGS frame. The default value for the SETTINGS_WEBTRANSPORT_MAX_SESSIONS parameter is "0", meaning that the endpoint is not willing to receive any WebTransport sessions. Note that the client does not need to send any value to indicate support for WebTransport; clients indicate support for WebTransport by using the "webtransport" upgrade token in CONNECT requests establishing WebTransport sessions (see Section 8.1).

The client MUST NOT send a WebTransport request until it has received the setting indicating WebTransport support from the server.

[[RFC editor: please remove the following paragraph before publication.]]

For draft verisons of WebTransport only, the server MUST NOT process any incoming WebTransport requests until the client settings have been received, as the client may be using a version of the WebTransport extension that is different from the one used by the server.

Because WebTransport over HTTP/3 requires support for HTTP/3 datagrams and the Capsule Protocol, both the client and the server MUST indicate support for HTTP/3 datagrams by sending a

SETTINGS_H3_DATAGRAM value set to 1 in their SETTINGS frame (see Section 2.1.1 of [HTTP-DATAGRAM]). Servers should also note that CONNECT requests to establish new WebTransport sessions, in addition to other messages, may arrive before this SETTING is received (see Section 4.5).

WebTransport over HTTP/3 also requires support for QUIC datagrams. To indicate support, both the client and the server MUST send a max_datagram_frame_size transport parameter with a value greater than 0 (see Section 3 of [QUIC-DATAGRAM]).

3.2. Extended CONNECT in HTTP/3

[RFC8441] defines an extended CONNECT method in Section 4, enabled by the SETTINGS_ENABLE_CONNECT_PROTOCOL setting. That setting is defined for HTTP/3 by [RFC9220]. A server supporting WebTransport over HTTP/3 MUST send both the SETTINGS_WEBTRANSPORT_MAX_SESSIONS setting with a value greater than "0" and the SETTINGS_ENABLE_CONNECT_PROTOCOL setting with a value of "1". To use WebTransport over HTTP/3, clients MUST send the SETTINGS_ENABLE_CONNECT_PROTOCOL setting with a value of "1".

3.3. Creating a New Session

As WebTransport sessions are established over HTTP/3, they are identified using the https URI scheme ([HTTP], Section 4.2.2).

In order to create a new WebTransport session, a client can send an HTTP CONNECT request. The :protocol pseudo-header field ([RFC8441]) MUST be set to webtransport. The :scheme field MUST be https. Both the :authority and the :path value MUST be set; those fields indicate the desired WebTransport server. If the WebTransport session is coming from a browser client, an Origin header [RFC6454] MUST be provided within the request; otherwise, the header is OPTIONAL.

Upon receiving an extended CONNECT request with a :protocol field set to webtransport, the HTTP/3 server can check if it has a WebTransport server associated with the specified :authority and :path values. If it does not, it SHOULD reply with status code 404 (Section 15.5.5 of [HTTP]). When the request contains the Origin header, the WebTransport server MUST verify the Origin header to ensure that the specified origin is allowed to access the server in question. If the verification fails, the WebTransport server SHOULD reply with status code 403 (Section 15.5.4 of [HTTP]). If all checks pass, the WebTransport server MAY accept the session by replying with a 2xx series status code, as defined in Section 15.3 of [HTTP].

From the client’s perspective, a WebTransport session is established when the client receives a 2xx response. From the server’s perspective, a session is established once it sends a 2xx response.

The server may reply with a 3xx response, indicating a redirection (Section 15.4 of [HTTP]). The user agent MUST NOT automatically follow such redirects, as the client could potentially already have sent data for the WebTransport session in question; it MAY notify the client about the redirect.

Clients cannot initiate WebTransport in 0-RTT packets, as the CONNECT method is not considered safe (see Section 10.9 of [HTTP3]). However, WebTransport-related SETTINGS parameters may be retained from the previous session as described in Section 7.2.4.2 of [HTTP3]. If the server accepts 0-RTT, the server MUST NOT reduce the limit of maximum open WebTransport sessions from the one negotiated during the previous session; such change would be deemed incompatible, and MUST result in a H3_SETTINGS_ERROR connection error.

The webtransport HTTP Upgrade Token uses the Capsule Protocol as defined in [HTTP-DATAGRAM]. The Capsule Protocol is negotiated when the server sends a 2xx response. The capsule-protocol header field Section 3.4 of [HTTP-DATAGRAM] is not required by WebTransport and can safely be ignored by WebTransport endpoints.

3.4. Subprotocol Negotiation

WebTransport over HTTP/3 offers a subprotocol negotiation mechanism, similar to TLS Application-Layer Protocol Negotiation Extension (ALPN) [RFC7301]; the intent is to simplify porting pre-existing protocols that use QUIC and rely on this functionality.

The user agent MAY include a WebTransport-Subprotocols-Available header field in the CONNECT request, enumerating the possible subprotocols. If the server receives such a header, it MAY include a WebTransport-Subprotocol field in a successful (2xx) response. If it does, the server SHALL include a single subprotocol from the client’s list in that field. Servers MAY reject the request if the client did not include a suitable subprotocol.

Both WebTransport-Subprotocols-Available and WebTransport-Subprotocol are Structured Fields [RFC8941]. WebTransport-Subprotocols-Available is a List of Tokens, and WebTransport-Subprotocol is a Token. The token in the WebTransport-Subprotocol response header field MUST be one of the tokens listed in WebTransport-Subprotocols-Available of the request. The semantics of individual token values is determined by the WebTransport resource in question, and are not registered in IANA’s "ALPN Protocol IDs" registry.

3.5. Limiting the Number of Simultaneous Sessions

This document defines a SETTINGS_WEBTRANSPORT_MAX_SESSIONS parameter that allows the server to limit the maximum number of concurrent WebTransport sessions on a single HTTP/3 connection. The client MUST NOT open more sessions than indicated in the server SETTINGS parameters. The server MUST NOT close the connection if the client opens sessions exceeding this limit, as the client and the server do not have a consistent view of how many sessions are open due to the asynchronous nature of the protocol; instead, it MUST reset all of the CONNECT streams it is not willing to process with the HTTP_REQUEST_REJECTED status defined in [HTTP3].

Just like other HTTP requests, WebTransport sessions, and data sent on those sessions, are counted against flow control limits. This document does not introduce additional mechanisms for endpoints to limit the relative amount of flow control credit consumed by different WebTransport sessions, however servers that wish to limit the rate of incoming requests on any particular session have alternative mechanisms:

* The HTTP_REQUEST_REJECTED error code defined in [HTTP3] indicates to the receiving HTTP/3 stack that the request was not processed in any way.

* HTTP status code 429 indicates that the request was rejected due to rate limiting [RFC6585]. Unlike the previous method, this signal is directly propagated to the application.

3.6. Prioritization

WebTransport sessions are initiated using extended CONNECT. While Section 11 of [RFC9218] describes how extensible priorities can be applied to data sent on a CONNECT stream, WebTransport extends the types of data that are exchanged in relation to the request and response, which requires additional considerations.

WebTransport CONNECT requests and responses MAY contain the Priority header field (Section 5 of [RFC9218]); clients MAY reprioritize by sending PRIORITY_UPDATE frames (Section 7 of [RFC9218]). In extension to [RFC9218], it is RECOMMENDED that clients and servers apply the scheduling guidance in both Section 9 of [RFC9218] and Section 10 of [RFC9218] for all data that they send in the enclosing WebTransport session, including Capsules, WebTransport streams and datagrams. WebTransport does not provide any priority signaling mechanism for streams and datagrams within a WebTransport session; such mechanisms can be defined by application protocols using WebTransport. It is RECOMMENDED that such mechanisms only affect scheduling within a session and not scheduling of other data on the same HTTP/3 connection.

The client/server priority merging guidance given in Section 8 of [RFC9218] also applies to WebTransport session. For example, a client that receives a response Priority header field could alter its view of a WebTransport session priority and alter the scheduling of outgoing data as a result.

Endpoints that prioritize WebTransport sessions need to consider how they interact with other sessions or requests on the same HTTP/3 connection.

4. WebTransport Features

WebTransport over HTTP/3 provides the following features described in [OVERVIEW]: unidirectional streams, bidirectional streams and datagrams, initiated by either endpoint. Protocols designed for use with WebTransport over HTTP/3 are constrained to these features. The Capsule Protocol is an implementation detail of WebTransport over HTTP/3 and is not a WebTransport feature.

Session IDs are used to demultiplex streams and datagrams belonging to different WebTransport sessions. On the wire, session IDs are encoded using the QUIC variable length integer scheme described in [RFC9000].

The client MAY optimistically open unidirectional and bidirectional streams, as well as send datagrams, for a session that it has sent the CONNECT request for, even if it has not yet received the server's response to the request. On the server side, opening streams and sending datagrams is possible as soon as the CONNECT request has been received.

If at any point a session ID is received that cannot a valid ID for a client-initiated bidirectional stream, the recipient MUST close the connection with an H3_ID_ERROR error code.

4.1. Unidirectional streams

WebTransport endpoints can initiate unidirectional streams. The HTTP/3 unidirectional stream type SHALL be 0x54. The body of the stream SHALL be the stream type, followed by the session ID, encoded as a variable-length integer, followed by the user-specified stream data (Figure 1).

```
Unidirectional Stream {
    Stream Type (i) = 0x54,
    Session ID (i),
    Stream Body (..)
}
```

Figure 1: Unidirectional WebTransport stream format

4.2. Bidirectional Streams

All client-initiated bidirectional streams are reserved by HTTP/3 as request streams, which are a sequence of HTTP/3 frames with a variety of rules (see Sections 4.1 and 6.1 of [HTTP3]).

WebTransport extends HTTP/3 to allow clients to declare and use alternative request stream rules. Once a client receives settings indicating WebTransport support (Section 3.1), it can send a special signal value, encoded as a variable-length integer, as the first bytes of the stream in order to indicate how the remaining bytes on the stream are used.

WebTransport extends HTTP/3 by defining rules for all server-initiated bidirectional streams. Once a server receives an incoming CONNECT request establishing a WebTransport session (Section 3.1), it can open a bidirectional stream for use with that session and SHALL send a special signal value, encoded as a variable-length integer, as the first bytes of the stream in order to indicate how the remaining bytes on the stream are used.

The signal value, 0x41, is used by clients and servers to open a bidirectional WebTransport stream. Following this is the associated session ID, encoded as a variable-length integer; the rest of the stream is the application payload of the WebTransport stream (Figure 2).

```
Bidirectional Stream {
    Signal Value (i) = 0x41,
    Session ID (i),
    Stream Body (..)
}
```

Figure 2: Bidirectional WebTransport stream format

This document reserves the special signal value 0x41 as a WEBTRANSPORT_STREAM frame type. While it is registered as an HTTP/3 frame type to avoid collisions, WEBTRANSPORT_STREAM is not a proper HTTP/3 frame, as it lacks length; it is an extension of HTTP/3 frame syntax that MUST be supported by any peer negotiating WebTransport. Endpoints that implement this extension are also subject to additional frame handling requirements. Endpoints MUST NOT send WEBTRANSPORT_STREAM as a frame type on HTTP/3 streams other than the very first bytes of a request stream. Receiving this frame type in any other circumstances MUST be treated as a connection error of type H3_FRAME_ERROR.

4.3. Resetting Data Streams

A WebTransport endpoint may send a RESET_STREAM or a STOP_SENDING frame for a WebTransport data stream. Those signals are propagated by the WebTransport implementation to the application.

A WebTransport application SHALL provide an error code for those operations. Since WebTransport shares the error code space with HTTP/3, WebTransport application errors for streams are limited to an unsigned 32-bit integer, assuming values between 0x00000000 and 0xffffffff. WebTransport implementations SHALL remap those error codes into the error range reserved for WEBTRANSPORT_APPLICATION_ERROR, where 0x00000000 corresponds to 0x52e4a40fa8db, and 0xffffffff corresponds to 0x52e5ac983162. Note that there are code points inside that range of form "0x1f * N + 0x21" that are reserved by Section 8.1 of [HTTP3]; those have to be skipped when mapping the error codes (i.e. the two HTTP/3 error codepoints adjacent to a reserved codepoint would map to two adjacent WebTransport application error codepoints). An example pseudocode can be seen in Figure 3.

```
first = 0x52e4a40fa8db
last = 0x52e5ac983162

def webtransport_code_to_http_code(n):
    return first + n + floor(n / 0x1e)

def http_code_to_webtransport_code(h):
    assert(first <= h <= last)
    assert((h - 0x21) % 0x1f != 0)
    shifted = h - first
    return shifted - floor(shifted / 0x1f)
```

Figure 3: Pseudocode for converting between WebTransport application errors and HTTP/3 error codes

WebTransport data streams are associated with sessions through a header at the beginning of the stream; resetting a stream may result in that data being discarded. Because of that, WebTransport application error codes are best effort, as the WebTransport stack is not always capable of associating the reset code with a session. The only exception is the situation where there is only one session on a given HTTP/3 connection, and no intermediaries between the client and the server.

WebTransport implementations SHALL forward the error code for a stream associated with a known session to the application that owns that session; similarly, the intermediaries SHALL reset the streams with corresponding error code when receiving a reset from the peer. If a WebTransport implementation intentionally allows only one session over a given HTTP/3 connection, it SHALL forward the error codes within WebTransport application error code range to the application that owns the only session on that connection.

4.4. Datagrams

Datagrams can be sent using HTTP Datagrams. The WebTransport datagram payload is sent unmodified in the "HTTP Datagram Payload" field of an HTTP Datagram (Section 2.1 of [HTTP-DATAGRAM]). Note that the payload field directly follows the Quarter Stream ID field, which is at the start of the QUIC DATAGRAM frame payload and refers to the CONNECT stream that established the WebTransport session.

4.5. Buffering Incoming Streams and Datagrams

In WebTransport over HTTP/3, the client MUST wait for receipt of the server's SETTINGS frame before establishing any WebTransport sessions by sending CONNECT requests using the WebTransport upgrade token (see Section 3.1). This ensures that the client will always know what versions of WebTransport can be used on a given HTTP/3 connection.

Clients can, however, send a SETTINGS frame, multiple WebTransport CONNECT requests, WebTransport data streams, and WebTransport datagrams all within a single flight. As those can arrive out of order, a WebTransport server could be put into a situation where it receives a stream or a datagram without a corresponding session. Similarly, a client may receive a server-initiated stream or a datagram before receiving the CONNECT response headers from the server.

To handle this case, WebTransport endpoints SHOULD buffer streams and datagrams until those can be associated with an established session. To avoid resource exhaustion, the endpoints MUST limit the number of buffered streams and datagrams. When the number of buffered streams is exceeded, a stream SHALL be closed by sending a RESET_STREAM and/or STOP_SENDING with the WEBTRANSPORT_BUFFERED_STREAM_REJECTED error code. When the number of buffered datagrams is exceeded, a datagram SHALL be dropped. It is up to an implementation to choose what stream or datagram to discard.

4.6. Interaction with HTTP/3 GOAWAY frame

HTTP/3 defines a graceful shutdown mechanism (Section 5.2 of [HTTP3]) that allows a peer to send a GOAWAY frame indicating that it will no longer accept any new incoming requests or pushes.

A client receiving GOAWAY cannot initiate CONNECT requests for new WebTransport sessions if the stream identifier is equal to or greater than the indicated stream ID.

An HTTP/3 GOAWAY frame is also a signal to applications to initiate shutdown for all WebTransport sessions. To shut down a single WebTransport session, either endpoint can send a DRAIN_WEBTRANSPORT_SESSION (0x78ae) capsule.

```
DRAIN_WEBTRANSPORT_SESSION Capsule {
  Type (i) = DRAIN_WEBTRANSPORT_SESSION,
  Length (i) = 0
}
```

After sending or receiving either a DRAIN_WEBTRANSPORT_SESSION capsule or a HTTP/3 GOAWAY frame, an endpoint MAY continue using the session and MAY open new streams. The signal is intended for the application using WebTransport, which is expected to attempt to gracefully terminate the session as soon as possible.

5. Session Termination

A WebTransport session over HTTP/3 is considered terminated when either of the following conditions is met:

- the CONNECT stream is closed, either cleanly or abruptly, on either side; or

- a CLOSE_WEBTRANSPORT_SESSION capsule is either sent or received.

Upon learning that the session has been terminated, the endpoint MUST reset the send side and abort reading on the receive side of all of the streams associated with the session (see Section 2.4 of [RFC9000]) using the WEBTRANSPORT_SESSION_GONE error code; it MUST NOT send any new datagrams or open any new streams.

To terminate a session with a detailed error message, an application MAY send an HTTP capsule [HTTP-DATAGRAM] of type CLOSE_WEBTRANSPORT_SESSION (0x2843). The format of the capsule SHALL be as follows:

```
CLOSE_WEBTRANSPORT_SESSION Capsule {
  Type (i) = CLOSE_WEBTRANSPORT_SESSION,
  Length (i),
  Application Error Code (32),
  Application Error Message (..8192),
}
```

CLOSE_WEBTRANSPORT_SESSION has the following fields:

Application Error Code: A 32-bit error code provided by the application closing the connection.

Application Error Message: A UTF-8 encoded error message string provided by the application closing the connection. The message takes up the remainder of the capsule, and its length MUST NOT exceed 1024 bytes.

An endpoint that sends a CLOSE_WEBTRANSPORT_SESSION capsule MUST immediately send a FIN. The endpoint MAY send a STOP_SENDING to indicate it is no longer reading from the CONNECT stream. The recipient MUST close the stream upon receiving a FIN. If any additional stream data is received on the CONNECT stream after receiving a CLOSE_WEBTRANSPORT_SESSION capsule, the stream MUST be reset with code H3_MESSAGE_ERROR.

Cleanly terminating a CONNECT stream without a CLOSE_WEBTRANSPORT_SESSION capsule SHALL be semantically equivalent to terminating it with a CLOSE_WEBTRANSPORT_SESSION capsule that has an error code of 0 and an empty error string.

In some scenarios, an endpoint might want to send a CLOSE_WEBTRANSPORT_SESSION with detailed close information and then immediately close the underlying QUIC connection. If the endpoint were to do both of those simultaneously, the peer could potentially receive the CONNECTION_CLOSE before receiving the CLOSE_WEBTRANSPORT_SESSION, thus never receiving the application error data contained in the latter. To avoid this, the endpoint

SHOULD wait until all of the data on the CONNECT stream is acknowledged before sending the CONNECTION_CLOSE; this gives CLOSE_WEBTRANSPORT_SESSION properties similar to that of the QUIC CONNECTION_CLOSE mechanism as a best-effort mechanism of delivering application close metadata.

6. Considerations for Future Versions

Future versions of WebTransport that change the syntax of the CONNECT requests used to establish WebTransport sessions will need to modify the upgrade token used to identify WebTransport, allowing servers to offer multiple versions simultaneously (see Section 8.1).

Servers that support future incompatible versions of WebTransport signal that support by changing the codepoint used for the SETTINGS_WEBTRANSPORT_MAX_SESSIONS parameter (see Section 8.2). Clients can select the associated upgrade token, if applicable, to use when establishing a new session, ensuring that servers will always know the syntax in use for every incoming request.

Changes to future stream formats require changes to the Unidirectional Stream type (see Section 4.1) and Bidirectional Stream signal value (see Section 4.2) to allow recipients of incoming frames to determine the WebTransport version, and corresponding wire format, used for the session associated with that stream.

6.1. Negotiating the Draft Version

[[RFC editor: please remove this section before publication.]]

The wire format aspects of the protocol are negotiated by changing the codepoint used for the SETTINGS_WEBTRANSPORT_MAX_SESSIONS parameter. Because of that, any WebTransport endpoint MUST wait for the peer's SETTINGS frame before sending or processing any WebTransport traffic. When multiple versions are supported by both of the peers, the most recent version supported by both is selected.

7. Security Considerations

WebTransport over HTTP/3 satisfies all of the security requirements imposed by [OVERVIEW] on WebTransport protocols, thus providing a secure framework for client-server communication in cases when the client is potentially untrusted.

WebTransport over HTTP/3 requires explicit opt-in through the use of an HTTP/3 setting; this avoids potential protocol confusion attacks by ensuring the HTTP/3 server explicitly supports it. It also requires the use of the Origin header, providing the server with the ability to deny access to Web-based clients that do not originate from a trusted origin.

Just like HTTP traffic going over HTTP/3, WebTransport pools traffic to different origins within a single connection. Different origins imply different trust domains, meaning that the implementations have to treat each transport as potentially hostile towards others on the same connection. One potential attack is a resource exhaustion attack: since all of the transports share both congestion control and flow control context, a single client aggressively using up those resources can cause other transports to stall. The user agent thus SHOULD implement a fairness scheme that ensures that each transport within connection gets a reasonable share of controlled resources; this applies both to sending data and to opening new streams.

A client could attempt to exhaust resources by opening too many WebTransport sessions at once. In cases when the client is untrusted, the user agent SHOULD limit the number of outgoing sessions the client can open.

8. IANA Considerations

8.1. Upgrade Token Registration

The following entry is added to the "Hypertext Transfer Protocol (HTTP) Upgrade Token Registry" registry established by Section 16.7 of [HTTP].

The "webtransport" label identifies HTTP/3 used as a protocol for WebTransport:

Value: webtransport

Description: WebTransport over HTTP/3

Reference: This document and [I-D.ietf-webtrans-http2]

8.2. HTTP/3 SETTINGS Parameter Registration

The following entry is added to the "HTTP/3 Settings" registry established by [HTTP3]:

The SETTINGS_WEBTRANSPORT_MAX_SESSIONS parameter indicates that the specified HTTP/3 endpoint is WebTransport-capable and the number of concurrent sessions it is willing to receive. The default value for the SETTINGS_WEBTRANSPORT_MAX_SESSIONS parameter is "0", meaning that the endpoint is not willing to receive any WebTransport sessions.

Setting Name: WEBTRANSPORT_MAX_SESSIONS

Value: 0xc671706a

Default: 0

Specification: This document

8.3. Frame Type Registration

The following entry is added to the "HTTP/3 Frame Type" registry established by [HTTP3]:

The WEBTRANSPORT_STREAM frame is reserved for the purpose of avoiding collision with WebTransport HTTP/3 extensions:

Code: 0x41

Frame Type: WEBTRANSPORT_STREAM

Specification: This document

8.4. Stream Type Registration

The following entry is added to the "HTTP/3 Stream Type" registry established by [HTTP3]:

The "WebTransport stream" type allows unidirectional streams to be used by WebTransport:

Code: 0x54

Stream Type: WebTransport stream

Specification: This document

Sender: Both

8.5. HTTP/3 Error Code Registration

The following entry is added to the "HTTP/3 Error Code" registry established by [HTTP3]:

Name: WEBTRANSPORT_BUFFERED_STREAM_REJECTED

Value: 0x3994bd84

Description: WebTransport data stream rejected due to lack of associated session.

Specification: This document.

Name: WEBTRANSPORT_SESSION_GONE

Value: 0x170d7b68

Description: WebTransport data stream aborted because the associated WebTransport session has been closed.

Specification: This document.

In addition, the following range of entries is registered:

Name: WEBTRANSPORT_APPLICATION_ERROR

Value: 0x52e4a40fa8db to 0x52e5ac983162 inclusive, with the exception of the codepoints of form 0x1f * N + 0x21.

Description: WebTransport application error codes.

Specification: This document.

8.6. Capsule Types

The following entries are added to the "HTTP Capsule Types" registry established by [HTTP-DATAGRAM]:

The CLOSE_WEBTRANSPORT_SESSION capsule.

Value: 0x2843
Capsule Type: CLOSE_WEBTRANSPORT_SESSION
Status: permanent
Specification: This document
Change Controller: IETF
Contact: WebTransport Working Group webtransport@ietf.org (mailto:webtransport@ietf.org)
Notes: None

The DRAIN_WEBTRANSPORT_SESSION capsule.

Value: 0x78ae

Capsule Type: DRAIN_WEBTRANSPORT_SESSION
Status: provisional (when this document is approved this will become permanent)
Specification: This document
Change Controller: IETF
Contact: WebTransport Working Group webtransport@ietf.org (mailto:webtransport@ietf.org)
Notes: None

9. References

9.1. Normative References

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[HTTP-DATAGRAM] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", RFC 9297, DOI 10.17487/RFC9297, August 2022, <https://www.rfc-editor.org/rfc/rfc9297>.

[HTTP3] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

[OVERVIEW] Vasiliev, V., "The WebTransport Protocol Framework", Work in Progress, Internet-Draft, draft-ietf-webtrans-overview-07, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-overview-07>.

[QUIC-DATAGRAM] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/rfc/rfc9221>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/rfc/rfc6454>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/rfc/rfc6585>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8441] McManus, P., "Bootstrapping WebSockets with HTTP/2", RFC 8441, DOI 10.17487/RFC8441, September 2018, <https://www.rfc-editor.org/rfc/rfc8441>.

[RFC8941] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/rfc/rfc8941>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9218] Oku, K. and L. Pardue, "Extensible Prioritization Scheme for HTTP", RFC 9218, DOI 10.17487/RFC9218, June 2022, <https://www.rfc-editor.org/rfc/rfc9218>.

[RFC9220] Hamilton, R., "Bootstrapping WebSockets with HTTP/3", RFC 9220, DOI 10.17487/RFC9220, June 2022, <https://www.rfc-editor.org/rfc/rfc9220>.

9.2. Informative References

[I-D.ietf-webtrans-http2]
Frindell, A., Kinnear, E., Pauly, T., Thomson, M., Vasiliev, V., and G. Xie, "WebTransport over HTTP/2", Work in Progress, Internet-Draft, draft-ietf-webtrans-http2-08, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-http2-08>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/rfc/rfc7301>.

Appendix A. Changelog

A.1. Changes between draft versions 02 and 07

The following changes make the draft-02 and draft-07 versions of this protocol incompatible:

* draft-07 requires SETTINGS_WEBTRANSPORT_MAX_SESSIONS (#86) and uses it for version negotiation (#129)

* draft-07 explicitly requires SETTINGS_ENABLE_CONNECT_PROTOCOL to be enabled (#93)

* draft-07 explicitly requires SETTINGS_H3_DATAGRAM to be enabled (#106)

* draft-07 only allows WEBTRANSPORT_STREAM at the beginning of the stream

The following changes that are present in draft-07 can be also implemented by a draft-02 implementation safely:

* Expanding stream reset error code space from 8 to 32 bits (#115)

* WEBTRANSPORT_SESSION_GONE error code (#75)

* Handling for HTTP GOAWAY (#76)

* DRAIN_WEBTRANSPORT_SESSION capsule (#79)

* Disallowing following redirects automatically (#113)

Authors' Addresses

Alan Frindell
Facebook
Email: afrind@fb.com

Eric Kinnear
Apple Inc.
Email: ekinnear@apple.com

Victor Vasiliev
Google
Email: vasilvv@google.com

WebTransport M. Thomson
Internet-Draft Mozilla
Intended status: Informational E. Kinnear
Expires: 13 April 2024 Apple Inc.
11 October 2023

Applying Per-Session Limits for WebTransport
draft-thomson-webtrans-session-limit-01

Abstract

Limits to how a WebTransport session uses QUIC resources like streams or data can help reduce the effect that one WebTransport session has on other uses of the same HTTP/3 connection. This describes mechanisms for limiting the number of streams and quantity of data that can be consumed by each WebTransport session.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://martinthomson.github.io/wt-session-limits/draft-thomson-webtrans-session-limit.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-thomson-webtrans-session-limit/.

Discussion of this document takes place on the WebTransport Working Group mailing list (mailto:webtransport@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/webtransport/. Subscribe at https://www.ietf.org/mailman/listinfo/webtransport/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/wt-session-limits.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

WebTransport in HTTP/3 [WTH3] provides applications with all the functionality of QUIC [QUIC] streams. In the case where a single connection includes a WebTransport session that needs to coexist with other WebTransport sessions or HTTP requests, the core draft does not offer any way to place limits on stream usage.

This document describes an additional layer of session-level flow control that governs the creation of streams and sets a session-level limit on the amount of data that can be exchanged in a session.

This document does not define a framework for prioritizing the streams created for a WebTransport session with other streams.

Note that this document is intended as input for [WTH3]. Although it is possible to define this as an extension to that protocol, integration of this design is simpler; see Section 4.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Protocol Definition

This document uses the following flow control capsules defined in [WTH2]:

* WT_MAX_DATA (Section 5.5 of [WTH2])

* WT_MAX_STREAMS (Section 5.7 of [WTH2])

* WT_DATA_BLOCKED (Section 5.8 of [WTH2])

* WT_STREAMS_BLOCKED (Section 5.10 of [WTH2])

These capsules are unchanged, except that where the WebTransport over HTTP/2 capsules refer to streams that flow over the HTTP/2 stream containing the entire WebTransport session, these capsules refer to separate limits as described in subsequent sections.

These capsules use the codepoints allocated in [WTH2].

3.1. Stream Limits

The WT_MAX_STREAMS capsule establishes a limit on the number of streams within a WebTransport session. Like the QUIC MAX_STREAMS frame (Section 19.11 of [QUIC]), this capsule has two types that provide separate limits for unidirectional and bidirectional streams that are initiated by a peer.

The session-level stream limit applies in addition to the QUIC MAX_STREAMS frame, which provides a connection-level stream limit. New streams can only be created within the session if both the stream- and the connection-level limit permit; see Section 4.6 of [QUIC] for details on how QUIC stream limits are applied.

Unlike the WT_MAX_STREAMS capsule or the QUIC MAX_STREAMS frame, there is no simple relationship between the value in this frame and stream IDs in QUIC STREAM frames. This especially applies if there are other users of streams on the connection.

The WT_STREAMS_BLOCKED capsule is sent to indicate that an endpoint was unable to create a stream due to the session-level stream limit.

3.2. Data Limits

The WT_MAX_DATA capsule establishes a limit on the amount of data that can be sent within a WebTransport session. This limit counts all data that is sent on streams of the corresponding type, excluding the stream header (see Sections 4.2 and 4.2 of [WTH3]). The stream header is excluded from this limit so that this limit does not prevent the sending of information that is essential in linking new streams to a specific WebTransport session.

Implementing WT_MAX_DATA requires that the QUIC stack provide the WebTransport implementation with information about the final size of streams; see Section 4.5 of [QUIC].

The WT_DATA_BLOCKED capsule is sent to indicate that an endpoint was unable to send data due to a limit set by the WT_MAX_DATA capsule.

Because WebTransport over HTTP/3 uses a native QUIC stream for each WebTransport stream, per-stream data limits are provided by QUIC natively. The WT_MAX_STREAM_DATA and WT_STREAM_DATA_BLOCKED capsules are not used and so are prohibited. Endpoints MUST treat receipt of a WT_MAX_STREAM_DATA or a WT_STREAM_DATA_BLOCKED capsule as a session error.

4. Negotiation

If the use of flow control capsules are merged into the main specification [WTH3], their use will be negotiated along with the use of WebTransport over HTTP/3. This is the simplest approach.

Alternatively, if this remains as an optional extension, new HTTP/3 settings will be needed to negotiate the use of these features. In the abstract, we could define settings that carry initial values for the three variables that are controlled by the session-level flow control capsules defined here. The presence of any of those settings would indicate that these limits will be respected if the capsule is sent.

Both peers need to indicate the setting before these capsules apply. If only one peer advertises any of these settings, that might indicate that they are willing to receive and respect session-level flow control capsules. However, such an endpoint cannot know when to start applying the limit.

5. Security Considerations

Aside from new exposure to the usual programming errors arising from increased protocol complexity, it is believed that the introduction of these capabilities only improves security as it provides better control over endpoint resource allocation.

6. IANA Considerations

This document has no IANA actions.

7. Normative References

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[WTH2] Frindell, A., Kinnear, E., Pauly, T., Thomson, M., Vasiliev, V., and G. Xie, "WebTransport over HTTP/2", Work in Progress, Internet-Draft, draft-ietf-webtrans-http2-06, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-http2-06>.

[WTH3] Frindell, A., Kinnear, E., and V. Vasiliev, "WebTransport over HTTP/3", Work in Progress, Internet-Draft, draft-ietf-webtrans-http3-07, 13 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-http3-07>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Martin Thomson
Mozilla
Email: mt@lowentropy.net

Eric Kinnear
Apple Inc.
Email: ekinnear@apple.com

QUIC Working Group
Internet-Draft
Intended status: Informational
Expires: 24 August 2024

V. Vasiliev
Google
21 February 2024

Stream Namespaces for QUIC
draft-vvv-quic-namespaces-00

Abstract

QUIC Stream Namespaces provide an extension to the QUIC protocol that enables multiplexing multiple logical groups of streams within the same connection, while providing flow control isolation.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-vvv-quic-namespaces/.

Discussion of this document takes place on the quic Working Group mailing list (mailto:quic@ietf.org), which is archived at https://example.com/WG. Subscribe at https://www.ietf.org/mailman/listinfo/quic/.

Source for this draft and an issue tracker can be found at https://github.com/https://github.com/vasilvv/draft-vvv-quic-namespaces.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

QUIC [RFC9000] provides an ordered bytestream abstraction called streams. Streams are subject to various flow control mechanisms that allow a network endpoint to control how much resources a peer is allowed to consume. Some of the flow control mechanisms are scoped to a single stream; others are global to the entire connection. The connection-level flow control mechanisms are a good fit in cases when all of the streams originate from the same entity; however, in cases when multiple logical entities share the same connection, a single global limit may lead to one entity starving another. This document provides a mechanism by which a single QUIC connection can have multiple namespaces, each with its own resource limits for streams.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Namespaces

A QUIC namespace is a 62-bit unique ID number. In the initial state, every namespace ID is assumed to exist, but have a MAX_STREAMS number associated with it set to 0 for all types of streams, and a MAX_DATA value of 0 in both directions. A peer opens a namespace by sending a combination of MAX_DATA and MAX_STREAMS frames for that namespace. The recepient may response with either its own MAX_DATA and MAX_STREAMS, confirming the response, or it may close the namespace. Frames that do not have a namespace ID associated with them are said to be a part of _the default namespace_.

Note that there is no way to set a namespace-specific initial_max_stream_data parameters; those remain connection-global.

4. Frames

4.1. NS frame

An NS frame (frame type=0x29c5) is a frame that alters the meaning of the frame that comes immediately after it. If the subsequent frame has a stream ID in it, that ID refers to the stream with the corresponding ID in the specified namespace. If the subsequent frame alters connection-global flow control limits, those limits are altered for the namespace in question, instead of the default namespace.

```
NS Frame {
  Type (i) = 0x29c5,
  Namespace ID (i),
}
```

Figure 1: NS Frame Format

The following frames are allowed to follow the NS frame: STREAM, RESET_STREAM, STOP_SENDING, MAX_DATA, MAX_STREAM_DATA, MAX_STREAMS, DATA_BLOCKED, STREAM_DATA_BLOCKED, STREAMS_BLOCKED. Extensions that define their own frames can define their own semantics of interacting with namespaces. If a frame that is not listed above and does not have extension semantics defined for it is prefixed with an NS frame, the recepient MUST close the connection with a PROTOCOL_VIOLATION error code. Same applies to an NS frame that is not followed by anything.

Note that this intentionally does not define NS prefix for the DATAGRAM frames [RFC9221], as datagrams already have pre-defined mechanisms for multiplexing (such as [RFC9297]) that may conflict with QUIC stream namespaces, and there is no technical advantage of using an NS frame with datagrams over doing multiplexing within the datagram payload.

4.2. CLOSE_NAMESPACE frame

A CLOSE_NAMESPACE frame indicates to the peer that the sender will not process any further data received for a given namespace. The sender can discard all of the state related to the namespace after sending this frame.

```
CLOSE_NAMESPACE Frame {
  Type (i) = 0x29c6,
  Namespace ID (i),
}
```

Figure 2: CLOSE_NAMESPACE Frame Format

5. Security Considerations

TODO Security

TODO: discuss the issue where the peer has to remember flow control limits for arbitrary unexpected namespaces.

6. IANA Considerations

TODO: add a transport parameter to negotiate this feature.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9221] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/rfc/rfc9221>.

7.2. Informative References

[RFC9297] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", RFC 9297, DOI 10.17487/RFC9297, August 2022, <https://www.rfc-editor.org/rfc/rfc9297>.

Acknowledgments

TODO acknowledge.

Author's Address

Victor Vasiliev
Google
Email: vasilvv@google.com